PART 2: Proposed Research

Background

The research proposed here will contribute to formal verification technologies for systems-on-a-chip (SoC) and hardware/software co-design.

Much current formal verification is directed at finding errors (debugging). Our research primarily concerns *assurance of correctness* though, of course, failure to prove correctness can reveal errors. Assurance of correctness adds significant value when failures would be exceptionally costly, for example to applications which need to protect sensitive data, such as confidential passwords and product keys stored on hand-held devices.

In the past, full formal assurance of correctness has only been practical for academic systems designed for proof of correctness [14, 23, 27, 2] or fragments of designs [19, 15]. Recent advances in automated theorem proving now make it plausible that realistic systems based on commercial-off-the-shelf (COTS) parts can be proved correct.

If successful, this project will result in possibly the first machine checked formal verification of software running on a formally verified COTS processor, and would provide data and methodology to enable future system designers to evaluate the costs and benefits of formal proof of correctness as part of their verification flow. As a case study we will develop a demonstrator application written in ARM assembler that communicates with one or more separate hardware components. We will investigate both tightly coupled co-processors communicating on the data bus and loosely coupled communication over an AMBA system bus. Although the detailed case studies will use the ARM processor and bus architecture, we aim to make our verification methods as widely applicable as possible.

The main background to this proposal is a predecessor EPSRC project entitled *Formal Specification and Verification of ARM6* in which we used the HOL4 theorem prover to formally verify implementations of all the instructions of an ARM6 processor (which is identical to the ARM610 [1], but without an MMU). The proof was achieved within a timescale that would have been impossible ten years ago. The approach used an algebraic modelling and proof technique developed by Harman and Tucker at Swansea University [13] and deployed on the ARM processor architecture at Cambridge by Fox [11, 9, 10, 8]. The goal of the research proposed here is to establish convincing proof of concept for the formal verification of complete systems, and for this it is necessary not only to model data processing (i.e. instruction execution), but also communication with peripherals.

Two previous projects by researchers in Texas, USA, provide additional background. In the first project, the verification of code on a verified processor called FM9001 was investigated, using the Boyer-Moore prover. This work is known as the CLI stack [3]. FM9001 is an academic design intended for formal verification. In a second project, Boyer and Yu formally described a substantial subset of the Motorola MC68020 processor in the logic of the Boyer-Moore prover and used the resulting model to verify compiler generated object code for various simple algorithms, including Quicksort and the Berkeley Unix C string library [4]. As far as we are aware, the MC68020 programmers model was not verified to correspond to the actual hardware.

Further background is provided by previous work at Cambridge, in which Paul Curzon developed a programming logic for the Vista assembly language for the Viper microprocessor [6]. Although the Vista semantics was not mechanically derived from the formal model of the Viper processor, it was developed in the context of prior work on processor specification and verification [5].

The project proposed here will be an advance over previous work in that: (i) we will use a COTS processor (ARM) that we have formally verified and (ii) we will explore 'sideways' linking to other components as well as vertical 'stacking' of software.

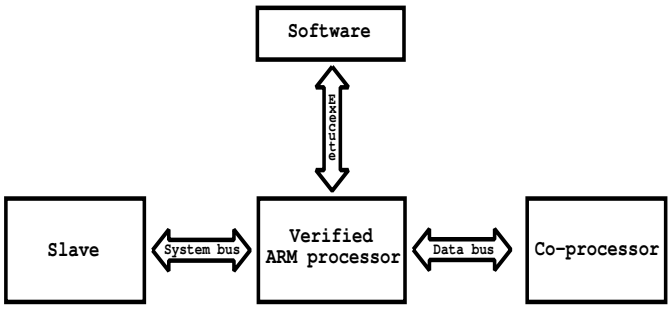
Prior theorem proving work includes early studies of asynchronous processor-memory communication by Joyce [17] at Cambridge, and recent work by Melham and Susanto [25] and Schmaltz and Borrione [26] on modelling the AMBA bus using ACL2 (the latest version of the Boyer-Moore theorem prover).

In a recent Grand Challenge [20] J Moore states "We are unaware of any practical work on the verification of input/output routines or the verification of code employing them". The proposal here will investigate this problem in the context of ARM.

Research Programme

Our goal is to research new modelling and theorem proving methods for reasoning about the functional correctness of combinations of hardware and software, such as those found in systems-on-a-chip.

