The Formal Verification of the Fairisle Switch: The 4x4 Switching Fabric

Paul Curzon

8th March 1994

1 Introduction

We describe work in progress to formally verify the Fairisle switch. To date we have formally verified the Fairisle 4x4 switching fabric using the HOL90 theorem prover [2, 1]. We formalised both the implementation and its behaviour. We then used formal logic to rigorously prove that the behaviour suggested by the description of the implementation satisfies the specified behaviour. In contrast to validation using inexhaustive testing, the results hold for all valid sets of inputs, not just for some small subset. Formal verification corresponds in this sense to exhaustive testing. However, the latter is infeasible for all but very small designs due to the number of cases to consider. Formal verification is feasible because of the use of mathematics (such as induction) to consider the results of many cases at once. It should be noted that formal verification deals with *formal descriptions* of real world objects and ideas. Theorems obtained are only as good as the descriptions used. The results are only as valid as the formal descriptions.

The time taken was comparable to the time originally spent designing, implementing and informally testing the element. No errors were found in the fabricated implementation. This was unsurprising as the element had been in service for some time prior to the formal verification work commencing. Undocumented "features" were found however and many errors were discovered in the initial versions of the formal specifications. This was also unsurprising since documentation of the design and its implementation was sparse.

2 The Implementation

The switching element was designed using a Hardware Description Language: Qudos HDL. This is a simple HDL that allows the structure of hardware to be specified. It does not allow behaviour to be specified directly. To formally verify the switching element, we needed a structural description of the implementation in a language with a formally defined semantics. Without this no formal reasoning about the behaviour of the circuit is possible. Unfortunately, Qudos HDL does not have a formal semantics. We therefore manually translated the original descriptions to HOL-HDL. This is a subset of the HOL logic with the flavour of a hardware description language. It allows descriptions to be given which are very similar to descriptions in Qudos HDL. The translation could have been done completely mechanically, involving only the changing of syntax. However, we made some superficial changes to the description. They did not alter the design, only the description of it. Both descriptions were intended to describe the same collection of logic gates. Two kinds of changes were made: adding extra layers to the hierarchy and simplifying the description using features of HOL which were not available in Qudos HDL. For example, in HOL we were able to represent the data inputs and outputs as words containing 4 bytes, rather than as unstructured 32-bit words as was necessary in Qudos HDL.

3 The Behavioural Specification

To formally verify a hardware design, a formal behavioural specification of the design and of the 43 modules in it is needed. Ideally the formal specification would be written first and the device implemented from it. The switching element had been designed and implemented without any form of formal behavioural specification being created. There was also only sketchy informal documentation. Therefore the formal specification was reverse engineered from the HDL description. This was a very time-consuming process taking between one and two man-months. The early versions of the specification contained many errors. To a large extent the formal verification process was used to aid the reverse engineering. As the proof progressed the specification was gradually corrected as the implementation was analysed and a better understanding of the design was obtained.

The formal behavioural specification of the switching element effectively describes a timing diagram for its inputs and outputs. It is given by a predicate over the input, output and state sequences. The predicate is true for sequences which are possible behaviours of the element. The functional behaviour of the element at a given time is dependent on the arbitration process. We specified this process as a function on the cell headers. Given a set of headers, the function describes the arbitration decision that will be made.

4 The Formal Verification

Each module in the design was formally verified against its behavioural specification separately. The separate proofs were then combined to give a result about the whole element. If the modules are used in other designs their proofs can be reused. Thus the proofs can track design changes relatively quickly. We verified all modules down to the gate level.

Approximately two man-months were spent performing the verification. The breakdown of the time is given in Figure 1. It shows the cumulative time taken as each module was verified. The time (one week) spent proving general word theorems is not included. On the whole, the simpler modules were verified first. The verifier had not previously performed a hardware verification though was a competent HOL user.

Much of the effort was expended in understanding informally why the implementation was correct. This was hampered to some extent by the delay between writing the specifications for a module and doing the proof. Errors in the specifications slowed progress. When the presence of an error was discovered, it had to be located. The manner in which the proofs failed usually gave a strong indication as to the location of the error. However, it was not always immediately clear whether the error was in the behavioural or structural specification of the module, or because a component module's specification was too weak. Determining which was so involved examining the specifications of other modules as well as the faulty one. The specification then needed to be corrected and the proof completed. The main errors were in the modules where there are plateaus in the graph. No errors were discovered in the actual implementation of the switching element.

Some errors were introduced in the structural specification in the translation from Qudos HDL to HOL-HDL. The most serious such error was that the connections of two wires were swapped. Errors also occured in the behavioural specifications because the specifier was not

