Welcome to the Accellera SystemVerilog Workshop

June 2, 2003

DAC 2003

Agenda

Introduction:
SystemVerilog Motivation
Vassilios Gerousis, Infineon Technologies
Accellera Technical Committee Chair

Session 1:
SystemVerilog for Design

Language Tutorial
Johny Srouji, Intel

User Experience
Matt Maidment, Intel

Session 2:
SystemVerilog for Verification

Language Tutorial
Tom Fitzpatrick, Synopsys

User Experience
Faisal Haque, Verification Central

Lunch: 12:15 – 1:00pm

Session 3: SystemVerilog Assertions

Language Tutorial
Bassam Tabbara, Novas Software

Technology and User Experience
Alon Flaisher, Intel

Using SystemVerilog Assertions and Testbench Together
Jon Michelson, Verification Central

Session 4: SystemVerilog APIs
Doug Warmke, Model Technology

Session 5: SystemVerilog Momentum

Verilog2001 to SystemVerilog
Stuart Sutherland, Sutherland HDL

SystemVerilog Industry Support
Vassilios Gerousis, Infineon

End: 5:00pm

Agenda

Introduction:
SystemVerilog Motivation
Vassilios Gerousis, Infineon Technologies
Accellera Technical Committee Chair

Session 1:
SystemVerilog for Design

Language Tutorial
Johny Srouji, Intel

User Experience
Matt Maidment, Intel

Session 2:
SystemVerilog for Verification

Language Tutorial
Tom Fitzpatrick, Synopsys

User Experience
Faisal Haque, Verification Central

Lunch: 12:15 – 1:00pm

Session 3: SystemVerilog Assertions

Language Tutorial
Bassam Tabbara, Novas Software

Technology and User Experience
Alon Flaisher, Intel

Using SystemVerilog Assertions and Testbench Together
Jon Michelson, Verification Central

Session 4: SystemVerilog APIs
Doug Warmke, Model Technology

Session 5: SystemVerilog Momentum

Verilog2001 to SystemVerilog
Stuart Sutherland, Sutherland HDL

SystemVerilog Industry Support
Vassilios Gerousis, Infineon

End: 5:00pm

Introduction to SystemVerilog: History, Motivation and Process

Vassilios Gerousis, Infineon Technologies

Accellera Technical Chairman

Accellera SystemVerilog Committee Chairman

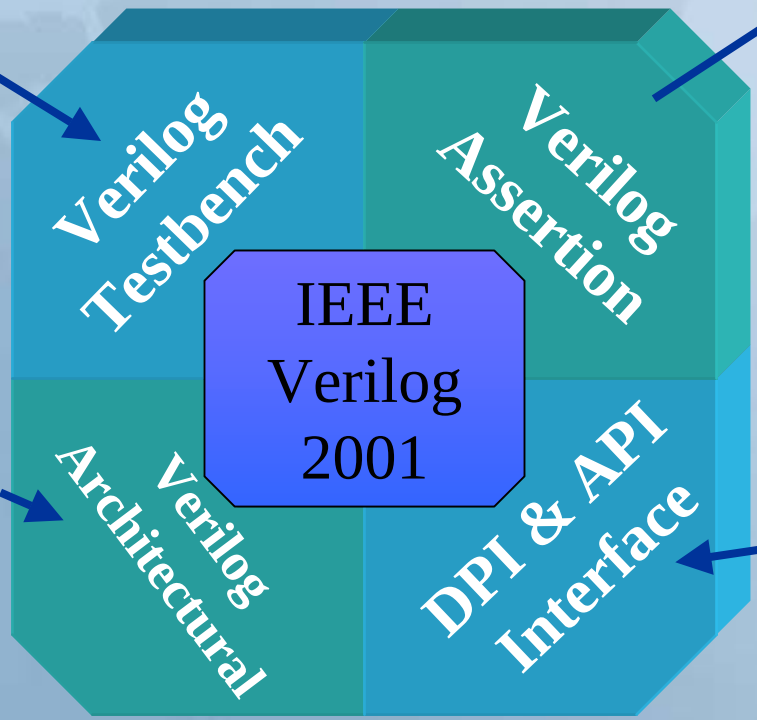
Session 1 Outline

- **History of SystemVerilog.**
- Verification Gap
- Components of HDVL
- Methodologies Of SystemVerilog – The HDVL of Nanometer design.

SystemVerilog Charter

- **Charter: Extend Verilog IEEE 2001 to higher abstraction levels for Architectural and Algorithmic *Design* , and *Advanced Verification*.**

Transaction-Level Full Testbench Language with Coverage

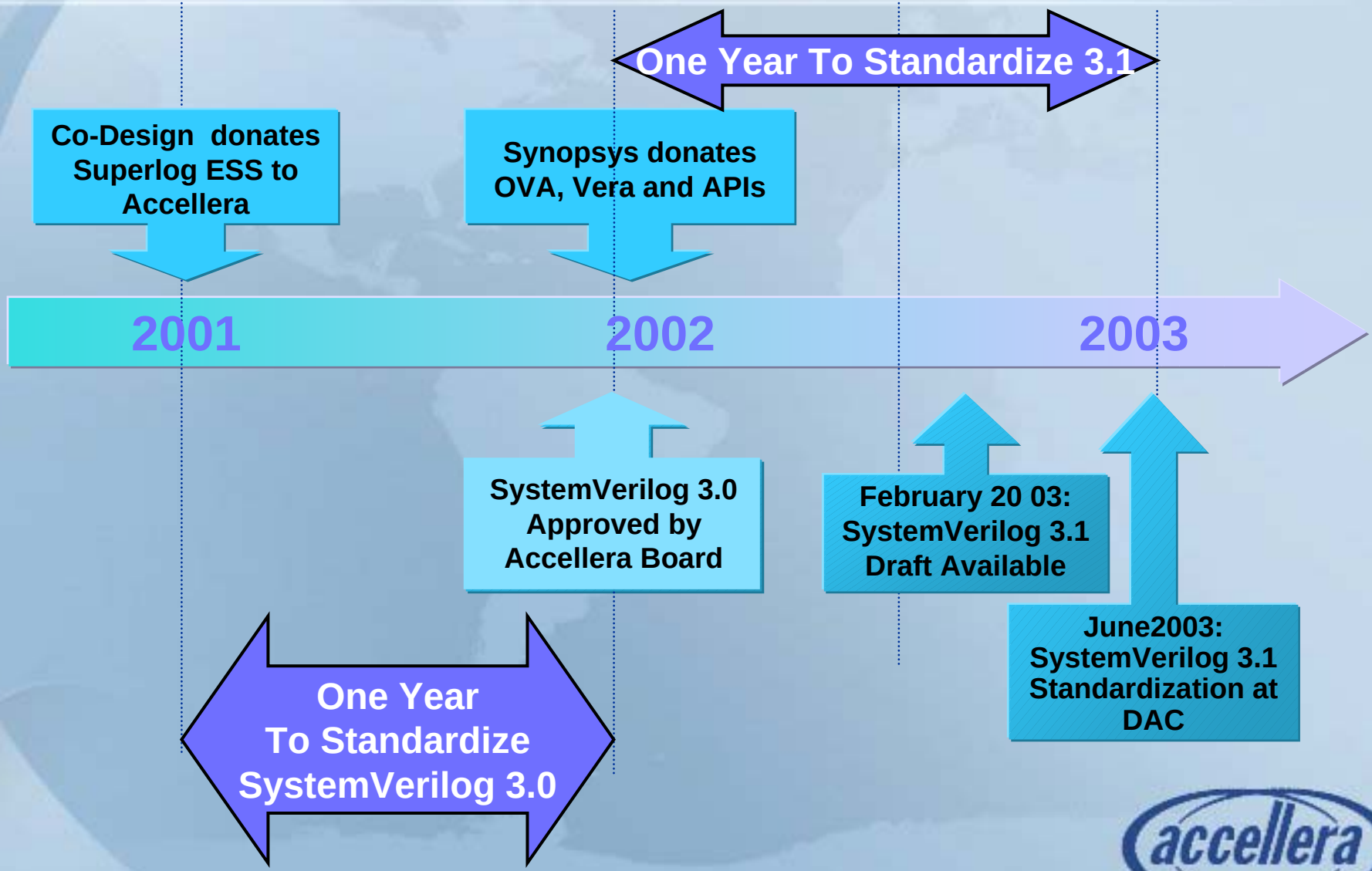


Advanced verification capability for semiformal and formal methods. The Assertion Language Standard For Verilog

