Navigating Your Way Toward UVM version 1.2

Strategies for Adopting UVM 1.2

David C Black

Doulos Inc.

March 21, 2015

SNUG San Jose

Agenda

Introduction

Objections

Measurements

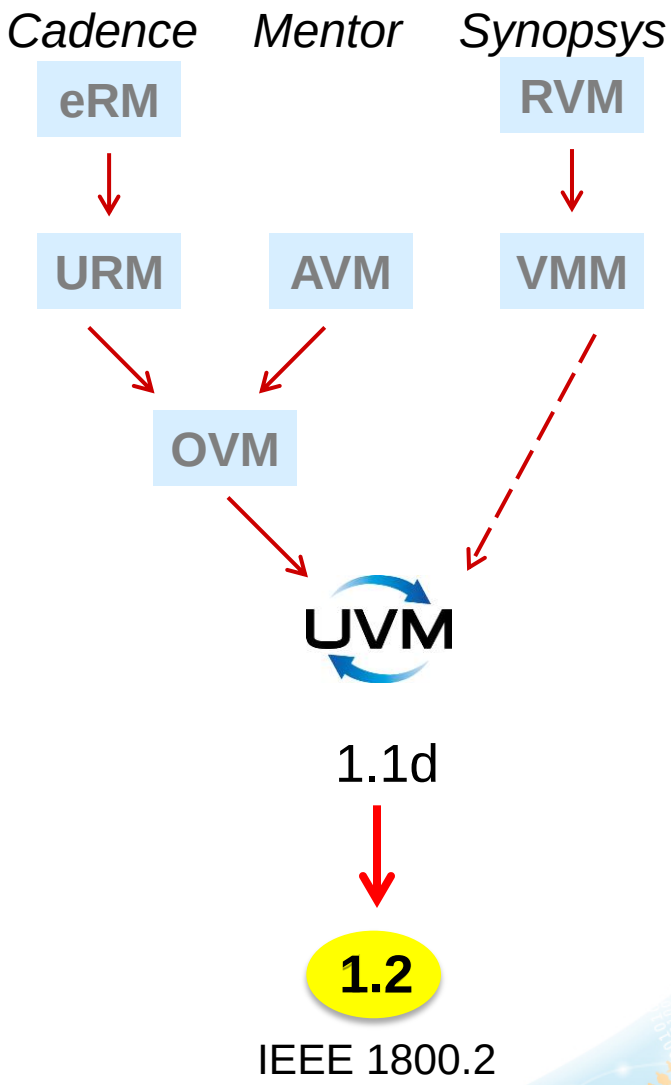
Lessons Learned

Best Practices

Conclusion

What is UVM 1.2?

- Latest version
- Still UVM and Universal
  - Interoperable and widely supported
  - Standard and moving to IEEE 1800.2
- New and improved
  - Safer (e.g. starting_phase)
  - Faster (e.g. objection propagation)
  - Cleaner (UVM_)
- Concerns
  - Slower (e.g. reporting overhead)
  - Backwards/forwards incompatibilities



UVM 1.2 Detailed Summary

- 13 Changes affecting backwards compatibility
  - Perl scripts provide automatic fixes
- 24 Semantic and API changes
- 76 Bug fixes
- 15% larger source than UVM 1.1

- Details can be read in the release notes
  - Download http://www.accellera.org/downloads/standards/uvm

UVM 1.2 Notable Changes

Enhancements

- Automatic objections
- set_propagate_mode()
- register transaction ordering
- Visitor pattern added
- VHDL backdoor
- May undo factory override
- DPI-C message rerouted
- uvm_integral_t

Safeties

- Data access policies added
- Guarded starting_phase variable
- uvm_enum_wrapper converter
- Component names checked

Incompatibilities

- Improved reporting (enhanced)
- UVM object's require ctor
- uvm_severity enum renamed
- Removed global factory reference
- set/get_config_* deprecated
- Removed objections from non-task based phases
- uvm_event parameterized
- UVM_ prefixing enumerations

UVM 1.2 Notable Changes

Enhancements

- Automatic objections
- **set_propagate_mode()**
- register transaction ordering
- Visitor pattern added
- VHDL backdoor
- May undo factory override
- DPI-C message rerouted
- uvm_integral_t

Safeties

- Data access policies added
- Guarded starting_phase variable
- uvm_enum_wrapper converter
- Component names checked

Incompatibilities

- Improved reporting (enhanced)
- UVM object's require ctor
- uvm_severity enum renamed
- Removed global factory reference
- set/get_config_* deprecated
- Removed objections from non-task based phases
- uvm_event parameterized
- UVM_ prefixing enumerations