We will combine the verification of 'vertical stacking' of software on hardware with the 'horizontal verification' of bus transactions:



Because correctness assurance particularly adds value when there are data security concerns, we will select case studies in this area. ARM processors are used to implement devices like mobile phones and PDAs where assurance that information doesn't leak would be valuable.

In the following sections we describe the components of our proposed research.

Communication and exceptions

We will extend and exploit our existing verified programmers model of the execution of ARM6 instructions. The extensions that are necessary are to handle communication with co-processors and slaves.

Co-processors provide tightly coupled external hardware support, via the ARM data bus, for functions like memory management and floating-point computation. When the master processor executes a co-processor instruction it sends it to any co-processors which may be attached. If a co-processor is busy at the time the instruction is sent, the processor will wait for it. If there is no co-processor capable of executing the instruction, an undefined instruction trap will be executed.

Slaves are loosely coupled components of a system-on-a-chip that communicate over a system bus with a bus master, normally a processor, via bus protocols. There are several AMBA protocols, and we plan to use models of them, in Z and higher order logic, that are being developed by Dr Malcolm Newey, of the Australian National University.

We will develop a framework in higher order logic for modelling communications between processors and components. We plan to look both at the simpler co-processor mechanism and also at system bus communication (AMBA). We aim to model communication hardware and to investigate proving that the detailed signalling constituting a transaction achieves the behavioural goal of transferring and processing data.

A challenge is to derive, from a formalisation of the communication protocol, a higher level executable model suitable for the symbolic execution needed in proofs. A specification typically says that requests must eventually be acknowledged, which is naturally formalised with an existential quantifier ($req(t) \Rightarrow \exists t'.\ t' > t \wedge ack(t')$). To create a symbolically executable model, the $\exists$ needs to be Skolemized to create an explicit response function, f say, such that ($req(t) \Rightarrow ack(f(t))$). We hope to be able to mechanically derive executable models by formal logical manipulation (carried out by theorem proving scripts) of models of the processor, busses and co-processor or slave.

We will model processor execution using our previously verified high-level model. A difficulty is that certain aspects of interrupt and exception handling behaviour can only be accurately represented using the micro-architecture (e.g. the effect of an interrupt or exception may depend on the pipeline state). Currently this behaviour is left unspecified, but we plan to lift abstractions of the micro-architecture into the high-level specification so that we can fully represent the complete behaviour. Thus we need to develop a specification in which a few aspects of the low-level pipeline implementation are visible. A challenge is to achieve this without making the model intractable, but whilst enabling exceptional behaviour to be proved to be safe (e.g. to show that interrupts cannot cause unwanted information leakage).

High level symbolic execution

Once we have added formal specifications of communication and exceptions to our ARM model, we need to investigate the infrastructure needed to support system level verification. It is difficult to produce complete and accurate formal models of high-level languages such as C++ and Java. Furthermore, it is extremely difficult to verify the correctness of compilers, interpreters and run-time environments for such languages. This means that establishing the behavioural correctness of software in a top-down manner, from code written in these languages to silicon, is beyond the scope of current technology.

Formally verifying the behavioural correctness of code at the assembly level is more readily achievable. To do this one must be able to symbolically execute sub-routines and then apply appropriate abstractions in order to reason about larger programs. As we and others have found [12], symbolic execution is a key technique underlying proofs. The Swansea algebraic processor modelling method we have used is based on functions, as are models in the Boyer-Moore logic. Traditional approaches in higher order logic use relations, but a key advantage of functions is that they make it easier to symbolically execute specifications.

An accurate functional specification of the ARM instruction set was developed as part of the ARM6 formal verification in HOL. This model was tested by executing ARM machine-code generated by the GNU assembler. In verifying the processor's correctness this model is symbolically executed, covering each possible single instruction program. We propose extending this work to support a bottom-up approach to the verification of complete assembly programs.

To reason more effectively we need to be able to manage symbolic execution at the more abstract level corresponding to assembly code. The higher level software models must be related to the concrete machine code execution of the processor via abstraction mappings [18, 24]. The particular abstractions needed may depend on the example under verification, so we need to build support for application specific higher level models on top of the standard instruction and bus models. Understanding how to define and execute such abstractions constitutes a core part of the work proposed here.

For example, consider the following code fragment:

```
ldmia r0!, {r1,r2}
ldmia r0!, {r3,r4}
adds  r6, r2, r4
adc   r5, r1, r3
stmia r0, {r5,r6}
```

One would like to be able to show that this piece of code performs 64-bit addition, taking arguments from the addresses `r0` and `r0+8` and storing the result at address `r0+16`. When working with larger sections of code (with conditional execution and loops) it will be challenging to

find ways to manage the size and complexity of the terms representing the programmer's model state.

