Formal Specification and Verification of ARM6

M.J.C. Gordon
Computer Laboratory
University of Cambridge

Final Report to EPSRC on grant GR/N13135

The titles of the sections that follow are taken from:

NOTES FOR GUIDANCE ON COMPLETING AN INDIVIDUAL GRANT REVIEW FORM AND REPORT (Form NX0119).

1 Background/Context

The use of theorem proving for processor verification started to be investigated in the 1980s and has continued since then. Early work done by groups at Cambridge [3], Calgary [18, 14] and Austin [16] established feasibility on simple academic designs. Following this first phase, two threads of research emerged: continued study of academic designs with increasingly sophisticated execution engines [23, 21, 2], and the application of theorem proving to fragments of real processors [20, 17].

At the start of our project it was thus established that complete academic processors and parts of commercial processors could be formally verified by automated theorem proving. Our goal was to investigate whether it was feasible to completely verify a real-world commercial processor, and if so to calibrate the effort needed. After discussions with ARM Ltd we chose ARM610 [1] as our target. This processor is similar to the still widely used ARM7, but embodies less sensitive IP and so is appropriate for public domain research (all our formal models are available on the web[1]). ARM6 has a 3-stage pipelined Von Neumann Architecture. The differences between it and ARM7 are that the latter has (i) a hardware debug capability, (ii) the 'Thumb' instruction architecture to support both 16-bit and 32-bit instruction formats and (iii) an enhanced multiplier.

In parallel to the automated theorem proving research described above, a group lead by Tucker and Harman at the University of Wales Swansea were evolving a method of structuring processor specifications using algebraic concepts [4]. They also developed pencil-and-paper proof methods and, with a PhD student Anthony Fox, applied these to verify by hand superscalar implementations of Hennessey and Patterson's widely used pedagogical DLX RISC processor [5, 13]. When the current project was funded, we recruited Fox as our postdoctoral research assistant, and it was thus natural to see if the Tucker/Harman approach was suitable for mechanisation. It turned out to work very well indeed, and this is the approach we ended up using to specify and verify ARM6.

The project was a collaboration with Professor Graham Birtwistle's group at the University of Leeds, who produced accurate functional simulation models of the ARM6 architecture in Standard ML. Dominic Pajak produced a specification of the ARM programmers view and Daniel Schostak produced a model of the micro-architecture [22] (a three-stage pipeline implementation). Both these models were validated by extensive testing. Pajak and Schostak spent

[1] `http://www.cl.cam.ac.uk/~mjcg/ARM/`

time during their PhD studies as summer interns at ARM, where they had the opportunity of discussing their models with experts, and are now both employed there full time.

Anthony Fox took Pajak and Schostak's models and produced formal specifications in higher order logic of the programmers view and the micro-architecture. The algebraic specification and proof method developed by Tucker, Harman and Fox at the University of Wales Swansea, was used as the basis for a mechanical verification by theorem proving that the micro-architecture correctly implemented the programmers view.

2 Key Advances and Supporting Methodology

We believe we have advanced the field through methodological insights into the deployment of theorem proving on a relatively large scale processor verification. We now have a much more accurate understanding of the effort required to verify future ARM-like processors, and we have a formal platform for future research by us and others.

An overview of the main scientific results of the project are listed below.

- As far as we know this project accomplished the first formal verification of all the instructions of a Commercial Off-The-Shelf (COTS) processor.
- Calibration of the current state-of-the-art of processor verification by theorem proving: ARM6 took two person years (future proofs would be faster due to experience gained).
- Mechanisation of the 'Swansea Approach' of Tucker and Harman and confirmation that it is effective on industrial scale examples.
- Importance of tight integration of symbolic execution and deduction. In particular, we used a showed the benefits of a functional approach (similar to Boyer/Moore modelling) over the relational methods traditionally used with higher order logic [19]. Although this made non-determinism harder to represent, symbolic execution becomes easier.
- Value of an abstraction/refinement approach to overcome complexity of detail when verifying multi-cycle instructions. At first we found the ARM implementations of multiply and and block data transfer instructions intractable, but by verifying abstracted versions and then adding detail by refinement we were able to complete the formal proofs.

