

NASCUG, DAC 2006

Patterns and Issues in SystemC Usage

John Aynsley, CTO, Doulos





Doulos and SystemC

- 6 years of experience with SystemC
- Delivered training in Europe, Scandinavia, Americas, India
- **Delivered training to over 100 companies from 180 customer sites**
- Authors of the OSCI SystemC Language Reference Manual
- **Authors of IEEE 1666™**
- Members of OSCI Language-WG, TLM-WG and Verification-WG



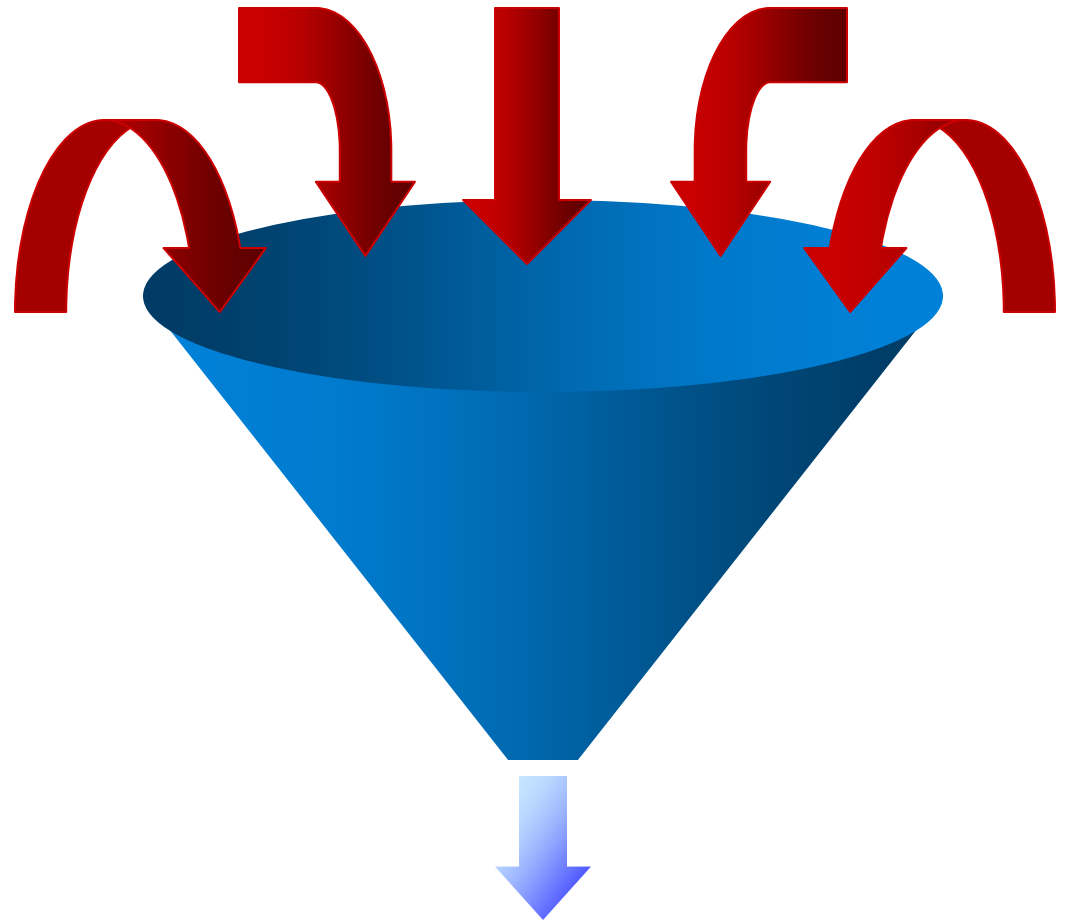
Market Surveys

- Doulos survey - DAC 2003
- <http://www.celoxica.com/designersurvey/reg.asp>
- http://www.esl-now.com/pdfs/survey_results.pdf
- http://www.mentor.com/products/c-based_design/eu_esl_survey.cfm

Distillation...



Many specific experiences and use cases



Here follows our analysis and vision...

Agenda

CONTENTS

Reasons for using TLM and SystemC

Languages, models and skills

Usage patterns

Abstraction and timing

Overview



Technical issue

Reasons for using TLM

Firmware /
software

Fast enough

TLM

Ready before RTL

RTL

Test bench

Accelerates product release schedule

Software development



Architectural modeling



Hardware verification

TLM = golden model

Reasons for using SystemC

Open source C++ simulator

Flexibility re platforms and licensing

No vendor lock-in

Easy integration

Industry standard IEEE 1666™

Common language across disciplines

Builds bridges between system, s/w and h/w

Better reference models, because above RTL



Drivers for Adoption

Key drivers are time-to-market and quality-of-verification

Driven by those responsible for modeling and verification

Testbench re-use is #1 reason to spend \$\$\$

Design productivity/synthesis is *not* a huge motivation

Hardware engineers least likely to drive adoption

But C/C++/SystemC synthesis is attractive and growing



“Glue” or “Lingua Franca”

