2019 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

# A Coverage-Driven Formal Methodology for Verification Sign-off

Ang Li, Hao Chen, Jason K Yu, Ee Loon Teoh, Iswerya Prem Anand

NVM Solutions Group

Intel Corporation

accellera SYSTEMS INITIATIVE

2019 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

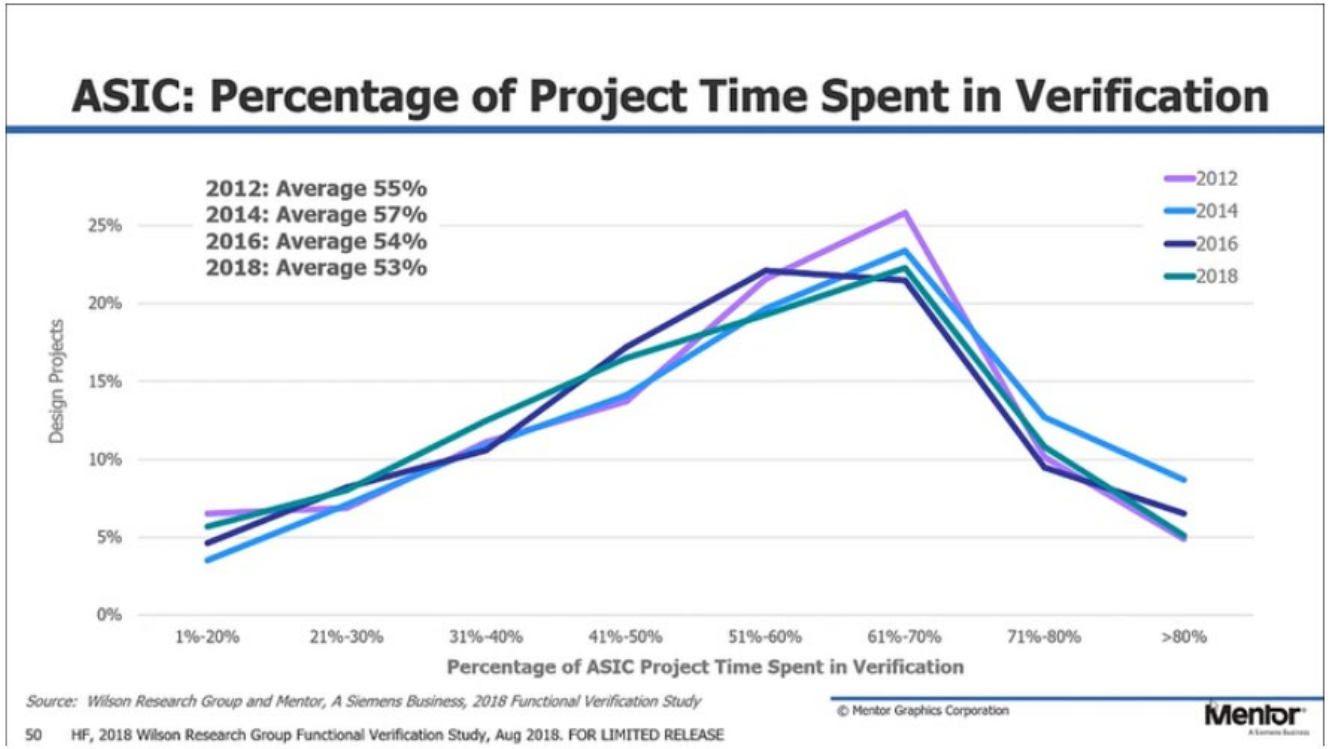
# Agenda

- **Introduction**
- Formal Coverage
- Verification Plan
- Coverage-driven Formal Signoff Flow
- Real-World Results: Micro-Sequencer Design
- Conclusion

accellera SYSTEMS INITIATIVE

# SoC Verification is Challenging

- SoC verification is full of hard problems:
  - Exhaustive corner cases at block level
  - FW/HW interaction at subsystem level
  - 3rd party IP integration at SoC top level
  - …
- Therefore SoC verification is time/resource intensive




Courtesy: Harry D. Foster “The 2018 Wilson Research Group ASIC and FPGA Functional Verification Study” [11]

# Formal Indeed Can Help, But

- Most verification teams still choose dynamic simulation for verification signoff on formal friendly blocks
- We believe the lack of a measurable formal verification flow with equivalent signoff metric is one major barrier to adopting FV exclusively for verification signoff

**Our verification engineers often ask:**

*"How do I know what has been verified in formal?"*

*"How do I know if I have enough assertions?"*

*"how do I know when a block verified using formal is ready for signoff?"*

**Our verification managers often ask:**

*"How can I monitor verification progress when a block is verified by formal?"*

*"Will formal cause coverage holes in my overall verification architecture?"*

*"Will formal cause reusability issues with class-based systemverilog testbenches?"*

# How to Gain Mainstream Acceptance for Formal Verification?

**Solution:**

A coverage-driven formal verification methodology with simulation-compatible signoff metrics

