Myth vs. Reality

Industry Trends in Functional Verification

November 2009

Harry D. Foster
Chief Verification Scientist
What Is Functional Verification?

- The process of demonstrating (or proving) that a design implementation satisfies its specified requirements.
Demonstrate Conformance

Simulation

10110101001

Specification

Implementation

Error

10100000
Prove Equivalence

Formal Verification

assert never (Error==1'b1);
What’s Going On In The Industry?
State of the Industry

There are three types of lies - lies, damn lies, and statistics.

-Mark Twain
Myth vs. Reality?
Slowing Adoption of New Technology
Is The Adoption Of New Technology Slowing?

- How will this effect functional verification?

- Perhaps complexity will no longer be a concern?
Slowing Adoption of New Technology

- 1979: 29,000 Transistors
  - 8086

- 1982: 134,000 Transistors
  - 80186

- 1985: 275,000 Transistors
  - 286

- 1989: 1,290,000 Transistors
  - Pentium

- 1993: 3.1M Transistors
  - Pentium Pro

- 1995: 5.5M Transistors
  - Pentium II

- 1997: 7.5M Transistors
  - Pentium III

- 1999: 9.5M Transistors
  - Pentium 4

- 2000: 42M Transistors
  - Itanium 2 (9MB cache)

- 2002: 220M Transistors
  - Dual Core Itanium

- 2004: 592M Transistors
  - Itanium 2

- 2005: 1.72B Transistors
  - Dual Core Itanium

- 2008: 2 Billion Transistors
  - Tukwila Quad Core

- 1999: 386
  - 275,000 Transistors
  - Pentium

- 1997: 1999
  - 7.5m+ Transistors
  - Pentium III

- 1995: 1995
  - 5.5M+ Transistors
  - Pentium II

- 1993: 1993
  - 3.1M+ Transistors
  - Pentium Pro

- 1989: 1989
  - 1,290,000 Transistors
  - Pentium

- 1985: 1985
  - 275,000 Transistors
  - 286

- 1979: 1979
  - 29,000 Transistors
  - 8086
Frequent Statements About the Slowing of Technology Adoption

“The problem is that Moore’s Law has collapsed,” he says. Coburn asserts that there has been a slowdown in the previously steady move to smaller geometries and larger wafer sizes.

Pip Coburn, Coburn Ventures
December 15th, 2008

“The slowdown in process technology transitions will mean that the semiconductor industry will be driven more by economics than technology …”

“You are not seeing these geometries rise and fall off the way they did before. Rather, they are living on.”

Len Jelinek, Director and chief analyst, Semiconductor Manufacturing, for iSuppli

“And, the customers have slowed down or delayed their technology transitions either by leveraging their existing installed base or just by delaying their new product introduction for later.”

Eric Meurice, Chairman,
President and CEO ASML Holding N.V.


Silicon Volume of Wafer Starts
(in 300mm Wafer Equivalents)

Volume of Wafer Starts by Linewidth (300mm wafer equivalents)

Source: VLSI Research, Silicon Demand, EDA Tech Forum June, 2009
Reticle Sales Trends

Reticle Revenue Share by Linewidth

Source: VLSI Research, Reticles, September 2008
IC Fab Capacity by Node *

IC Fab Capacity by Node (MSI) January 2004 - to June 2008

Source: VLSI Research, December 15, 2008

* Note: Realized
Capacity Utilization Improvement in March & April Favors 65/45nm Technology

Capacity by Node - April

Source: Selantek Capacity Analysis, May 5, 2009
Adoption of Leading-Edge Semiconductor Technology Is at the Same Rate as in the Past
Myth vs. Reality?

Rising Design Costs Will Limit New Applications
Transistors Produced per Electrical Engineer
Nearly 4-Orders of Magnitude since 1985

Gary Smith EDA, 2008 Seat Count Analysis VLSI Research, 2008 - Transistors Produced Analysis
EDA Revenue Is Flat 2% of IC Revenue

Source: Mentor Graphics, EDAC MSS & SIA WSTS
EDA Cost per Transistor vs Total IC Revenue per Transistor

Source: SIA, VLSI Research, Federal Reserve

Note: EDA Cost Consists of EDA License and Maintenance revenue adjusted for Inflation… 1985 - 2007
SOC Design Costs Forecasted to Exceed $100 Million Within 3 Years

Impact of Design Technology on SOC Consumer Portable Implementation Cost

- **2007e**: Hardware Costs, Software Costs
- **2008f**: Hardware Costs, Software Costs
- **2009f**: Hardware Costs, Software Costs
- **2010f**: Hardware Costs, Software Costs
- **2011f**: Hardware Costs, Software Costs
- **2012f**: Hardware Costs, Software Costs