Design Abstraction: Interface semantics, abstract data types, abstract operators and expressions

Direct C interface, Assertion API and Coverage API

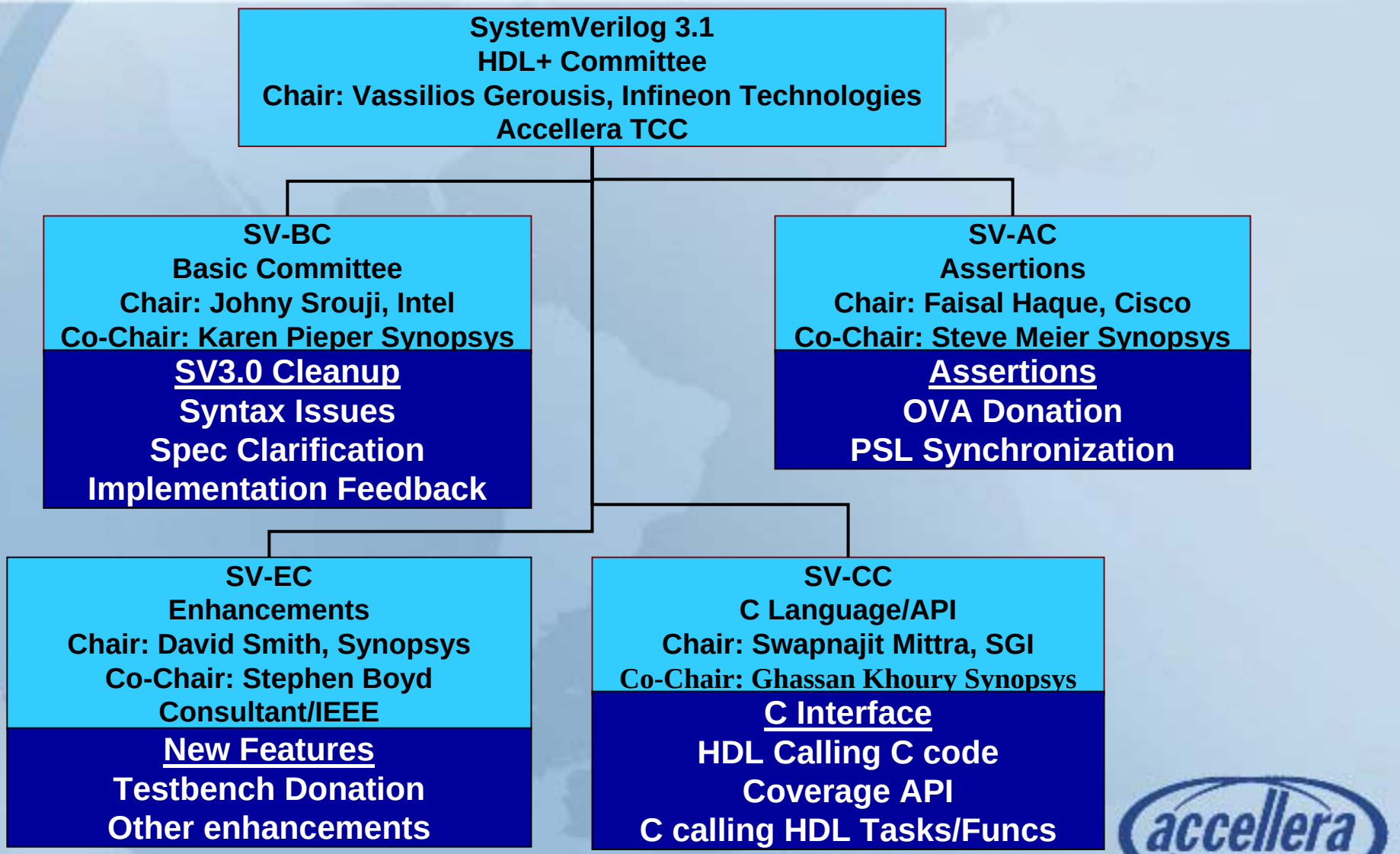
SystemVerilog Standardization Timeline



Developed By 40+ Verilog and Verification Experts

SystemVerilog 3.1 Basic Committee	SystemVerilog 3.1 Testbench Committee	SystemVerilog 3.1 Assertion Committee	SystemVerilog 3.1 C-interface Committee
Kevin Cameron Cliff Cummings* Dan Jacobi Jay Lawrence Matt Maidment Francoise Martinolle* Karen Pieper* Brad Pierce David Rich Steven Sharp* Johny Srouji Gord Vreugdenhil* * IEEE Verilog Member	Stefen Boyd* Dennis Brophy Michael Burns Kevin Cameron Cliff Cummings* Tom Fitzpatrick* Peter Flake Jeff Freedman Neil Korpusik Jay Lawrence Francoise artinolle* Don Mills Mehdi Mohtashemi Phil Moorby Karen Pieper* Brad Pierce Arturo Salz David Smith Stuart Sutherland*	Roy Armoni Surrendra Dudani Cindy Eisner Tom Fitzpatrick* Harry Foster Faisal Haque John Havlicek Richard Ho Adam Krolnik* David Lacey Joseph Lu Erich Marschner Steve Meier Prakash Narain Andrew Seawright Bassam Tabbara	John Amouroux Kevin Cameron Joao Gaeda Ghassan Khoory Andrzej Litwiniuk Francoise Martinole* Swapnajit Mittra Michael Rohleder John Stickley Stuart Swan Bassam Tabbara Kurt Takara Doug Warmke

