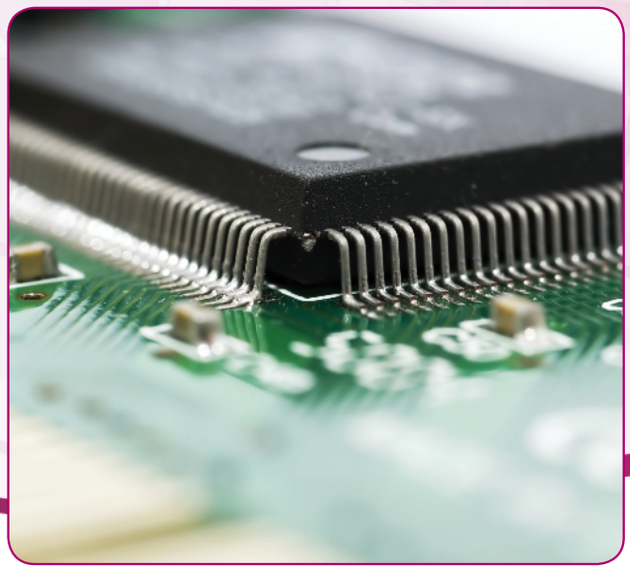
MATHEMATICS MATTERS

Predicting Performance: Mathematical Verification of Chips



The modern world is built on computer chips. From laptops to mobile phones, cars to TV sets, the integrated electronics in these devices power our day-to-day existence. In order to make sure their chips will work properly before they go to market, designers are increasingly turning to mathematics.

In the information age the computer chip is key — the average person will probably utilise the power of many integrated circuits several times before they even leave the house each day. With worldwide demand for their wares, the work of computer chip designers is big business. But, with the cost of making the first prototype of a chip running into the millions, getting it wrong can be an expensive mistake.

Ultimately, chips are fabricated using a collection of photolithographic "masks", which define the various microscopic structures to be laid out on the silicon chip. The cost of each set of masks can run to millions of dollars. And, if an error in logical functioning is discovered in a fabricated chip, tracking down the cause — a process known as "post-silicon debugging" — is very difficult. To have the best chance of getting it right first time, extensive computer simulation is used to check the circuit design before it is converted to masks and fabricated. Up to half of the time spent designing a new chip is dedicated to checking it will work properly.

Such is the complexity of these chips that it's not even remotely feasible to simulate all their possible operations and configurations. A chip stores data as a collection of zeroes and ones and the "state" of the chip describes the overall configuration of these zeroes and ones at any one point. The number of different states a chip could be in vastly exceeds the number of atoms in the universe. Checking them all would take an unimaginably long time. Nonetheless, extensive simulation, with sophisticated strategies to ensure good coverage of the design, is the mainstay of design validation and debugging.

Designers have to box clever, and are increasingly turning to mathematical reasoning to verify key functionalities of the chip. By modelling the design mathematically, large collections of its operating states can be expressed compactly by mathematical formulae. Mathematical proofs can then be used to analyse their correctness. Tom Melham, Professor of Computer Science at the University of Oxford, and a long-time research collaborator with chip manufacturer Intel Corporation, works in this mathematical field of "formal verification".

> "To ensure the best chance of getting it right first time, mathematics is used to verify the logical design of the chip before it is constructed."

The mathematical attack has its limitations too though: the proofs involved are themselves very large. They are practical only with the aid of highly sophisticated algorithms for computer "theorem proving". Unable to reason about the entire chip, verification engineers are reduced to analysing as much of its functionality as they can.

One method, known as "bounded model checking", sees the functionality of the chip checked only up to a limited number of cycles of operation. Each time the zeroes and ones are re-written, a cycle of operation is complete and the state of the chip alters. Although this method will miss any error that manifests itself after this limit, it can still often cover much more of the chip's functionality and so provides a valuable net to trap potential design bugs.

In bounded model checking, the sequence of states through which the chip has passed up to the chosen bound is represented by a formula. The variables in the formula represent a numeric code for the individual states the device has been in. A second formula is created which describes an error in one or more of these states. The conjunction of these two formulae describes a fault happening at any point up to the boundary. If there is any combination of values that variables can take on that make this combined formula hold true, a bug has

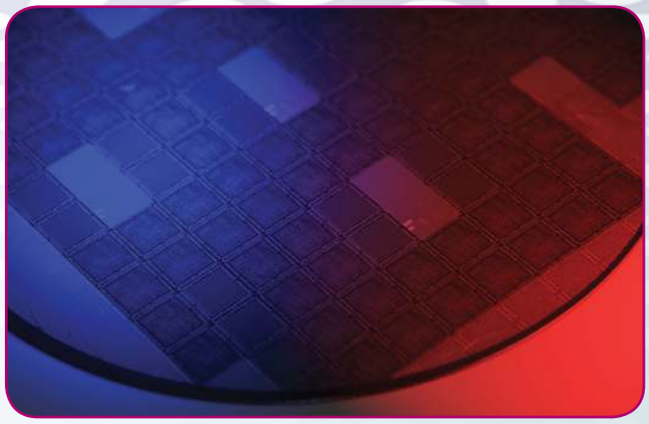
Institute of mathematics & its applications

National HE STEM Programme

occurred and the chip design is faulty. The exact value of these variables identifies the state in which the bug has happened.

