DRAFT

The Notion of Proof in Hardware Verification

Avra Cohn

University of Cambridge Computer Laboratory
New Museums Site, Pembroke Street
Cambridge, CB2 3QG, England.

Abstract:

Recent advances in the field of hardware verification have raised some fresh (and some familiar) issues to do with the scope and limitations of formal proof. In this note, some of these are considered in the context of the Viper verification project. Viper is a microprocessor designed by W. J. Cullyer, C. Pygott and J. Kershaw, of the Royal Signals and Radar Establishment of the U.K. Ministry of Defense, for use in safety-critical applications. Much to their credit, the designers intended from the start that Viper be formally verified; they presented Viper's more abstract specifications in a language suitable for formal reasoning, and they placed the design in the public domain. Viper microprocessors are currently being marketed as verified chips. The formal proof aspects of the verification work have been carried out at the Computer Laboratory of the University of Cambridge. To date, some important properties of a register-transfer level model of Viper, relative to a more abstract functional specification, have been proved (by the author) using the HOL proof generating system. 'Verified' systems such as Viper seem likely to become commonplace in the near future. Whilst proofs about the abstract models of such systems are obviously a vital contribution to our trust in them, it is also important (not least in safety-critical applications) that the limitations of the approach be understood.

Some of the material in this note appears in [7].

1 Introduction

The verification of hardware systems has recently become an attractive application area for theorem provers for several reasons. First, hardware verification is in many ways a more tractable problem than software (program) verification; it is often easier to write a clear specification which captures the functionality of a system for hardware than for software – and hardware proofs tend to have a certain uniformity of structure which is well suited to mechanical treatment. Second, there are

compelling economic reasons for trying to get hardware correct early on, as correcting errors in a chip can involve expensive re-fabrication, not merely the editing of text. Finally, it is becoming increasingly important to invest time and effort in the verification of hardware which is intended for *safety-critical* applications.

Several issues pertaining to the scope and limitations of verification have recently been raised by the project to formally verify aspects of the Viper microprocessor at the University of Cambridge. In this note, some of the issues are discussed; although this is done in the context of Viper, the remarks may be more general. The intention is to encourage intelligent understanding of the sense in which a piece of hardware can be called "verified", and not to undermine research in verification, or to discredit this vital work. As devices begin to appear on the market purporting to be "verified" or "mathematically proved" – possibly with the implication that they cannot therefore fail – a sharp watch must be kept for unqualified claims, and for failures to convey the sense, extent and nature of the verification effort. This applies particularly to very hazardous applications such as nuclear power plant control.

Various computing systems in recent years have been claimed to do verification or proof, or something akin, when in fact they were doing something distinct from (if no less valuable than) formal proof in an explicit and well-understood logic. That is, they have done simulation, or reasoning in *ad hoc* logical systems, or informal reasoning, etc. For present purposes, "verification" is taken to mean formal proof in the usual mathematical sense of a sequence of valid inference steps.

2 The Viper Microprocessor

Viper [8,9,20,10,11,24] is a microprocessor designed by W. J. Cullyer, C. Pygott and J. Kershaw at the Royal Signals and Radar Establishment of the U.K. Ministry of Defense (henceforth "RSRE") for use in safety-critical applications such as civil aviation and nuclear power plant control. Viper chips are now commercially available. They are currently finding uses in areas such as the deployment of weapons from tactical aircraft [12]. To support safety-critical applications, Viper has a particularly simple design; for example, interrupts in the usual sense are not permitted; the instruction set is kept to a minimum; and the machine is designed to stop if it detects itself in an error or illegal configuration. (The stopping feature is intended to support the running of several Vipers simultaneously for increased reliability.) The simplicity of the design makes it amenable to formal analysis using current techniques.

Aspects of the formal, mechanical verification of Viper were subcontracted to the Hardware Verification Group at the University of Cambridge from early 1986 to late 1987. The results of this project are reported in [6] and [7]. A pilot study for the main proof is reported in [5].

3 The HOL Verification System

The verification of Viper has been approached within HOL (Higher Order Logic) [2,14,15], a theorem-proving system derived from R. Milner's LCF system (Logic for Computable Functions) [13,23] and based on the version of higher order logic

formulated by A. Church [3]. HOL was implemented by M. Gordon at the University of Cambridge and is currently in use by the Hardware Verification Group at Cambridge and at several sites throughout the world. "Verification" was understood by Viper's designers at RSRE (as by the LCF and HOL communities) to mean complete, formal proof in an explicit and well-understood logic. Proofs in HOL are normally constructed interactively, combining machine assistance with user-guidance, and not fully automatically (although the extent to which this is so is a function of user-designed proof strategies).