Notes:
1. Total Hardware Engineering Costs + EDA Tool Costs
2. Total Software Engineering Costs + Electronic Software Design Tool Costs

Source: 2007 ITRS Roadmap
Software Developers Outnumber Hardware Designers 2-to-1

(in 000’s)

System Design Has Shifted to the Semiconductor Suppliers

Much of what was part of the end-system is now incorporated within a System-On-Chip
Rising Design Costs Limit New Applications
Semiconductor Companies Are Assuming
More of the System and Embedded Software Engineering Design Responsibility
Myth vs. Reality?
Verification Is Keeping Up With Moore’s Law
An Optimistic View of the Productivity Gap

Let’s assume...

- Number of transistor doubles every 18 months (58% / yr)
- Amount of logic we can design doubles every 2 years (41% / yr)
- Amount of logic we can verify doubles every 2.5 years (25% per year)
Productivity Gap

- The Verification Gap
  - Many companies still using 1990’s verification technologies
  - Traditional verification techniques can’t keep up

*Based on data from the 2003 ITRS, Collett International 2004 FV Survey, and customer surveys*
Code Coverage

Statement Coverage Report Example

```plaintext
nextstate_proc : PROCESS ( 
  cpu_wr, 
  cpuwait_int, 
  current_state 
) 
BEGIN 
  CASE current_state IS 
    WHEN RESET => 
      next_state <= HIT; 
    WHEN HIT => 
      IF (cpuwait_int = '1') THEN 
        next_state <= MISS; 
      ELSIF (cpu_wr = '1') THEN 
        next_state <= WRITE; 
      ELSE 
        next_state <= HIT; 
      END IF; 
    WHEN WRITE => 
      next_state <= HIT; 
    WHEN MISS => 
      next_state <= HIT; 
    -- coverage off 
    WHEN OTHERS => 
      next_state <= RESET; 
    -- coverage on 
  END CASE; 
END PROCESS nextstate_proc;
```
The Verification Gap

Directed Test

State-of-the-Art Verification Circa 1990

- 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 John Mayer CD, and turn on the windshield wipers
A Few Weeks Later. . . . .
The Verification Gap

Directed Test

State-of-the-Art Verification Circa 1990

- 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
Concurrency Challenge

- A purely directed-test methodology does not scale
  - Imagine writing a directed test for this scenario!
  - Truly heroic effort—but not practical
Today’s Concurrency Challenge
Verification is Keeping Up With Moore’s Law

No, but we could as an industry do better!
What Are We Doing To Close The Gap?
Moore with Less

Verification for Every Man, Woman, and Child in India
Myth vs. Reality?

70% of the project effort is spent in verification....
Design Engineers Are Becoming Verification Engineers

Source: 2007 Farwest Research IC/ASIC Functional Verification Study, Used with Permission
More and More Verification Cycles
Faster Computers

The graph shows the trend of MIPS (Million Instructions Per Second) over the years from 1975 to 2005. The data points indicate a steady increase in computational speed over time, with a linear trend line illustrating the growth.
Lots of Computers

AMD Grid Growth 2001-2006
(Relative to 2001 = 1.0)

<table>
<thead>
<tr>
<th>Year</th>
<th># of Servers</th>
</tr>
</thead>
<tbody>
<tr>
<td>1996</td>
<td>50</td>
</tr>
<tr>
<td>2006</td>
<td>5000+ (over 10,000 CPUs)</td>
</tr>
</tbody>
</table>

Source: The AMD Grid: Enabling Grid Computing for the Corporation, August 2006
So, with all this effort, what’s the results?
Results
2/3 Projects Miss Schedule

Designs completed on time according to project's original schedule
Results

77% Respins Due to Functional Bugs

Results

Types of Flaws

Results
Causes of Functional Flaws

Causes:
- Incorrect or Incomplete Specification: 35%
- Changes in Specification: 41%
- Design Error: 60%
- Flaw in Internal Reused IP: 18%
- Flaw in External IP: 15%
- Other: 3%

Source: 2007 Far West Research and Mentor Graphics
Myth vs. Reality?