Although this work will be example driven, the aim will be to develop techniques and tools that are as generic as possible.

Modelling data security (case studies)

As an application of our models and verification tools, we will perform case studies designed to show proof of concept of the formal assurance of information security of software running on a processor accessing data via a bus.

Our goal will be to devise a case study that is simple enough to be completed during the project, yet is realistic enough to show the feasibility of the methods to our beneficiaries. In our experience, devising good examples is hard, so we expect to put a lot of early effort into it.

The development of the case study will be the first step of the research. We hope to get ideas both from our partner ARM and from third party application providers.

We expect our example to consist of a small hardware design of a component that provides storage to sensitive data and code running on the processor that can access it. There are various kinds of information flow issues that we will consider. We propose to formulate security semantically, directly in terms of execution on our communication and data processing model.

The simplest case to specify is when an application should produce outputs that are independent of the sensitive data (e.g. performing a task that doesn't need access to secret keys). There are a variety of frameworks for specifying information flow security [22, 16, 7]. A simple idea, that is the essence of several recent semantic formulations of secure information flow, is to split the system state into a high security (e.g. secret) part h and a low security (e.g. public) part l. The state is then a pair $\langle h, l\rangle$. We may want to ensure a computation step has the property that from every starting state $\langle h, l\rangle$, the next state in the computation is such that the new l will be independent of the original h, i.e. the new low security component l does not leak information from the initial high security component h. If a function F maps a state to the next state in a computation, then we want $\forall l.\ \exists l'.\ \forall h.\ \exists h'.\ \langle h', l'\rangle = F\langle h, l\rangle$. The starting point of our case study will be seeing if abstract formulations like this can be developed into tractable concrete and convincing security specifications for executions of ARM based systems. We expect to liaise with ARM to ensure the security specifications are adequate.

A more tricky case to specify is when low security observable outputs are computed from high security data that should not be revealed. A simple example is when a secret key is used for login authentication: a user inputs a string and the system says whether it matches a member of a stored high security set. The matching of the input to the stored set may be done by 'trusted' login code and one may want to be certain that only this code sees the secret data, so that the execution of other code is independent of it. This can be formulated by defining a predicate on system states that identifies a subset of secure states. The software and hardware is then organised so that (i) secret data can only be seen in secure states, and (ii) trusted code can only be executed in secure states. The research challenge for our case study is to convincingly formulate this in terms of executing the formal models of the processor and peripherals.

For our case study we plan to use, as an example of trusted code, an implementation of the AES Algorithm (Rijndael) that has already been modelled in higher order logic and verified by Konrad Slind of the University of Utah [24]. In collaboration with Slind, we will create verified implementations of AES in ARM software. We expect to achieve this by verifying a compiler from Slind's AES specification to ARM assembler. Slind's specification is in the functional language TFL[1] that is embedded in higher order logic and supported by the HOL system. Although formally verifying a compiler for a standard programming language, or even all of TFL, would be a major undertaking, well beyond the scope of this project, we think compiling a subset of TFL sufficient for AES should be tractable.

As part of the case study, we will devise, with ARM's help, a demonstration to show how to verify security properties of practical importance. Developing the details is part of the research. For example, we might have a thread that manipulates streams of low security data (e.g. copying it from an input to an output) and another that uses AES for encryption/decryption. The goal could then be a 'proof of separability' [21] to show that the low security thread cannot be influenced by the high security one.

If we are successful in verifying security properties with the software implementation of AES, we plan, also in collaboration with Professor Slind, to create a hardware implementation, probably as a tightly coupled co-processor. We will then investigate the formulation and verification of security properties for this. Finally, we hope to consider implementations of AES that combine software with hardware acceleration.

We will use this case study to explore how formal verification can cope with a range of hardware/software decompositions. Our hope would be to structure the proofs into lemmas, some of which are purely 'mathematical' and can be used for all implementations, and others are 'engineering' and verify implementation specific details.

Recently ARM announced a security extension called TrustZone that adds an additional security bit to the state. We plan, with ARM's help, to experiment with adding similar hardware extensions to our ARM model. We will investigate formulating and proving general semantic theorems about the extended architecture to aid in the verification of particular application-specific security properties. An important goal is to devise methods to formally

[1] `http://www.cl.cam.ac.uk/users/kxs/tfl.html`

assure that trusted code and high security peripherals can only be accessed when the hardware is in a secure state. We intend to study how security support in hardware can simplify formal verification of desirable properties.

