A Logical Perspective of Formal Verification
A narrative on the Genesis and Evolution of the Formal Verification Group at IIT Kharagpur.

Dr Pallab Dasgupta
Professor, Department of Computer Science & Engineering, IIT Kharagpur

On Aug 13, 2009, a relatively private condolence meeting took place at the famous IBM TJ Watson Research Centre, USA in memory of the departed soul of a former Computer Science professor of the Indian Institute of Technology Kharagpur. Among those present were some of the most senior architects of IBM's state-of-the-art processors, who fondly recollected having learnt the basics of Computer Architecture from this professor. Over the next month or so, literally hundreds of obituaries poured in from his students all over the world. As his son, I learnt that a teacher lives on in the hearts of generations of students, and that is one of the most gratifying aspects of this profession.

Having grown up in the campus of the oldest IIT, I was fortunate to witness an era where the size of computers shrunk from the size of a room to the size of a button. The cost of the transistor defied all laws of inflation and in the process, the semiconductor industry empowered mankind with unbelievable computational power. Behind this success story lays fascinating outlooks in computer architecture, scalable optimization algorithms and tools, and fundamental innovations in device physics.

What is often not perceptible at the surface is that the entire process of building an integrated circuit from its concept is based on the foundations of symbolic logic, namely the ability to capture the behaviour of electronic devices in terms of mathematical logic. Electronic design automation, which has equipped the chip designer with semi-automated tools for developing a chip having billions of transistors, would not be a reality had it not been based on logic and automated deduction.

My father was not a logician by training, but his PhD made several fundamental contributions in circuit minimization based on logic optimization. By the time I studied the subject of logic optimization as an undergraduate student of Computer Science, the circuit industry was already using these and more advanced techniques ubiquitously. The history of mathematics has never witnessed a more rapid adoption of the advances in logic based techniques as it has happened in the chip design industry.

The type of logic commonly used for digital integrated circuits such as processors is called Boolean logic or alternatively propositional logic. In simple terms, the circuit description defines the functionality of the circuit and its components in terms of logic formulae. Given the end-to-end functionality of a circuit, a synthesis tool automatically deduces the optimal combination of elementary circuit components (called logic gates) that together realize that functionality. The designer specifies the functionality of concurrent communicating circuit modules using high-level languages such as Verilog or VHDL, and the synthesis tools perform the underlying logic optimization automatically.

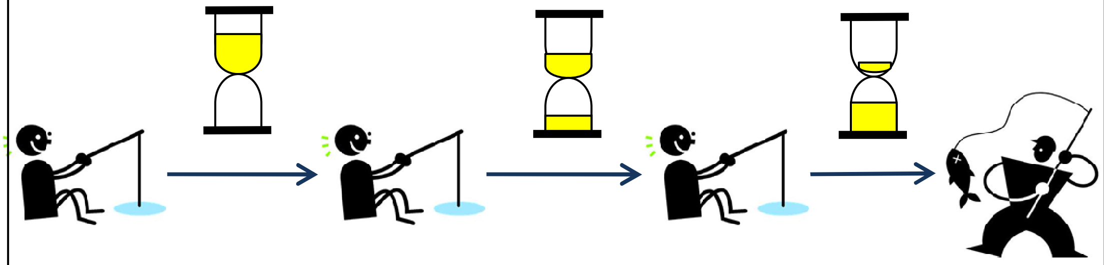
As the complexity of the circuits evolved from a few hundreds of gates to billions of gates, as is prevalent today, the task of verifying that the circuit has been designed correctly has become the most dominating task in the chip industry. Verification has not only become very complex, often accounting for more than 70% of the design cycle time – it has also become a very critical task considering the use of electronic chips in a wide range of safety critical embedded systems, including medical instruments, automotive / avionic control systems, and atomic reactors. With the emerging ubiquity of embedded software based control, the verification task now needs to address both the hardware and the software running on it.