Accellera SystemVerilog 3.1 Organization

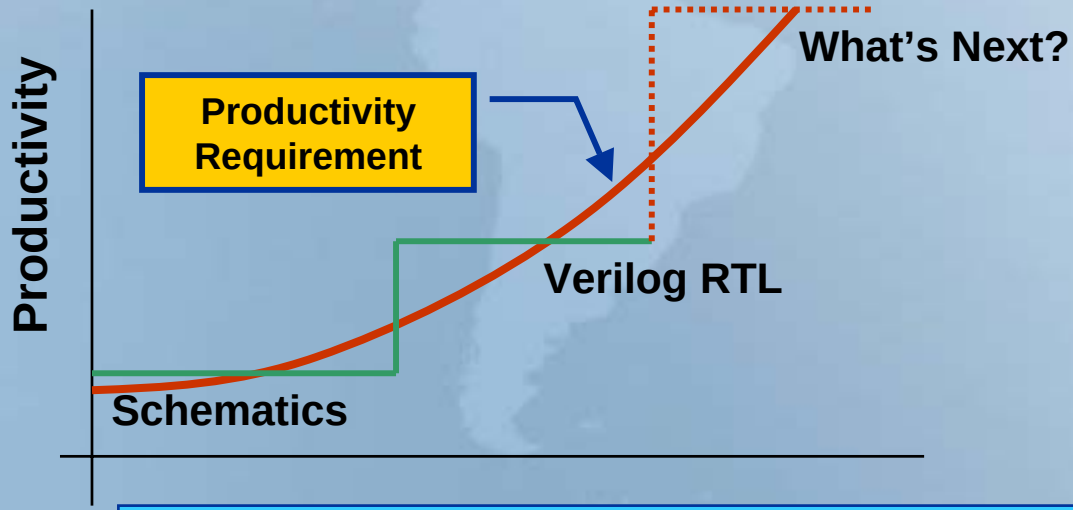


Session 1 Outline

- History of SystemVerilog.
- **Verification Gap**
- Components of HDVL
- Methodologies Of SystemVerilog – The HDVL of Nanometer design.

History is Repeating Itself

- Today's Complex Designs Are Getting Too Big For Verilog to Keep Up
  - 100's of pages of design code not uncommon
  - 2X – 3X (2000's pages) of Testbench
  - More code means more bugs

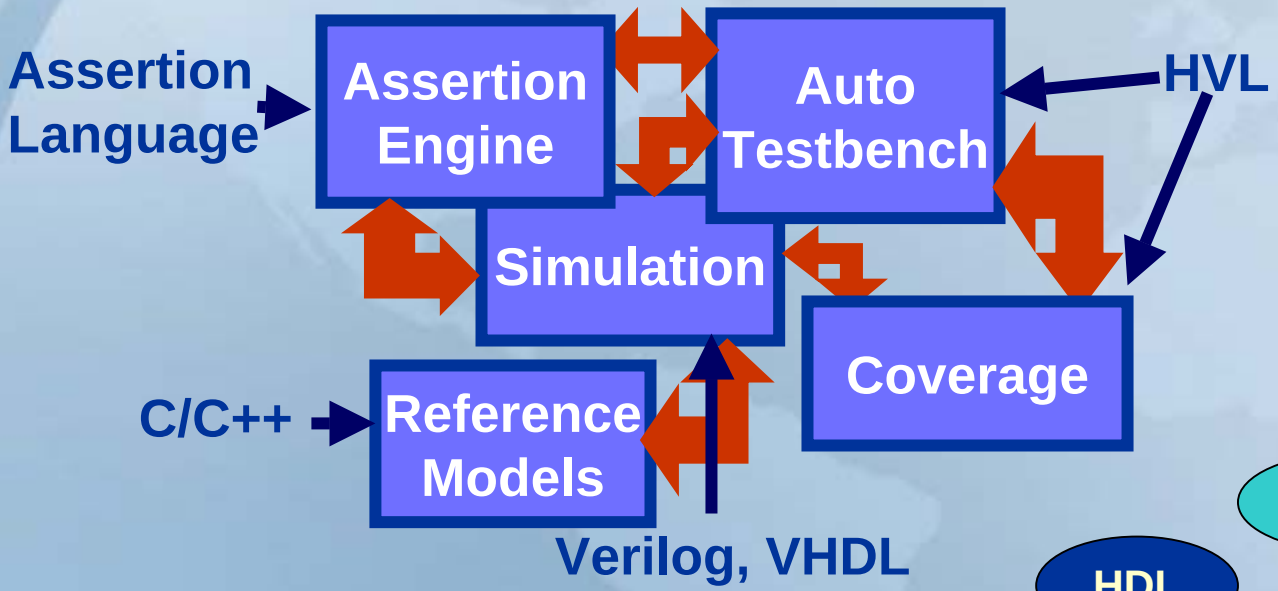


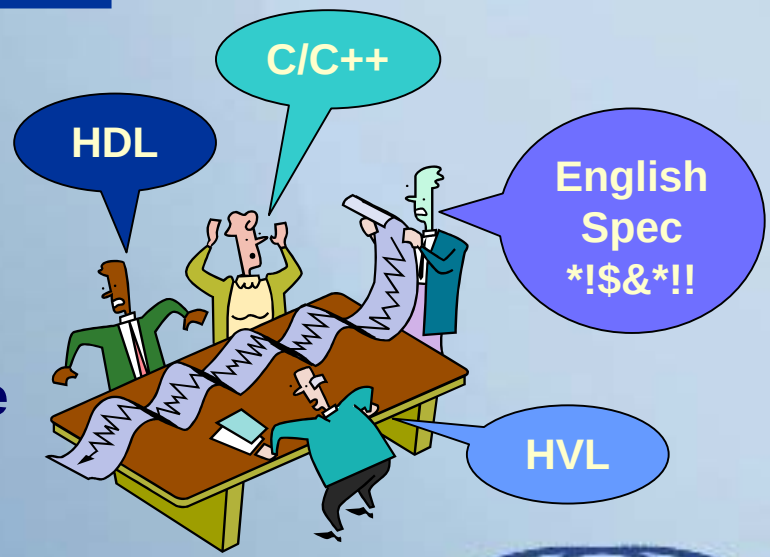
Verilog RTL Today is Where Schematics Were 15 Years Ago

The Verification GAP

- Verification is 60% to 80% of design cycle time.
  - Testbench is about 2X to 5X the RTL code.
  - Formal tools can be applied on block level.
  - Block-based testbench cannot be used at the system integration level. (bottom-up verification).
  - Top down verification is not practical in current tools and languages.
- Design Implementation has been automated to a certain degree (synthesis).
  - RTL synthesis provides biggest productivity.
  - Behavioral synthesis has not been as successful so far.
- Verification: No one has been successful to automate verification to provide similar productivity as synthesis.

Inefficient Multi-lingual RTL Flows





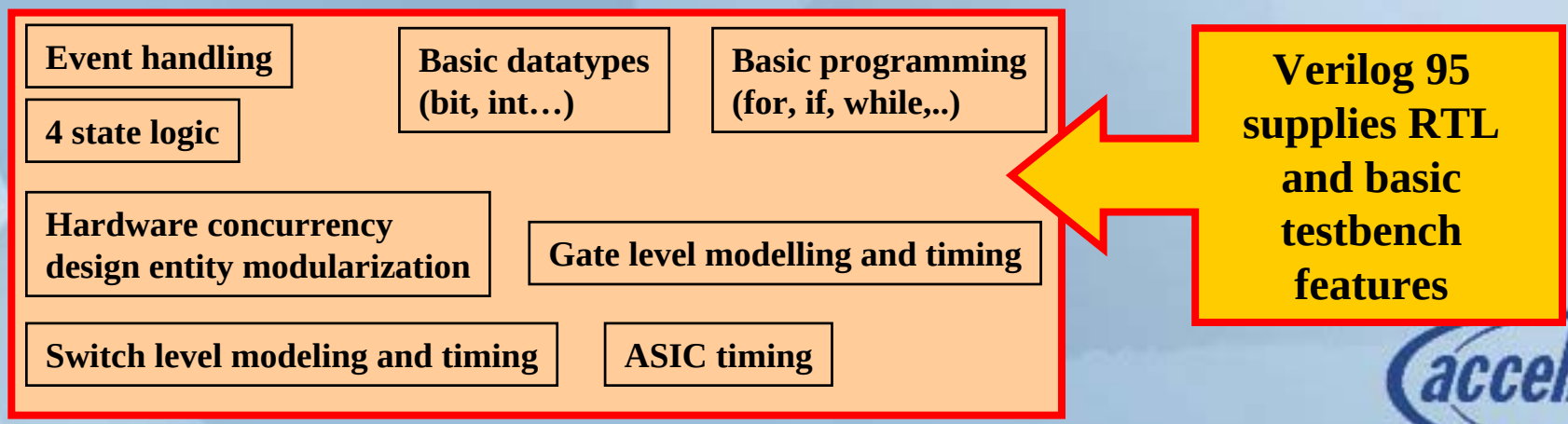
Multiple input formats:

