CovVise:
How We Stopped Throwing Away Interesting Coverage Data

http://www.veripool.org/papers/CovVise_SNUGBos09_pres.pdf

Wilson Snyder
Cavium Networks
wsnyder@wsnyder.org

Robert Woods-Corwin
NVIDIA
covvise@rwoodsco.fastmail.fm

September 21, 2009

CovVise

SiCortex

CovVise was created at and for SiCortex, Inc.

SiCortex formally closed in June 2009.

R.I.P.

SiCortex

CovVise

Agenda

- Tradition Dictates
  - And Our Deviations
- CovVise Language Extensions
  - SystemC Extensions
  - SystemVerilog Extensions
- CovVise Post-Simulation
  - CovVise Database
  - CovVise Web Interface
  - Metrigator
- Conclusions
- Q&A

CovVise

Tradition: Verification Team Does It

- Traditionally,
  - Only the Verification Team adds coverage
- Let the Designers also add coverage!
  - They already add “line coverage”
  - Just as they now add assertions, **when writing RTL**
  - Fifo full, empty, unlikely cross products, bypasses
- Avoid duplication
  - Some coverage is much easier when in RTL
  - And some best left to the verification team (interfaces)
- Keep RTL simple… Later slides

CovVise

Tradition: Coverage is Done Late

BOSTON 2009

- Traditionally,
  - Coverage is done near the end of the project
  - Quantifies that little was missed
- Learn from “Test Driven Development”
  - (Test Driven Development == Write tests before code)
  - Write the “test” of verification code, I.E. the coverage, FIRST!
  - Saves writing focused test when random hits unexpected bins
  - Focuses effort on big missing coverage items
    - So find important bugs faster
  - Reduces chance that interesting coverage cases are forgotten
  - Provides good metric for management

CovVise

Tradition: All Hits Are Equal

- Traditionally,
  - Per bin, all test hits against that bin contribute to a single sum
- Which is better?
  - 100 tests that hit a bin once?
  - One test that hits a bin 100 times?
  - Both count the same after aggregation some tools!

$1*100 \neq 100*1$

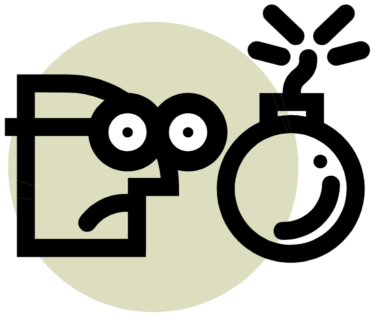
- Require some number of hits per test to count this bin as "covered"
  - Prevents initialization-only from covering bins
  - Insures good random or strong focused coverage

CovVise

Tradition: Only Passing Tests Count

“The Big Takeaway”

- Traditionally,
  - Coverage is only collected on passing tests
  - Failures shouldn’t count towards coverage goals
- But learn from the Challenger disaster
  - Didn’t graph failing cases, only successful ones
- If a bin is hit **only** by a failing test
  - This bin is unlikely to be impossible
  - This bin may indicate a bug is hiding behind it
  - Conversely, fixing the failing test would improve coverage
  - Focus effort on testing around this bin



CovVise

Agenda

- Tradition Dictates
  - And Our Deviations
- CovVise Language Extensions
  - SystemC Extensions
  - SystemVerilog Extensions
- CovVise Post-Simulation
  - CovVise Database
  - CovVise Web Interface
  - Metrigator
- Conclusions
- Q&A

CovVise

SystemC Extensions

- We extended SystemC to provide coverage ala SystemVerilog, via the SystemPerl pre-processor

```
SC_MODULE(myModule) {           // A SystemC Module
  SP_COVERGROUP myGroup (
    coverpoint myCoverPoint {
      bins seven = 7;                 // single value
      bins three_to_five = [3:5];   // range
      bins members_of_enum = enum_type; // enum
    };
  );
  ...
  void process() {             // a method of the class
    if (sampling_signal) {          // when to sample
       SP_COVER_SAMPLE(myGroup);  // increment
    }
```

CovVise

SystemC Multidimensional Crosses

- We also allow crosses, illegals (asserts) and ignores

```
SC_MODULE(myModule) {
  ...
  SP_COVERGROUP myGroup (
    coverpoint first {
      bins three_to_five = [3:5];
    };
    coverpoint second[8] = [0:7];
    cross myCross {
      rows = {first};
      cols = {second};
    };
    illegal_bins_func = myCross_illegal()
    ignore_bins_func = myCross_ignore()
  );
```

CovVise

SystemVerilog Extensions

SNUG BOSTON 2009

- RTL is procedural
  - Coverage should be too... Alas the language doesn't allow this
- Allowed Designers to use $ucover_* macros, for example

```
always @* begin
  if (...) begin
    $ucover_clk(clock, label)
```

- Vpassert (part of Verilog-Perl) expands this to:

```
reg _temp;
label: cover property (@(posedge clock) _temp)
always @* begin
  _tempsig = 0;
  if (...) begin
    _tempsig = 1;
```

