Test E. R.:
Our Goal to Triage a Million Tests A Day

Wilson Snyder
<wsnyder@wsnyder.org>
Bryce Denney and Robert Woods-Corwin

October 11, 2007

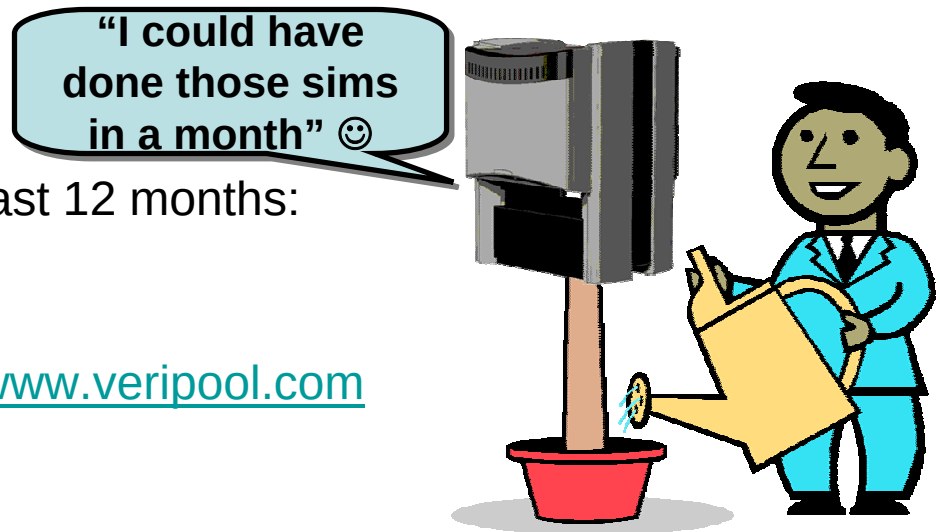
Agenda

- What have we grown?
- Test drivers: From Vtest to BugVise
- What do tests look like?
- What tests to run?
- How to triage?
- Conclusions
- Q&A

What Have We Grown?

- John Mucci described our system - (Thanks ☺)

- What did it take to verify this?
  - 1.2 million lines of RTL -> 198 million transistors
  - 32,000 parameterized tests
    - SystemC, Verilog, and Perl
    - <20% used a license
  - 22,000,000 test runs over the last 12 months:
    - 230 compute years
    - 2.1 hours of “real chip” time
  - More in a DAC paper at http://www.veripool.com



- What ran these? The driver system, which is this talk…

Vtest: Test system of yore

- Vtest is our test driver system
  - Evolving since 1999 across 4 companies
  - Launch any test run with a single line command
  - Picks random tests to run, and random seeds
  - Runs the tests, perhaps in parallel across sim/board farm
  - Reserves licenses and other complicated resources
- Used for verification, software test, and real hardware tests
  - Past projects: power on box, load up /root, program routers, start traffic generators, extract internal logic analyzer state, download HP logic analyzer, etc..
- Summarizes results of runs and adds to database

BugVise

- After 8 years, Vtest is due for a rewrite.
- Scale to a million tests a day.
  - Use what's convenient to us, which is → With 6,000 CPUs at 10 CPU minutes per test it yields 1M tests/day!

- Rewrite the code with long-term in mind:
  - Separate the company and project specifics
  - Core modules and plug-ins, and clean API
  - Make it friendly to SQA folks
  - Open Source
- We'll now look at where we did well in the past, and what we'd like to do…

What's in a test name?

- Vtest test names consist of 4 parts:

Unit Name - How to build the simulation executable.

Test Name - What test to run. Tests run "on" units.

Var Name - Controls an aspect of the test or build.

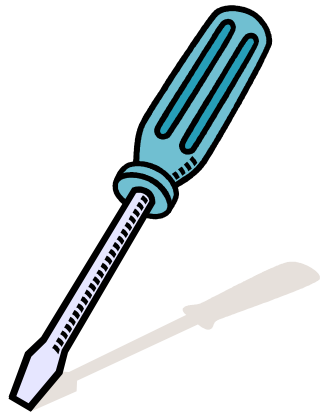
Var Value - What to set the Var to.

```
cpu_beh/cpu_rand,loops=1000
```

- BugVise makes everything a test – No units versus tests.

How to Parameterize Tests?

