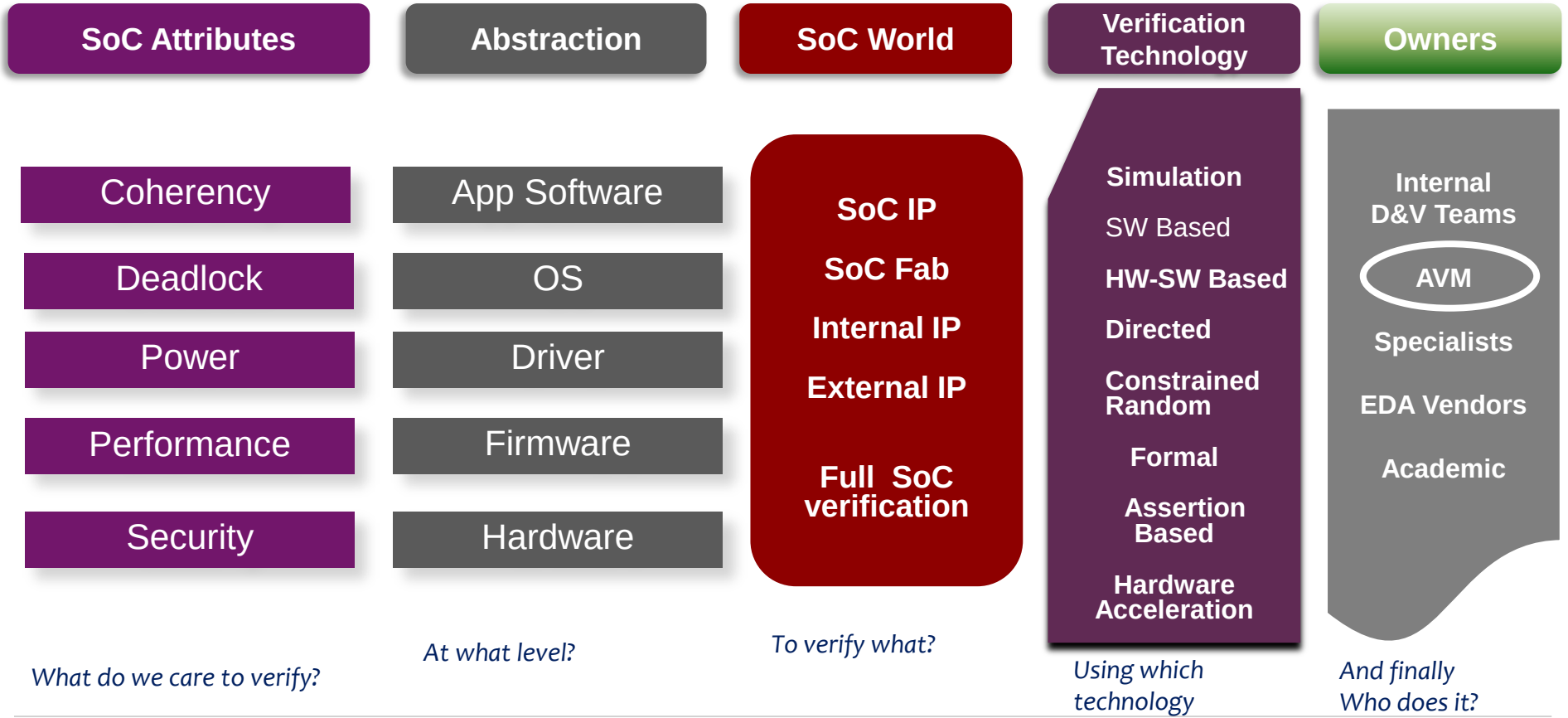
The Ten Myths About Formal

Professor Ashish Darbari
Principal Hardware Design Engineer
Imagination Technologies

www.imgtec.com

V&V Story at Imagination: The view from 30,000 feet



The Advanced Verification Methodology Group

www.imgtec.com

Advanced Verification Methodology Group

Pioneering new ideas and methodologies	**Filed 8 patents ; published 3 papers**
Methodology and Flows	**Several methodology and guideline documents**
Training	**Delivered formal training to nearly 60 engineers**
Verification Roadmaps	**Defined formal and verification efficiency roadmaps**
Project Work and Project Support	**Supported over 25 projects in last 2 years in PowerVR, MIPS and SoC Platform groups**