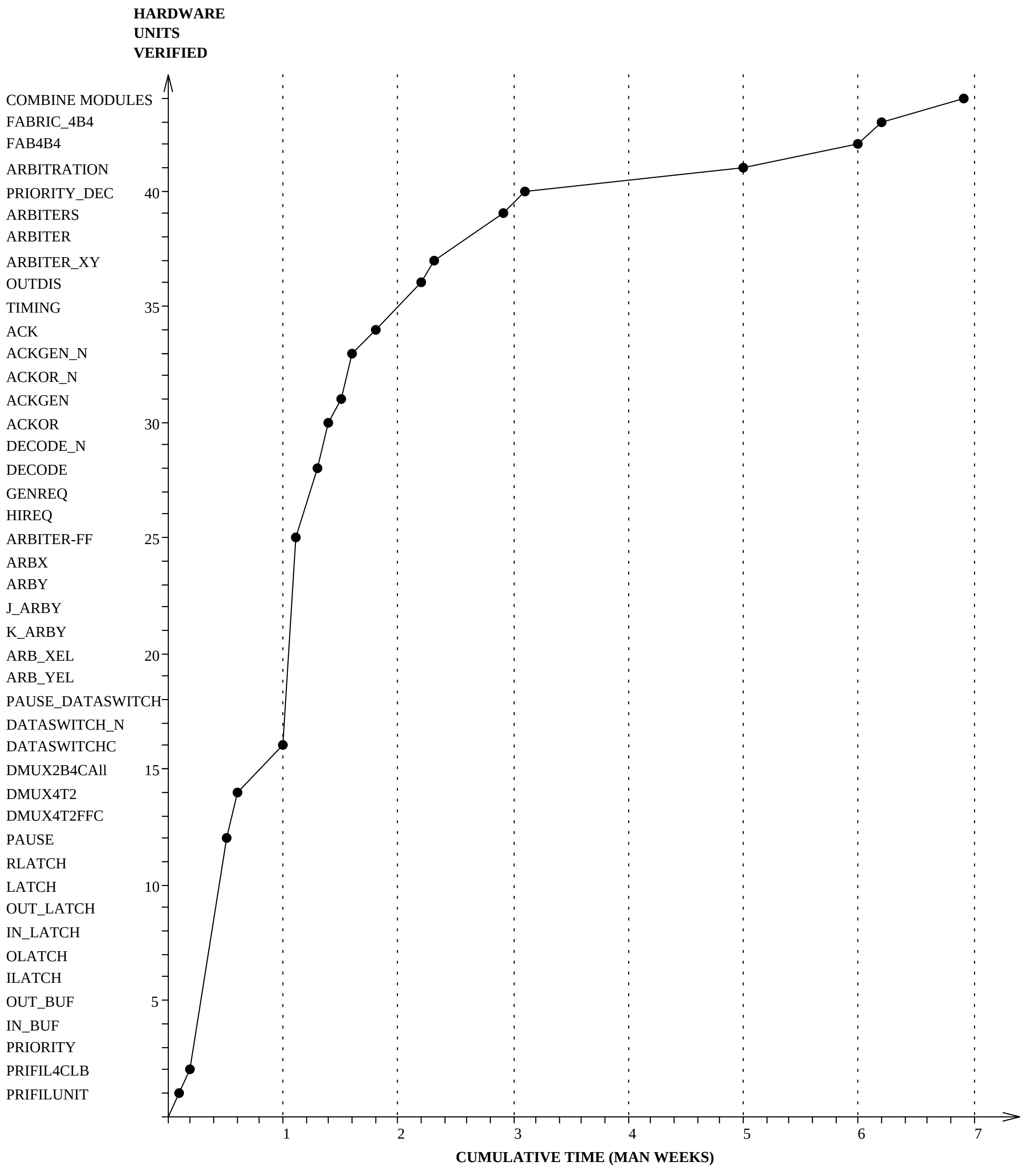

Figure 1: Time taken to verify the switching element

originally familiar with the designs being specified and because very little informal documentation was available. The specifications would probably have contained fewer errors if written by the designers during the design process, or if there had been informal documentation for each module, rather than just for the top levels of the design. All errors found during the course of the verification were corrected and the proofs completed successfully.

No detailed record of the original time spent designing and testing the element was recorded. The design evolved from earlier designs, and several different designs were produced at the same time, making it difficult to accurately estimate the time scale involved. However, it has been estimated that had it been designed from scratch the initial design time would have been in the order of several months. The time spent testing would have been in the order of several weeks. However, errors were discovered after the testing process had been completed when the fabric was installed and in use. These errors would have been uncovered by formal verification had it been conducted before the element was installed rather than afterwards. Thus the time spent to formally specify and verify the design was not unreasonable. Furthermore, it is likely that it would have been much quicker if done as the element was designed since much of the time was spent attempting to understand the design. Had formal verification been applied to the ancestors of the design, the formal verification could have tracked the changes made, with a minimal amount of time spent adapting the proof for the new generation.

The final correctness statement contains an assumption about the environment in which the element operates. It states that the cell header does not arrive at an inopportune moment relative to the frame start signal. The fabric has no control over when these signals arrive as they are determined by the external environment. The design of the port controllers must ensure that the assumption is upheld. The condition was known about by the designers, though did not appear in the informal documentation. It was not originally known of by the verifier but was discovered during the verification.

The proof has been adapted to one of the other versions of the element that was designed at the same time. Due to the modular nature of the design and proof, only the modules which were affected needed to be reverified. It took months to complete the original verification. It took only a few days to modify the specifications and redo the verification for the new design.

References

[1] Paul Curzon. The formal verification of the Fairisle ATM switching element. Technical Report 329, University of Cambridge Computer Laboratory, 1994.

[2] Paul Curzon. The formal verification of the Fairisle ATM switching element: an overview. Technical Report 328, University of Cambridge Computer Laboratory, 1994.