Because SystemC is C++, is open, is “free”

SystemC wrappers

around C / C++ / HDL models

around proprietary models

automatically generated

Models

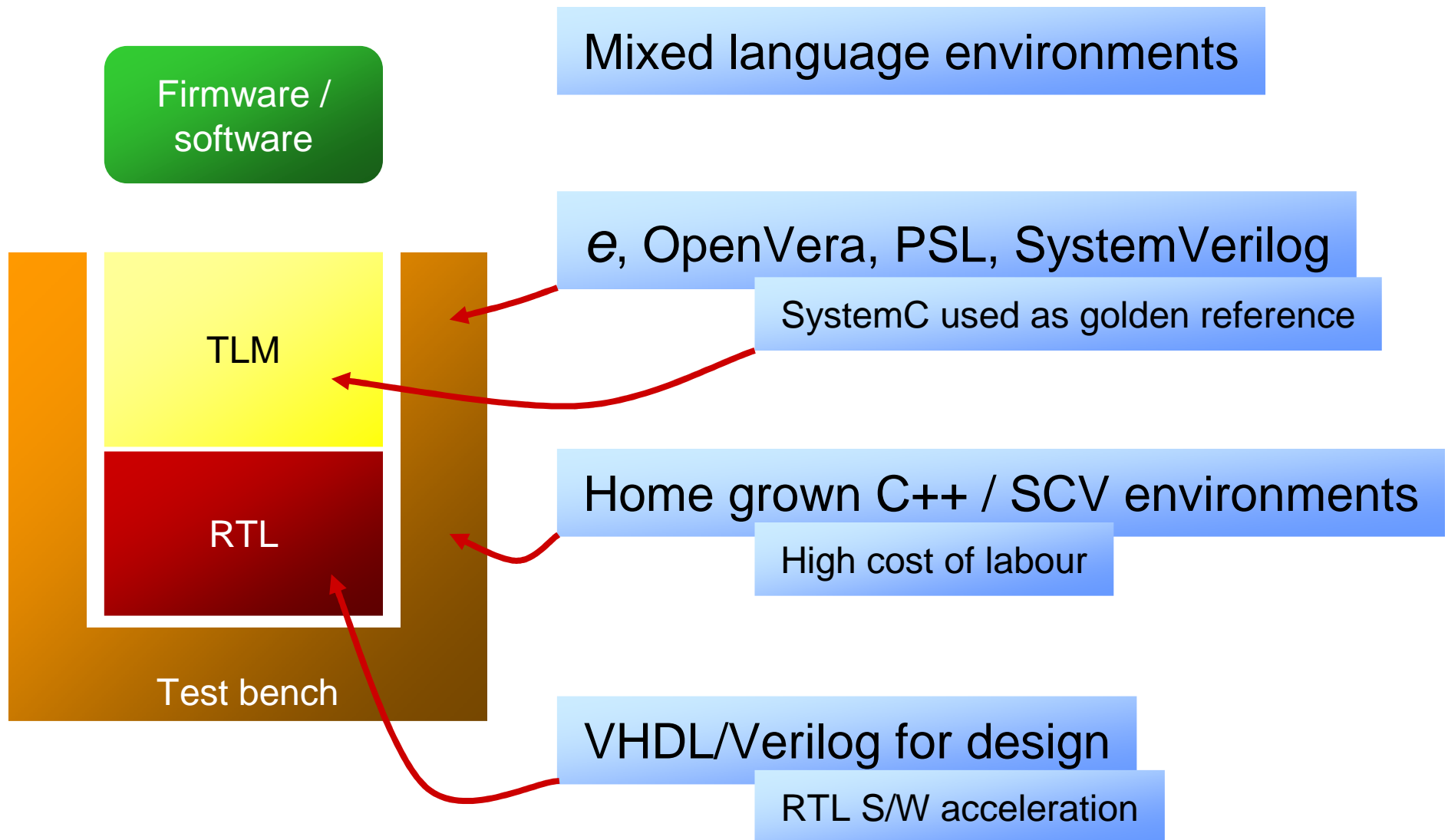
SystemC glue

stick concurrency on top of C++

stick on a Tcl or Python interpreter

stick on public C++ libraries, e.g. Boost

Languages





versus SystemVerilog

SystemC is C++, SystemVerilog is an HDVL

SystemC and SystemVerilog usage increasing

at the expense of VHDL / Verilog

SystemC leads SystemVerilog

SystemVerilog growing faster



C++ Skills

Most users choose to become skilled in C++ / OO

Some companies successful with libraries and “kits”

Hide-the-details methodologies

Still requires C++ expertise

Growing use of external consultants



Pattern of Adoption

Either

Corporate strategy imposed by central group

EDA vendor methodology or home-grown

“Kit” to hide details

Hard to sell

Individual project use by clever C++ programmers

Struggle to fill holes, e.g. functional+code coverage, assertions

Needs a big investment in methodology and process

Need for standard TLM interfaces and coding style

Adoption and Abstraction

Legacy projects

Low level dependencies force CA modeling

Simulation speed severely limited

Points to need for a clean top-down system model

1st generation SystemC usage is CA or BCA

2nd generation is untimed / PV

easy to interface

3rd generation will be CX / PVT ?

