Formal Verification in Semiconductor Designs

Vigyan Singhal
CEO, Oski Technology

Agenda

- Introduction to Semiconductor World and Oski Technology
- Formal Verification Concepts
Speaker Background

- Vigyan Singhal, CEO, Oski Technology

Education
- B.S. Computer Science, IIT Kanpur, India, 1989
- Ph.D. Computer Science, University of California at Berkeley, 1995

Jobs in large companies
- Summer intern, Motorola, Austin, Texas, 1993
- Research Scientist, Cadence, San Jose, California, 1995-1999

Silicon valley start-ups
- Jasper Design Automation, 1999-2005 (bought by Cadence)
- Elastix 2006-2008 (bought by eSilicon)
- Oski Technology 2005-2014
Different Viewpoints, Different Formal Goals

University PhD Researcher
(4 yrs 1991-1995)
(UC Berkeley)
Goal: advance state-of-the-art

EDA tool developer
(10 yrs 1995-2005)
(Cadence, Jasper)
Goal: build competitive tools

Semiconductor tool user
(9 yrs 2005-2014)
(Oski)
Goal: optimize $ and time-to-market
Electronic Chip Design Flow

- About 20-50 different EDA tools in any design flow
- Tools help improve cost, performance and power of chips
- Oski technology can reduce tool runtimes by 1,000,000X!
- EDA companies are staffed by highly skilled programmers
  - Many have PhD from top schools (Stanford, Berkeley, MIT, CMU, Oxford)
Manufacturing an Electronic Chip

1. Wafers sliced from ingot
2. Polished wafer
3. UV light to expose
4. View of die
5. Glass mask
6. Projected image of die seen through mask
7. Wafer with multiple die
8. Wafer holder attached to aligner
Types of Post-Silicon Flaws

Verification is the still the largest problem

Bug-fix Cost Rises Exponentially

- Block-level design
- Block-level verification
- Chip-level verification
- ECO phase
- Tapeout
- Silicon is back

Cost:
- $10M
- $1M
- $100k
- $10k
- $1k
- $100
- $10M
My story: Funded Jasper by “Pay-per-bug”!

- My first successful business model
- “Bug” is defined as something that violates the specification AND customer decides to fix it
- $/bug varies, depending on design stage

Source: xkcd.com
Leading Semiconductor Customers are Global

Intel
NVIDIA
Qualcomm
TI
ARM
ST Micro
Infineon
Samsung
Huawei
MediaTek
EDA Supports IP and Services Industry

- Electronics Industry - $2 Trillion
- Semiconductor Industry - $300B
- Electronic Design Automation (EDA) Industry - $8B
Players in Semiconductor Market

- **Systems and Electronics**
  - Apple, Cisco, Dell, Lenovo, Samsung

- **Silicon suppliers (electronic chips)**
  - Intel, Qualcomm, MediaTek, Samsung

- **EDA and Services**
  - Cadence, Synopsys, Oski Technology

- **Design IP**
  - ARM, Imagination, Synopsys

- **Fabs**
  - TSMC, Samsung, GlobalFoundries

Flow:
- Electronic chips from Silicon suppliers to Systems and Electronics
- IC design software from EDA and Services
- IC design blocks from Design IP
- Manufacturing from Fabs
Oski Supports the Semiconductor Eco-system

- Silicon suppliers (electronic chips): Intel, Samsung, Apple, NVIDIA, Cisco, Sony
  - EDA: Cadence, Jasper, Mentor, Synopsys
  - Design IP: ARM, Imagination, Cadence, Synopsys
  - Services & Methodology
  - Integrity
  - Tools
  - IC design blocks
  - Feature Requests

Oski TECHNOLOGY CONFIDENTIAL 15
Oski Focuses on the Most Challenging Problems

- Catch corner case bugs early
- Increase verification efficiency
- Replace block-level simulation
- Enable formal sign-off

End-to-End Formal

Assertion-based Verification (ABV)