The industry is evolving its verification capabilities?
Productivity Gap

- **The Verification Gap**
  - Many companies still using 1990’s verification technologies
  - Traditional verification techniques can’t keep up

*Based on data from the 2003 ITRS, Collett International 2004 FV Survey, and customer surveys*
An Optimistic View of the Productivity Gap

- Number of transistor doubles every 18 months (58% / yr)
- Amount of logic we can design doubles every 2 years (41% / yr)
- Amount of logic we can verify doubles every 2.5 years (25% per year)

Functional Verification Technique Adoption

- Code Coverage Analysis: 48%
- Functional Coverage: 40%
- Assertions: 37%
- Formal Property Checking: 19%
Verification Is Keeping Up With Moore’s Law

No, but we could as an industry do better!
Logic Is A Branch of Philosophy

- Origins can be traced back to the ancient Greeks
- Logic is concerned with reasoning about behavior
Propositions from classic logic

- Take the universe as our model
  - *The moon is a satellite of the earth.*
  - *The moon is rising (now).*
Classic Logic

- model
- property
- True / False
Propositions from classic logic

- What you can’t express in classic logic
  - *The moon is rising again and again.*
Reactive Systems

- A more expressive logic is required to express properties for reactive systems.

- Temporal logic, simple way to reason about reactive systems without explicitly specify time
Temporal Logic

- Statements about reactive systems depend on time.
  - For example...
    - P and Q are mutually exclusive for all values of time

- Temporal logic can describe the ordering of events in time without introducing time explicitly.
  - Without temporal logic, we would be forced to explicitly write equations involving time:
  - For example, \( \forall t.!(P(t) \& Q(t)) \)
SVA Language Structure

- Sequence
  - Temporal delay ##

```plaintext
start  ##1  transfer

clk

start

transfer
```
SVA Language Structure

- Sequence
  - Temporal delay ##

```
start   ##2   transfer
```

```
clk
```

```
start
```

```
transfer
```

```
SVA Language Structure

Properties

- Overlapping sequence implication operator $|->$

```
ready ##1 start $|->$ go ##1 done
```

```
assertion property ( @(posedge clk) ready ##1 start $|->$ go ##1 done );
```
### SVA Language Structure

**Properties**

- Non-overlapping sequence implication operator $|=>$

$$\text{ready} \, \#\#1 \, \text{start} \, |=> \, \text{go} \, \#\#1 \, \text{done}$$

<table>
<thead>
<tr>
<th>clk</th>
<th>ready</th>
<th>start</th>
<th>go</th>
<th>done</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**NOTE:** A $|=>$ B is the same as A $|->$ $\#\#1$ B
Bus-Based Design Example
Nonpipelined Bus Interface

Master

I/F

clk
rst_n
sel[0]
en
addr
write
rdata
wdata

I/F

Slave 0

HDF – 2009
Non-Burst Write Cycle

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>addr</td>
<td></td>
<td></td>
<td></td>
<td>Addr 1</td>
<td></td>
</tr>
<tr>
<td>write</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>sel[0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>en</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>wdata</td>
<td></td>
<td></td>
<td></td>
<td>Data 1</td>
<td></td>
</tr>
</tbody>
</table>

conceptual state

INACTIVE  | START  | ACTIVE  | INACTIVE
Non-Burst Read Cycle

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>addr</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Addr 1</td>
</tr>
<tr>
<td>write</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>sel[0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>en</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rdata</td>
<td></td>
<td></td>
<td></td>
<td>Data 1</td>
<td></td>
</tr>
</tbody>
</table>

conceptual state

<table>
<thead>
<tr>
<th></th>
<th>INACTIVE</th>
<th>START</th>
<th>ACTIVE</th>
<th>INACTIVE</th>
</tr>
</thead>
</table>

Conceptual Bus States

INACTIVE
\[ sel[0] == 0 \]
\[ en == 0 \]

no transfer

START
\[ sel[0] == 1 \]
\[ en == 0 \]

setup

TRANSFER

ACTIVE
\[ sel[0] == 1 \]
\[ en == 1 \]

setup

no transfer
## Interface Requirements

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Bus legal transitions</strong></td>
<td></td>
</tr>
<tr>
<td>p_state_reset_inactive</td>
<td>Initial state after reset is INACTIVE</td>
</tr>
<tr>
<td>p_valid_inactive_transition</td>
<td>ACTIVE state does not follow INACTIVE</td>
</tr>
<tr>
<td>p_valid_start_transition</td>
<td>Only ACTIVE state follows START</td>
</tr>
<tr>
<td>p_valid_active_transition</td>
<td>ACTIVE state does not follow ACTIVE</td>
</tr>
<tr>
<td>p_no_error_state</td>
<td>Bus state must be valid: !(se==0 &amp; en==1)</td>
</tr>
<tr>
<td><strong>Bus stable signals</strong></td>
<td></td>
</tr>
<tr>
<td>p_sel_stable</td>
<td>Slave select signals remain stable from START to ACTIVE</td>
</tr>
<tr>
<td>p_addr_stable</td>
<td>Address remains stable from START to ACTIVE</td>
</tr>
<tr>
<td>p_write_stable</td>
<td>Control remains stable from START to ACTIVE</td>
</tr>
<tr>
<td>p_wdata_stable</td>
<td>Data remains stable from START to ACTIVE</td>
</tr>
</tbody>
</table>

![Diagram](image)
Use Modeling Code to Simplify Coding

