Assertion-Based Verification

Harry D. Foster
Chief Scientist Verification
IC Verification Solutions Division

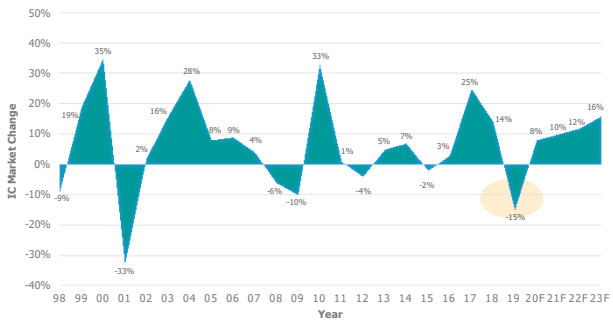
February 2020

Mentor
A Siemens Business

Outline

- Industry Pulse
- How Verification is Done Today
- What Makes Verification Difficult
- Observability and Controllability Challenge
- Assertion-Based Verification
- Industry Case Studies
- Summary

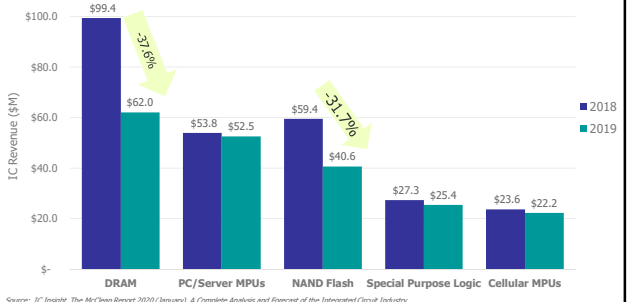
Worldwide IC Market Growth



Source: IC Insight, The McClean Report 2020 (January), A Complete Analysis and Forecast of the Integrated Circuit Industry

Largest IC Product Categories

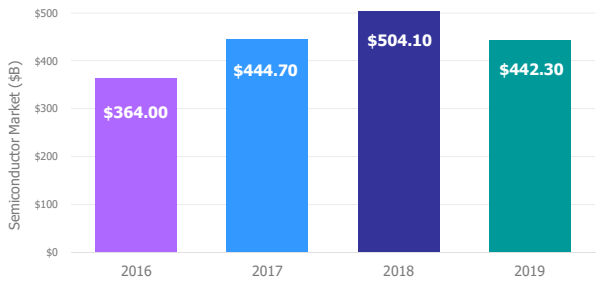
Decline in the DRAM an NAND flash memory markets for 2019



Source: IC Insight, The McClean Report 2020 (January), A Complete Analysis and Forecast of the Integrated Circuit Industry

Mentor
A Siemens Business

Overall World Wide Semiconductor Market

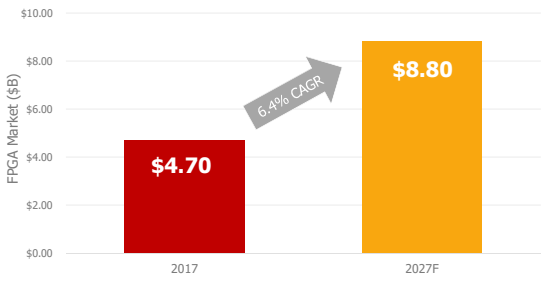


Source: IC Insight, The McClean Report 2020 (January), A Complete Analysis and Forecast of the Integrated Circuit Industry

5 HF, UT Austin, May 2020

Mentor

Global FPGA Market



Source: IBS, Semiconductor Market Analysis, 2017 Review, 2018 Projections, February 14, 2018

6 HF, UT Austin, May 2020

Mentor

FPGA Applications

7 HF, UT Austin, May 2020

Mentor

HOW VERIFICATION IS DONE TODAY

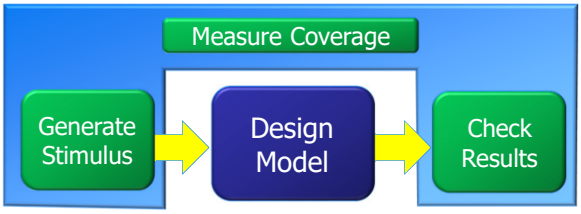
Mentor® A Siemens Business

What is Verification?