Formal Apps

Automatic Formal

Complexity & Benefits

Adoption

OSKI TECHNOLOGY CONFIDENTIAL
Oski Methodology Leadership

- Focus on hands-on methodology transfer at customers
  - Through live verification projects and advanced formal training

- Best Paper Awards, using formal methodology
  - 2006 DVCon: Formal Testplanning
  - 2012 DAC: Bypass Logic Verification (with Cisco)
  - 2014 DVCon: Formal Sign-off with Bounded Proofs (with Samsung)

- DAC 2012 Live 72-hour Challenge
- Sponsor Best Solver in annual HWMCC contest
- Quarterly Oski Decoding Formal Club meetings
Scarcity of Formal Verification Talent

- Best expertise comes from analytical problem solving
  - Oski’s first recruitment test is a mathematics test!
  - Problem-solving interview

- Silicon valley companies (Apple, Google) are paying highest salaries for formal verification experts

- Oski Technology solves hardest problems for large semiconductor companies
  - ARM, Broadcom, Huawei, Qualcomm, NVIDIA, Samsung
Formal Verification Concepts

System-on-Chip (SoC) Design, Verification

- **Design:**
  - Specification to micro-architecture and down to implementation

- **Verification:**
  - Block-level (scheduler) to unit-level (USB interface unit) to chip-level (Mobile applications processor)
  - Verification consumes about 70% of resources
  - Each silicon re-spin costs more than $2M
  - Most frequent cause of re-spin: functional bugs

<table>
<thead>
<tr>
<th>Specification + constraints</th>
<th>System architecture + estimates</th>
<th>RTL/IS Implementation + results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processors, IPs, Memories, Busses</td>
<td>Custom HW, Memory, Interface</td>
<td>Registers, ALUs/PUs, Memories, Gates</td>
</tr>
<tr>
<td>μProcessor, Memory, Interface</td>
<td>IP Comp., Interface</td>
<td>ALUs/FUs, Mem, RF</td>
</tr>
<tr>
<td>Control, Pipeline, IF FSM</td>
<td>Datapath, Control, IF FSM</td>
<td>IP Netlist, Memory</td>
</tr>
<tr>
<td>Control, Memory, RAM, PC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>State, State</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

OSKI TECHNOLOGY CONFIDENTIAL
Simulation-based Verification

- BFM creates API for tests
- Tests responsible for input generation, and for covering all the interesting cases
- Checker verifies end-to-end correctness (higher-level than design)
Why Formal Verification?

- Traditional method (testing & simulation)
  - Design Under Test
  - Test vectors
  - Patterned vectors
  - Cannot cover all possible cases
  - Possibility of surviving subtle (corner case) bugs

- Formal Verification
  - System Model
  - Test properties
  - Deadlock ...
  - Test results
  - True or False
  - Equivalent to simulating all cases in simulation
  - No bug (according to the property)
  - Property checking (model checking)
**Model Checking Use Model**

Does the design have a given desirable property?

- **Properties** - Specification
- **Constraints** - Assumptions needed to prove Properties
- **Design under test** - Implementation

![Diagram]

Verified or CounterExample

\[ p \]

\[ q \]
Model Checking

Is there a sequence of input assignments such that p is 1 at any finite time time?

Combinational gates + flops (with initial values)
Model Checking

Is there a sequence of input assignments such that p is 1 at any finite time time?

Combinational gates + flops (with initial values)
Complexity: PSPACE-hard (Aziz 93), harder than NP-Complete
**Key Notion: “Reachable States”**

*reachable states*: the set of states reachable (under any input sequence) from the input state

The behavior of the design from unreachable states is not relevant to the correctness of the property

Finding the set of reachable states is hard for many designs (more than 100 registers => more than $2^{100}$ possible states!)