- Works with SV formal tools, too

CovVise

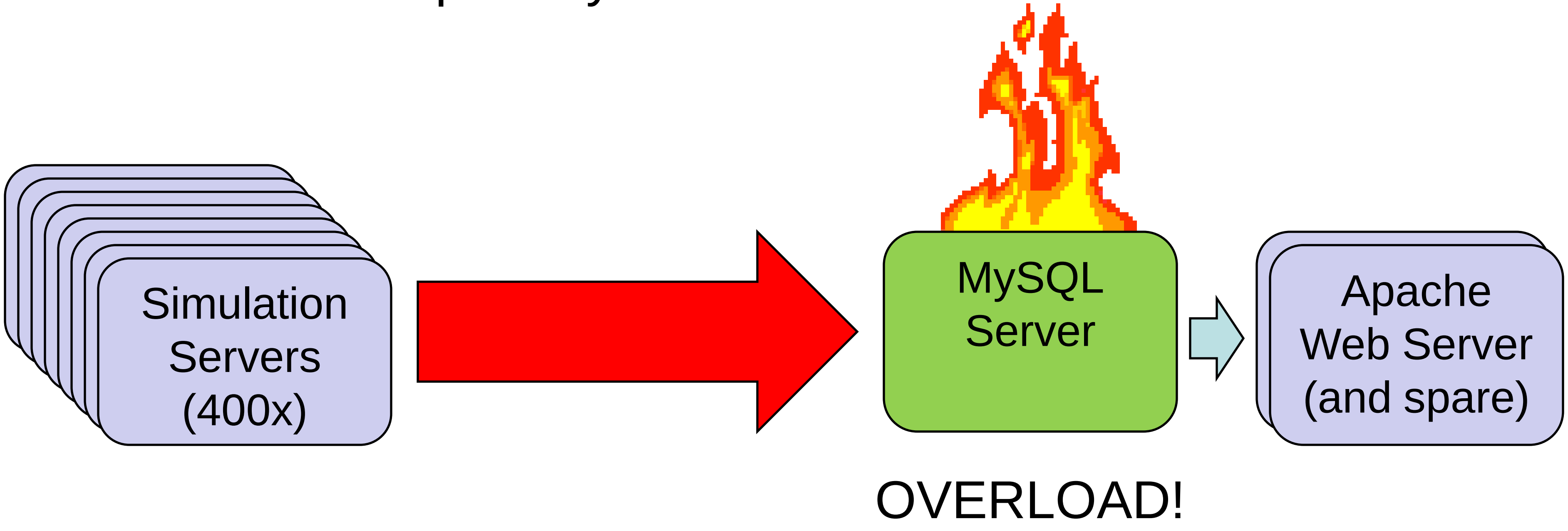
Agenda

- Tradition Dictates
  - And Our Deviations
- CovVise Language Extensions
  - SystemC Extensions
  - SystemVerilog Extensions
- CovVise Post-Simulation
  - CovVise Database
  - CovVise Web Interface
  - Metrigator
- Conclusions
- Q&A

CovVise

CovVise Database 1

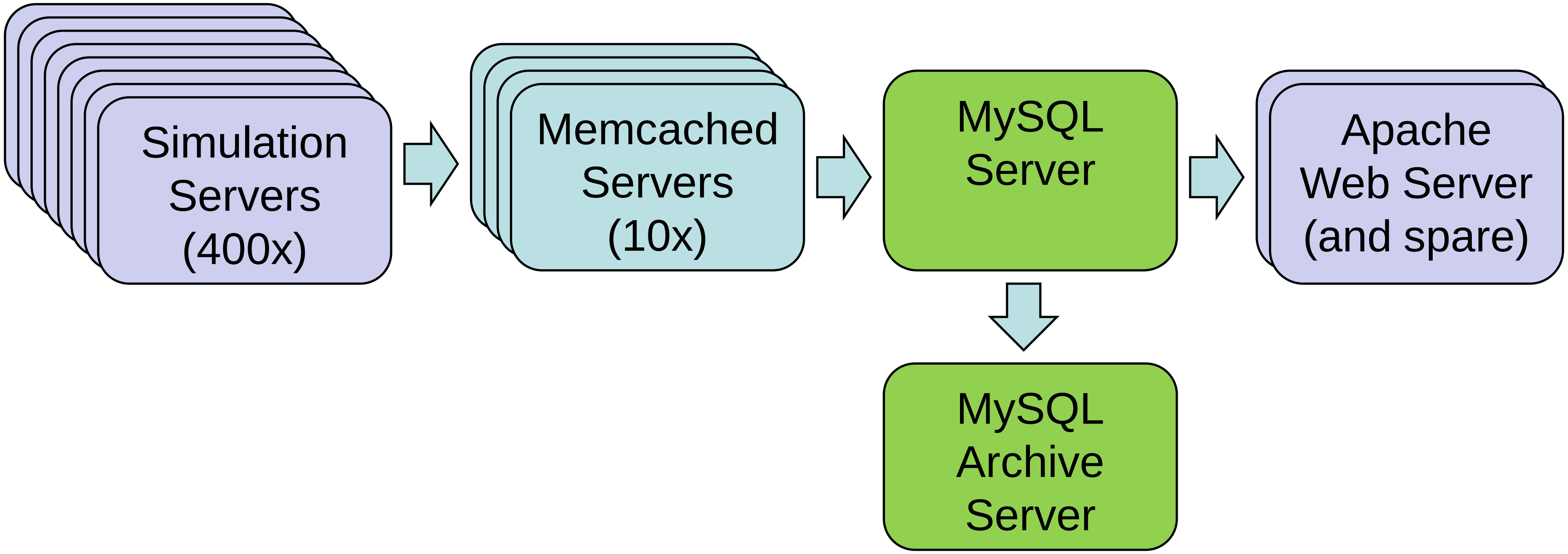
- High demands: 100k tests * 100k bins
- 10B inserts per day



CovVise

CovVise Database 2

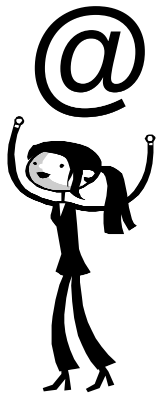
- High demands: 100k tests * 100k bins
- 10B inserts per day
- Old data auto-pushed to archival database



