

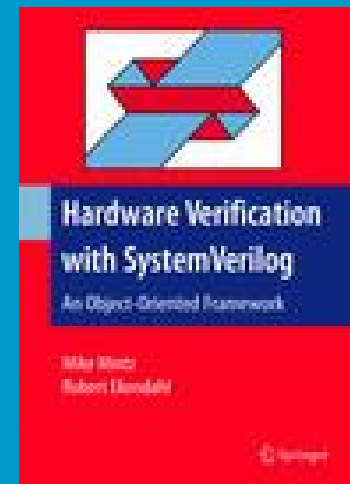
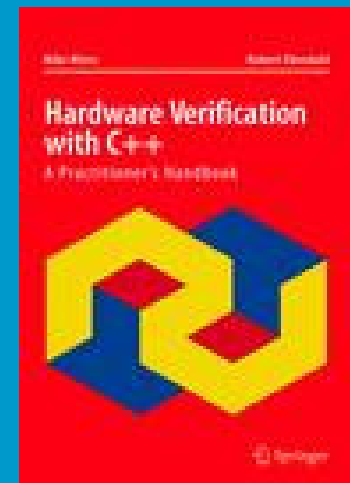
Teal `n Truss

**Useful
Open Source
Verification Code**

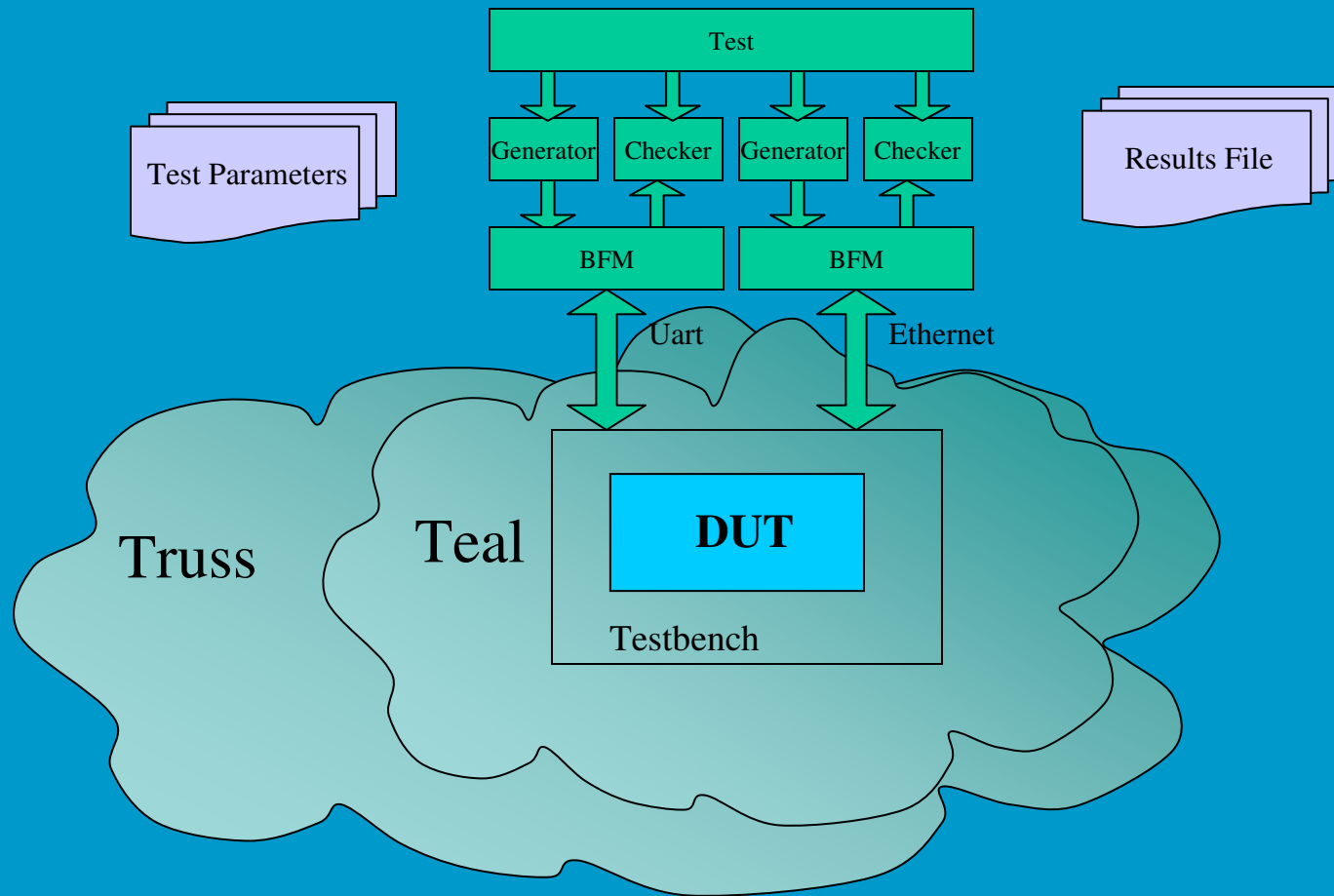
Mike Mintz
mike@trusster.com