=> state space explosion problem!
Reachability Example

Checker: \( (st == 2'b01) \Rightarrow \neg b \)

```verilog
input a;
reg b;
reg [1:0] st;

always @(posedge clk or negedge rst)
    if (~rst) st <= 2'b00;
    else case( st )
        2'b00: if (~a) st <= 2'b01;
        2'b01: st <= 2'b10;
        2'b10: if (a) st <= 2'b00;
    endcase

always @(posedge clk or negedge rst)
    if (~rst) b <= 1'b0;
    else if (~a | b) b <= 1'b0;
    else b <= 1'b1;
```
Reachable State Computation: Explicit

1. Reached set = \{000\}
   State to explore = 000
2. Reached set = \{000, 001, 010\}
   State to explore = 001
3. Reached set = \{000, 001, 010\}
   State to explore = 010
4. Reached set = \{000, 001, 010, 100, 101\}
   State to explore = \{100, 101\}
5. Reached set = \{000, 001, 010, 100, 101\}
   State to explore = \{101\}
6. Reached set = \{000, 001, 010, 100, 101\}
   **Num of steps = Num of reached states**
SAT-based Model Checking

- Bounded MC
  - Incomplete: can only find bugs, no proofs
  - [Biere 99]

- Unbounded MC
  - Complete, used for proofs
  - Induction-based [Sheeren 00]
  - Backward reachability [McMillan 02, Ganai 04]
  - Interpolants [McMillan03]
  - Property Directed Reachability [Bradley11]
**Background: SAT Problem Definition**

Given a CNF formula, \( f \):

- A set of variables, \( V \) \( (a, b, c, d) \)
- Conjunction of clauses CNF \( (C_1, C_2, C_3) \)
- Each clause: disjunction of literals over \( V \)

Does there exist an assignment of Boolean values to the variables, \( V \) which sets at least one literal in each clause to ‘1’?

Example:

\[
(a + b + \overline{c})(\overline{a} + c)(a + \overline{b} + c)
\]

\( C_1 \quad C_2 \quad C_3 \)

\[
a = 1 \\
b = 0 \\
c = 1
\]
Background: Combinational SAT

Can circuit output “p” be 1?

\[(a \lor \neg g) \land (b \lor \neg g) \land (\neg a \lor \neg b \lor g)\]

\[\land (\neg g \lor p) \land (\neg c \lor p) \land (g \lor c \lor \neg p)\]

\[\text{CNF}(p)\]

p is satisfiable when the formula \(\text{CNF}(p) \lor p\) is satisfiable

Complexity: NP-complete (Cook 71)
SAT-based Model Checking

- Bounded MC
  - Incomplete: can only find bugs, no proofs
  - [Biere 99]

- Unbounded MC
  - Complete, used for proofs
  - Induction-based [Sheeren 00]
  - Backward reachability [McMillan 02, Ganai 04]
  - Interpolants [McMillan 03]

Complexity: PSPACE-complete (Aziz 93)
**Bounded Model Checking (Biere 99)**

- Unfold the model $k$ times:
  \[ U_k = C_0 \land C_1 \land \ldots \land C_{k-1} \]

- Use SAT solver to check satisfiability of
  \[ I_0 \land U_k \land Z_k \]

- A satisfying assignment is a counterexample of $k$ steps
**Induction-based (Simple Case)**

- **Base step (initial state is not a bad state)**
  - \( I_0 \land Z_0 \) is unsatisfiable

- **Induction step: (if n-th state is not bad, then (n+1)-th states are not bad)**
  - \( \neg Z_n \land U_n \land Z_{n+1} \) is unsatisfiable
  - (i.e. \( \neg Z_n \implies \neg Z_{n+1} \))

- If Base and Induction are each unsatisfiable, property is true
- Else, need to increase induction step (next slide)
Induction-based (Simple Case)

- Simple induction may be too simple

Solution:
- Require $Z$ to be true for more than 1 cycle
Induction-based (k-case)