CovVise

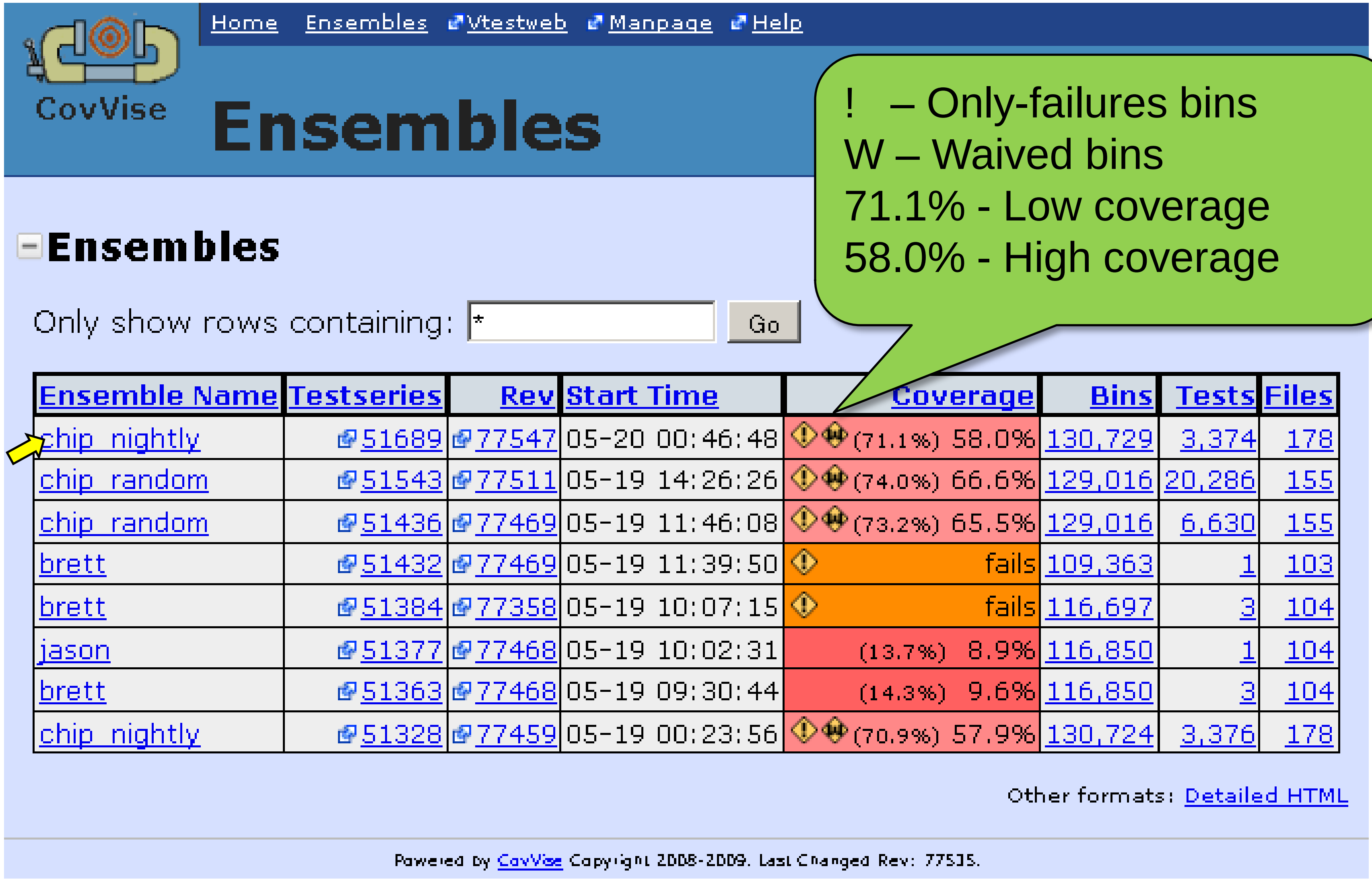
CovVise Web Interface

- Users interface to CovVise data through the web
- “Simple enough even a VP can use it” ™ ☺
- Data is presented as hierarchy of coverage “pages”
- The interface begins with CovVise home page, which list “ensembles” of test runs…



CovVise

Web: Ensembles



Home Ensembles Vtestweb Manpage Help

CovVise

Ensembles

Ensembles

Only show rows containing: * Go

Ensemble Name	Testseries	Rev	Start Time	Coverage	Bins	Tests	Files
chip_nightly	51689	77547	05-20 00:46:48	! W (71.1%) 58.0%	130,729	3,374	178
chip_random	51543	77511	05-19 14:26:26	! W (74.0%) 66.6%	129,016	20,286	155
chip_random	51436	77469	05-19 11:46:08	! W (73.2%) 65.5%	129,016	6,630	155
brett	51432	77469	05-19 11:39:50	! fails	109,363	1	103
brett	51384	77358	05-19 10:07:15	! fails	116,697	3	104
jason	51377	77468	05-19 10:02:31	(13.7%) 8.9%	116,850	1	104
brett	51363	77468	05-19 09:30:44	(14.3%) 9.6%	116,850	3	104
chip_nightly	51328	77459	05-19 00:23:56	! W (70.9%) 57.9%	130,724	3,376	178

