End to End Formal Verification Strategies for IP Verification

Jacob Ryan Maas, Nirabh Regmi, Krishnan Palaniswami, Ashish Kulkarni

Microsoft Corporation

Introduction

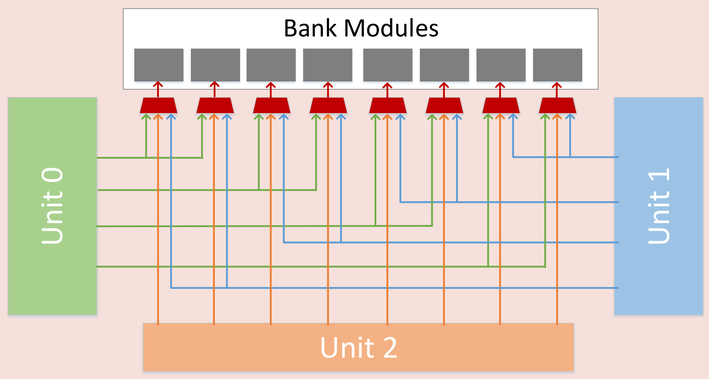
In End to End (E2E) formal verification, IPs are described fully using formal properties based off the IP specification. Registers are checked with automatically generated assertions, from an IPXACT XML file. Standard interfaces are checked with formal property proof kits, instead of VIP. The quality and completeness of these assertions is checked by frequent reviews against the IP specification with the design team. Once all formal properties are signed off by design, and all assertions and proof kits have been proven, the IP is then considered fully verified.

E2E formal verification does not carry the overhead that comes with writing UVM testbenches, such as the authoring of custom agents, sequence libraries, and scoreboards. Also IP specification is available before RTL is delivered, allowing DVEs to get a head start on developing the verification environment.

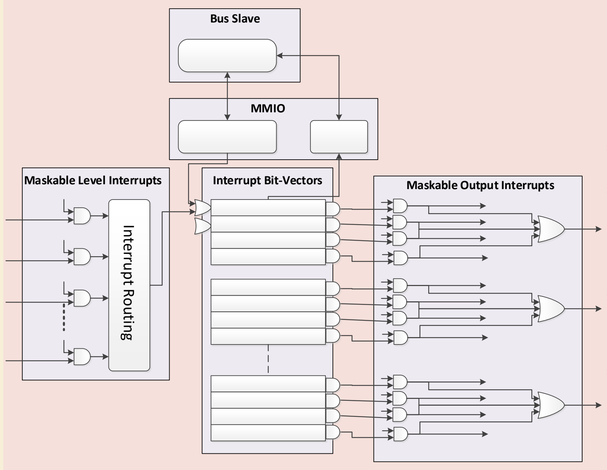
Careful appraisal of compatibility with the methodology is critical when selecting an IP for formal verification. Generally, IPs need to be small and relatively simple. Two IPs: Interrupt Controller (INTC) and Crossbar (XBAR) were verified using this methodology.

Block I: Crossbar

- Controls access to shared memory space from three different requesters.
- Supports round robin, fixed, and LFSR priority schemes.
- Single requestor may lock exclusive access to a channel.
- Most complexity contained in “Bank” arbitration logic.



Block II: Interrupt Controller



- May accept level or edge interrupts.
- Routes incoming interrupts to flag registers, which are collapsed into single output interrupt lines.
- Output interrupt lines may be set and cleared by the bus, allowing SW controlled interrupts.
- Input and output interrupt lines may be gated.
- Highly symmetric logic.

Challenges

Quality: Since most EDA tools did not have good coverage reporting tools, extra effort had to be made to ensure that all features/aspects of the design were covered by the written assertions.

Verification IP: Not all EDA tools had proof kits for most standard interfaces and configurations.

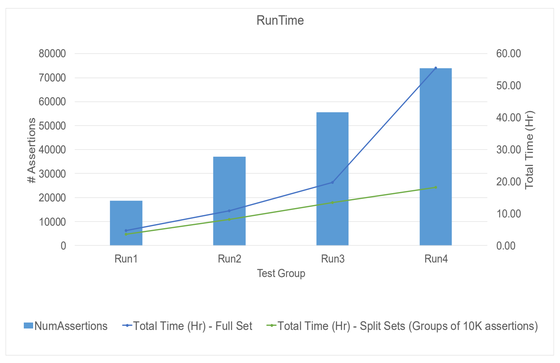
Convergence: Templated assertion generation was used for the INTC due to its high symmetry, which ultimately produced 420K assertions and led to very long compile times. By dividing the testbench into to sets of 10K-40K assertions, total runtimes fell to well within practical limits.

Reporting: At the time of its development, the formal tools available did not provide coverage metrics that could be easily compared to simulation coverage metrics.

Strategies

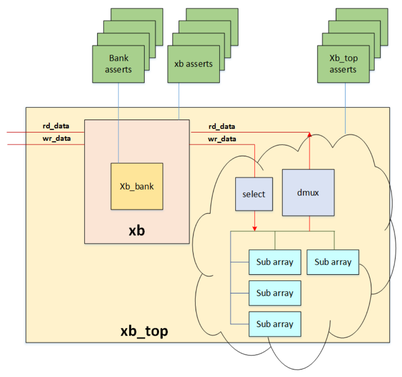
Divide And Conquer

- Runtimes measured with four sets of assertions, with dividing and without.
- Even in smaller sets, Divide and Conquer strategy yields => 24% reduction in runtime.
- Runtime benefit of Divide and Conquer has nonlinear growth with respect to assertion count.