- Complicated to learn and maintain
- Limits communication cooperation
- Inhibits tool / methodology performance
- Kills productivity
- Inefficient maintenance and Reuse
- Limits tools and methodology innovation

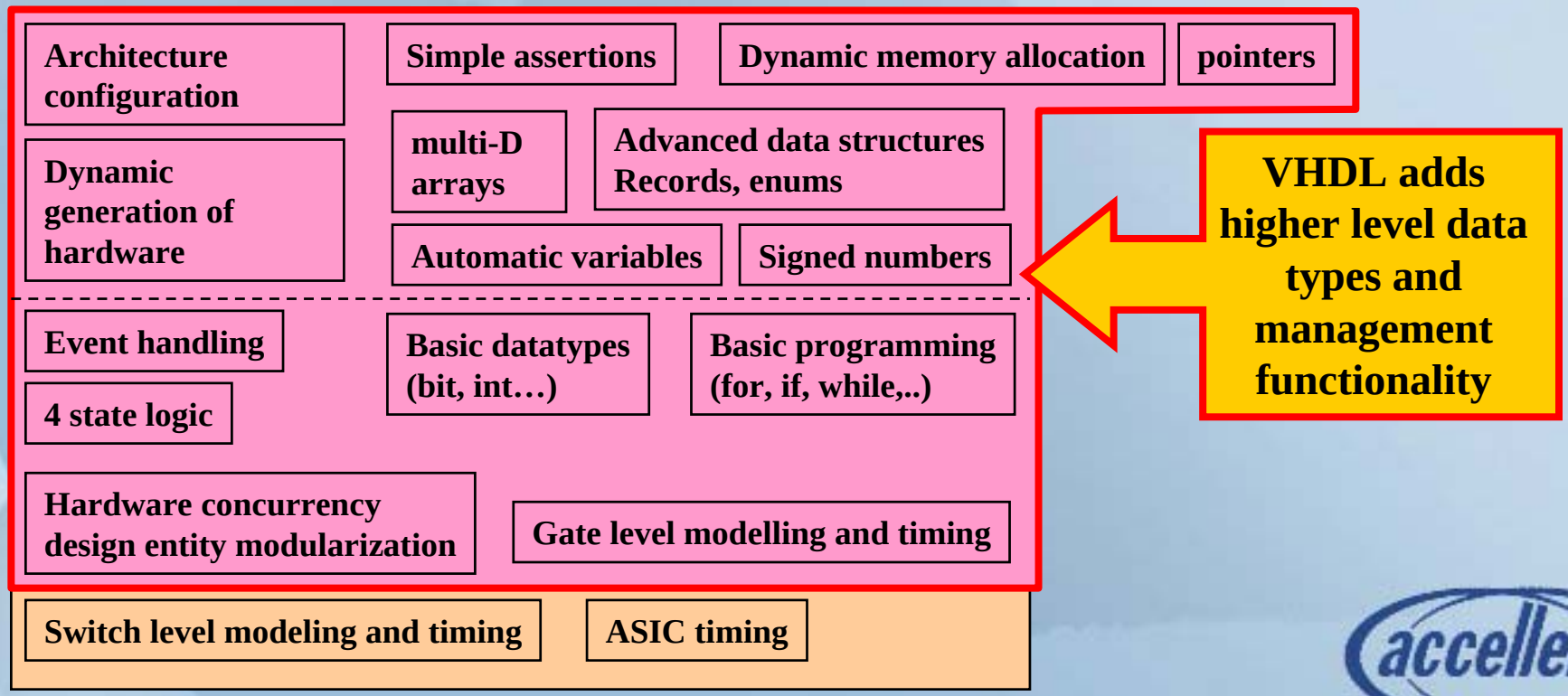
Session 1 Outline

- History of SystemVerilog.
- Verification Gap
- **Components of HDVL**
- Methodologies Of SystemVerilog – The Only HDVL of Nanometer design