Verification is a process of ensuring that a design implementation meets its specification.

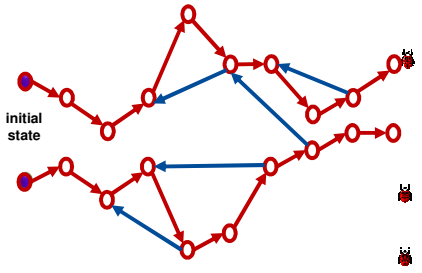
Simulation-Based Techniques

—Fundamental verification technique in use today
—Generally scales well
—Testing all possible states is generally incomplete



Assertions can be used to check results and measure coverage

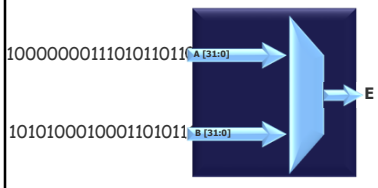
Simulation Traversal Through the State Space




```
// SystemVerilog Assertion
property p_comp;
  @(posedge clk)
    E |-> (A==B);
endproperty
assert property (p_comp);
```


Time Explosion Problem

- How long would it take to exhaustively simulate this example?




```
// SystemVerilog Assertion
property p_comp;
  @(posedge clk)
    E |-> (A==B);
endproperty
assert property (p_comp);
```

2^{64} vectors X 1 vector every micro-second = 584,941 years

An extremely fast simulator by today's standards!

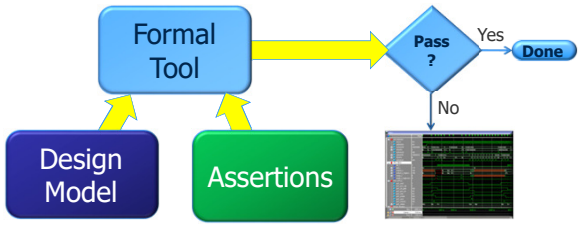
Simulation and the Time Explosion Problem

2^{64} vectors X 1 vector every micro-second = 584,941 years

Mentor

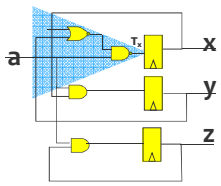
Formal-Based Techniques

- Does not require a testbench or input stimulus!
- Automatically uses algorithms to verify the functionality
- Verification can be complete
- Complements simulation-based techniques



Mentor

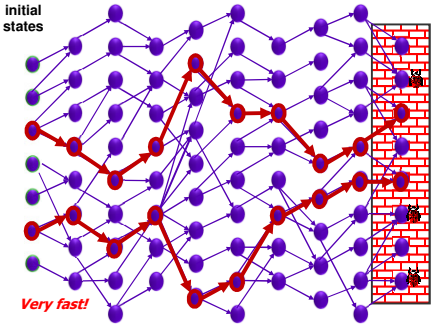
Conceptual Formal Tool



$T_x(a,x,y)$ *// next state*

Mentor

How is formal different than simulation?




```
// SystemVerilog Assertion
property p_comp;
  @(posedge clk)
    E |-> (A==B);
endproperty
assert property (p_comp);
```


Mentor

Mentor A Siemens Business

State Space Explosion

How many states exist in a typical design today?