Agenda

Introduction

Objections

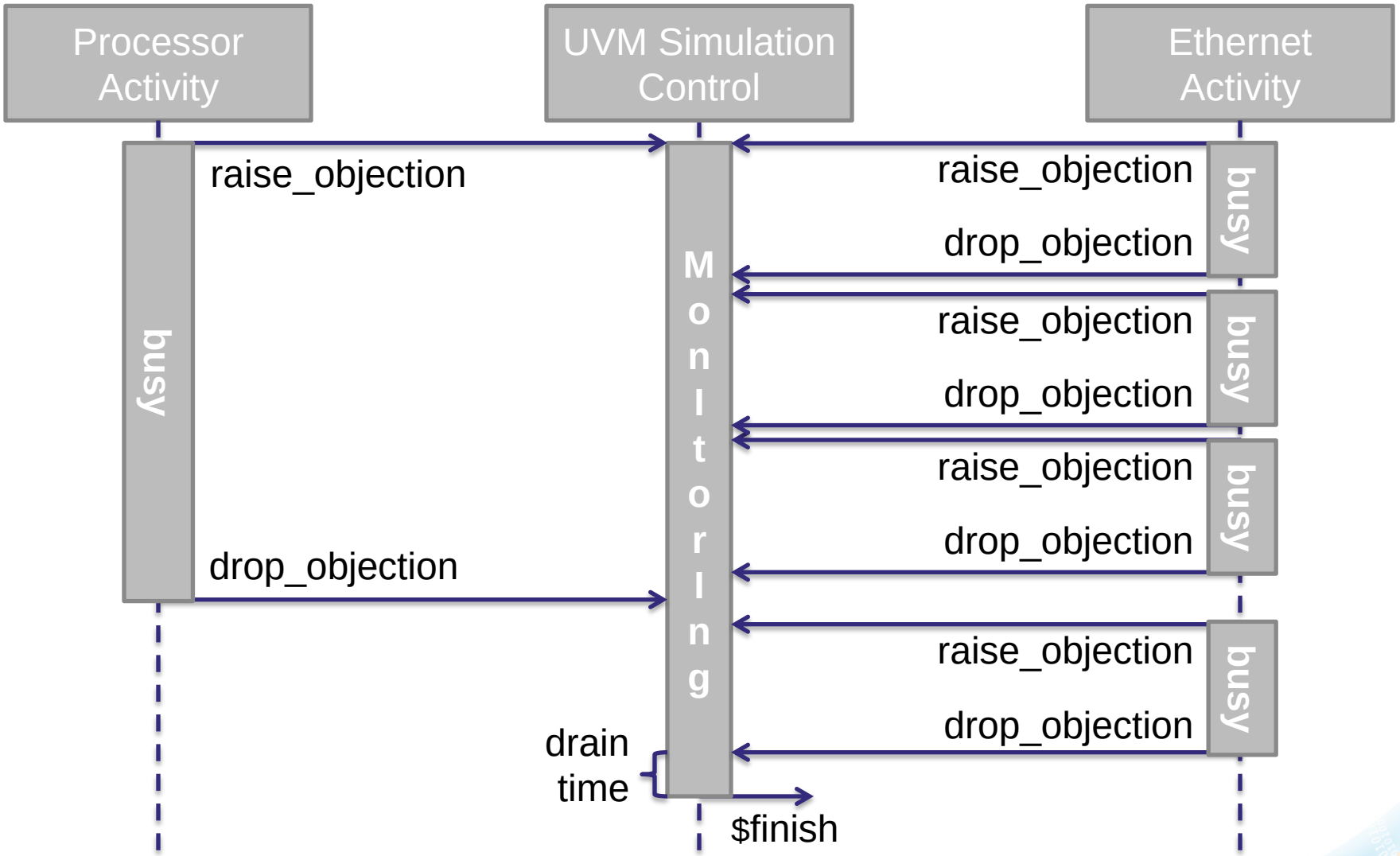
Measurements

Lessons Learned

Best Practices

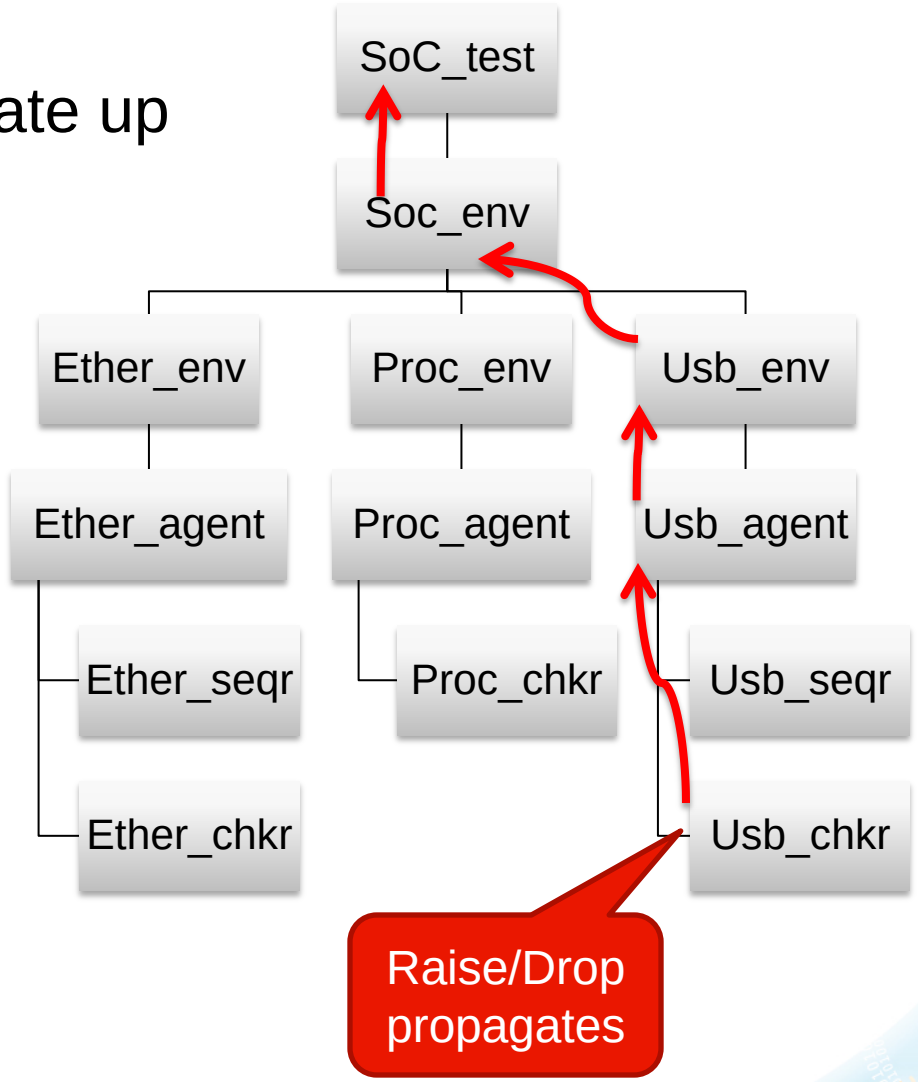
Conclusion

UVM Objections



Propagating Objections

- UVM 1.1 objections propagate up component hierarchy
  - Add to performance overhead
  - Needed for objection debug
- UVM 1.2 option to disable objection propagation
  - Applies to all phases



Turning off objection propagation

Read paper for download location

- Use phase_started component callback

```
function void phase_started(uvm_phase phase);
  uvm_task_phase task_based;
  // Is this a run-time (task based phase)?
  if ($cast(task_based,phase.get_imp())