Hierarchical Strategy

- Testbench is organized such that identical block instantiations are verified only once.
- Unique instantiations and block connections verified by parent level assertions.
- Hierarchical Strategy saw earlier sign-off by reducing DVE workload.



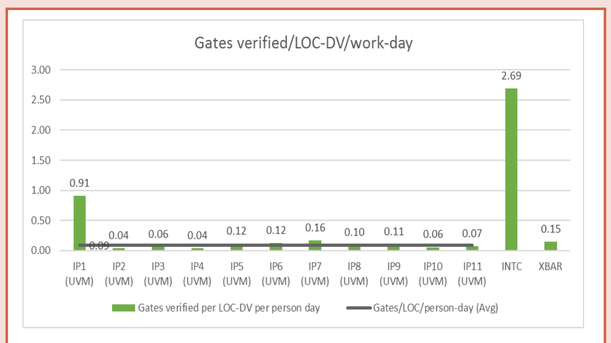
Abstraction
Large families of assertions to be reduced to equivalent, single assertions with limited addition of Verilog in the glue logic. This reduction in property count significantly improved runtime, despite checking identical functionality.

Conventional Methods
Tool specific optimizations, constraining inputs.

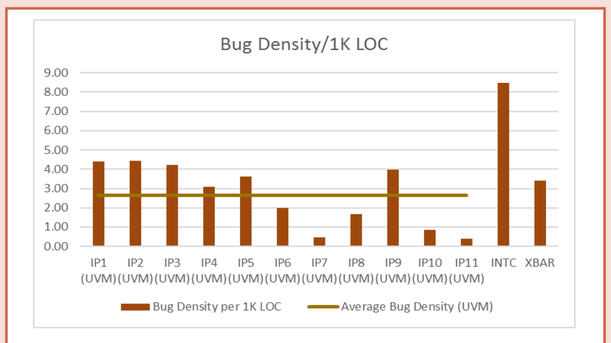
Results

To quantify and analyze the quality and ROI on E2E formal verification methodology and to normalize the complexity of various IPs verified by different methods, the following metrics were considered:
1. Gates verified per LOC-DV (Lines of Code—Design Verification) per day
2. Bug density per 1K LOC-DV
3. Bug escaped per IP testbench

Testbench	GateCount	DV Resources	Duration (Work days)	LOC-DV	Total Person-Days	Gates verified per LOC-DV per person day	Bug Escape	Bugs Found per 1K LOC
IP1 (UVM)	3170202	2	178	9795	356	0.91	1	4.39
IP2 (UVM)	381608	2	182	24725	364	0.04	0	4.45
IP3 (UVM)	1255701	3	215	33311	645	0.06	2	4.23
IP4 (UVM)	75593	1	156	12564	156	0.04	0	3.10
IP5 (UVM)	108912	1	142	6630	142	0.12	0	3.62
IP6 (UVM)	541479	1	172	25729	172	0.12	0	1.98
IP7 (UVM)	299693	1	146	12488	146	0.16	0	0.48
IP8 (UVM)	129607	1	151	8914	151	0.10	0	1.68
IP9 (UVM)	89230	1	109	7786	109	0.11	7	3.98
IP10 (UVM)	41387	1	127	5873	127	0.06	0	0.85
IP11 (UVM)	486092	1.25	147	35376	183.75	0.07	0	0.40
INTC	304174	1	87	1300	87	2.69	0	8.46
XBAR	61118	1	82	5000	82	0.15	1	3.40



Based on the gates verified per LOC-DV per person-day, it can be inferred that equal, if not greater, efficiency can be achieved using formal E2E compared to simulation/UVM methodologies.



Formal E2E can find more bugs per LOC-DV than UVM testbenches, primarily due to the fact that only assertions and minimal glue logic (if any) are needed for the formal testbench.

From the experiment it is clear that formal E2E testbenches can achieve the same quality of IP verification as UVM testbenches in terms of bug escapes.

Advantages

Even though E2E formal methodology requires additional time reviewing assertions, savings were realized through:

Testbench Coding: Less code required compared to UVM testbench. Testbench is ready on first day of RTL delivery. First RTL bugs can be discovered as soon as the first RTL release.

Coverage Closure: Proofs are exhaustive and thus there is no coverage closure step.

Regressions: Random testing and nightly regressions are not required due to exhaustive proof.

Bug Reproduction: Fail sequences are available as soon as proofs complete, so debugging can start immediately without waiting to rerun a failing test as in simulation.

Simplified Testbench: Repeated patterns, symmetries, and reused sub-blocks allow for simplification of the testbench using the strategies described earlier, saving runtime without having to compromise on the quality of verification.

Recommendations for Deploying E2E Formal Verification

IP Selection: Any IP which can be fully specified in SVA is generally a good candidate. Ideal candidates should have a shallow cone of logic, low gate count, and low complexity.

Enforcing Assertion Quality: At the time of our research, existing formal tools did not offer a solution for coverage tracking similar to that of simulation based metrics. Without a coverage solution, extensive reviewing between DEs and DVEs must be done to enforce testbench quality.

Aforementioned Strategies: We also recommend employing the aforementioned strategies when deploying E2E formal verification, on IPs where those strategies apply. Use of these strategies reduces assertion counts and runtimes, without compromising the quality of the verification.

Conclusion

End-To-End Formal Verification is well suited for certain types of highly symmetric and control path intensive designs. This methodology is worth the savings realized in verification effort and resources, and is able to achieve at least the same quality of verification compared to UVM testbenches. It definitely provides some compelling cost benefit against traditional simulation/UVM testbenches.

Microsoft