Formal Technology – Tools and Applications

- **Formal traditionally has three flavours:**
  - model checking
  - theorem proving and
  - equivalence checking
- **Designing formal tools requires background in formal**
- **Application of formal however is not rocket science**
  - Application of formal DOES NOT require a PhD, just requires “common-sense”
- **So why has the adoption of formal in industry always been a challenge?**
  - Why is formal not as popular as simulation or emulation?
- **It is because of the many myths that prevail…**

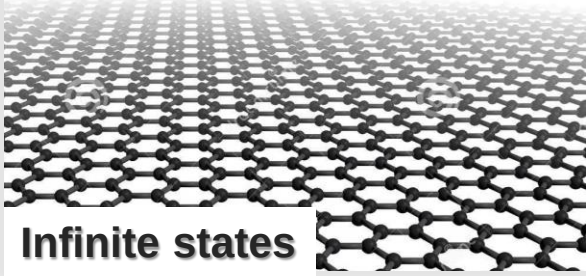
Myth #1

- **Formal requires a PhD**
  - Gradually disappearing but still continues to be around
  - Because there is a poor understanding of “how to use formal effectively”
  - The use of specialist terms and their mathematical notations - engineers not trained
  - The wide gap between high-end academic research and “what engineers care about in industry” doesn’t help either
  - Books, papers, and vendor driven training material help to mitigate this gap; however in some cases can make this even worse

Myth #2

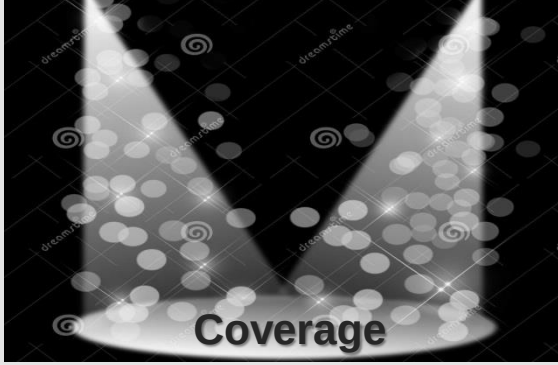
- **Formal is hard because you need specifications**
  - People say formal requires you to write specifications in a specific way
  - You need to know a lot about the design before you can start
  - But this is not true!
- **So why do people say this…**
  - Because people may have not specifications in the first place?
  - Formal blurs the distinction between an informal specification and an executable specification
  - Wrong or incomplete specification – formal highlights a problem pretty quickly – yes even on a design with multi-million flops!
- **Clear, precise, non-ambiguous specifications are a key for all forms of design and verification and yes:**
  - RTL is not the substitute for a specification

Myth #3



- **Formal doesn't scale**
  - A convenient reason for a number of people to not ever try
  - Some of the things people say:
    - Cannot verify a big design with formal
    - Need to choose the right design size
    - Definitely cannot verify a SoC with formal
    - How can you verify all the states exhaustively?
    - Billions of transistors, surely it is impossible!
- **Formal is the “art of possible”**
  - Clear spec, problem solving can get you very far
  - Automated flows can reduce the “perceived” workload

Myth #4

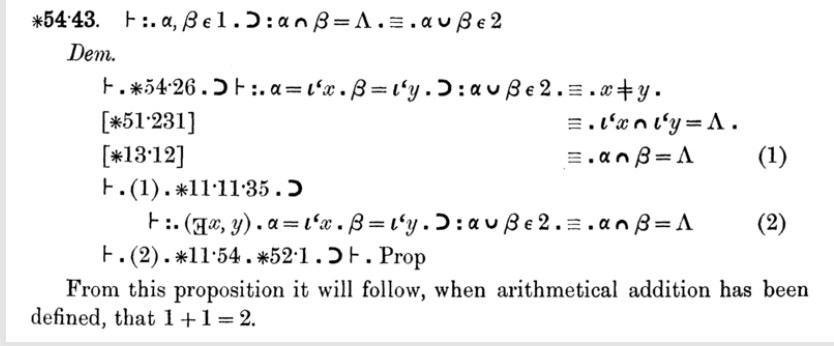


- **Formal doesn't have any coverage**
  - How do we know we have done enough?
  - What about metrics – obsession with “metric driven verification”
  - What can we do with “inconclusive proofs”?
  - What do full proofs tell us?
  - What about constraints?
  - What about completeness?
  - Using formal is dangerous without coverage, there is no coverage in formal
- **Formal tools offer:**
  - Structural – code, toggle, FSM
  - Semantic coverage – trace, cover properties, over-constraint detection
  - Mutation based analysis offers an extra layer of confidence