Mentor

Time and State Space Explosion

- Simulation suffers from time explosion
- Formal suffers from state explosion
- Together they complement each other

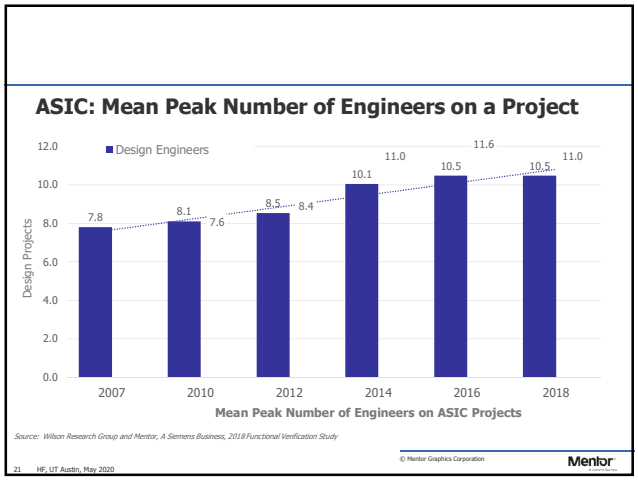
Mentor

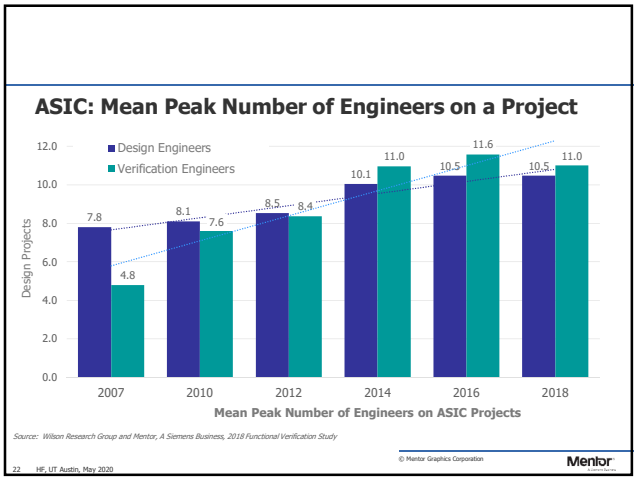
WHAT MAKES VERIFICATION DIFFICULT

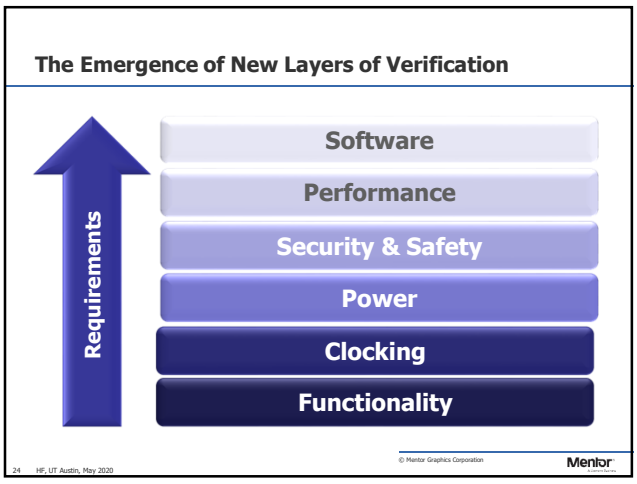
INDUSTRY DRIVERS

Rising Design Complexity

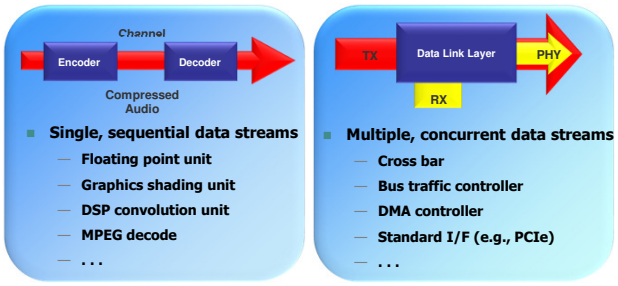
Mentor® A Siemens Business







What Makes Verification Difficult?