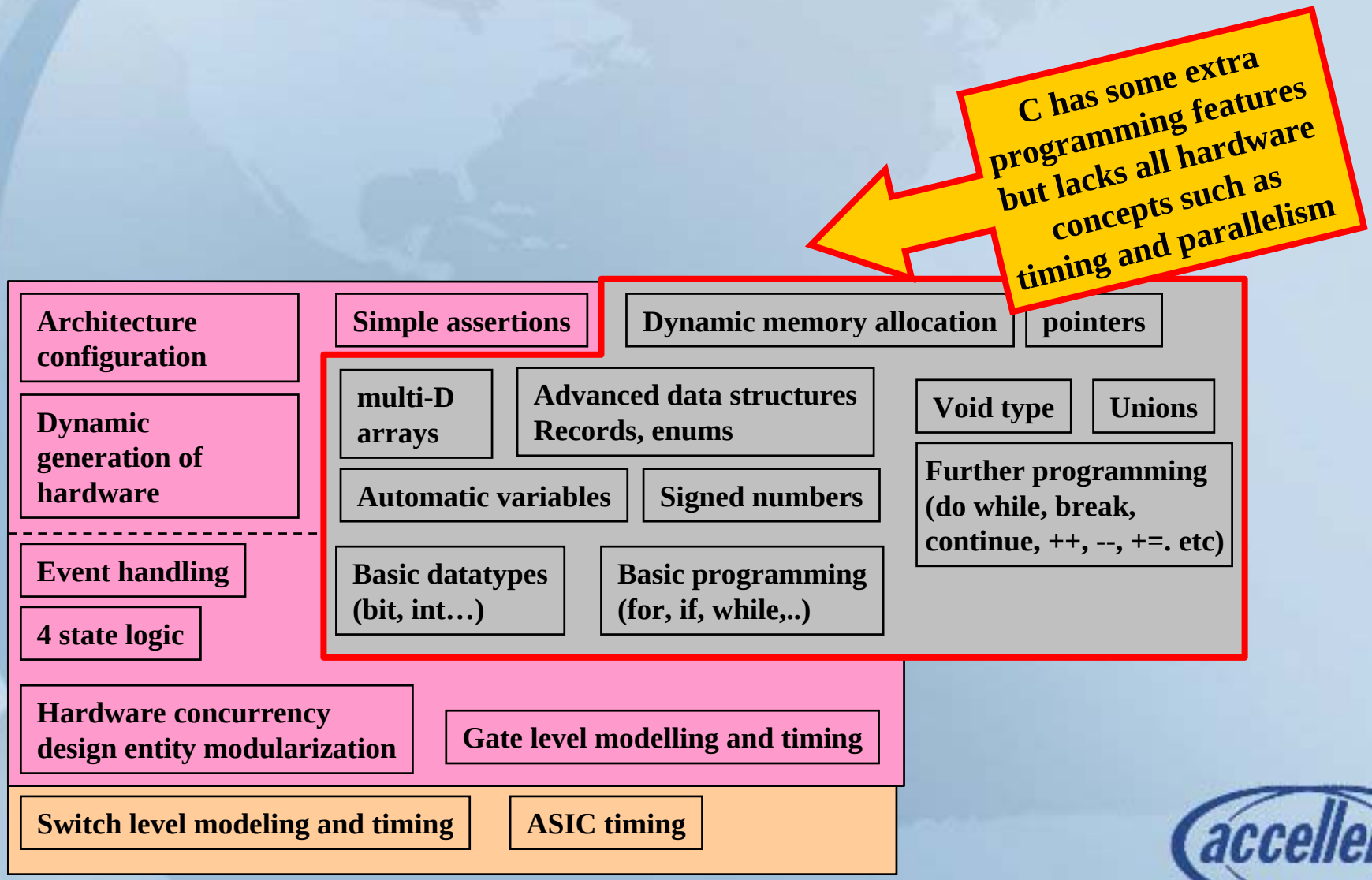
Semantic Concepts: Verilog 1995



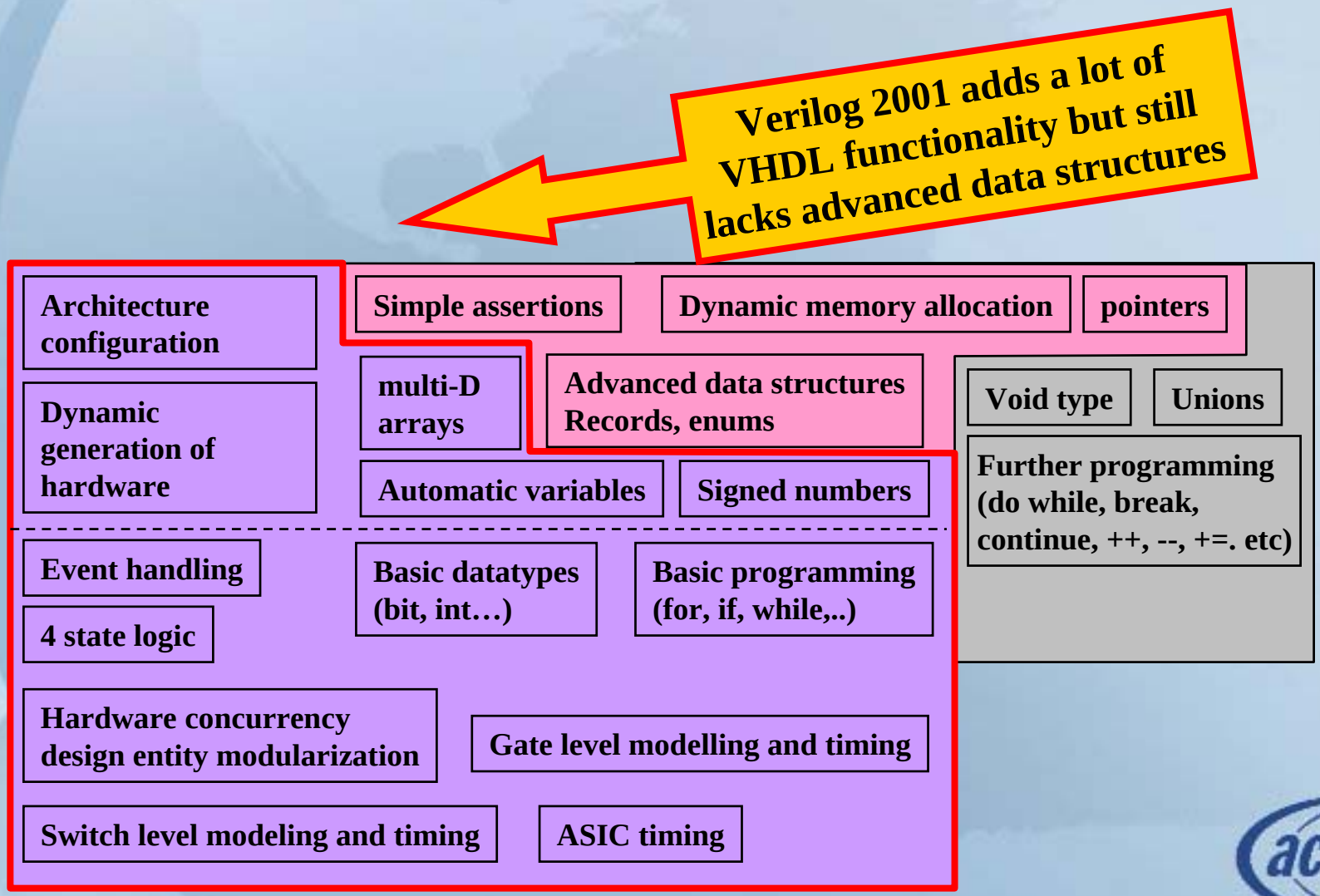
Semantic Concepts: VHDL



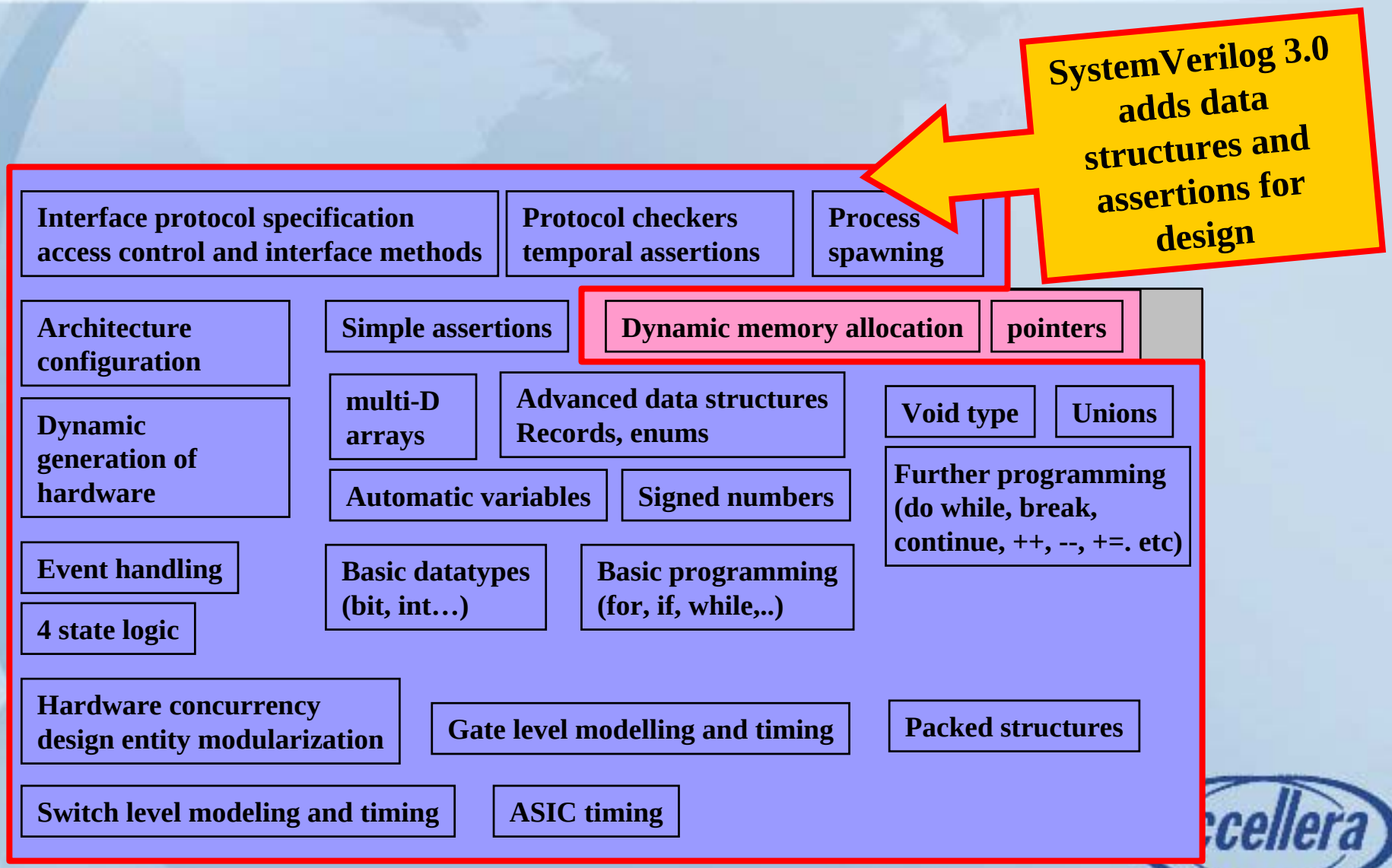
Semantic Concepts: C



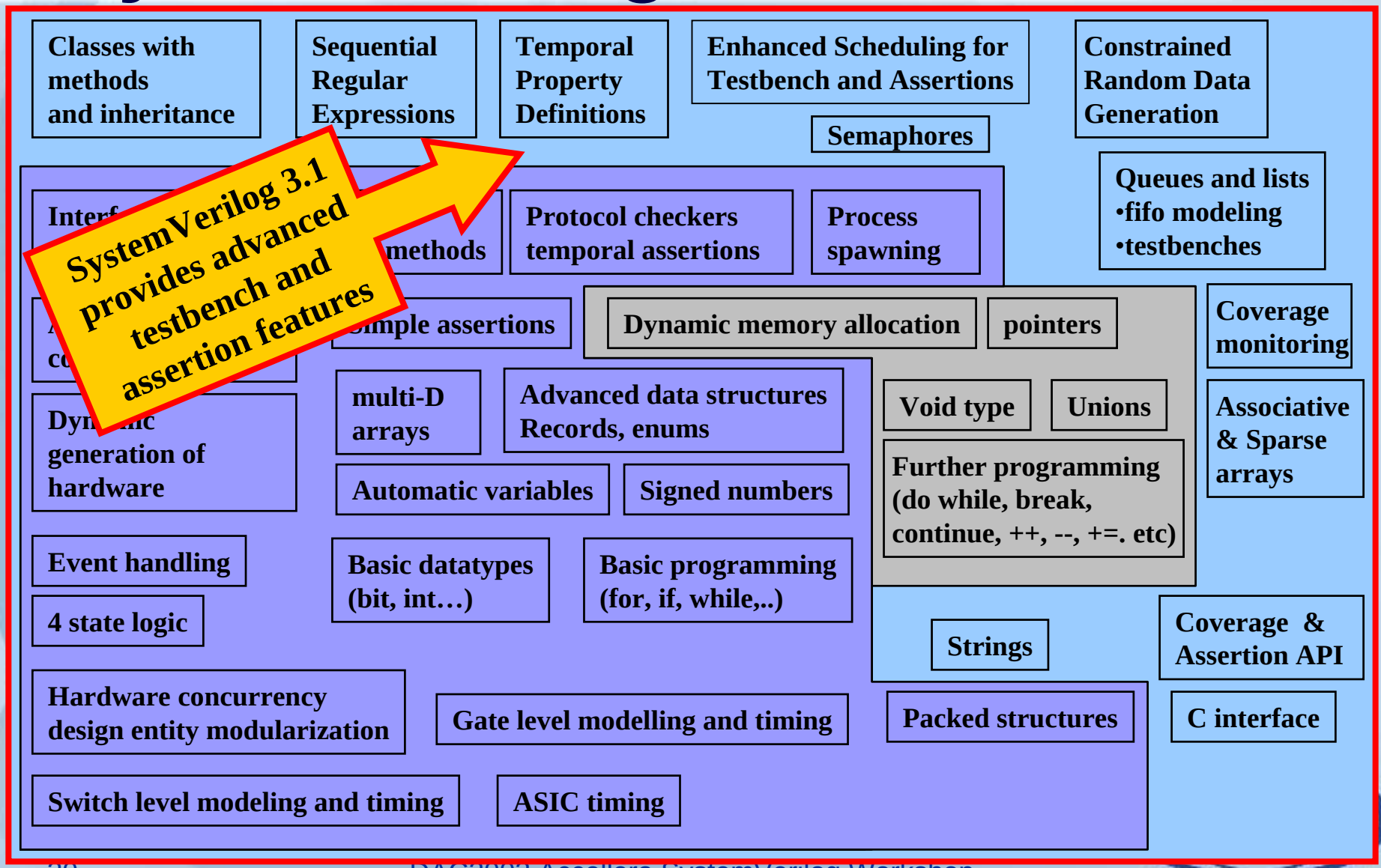
Semantic Concepts: Verilog 2001



Semantic Concepts: SystemVerilog 3.0 Accellera Standard – June 2002