- Vtest variables configure tests:
  - Not declared, just reference in Perl, Verilog, C or assembler
    - This makes it easy to use them
  - Program to cross reference their use
    - But with 2,100 of them, they're hard to browse!



- BugVise variables:
  - Explicitly declared in tests
  - Typed
    - "Durations" for example, so can specify times in specific units
  - Error checking for misspellings
  - Given a test, get list of all variables possible to set
  - Web interface to search all tests for use of a variable

How to list tests?

- Vtest Test Names get long, so can be shortened
  - Looks like a "Makefile"

```
cpu_rtl/cpu_random,ops=1000:     cpu_random_ops
```

  - Thus a "vtest cpu_random_ops" is the same as typing the long name.
  - You can build on existing tests

```
cpu_random_ops,rdPct=90:     cpu_rtl_basic  cpu_a
cpu_random_ops,rdPct=10:     cpu_rtl_basic  cpu_b
```

  - You can have a single target run multiple tests:

```
cpu_a cpu_b: cpu_both
```

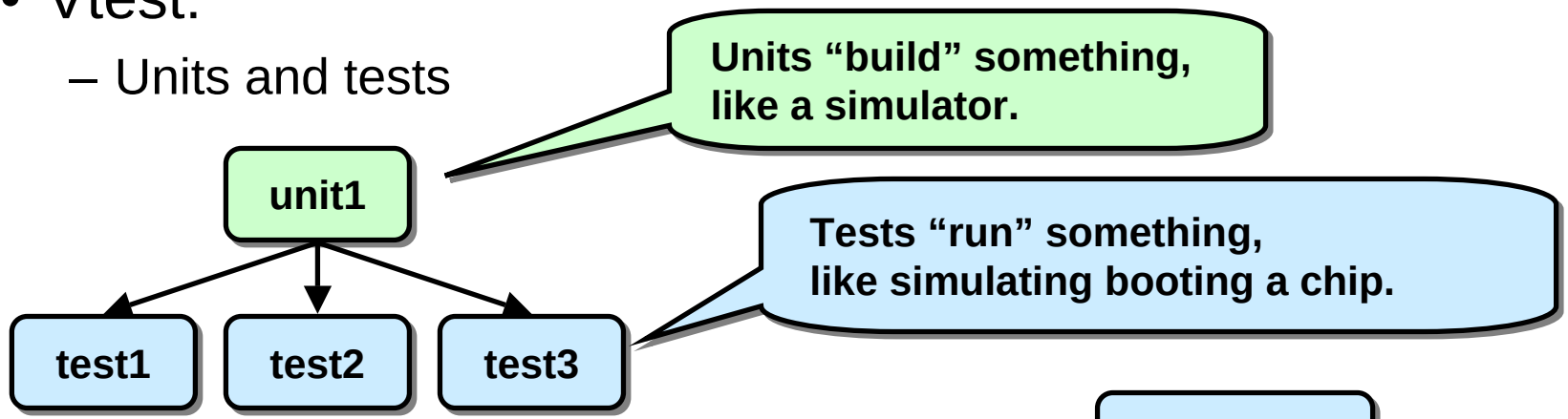
  - You can build on lists of tests (run 100 runs of both tests)

```
cpu_both,runs=100:           rack_both_runs
```

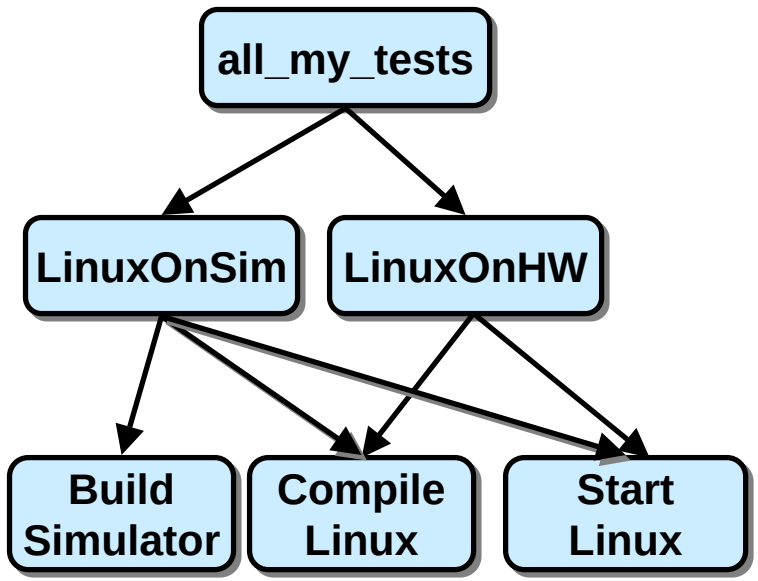
- BugVise also allows lists made by programs or database queries.