- **Single, sequential data streams**
  - **Floating point unit**
  - **Graphics shading unit**
  - **DSP convolution unit**
  - **MPEG decode**
  - **. . .**
- **Multiple, concurrent data streams**
  - **Cross bar**
  - **Bus traffic controller**
  - **DMA controller**
  - **Standard I/F (e.g., PCIe)**
  - **. . .**

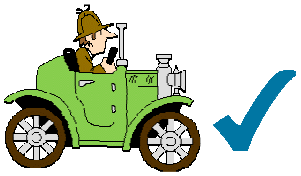
Sequential data streams ***1x*** **number of bugs**

Concurrent data streams ***5x*** **number of bugs**

-Ted Scardamalia, internal IBM study

Directed-Test Approach

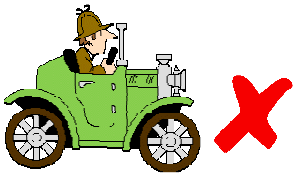
- Imagine verifying a car using a directed-test approach
  - —*Requirement: Fuse will not blow under any normal operation*
  - —*Scenario 1: accelerate to 37 mph, pop in the new Billie Eilish CD, and turn on the windshield wipers*



A FEW WEEKS LATER…

Directed-Test Approach

- Imagine verifying a car using a directed-test approach
  - —*Requirement: Fuse will not blow under any normal operation*
  - —*Scenario 714: accelerate to 48 mph, roll down the window, and turn on the left-turn signal*



The Concurrency Challenge

- A purely directed-test methodology does not scale
  - Imagine writing a directed test for this scenario!
  - Truly heroic effort—but not practical

29 HF, UT Austin, May 2020

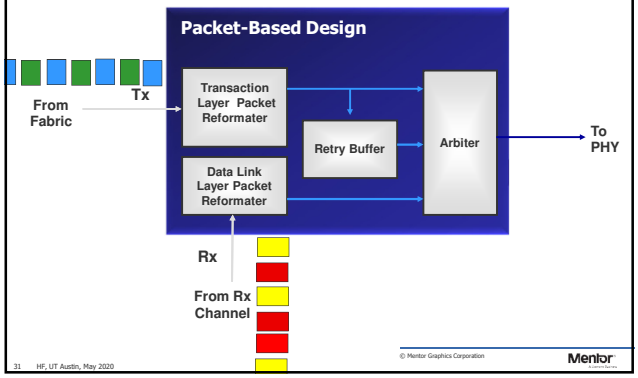
Finding Corner Case Bugs Due to Concurrency

Directed-test-based simulation finds the bugs you can think of…

Constrained-random simulation finds the bugs you never anticipated!

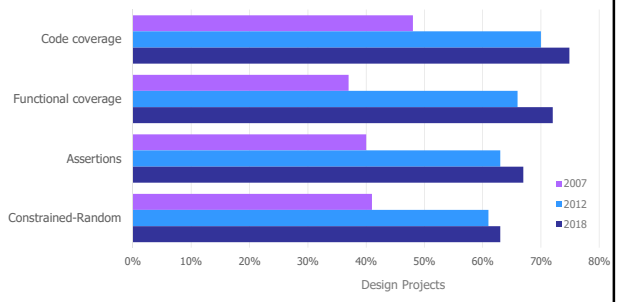
30 HF, UT Austin, May 2020

Concurrency is Complicated to Verify



31 HF, UT Austin, May 2020

A Maturing Industry to Address Growing Complexity



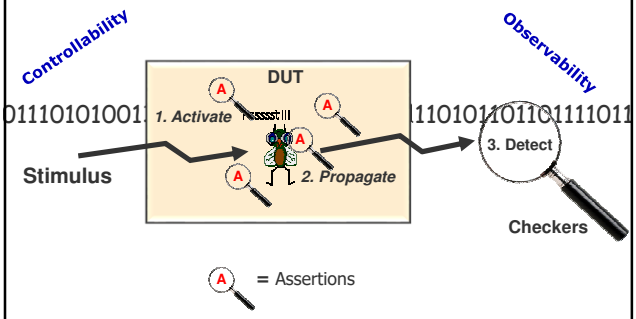
Source: Wilson Research Group and Mentor, A Siemens Business, 2018 Functional Verification Study