The task of program verification, that is, whether a software program performs its intended task correctly is a much older and fundamental problem in Computer Science. Programs are also logical specifications, but the underlying logic is richer than Boolean logic and therefore more complex. In a seminal work presented in 1931 that shook the foundations of logic and deduction, Kurt Godel had shown that any logical system capable of expressing all relations between natural numbers was necessarily incomplete. In simple terms this means that if a framework based on logic had the power of capturing arithmetic relations between natural numbers, then there will be formulas in that logic whose truth cannot be determined by logical deduction. Since programs use arithmetic, it follows that in general we can have no automated system of deduction that conclusively proves the correctness of any given program. Mathematical verification techniques for programs therefore primarily rely on abstractions and conservative approximations.

Digital circuits, unlike programs, are finite state systems, and therefore the task of deducing whether a circuit will ever reach a bad state is always decidable. An automated system for deciding whether a circuit behaves correctly has two inputs – the *specification,* which defines the correct end-to-end behavioural requirements of the circuit, and the *implementation,* which is the Boolean logic of the circuit and its components. If the specification is also defined in terms of formal logic, then the automated system is called a *formal verification system*, and the task of deducing whether the implementation models (or implies) the specification is called a *model checking* problem.

The formative years of the formal verification research group at IIT Kharagpur coincided with the early years of adoption of this technology by the chip design and electronic design automation companies. The intuitive appeal of formal verification is quite significant. Circuits and programs have been traditionally verified through simulation and testing, however such verification is not complete unless it is carried out with all possible inputs. The input spaces for most real world systems are typically so large, that exploring all input combinations is infeasible in practice. In a complex system, this leaves the system vulnerable to errors that may happen on those inputs that were left out. For example, in a complex railway yard the number of ways in which trains may arrive and leave is so large that it is impossible to test a signalling system for all possibilities. On the other hand, formal verification proves a specified requirement (such as mutual exclusion among the conflicting routes in the yard) through logical deduction, and therefore, the coverage of behaviours is exhaustive leaving no scope for error.

The foremost challenge faced by the formal verification community was in choosing the language for defining the specification. The language should be expressive enough to capture the design intent but at the same time should not trade off the decidability of the underlying logical system in favour of expressive power. Specifically the specification involves the notion of logical behaviours over time, which is not explicitly captured by Boolean logic. For example, the requirement that a traffic signal must turn yellow for some time before it turns red has a temporal connotation that cannot be expressed in Boolean logic.



If the bait is cast then within an hour there is a catch.

Illustration: ***Temporal Logic allows us to specify timed behaviors***

In 1977, Dr Amir Pnueli proposed the use of a specification language for formal verification which extends Boolean logic with temporal operators. Thus *linear temporal logic* (LTL), for which Dr Pnueli was awarded the Turing award in 1996, enables the formal specification of properties over time within the syntactic fabric of a decidable propositional logic. LTL remained within the ambit of program verification, until the circuit community pushed for the development of formal language standards for writing assertions or formal properties of circuits. Today it is believed within sections of the verification community that one of the good things that came out of the Pentium FDIV bug of 1994 was the rapid adoption of assertion based verification in the semiconductor industry. Today assertions are used extensively by all chip design companies and formal verification of functional units in integrated circuits has become standard practice.

We started the Formal Verification Research Group at IIT Kharagpur at a strategic point of time, when the leading chip design companies (such as Intel, IBM) and electronic design automation companies (such as Synopsys) were collaborating through the Accellera consortium to merge their individual assertion languages (Forspec of Intel, Sugar of IBM and Open-Vera of Synopsys) into a uniform language standard called SystemVerilog Assertions. This paved the way for designers to express the design intent through formal properties – possibly the widest use of logic among non-logicians.

While the task of developing language standards for formal property specification was being pursued, computer scientists and logicians were working overtime to address a very important bottleneck in formal verification called the *state explosion problem*. Circuits are inherently component based concurrent systems and hence the number of states of a circuit is a product of the number of states of the components. To understand the problem of state explosion, consider a three aspect railway signal, which has only three states, namely red, yellow and green. The railway yard of Howrah station near Kolkata has more than 150 signals, and therefore the number of possible combinations of the signal states is 3^{150}, which is more than the Avogadro constant!! Some of these combinations are bad because they allow trains on conflicting routes, and the verification task is to determine whether such combinations are allowed by any sequence of execution of the signalling circuitry.