How to specify prerequisites?

- Vtest:
  - Units and tests



- BugVise:
  - Everything is a test
  - Tests can depend on other tests
  - Thus a graph of tests with “prerequisites”



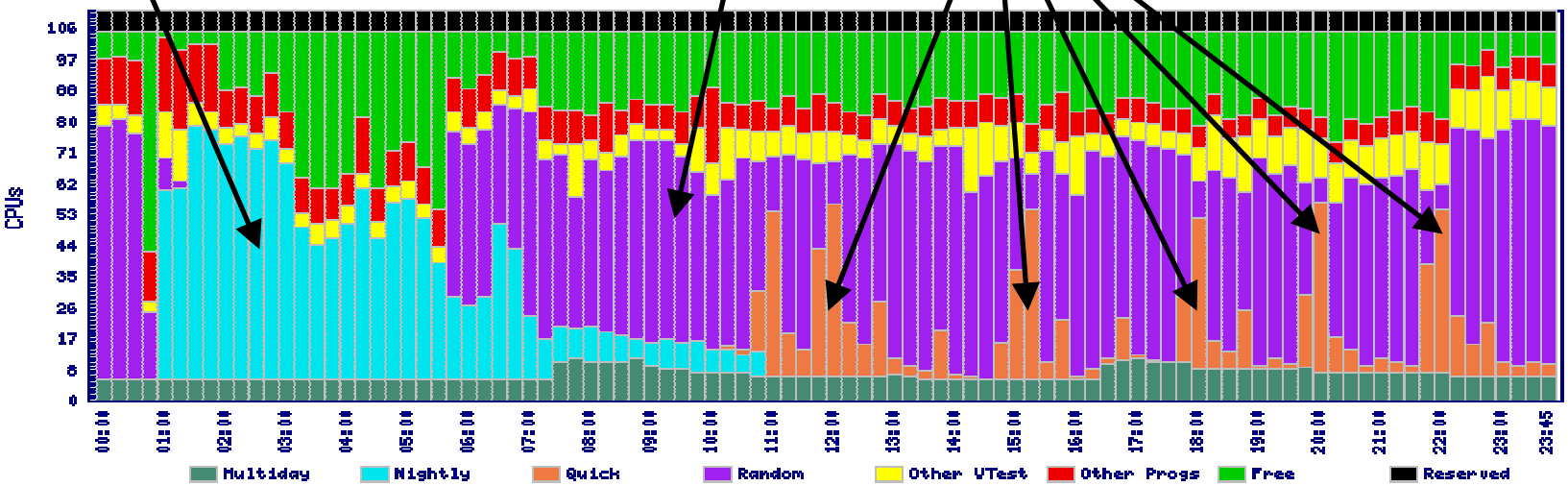
Where to allocate computes?

- Vtest Web view from one day of testing

Random:
Using last “quick” revision, Search for failing seeds.

Nightly:
Using last “quick” revision, run focused tests and check haven’t lost ground.

Quick:
If any files were committed, check sanity of mainline.



What tests to run?

- Vtest:
  - Manually made categories using test lists
  - Weighted-random selection of N tests out of all listed tests
    - Provides easy knob for changing use of CPU time;
      Just changing ~run_chance=50 to 25 reduces runtime by half.
    - Tests that didn't run recently get higher priority
- BugVise:
  - In general, Vtest's approach worked well
  - Analyze how often X fails, and use to weight X's run chance
  - Correlate tests that fail together (X,Y), and if X fails, run more Y
  - Automatically use coverage data
    - Feed coverage dispersion results into run chance of random tests
    - Score tests using mutation analysis on the RTL?

Recording/Finding test results?

- Vtest tracks all test runs in a web database:
  - Did this test ever work, and when?
  - What revisions did the test pass/fail on?
  - 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 for bug2222
r5331	2 pass 2 fail w/mod	wsnyder	Doing something bad
r5330			
r5329	1 pass	pholmes	New incredible MPI fabric test added

Recording/Finding test results?