Semantic Concepts: SystemVerilog 3.1

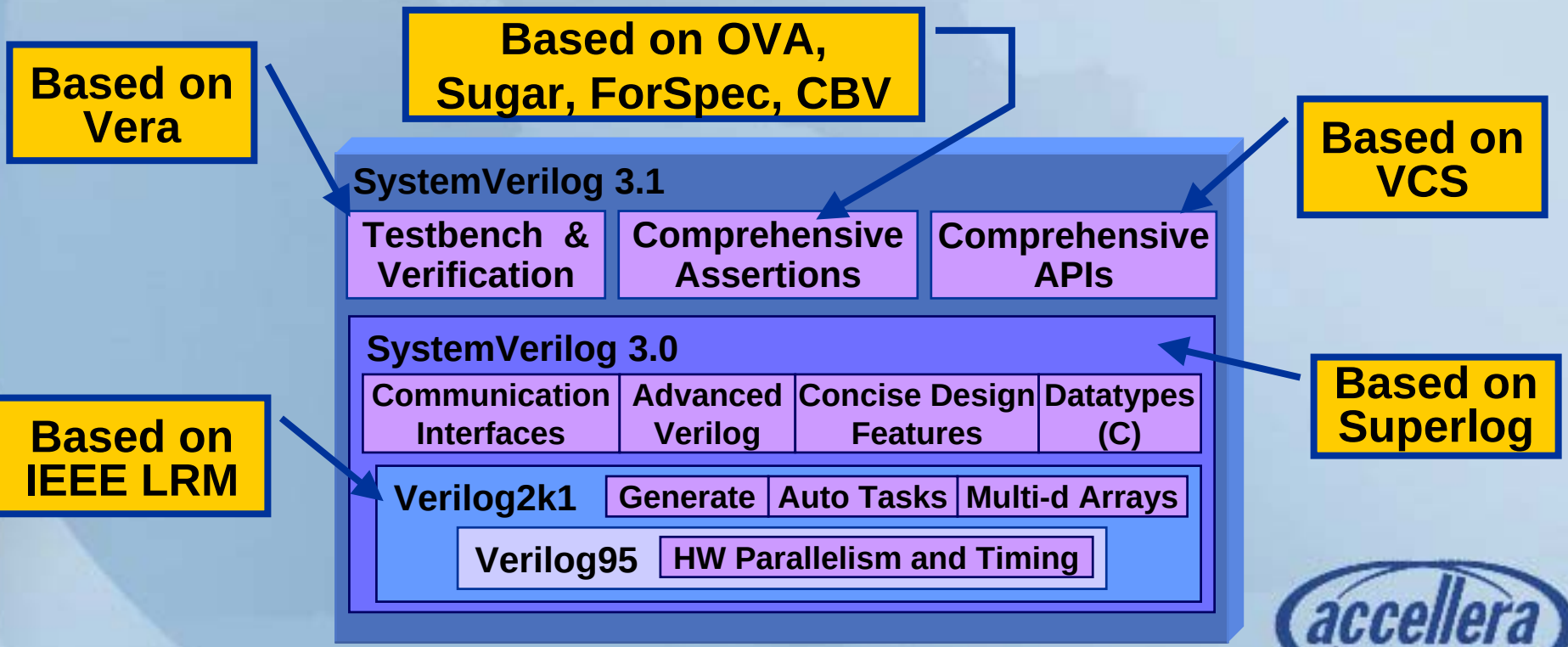


SystemVerilog Messages - Benefits

Accellera Standard, Next Generation Verilog, HDVL Language

- Unifying design verification – simplifies flow, team work
- Speeds operations – concise code = fewer bugs, quicker usage
- Evolution from Verilog – easy to learn and incrementally adopt