3 Project Plan Review

The objectives in the proposal were as listed in the following box:

1. to advance the practice of formal hardware specification and verification via a major industrial-strength case study
2. to develop an executable formal specification of the ARM6 microprocessor at the instruction set and at the pipeline levels of architecture
3. to formally verify the equivalence between a manageable yet representative subset of these descriptions
4. to formally specify and reason about selected, challenging ARM6 subsystems, some current, some of future interest

We ended up achieving 1, 2 and 3 of these, but we did not attempt 4. There were two reasons for this.

(i) Towards the end of the project we realised that not only would we be able to *formally verify the equivalence between a manageable yet representative subset* (see item 3 above) – we might even be able to verify the implementation of *all* the ARM6 instructions. As this would be a first, and enable exciting future research, we decided to concentrate on achieving this, which we eventually did.

(ii) We had intended that 4 would be partly tackled by a PhD student attached to the project. We recruited a student, but at the last minute he was awarded a personal studentship from Trinity College, Cambridge, which he decided to accept. Unfortunately, we were unable to recruit a replacement within the timescale of the project, so the studentship remained unfilled.

In the rest of this section we give a very brief overview of the programmer and micro-architecture views of ARM6 and the formal verification of their equivalence that we accomplished. This work was extremely detailed and so we cannot do more than summarise it here. However, a longer overview in available online in a paper entitled *ARM6 Formal Verification: Experience with a Commercial Microprocessor* [8], and details of the verification of the two most complex instruction classes are also available online: data transfers [6] and multiply [7].

3.1 Programmer's view

Version 4 of the ARM architecture was modelled. This has the following features.

- It is a 32-bit RISC architecture.
- There are six operating modes and the registers are arranged into overlapping banks. The program counter is register fifteen.
- There is a program status register (CPSR) and five saved versions (SPSR registers).
- All instructions are conditionally executed (there are four condition flags).
- There are seven types of exceptions: reset, undefined instruction, software interrupt, prefetch abort, data abort, normal interrupt and fast interrupt.
- There are eight main instruction classes (Table 1) plus coprocessor instructions.

3.2 The micro-architecture

The ARM6 is a three stage pipelined processor with a multi-cycled execute stage. A swap instruction, for example, is fetched, decoded and then takes four (or six) cycles to execute.

The HOL specification of the ARM6 was derived from Schostak's specifications, which consisted of a number of tables. The HOL specification represents each table by a function in higher order logic. The overall cycle level behaviour of the processor is specified using a next state function in a style derived from the Tucker/Harman algebraic approach.

Table 1: The ARM instruction classes.

Class	Instructions
Branch and Branch with Link	B, BL
Data Processing	ADD, ADC, SUB, SBC, RSB, RSC, CMP, CMN, AND, ORR, EOR, MOV, MVN, BIC, TST, TEQ
Multiply and Multiply Accumulate	MUL, MLA
PSR Transfer	MRS, MSR
Single Data Transfer	LDR, STR
Block Data Transfer	LDM, STM
Single Data Swap	SWP
Software Interrupt and Exceptions	SWI

The initial HOL processor model left out: hardware interrupts; coprocessor instructions; swaps; multiplies and block data transfers. The design was progressively extended with the inclusion of the swaps, followed by the block transfers and then the multiplies. At each stage the design was verified with respect to an instruction set model which only covered the instructions implemented. This approach enabled working verifications to be completed (and archived) before adding new features which would take some time to verify.

3.3 The formal verification

The correctness of the ARM6 is expressed using data and temporal abstraction maps. The data abstraction projects out the memory and registers from the processor's state space. The processor's program counter has value $pc+8$ because it is used for instructions fetch (i.e. it is two instructions, or eight bytes, ahead of the instruction being executed) and the data abstraction accounts for this by subtracting eight. It is shown that the data abstraction is a surjective map from the initial states implementation to the initial (all) states of the specification; this proves that the implementation is not partial (or trivial). The temporal abstraction is defined using a *duration map*: this gives the number of cycles needed to complete instruction execution from a given processor state (it is similar to clock functions used in the Boyer-Moore approach).

Store instructions require special attention when the memory address is $pc + 4$ or $pc + 8$; instruction fetch and decode are invalidated by this localised self modification of code. Two approaches to this were tried before settling on a third solution. The first approach was to block writes to these addresses and the second solution was to 'fix' the processor implementation by ensuring that the pipeline's state is correctly updated. Both of these methods have the disadvantage that they do not reflect the actual ARM6 behaviour. The third method was to modify the ISA model so as to reflect the pipelined behaviour; this was comparatively simple to specify and verify. The data abstraction projects out the opcodes of the fetched and decoded instructions.