2019 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

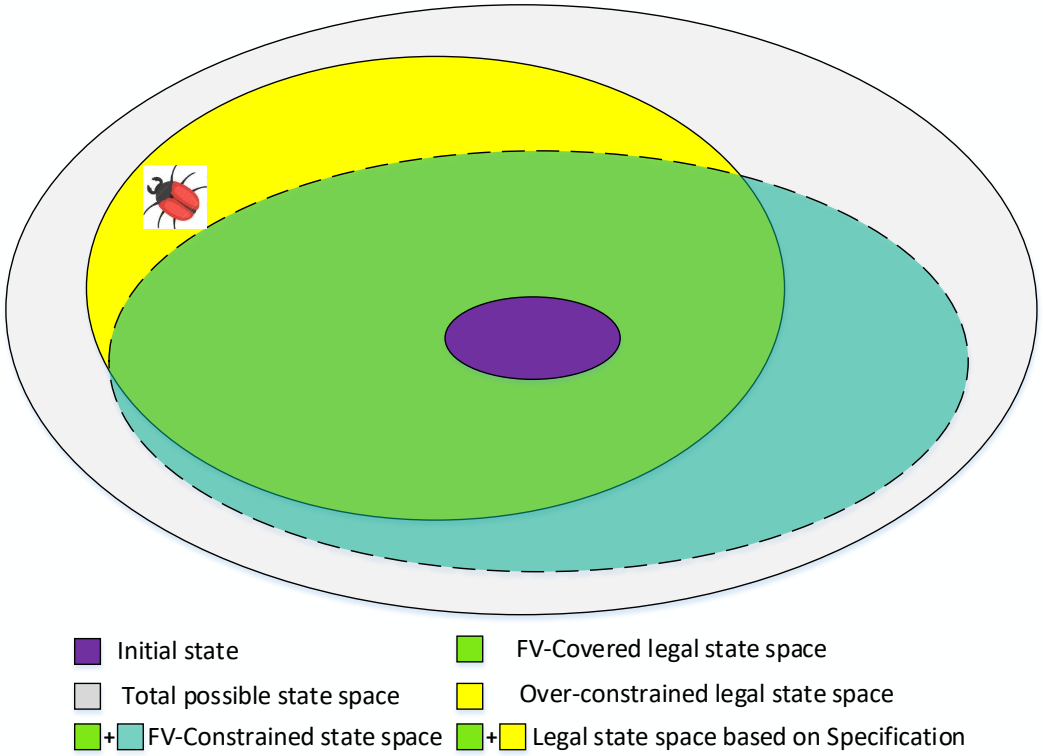
# Agenda

- Introduction
- **Formal Coverage**
- Verification Plan
- Coverage-driven Formal Signoff Flow
- Real-World Results: Micro-Sequencer Design
- Conclusion

accellera SYSTEMS INITIATIVE

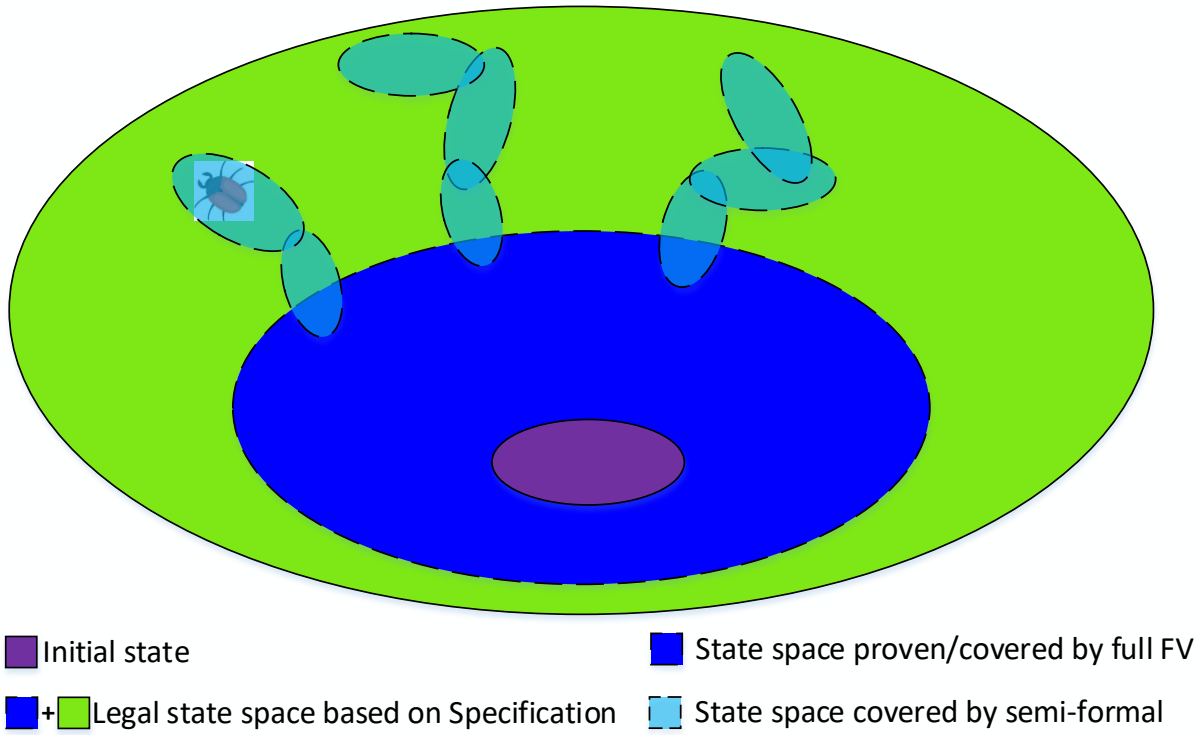
# Why is Formal Coverage Important?