Not much use outside methodology groups / vendors



Confusion over Role of TLM

Implementation-level component models

require too much accuracy for TLM

because use cases are not known in advance

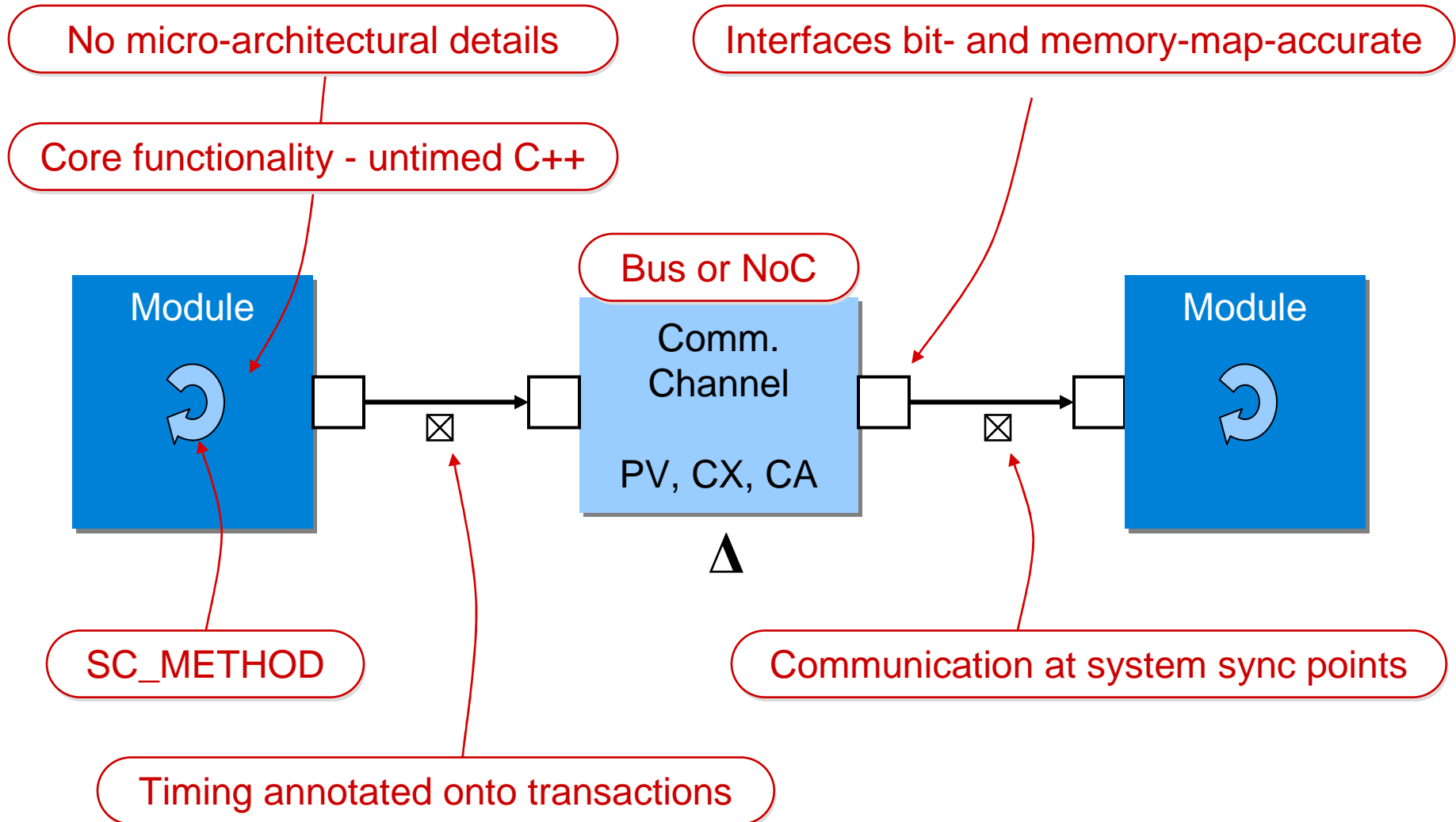
Companies give up on TLM when...

too inaccurate, too slow, too far from silicon

Sweet spot for TLM is top-down flow

Abstract models used before making implementation choices

Usage Pattern for More Speed





Conclusion

Use SystemC if...

you want to get this year's software working with next year's silicon

you want a golden model you can re-use from architecture to RTL verification

More info: john.aynsley@doulos.com