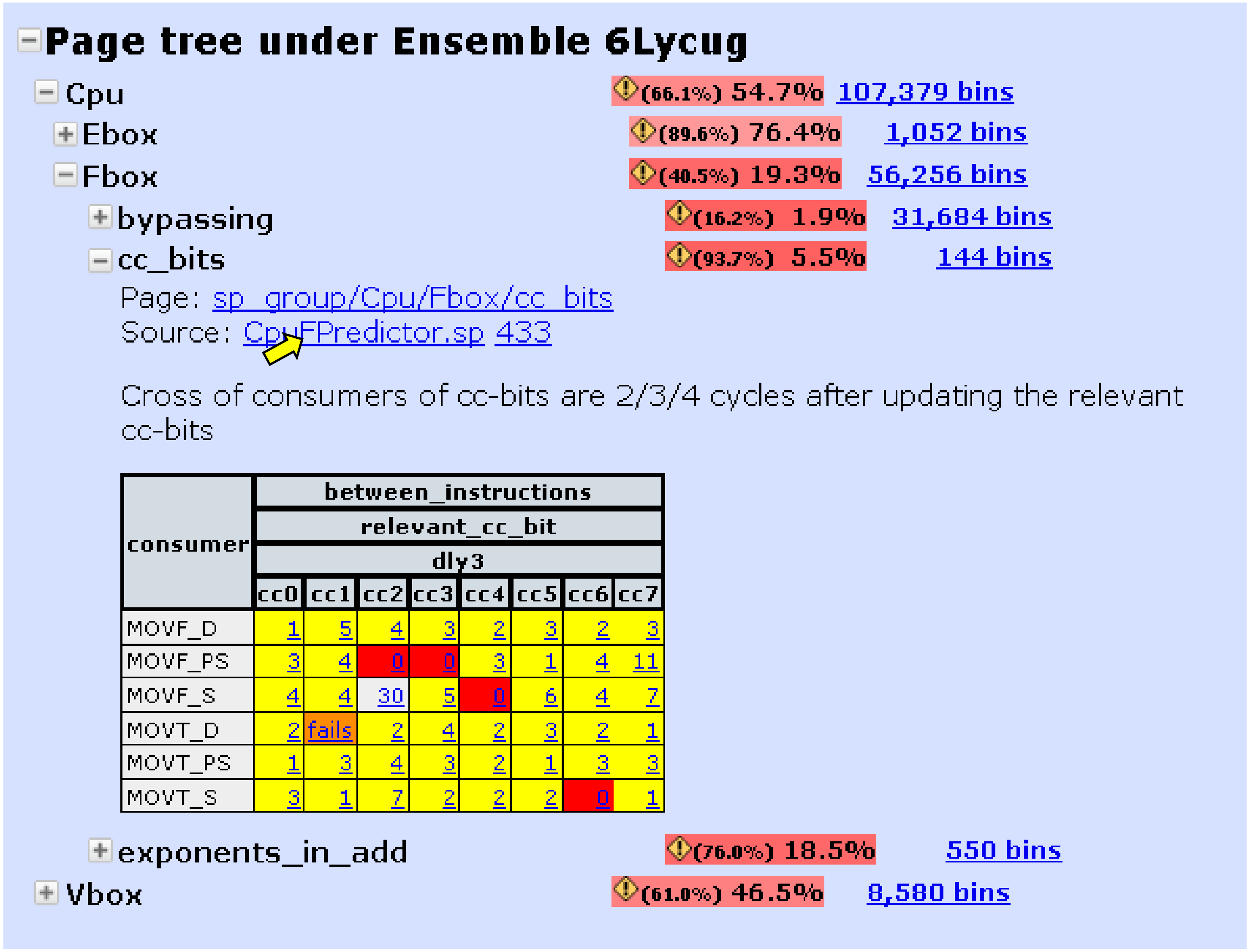
Other formats: Detailed HTML

Powered by CovVise Copyright 2008-2009. Last Changed Rev: 77535.

! – Only-failures bins
W – Waived bins
71.1% - Low coverage
58.0% - High coverage

CovVise

Web: Page Tree



Page tree under Ensemble 6Lycug

− Cpu	(66.1%) 54.7%	107,379 bins
+ Ebox	(89.6%) 76.4%	1,052 bins
− Fbox	(40.5%) 19.3%	56,256 bins
+ bypassing	(16.2%) 1.9%	31,684 bins
− cc_bits	(93.7%) 5.5%	144 bins

Page: sp_group/Cpu/Fbox/cc_bits
Source: CpuFPredictor.sp 433

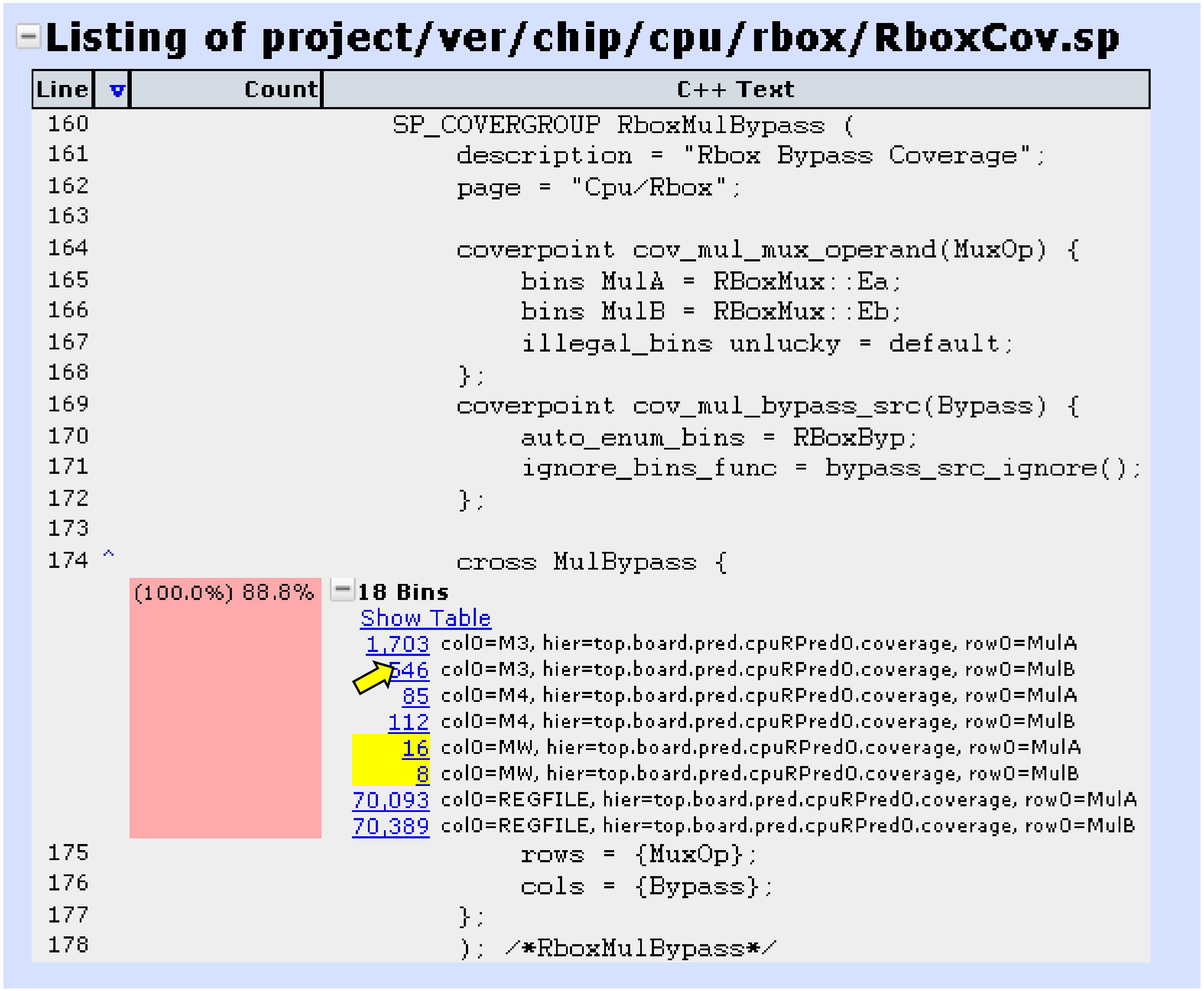
Cross of consumers of cc-bits are 2/3/4 cycles after updating the relevant cc-bits