4 The Models of Viper

The designers of Viper, who deserve a great deal of credit for the promotion of formal methods, intended from the start that Viper be formally verified. Their approach was to specify Viper in a hierarchy of decreasingly abstract levels, each of which concentrated on some specific aspect of the design. That is, each level was to be a specification of the more abstract level above it (if any), and an implementation of the one below (if any). The verification effort would then be simplified by being structured according to the abstraction levels. These levels of description were characterized by the design team at RSRE. The first two levels, and part of the third, were written by them in a logical language amenable to reasoning and proof (a predecessor of HOL's higher order logic). (The systematic study of abstraction hierarchies and mechanisms in the modelling of hardware is discussed by T. Melham in [21] and [22].)

The highest level specification of Viper is a simple state transition function describing the way in which an abstract state (representing Viper's memory and its visible registers) changes as Viper executes each of its possible instruction types (see [8] and [6] for details). The specification is thus an operational semantics of the instruction set. It characterizes no more than the fetch-decode-execute cycle of Viper; it does not specify all of the possible behaviours of the actual microprocessor. In particular, the capacity of Viper to be *reset* externally by an operator (i.e. to have its registers cleared from the outside) is not covered; nor is its capacity for 'timing itself out' due to memory failure after some fixed number of clock cycles. As will be discussed, any verification of a more concrete model relative to this top level specification must consequently be limited to the behaviour manifest at the top level. That is, no such proof can establish that Viper resets or times-out in an acceptable way.

The next (more concrete) level is called the *major state* level. At this level, an instruction is processed via a sequence of events rather than in a single step. An event may affect the visible registers or the memory of the top level specification, *or* any of several internal registers (which comprise the *internal* state). These internal registers are still part of an abstract view of Viper, and do not necessarily correspond to parts of the actual Viper chip. The 'next' event in a sequence is determined according to the current event, the visible state, and the internal state; and some events are recognizably terminal and some initial, in the sequences. From all of this, a new state transition function can be extracted and its properties established by proof. In particular, the cumulative result of the sequence of events which processes each of the instruction types can be inferred, and then compared to the result of the corresponding high level state transformation function.

The 'block model' is the most concrete level considered in the formal verification project (although the RSRE design continues down to the gate-level circuit design, which could in principle also be formally verified). The block model was presented by the designers in a form which was partly pictorial and partly textual (and functional). The model consists of 'blocks', that is, computational units such as Viper's instruction decoder, its arithmetic-logic unit (ALU), and its memory. Information passes between blocks, and to/from the outside world at fixed clock cycles. The functional specifications describe only the internal combinational logic of the various blocks. Neither their behaviour over time (e.g. the delay units which give them memory), nor the connections between separate blocks are covered by the functional specifications; the pictorial specification fills in the rest of that information. Much as at the major state level, the concept of single instructions being processed via sequences of steps is built into the block model. In addition, several smaller steps implement each major step. The block definitions and the pictorial information were supplied by the designers at RSRE; a fully formal description had to be constructed from these sources, and the block machine's behaviour patterns had to be logically inferred from the formal description.

The block model isolates the computational behaviour of Viper; to relate it to either of the more abstract specifications (the top level specification or the major state model), computational behaviours such as additions, shifts, negations and comparisons had to be considered in detail. The block model also specifies more of the actual behaviours of Viper (e.g. the behaviour of resets and time-outs) than appear in the top level formal specification. At the block level, one begins to approach the functional units and connectivity of the actual circuit, though still in a rather abstract way.

5 The Viper Verification Project

The correctness proof of the major state level of Viper relative to its top level specification was straightforward (if lengthy) in HOL, since the possible execution sequences of the model were explicitly given. That is, the conditions under which one event follows another were explicitly defined. The proof consisted, therefore, of a number of cases (one for each instruction type), in which the cumulative effects of the sequence of events processing that instruction type were inferred from the definition of the model. In each case, the effects were then proved equivalent to the effects specified at the top level, for components appearing at both levels. It also had to be proved that every possible execution sequence has been considered, to justify the case analysis.