About Me

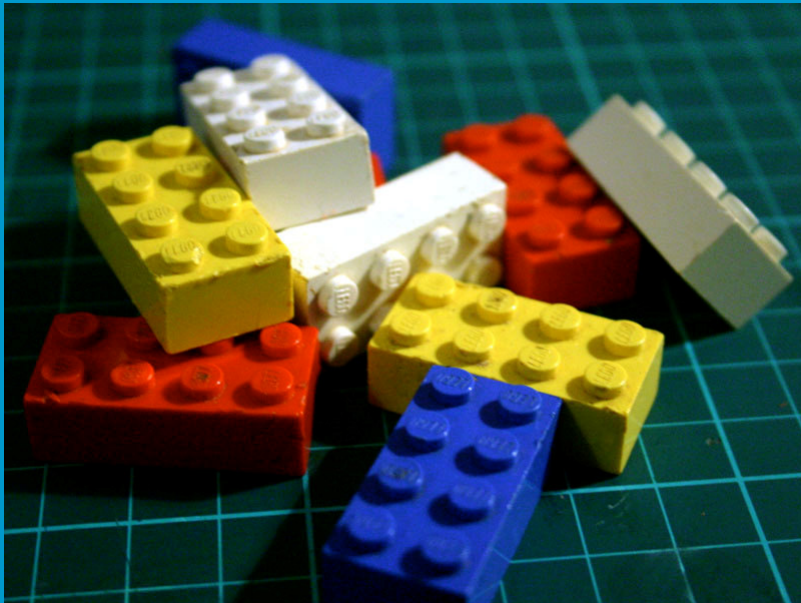
- Co-authored “Teal” and “Truss”
- Co-authored two books:
 - Hardware Verification with C++
 - Hardware Verification with SystemVerilog
- 25+ years industry experience