- Recall: $U_k = C_0 \land C_1 \land \ldots \land C_{k-1}$
- Two formulas to check:
  - Base case: $I_0 \land U_k \land Z_k$ is unsatisfiable
  - Induction step: $\neg Z_0 \land U_k \land Z_k$ is unsatisfiable
    - (i.e. $\neg Z_0 \Rightarrow \neg Z_k$)

- If both are valid, then $\neg Z$ always holds
  - If Base is satisfiable, found a counter-example
  - Else, increase $k$ and try again
History

- 1980: [Clarke, CMU] Theory for formal verification for parallel programs
- 1990: [Burch, CMU] Academic implementation of formal verification for hardware designs, using BDDs
- 1995: First commercial formal verification: AT&T FormalCheck, IBM RuleBase
- 1999: [Sheeran, Chalmers] SAT Induction-based model checking
- 2001: [Moskewicz, Princeton] Fast SAT solvers
- 2002: [Chauhan, CMU] Automatic abstraction refinement, commercialized in Synopsys Magellan
- 2003: [McMillan, Cadence] SAT Interpolant-based model checking
- 2003: Formal verification for block-level: Jasper, Synopsys Magellan
- 2005: Assertion/property languages standardized: PSL, SVA
- 2011: [Bradley, Middle School Teacher] Property Directed Reachability
Awards

- ACM Turing Awards
  - Amir Pnueli (1996)
    - Temporal Logic
  - Edmund Clarke, Allen Emerson, Joseph Sifakis (2007)
    - Model Checking
  - Leslie Lamport (2013)
    - Liveness and Safety Proofs in Distributed Systems

- Annual Model Checking competition
  - Winners are from leading universities
Commercial Practice

- **First-gen tools (1995):** AT&T FormalCheck and IBM RuleBase
  - Needed significant user expertise, required design modification

- **Second-gen (1998 – 2006):**
  - Mostly bug-finding, used for local checks
  - Averant, Real Intent, o-In, Magellan

- **Third-gen (2003 – 2010):**
  - Assertion-based verification
  - Jasper (acquired by Cadence in 2014 for $170M)
  - Coverity (acquired by Synopsys in 2014 for $375M)

- **Fourth gen (2007-):**
  - Replace traditional verification methods by abstraction techniques
  - Oski Technology
Open Research Problems

- Implement abstraction refinement over proof engines
  - Allow arbitrary pairing of refinement algorithm with proof engine

- Automatic (or semi-automatic) methods for new abstraction
  - Possibly based on high-level design information: automatically extracted, or user-specified
Summary

- Formal verification is practical
  - but, for well-selected problems...

- Successful application requires
  - Formal testplanning
  - Dedicated resources
  - Expertise for verification strategy

- Watch out for commercialization of
  - Formal methods for coverage closure
  - Constraint specification

- Open problems in
  - Integration, and co-operation of proof engines
  - Automation in abstraction and abstraction-refinement methods
Oski Takes You on a Path to a Distinguishing Career

Oski

The leading company in formal verification methodology

We work with leading-edge semiconductor companies, NVIDIA, Samsung, Huawei, Broadcom, Qualcomm ...

Most formal experts in industry work at Oski, solving customers’ verification challenges together

You

Will become a formal expert (< 100 in the world), setting yourself apart to a distinguishing career

Will have the opportunity to work with smart people around the globe, contributing to the development of latest technology

Will be in good hands learning, growing and succeeding in your first job
Oski Technology Recruiting at SNU

- Want to hire 1 or 2 very talented engineers

Requirements
- Love problem solving
- Like working with others
- Not scared of open problems and unknowns
- Likes travel (US, India, Taiwan, UK)

Benefits
- Become expert in an area with less than 100 experts in the world!
- Overtake engineers with 20+ years of experience!
- Attractive compensation, and shares in a US startup

If yes, come at 16:00 today to:
- Lotte International Education Hall (Bldg# 152-1) Suite #207
- Will have to take a mathematics exam
- Followed by a problem-solving interview, today or later