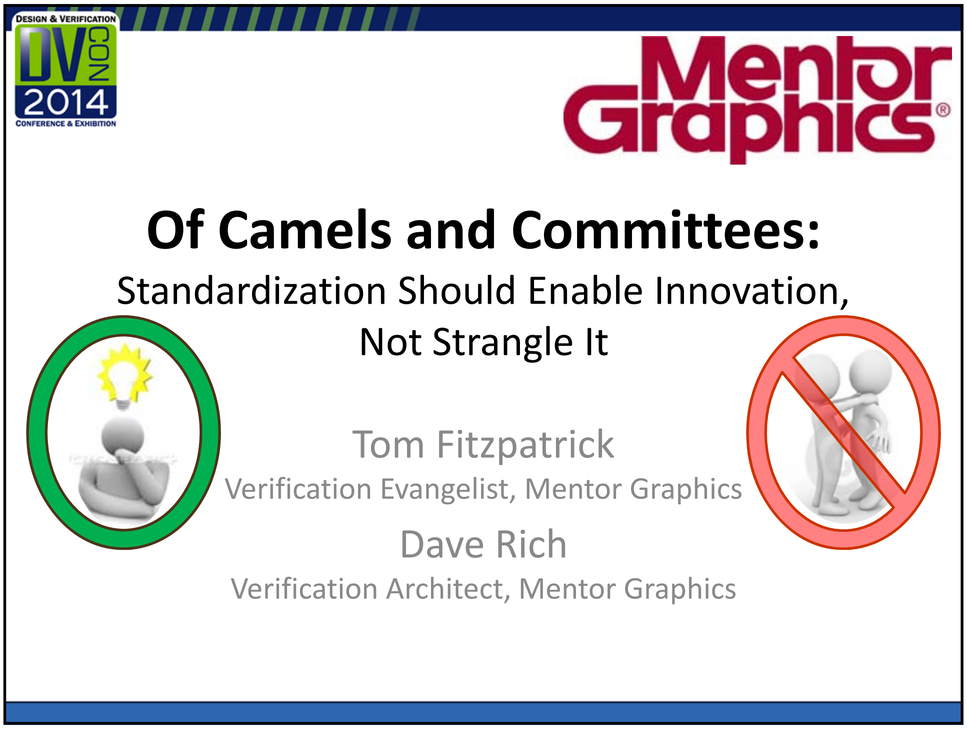


Of Camels and Committees:

Standardization Should Enable Innovation, Not Strangle It

Tom Fitzpatrick
Verification Evangelist, Mentor Graphics

Dave Rich
Verification Architect, Mentor Graphics

The First Commandment For Effective Standards

The Ten Commandments for Effective Standards

Practical Insights for Creating Technical Standards

Tried and true

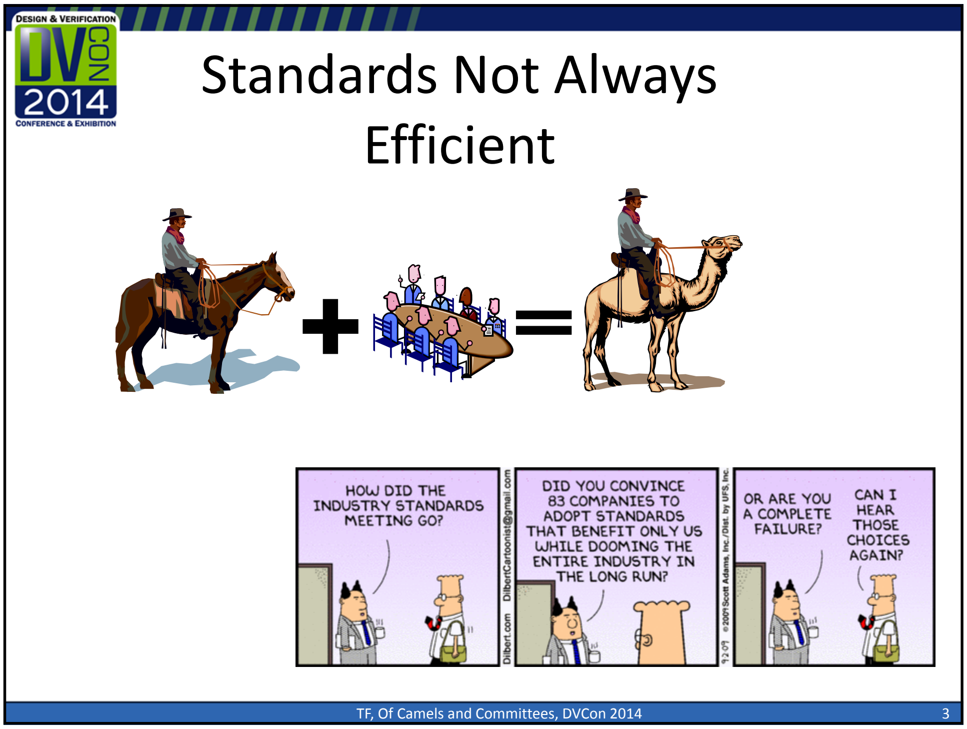
Karen Bartleson

Cartoons by Rick Jamison

"Cooperate on Standards Compete on Products"

