unless addressed. Additional questions need to be asked about the division of labour in the whole effort surrounding formal analysis of VHDL programs. Should, for instance, the approach taken by Siemens in providing an efficient oracle be taken? That way the symbolic simulator would be left to do what it does best, and the proof system is left unencumbered by a task for which it may not be suited. Certainly given a formal semantics for the language, such a system is not inconceivable. Further research must also be done into ways in which non-quiescent programs may be compared.

The way forward in the formal analysis of VHDL needs to stem from a firm understanding of the semantics of the full language. The work that has been presented here is a first step towards that goal. Further research, because of the real-world nature of its implications to the design of complex systems, represents a true test for the maturity of the formal methods approach. The benefits of verification over validation have been expressed for quite some time. Furthermore, formal systems and methodologies abound. A formal semantics and practical associated tools for full VHDL presents a unique and realisable challenge for the future.

It shows that, given the appropriate assumptions about quiescence and well-formedness, sub-components of a larger design may be replaced by equivalent ones.

7.2 The Methodology

The methodology used in working with the embedded semantics in Chapter 6 is analogous to those used in the projects described in Chapter 2. Specifically, it has been possible to symbolically simulate Femto-VHDL programs via proof to gain a general understanding of their behaviour. Once obtained, such behaviours may be compared, again through formal proof, for equivalence or non-equivalence to other similarly derived ones.

There are three problems with the methodology. The first is that the user must carefully specify the initial signal interrelationships in order to keep the analysis from degenerating into an exhaustive simulation. The second is that there a great deal of user interaction involved in producing any kind of result. Finally, analysis is limited to only those Femto-VHDL program which quiesce.

7.3 The Future

Formal analysis of the VHDL simulation model has been a fruitful exercise. The research presented here has however been constrained to only a subset of the language and a restricted form of reasoning. Furthermore, analysis of individual programs, despite automation of the semantics in a proof assistant, is also a very user-intensive process.

As noted earlier, a formal semantics of full VHDL is a useful object in and of itself, as it would provide a clear specification of the language for future implementors. Indeed, it has been used directly in the development of a symbolic simulator in the HOL system. It also has use as an unambiguous reference for language practitioners. Since VHDL is used by such a wide cross-section of the CAD community, such a specification of the full language would be of immediate relevance and use. Scope for additional research in the development of such a semantics could be found in crystallising the operation of user-defined the resolution functions. Recall that these been simplified out of the work presented here.

The methodology for reasoning about individual Femto-VHDL programs that has been presented could in theory be extended to cover the full language. The inherent problems associated with excessive user intervention will, however, persist

CHAPTER 7

CONCLUSIONS

In the previous chapters, an exposition of the semantics of Femto-VHDL, a subset of full VHDL, was presented. The semantics was defined in an operational manner that closely adheres to the informal description of the simulation model of full VHDL. Furthermore, it has been embedded in the HOL system in order to automate reasoning about programs written in the subset.

7.1 The Semantics

In contrast to the approaches highlighted in Chapter 2, the semantics given in Chapters 3 and 4 was derived independently of the proof assistant in which it has been embedded. As a result, it is not bound by constraints of the object system, and the most suitable formalism for describing such a dynamic system as VHDL was able to be used.

Because of the precision provided by using the operational framework, the characterisation presented could be used as part of the specification for the design of simulators. In a full semantics for the language, it would not merely constitute a part of such a specification, but encompass it entirely. Since the semantics has been written to take into account the implied concurrency of VHDL processes, it need not simply specify sequential simulators, but also high-performance, parallel ones.

Since a uniform approach was taken in characterising what is meant by an individual simulation cycle, there has been no need to treat 0-delay signal assignments any differently from unit-delay ones. As was seen in the examples of Chapter 6, the simulation kernel itself provides the abstraction mechanism for resolving multiple 0-delay steps into an overall view of "instantaneous" assignments. The key to understanding the way VHDL works with time was to do away with it entirely. In its stead, the concept of a point of computation was used.

One important property about the semantics has also been proved. Theorem 4.4.1 reinforces one of the most basic tenets of any hardware description language.