  begin
    uvm_objection obj;
    obj = phase.get_objection();
    obj.set_propagate_mode(0); // UVM 1.2
    obj.set_drain_time(uvm_top,10ns);
  end
endfunction
```

Agenda

Introduction

Objections

Measurements

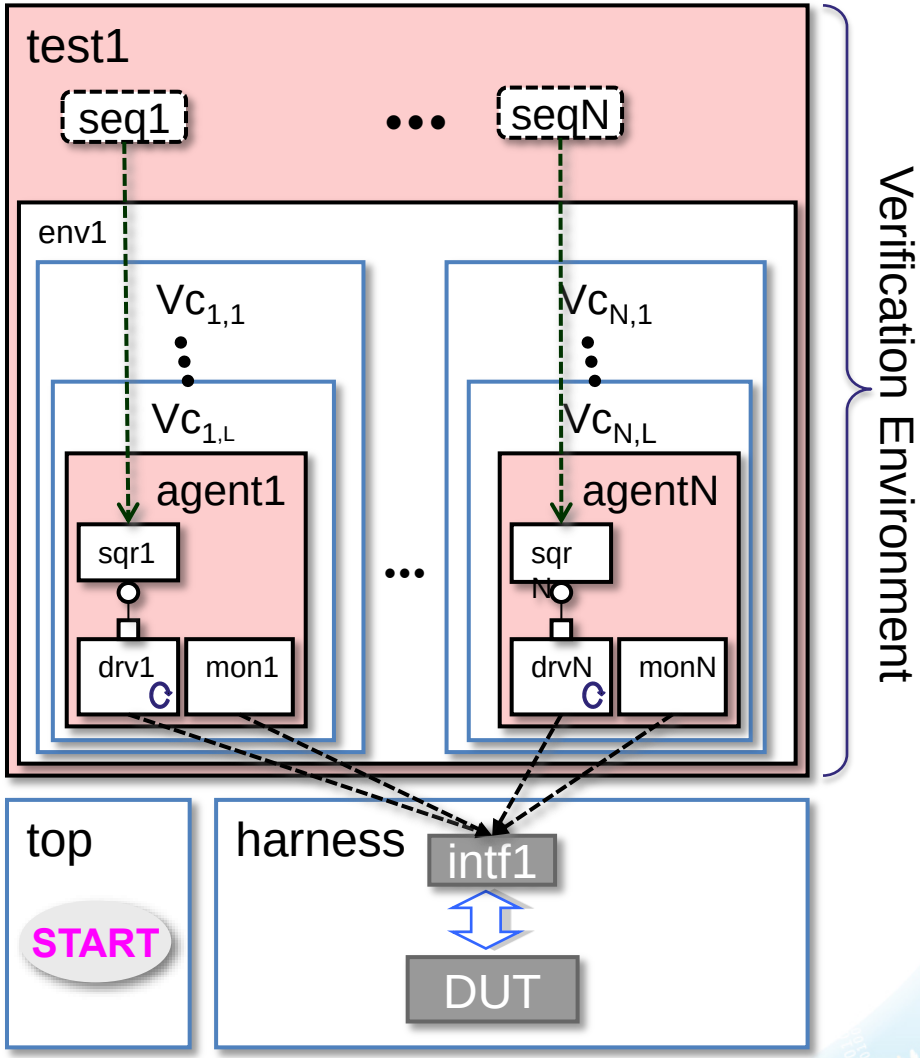
Lessons Learned

Best Practices

Conclusion

Measuring Simulation Performance

- Requirements
  - Scalable verification environment
  - Simplistic hardware design
  - Limit host simulation timestamps to before and after code of interest
  - Simulate a suitably large number of operations to ensure statistical significance
  - Agnostic script launches runs for many configurable options
    - UVM version, depth, propagation, etc…