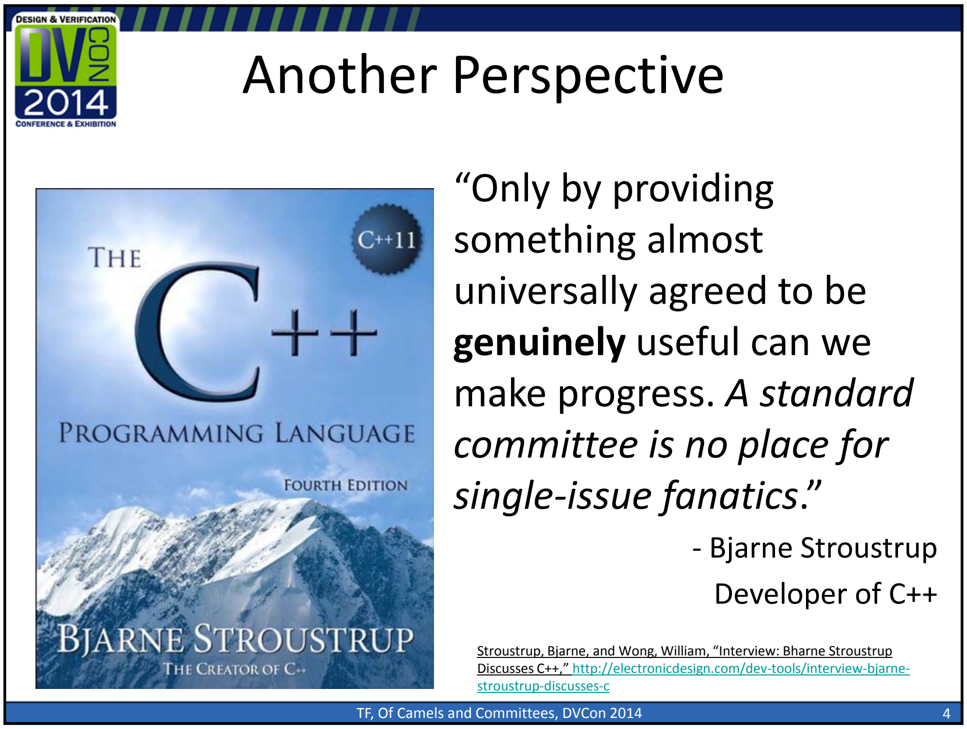
- Karen Bartleson

TF, Of Camels and Committees, DVCon 2014 2



Standards Not Always Efficient

HOW DID THE INDUSTRY STANDARDS MEETING GO?
DID YOU CONVINCE 83 COMPANIES TO ADOPT STANDARDS THAT BENEFIT ONLY US WHILE DOOMING THE ENTIRE INDUSTRY IN THE LONG RUN?
OR ARE YOU A COMPLETE FAILURE?
CAN I HEAR THOSE CHOICES AGAIN?
Dilbert.com DilbertCartoonist@gmail.com
©2007 Scott Adams, Inc./Dist. by UFS, Inc.



Another Perspective

"Only by providing something almost universally agreed to be **genuinely** useful can we make progress. *A standard committee is no place for single-issue fanatics.*"

- Bjarne Stroustrup

Developer of C++

Stroustrup, Bjarne, and Wong, William, "Interview: Bharne Stroustrup Discusses C++," http://electronicdesign.com/dev-tools/interview-bjarne-stroustrup-discusses-c

Standardization Can Remove Choice

IEEE

IEEE Standard for Verilog[®] Hardware Description Language

IEEE Computer Society

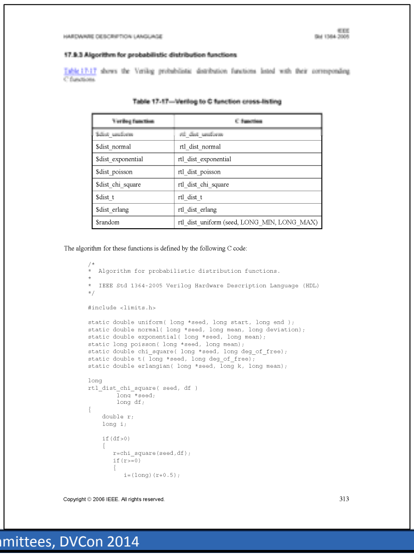
Sponsored by the
Design Automation Standards Committee

1364™

IEEE
3 Park Avenue
New York, NY 10016-5997, USA

7 April 2006

IEEE Std 1364™-2005
(Revision of IEEE Std 1364-2001)



HARDWARE DESCRIPTION LANGUAGE

IEEE
Std 1364-2005

17.9.3 Algorithm for probabilistic distribution functions

Table 17-17 shows the Verilog probabilistic distribution functions listed with their corresponding C functions.