The Big Picture



Building Philosophy

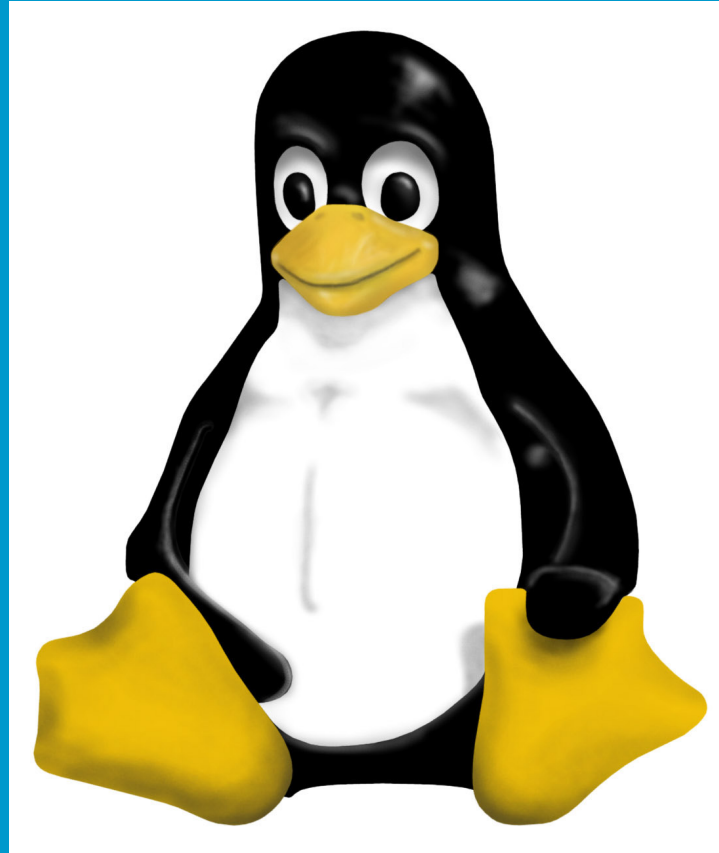


Teal



Truss

What is Teal and Truss?



A Lego Block - Dictionary

directed_test.txt

```
calea_probability 75
#include basic_test.txt
//override force_error
force_error 1
```

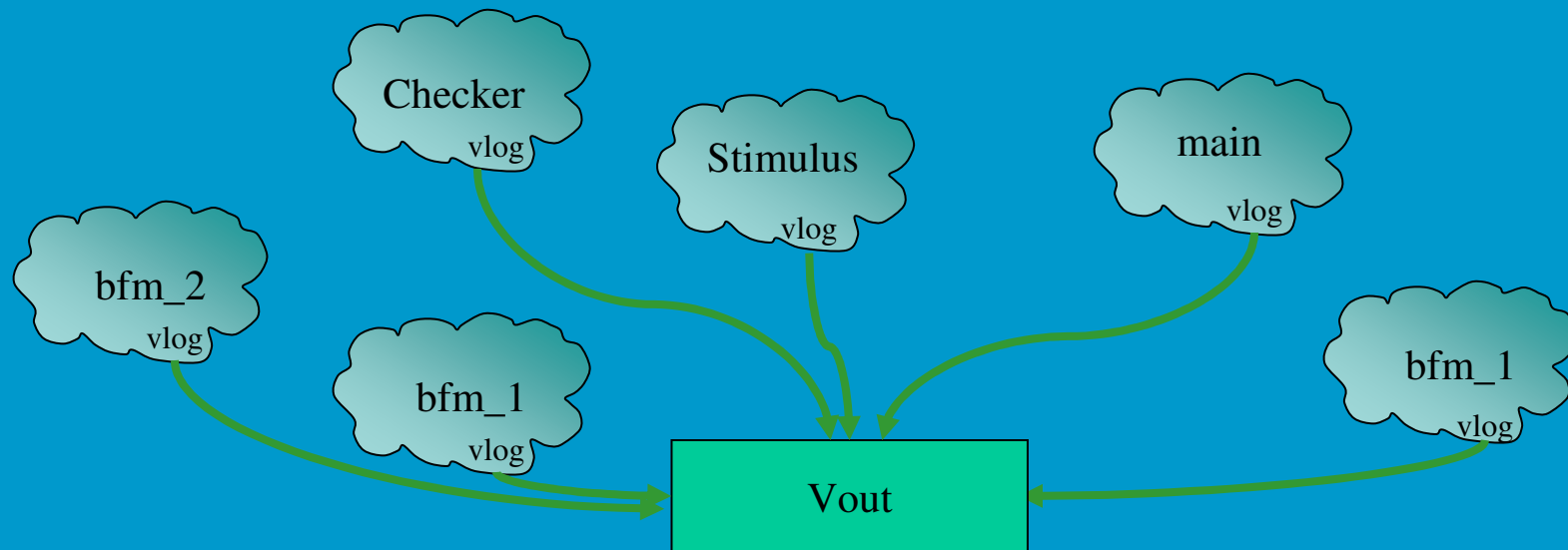
calea_probability	75
force_error	1
dma_enable	1
baud_rate	115200 921600
ds3_top_0_num_ds3	23

basic_test.txt

```
force_error 0
dma_enable 1
baud_rate 115200 921600
ds3_top_0_num_ds3 23
```

```
teal::dictionary::read_file ("path");
teal::dictionary::put ("name", "value", bool force);
value = teal::dictionary::get ("name");
value = teal::dictionary::get ("name", default);
```