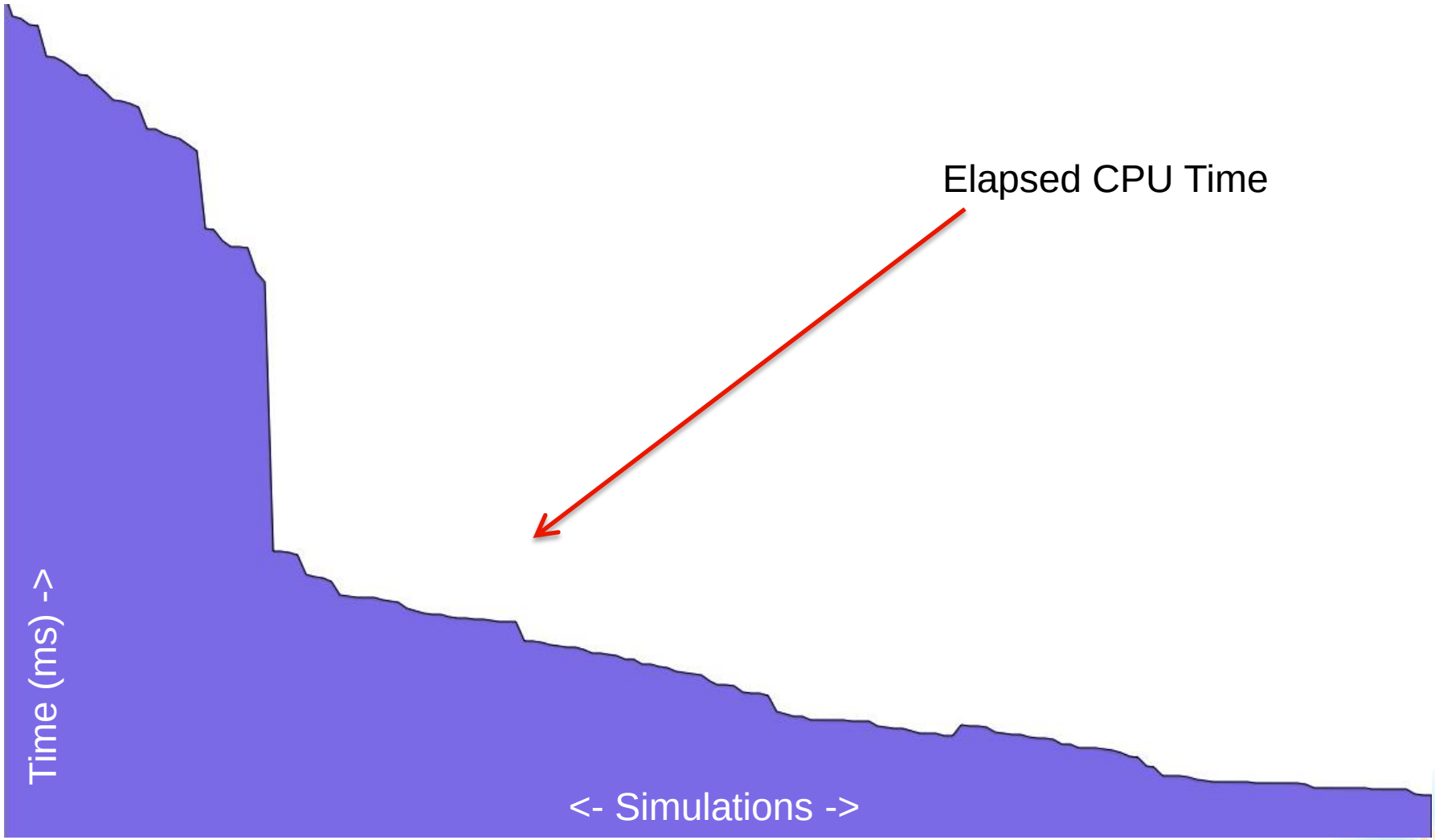
Measuring CPU time

Read paper for download location

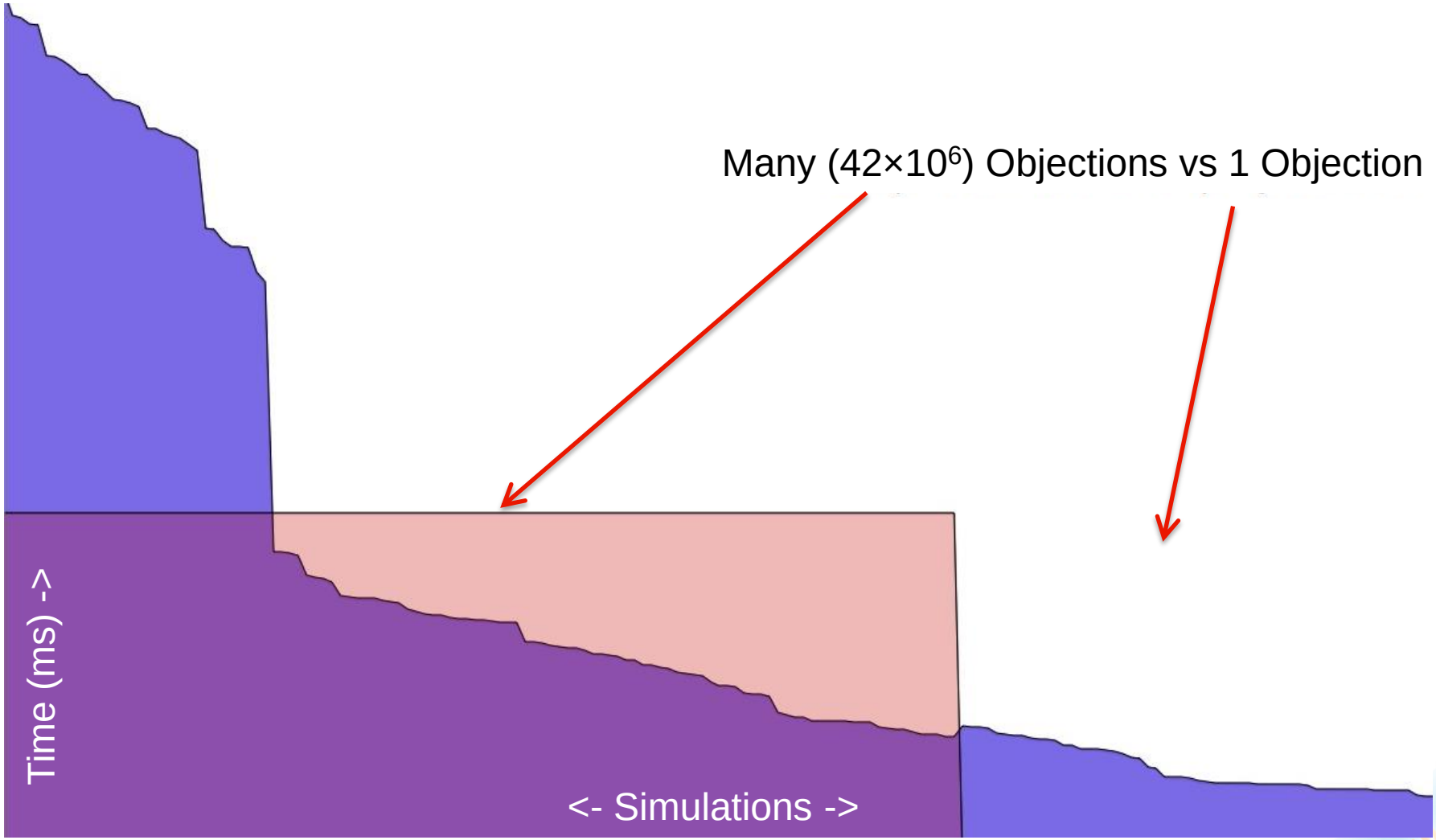
```
import "DPI-C" get_cpu_time          // elapsed
  = function real get_cpu_time(); // seconds
import "DPI-C" get_wall_time         // wall
  = function real get_wall_time();// seconds
```

```
#include "svdpi.h"
#ifndef WIN32
#include <time.h>
extern "C" double get_cpu_time(void) {
  timespec t;
  if (clock_gettime(CLOCK_PROCESS_CPUTIME_ID,&t))
    return 0; // Handle error
  return double(t.tv_sec)+double(t.tv_nsec) * 1.0e-9;
}
#endif
```