Table 17-17—Verilog to C function cross-listing

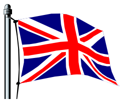
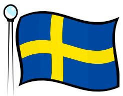
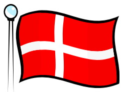
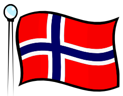
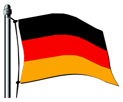
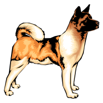
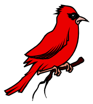
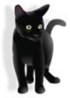
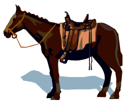
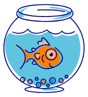
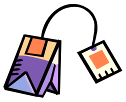
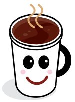
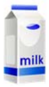
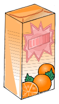
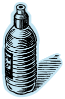
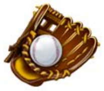
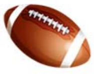
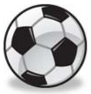
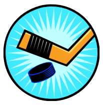
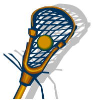
Verilog function	C function
$dist_uniform	rtl_dist_uniform
$dist_normal	rtl_dist_normal
$dist_exponential	rtl_dist_exponential
$dist_poisson	rtl_dist_poisson
$dist_chi_square	rtl_dist_chi_square
$dist_t	rtl_dist_t
$dist_erlang	rtl_dist_erlang
$random	rtl_dist_uniform (seed, LONG_MIN, LONG_MAX)

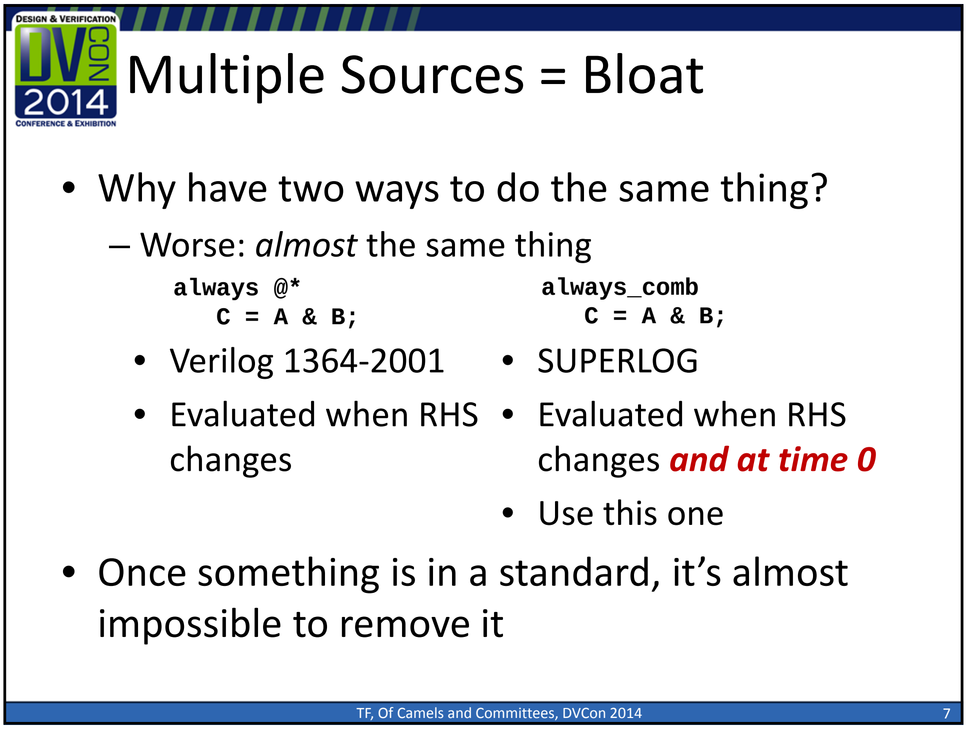
The algorithm for these functions is defined by the following C code:

```
/*
*  Algorithm for probabilistic distribution functions.
*
*  IEEE Std 1364-2005 Verilog Hardware Description Language (HDL)
*/

#include <limits.h>

static double uniform( long *seed, long start, long end );
static double normal( long *seed, long mean, long deviation);
static double exponential( long *seed, long mean);
static long poisson( long *seed, long mean);
static double chi_square( long *seed, long deg_of_free);
static double t( long *seed, long deg_of_free);
static double erlangian( long *seed, long k, long mean);

long
rtl_dist_chi_square( seed, df )
        long *seed;
        long df;
{
    double r;
    long i;

    if(df>0)
    {
       r=chi_square(seed,df);
       if(r>=0)
       {
          i=(long)(r+0.5);
```

