

# Formal Verification

## Academic Research in the UK

**Kerstin Eder**

**Design Automation and Verification**

# UK Research in Verification

---



# University of Cambridge

---

## Programming, Logic and Semantics Group

- Mike Gordon

- *“Why higher order logic is a good formalism for specifying and verifying hardware”* [1986/1991]
- HOL theorem prover



- Larry Paulson

- interactive theorem prover Isabelle
- applying automated theorem provers to verification problems
- applying set theory to specification and verification



- Alan Mycroft

- from semantic models of programming languages to actually building optimising compilers
- Collaborations with ARM and Microsoft

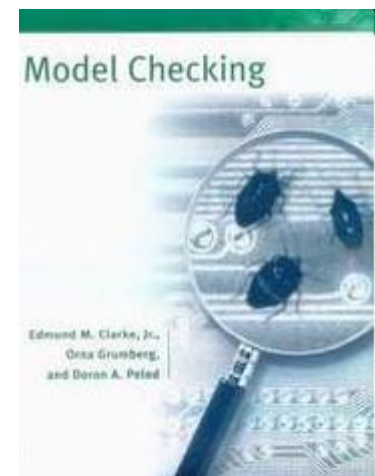


# University of Warwick

---

## Formal Methods Research Group

- Doron Peled
  - Concurrency theory
  - Semantics of Programming Languages
  - Formal Verification, Formal Specification
  - Model Checking, Finite Automata
  - Software Testing, Temporal Logics, Partial order methods and Traces.
  
- Recent work on combining model checking and testing



# University of Oxford

---

## Verification Research Theme

- **Bill Roscoe**
  - Concurrency, Verification, FDR
  - Analysis of cryptographic protocols
- **Marta Kwiatkowska**
  - Modelling and quantitative verification of probabilistic systems
- **Tom Melham**
  - Formal logic, mechanised reasoning, model checking and theorem proving, formal verification



# Verification Methodology

Robert B. Jones, John W. O'Leary, Carl-Johan H. Seger, Mark D. Aagaard, and **Thomas F. Melham**. Practical formal verification in microprocessor design. *IEEE Design & Test of Computers*, 18(4):16-25, July/August 2001.

- **Methodology addresses:**
  - Realism
  - Transparency and soundness
  - Structure
  - Incrementality and recoverability
  - Debugging and feedback
  - Top-down and bottom-up
  - Regression and reuse
- Work in collaboration with Carl Seger (Intel)
- Forte verification environment (Intel)



## Practical Formal Verification in Microprocessor Design

Robert B. Jones, John W. O'Leary, and Carl-Johan H. Seger  
Intel

Mark D. Aagaard  
University of Waterloo

Thomas F. Melham  
University of Glasgow

Practical application of formal methods requires more than advanced technology and tools; it requires an appropriate methodology. A verification methodology for data-path-dominated hardware combines model checking and theorem proving in a customizable framework. This methodology has been effective in large-scale industrial trials, including verification of an IEEE-compliant floating-point adder.

■ **FUNCTIONAL VALIDATION** is one of the major challenges in chip design today, with test generation, test bench construction, and simulation consuming a significant portion of the design effort. Throughout the 1990s, formal verification emerged as a promising complement to conventional simulation-based validation.<sup>1</sup> Most formal verification research concerns algorithms and focuses on tool capacity limits. Yet almost any serious verification effort faces many practical difficulties besides capacity. In a large verification project, the effort required to organize the multitude of tasks, specifications, and verification scripts can limit the quality and productivity of the work.

We tackle this problem by coupling our research on verification algorithms and tools with research on verification methodology. Our goal is to address the realities of design practice—rapid changes and incomplete specifications—while producing high-quality results and improving verification productivity. Our methodology systematically organizes a large verification effort's many interdependent activities and provides a guiding structure for the verification process.

Any formal verification tool researcher is keenly aware that what is a routine verification for the technology expert or tool developer may be very difficult for others to duplicate. Our methodology addresses this problem by tailoring a formal, custom-built verification framework, Forte, to industrial-scale circuits and industrial design environments. Forte combines an efficient, linear temporal logic model-checking algorithm, called symbolic trajectory evaluation (STE),<sup>2</sup> with lightweight theorem proving. FL—a custom, general-purpose functional programming language—tightly integrates the model checker and the theorem prover. This combination of model checking, theorem proving, and a general-purpose programming language makes the verification environment customizable and lets large verification efforts be organized effectively.

Our methodology has evolved over several years of use on fully custom, high-performance

# University of Southampton

---

## DSSE: Dependable Systems & Software Engineering Group

- **Michael Butler (Head of group)**
  - The Refinement Calculus and tool support for refinement
  - Refinement of distributed systems
  - Formal modelling and analysis of business transactions
  - CSP and B (Rodin)
  - Model checking
  
- **Joao Marques-Silva**
  - algorithms for boolean satisfiability and extensions
  - formal methods, model checking, system specification and verification
  - application of formal methods
  - design automation



# Conclusion

---

- In academic context, verification is often taken to mean **Formal Verification**
  - Raise awareness for simulation-based verification and associated interesting research problems
- UK good base providing foundations for formal verification

More industrial collaboration needed