consumer	between_instructions							
	relevant_cc_bit							
	dly3							
	cc0	cc1	cc2	cc3	cc4	cc5	cc6	cc7
MOVF_D	1	5	4	3	2	3	2	3
MOVF_PS	3	4	0	0	3	1	4	11
MOVF_S	4	4	30	5	0	6	4	7
MOVT_D	2	fails	2	4	2	3	2	1
MOVT_PS	1	3	4	3	2	1	3	3
MOVT_S	3	1	7	2	2	2	0	1

+ exponents_in_add	(76.0%) 18.5%	550 bins
+ Vbox	(61.0%) 46.5%	8,580 bins

CovVise

Web: Files



Listing of project/ver/chip/cpu/rbox/RboxCov.sp

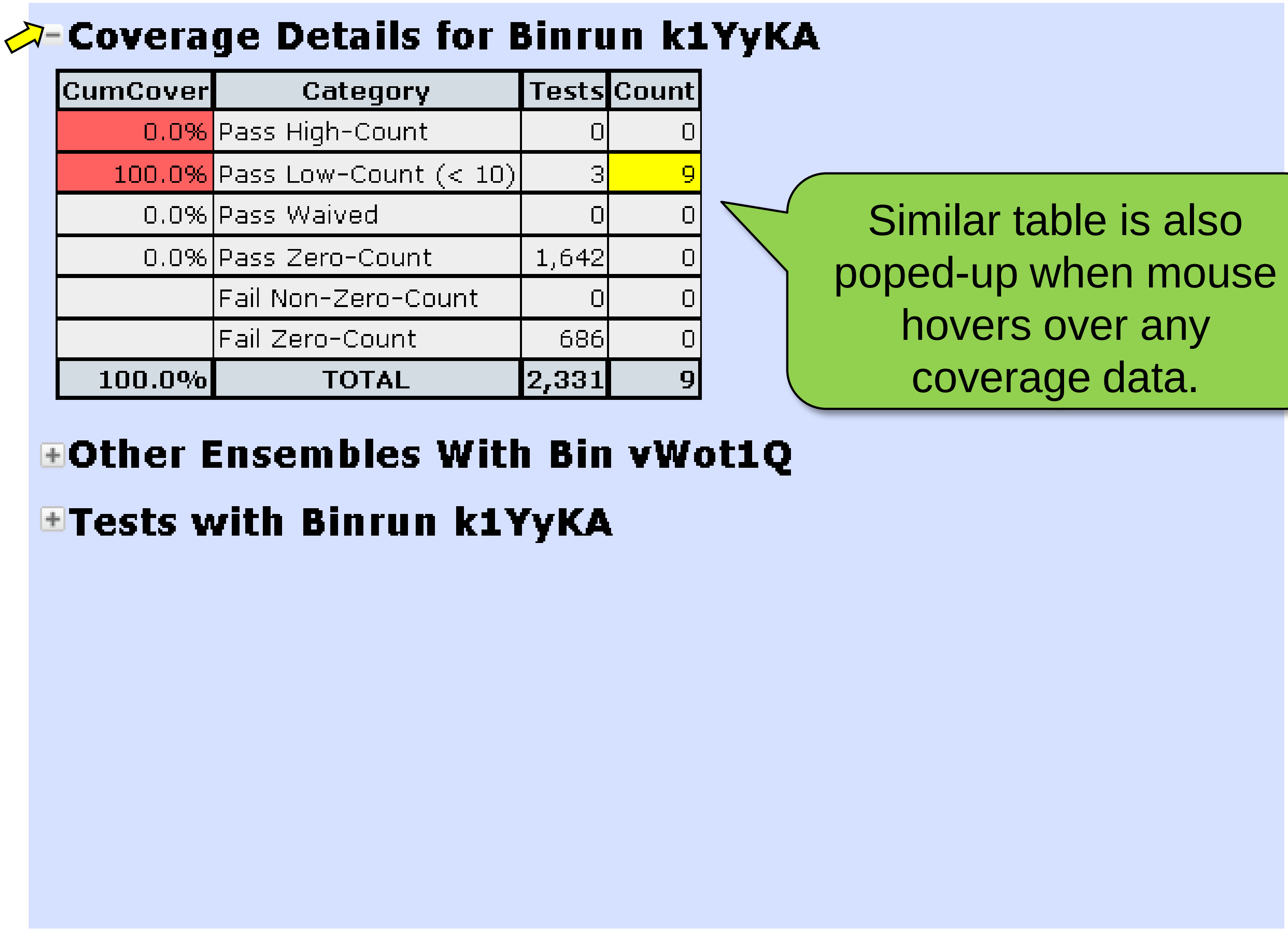
```
Line ▼        Count                    C++ Text
                      SP_COVERGROUP RboxMulBypass (
                          description = "Rbox Bypass Coverage";
                          page = "Cpu/Rbox";

                          coverpoint cov_mul_mux_operand(MuxOp) {
                              bins MulA = RBoxMux::Ea;
                              bins MulB = RBoxMux::Eb;
                              illegal_bins unlucky = default;
                          };
                          coverpoint cov_mul_bypass_src(Bypass) {
                              auto_enum_bins = RBoxByp;
                              ignore_bins_func = bypass_src_ignore();
                          };

     ^                    cross MulBypass {
   (100.0%) 88.8%   18 Bins
                    Show Table
                     1,703 col0=M3, hier=top.board.pred.cpuRPred0.coverage, row0=MulA
                       546 col0=M3, hier=top.board.pred.cpuRPred0.coverage, row0=MulB
                        85 col0=M4, hier=top.board.pred.cpuRPred0.coverage, row0=MulA
                       112 col0=M4, hier=top.board.pred.cpuRPred0.coverage, row0=MulB
                        16 col0=MW, hier=top.board.pred.cpuRPred0.coverage, row0=MulA
                         8 col0=MW, hier=top.board.pred.cpuRPred0.coverage, row0=MulB
                    70,093 col0=REGFILE, hier=top.board.pred.cpuRPred0.coverage, row0=MulA
                    70,389 col0=REGFILE, hier=top.board.pred.cpuRPred0.coverage, row0=MulB
                              rows = {MuxOp};
                              cols = {Bypass};
                          };
                      ); /*RboxMulBypass*/
```