313

Solver Innovation: Einstein Puzzle

Nationality					
House Color	🟥	⬜	🟨	🟩	🟦
Pet					
Drink					
Sport					



Multiple Sources = Bloat

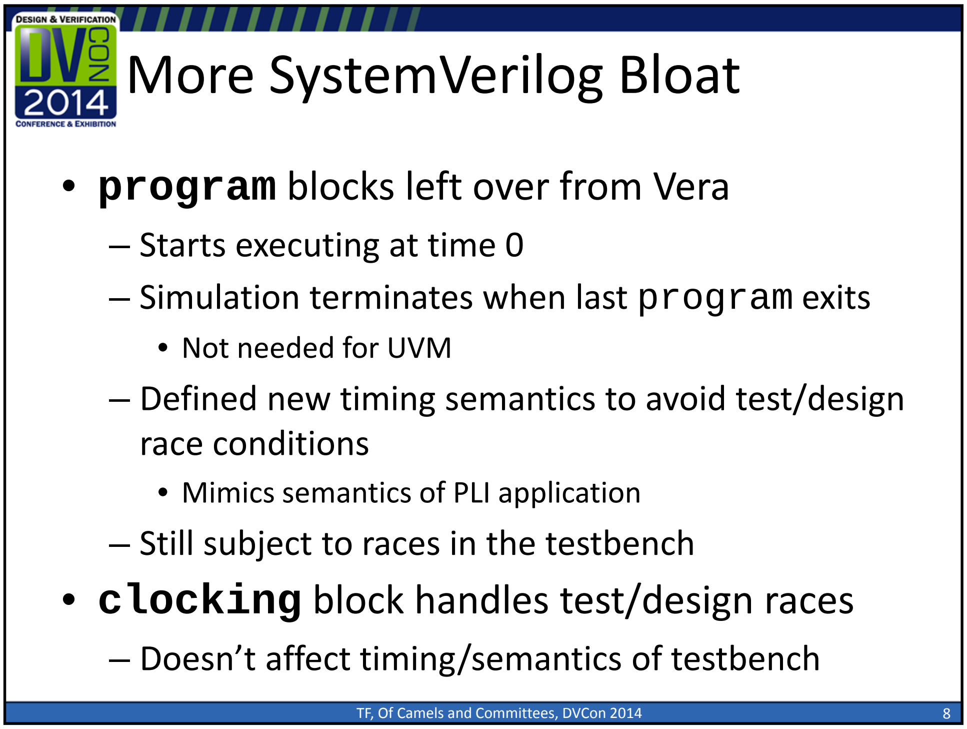
- Why have two ways to do the same thing?
  - Worse: *almost* the same thing

```
always @*                 always_comb
   C = A & B;                C = A & B;
```

- Verilog 1364-2001
- Evaluated when RHS changes

- SUPERLOG
- Evaluated when RHS changes ***and at time 0***
- Use this one

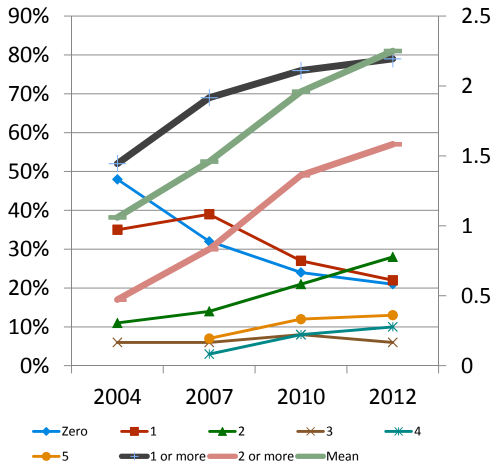
- Once something is in a standard, it's almost impossible to remove it



More SystemVerilog Bloat

- **`program`** blocks left over from Vera
  - Starts executing at time 0
  - Simulation terminates when last `program` exits
    - Not needed for UVM
  - Defined new timing semantics to avoid test/design race conditions
    - Mimics semantics of PLI application
  - Still subject to races in the testbench
- **`clocking`** block handles test/design races
  - Doesn't affect timing/semantics of testbench