- Reason #1: Uncovers over-constraints
  - Unintentional over-constraints make legal design state space unreachable
  - Solution: RTL stimuli coverage and functional coverage



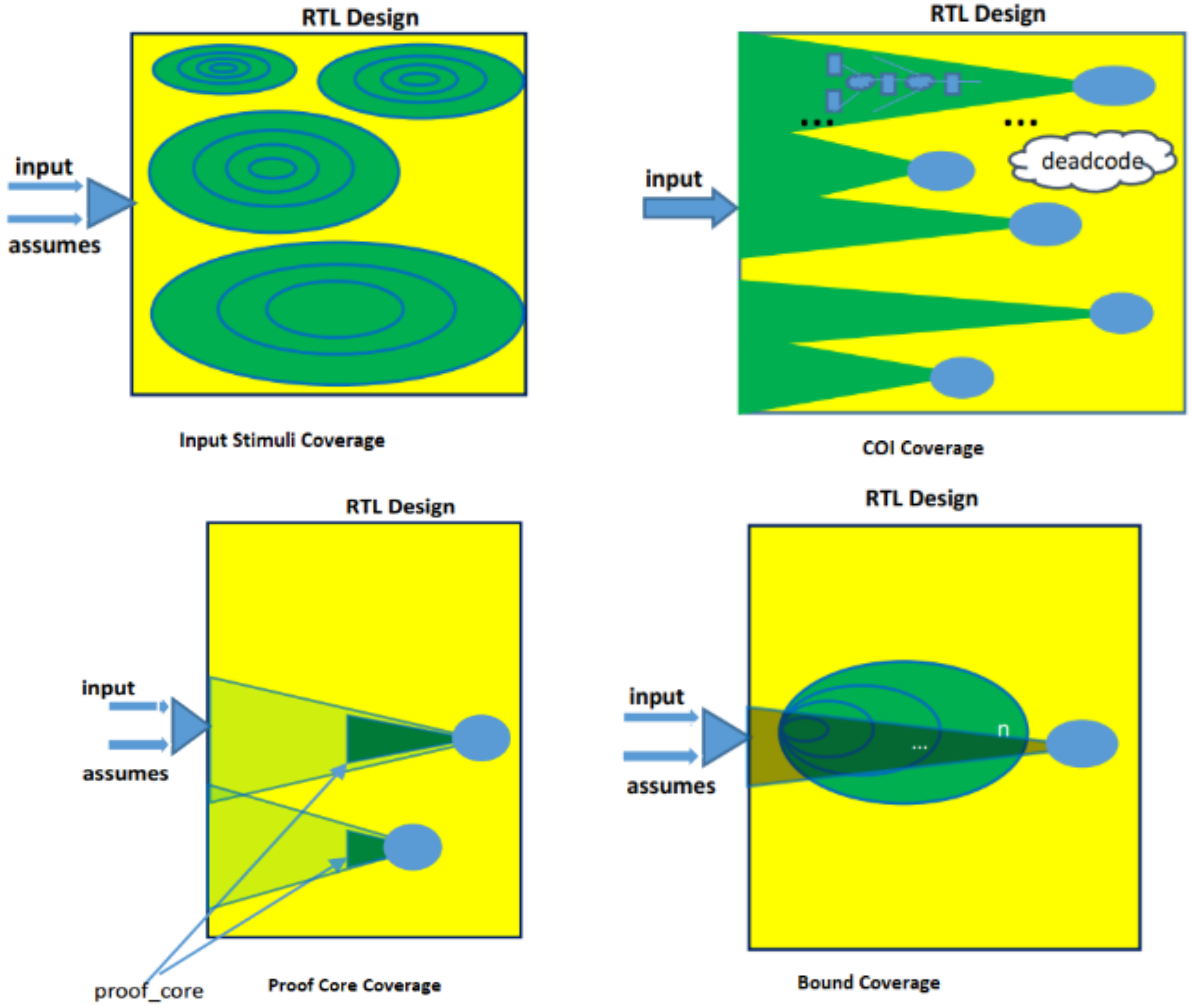
# Why is Formal Coverage Important?

- Reason #2: Validates Bounded Proofs
  - Bounded proofs are often inevitable
  - Solution: bounded proof coverage and functional coverage