A Lego Block - Logging



basic_results.txt

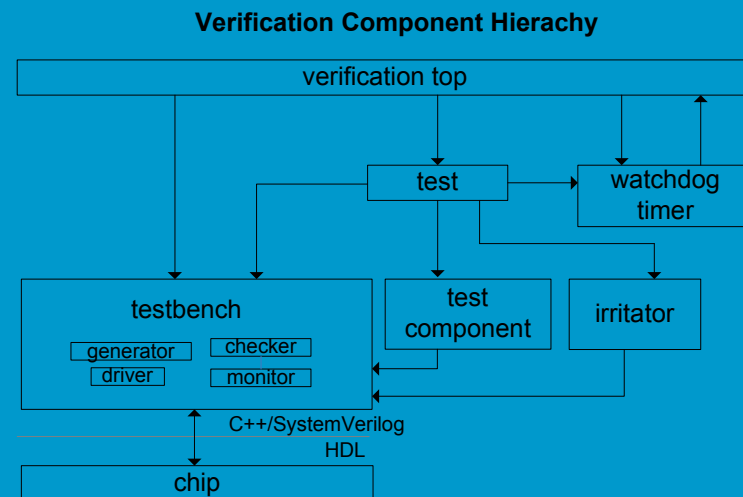
```
[0 ns] Note: Starting thread user_main  
[10 ns] [main] Waiting for reset_n.  
[12387 ns] [Stimulus] reset_n occurred. Starting up...  
[99899 ns] [bfm_1]Error: unexpected interrupt vector: 0x51  
[180000 ns] [main] Test completed and passed.
```

Building with Legos ?!?

- Legos are only small part of Verification!
- Provides overall structure
 - “the dance”
 - channels
 - Defines some common component classes

Top level components

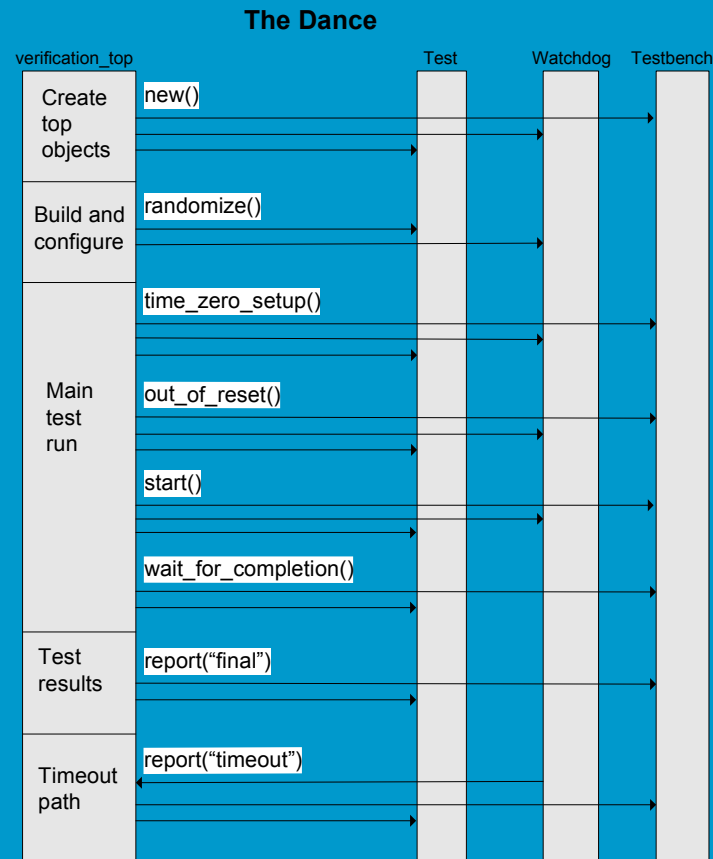
- SAME interface for all testbenches and tests!
- Testbench creates all the monitors, clocks, drivers, checkers, etc.
- Test selects features to exercise and goals to be met.