The correctness of the block model is more difficult to establish. The first task in the verification effort was to derive a functional specification of the block model in a formal logic suitable for reasoning and proof, since it is not obvious how to reason formally about a schematic diagram indicating the transfer of information to and from its sub-units simultaneously.

The second task was to infer the behaviour of the block model *using* its functional representation. As in the major state level proof, it first had to be inferred, for each instruction type, what were the accumulated effects on the registers of the block model after all of the steps which process that instruction had been performed. This involved extracting from the formal representation: (i) the conditions under which

one step lead to another, and (ii) any assumptions that had to be made about initial states and 'normal' behaviour in order to resolve the state transitions. (These were implicitly determined by the functional representation of the block model; at the major state level, the conditions were given explicitly, and no assumptions were required.) Normal behaviour means behaviour which is within the scope of the high level specification. For example, as mentioned, it had to be assumed in the verification of the Viper block model that the machine was not reset at any time during the course of processing an instruction; and that the block machine's time-out facility was never invoked. The initial conditions, for example, had to include the assumptions that at the start of processing each instruction (i) the time-out counter was not set at its maximum value, and (ii) no errors were being signalled. It also had to be shown, as before, that the state transition conditions covered all logical possibilities, to ensure that no possible instruction types had been omitted from the analysis.

The third task would be to verify the results of the block model relative to the results of the top level transformation function at each instruction type. The first and second tasks of the block model verification have been completed to date, giving a provably correct and complete description of the behaviour of the formal representation of the block model (under the assumptions mentioned above); but for practical reasons, the third task has not been completed. All three tasks are discussed in [7].

6 Limitations of Proof in Hardware Verification

The notion of formal proof began to receive serious attention in its own right just before the age of computing. Since computers have been used to assist with formal proofs, there has been renewed discussion of what proof is and what it actually ensures. This may be in part because there is no prior reason to insist that machines construct proofs in the *way* that mathematicians do; nor is there yet any well-agreed 'standard of evidence' that a proof has been successfully completed by a machine, of the sort that mathematicians are required to supply. In this section, attention is drawn to some of the fresh concerns which have been raised by the Viper verification project.

6.1 Chips and Intentions Cannot be Verified

Ideally, one would like to prove that a chip such as Viper correctly implemented its intended behaviour in all circumstances; we could then claim that the chip's behaviour was predictable and correct, as intended. In reality, neither an actual *device* nor an *intention* are objects to which logical reasoning can be applied. The intended behaviour rests in the mind of the architects and is not itself accessible. It can, of course, be reported in a formal language – but not with checkable accuracy. Similarly, a material device can only be observed and measured; it cannot be verified. Again, a device can be described in a formal way, and the description verified; but as with intentions, there is no way to assure the accuracy of the description. Indeed, any description is bound to be inaccurate in *some* respects, since it cannot be hoped to mirror an entire physical situation even at an instant, much less as it evolves through time; a model of a device is necessarily an abstraction (a simplifi-

cation). In short, verification involves a pair of *models* which bear an uncheckable and possibly imperfect relation to the intended design and to the actual device.

Although these points seem obvious, they are not merely philosophical quibbles. Errors were found both in the top level specification of Viper and in its major state model, none of which was either intended by the designers *or* evident in the manufactured Viper chips. (These errors are discussed in [6].) The errors were fairly minor and quickly repaired, but their presence highlights the rather limited sense in which an actual product can be said to have been verified against the architect's intended design or against the actual chip: there remains the danger that – secure as the proof may be – the models themselves may be wrong.

There is no complete solution to this problem, but there are avenues of approach to be explored. In particular, as we produce clearer and more concise and readable abstract specifications, their intuitive plausibility should be increased. At the other extreme, as we devise more realistic and detailed models, their correspondence with actual devices should become more convincing. Attention has been drawn to these points by T. Melham [21].

6.2 Links between Designer, Verifier and Manufacturer

That the actual Viper chips appear not to suffer from the errors found in the models also illustrates the still quite abstract nature of the research described in [6] and [7]. The chips were already in the process of being built by the time the subcontracted verification work began on the major state model at Cambridge; and they had been built and were being advertised by the time the work on the block model was undertaken. Whilst it is possible in theory that an error in an abstract specification had been reflected in the circuit design given by RSRE to the manufacturers – the abstract specifications were no doubt in the architects' minds while they designed the circuit – it seems more likely, because of the indirect links between the designers' abstract specifications, the circuit design process, the manufacturers, and the verifiers, that problems in the specification would *not* propagate down to chip problems. In fact, it would seem to be the case that the manufacturers worked from different 'design texts' than the verifiers. Until common models in a common language are adopted, we are only studying models which bear an informal connection to the devices they are modelling. In this respect, too, there is good reason to hope that a common language will be agreed and an integrated approach taken in the future.