# Formal Coverage Types

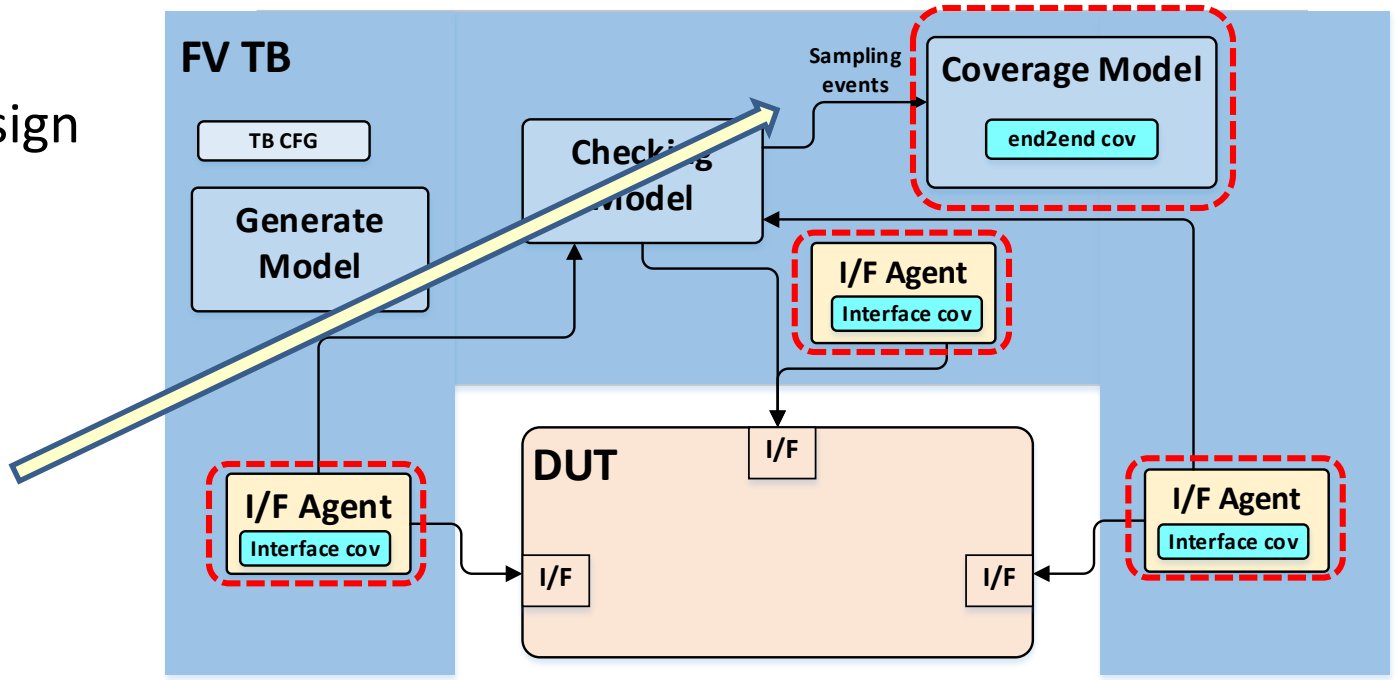
- Automatic code coverage:
  1. **Input stimuli coverage**: determine reachable design state space under input constraints
  2. **Cone of Influence (COI) coverage**: determine how completely your assertions cover the design
  3. **Proof core coverage**: determine proven state space covered by your assertions
  4. **Bound coverage**: check the quality of the bounded proof




Courtesy: X. Feng (Oracle), X. Chen, A. Muchandikar(Synopsys)
"Coverage Models for Formal Verification", DVCon 2017 [8]

# Formal Coverage Types Cont'd

- Functional coverage:
  - User-defined coverage based on design specifications
    - Configuration register values
    - I/F transactions
  - Used as a proof of feature checked
  - Defines a minimum set of exercised design behaviors required for signoff



# Formal Signoff using Coverage

- All types of formal coverage are used together to achieve verification signoff

| FV Signoff Checklist | Stimuli Coverage | COI/Proofcore Coverage | Bound Coverage | Functional Coverage |
|---|---|---|---|---|
| Is my formal testbench over-constrained to disallow legal stimuli? | √ | | | √ |
| Are my checkers complete? | | √ | | |
| Are my bounded proofs reaching sufficient depth? | | | √ | √ |
| How much of the design has been formally verified? | √ | √ | | √ |

# Agenda

- Introduction
- Formal Coverage
- **Verification Plan**
- Coverage-driven Formal Signoff Flow
- Real-World Results: Micro-Sequencer Design
- Conclusion

# Functional Requirements in Vplan

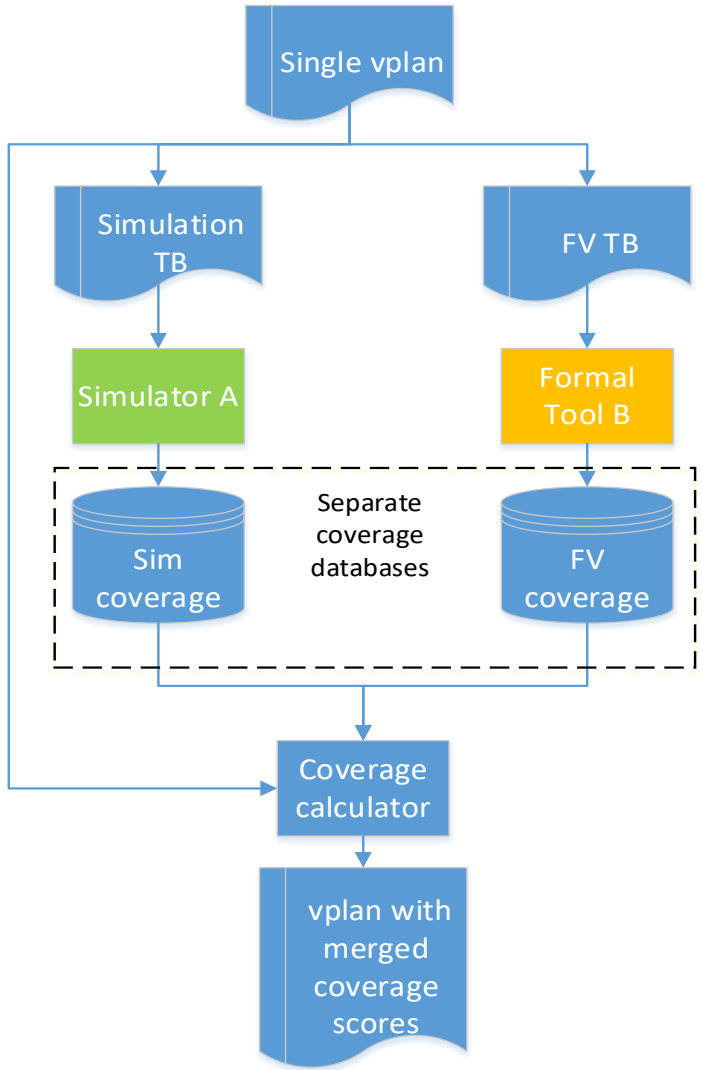
- Verification plan (vplan)
  - set clear goals using measurable metrics
  - a prioritized list of functional requirements

| Requirement Type | Description | Implementation Method in FV |
|---|---|---|
| **Generate** | defines stimulus generation | **assume** properties + SV modeling code |
| **Check** | defines behavioral checks | **assert** properties + SV modeling code |
| **Cover** | defines design behaviors that need to be covered during verification | **cover** properties + **covergroups** + SV modeling code |