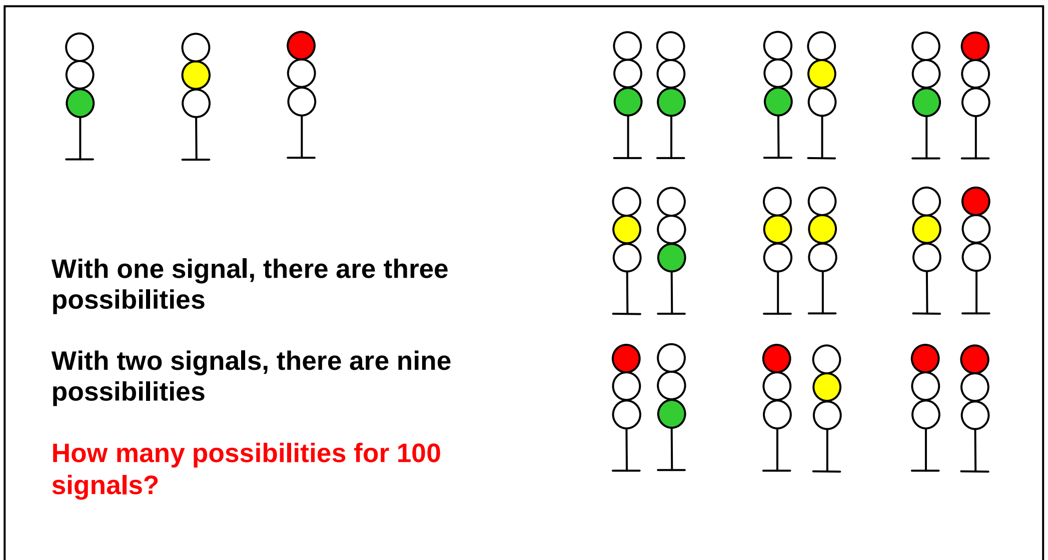

With one signal, there are three possibilities
With two signals, there are nine possibilities
How many possibilities for 100 signals?

Illustration: ***The State Explosion Problem***

The state explosion problem introduces an engineering dimension to the problem of formal verification. A theoretically complete proof procedure becomes infeasible because the state space defined by the consequents of the logic is too large to be stored or traversed explicitly. Contributions by Prof Edmund Clarke, Prof Alan Emerson and Dr Joseph Sifakis, for which they received the Turing award in 2007, established symbolic techniques which allow the proof procedure to work on a succinct representation of the state space, that is, *without* making the state space explicit. While this and other engineering innovations have helped in scaling the technology significantly, there is a still a very large gap between the requirements of formal verification and the feasibility of existing technology.

The focus of our research group has been shaped through the understanding of the practical difficulties in leveraging formal verification technology in industry. The first challenge came from a group within Intel, which posed a simple question with a complex answer – *Can the task of verifying a formal property on a circuit be decomposed to proving local properties on the component modules of the circuit?* The question essentially asks for a divide-and-conquer approach to formal verification. To us it was clear that attempting to automate the division of the functionality would not be practical in the absence of domain knowledge, since the state explosion problem would seriously affect the applicability of the technology. Hence we proposed a bottom up approach where the existing practice of writing assertions for the components could be leveraged to prove architectural properties through automated deduction. This new paradigm of verification was named *design intent verification*, since it involved a meta-level of verification where the design intent expressed in terms of architectural properties was being verified in terms of component properties.