32 HF, UT Austin, May 2020

Mentor® A Siemens Business

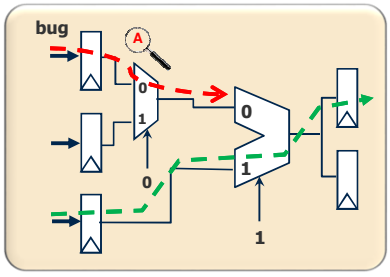
OBSERVABILITY & CONTROLLABILITY

Fundamental Challenge of Verification



Observability vs. Controllability

Test didn't set up the condition to propagate the bug



Assertions improve observability and reduce the need to propagate bugs

Controllability and Observability

In Context of Code Coverage

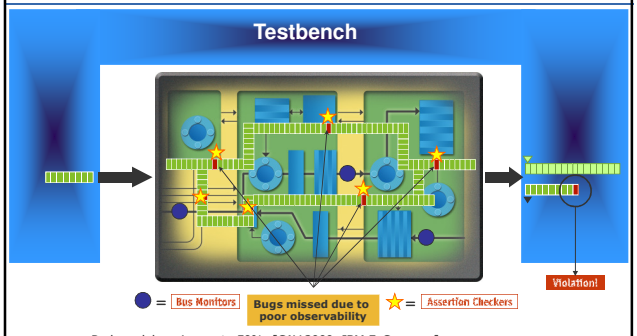
```
input in;
output out;
output [7:0] counter;

reg [7:0] counter;

wire counter_overflow = (counter == 8'hff);
wire counter_underflow = (counter == 8'h00);

always @(posedge clk or negedge xreset)
begin
        if (xreset == 0)
                counter <= 0;
        else if (en && in && !counter_overflow)
                counter <= counter + 1;
        else if (en && !in && !counter_underflow)
                counter <= counter - 1;
        else
                counter <= counter;
end
```

Poor Observability Misses Bugs

- Code coverage measures controllability
- 100% code coverage does not mean all bugs are detected *[S. Devadas, A. Ghosh, and K. Keutzer. DAC 1996]*
- DAC paper study found cases where:

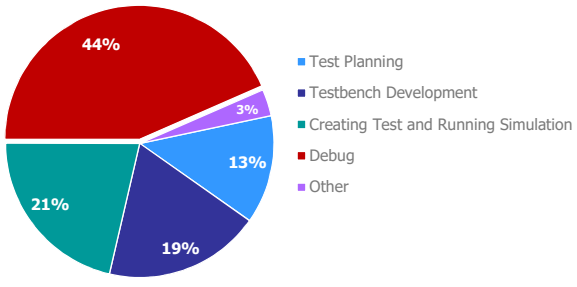
Code Coverage Achieved	% of covered lines observable
90% Covered	Only 54% Observable
100% Covered	Only 70% Observable

Assertions Improve Observability



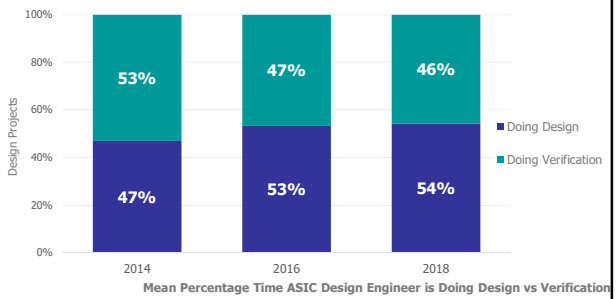
- Reduce debugging up to 50% [CAV 2000, IBM FoCs paper]
- Bugs detected closer to their source due to improved observability

Where Verification Engineers Spend Their Time



Source: Wilson Research Group and Mentor, A Siemens Business, 2018 Functional Verification Study

Mean % Time Design Engineer is Doing Design vs Verification



Mean Percentage Time ASIC Design Engineer is Doing Design vs Verification

Source: Wilson Research Group and Mentor, A Siemens Business, 2018 Functional Verification Study