# Formal and Simulation Co-Verification

- Feature-based verification experiment with formal and simulation

| VPLAN | | | | |
|---|---|---|---|---|
| FEATURE | REQUIREMENT ID | REQUIREMENT TYPE | METHOD | COVERAGE SCORE |
| Feature 1 | FEATURE1.GEN.01 | Generate | Simulation | |
| Feature 1 | FEATURE1.CHK.01 | Check | Simulation | |
| Feature 1 | FEATURE1.COV.01 | Cover | Simulation | 80% |
| Feature 2 | FEATURE2.GEN.01 | Generate | Formal | |
| Feature 2 | FEATURE2.CHK.01 | Check | Formal | |
| Feature 2 | FEATURE2.COV.01 | Cover | Formal | 100% |



2019 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

# Agenda

- Introduction
- Formal Coverage
- Verification Plan
- **Coverage-driven Formal Signoff Flow**
- Real-World Results: Micro-Sequencer Design
- Conclusion

accellera SYSTEMS INITIATIVE

# Stage 1: Planning Stage

- Create vplan based on design spec:
  - Generate requirement
  - Check requirement
  - Cover requirement

# Stage 2: Testbench Coding Stage

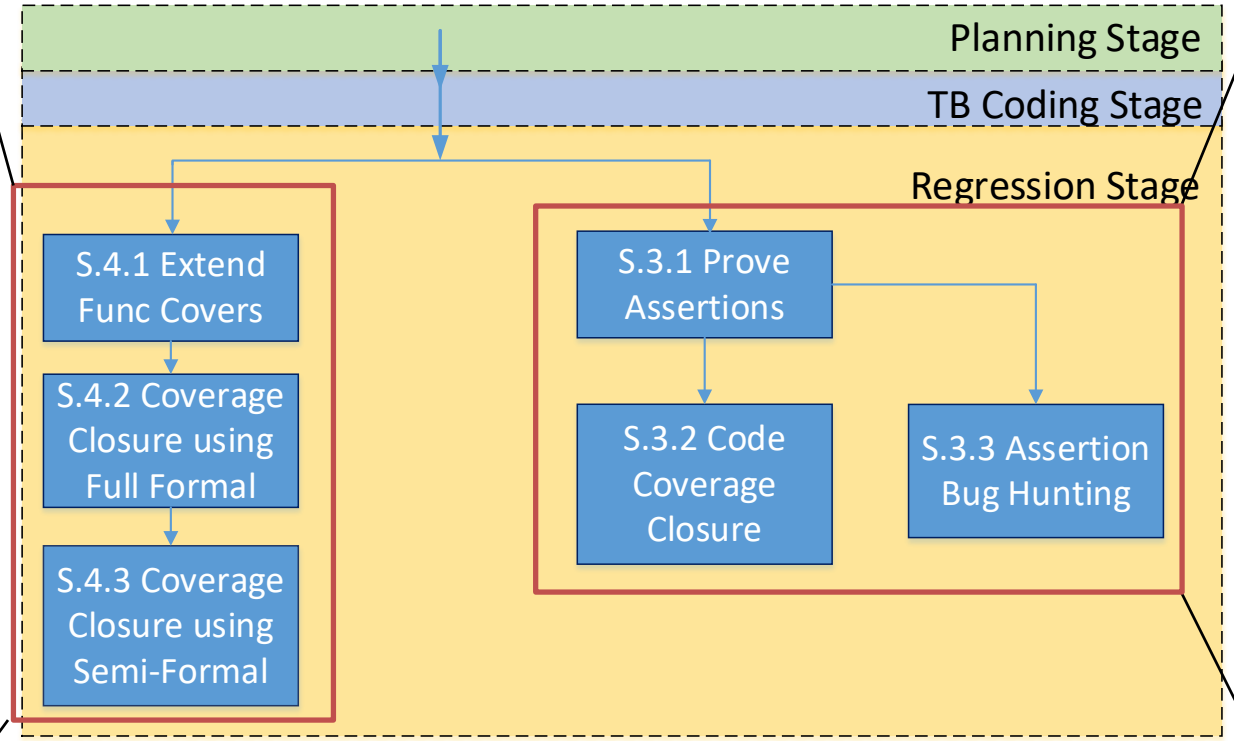
Implement FV testbench:

- **Generate** requirement -> generate model, I/F agent
- **Check** requirement -> e2e checking model
- **Cover** requirement -> coverage model

# Stage 3: Regression Stage

**Coverage closure regression**

- Regression with coverage model enabled
- Purpose is to cover required design state space
- Assertions are evaluated in cover traces