The general goal of this case study is to explore the deployment of formal proof across a range of hardware/software combinations in order to provide insight into formal verification issues for systems-on-a-chip. Because the case study concerns security, we plan also to contribute to the specification and verification of security properties for hardware/software. The unique feature of the work is that we will formulate properties directly in terms of the semantics of execution on COTS parts.

Methodology

Our research methodology is example driven: we will specify and verify a simple complete system containing one of more ARM processors running software and linked to one or more peripheral components. We plan to use a security application as demonstrator, because this area is where assurance by formal proof is likely to add significant value.

Success criteria

The success of the project depends on achieving a number of goals.

- Devising a tractable formal model of exceptions for the ARM processor.
- Devising a tractable formal model of co-processor and master-slave communication.
- Devising and implementing an efficient method of symbolically executing the semantics of ARM software and communication transactions.
- Devising a tractable formal model of secure states of the ARM processor, and how data and system bus communication ensures secure signalling.
- Devising convincing formalisations of useful security properties for the execution of software communicating with co-processor hardware.
- Demonstrating the tractability of our hardware and software models and algorithms with convincing case studies.
- Obtaining new insights into methods for deploying theorem proving for system level formal verification, and useful data on their costs and benefits.

The scientific challenge of this project involves developing new semantic models and supporting them effectively with new theorem proving tools. The risk is that the semantics will turn out to be intractable, so that we are unable to demonstrate significant case studies.

Project management

The project will be managed by the Principal Investigator (Mike Gordon) in collaboration with Dr Anthony Fox, who is the co-author of this proposal. Fox will work full-time on scientific research and Gordon will work part time on it. The development of the scientific goals of the research, especially the case study, will be done in liaison with ARM to ensure the problems studied are realistic.

Collaboration

We will liaise with our partners at ARM Limited about modelling issues, and to get guidance on the industrial potential of the research. In particular, ARM will assist us in devising demonstrator case studies to investigate the possibility of formal assurance of commercially significant security properties.

We will collaborate with Professor Konrad Slind of the University of Utah on (i) the implementation of symbolic execution tools for ARM code, and (ii) a case study based on his higher order logic model of AES encryption.

We plan to collaborate with Dr. Malcolm Newey, of the Australian National University, on modelling bus transactions. He is developing formal models of ARM's AMBA bus architectures in Z and higher order logic, and we intend to evaluate these for use in our case study.

Relevance to Beneficiaries

We hope to demonstrate the use of theorem proving to provide formal assurance of the functional correctness of electronic systems composed of COTS parts.

A main beneficiary is intended to be our collaborator and industrial mentor, ARM Ltd. ARM is a strong believer in formal verification and applies commercial equivalence checking, model checking and other automatic proof tools to its current hardware design flow. However, at present ARM do not use any theorem proving. We hope the research proposed here will enable them to evaluate the costs and benefits of theorem proving, and decide whether it might be appropriate for future use in verification. If the case study is successful, ARM might try using the theorem proving tools and methods developed in the project on-site on live projects related to TrustZone designs.

Other designers of CPU cores are also possible beneficiaries, especially those that require high assurance, such as Common Criteria Evaluation Assurance Level 7 (EAL7).[2]

We hope our case study will make a significant scientific contribution to the methodology of formal verification of security critical code, and thus implementers of such code are potential beneficiaries. In particular, we hope the results of our research collaboration with Professor Slind will benefit those investigating challenges[3] raised by implementing cryptographic modules, especially if Level 4

[2] www.clusit.it/whitepapers/iso15408-3.pdf

[3] www.cryptol.net/challenges.htm

Design Assurance is sought.[4].

We believe this project will provide data on the costs and practical feasibility of formally verifying, by machine checked proof, applications based on commercial embedded processors. It will contribute to determining the currently available "best practise" for assurance in critical systems. Thus we expect standards organisations and insurers to be potential beneficiaries.

Dissemination & Exploitation

Dissemination of the results of the project will occur via the normal channels of conference presentations and journal publications. In addition, we will establish and maintain a web site devoted to making results and progress reports available during the project.

The ARM6 specification in higher order logic is publicly available as part of the HOL4 distribution. We will continue to maintain this and will add to the project directory at SourceForge any improvements to the processor model (e.g. to handle exceptions better).

As ARM will be helping us devise hardware extensions for our case study, any public release of the details of these extensions will need prior approval from ARM.