Source: *Hardware Verification in C++*, page 97

The Dance

- Common interaction for all major components
- Echoes at the VIP level
- Further echoes throughout the entire system (driver, monitor, etc)



Source: *Hardware Verification in C++*, page 97

Verification component base class

- Used by the Dance and the major classes
- Method names are conventions elsewhere
- Abstract base class, with only a name and a logger

```
class verification_component {  
  
    verification_component (const  
        std::string&);  
    virtual void randomize () = 0;  
  
    virtual void time_zero_setup () = 0;  
  
    virtual void out_of_reset (reset r) = 0;  
  
    virtual void start () = 0;  
  
    virtual void wait_for_completion () = 0;  
  
    virtual void report (const std::string) = 0;  
  
};
```

Normal System Parts

- Shutdown –calls the report()
- Watchdog – has a fail-safe HDL component
- Error_limit – stop sim after N errors

Normal System Parts(2)

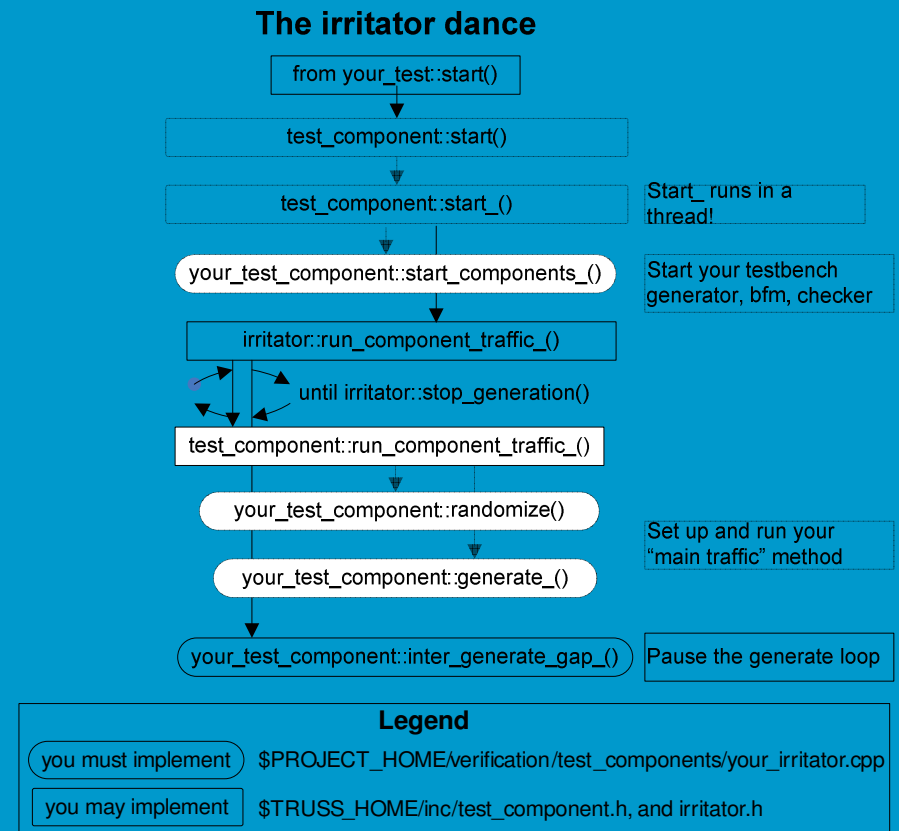
- Channel – one common interconnect
 - Simple put(), get() size() interface
 - Bounded or not
 - Has add_listener()

Advanced Testing

- Checker base class
- goals
- checker watcher

Advanced VIP capabilities

- test_component
 - One shot feature exerciser
 - Looks sort of like a top level component
 - Base for irritator
- Irritator
 - For background traffic or goals
 - Little more framework, but still flexible



Source: *Hardware Verification in C++*, page 121

Truss does not have...

- A generic Data base class
- A generic transactor base class
- Any rules for the drivers, monitors, checkers (We do have conventions and idioms).
- Templating out the wazoo

Why Use Teal and Truss?

- Company independent
- Provides a common base for others
- Minimal assumptions
- Separates universal components from domain specific code