Use of Embedded CPUs Increasing



- Mean # of CPUs:
  - 2004: 1.06
  - 2012: 2.25

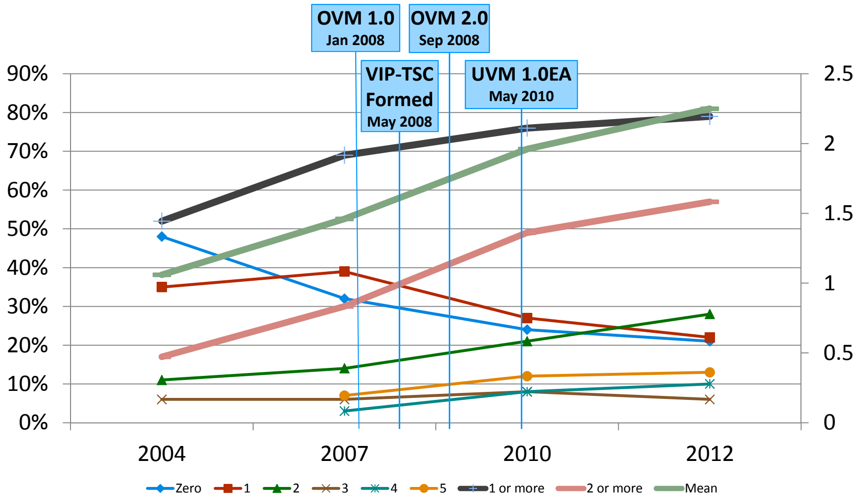
  ⇧212%
- % Designs with >= 1 CPUs
  - 2004: 52%
  - 2012: 79%

  ⇧152%
- % Designs with >= 2 CPUs
  - 2004: 17%
  - 2012: 57%

  ⇧335%

Source: Wilson Research Group and Mentor Graphics, 2012 Functional Verification Study

UVM Targeting the Past?



OVM 1.0 Jan 2008

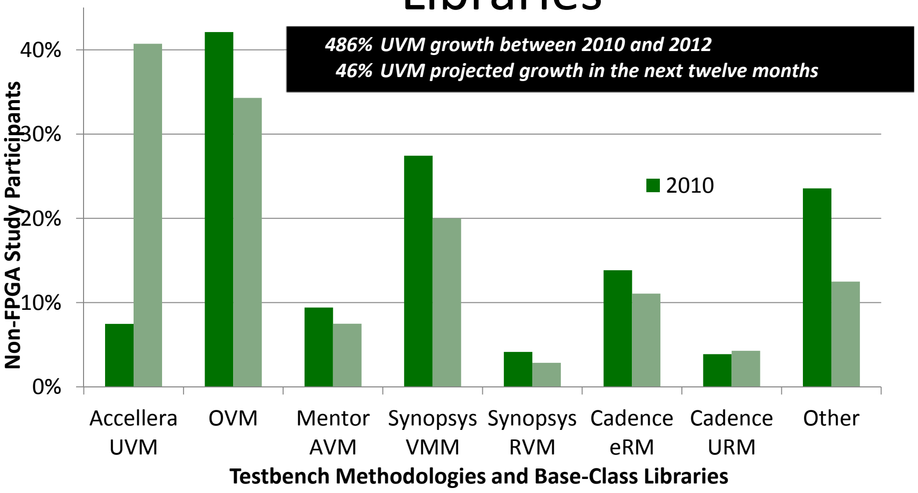
OVM 2.0 Sep 2008

VIP-TSC Formed May 2008

UVM 1.0EA May 2010

Source: Wilson Research Group and Mentor Graphics, 2012 Functional Verification Study

Standardization in Base Class Libraries



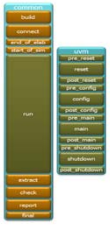
486% UVM growth between 2010 and 2012
46% UVM projected growth in the next twelve months

Source: Wilson Research Group and Mentor Graphics, 2012 Functional Verification Study

User Feedback is Critical

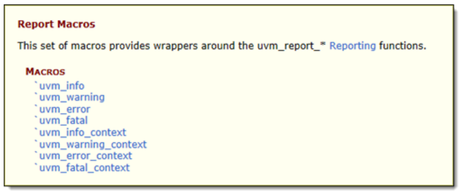
- New code should benefit users
  - Phasing still doesn’t work
  - “Top Priority”
  - Don’t repeat mistakes
- UVM1.2 introduces 13 backward-incompatible changes
  - Considerable performance degradation
  - No extended user feedback period

	Classes	Files	Lines
UVM1.0-p1	288	125	65534
UVM1.1	311	131	66660
UVM1.1a	322	135	67307
UVM1.1b	317	134	67724
UVM1.1c	317	135	67969
UVM1.1d	316	133	67965
UVM1.2	350	144	75445



Troublesome New UVM Features: UVM Messaging

1.1d



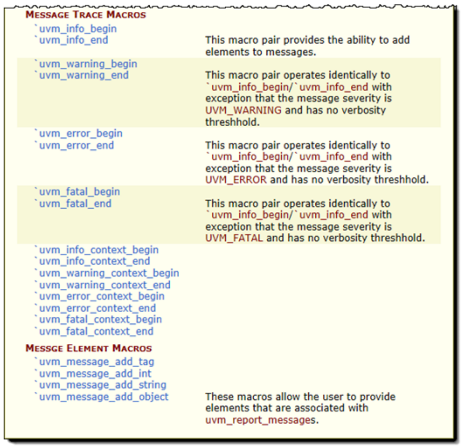
Report Macros

This set of macros provides wrappers around the uvm_report_* Reporting functions.