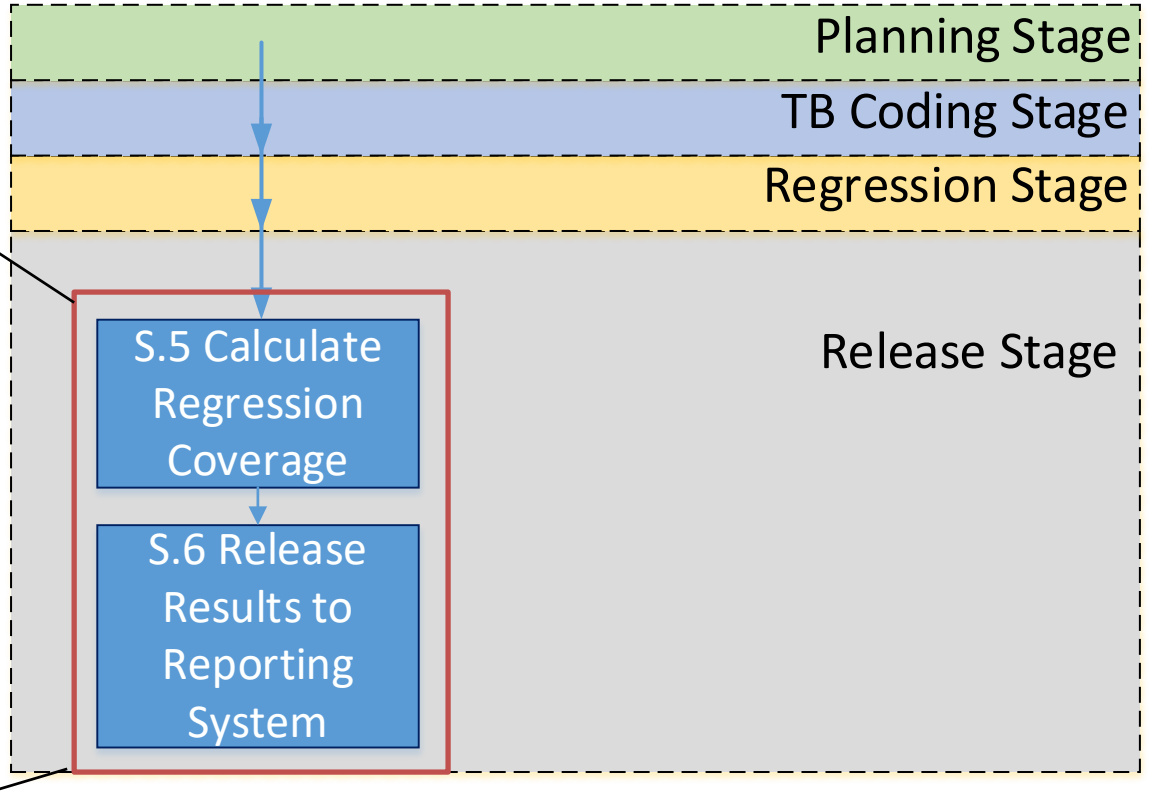



**Assertion proving regression**

- Regression with coverage model disabled
- Purpose is to prove all assert properties
- Code coverage analyzed on top of assertion proof results

# Stage 4: Release Stage

- Two metrics to track progress:
  - ✓ Implementation status for all type of requirements
  - ✓ Coverage scores for **cover** requirements
- Coverage scores are derived from percentage of cover points reached/exercised



# FV Signoff Criteria

- Verification signoff for the design can be declared when the following criteria are met:

1. 100% functional coverage reached. No assertion failures evaluated in any functional cover trace
2. 100% COI and Reachability coverage reached with waivers (for dead code, etc.)
3. All assertions are either fully proven or bounded proven with sufficient bounds
4. No conflict in FV constraints

2019 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

# Agenda

- Introduction
- Formal Coverage
- Verification Plan
- Coverage-driven Formal Signoff Flow
- **Real-World Results: Micro-Sequencer Design**
- Conclusion