ASSERTION-BASED VERIFICATION

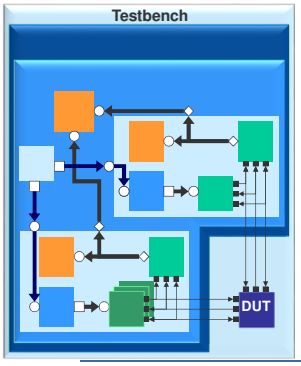
Assertion-Based Verification

"How can one check a large routine in the sense of making sure that it's right? In order that the man who checks may not have too difficult a task, the programmer should make a number of definite assertions which can be checked individually, and from which the correctness of the whole program easily flows."

Alan Turing, 1949

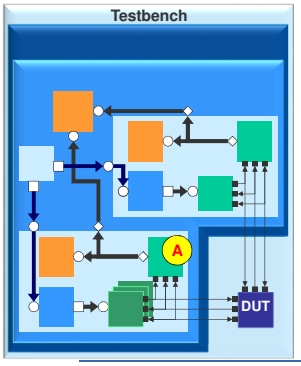
Property

- **Property**
  - — a statement of design intent
  - — used to specify behavior



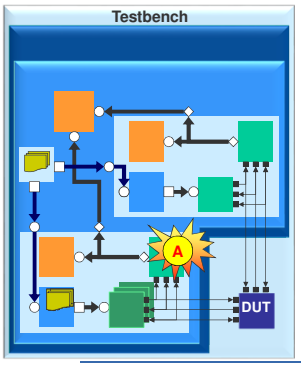
Assertion

- **Property**
  - — a statement of design intent
  - — used to specify behavior
- **Assertion**
  - — A verification directive



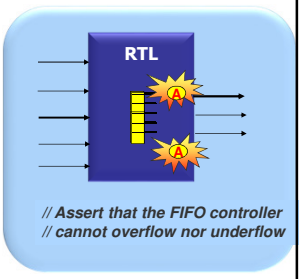
High-Level Assertion

- **Property**
  - — a statement of design intent
  - — used to specify behavior
- **Assertion**
  - — A verification directive
- **High-level**
  - — Architectural focused
  - — Can be part of testbench

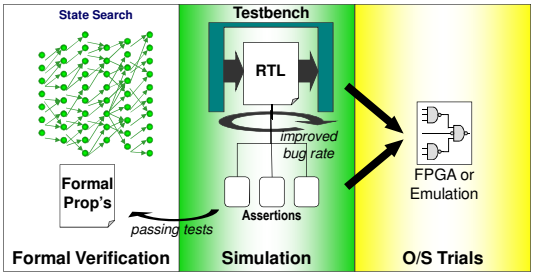


Low-Level Assertion

- **Property**
  - — a statement of design intent
  - — used to specify behavior
- **Assertion**
  - — A verification directive
- **High-level**
  - — Architectural focused
  - — Can be part of testbench
- **Low-level**
  - — Implementation focused
  - — Embedded in or bind to the RTL



How Assertions Are Used Today



[Foster, Larsen, Turpin - DVCon 2006]

Who should create the assertions?

Verification Engineer

- High-Level Assertions
  - ❖ Requirement focused
  - ❖ Black-box assertions
- Accounted for in testplan
- Compliance traceability
- Create reusable ABV IP

Design Engineer

- Low-Level Assertions
  - ❖ Implementation focused
  - ❖ White-box assertions
- Not accounted for in testplan
- Improve observability
- Reduce debugging time

Who should create high-level assertions?

Verification Engineer

- High-Level Assertions
  - Requirement focused
  - Black-box assertions
- Accounted for in testplan
- Compliance traceability
- Create reusable ABV IP

Design Engineer

- Low-Level Assertions
  - Implementation focused
  - White-box assertions
- Not accounted for in testplan
- Improve observability
- Reduce debugging time

Who should create low-level assertions?