We will also make available any formal verification tools we develop. The research directions and results will be discussed with engineers at ARM Ltd and other companies with which we have contacts.

Justification of resources

Equipment.
We request a linux PC for office work and a laptop for home working, travel and field demonstrations. We would make an initial purchase at the start of the project, and then an upgrade after two years. As theorem proving is computationally demanding, we are specifying a powerful processor and additional memory.

We also request VMware licences to enable us to run Windows applications (e.g. Microsoft Office software, especially PowerPoint for presentations).

We request funds to buy some time of a computer officer (CO) to install compatible systems and ensure smooth inter-operation between the different platforms and to support the installation and deployment of code management systems accessible from both systems as well as via virtual machines. Charges for Microsoft office software (to run under VMware), file storage, backups and archiving, network connection and printing are also budgeted for.

Personnel.
We are requesting support for Dr Anthony Fox, who was responsible for the ARM6 formal verification which this project builds on.

Travel.
Scientists in the USA whom it would be valuable to visit for research discussions include: Gopalakrishnan, Grundy, Harrison, Hunt, Moore, O'Leary, and Slind. Other researchers include: Hurd, Luk, Melham, Susanto (UK), Sheeran (Sweden) and Paul (Germany). We have budgeted for 3 overseas trips per annum to cover the investigator and RA, plus 20 days of UK travel. Major conferences likely to be of interest to us are: CADE, CHARME, DAC, DATE, DVCon, FMCAD, CAV, TPHOLS, TACAS. However, it is likely that other meetings and workshops will appear that are appropriate for the presentation and discussion of our work. We also may wish to attend commercial training courses to enable us to understand better the needs of industry.

Changes made in response to referees evaluation

This proposal is a heavily revised resubmission of an earlier one that was unsuccessful. We have carefully studied the referees comments and made substantial changes to the research plan.

One criticism was: "It is a pity that ARM does not show more interest in this project, for example through a more formal collaboration". We have addressed this by making ARM a formal collaborator and involving them in the drafting of the revised proposal. A substantial component of the re-emphasis of the research comes from this. In particular, the case study is now focused on information assurance, which is a particular current interest of ARM.

Another worry expressed by a referee was that the Pie board, which was originally mentioned as a possible a central component of the case study, was "not representative of the state-of-the-art". The case study in the revised proposal has been completely redesigned, and we are no longer considering using the Pie board. With ARM's help, we hope to ensure the case study will produce results relevant to current designs.

Some of the costs were queried. We have both reduced the cost of the project and provided greater justification for the remaining expenditure.

We hope the revised project is slimmer and more focused than its predecessor. It also has an additional adventurous component: developing direct semantic formulations of information assurance properties for concrete hardware and software models.

Acknowledgements

We thank Professor Konrad Slind of the University of Utah for helping us develop the parts of this proposal that use his verified AES model.

We also thank Dr Joe Hurd of Oxford University for helping us formulate some of the information assurance aspects of the case study.

[4] csrc.nist.gov/publications/fips/fips140-2/fips1402.pdf

References

[1] ARM Ltd. *ARM610 Data Sheet*. ARM Ltd, Document Number; ARM DDI 0004E, Cambridge, January 1996.

[2] C. Berg, S. Beyer, C. Jacobi, D. Kröning and D. Leinenbach. Formal Verification of the VAMP Microprocessor (Project Status), Symposium on the Effectiveness of Logic in Computer Science (ELICS02), Technical Report MPI-I-2002-2-007, Max-Planck-Institut für Informatik Saarbruecken, Germany, 2002, pages 31-36, edited by Witold Charatonik and Harald Ganzinger

[3] W.R. Bevier, W.A. Hunt, J S. Moore, and W.D. Young. Special issue on system verification. *Journal of Automated Reasoning*, 5(4):509–530, 1989.

[4] R. Boyer and Y. Yu. Automated Proofs of Object Code for a Widely Used Microprocessor. *Journal of the ACM*, 43(1):166–192, January, 1996.

[5] A. J. Cohn. The Notion of Proof in Hardware Verification. *Journal of Automated Reasoning*, 5:127–13, 1989.

[6] Paul Curzon. Deriving Correctness Properties of Compiled Code. *Formal Methods in System Design*, Volume 3, Numbers 1/2, pages 83-115, August 1993.

[7] Adam Darvas and Reiner Hähnle and David Sands. A Theorem Proving Approach to Analysis of Secure Information Flow. `citeseer.nj.nec.com/561672.html`

[8] A.C.J Fox. Formal specification and verification of ARM6. In David Basin and Burkhart Wolff, editors, TPHOLs '03, volume 2758 of LNCS, pages 25-40. Springer-Verlag, 2003.