CovVise

Web: Binruns 1



Coverage Details for Binrun k1YyKA

CumCover	Category	Tests	Count
0.0%	Pass High-Count	0	0
100.0%	Pass Low-Count (< 10)	3	9
0.0%	Pass Waived	0	0
0.0%	Pass Zero-Count	1,642	0
	Fail Non-Zero-Count	0	0
	Fail Zero-Count	686	0
100.0%	TOTAL	2,331	9

Other Ensembles With Bin vWot1Q

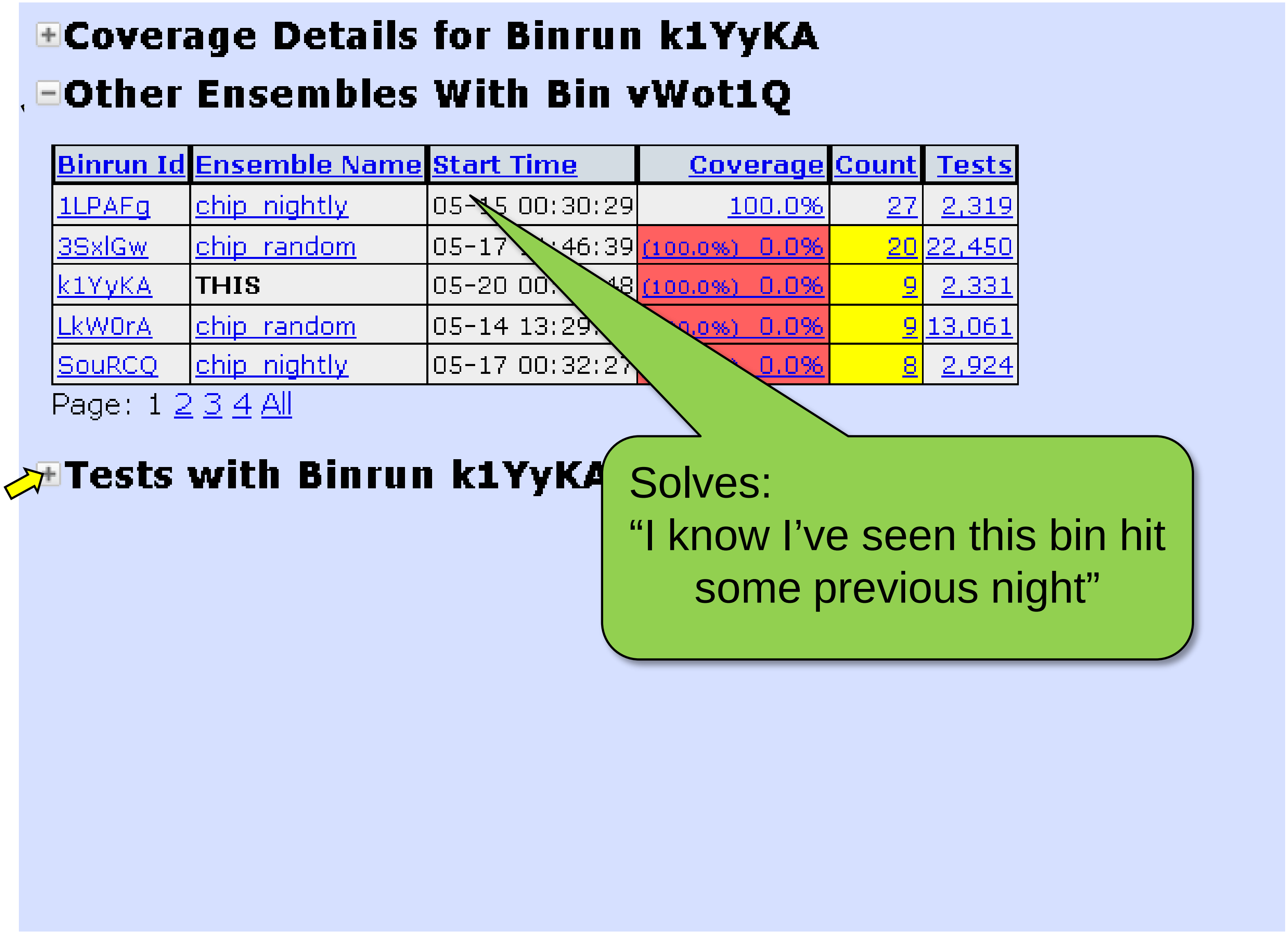
Tests with Binrun k1YyKA

Similar table is also poped-up when mouse hovers over any coverage data.

CovVise

Web: Binruns 2

BOSTON 2009



⊞ **Coverage Details for Binrun k1YyKA**

⊟ **Other Ensembles With Bin vWot1Q**

Binrun Id	Ensemble Name	Start Time	Coverage	Count	Tests
1LPAFg	chip_nightly	05-15 00:30:29	100.0%	27	2,319
3SxlGw	chip_random	05-17 [illegible]:46:39	(100.0%) 0.0%	20	22,450
k1YyKA	THIS	05-20 00:[illegible]48	(100.0%) 0.0%	9	2,331
LkW0rA	chip_random	05-14 13:29:[illegible]	[illegible].0%) 0.0%	9	13,061
SouRCQ	chip_nightly	05-17 00:32:27	[illegible] 0.0%	8	2,924