Verification Engineer

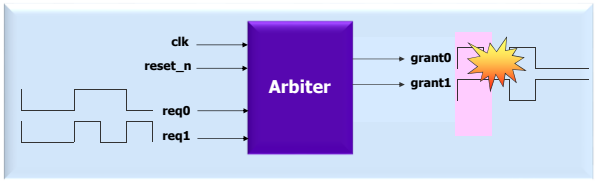
- High-Level Assertions
  - Requirement focused
  - Black-box assertions
- Accounted for in testplan
- Compliance traceability
- Create reusable ABV IP

Design Engineer

- Low-Level Assertions
  - Implementation focused
  - White-box assertions
- Not accounted for in testplan
- Improve observability
- Reduce debugging time

Specifying Design Intent

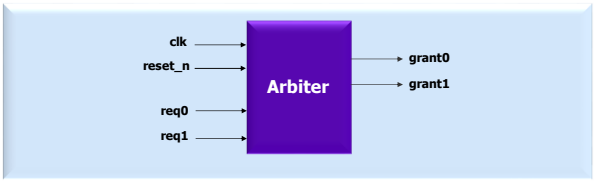
Assertions allow us to specify design intent in a way that lends itself to automation



// Assert that the grants for our simple arbiter are mutually exclusive

Identifying the Error Condition

For our arbiter example, we can write a Boolean expression for the error condition, as follows:



(grant0 & grant1) // error condition

Checking the Error Condition before Assertions

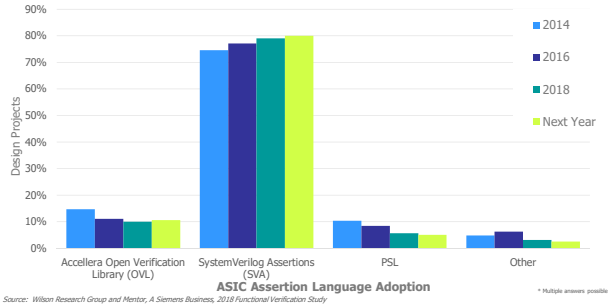
- Doesn't lend itself to automation.

```
module arbiter (clk, rst_n, req0, req1, grant0, grant1);
  . . .
  always @(posedge clk or negedge rst_n) begin
    if (rst_n != 1'b0) // active low reset
      if (grant0 & grant1)
        $display ("ERROR: Grants not mutex");
. . .
endmodule
```

Error Condition Boolean Expression

Mentor

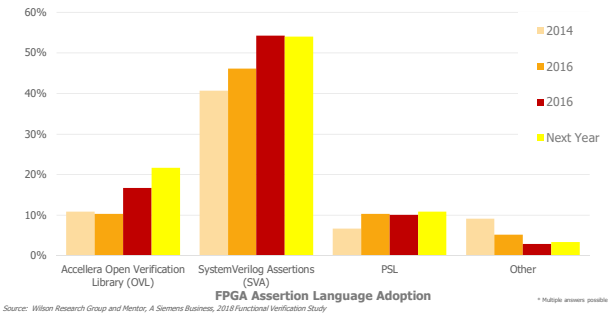
ASIC: Assertion Language Adoption



Source: Wilson Research Group and Mentor, A Siemens Business, 2018 Functional Verification Study

Mentor

FPGA: Assertion Language Adoption

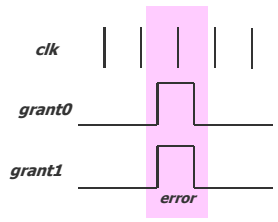


Source: Wilson Research Group and Mentor, A Siemens Business, 2018 Functional Verification Study

Mentor

IEEE 1800 SystemVerilog Mutex Example

grant0 and grant1 must be mutually exclusive



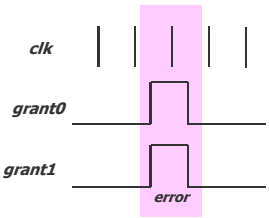

```
assert property ( @(posedge clk) disable iff (~rst_n) !(grant0 & grant1));
```


