What is RAPID?
- Hardware-software co-design initiative in Oracle Labs
- Goal: improve energy efficiency of database-processing systems
- In this context, we are verifying the RAPID System On Chip

What was the primary verification methodology for RAPID?
- UVM-based verification environment for units and SoC

Why did we explore the use of formal model checking?
1. To meet our project SCHEDULE
2. To ensure design QUALITY

What was our prior experience using formal methods?
- NONE

How long did this entire effort take?
- 7 person months

What were our initial expectations?
1. Verify some SoC-level connections
2. Explore if formal could help verify a couple of simple units

What did we achieve?
- SCHEDULE: We exceeded our goals for units targeted with formal verification
- QUALITY: These units were fully functional in first silicon

Formal Strategies

State Exploration with Formal Model Checking

Formal Assurance
- Objective: to completely verify the unit
- Properties cover entire state space
- Replaces simulation for verifying unit
- Challenge: How do you know you have adequate properties?

Bug Hunting with Formal Model Checking
- Objective: to find bugs – especially the elusive ones
- Properties target corner cases and key features
- Complements simulation for verifying unit
- Can be applied Early (pre-simulation) and Late (post-simulation/coverage)

Evolution of the RAPID Formal Plan

SoC Connectivity proofs
- Interrupts
- Routing of events to counters
- DFT signals
- Simple connection (or conditional) properties
- Important to blackbox units not relevant to connectivity for productivity

Formal assurance
- System Interrupt Controller (SIC)
- Event Count Monitor (ECM)
- Least Recently Used Arbiter (LRU)
- SRAM controller (SCR)
- These units were of low to moderate complexity
- Behavior and properties could be described thoroughly for unit
- For each unit, the set of assertions and assumptions were reviewed by designers and peers for thoroughness

Deeper Formal proofs
- Fuse Controller (FUSE)
- Deep state machine (>500 cycles)

Early Bug Hunting
- Memory Interface System (MIS)
- System already verified using UVM testbench
- Objective to verify connectivity
- Properties to cover key invariants and corner case behaviors
- Clock Control Unit (CCU)
- Bug found in random SoC simulation for a clock mode
- Simulation of all modes was not practical
- Formal model checking was used to find other clock modes for which bug may occur

Late Bug Hunting
- Memory Interface System (MIS)
- New features subject to formal model checking prior to simulation
- Bugs found and debugged faster
- Assertions with bounded proofs promoted to simulation

Design for Formal Verification
- SRAM Parity Error (SPE)
- Error collection and propagation architecture and design altered to make it formal assurance friendly
- Significant productivity savings over simulation based verification
- Truly rapid time to verification closure

Organizational Capability
- Developing skills across RAPID DV team
- Targeting more units across the project
- Formal verification is now a consideration during verification planning for most units

Examples from RAPID Assertion Library
- To simplify the assertion specification, we developed a simple Python template for properties and generated SVA from them

Python
- def_connect
- def_imp_range
- def_eventually
- def_cond
- def_until

SVA
- A, delay
- A, B, delay
- A, B, C, delay

Results

UNIT | STRATEGY | REGISTER COUNT | ASSERTION COUNT | RUN TIME | BUG COUNT
--- | --- | --- | --- | --- | ---
SIC | Assurance | ~4200 | ~3000 | 3m | 5
ECM | Assurance | ~5500 | ~1500 | 5m | 15
LRU | Assurance | ~75 | ~60 | 120m | 2
SRC | Assurance | ~2500 | ~1500 | 600m | 39
FUSE | Bug Hunting | ~4500 | 2 | 110m | 4
MIS | Bug Hunting | ~7000 | ~2100 | 720m | 8
SPE | Assurance | ~350 | ~150 | 2m | 8

The key to an optimal SoC verification plan is a balanced application of Formal and Simulation methods

FORMAL
- Excellent for Unit Verification
- Excellent for System Verification and units not suited for formal model checking
- Can replace simulation for units verifiable with Assurance Strategy
- Simulation still needed to verify system behavior and interoperability

SIMULATION
- Early Bug Hunting can accelerate time to productive simulations
- Late Bug Hunting can help find critical corner case bugs
- Simulation is primary verification method for these units
- Debugging is very precise
- Debugging may take longer. Assertions will help accelerate debug
- Not applicable for system verification

Formal + Simulation
- Necessary for coverage sign-off when not using assurance
- Challenges: How do you know you are done?

Conclusion
- Perception?
  - Formal is only for experts
  - Needs special skills and talent
  - Is hard to use and deploy
  - Is impractical for real world designs
  - Needs a lot of effort before benefits can be reaped

- Reality
  - Needs planning, diligence and persistence
  - Capability can evolve from modest beginnings
  - Targeted strategy will produce immediate results
  - Leverages and infuses fundamental Assertion Based Verification principles

Acknowledgements
- Thanks to the RAPID team for their enthusiastic participation
- Thanks to Roger Sabbagh for his persistent encouragement
- Thank you for visiting this poster!