```vhdl
`ifdef ASSERTION_ON

// Map bus control values to conceptual states

if (rst_n) begin
    bus_reset = 1;
    bus_inactive = 1;
    bus_start = 0;
    bus_active = 0;
    bus_error = 0;
end
else begin
    bus_reset = 0;
    bus_inactive = ~sel & ~en;
    bus_start = sel & ~en;
    bus_active = sel & en;
    bus_error = ~sel & en;
end
`endif
```

---

**INACTIVE**
- sel[0] == 0
- en == 0

**START**
- sel[0] == 1
- en == 0

**ACTIVE**
- sel[0] == 1
- en == 1

Transitions:
- Setup (INACTIVE to START)
- Transfer (START to ACTIVE)
- Setup (ACTIVE to INACTIVE)
property p_valid_inactive_transition;
@'(posedge clk) disable iff (bus_reset)
  (bus_inactive) =>
  ((bus_inactive) || (bus_start));
endproperty

a_valid_inactive_transition:
  assert property (p_valid_inactive_transition);
Formalize Properties

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bus legal transitions</td>
<td>Only ACTIVE state follows START</td>
</tr>
<tr>
<td>p_valid_start_transition</td>
<td>Only ACTIVE state follows START</td>
</tr>
</tbody>
</table>

```vhdl
property p_valid_start_transition;
@ (posedge clk) disable iff (bus_reset)
  (bus_start) |=> (bus_active);
endproperty

a_valid_start_transition:
  assert property (p_valid_start_transition);
```
Demonstrate Conformance

Simulation

Design Under Verification

assert property (p_valid_start_transition);
assert never (Error==1'b1);
Myth vs. Reality?

The biggest bottleneck in the flow is simulation performance?
Debugging is the Bottleneck

Effort Allocation of Dedicated Verification Engineers by Type of Activity

- Verification Debug: 52%
- Testbench Development: 34%
- Other: 14%
### Assertion-Based Verification of a 32 thread SPARC™ CMT Processor

**DAC 2008, Turumella, Sharma**

<table>
<thead>
<tr>
<th>Category</th>
<th>Unique</th>
<th>Instantiated</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low-Level</td>
<td>3912</td>
<td>132773</td>
</tr>
<tr>
<td>Interface</td>
<td>5004</td>
<td>44756</td>
</tr>
<tr>
<td>High-Level</td>
<td>1930</td>
<td>18618</td>
</tr>
</tbody>
</table>

#### Average Debug Time

![Debug Time Chart]

- **Formal**: 85%
- **Sim + Assert**: >50%
- **Sim + None**: 0%

**Legend**

- Green: Formal
- Yellow: Sim + Assert
- Red: Sim + None
Productivity Gains by Raising the Level of Abstraction

#include "systemc.h"
SC_MODULE(adder) // module (class) declaration
{ sc_in<int> a, b; // ports
 sc_out<int> sum; void do_add () // process { sum = a + b; }
 SC_CTOR(adder) // constructor (SC_METHOD (do_add); // register do_add to kernel sensitive << a << b; // sensitivity list of do_add });
TLM is Much More Efficient

<table>
<thead>
<tr>
<th>Source</th>
<th>Simulation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Functional</td>
<td>10,000x (1 min)</td>
</tr>
<tr>
<td>Structural</td>
<td>1,000x</td>
</tr>
<tr>
<td>Transaction</td>
<td>100x</td>
</tr>
<tr>
<td>Cycle</td>
<td>10x</td>
</tr>
<tr>
<td>RTL</td>
<td>1x (7 days)</td>
</tr>
</tbody>
</table>

- C/C++
- Untimed TLM SystemC
- Timed TLM SystemC
- Cycle Accurate TLM
- RTL

Function arguments
Transaction
Transaction
Protocol
Protocol
Myth vs. Reality?

- Slowing Adoption of New Technology
  - No

- Rising Design Costs Will Limit New Applications
  - Software component driving cost

- Verification Is Keeping Up With Moore’s Law
  - No, but industry could do better by evolving its verification processes
Evolving Verification Process Capabilities

low ← Evolving Capabilities → high