Preface

Microprocessors are the workhorses of modern high-assurance systems. New cars typically utilize dozens of microprocessors, managing vital functions such as braking, ignition, traction control, and air bags. Most of the critical functions on modern aircraft, including flight control, engine control, flight management, and pilot displays, are microprocessor based. Medical devices, transportation systems, power grids, communications infrastructure, defense systems, and numerous other high-confidence applications are all dependent on microprocessor control.

Over the last 30 years, the microprocessors deployed in high-assurance systems have increased in sophistication from simple 8-bit chips with less than 10,000 transistors to 32- and 64-bit chips with millions of transistors, pipelining, floating point, specialized functional units (e.g., for cryptography), memory management, cache management, and so on. The system software for these systems has likewise increased from simple interrupt handlers to full-fledged operating systems capable of supporting multiple concurrent applications with space and time partitioning. Finally, the application logic on these microprocessor systems has increased from a few hundred lines of assembly code to millions of lines of high-level language code. The means of expressing the application design has changed as well, progressing from schematics and assembly code to high-level languages for both hardware and software. Indeed, many high-assurance applications are now developed using model-based development languages, with the final source code (which may be a mixture of programming language and hardware description language source text) autogenerated from the models.

This begs the question: how are we to trust this complex stack of microprocessor hardware, microcode, operating system kernel, runtime libraries, and application logic? Simulation and testing have shown to be inadequate, as several high-profile design flaws have manifested themselves in the field. How do we even specify the important safety and/or security properties for complex systems, much less assure that implementations maintain them?

In this book, we examine several leading-edge design and verification technologies that have been successfully applied to microprocessor systems at various levels – from arithmetic circuits to microcode to instruction sets to operating systems to applications. We focus on recent hardware, software, and system designs
that have actually been built and deployed, and feature systems that have been certify
ed at high Evaluation Assurance Levels, namely the Rockwell Collins AAMP7G
microprocessor (EAL7) and the Green Hills INTEGRITY-178B separation kernel
(EAL6+). The contributing authors to this book have endeavored to bring forth
compelling new material on significant, modern design and verification efforts;
many of the results described herein were obtained only within the past year.

This book is intended for practicing computer engineers, computer scientists,
professionals in related fields, as well as faculty and students, who have an interest
in the intersection of high-assurance design, microprocessor systems, and formal
verification, and wish to learn about current developments in the field. It is not in-
tended as a tutorial for any of the aforementioned subjects, for which excellent texts
already exist.

The approach we have taken is to treat each subject that we examine in depth.
Rather than presenting a mere summary of the work, we provide details: how ex-
actly the design is specified and implemented, how the design is formalized, what
the exact correctness properties are, and how the design is shown to meet its spec-
ification. Thus, for example, the text describes precisely how a radix-4 SRT divider
for a commercial microprocessor is implemented and proven correct. Another chap-
ter details how a complete AES-128 design is refined from an abstract specification
traceable back to the FIPS-197 document all the way down to a high-performance
hardware-based implementation that is provably equivalent to the FIPS-197 specifi-
cation. The contributors to this book have made an extraordinary effort to produce
descriptions of their work that are as complete and detailed as possible.

Just as important, this book takes the time to derive useful correctness statements
from basic principles. The text formally develops the “GWV” family of information
flow theorems used in the certifications of the AAMP7G as well as the INTEGRITY-
178B kernel, proceeding from a simple model of computing systems (expressed
in the language of the PVS theorem prover) called the “calculus of indices”, and
formally developing the GWVr1 and GWVr2 information flow theorems. The text
presents a proof of how the GWV formulation maps to classical noninterference, as
well as a proof demonstrating that a system can be shown to uphold the GWVr1 in-
formation flow specification via model checking. Another example of development
from basic principles can be found in the chapter detailing the refinement frame-
works used in the verification of the seL4 microkernel.

Along the way, we delve into a number of “tools of the trade” – theorem provers
(e.g., ACL2, HOL4, Isabelle/HOL, PVS), model checkers (BAT, NuSMV, Prover),
and equivalence checkers – and show how formal verification toolchains are increas-
ingly able to parse the actual engineering artifacts under analysis, with the result that
the formal models are much more detailed and accurate. Another tool trend noted
in several chapters is the combination of theorem proving, model checking, sym-
bolic simulation, etc., to produce a final verification result. A notable example of
this combination of techniques documented in the text is the process used by Cen-
taur Technology to verify their x86 compatible processors. The book also highlights
ways in which ideas from, for example, theorem proving and compiler design, are
being combined to produce novel and useful capabilities.
We open with a description of the ACL2 theorem prover, utilized in a number of the succeeding chapters, and then proceeds from the design and verification of basic high-performance hardware found in modern microprocessors (e.g., a divider circuit), to larger functional units (FPUs and cryptographic units), to pipelines, and then on to microcoded functions. The book then addresses the microprocessor at the instruction set level, focusing on machine code proofs, particularly the “decompilation into logic” approach pioneered by users of the HOL4 system. After developing some basic information flow theorems, the book turns its attention to operating system verification, with chapters on recent successes in information flow verification (INTEGRITY-178B) as well as functional correctness (seL4). We then progress to the application level, with a chapter dealing with the specification and checking of software contracts for conditional information flow, targeting the SPARK high-assurance Ada language subset. The book concludes with a description of tools and techniques for model checking information flow properties, applicable to both hardware and software systems that have been implemented using model-based development tools such as Simulink or SCADE.

Looking back over the year or so since I was first contacted by my superb editor at Springer, Charles Glaser, about the possibility of doing a book, I am both surprised and pleased to see that the finished product is quite close to my original, overly ambitious vision (as a colleague, perhaps rightly, called it at the time). To the extent that this book succeeds in describing compelling new results in the design and verification of high-assurance microprocessor systems, I have my contributing authors to thank; conversely, I am solely to blame for any shortcomings in reaching that goal. I would also like to thank my employer, Rockwell Collins, Inc., particularly John Borghese, Ray Kamin, Matt Wilding, and Ray Richards in the Advanced Technology Center, for providing me with time and facilities to work on the book. Several farsighted US government employees also played a significant role in the development of the technologies described herein, and I wish to especially acknowledge the efforts of Bill Legato, Brad Martin, and Mark Vanfleet of DoD, Jahn Luke and Dave Homan of AFRL, as well as Rick Butler and Paul Miner of NASA. Finally, I wish to thank my family and friends for their patience and support.

Cedar Rapids, IA                                      David S. Hardin
Design and Verification of Microprocessor Systems for High-Assurance Applications
Hardin, D.S. (Ed.)
2010, XIV, 436 p., Hardcover