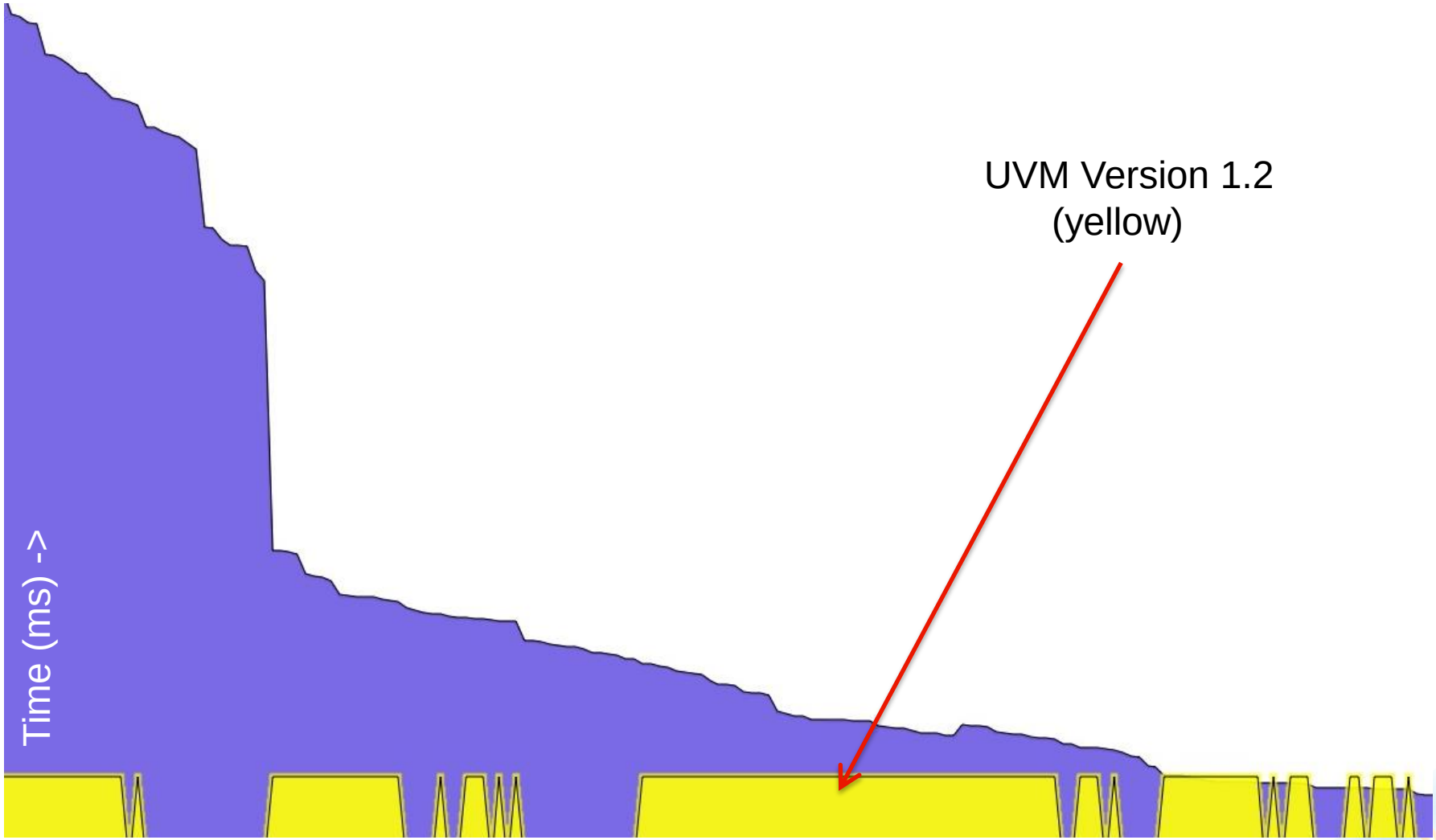
Raw graph



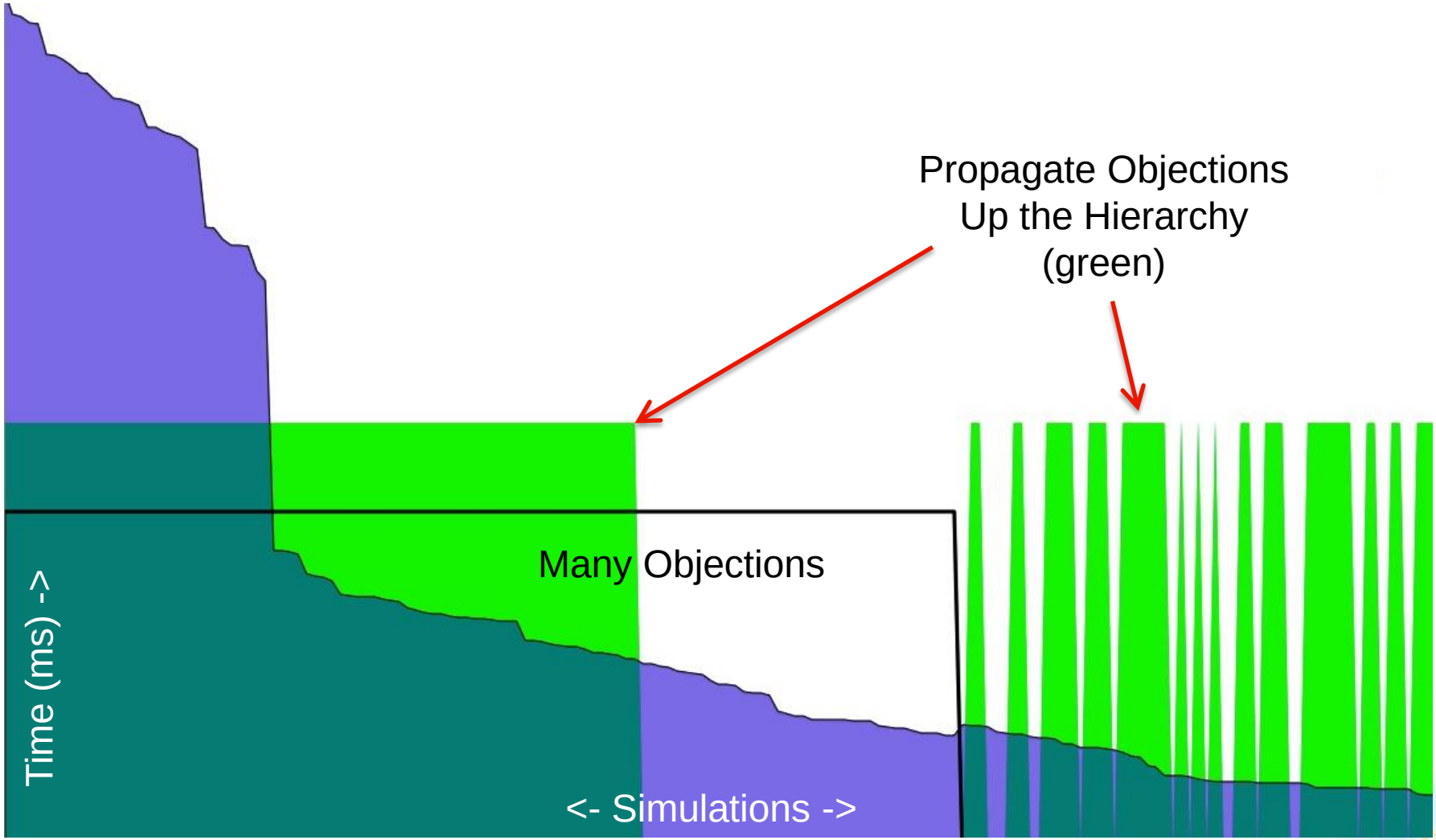
Objections 21,000,000:1



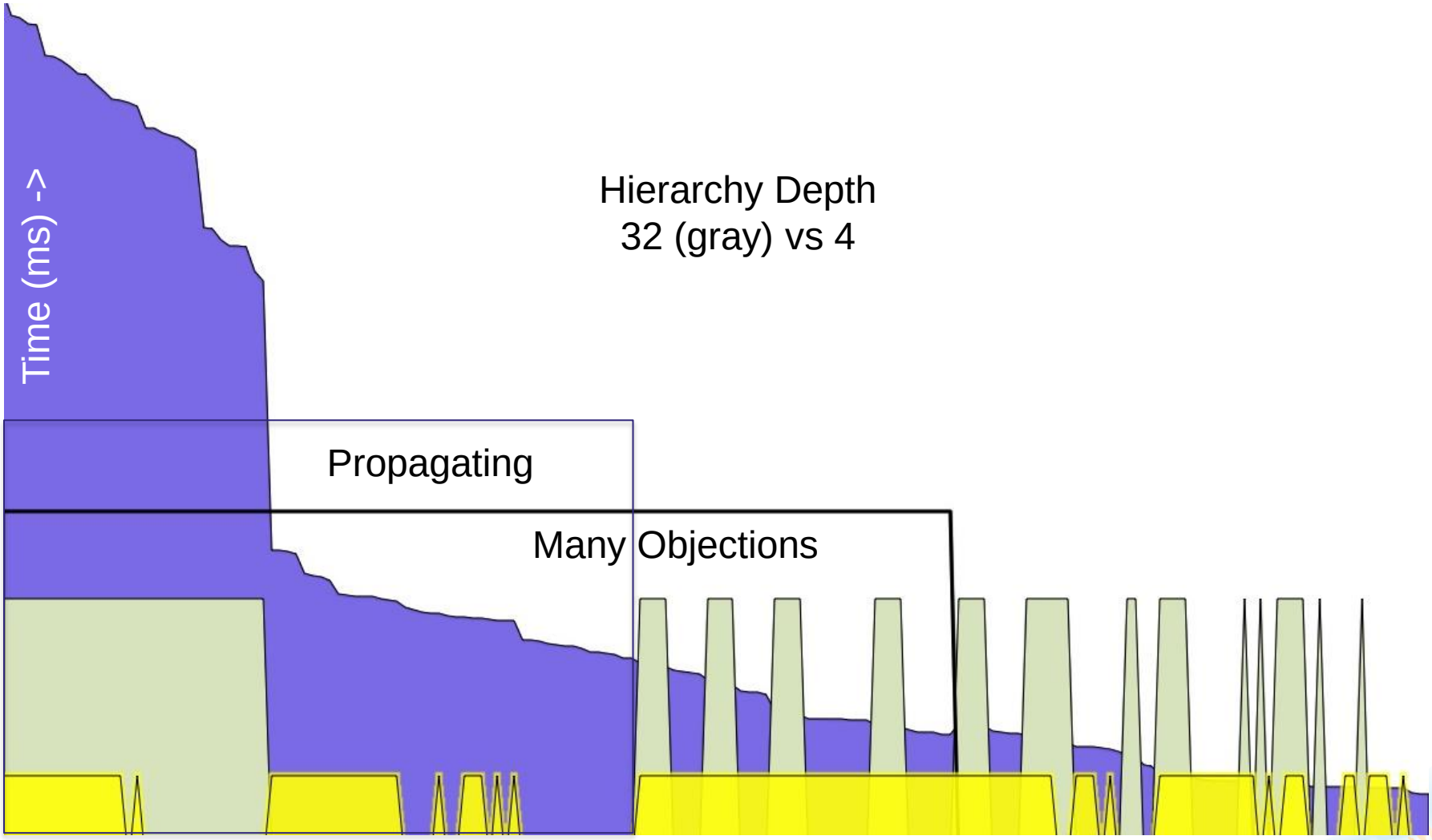
UVM 1.2 vs 1.1d



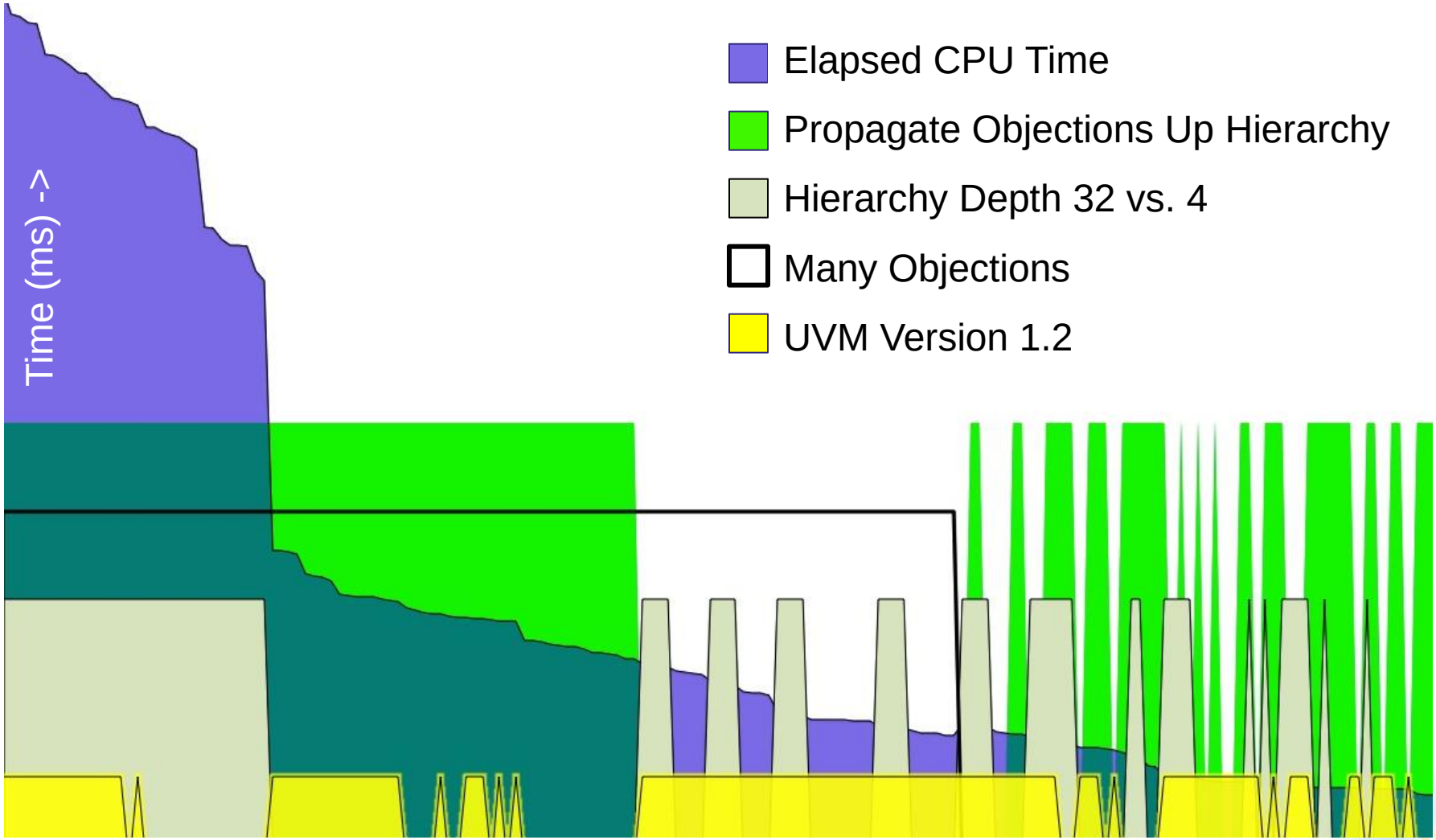
Effects of Objection Propagation



Effects of Deep Hierarchy



Combined graph



Agenda

Introduction

Objections

Measurements

Lessons Learned

Best Practices

Conclusion

Managing Backward Compatibility

- Use conditional compilation

```
`ifndef UVM_POST_VERSION_1_1
   // UVM 1.1 and earlier here
   // or if you don't support it:
   assert(0) else
     `uvm_fatal("","This code requires UVM 1.2 or later")
`else
   // UVM 1.2 and later here
`endif
```

Managing Phases

- Drain-time and propagation mode attach to task-based phases
  - E.g. main_phase and run_phase