Macros

```
`uvm_info
`uvm_warning
`uvm_error
`uvm_fatal
`uvm_info_context
`uvm_warning_context
`uvm_error_context
`uvm_fatal_context
```

1.2



Message Trace Macros

`uvm_info_begin `uvm_info_end	This macro pair provides the ability to add elements to messages.
`uvm_warning_begin `uvm_warning_end	This macro pair operates identically to `uvm_info_begin/`uvm_info_end with exception that the message severity is UVM_WARNING and has no verbosity threshhold.
`uvm_error_begin `uvm_error_end	This macro pair operates identically to `uvm_info_begin/`uvm_info_end with exception that the message severity is UVM_ERROR and has no verbosity threshhold.
`uvm_fatal_begin `uvm_fatal_end	This macro pair operates identically to `uvm_info_begin/`uvm_info_end with exception that the message severity is UVM_FATAL and has no verbosity threshhold.
`uvm_info_context_begin `uvm_info_context_end `uvm_warning_context_begin `uvm_warning_context_end `uvm_error_context_begin `uvm_error_context_end `uvm_fatal_context_begin `uvm_fatal_context_end	

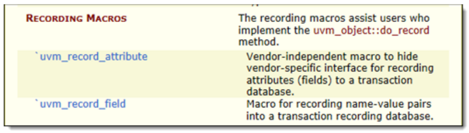
Messge Element Macros

`uvm_message_add_tag `uvm_message_add_int `uvm_message_add_string `uvm_message_add_object	These macros allow the user to provide elements that are associated with uvm_report_messages.

- 20 new macros added
- Add multiple fields to a message
- *Different actions per field*
- 20% performance penalty
  - Just to process existing messages
  - without using new features

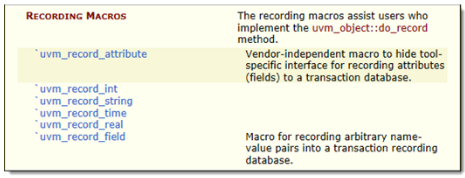
Troublesome New UVM Features: UVM Transaction Recording

1.1d



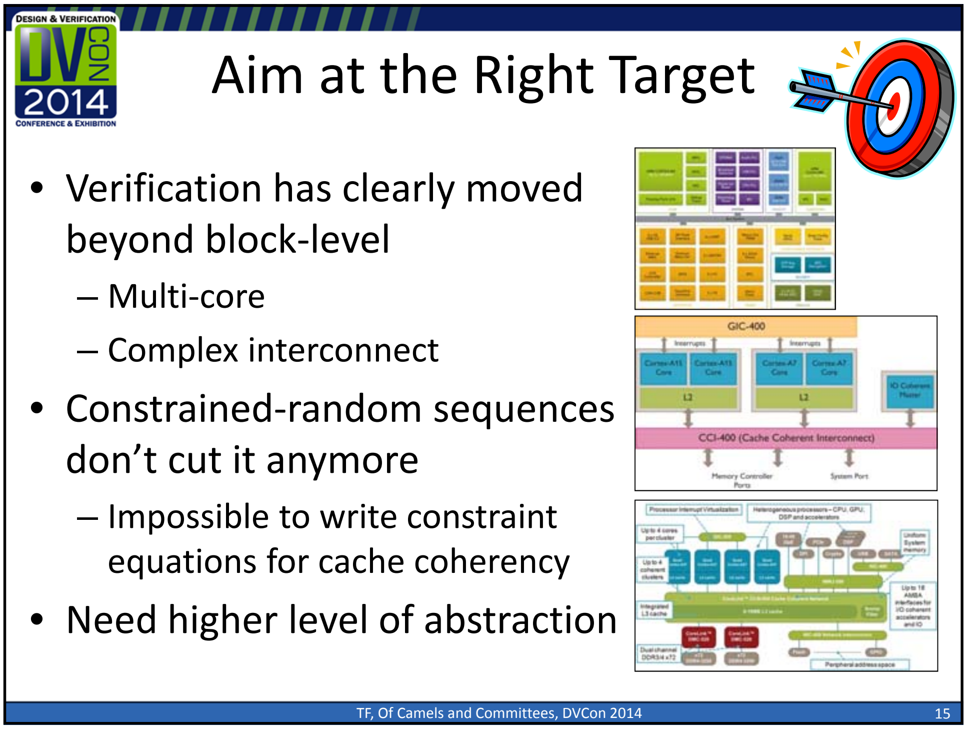
Recording Macros	The recording macros assist users who implement the uvm_object::do_record method.
`uvm_record_attribute	Vendor-independent macro to hide vendor-specific interface for recording attributes (fields) to a transaction database.
`uvm_record_field	Macro for recording name-value pairs into a transaction recording database.