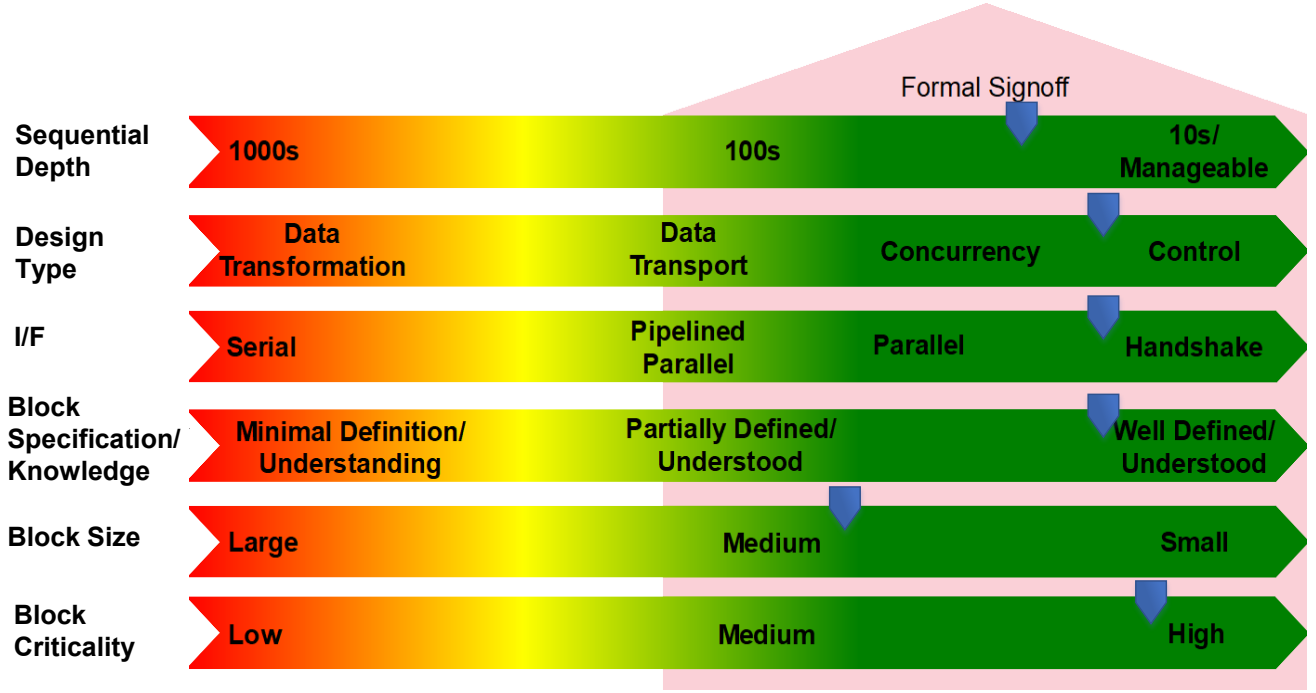
accellera SYSTEMS INITIATIVE

# FV Signoff Block Background

- Formal signoff for a complex micro-sequencer block

| Micro-sequencer Design Info | | |
|---|---|---|
| # of Requirement | Flop Count | Line of RTL |
| 104 | 16K | 4K |

- This block is a highly concurrent control-oriented design
- Hard to exhaustively simulate all corner cases in the past
- ROI results obtained are shown in following slides




Courtesy: C. Komar, M. Diehl, “Optimizing IP Verification – Which Engine?”, DVCon 2017 [12]

# Design Spec Improvement

- FV necessitates very accurate and complete design specifications.
  - For the micro-sequencer design we verified in FV, over 70% of the requirements in the specification were improved and clarified. One example:

**Before:**

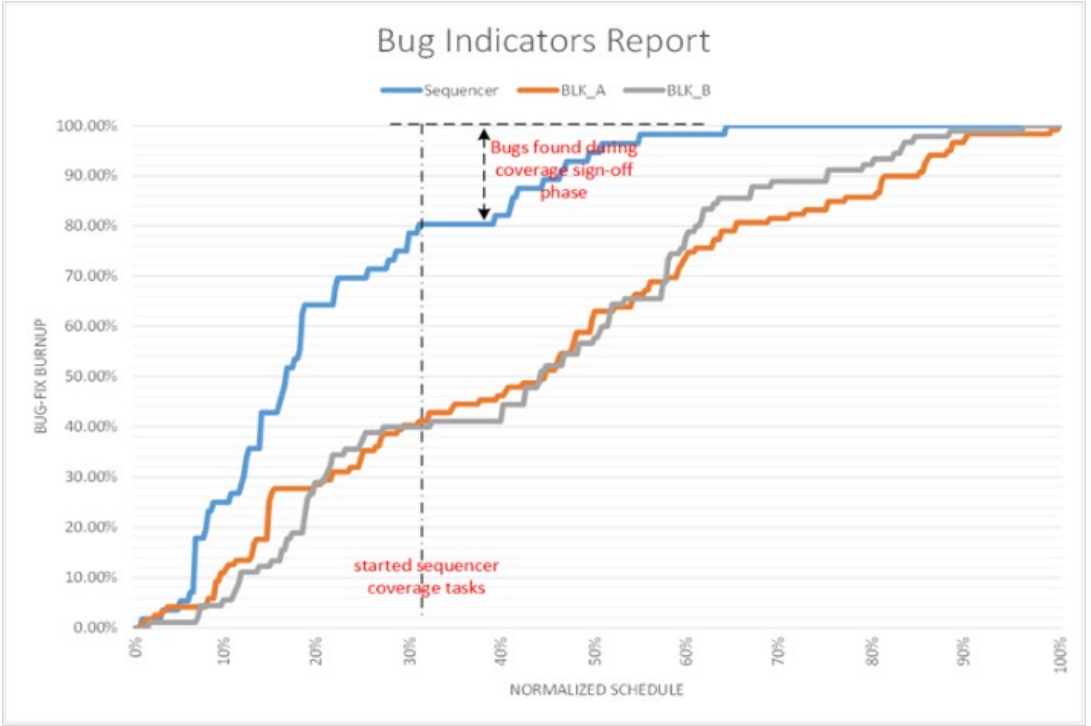
*The micro-sequencer engine shall continuously monitor the state of all input signals defined in the registers T_Array, and use the settings in the V_Array registers to determine which, if any, inputs are currently considered Active. If an input is considered Active, the V_PENDING field of the corresponding V_Array register shall be set.*

**After:**

*The micro-sequencer engine shall continuously monitor the state of all input signals defined in the registers T_Array, and use the settings in the V_Array registers* ***(<input_port_name>.v_array[i])*** *to determine which, if any, inputs are currently Active.* ***An input is "Active" when the conditions specified by V_COND ((<input_port_name>.v_array[i].v_cond) and V_BHV (<input_port_name>.v_array.v_bhv) are met (v_act[i] shall be pulsed in the same clock cycle).*** *If an input is Active, the V_PENDING field of the corresponding V_Array register* ***(<input_port_name>.v_array[i].v_pending)*** *shall be set* ***in the next cycle if the microsequencer is enabled (seq_en = 1)****.*