- BugVise:
  - Search by test name, error message,etc
  - Make associations between tests visually obvious

+/-	TestId	Name	Pass	Fail	Message
⊞	t12291	QuickTests	63	0	pass
⊟	t12344	NightlyTests	21321	22	mixed
⊟	t12345	FaultTest	4	1	mixed
	t12370	FaultTest::1	0	1	Assert failed
	t12371	FaultTest::2	1	0	pass
	t12372	FaultTest::3	1	0	pass
	t12373	FaultTest::4	1	0	pass

A Test Failed, Now What?

- Vtest Automated Triaging:
  - Known-good test fails
    - Determine revision range that broke test and email authors

To: wsnyder
From: regress
Subject: Vgripe: 1 new quick failure

1) pci/FredTest
%Error: Hang on PCI bus
Failing 1 day.
Passes in r5329. Fails in r5333.

Relevant commits:
r5331 | wsnyder | Tuesday 11:22:11
Doing something bad

  - Random test fails
    - We manually pre-marked each random test with interested parties
    - Email them the failure
  - Nightly test fails
    - Email whole distribution list and let humans sort it out
    - Bug tracking can set regexps to tag failures, but not used much.

How'd it work?

- Was Vtest Triaging successful?
  - Gripe emails are extremely effective
  - Random failure emails are OK, but
    - Sometimes to wrong person
    - "Lag" in reporting problem that was already fixed.
  - Nightly failures required a lot of work
    - Some issues ignored day after day
    - Hard to remember what we've fixed already
    - Daily meetings when approaching signoff
  - Need combination of approaches
  - Reducing human triage time must be the main goal!

Over the last year, on average 0.6% of test runs failed. At 1M tests/day, that's 6,000 failures/day!

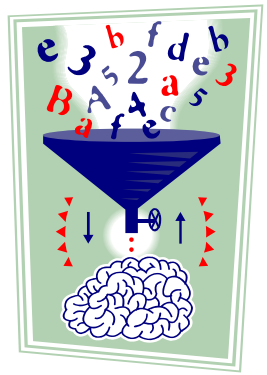
What do users want?

- What do people want to know?
  - What failures am I on the hook for?
  - Has someone fixed this yet?
  - What failures are new? New failing seeds? Etc?
  - How do I rerun all failures like it?

- Other features
  - Group like failures together
  - Be able to reassign failures to other people
  - Allow easy promotion to bugs
    - Paste in rerun information, errors, revision, etc.

How to Auto-Triage?

- How do we categorize failures?



1. If same test fails like last time
   - Attach it to earlier failures.
2. Rerun on a different server
   - If passes, it's a flaky server. Assign to system triager.
3. Rerun with fixed seed
   - If passes, it's a random failure. Assign to "watcher" for that test.
4. Rerun to isolate exact revision range that fails
   - If fails, assign to person(s) committing the change.
5. Group related to matching regexps in open bugs.
6. Group by similar error message

How might triaging look?

Test Results:

Change Selected: [Change Triage To: ddrRand]

TestId	Name	Age	Sel	Triage	Test Output Message
t12370	FaultTest::1	13 h	☐	srvDead	Disk failed
t12311	DdrTest::6	9 d	☑	ddrRand	Reg not implemented

Triage Groups:

File a real bug, and track it.

Change Selected: [Take] [Bugify] [Email] [Close]

Name	Age	Tests	Status	Sel	Owner	Triage Comment
srvDead	13 h	22 fail	New	☑	wsnyder	Autotriaged: server failed, rerun ok
ddrRand	9 d	1 fail	Taken	☐	denney	Need to fix the spec

Conclusion

BugVise wants your help!

- Open source tools are built up over years of suggestions.
  - Your suggestions are needed, and may help everyone else out!

- Ideas
  - What makes a good test driver system?
  - What have you tried?

- Resources
  - Would you like to use BugVise?
  - Are you interested in Co-Developing BugVise with us?

Conclusions

- Vtest well handled 50k tests/day
- Great for verification, ok for SQA tesing
- BugVise should let us scale to 1,000,000
- Smarter selection of tests is good, but…
- Better triaging is critical
- You can help – and use it!

Public Tool Sources

- This presentation and the open source tools we used are available at http://www.veripool.com
  - BugVise – Coming soon!
  - 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, etc
  - Vregs – Extract register and class declarations from documentation