1.2



Recording Macros	The recording macros assist users who implement the uvm_object::do_record method.
`uvm_record_attribute	Vendor-independent macro to hide tool-specific interface for recording attributes (fields) to a transaction database.
`uvm_record_int `uvm_record_string `uvm_record_time `uvm_record_real `uvm_record_field	Macro for recording arbitrary name-value pairs into a transaction recording database.

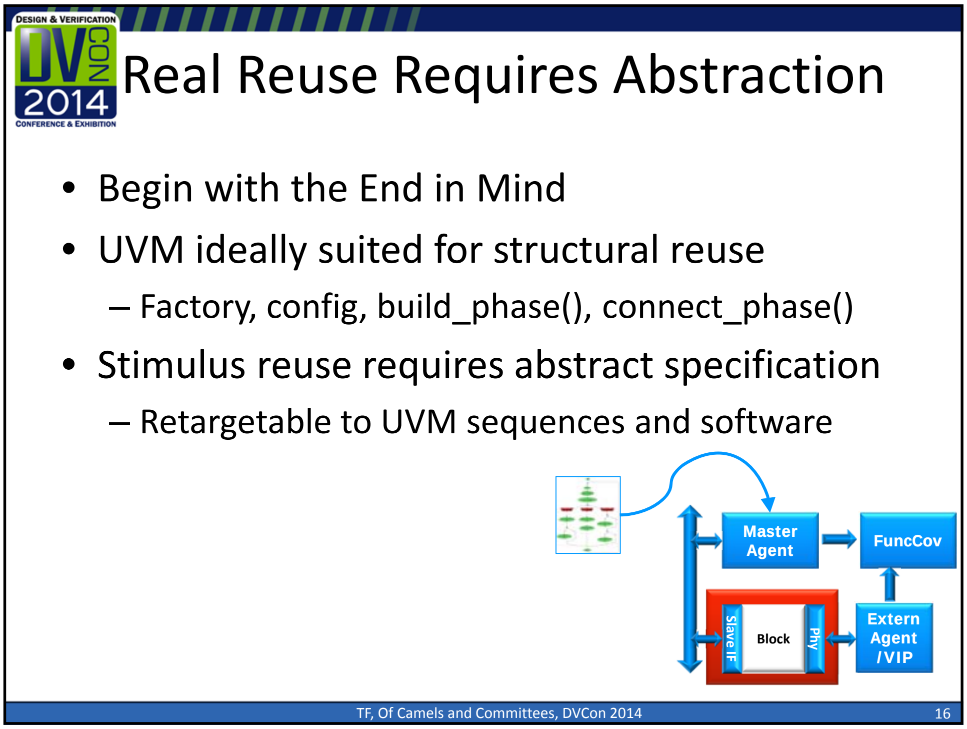
- Text-based reporting in the standard
  - Replaced by vendor-specific implementation for tools
- **NOT** Backward Compatible
  - Requires additional work by vendors to support 1.1d and 1.2
- **65% performance penalty in 1.2 vs. 1.1d**
  - Recording overhead 76% in 1.1d
  - Recording overhead **126%** in 1.2



Aim at the Right Target

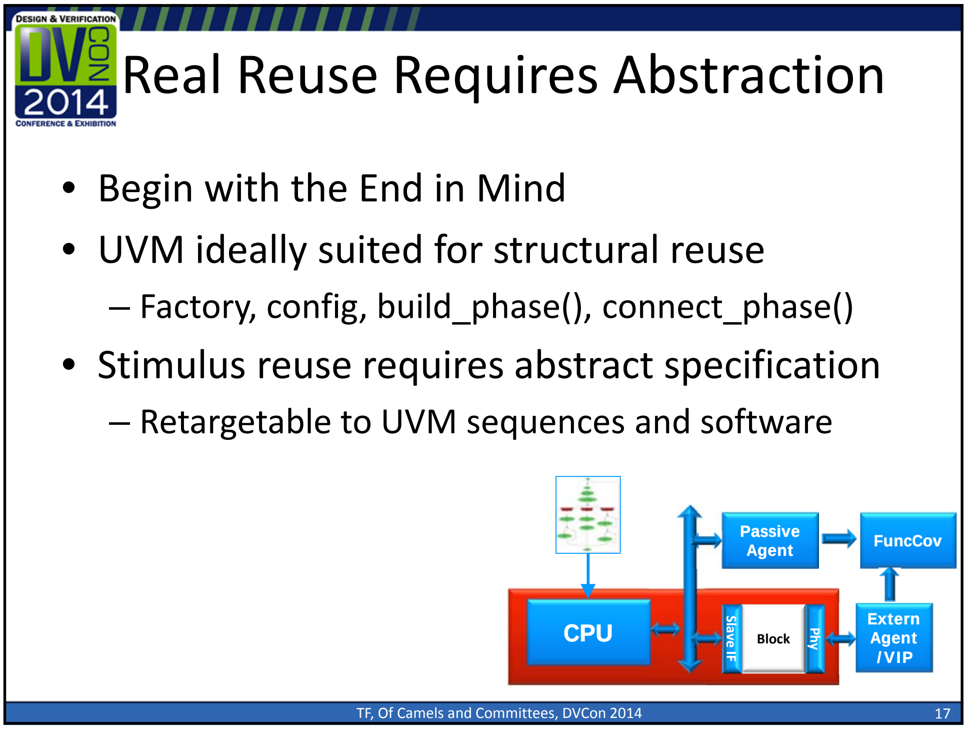
- Verification has clearly moved beyond block-level
  - Multi-core
  - Complex interconnect
- Constrained-random sequences don't cut it anymore
  - Impossible to write constraint equations for cache coherency
- Need higher level of abstraction