Myth #5

- **Formal takes a long time**
  - This is another red herring
- **You can get started with very few steps:**
  - Define your clocks and reset
  - Apply your design reset
  - Load an initial state
  - Define effort and solvers
  - Hit the prove command
- **Running the tool itself is also not long specially if you have:**
  - Bugs in the design, or bugs in the test bench (missing constraints, wrong asserts/covers)
  - Full proofs requires ‘appropriate’ formal models; just as getting optimal results in simulation requires optimal simulation model
  - Clear and precise specs hold the key for a quick turn around

Myth #6



$*54{\cdot}43.\quad \vdash :. \alpha, \beta \in 1 . \supset : \alpha \cap \beta = \Lambda . \equiv . \alpha \cup \beta \in 2$

Dem.

$\vdash . *54{\cdot}26 . \supset \vdash :. \alpha = \iota`x . \beta = \iota`y . \supset : \alpha \cup \beta \in 2 . \equiv . x \neq y .$

$[*51{\cdot}231] \qquad \equiv . \iota`x \cap \iota`y = \Lambda .$

$[*13{\cdot}12] \qquad \equiv . \alpha \cap \beta = \Lambda \qquad (1)$

$\vdash . (1) . *11{\cdot}11{\cdot}35 . \supset$

$\vdash :. (\exists x, y) . \alpha = \iota`x . \beta = \iota`y . \supset : \alpha \cup \beta \in 2 . \equiv . \alpha \cap \beta = \Lambda \qquad (2)$

$\vdash . (2) . *11{\cdot}54 . *52{\cdot}1 . \supset \vdash . \text{Prop}$

From this proposition it will follow, when arithmetical addition has been defined, that $1 + 1 = 2$.

- **Formal is only useful for building proofs**
  - This has been one of the oldest myths
  - Perhaps down to the fact that the earliest work on formal was centred on “proofs”
  - Whereas theorem proving is mostly about proofs
  - Model checking and equivalence checking is about “proofs” and “bug hunting”

- **In fact model checkers are fantastic in bug hunting**
- **One way of looking at it is “if the tool cannot find any bugs it will find a proof”**

Myth #7

- **Formal is only useful for finding corner-case bugs**
  - It is great at this
- **But also equally awesome at doing bring up and basic testing:**
  - Can I fill my FIFO?
  - Can I drain my buffer?
  - Can I ever get two mutually exclusive threads to get access to a shared resource when it shouldn't?
- **Why would you write a Verilog/VHDL test bench when you can write a cover property or a simple assertion?**

Myth #8

- **Formal is only useful for verification**
  - Traditionally it has come to this connotation but it is not true at all
- **Lots of uses of formal**
  - Validation
  - Visualising scenarios
  - Design bring up – designers use it for establishing there are no bugs in their block
  - Verification engineers use it for hunting bugs
  - Architects and designers can use formal to specify:
    - interfaces and
    - the finer details of the design and this leads to
    - the specs becoming the executable checks
  - Automated flows: Linters, X-propagation issues, CDC, pin-connectivity, the list goes on…

Myth #9

- **Have done formal, don't need simulation**
  - What an error of judgement?
  - Simulation is necessary to drive all key interface assertions at system level (specially those used as constraints for block level formal)
- **Common mode errors can be eradicated**
  - Re-exercising a few chosen sets of asserts in simulation gives you an extra level of checking
  - In those cases where formal was unable to complete proofs simulation provides a key way to
  - Bias the simulation to hit those scenarios
- **Guarantees obtained from formal need to be revalidated by simulation**

Myth #10

~~Formal~~
Simulation

- **Have done simulation, have 100% coverage, don't need formal**
  - This myth has, and will continue to cause mayhem
  - Simulation cannot find all the bugs
  - Driving complex sequences into designs is a non-trivial exercise and
  - Coverage only gives you a ball-park estimate that design works “somewhat”
    - Not that it always works and is starvation free, deadlock free, livelock-free…
    - No guarantees
- **Get your guarantees whenever you can, simulate if you must**

Summary

- **Verification is “risk management”**
- **Don’t believe in myths**
- **To verify multi-million flop designs to get the right trade-off between quality and time-to-market you need formal:**
  - Early
  - Good Methodology
  - Efficiency and Reuse
- **And think of combining it with simulation**
  - There must be no religion other than rational thinking
  - Need a careful balancing of choices and methodologies
- **Nobody wants a chip come back for post-silicon debug**
- **Though that’s also an area where formal can be used ☺**

Q&A

Thank you!