6.3 The Lack of a Fully Formal Description

At more concrete levels of description, the situation may be further complicated by not beginning with fully formal descriptions. For example, Viper's top level specification and its major state level were both supplied in a logical language; but its block level model was given was partly formally and partly pictorially (as was natural). Combining these two parts required both ingenuity and some guesswork. The guesses were based on the co-incidence of line names, on the names of bound variables in function definitions, and on annotations in the text of the definitions. None of these notational devices can be regarded as a formal specification. Before verification can be meaningfully applied in such cases, a fully formal description must be produced. Once again, however, accuracy cannot be checked; the new

formal description may be a flawed translation of the pictorial specification, or a flawed combination of picture and text, but this cannot be rigorously tested. One may therefore end up proving properties of a formal description bearing an imperfect relation to the intended design - and possibly never know it.

In fact, this *was* a problem in the block level representation of Viper; in the author's first attempt at a formal representation of the Viper block diagram, there were a pair of interchanged line names. This flawed description was subsequently used to deduce plausible-looking block results. The error in the representation was discovered (rather later in the proof) and only by an unsystematic inspection. This problem of the accuracy of a representation could appear at the gate level, the transistor level, or any other level at which a linguistic description has to be constructed creatively from a pictorial one, i.e. at which diagrams are the usual and natural mode of specification. This further limits the sense in which a system can be called verified.

This problem is at least partly addressed by the previous section; if the designers, for example, are in a position to read and scrutinize the formal description derived from the informal specification, they may well be able to spot mistakes, particularly those which require a deep understanding of the design.

6.4 The Level and Completeness of the Models

As verification relates a less abstract implementation to a more abstract specification, it is important to be explicit about the level of abstraction and the degree of completeness of the models in question. We say that a device has been verified "at the major state level" or "at the register transfer level", and so on – it is not enough to say simply "verified". For example, Viper's major state machine has been fully verified with respect to its top level specification; but the proof establishing the equivalence of these two sets of results depends only on the flow of control in the two models, and does not depend on any of the computational behaviours of Viper. (That is, the same formal expression represents the arithmetic-logic unit in both levels, so that expression is never evaluated.) Therefore, the fact that Viper has been verified "at the major state level" does not actually ensure very much; the essence of the microprocessor (the behaviour of its ALU) has not, at that stage, been treated. Viper certainly could not, on that basis alone, be usefully called "verified".

The block model of Viper does concern itself with Viper's arithmetic and logical operations, and with the transfer of information between registers and memory. Thus verifying Viper to the block level would be a significant step towards a "verified" microprocessor. (In any case, the proof has not been fully completed at this level.) However, the block model does not concern itself with gate layout, transistors, electrical effects, timing problems, or many similar areas in which unsuspected errors would seem particularly likely to be appear. (In those areas, enormous amounts of research remain to be done on finding useful, tractable models, even before we begin to verify them.) Thus, again, the term "verified" cannot be properly used without an indication of the levels of the models involved. At *every* level of abstraction, some properties are included and some ignored..

In addition, the models involved may be incompletely specified. For example, Viper's highest level specification is complete only for the processing of instructions,

and does not cover such features as resetting or timing-out the machine, or other possible behaviours specified at the block level. This, from the outset, restricts any analysis to the high level behaviour alone, again missing the more subtle and perplexing issues.

6.5 Normality Assumptions

In discussing what was proved in the Viper verification project, it was indicated that certain assumptions had to be made (about initial and normal behaviours) in order to infer the cumulative effects of processing instructions. These assumptions are perfectly natural, and reflect the fact that devices are intended to operate only under certain conditions. The only cause for concern here is if these assumptions are ignored when claims are made about what was proved. In the formal correctness statement, of course, any persistent assumptions will appear explicitly as the antecedents of an implication. It is in informal summaries (advertising material and so on) that the assumptions can easily be overlooked.