- Drain-time and propagation mode need to be set before any objections are raised (i.e. at t=0)

  Either:
  - Ensure no objections at time zero
  - Use phase_started() to avoid a race condition at t=0

Agenda

Introduction

Objections

Measurements

Lessons Learned

Best Practices

Conclusion

Simulation Speed Considerations

- Turning off objection propagation clearly improves speed
- Reduce the total number of objections
  - top-level virtual sequences object once for their duration
- For last transaction, either
  - Mark end of transaction with **item_done**()
  - Use phase_ready_to_end() for last transaction
    - maximum invocation of 20
    - See figure ‘Extending the last transaction‘ in the paper

Best Practice

UVM 1.2 Adds Safety

- Locking the starting_phase data member during ‘use’ (read access to end of sequence)
- Error on raising objections in non-task (function) phases
- Consistent naming of enumerations with UVM_ prefix
- Passing enumerations from command-line by name
- Deprecated duplicate functionalities
  - get/set_config_int/string vs config_db#(int/string)::get/set
  - factory reference no longer available

Agenda

Introduction

Objections

Measurements

Lessons Learned

Best Practices

Conclusion

Take aways

- UVM 1.2 is a step forward
  - Safer
  - Mostly faster
  - Basis for IEEE standard
- Coding considerations w.r.t. backward compatibility
  - Less likely under IEEE
  - Use ifdef's where concerned
- Get educated
  - Learn the new features
  - Understand how to properly apply and use UVM

- Source code ➔ doulos.com/uvm

DOULOS

snug 25
Synopsys Users Group

Thank You