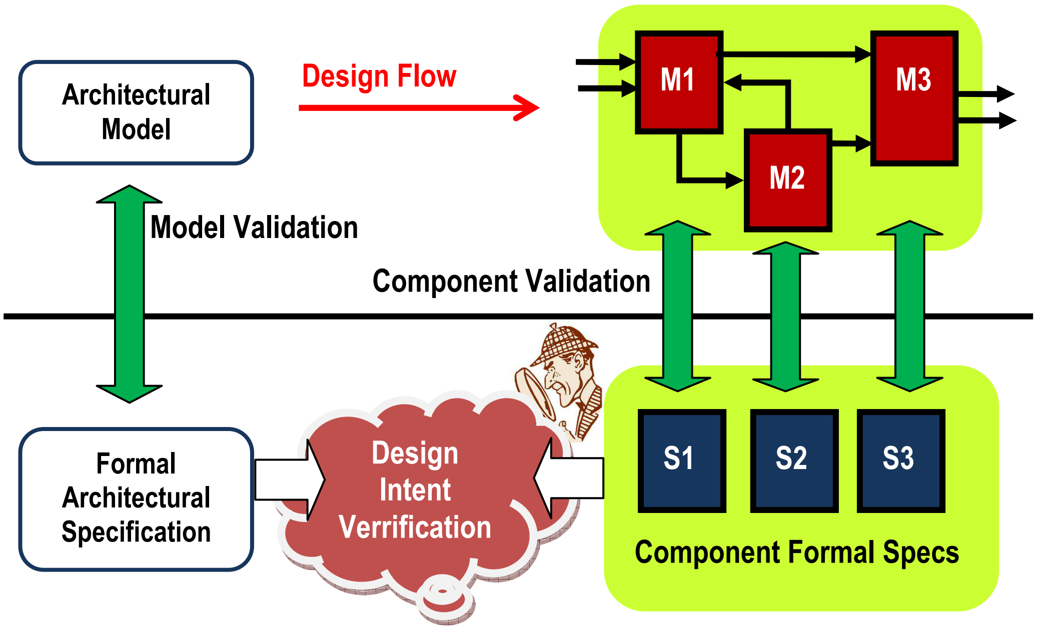


Illustration: ***Design Intent Verification***

In principle the design intent verification paradigm introduces a logical basis for engineering component based designs. We realized very soon that this is a powerful concept which can be applied in a wide variety of domains. In the years that followed, we applied this technique on a wide variety of problems – each requiring proof procedures of different types, but with the same underlying philosophy.

One of the most interesting engineering applications of design intent verification was in the automotive domain. Complex features in automotive systems require coordinated behaviours involving multiple distributed components. In order to achieve the desired timing in an end-to-end functionality, the designer must budget the time among the component actions. Since components are typically outsourced and a component may participate in multiple features, time budgeting is a complex global constraint satisfaction problem, which essentially justifies the specification of timing requirements in sub-system technical specifications. In collaboration with General Motors we developed a framework which extended our earlier framework to real time specifications, thereby adding fundamentally new decision procedures.

Illustration: ***Design Intent Verification***

The core principles of design intent verification have more recently been used for formally verifying the architectural power intent in complex low power digital integrated circuits. In simple terms, to save power consumption (and thereby extend battery life) most processors and system-on-chip designs regulate the voltage, frequency and power on multiple islands of the chip (called power domains). The architectural power management strategy is implemented through a global controller which orchestrates the power controllers of the various domains. In collaboration with Synopsys we developed a tool flow for verifying this orchestration at a feasible level of abstraction, by extracting out properties of individual power domains.

Yet another enriching experience was in developing formal specification languages and verification techniques for analog and mixed-signal circuits, which are essentially infinite state systems because the time and value domains are dense. We applied our techniques on circuits from the power management domain (power regulators, battery chargers, etc) and used them to develop frameworks for verifying the integration of cell phone power management units. This was done in collaboration with the National Semiconductor design centre at Scotland (now acquired by Texas Instruments). We have also been working with companies like Freescale Semiconductors and Texas Instruments on analog verification through the prestigious consortium, Semiconductor Research Consortium (SRC).

Symbolic logic and automated deduction will play an increasingly important role in engineering practices in the years to come. New formal approaches will have a remarkable impact on knowledge discovery, meta-level reasoning and decision support in large data-intensive cyber physical systems. I strongly believe that formal methods and AI techniques will play an important role in the roadmap for systems such as smart electrical power grids, climate and environment monitoring systems, and integrated disaster management systems. In order to address these problems in a structured way, the country needs to build teams involving individuals having core competence in formal methods, logic and deduction and domain experts from these niche domains.

I grew up in an era, where academicians were encouraged to focus on fundamental research and applied research was relegated to the industry. I have been fortunate to work in the area of formal verification where academic research has been driven by serious requirements from the industry. My own experience taught me that the types of problems which come from the industry often necessitate fundamental research of the highest complexity and is therefore equally rewarding in terms of research publications. More significantly, there is immense satisfaction in witnessing the deployment of one’s research in industrial practice.