Mentor

Mentor A Siemens Business

IEEE 1850 PSL Mutex Example

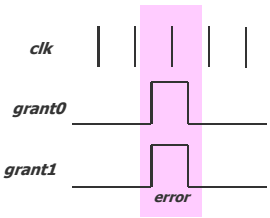
grant0 and grant1 must be mutually exclusive




```
assert always (!(grant0 & grant1) abort ~rst_n) @(posedge clk);
```


Accellera OVL Mutex Example

grant0 and grant1 must be mutually exclusive




```
ovl_never a_mutex (clk, rst_n, (grant0 & grant1));
```


INDUSTRY CASE STUDIES

Published Data on Assertions Use

Percentage bugs found by various techniques

Assertion Monitors	34%
Cache Coherency Checkers	9%
Register File Trace Compare	8%
Memory State Compare	7%
End-of-Run State Compare	6%
PC Trace Compare	4%
Self-Checking Test	11%
Simulation Output Inspection	7%
Simulation Hang	6%
Other	8%

Kantrowitz and Noack *[DAC 1996]*

Assertion Monitors	25%
Register Miscompare	22%
Simulation "No Progress"	15%
PC Miscompare	14%
Memory State Miscompare	8%
Manual Inspection	6%
Self-Checking Test	5%
Cache Coherency Check	3%
SAVES Check	2%

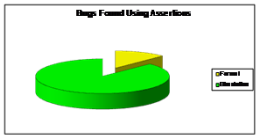
Taylor *et al.* *[DAC 1998]*

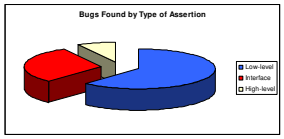
- **17%** of bugs found by assertions on **Cyrix** M3(p1) project *[Krolnik '98]*
- **50%** of bugs found by assertions on **Cyrix** M3(p2) project *[Krolnik '98]*
- **85%** of bugs found using over 4000 assertions on an **HP** server chipset project *[Foster and Coelho HDLCon 2001]*
- Thousands of assertions in Intel Pentium project *[Bentley 2001]*
- **10,000** OVL assertion in Cisco project *[Sean Smith 2002]*

DAC 2008 Sun paper with lots of metrics

Assertion-Based Verification of a 32 thread SPARC™ CMT Processor
[Turumella, Sharma, DAC 2008]

Category	Unique	Instantiated
Low-Level	3912	132773
Interface	5004	44756
High-Level	1930	18618

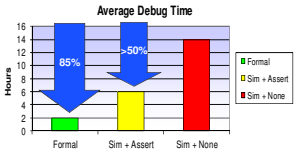


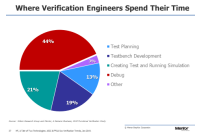


Significant reduction in debugging time

Assertion-Based Verification of a 32 thread SPARC™ CMT Processor
[Turumella, Sharma, DAC 2008]

Category	Unique	Instantiated
Low-Level	3912	132773
Interface	5004	44756
High-Level	1930	18618

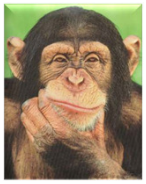




SUMMARY

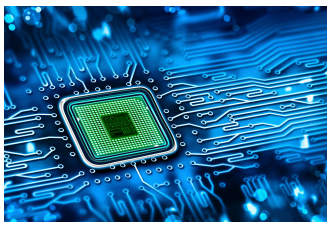
Assertion-Based Verification

- The process of creating assertions forces the engineer to think. . . and in this incredible world of automation, there is no substitute for thinking.



For Additional Info on Industry Trends

- Latest Wilson Research Group Functional Verification Study
  - Verification Horizon Blog
  - **http://go.mentor.com/558dr**



Mentor®
A Siemens Business

www.mentor.com