Highly sophisticated computer algorithms called satisfiability solvers ("SAT solvers") are used to check properties like this. Theoretical analysis shows that SAT belongs to a class of problems for which there is no known fast algorithm. Indeed, determining whether or not it is possible to solve these problems efficiently is one of the major open problems of Computer Science today. Nonetheless, there has been spectacular progress over the past decade in designing SAT solving programs that work remarkably well on real-life problems.

Complementary to these sophisticated algorithms is the mathematical technique of "abstraction". Rather than modelling the whole problem, computer scientists like Professor Melham abstract out only the most important features of the chip — the ones most key to demonstrating its successful operation. These abstract models can be much more tractable for mathematical proofs than the whole design.



The challenge with abstraction comes in deciding just how far to abstract — if you go too far then your model isn't a sufficiently precise representation of what the chip is doing. This could lead to a bug being missed, with all the expensive ramifications that brings. But not abstracting enough leaves a model that's too large and complex to analyse mathematically.

One of the leading methods for creating a good abstraction is to deliberately abstract quite a long way and then gradually hone in by checking if any bugs that are flagged up are real or just ghosts of the abstraction process. Eventually the abstraction converges to a level that is a good representation of the chip but which doesn't need too much computing time.

Computer chips are everywhere and becoming increasingly complex. With mathematics, manufacturers can verify their chips, so that the ever-growing legion of electronic devices — fast becoming ubiquitous in modern life — can continue to push technological boundaries.

TECHNICAL SUPPLEMENT

Boolean algebra and SAT solvers

The logical formulae used in formal verification are commonly expressed with the operators of Boolean algebra, devised by English mathematician George Boole in 1847. Boolean algebra is the algebra of "truth-values", TRUE and FALSE, often written as 1 and 0. In the usual formulation, the operations are binary conjunction ("∧") and disjunction ("∨") together with negation ("¬") and the constants 0 and 1. Indeterminate truth-values are named by variables. So, for example, "x ∧ (y ∨ ¬z)" means "x, and y or not z". As the American mathematician Claude Shannon observed in the 1930s, these operations provide an algebraic basis for design and analysis of digital circuits – providing the fundamental mathematics behind the huge industry of digital chip design that we have today.

In formal verification, a range of sophisticated computer data structures and algorithms are employed to analyse Boolean formulae representing circuits. One extremely important class of algorithm is "SAT solvers". A Boolean formula is "satisfiable" if there is an assignment of truth-values to its variables that make it hold; an algorithm that solves the "SAT problem" is one that, given any Boolean formula F, can determine if F is satisfiable or not. (In practice, most algorithms deliver an actual satisfying assignment of values to the variables if the formula is indeed satisfiable.) SAT was the first problem to be shown to be "NP-complete", which implies that there is no known algorithm that efficiently determines satisfiability of every possible Boolean formula. Among the most famous and important unsolved problems of Theoretical Computer Science is the "P versus NP" question, which asks whether every problem (such as SAT) whose solutions can efficiently be verified to be solutions can also be solved efficiently. A definite answer either way would have profound and far-reaching practical consequences — not least to modern microelectronics design.

Although theoretical analysis suggests that SAT is in principle intractable, in practice modern SAT solvers are remarkably effective on the types of satisfiability problems that arise in actual practice in formal verification. Most of these procedures have their roots in the "DPLL" algorithm for Boolean satisfiability, introduced in 1962 by mathematicians and computer scientists Martin Davis, Hilary Putnam, George Logemann and Donald W. Loveland. Supplied with heuristics honed over years of intense competition between research groups, today's highly engineered DPLL procedures can tackle huge formulas with millions of variables. Already the cornerstone of formal verification for chip design, modern SAT solvers have immense promise as practical algorithmic tools in many other areas.

Abstraction

Abstraction is the act of isolating for separate consideration the important aspects or properties of a complex object, and ignoring the remaining ones as being irrelevant to the task in hand. A fundamental technique for controlling complexity, abstraction of various kinds is ubiquitous in computer science. The type of abstraction most used in formal verification is framed mathematically as a "Galois connection", a specific type of correspondence between two partially ordered sets. Here, the partial ordering is the relationship of satisfaction between formal circuit models and the correctness properties we wish to show they have. Framing abstractions in this way allows computer scientists to characterize precisely the conditions under which properties obtained on abstracted models are still valid for the more complex models they come from.

References

Armin Biere, Marijn Heule, Hans van Maaren, Toby Walsh (Editors), ***Handbook of Satisfiability.*** Frontiers in Artificial Intelligence and Applications, vol. 185 (IOS Press, 2009).

Edmund Clarke, Armin Biere, Richard Raimi and Yunshan Zhu, "Bounded Model Checking Using Satisfiability Solving", ***Formal Methods in System Design***, vol. 19, no. 1 (July 2001), pp. 7–34.

Orna Grumberg, "Abstraction and Refinement in Model Checking", ***Proceedings of the 4th international conference on Formal Methods for Components and Objects***, edited by Frank S. de Boer, et al. (Springer-Verlag, 2006), pp. 219–242.

The IMA would like to thank Professor Tom Melham, University of Oxford, for his help in the preparation of this document.