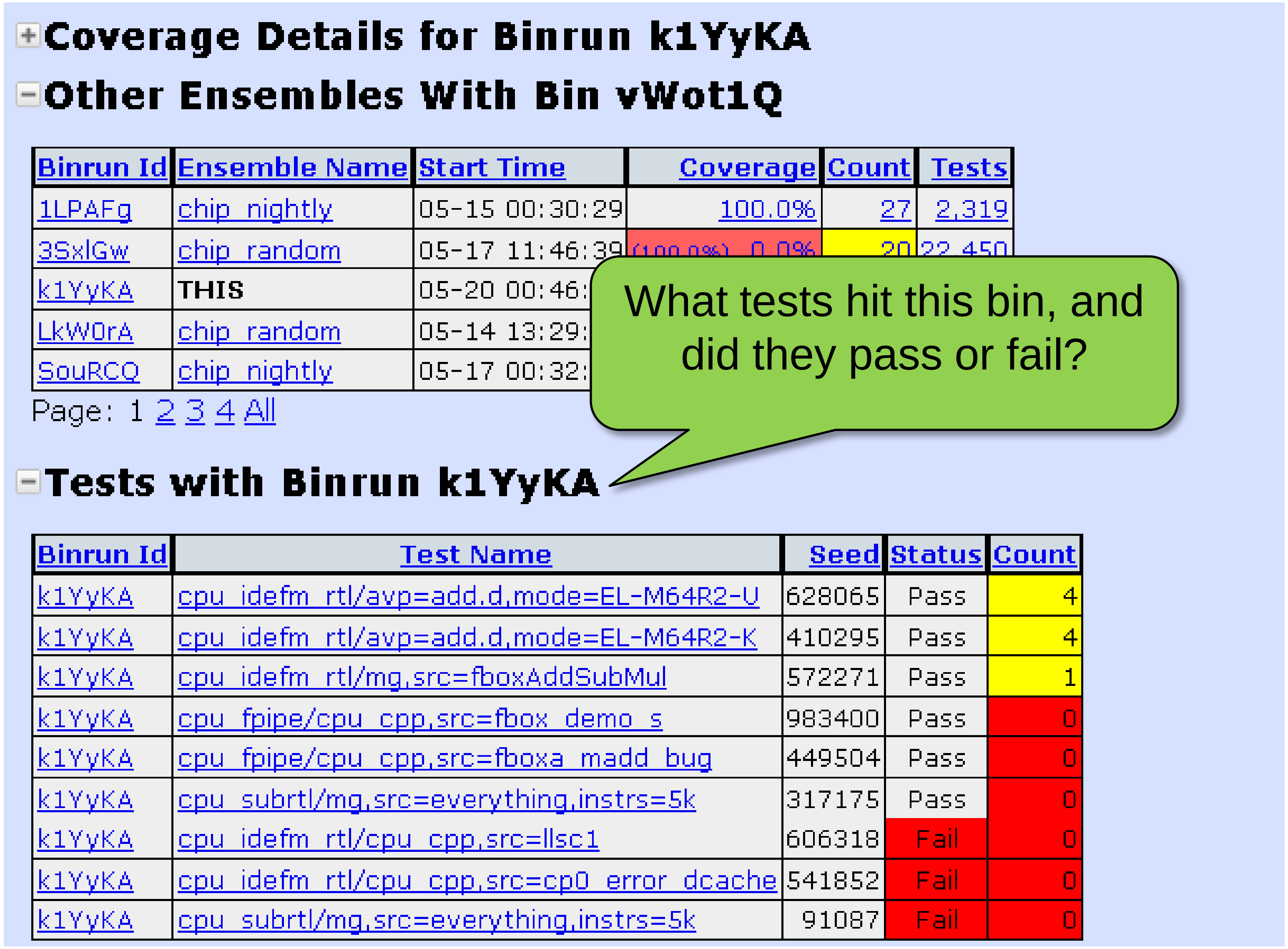
Page: 1 2 3 4 All

⊞ **Tests with Binrun k1YyKA**

Solves:
"I know I've seen this bin hit some previous night"

CovVise

Web: Binruns 3



⊞ **Coverage Details for Binrun k1YyKA**

⊟ **Other Ensembles With Bin vWot1Q**

Binrun Id	Ensemble Name	Start Time	Coverage	Count	Tests
1LPAFg	chip_nightly	05-15 00:30:29	100.0%	27	2,319
3SxlGw	chip_random	05-17 11:46:39	(100.0%) 0.0%	20	22,450
k1YyKA	THIS	05-20 00:46:			
LkW0rA	chip_random	05-14 13:29:			
SouRCQ	chip_nightly	05-17 00:32:			

Page: 1 2 3 4 All

⊟ **Tests with Binrun k1YyKA**

What tests hit this bin, and did they pass or fail?

Binrun Id	Test Name	Seed	Status	Count
k1YyKA	cpu_idefm_rtl/avp=add.d,mode=EL-M64R2-U	628065	Pass	4
k1YyKA	cpu_idefm_rtl/avp=add.d,mode=EL-M64R2-K	410295	Pass	4
k1YyKA	cpu_idefm_rtl/mg,src=fboxAddSubMul	572271	Pass	1
k1YyKA	cpu_fpipe/cpu_cpp,src=fbox_demo_s	983400	Pass	0
k1YyKA	cpu_fpipe/cpu_cpp,src=fboxa_madd_bug	449504	Pass	0
k1YyKA	cpu_subrtl/mg,src=everything,instrs=5k	317175	Pass	0
k1YyKA	cpu_idefm_rtl/cpu_cpp,src=llsc1	606318	Fail	0
k1YyKA	cpu_idefm_rtl/cpu_cpp,src=cp0_error_dcache	541852	Fail	0
k1YyKA	cpu_subrtl/mg,src=everything,instrs=5k	91087	Fail	0