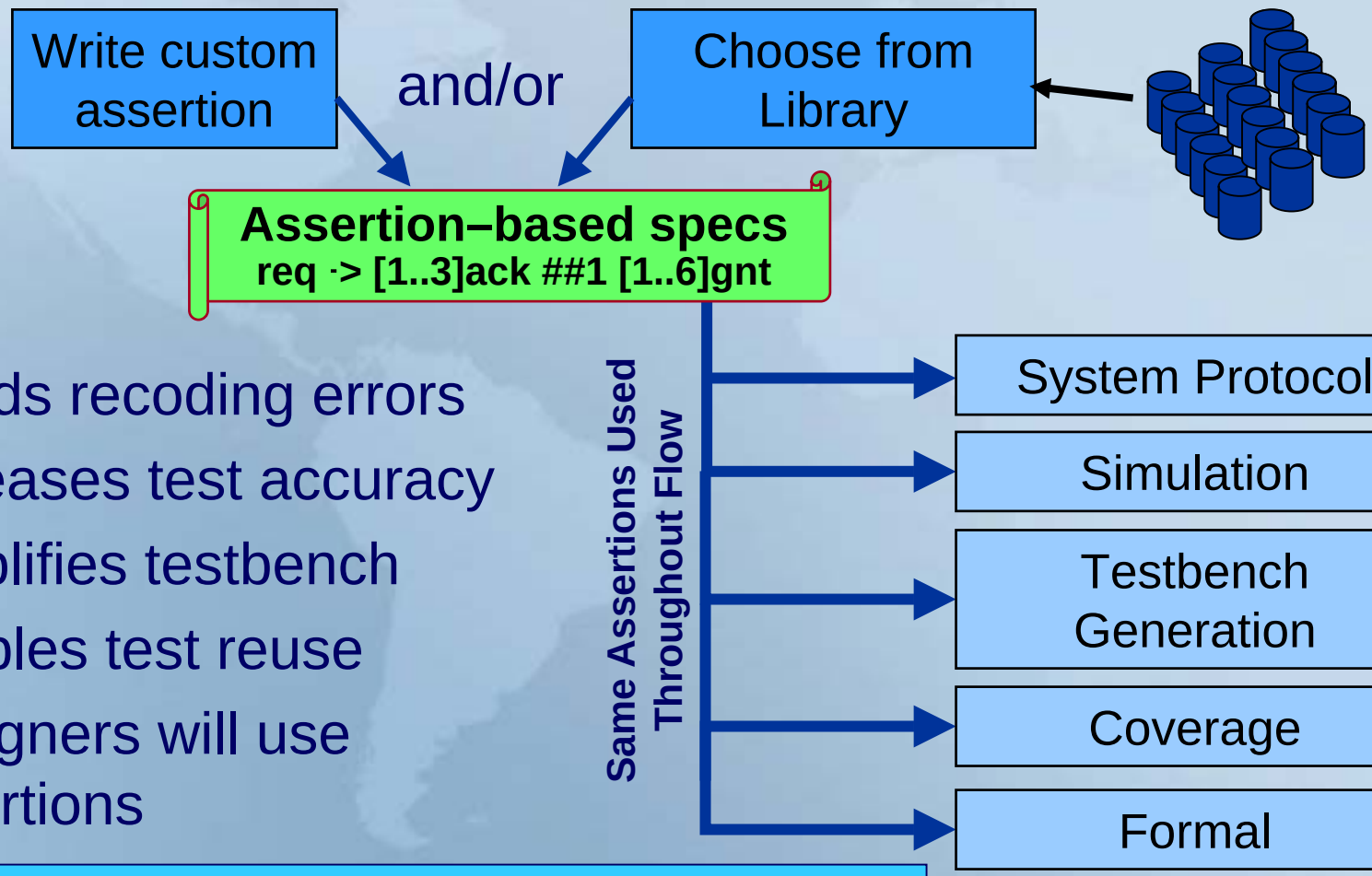
Based on technologies proven by EDA vendors and users



Session 1 Outline

- History of SystemVerilog.
- Verification Gap
- Components of HDVL
- **Methodologies Of SystemVerilog – The HDVL of Nanometer design**
  - Automated Testbench
  - Verification IP
  - Assertion Speeds up verification.
  - Platform design and HW/SW design.