In the end, for example, the effect of each of Viper's instructions on the registers of its block model was deduced. This was done by assuming that the machine was initialized in a reasonable way, and assuming that that it was run under certain ideal conditions. The effect was not deduced, say, of assuming that a reset operation could occur – it could have been, but to no useful end, since that effect is not specified at the top level. Thus, even a fully verified block design could remain incorrect in its resetting behaviour, and the error could propagate, despite the proof, down to the chip itself. This illustrates the importance of knowing the conditions under which the block model has been analyzed.

6.6 Putting Formal Proof in Context

Finally, the correctness of an abstract representation of a system must be placed in context when we talk about its reliability in safety-critical applications. The author claims no expertise in the field of reliability, but this much is obvious: that an abstract and limited sense of correctness (for example, for Viper, the equivalence of a register transfer level specification to a functional specification of the fetch-decode-execute cycle) is only one of many issues which have to be considered collectively. Aside from possible problems at more concrete levels of description, which have already been discussed, safety will also depend on factors as yet outside of the world of formal description: these range from issues of social administration and communication, as well as staff training and group behaviour, at one end, to the performance of mechanical and chemical parts, and so on, at the other. One has only to contemplate the mass catastrophes of the last ten years or so to perceive the predominant role played by these extra-logical factors.

It is the author's guess (though, again, not an expert opinion) that the sort of abstract design correctness discussed here, though of undoubted importance, is still a relatively minor contribution to the overall reliability of real systems. This seems so at least at the present state of research into representation and proof, and with the present weak links between designer, verifier and manufacturer. That is, using a hardware design verified only at a fairly abstract level – and only under idealized operating conditions – as part of the control system in very hazardous

applications (in which large populations or land masses may be destroyed) does not yet seem significantly safer than using any other design. If only because of the number of extra-logical factors involved, the use of the word "verified" must under no circumstances be allowed to confer a false sense of security.

7 Conclusions

Various of the limitations on the use of the word "verified" are obscured in claims such as the following (both taken from promotional material):

> "VIPER is the first commercially available micropressor with ...a formal specification and a proof that the chip conforms to it." [26,27]
>
> "One unique feature of Viper is that the instruction set is specified mathematically ...and the gate-level logic design has been proven to conform to this specification." [16]

As discussed, a chip as such cannot be verified – but this is perhaps just an imprecise use of words. The second example, depending on one's interpretation of "proven", could be called a false claim; no formal proofs of Viper (to the author's knowledge) have thus far been done at or near the gate level. The gate level design of Viper *has* been checked by C. Pygott using an innovative simulation method called intelligent exhaustion [25], but it has not yet been formally verified. Such assertions as those quoted, taken as assurances of the impossibility of design failure in safety-critical applications, could have catastrophic results. To summarize:

- Neither an intended behaviour nor a physical chip is an object to which the word "proof" meaningfully applies. Both an intention and a chip may themselves may be inadequately represented in formal language, and this is not itself verifiable.
- Because of the present weak links between designer, verifier and manufacturer, it is not at all obvious that errors deduced in very abstract specifications are likely to manifest themselves in actual products. We must then ask how much extra security verification currently affords. (This is an argument for continued research, not against verification!)
- Any verification effort is necessarily limited to those behaviours specified at the most abstract level. It should be clearly stated when a system is called "verified" which actual features are not covered.
- It should also be clearly stated to what level of concreteness the specifications extend. It seems fair to expect that the more concrete the models, the greater is the likelihood of finding errors in the design, particularly errors which would propagate through to the actual product. Since any model is an abstraction of a material device, it is *never* correct to call a system "verified" without reference to the level of the models used.
- Any working assumptions about initial states or normal behaviours should also appear in verifications claims. Particularly in informal descriptions, the assumptions may not always be evident.

- A proof that one specification implements another – despite being completely rigorous, expressed in an explicit and well-understood logic, and even checked by another system – should still be viewed in context of the many other extra-logical factors which affect the correct functioning of hardware systems. In addition to the abstract design, everything from the system operators to the mechanical parts must function correctly – and correctly together – to avoid catastrophe.