GIC-400
Interrupts
Interrupts
Cortex-A15 Core
Cortex-A15 Core
Cortex-A7 Core
Cortex-A7 Core
L2
L2
IO Coherent Master
CCI-400 (Cache Coherent Interconnect)
Memory Controller Ports
System Port
Processor Interrupt Virtualization
Heterogeneous processors – CPU, GPU, DSP and accelerators
Up to 4 cores per cluster
Up to 4 coherent clusters
Uniform System memory
Integrated L3 cache
Up to 18 AMBA interfaces for I/O coherent accelerators and IO
Dual channel DDR3/4 x72
Peripheral address space



Real Reuse Requires Abstraction

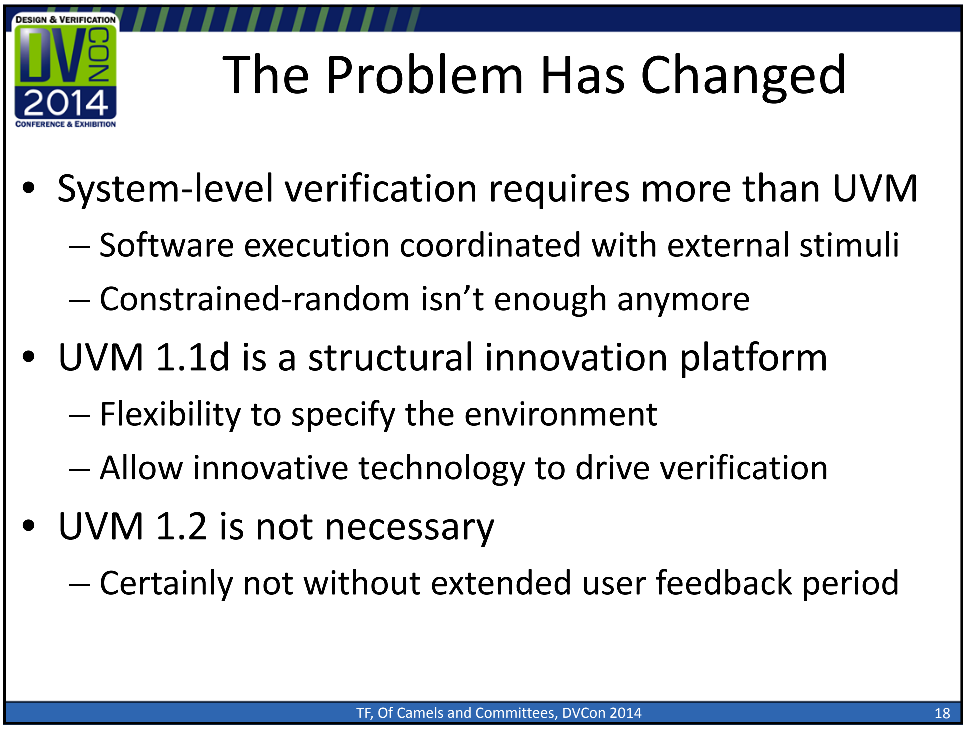
- Begin with the End in Mind
- UVM ideally suited for structural reuse
  - Factory, config, build_phase(), connect_phase()
- Stimulus reuse requires abstract specification
  - Retargetable to UVM sequences and software



Real Reuse Requires Abstraction

- Begin with the End in Mind
- UVM ideally suited for structural reuse
  - Factory, config, build_phase(), connect_phase()
- Stimulus reuse requires abstract specification
  - Retargetable to UVM sequences and software

Passive Agent
FuncCov
CPU
Slave IF
Block
Phy
Extern Agent /VIP



The Problem Has Changed

- System-level verification requires more than UVM
  - Software execution coordinated with external stimuli
  - Constrained-random isn't enough anymore
- UVM 1.1d is a structural innovation platform
  - Flexibility to specify the environment
  - Allow innovative technology to drive verification
- UVM 1.2 is not necessary
  - Certainly not without extended user feedback period

The Answer is NOT:

Thank You