# Quality of Bugs Discovered

- In terms of bug finding:
  - **32** bugs found in total
  - **5** high quality bugs that mostly led to re-architecture of the design
  - **6** bugs found during coverage closure phase
  - Comparative study shows the debug cycle is **30%** shorter than similar blocks verified in simulation




| Benchmark Design Info | | | |
|---|---|---|---|
| DUT | # of Requirement | Flop Count | Line of RTL |
| Micro-sequencer | 104 | 16K | 4K |
| BLK_A | 100 | 7K | 16K |
| BLK_B | 103 | 4K | 6K |

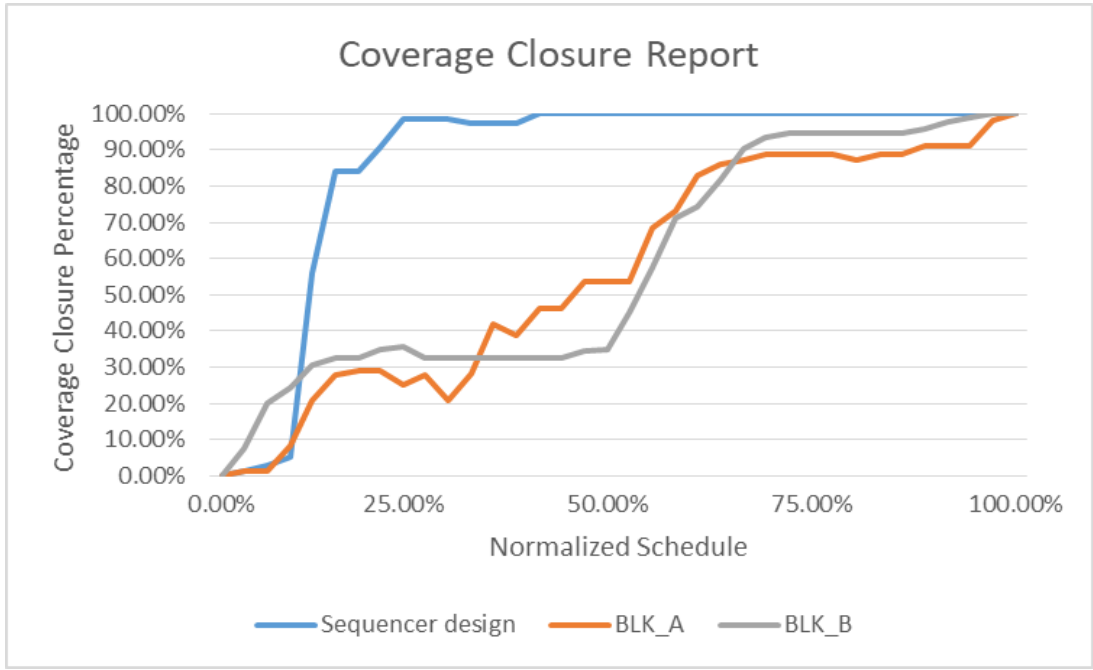
# Execution Time Improvement

- We also found overall execution time improvement using the FV signoff methodology. Two reasons:

1. FV testbench is in general simpler.

2. Coverage closure is in general faster using FV.

| DUT | TB Complexity (line of codes) | Verif Methodology |
|---|---|---|
| micro-sequencer | ~4,000 | FV |
| BLK_A | ~20,000 | Simulation |
| BLK_B | ~13,400 | Simulation |



2019 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

# Agenda

- Introduction
- Formal Coverage
- Verification Plan
- Coverage-driven Formal Signoff Flow
- Real-World Results: Micro-Sequencer Design
- **Conclusion**

accellera SYSTEMS INITIATIVE

# Conclusion

- Formal signoff is REAL!
  - Comprehensive
  - Speedy
  - Efficient
- Methodology is key to wider adoption of FV
  - Deterministic
  - Measurable
  - Repeatable

# References


- [1] Erik Seligman, Tom Schubert, and M V A. Kiran Kumar, “Formal Verification, An Essential Toolkit for Modern VLSI Design,” Morgan Kaufmann, 2015
- [2] Douglas L. Perry, Harry Foster, “Applied Formal Verification,” McGraw-Hill, 2005
- [3] Anish Mathew, “Coverage-Driven Formal Verification Signoff on CCIX Design”, JUG 2017.
- [4] N. Kim, J. Park, H. Singh, V. Singhal, “Sign-off with Bounded Formal Verification Proofs”, DVCon 2014.
- [5] I.Tripathi, A. Saxena, A. Verma, P. Aggarwal, “The Process and Proof for Formal Sign-off: A Live Case Study”, DVCon 2016.
- [6] B.Wang, X. Chen, “Coverage Driven Signoff with Formal Verification on Power Management IPs”, DVCon 2016.
- [7] M A. Kiran Kumar, E. Seligman, A. Gupta, S. Bindumadhava, A. Bharadwaj, “Making Formal Property Verification Mainstream: An Intel Graphics Experience”, DVCon 2017.
- [8] X. Feng, X. Chen, A. Muchandikar, “Coverage Models for Formal Verification”, DVCon 2017.
- [9] J. R. Maas, N. Regmi, A. Kulkarni, K. Palaniswami, “End to End Formal Verification Strategies for IP Verification”, DVCon 2017.
- [10] M. Munishwar, X. Chen, A. Saha, S. Jana, “Fast Track Formal Verification Signoff”, DVCon 2017.
- [11] Harry D. Foster, “The 2018 Wilson Research Group ASIC and FPGA Functional Verification Study”
- [12] C. Komar, M. Diehl, “Optimizing IP Verification – Which Engine?”, Tutorial from Cadence, DVCon 2017.