For a long time, mechanical theorem-proving was sufficiently difficult that researchers frequently drew upon simple (or occasionally less simple) mathematical problems on which to exercise their mechanical proof systems. Advances in theory as well as in technology have now made proof efforts feasible which once appeared impossibly large, uneconomic, or labour-intensive. Sophisticated theorem-proving environments, together with modern work-stations, operating systems and editors, have supported this progress. The proofs, for example, of the basic theorems of arithmetic [1], or of the correctness of schematic compiling algorithms [4] – to choose two examples – were challenging problems in their time, yet current verification efforts are focusing on properties of realistic (and sometimes commercial) hardware designs. Besides the Viper microprocessor, examples include the verification by W. Hunt in the Boyer-Moore system of the FM8501 [18], a computer designed (by Hunt) for the purpose of verification; the verification in HOL by J. Joyce [19] of Tamarack, a computer designed by M. Gordon, also for the purpose of verification; the verification in HOL by J. Herbert of the ECL chip [17], a network interface designed by A. Hopper as part of the Cambridge Fast Ring; and the verification in HOL by T. Melham of the T-Ring [22], a very simple ring network designed by D. Gaubatz and M. Burrows.

It would seem, in conclusion, that we are now beginning to be able to verify real hardware designs to useful levels of detail. None of the remarks in this note should be taken as pessimistic – just cautious. As "verified" hardware begins to be used in life-critical applications (which could include fly-by-wire aircraft, bomb deployment systems, nuclear power stations, medical equipment, automotive braking systems, railway signallers, and so on), it will become increasingly important to insist that the word "verified" and its synonyms are modified, qualified and explained so that we know exactly what claims are being made, and can assess them intelligently.

8 Future Work

At the beginning of Section 6, the problem was mentioned of establishing a standard of evidence for having achieved a proof in a mechanical theorem-prover. In this capacity, neither the long chains of primitive inferences which proofs comprise nor the particular procedures which have constructed these proofs have so far found much favour. The Viper block model proofs consist in several million primitive inference steps, for example; and the procedures which generate them comprise dozens of pages of code in the functional programming language ML. The question of proof evidence is typical of a variety of foundational issues which have not been broached in this note, but which at some point must also be addressed. For example, the consistency of any abstruse or special-purpose logic has to be established; this is a standard problem but not always easy. Worse, it could be asked on what basis we

place our confidence in the implementation of a theorem-proving methodology (and the operating system on which it runs, the hardware of which the host machine is built, and so on).

One pragmatic answer (which is a topic of planned research at Cambridge) is that we can reduce the number of systems in which we must trust by agreeing on a standard for 'proof deliverables'. That is, we could agree on a proof output format such that the proofs produced at one site could be independently (and mechanically) checked at another. This idea, in the context of hardware verification, is due to K. Hanna. Part of its attraction is that proof checking is generally much less difficult than proof construction.

Another research goal is to find a uniform representation language for everyone involved in producing a hardware device: designers, clients, verifiers, fabricators, etc. This would help to integrate the various communities, and thus to reduce the danger, for example, that the models which are verified differ from the plans used by the manufacturers. It also would increase the chances that the errors turned up by verification were actual errors in the physical devices. Higher order logic has been proposed for this purpose, but any more or less standard logic could be a candidate.

A very large step toward reliable systems would be a verification effort extending all the way from the software level down to the gate level. Research is currently being planned in this area (i) jointly at the University of Cambridge and at SRI International, Cambridge, and (ii) at Computational Logic Inc. in Texas.

Finally, research is continuing at various places into models for more realistic levels of representation of hardware, in the hope of expressing and locating the more subtle and worrying errors that beset digital systems. Once the models are found, there appears to be no shortage of theorem-proving tools with which to verify them.

9 Acknowledgements

Many thanks to Tom Melham and Mike Gordon for helpful comments and discussions. Thanks also to Thomas Forster. The opinions expressed are the author's alone. The Viper verification work at Cambridge was supported by a grant from RSRE. The preparation of this note was suggested by Larry Wos, and was supported by a grant from the U.K. Science and Engineering Research Council.

References

[1] R. S. Boyer and J S.Moore, A Computational Logic, Academic Press, 1979

[2] A. Camilieri, M. Gordon and T. Melham, Hardware Verification using Higher-Order Logic, Proceedings of the IFIP WG 10.2 Working Conference: From H.D.L. Descriptions to Guaranteed Correct Circuit Designs, Grenoble, September 1986, ed. D. Borrione, North-Holland, Amsterdam, 1987

[3] A. Church, A Formulation of the Simple Theory of Types, Journal of Symbolic Logic 5, 1940

[4] A. Cohn, Machine Assisted Proofs of Recursion Implementation, Ph.D. Thesis, Dept. of Computer Science, University of Edinburgh, 1979