The main top-level correctness statement conforms to the Tucker/Harman framework using the ARM temporal and data abstractions we developed. It is an algebraic formulation of the

normal commuting digram relating abstract and concrete state machines.

4 Research Impact and Benefits to Society

The main beneficiaries of this work are potential formal verifiers of ARM-like processors. We have established the effort needed for this (2 person years) and have accumulated much methodological wisdom on how to conduct such proofs.

We have created complete and public-domain[2] models of ARM6 and we hope others can build on this (see Section 6).

A side-effect of the work has been enhancements to the HOL4 public-domain system, particularly support for words and symbolic execution.

5 Explanation of Expenditure

As explained in Section 3, we didn't appoint a PhD student to the project but, with EPSRC's permission, studentship funds were used to extend the appointment of Dr. Anthony Fox.

6 Further Research or Dissemination Activities

We have submitted a successor project to EPSRC entitled *Formal Specification and Verification of ARM-based Systems*. Details of this application are available on the web[3] and it is currently being refereed.

We have had contact from both a company and a University (both in the USA) about future formal verification research on XScale (Intel's implementation of ARM). We are also liaising with Professor Konrad Slind (of the University of Utah), a past collaborator, about using our verified ARM6 models to create highly assured implementations of AES encryption, which he has modelled and analysed in higher order logic. If our successor project is funded, then we hope to explore and deepen these contacts further.

We have also had fairly detailed discussions with a US Government Agency about using our models to support the formal verification of ARM7 assembly code implementing Elliptic Curve Cryptography (ECC) primitives. An examination by Anthony Fox of examples of hand proofs of such code shows that our models should be suitable for this task, and we hope to pursue this.

References

[1] ARM Ltd. *ARM610 Data Sheet*. ARM Ltd, Document 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] A. J. Cohn. A Proof of Correctness of the VIPER Microprocessor: The First Level. In G. Birtwistle and P. A. Subrahmanyam, editors, *VLSI Specification, Verification and Synthesis*, pages 27–71, Norwell, Massachusetts, 1988. Kluwer.

[2] `http://cvs.sourceforge.net/viewcvs.py/hol/hol98/examples/arm6/`

[3] `http://www.cl.cam.ac.uk/~mjcg/proposals/ARM2/`

[4] A.C.J. Fox and N.A. Harman. Algebraic Models of Correctness for Microprocessors. Formal Aspects of Computing, 12(4): 298-312, 2000.

[5] A.C.J Fox and N.A. Harman. Algebraic models of correctness for abstract pipelines. The Journal of Logic and Algebraic Programming, 57(1-2): 71-107, 2003.

[6] A.C.J Fox. Verifying the ARM Block Data Transfer Instructions. Presented at DCC 2004, Barcelona, 2004. `http://www.cl.cam.ac.uk/users/acjf3/papers/bdt.pdf`

[7] A.C.J Fox. Verifying ARM6 Multiplication. Unpublished report. `http://www.cl.cam.ac.uk/users/acjf3/papers/mul.pdf`

[8] A.C.J Fox. ARM6 Formal Verification: Experience with a Commercial Microprocessor. To be presented under Emerging Trends at TPHOLs '04, Utah, USA, 2003. `http://www.cl.cam.ac.uk/users/acjf3/papers/r.pdf`

[9] 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.

[10] A.C.J. Fox. Formal verification of the ARM6 micro-architecture. Technical report No. 548, University of Cambridge Computer Laboratory, November 2002. `www.cl.cam.ac.uk/users/acjf3/papers/tr548.ps.gz`

[11] 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`

[12] 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`

[13] 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`

[14] Brian T. Graham. *The SECD Microprocessor, A Verification Case Study.* Kluwer International Series in Engineering and Computer Science. Kluwer Academic Publishers, Boston, 1992.

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

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

[17] 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`

[18] 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.

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

[20] 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.

[21] 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.

[22] D. Schostak Methodology for the Formal Specification of RTL RISC Processor Designs (With Particular Reference to the ARM6), PhD thesis, The University of Leeds, 2003. `http://www.comp.leeds.ac.uk/research/pubs/theses/schostak.pdf`

[23] 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.