[9] A.C.J. Fox. An Algebraic Framework for Modelling and Verifying Microprocessors using HOL. Technical report No. 512, University of Cambridge Computer Laboratory, April 2001.
`www.cl.cam.ac.uk/users/acjf3/papers/tr512.ps.gz`

[10] A.C.J. Fox. A HOL Specification of the ARM Instruction Set Architecture. Technical report No. 545, University of Cambridge Computer Laboratory, June 2001. `www.cl.cam.ac.uk/users/acjf3/papers/tr545.ps.gz`

[11] A.C.J. Fox. Algebraic Models for Advanced Microprocessors, PhD thesis, University of Wales Swansea, 1998. `www.cl.cam.ac.uk/users/acjf3/papers/fox98.ps.bz2`

[12] David Greve. Symbolic Simulation of the JEM1 Microprocessor. In *Formal Methods in Computer-Aided Design* (FMCAD '98), edited by Ganesh Gopalakrishnan and Phillip Windley, Springer LNCS 1522, pages 321–333, 1998

[13] N.A. Harman and J.V. Tucker Algebraic models of microprocessors: architecture and organisation, *Acta Informatica*, 33 (1996), 421-456.

[14] W. A. Hunt. Microprocessor Design Verification. *Journal of Automated Reasoning*, 5(4):429–461, 1989.

[15] R. B. Jones, J. W. O'Leary, C.-J. H. Seger, M. D. Aagaard and T. F. Melham. Practical Formal Verification in Microprocessor Design. IEEE Design & Test of Computers, vol. 18, no. 4, July/August, 2001, pages 16–25.
`ftp://ftp.dcs.gla.ac.uk/pub/users/tfm/Pubs/PracFV.pdf`

[16] Rajeev Joshi and K. Rustan M. Leino. A semantic approach to secure information flow, Science of Computer Programming, 37, 1–3, pages 113–138, 2000
`citeseer.nj.nec.com/joshi00semantic.html`

[17] J. Joyce. Formal Verification and Implementation of a Microprocessor. In G. Birtwistle and P. Subrahmanyam, editors, *VLSI Specification, Verification and Synthesis*, pages 129–159. Kluwer, 1987.

[18] T. F. Melham. Higher Order Logic and Hardware Verification. Cambridge Tracts in Theoretical Computer Science, vol. 31, Cambridge University Press, 1993.

[19] S. Miller and M.Srivas. Formal Verification of the AAMP5 Microprocessor: a Formal Case Study in the Industrial Use of Formal Methods. In *Proceedings of WIFT 95, Boca Raton*, 1995.

[20] J Strother Moore. A Grand Challenge Proposal for Formal Methods: A Verified Stack. 10th Anniversary Colloquium of the UN University International Institute for Software Technology: Formal Methods at the Crossroads, Lisbon, Portugal, March, 2002.
`www.cs.utexas.edu/users/moore/publications/grand-challenge.ps`

[21] J. Rushby. Design and Verification of Secure Systems. ACM Operating Systems Review Vol. 15, No. 5, pages 12-21.

[22] A. Sabelfeld and A. Myers. Language-Based Information-Flow Security. IEEE Journal on Selected Areas in Communications, 21(1), 2003.

[23] J. Sawada and W. Hunt. Processor Verification with Precise Exceptions and Speculative Execution. In *Tenth International Conference on Computer Aided Verification, Vancouver, Canada*, July, 1998.

[24] Konrad Slind. A Verification of Rijndael in HOL. Supplementary Proceedings of TPHOLs'02, NASA Conference Proceedings CP-2002-211736.
`www.cs.utah.edu/~slind/papers/rijndael.pdf`

[25] Kong Woei Susanto. An Integrated Formal Approach for System on Chip. Proceedings of *The International Workshop in IP Based Design 2002*, October 2002, Grenoble, pages 119-123.

[26] Julien Schmaltz and Dominique Borrione. Validation of a Parameterized Bus Architecture Model. Fourth International Workshop on the ACL2 Theorem Prover and Its Applications (ACL2-2003).
`www.cs.utexas.edu/users/moore/acl2/workshop-2003/`

[27] S. Tahar and R. Kumar. A Practical Methodology for the Formal Verification of RISC Processors. *Formal Methods in Systems Design*, 13(2):159–225, September 1998.