[5] A. Cohn and M. Gordon, A Mechanized Proof of Correctness of a Simple Counter, University of Cambridge, Computer Laboratory, Tech. Report No. 94, 1986

[6] A. Cohn, A Proof of Correctness of the Viper Microprocessor: the First Level, VLSI Specification, Verification and Synthesis, eds. G. Birtwistle and P.A. Subrahmanyam, Kluwer, 1987; Also University of Cambridge, Computer Laboratory, Tech. Report No. 104

[7] A. Cohn, Correctness Properties of the Viper Block Model: The Second Level, Proceedings of the 1988 Conference on Hardware Verification, Banff, Canada (To be published by Springer-Verlag); Also University of Cambridge, Computer Laboratory, Tech. Report No. 134

[8] W. J. Cullyer, Viper Microprocessor: Formal Specification, RSRE Report 85013, Oct. 1985

[9] W. J. Cullyer, Viper — Correspondence between the Specification and the "Major State Machine", RSRE report No. 86004, Jan. 1986

[10] W. J. Cullyer, Implementing Safety-Critical Systems: The Viper Microprocessor, VLSI Specification, Verification and Synthesis, eds. G. Birtwistle and P.A. Subrahmanyam, Kluwer, 1987

[11] W. J. Cullyer, J. Kershaw and C. Pygott, forthcoming book on Viper

[12] C. Gane (Computing Devices Company Ltd.), Computing Devices, Hastings' VIPER-VENOM Project: VIPER in Weapons Stores Management, SafetyNet: Viper Microprocessors in High Integrity Systems, Enq. No. 021, Issue 2, July-August-September 1988, Viper Technologies Ltd., Worcester, England

[13] M. Gordon, R. Milner and C. P. Wadsworth, Edinburgh LCF, Lecture Notes in Computer Science No. 78, Springer-Verlag, 1979

[14] M. Gordon, HOL: A Machine Oriented Formulation of Higher-Order Logic, University of Cambridge, Computer Laboratory, Tech. Report No. 68, 1985

[15] M. Gordon, HOL: A Proof Generating System for Higher-Order Logic, University of Cambridge, Computer Laboratory, Tech. Report No. 103, 1987; Revised version in VLSI Specification, Verification and Synthesis, eds. G. Birtwistle and P.A. Subrahmanyam, Kluwer, 1987

[16] M. P. Halbert (Cambridge Consultants Ltd.), Selfchecking Computer Module Based on the Viper1A Microprocessor, SafetyNet: Viper Microprocessors in High Integrity Systems, Enq. No. 017, Issue 2, July-August-September 1988, Viper Technologies Ltd., Worcester, England

[17] J. Herbert and M. J. C. Gordon, A Formal Hardware Verification Methodology and its Application to a Network Interface Chip, IEE Proceedings, Computers and Digital Techniques, Special issue on Digital Design Verification, Vol. 133, Part E, No. 5, 1986; Also in draft version: University of Cambridge, Computer Laboratory, Tech. Report No. 66, 1985

[18] W. A. Hunt Jr., FM8501: A Verified Microprocessor, University of Texas, Austin, Tech. Report 47, 1985

[19] J. J. Joyce, Formal Verification and Implementation of a Microprocessor, VLSI Specification, Verification and Synthesis, eds. G. Birtwistle and P.A. Subrahmanyam, Kluwer, 1987

[20] J. Kershaw, Viper: A Microprocessor for Safety-Critical Applications, RSRE Memo. No. 3754, Dec. 1985

[21] T. Melham, Abstraction Mechanisms for Hardware Verification, VLSI Specification, Verification and Synthesis, eds. G. Birtwistle and P.A. Subrahmanyam, Kluwer, 1987

[22] T. Melham, forthcoming Ph.D. Thesis, University of Cambridge, Computer Laboratory

[23] L. Paulson, Logic and Computation, Cambridge University Press, 1987

[24] C. H. Pygott, Viper: The Electronic Block Model, RSRE Report No. 86006, July 1986

[25] C. H. Pygott, Formal Proof of a Correspondence between the Specification of a Hardware Module and its Gate Level Implementation, RSRE Report No. 85012, Nov. 1985

[26] Viper Microprocessor: Verifiable Integrated Processor for Enhanced Reliability: Development Tools, Charter Technologies Ltd., Publication No. VDT1, Issue 1, Dec. 1987

[27] Application for Admission and Registration Form, Second VIPER Symposium, RSRE, Malvern, England, 6-7 September, 1988