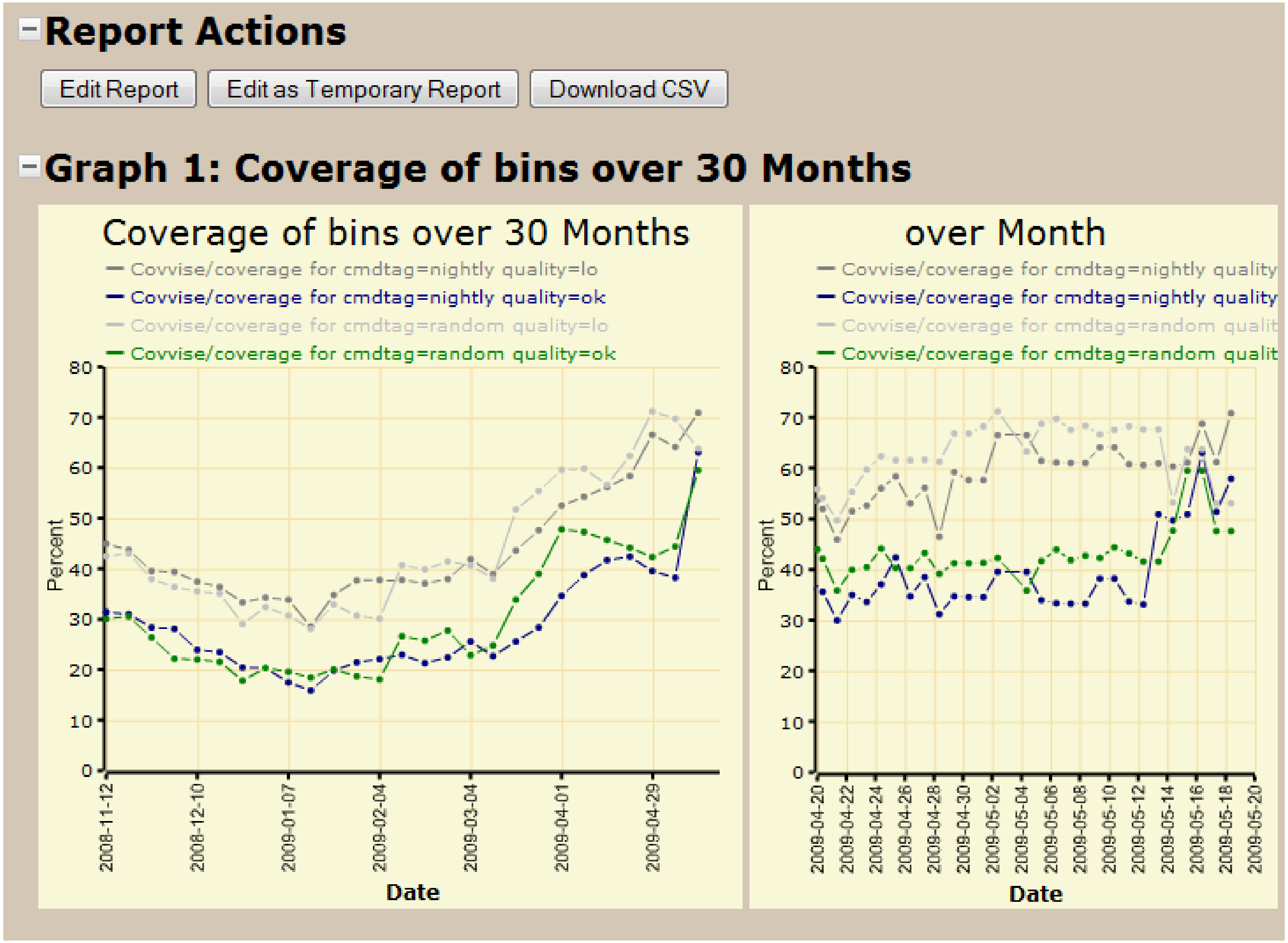
CovVise

Metrigator

- Metrigator: Our verification metrics database
  - CovVise coverage (percent low coverage, ok coverage, number of bins)
  - Bug count (total, closed, per-component, by priority, etc)
  - Bug closure rate (total, per-component, by priority, etc)
  - Source code commits (size, number of edits)
  - Verification test success (number of tests, failures)
- Spots Correlated Trends
- Appeases Management ☺

CovVise

Metrigator Coverage Graph



Report Actions

Graph 1: Coverage of bins over 30 Months

Edit Report
Edit as Temporary Report
Download CSV
Coverage of bins over 30 Months
Covvise/coverage for cmdtag=nightly quality=lo
Covvise/coverage for cmdtag=nightly quality=ok
Covvise/coverage for cmdtag=random quality=lo
Covvise/coverage for cmdtag=random quality=ok
Percent
Date
2008-11-12
2008-12-10
2009-01-07
2009-02-04
2009-03-04
2009-04-01
2009-04-29
over Month
Covvise/coverage for cmdtag=nightly quality
Covvise/coverage for cmdtag=nightly quality
Covvise/coverage for cmdtag=random qualit
Covvise/coverage for cmdtag=random qualit
2009-04-20
2009-04-22
2009-04-24
2009-04-26
2009-04-28
2009-04-30
2009-05-02
2009-05-04
2009-05-06
2009-05-08
2009-05-10
2009-05-12
2009-05-14
2009-05-16
2009-05-18
2009-05-20

CovVise

Agenda

- Tradition Dictates
  - And Our Deviations
- CovVise Language Extensions
  - SystemC Extensions
  - SystemVerilog Extensions
- CovVise Post-Simulation
  - CovVise Database
  - CovVise Web Interface
  - Metrigator
- Conclusions
- Q&A

CovVise

Conclusions

- Verification Engineers got
  - SystemC extended with SystemVerilog-ish coverage
  - Data collected on failing tests, to easily detect interesting bins
- Designers got
  - Coverage as part of normal RTL procedural statements
  - Easy browsing of data
- Management got
  - Early coverage for progress tracking and work reduction
  - Pretty graphs
- If we were to do it again?
  - Start coverage insertion even earlier
  - [SiCortex,] Don’t run out of money ☺

CovVise

Open Source

- The open source design tools are available at http://www.veripool.org
  - These slides + paper at http://www.veripool.org/papers/
  - CovVise – Have you been paying attention?
  - SystemPerl – /*AUTOs*/ for SystemC
  - Verilog-Perl – Toolkit with Preprocessing, Renaming, etc
  - Verilator – Compile SystemVerilog into SystemC

- Additional Tools
  - Make::Cache - Object caching for faster compiles
  - Schedule::Load – Load Balancing (ala LSF)
  - Verilog-Mode for Emacs – /*AUTO…*/ Expansion
  - Vregs – Extract register and class declarations from documentation