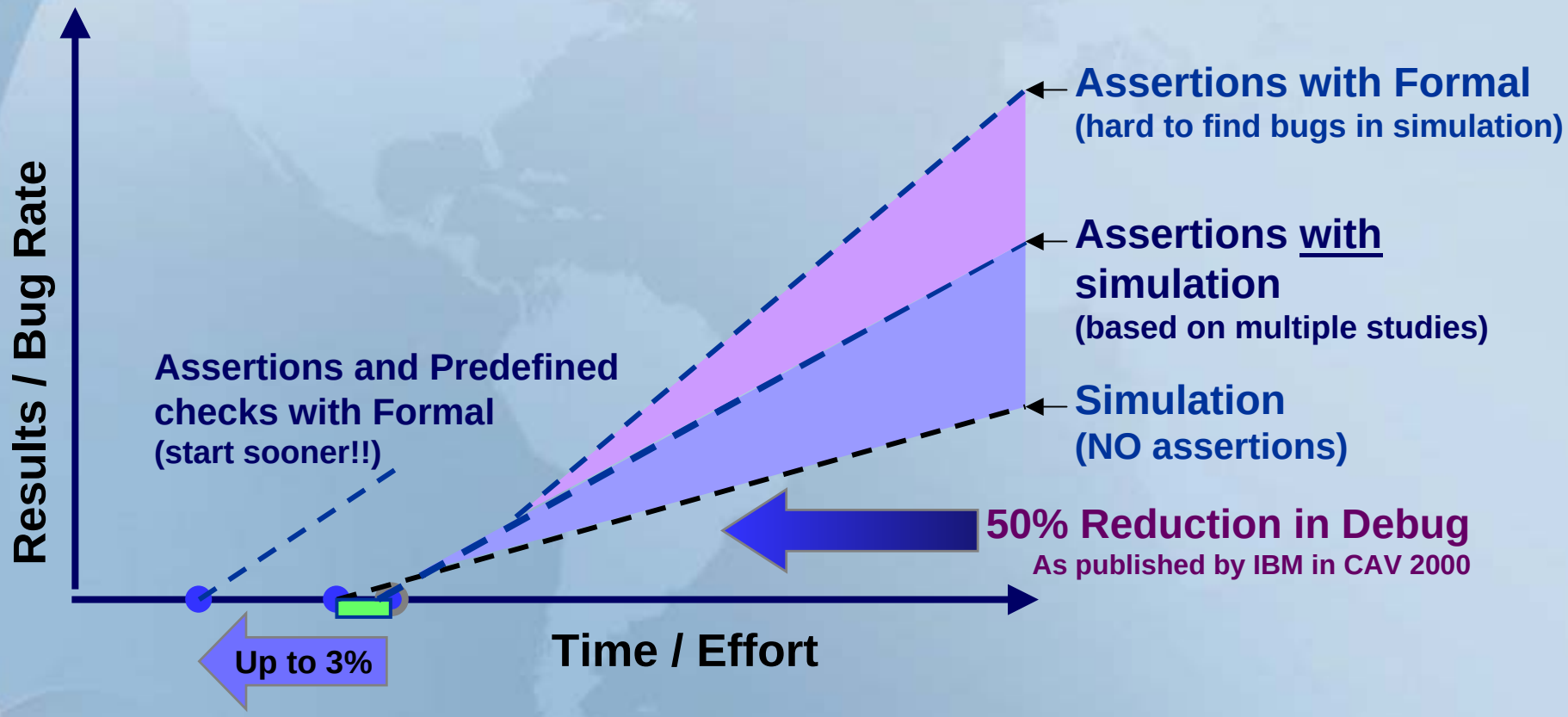
SystemVerilog Assertion-Based Verification



- Avoids recoding errors
- Increases test accuracy
- Simplifies testbench
- Enables test reuse
- Designers will use assertions

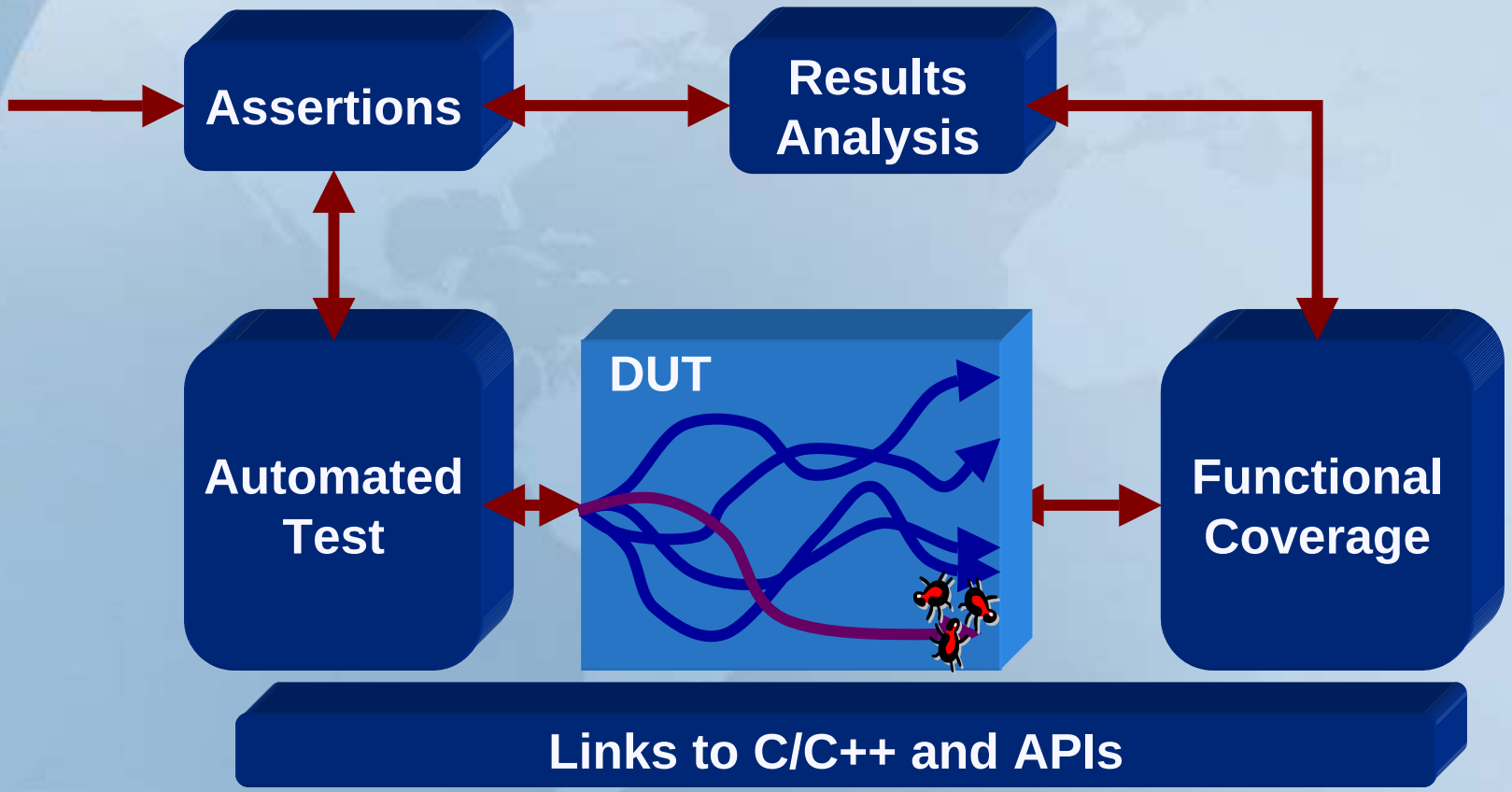
Assertions as part of HDVL Allow Specification to Drive Verification

Assertion Effort Payback



Source: Harry Foster (FVTC Chair)

SystemVerilog 3.1 Offers The Best of Testbench and Assertions



Simplify Verification

IP Packaging

- Support IP to include testbench and assertions
- Create independent module or interface with testbench and assertions to allow
  - VIP creation independent of implementation
  - Mixed language simulation
  - Formal and simulation verification
- Construct parameterizable VIP

Verification IP Packaging Example

```
interface busP
```

ASSERTIONS

```
sequence read_t;
  @(posedge clk) req ##[1:4] grant;
endsequence
 t1: assert property (read_t ##[1] !rd_r[*8]);
 c1: cover property (req;[4] ack ##1 grant);
 …
```

TESTBENCH

```
program test (…)
 …
 default clocking @(posedge clk);endclocking
 …
 gen_trans(par1);
 check_response(pkt2);
 …
endprogram
```

Protocol Verification IP (Testbench + Assertions)

Instantiate *protocol* as master and slave

```
busP bus_master(…,clock);
busP bus_slave(…,clock);
```

SystemVerilog Enables Platform Implementation

SystemVerilog can be used for the entire platform, or provide transparent, verifiable transaction-to-signal linkage