A Case Study on Model Checking and Refinement of Abstract State Machines

Yassine Mokhtari, Meral Shirazipour and Sofiène Tahar
Electrical and Computer Engineering Department
Concordia University, Montréal, Canada
Email: {mokhtari,m_shiraz,tahar}@ece.concordia.ca

1 Introduction

Abstract State Machines (ASMs) [6] are used for specifying wide ranges of applications in a high abstraction level. ASMs bring together specification methods and computational models. Thus, they are a successful methodology for validating and verifying any system. On the other hand, model checking [4] has been proven to be a valuable approach for *automatic* formal verification. In spite of the successful use of this technique, model checkers have their limitations, in the sense that they cannot verify large designs due to the *state explosion* problem [4].

A possible solution to overcome this limitation could be the reduction of the system to an abstract model with a reduced state space. To start, an abstract ground model can be obtained using ASMs. This will give an overview of the problem. The key point will be the refinement of this abstract model in order to reach a version close to the actual system. The refined model must operate in a state space allowed by model checkers and in which requirements, e.g., safety properties can be expressed and verified.

This work can be viewed as an investigation on applying model checking and refinement mapping on ASM models, in the context of hardware design. We analyze the applicability of automatic verification of ASM models using the ASM-Workbench and its interface ASM-SMV [3]. The used example is a module of an ATM switch fabric, the Timing block [7], on which we show how the validation and verification are performed. A specification and an implementation of the Timing block are first specified in the *ASM-Workbench* language [2]. These models are validated separately by simulation using the Workbench. Then each ASM model is automatically transformed to the SMV language using the ASM-SMV interface. The Model checker SMV [8] is used to verify properties on both specification and implementation. Then refinement is performed to prove that the implementation corresponds with the specification.

2 Fairisle ATM Switch Fabric

The studied case is the Fairisle ATM switch fabric, a real ATM switch designed and in use at the University of Cambridge for multimedia applications [7]. This ATM switch is composed of *input port controllers, output port controllers* and a *switch fabric*. The fabric's functionality is cyclic and consists of waiting for cells to arrive, read and process them by switching data cells from the input ports to the output ports, and send acknowledgments. The input port controllers receive acknowledgment signals from the fabric and either send new data, retransmit the previous cell, or stop sending data. The port controllers and switch fabric use the same clock. They also use a higher-level cell frame clock: the *frameStart* signal. This ensures that cells arrive in a synchronous manner. The cells from all the input ports start when the *active bit* of any input port goes high. If no bit from the input ports is high, then the frame is inactive.

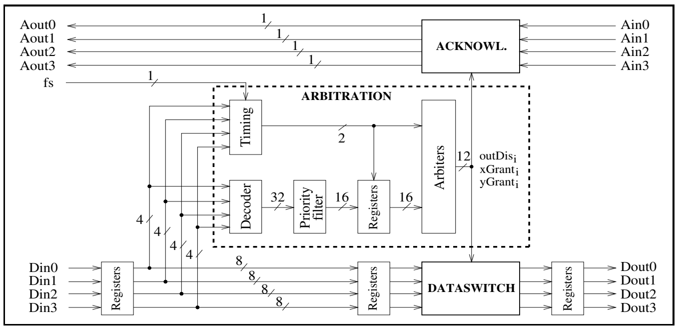

Figure 1: Fairisle ATM Switch Fabric

The ATM switch's implemention consists of an arbitration unit, an acknowledgment unit and a data switch unit (see Figure 1). The arbitration unit consists of a Timing module, a Decoder, a Priority Filter and a set of Arbiters.

The Timing block, our case study, controls the timing of the arbitration decision. Figure 2 illustrates the abstract behavior model of the Timing block and its implementation. For more details about the implementation refer to [7].

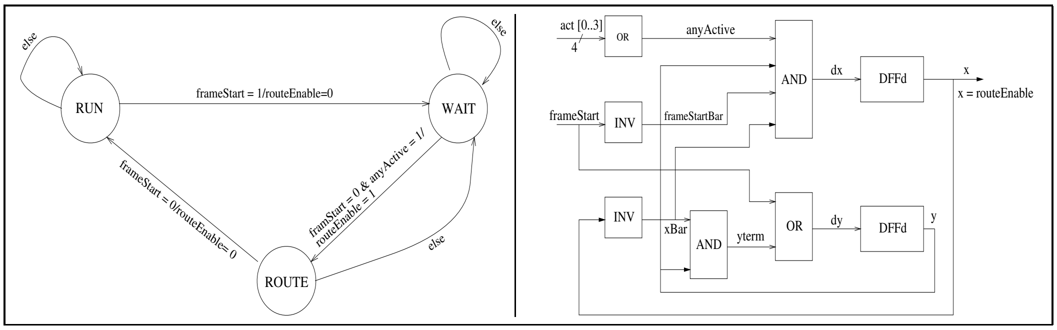

Figure 2: Timing Block: Specification and Implementation

3 ASM Modelling and Validation

The ASM modelling is done as follows: the state of the machine is represented by *nullary dynamic functions*. The behavior of the machine is modelled by transition rules. During each cycle of execution, the state machine evaluates the incoming signals, that is the inputs provided by the environment by means of *external functions*. It then computes the new state and the outgoing signals.

Validation is carried out by tests, ensuring that desirable scenarios were allowed by our model, and undesirable ones forbidden. The key success of validation is the executability of our model, since ASMs describe systems by transitions. This was performed by simulating the ASM models, specification and implementation (Figure 2) of the Timing block, using the *ASM Workbench*.

4 Model Checking and Refinement Mapping

Once an ASM model is transformed into the model checker language, we can perform the model checking of safety and liveness properties. For example, in the Timing block, we are interested in the verification of a liveness property stating that arrived cells are eventually routed. We formalize this requirement in CTL:

$$\mathbf{AG}(\neg frameStart \wedge anyActive \rightarrow \mathbf{AF}\ routeEnable)$$

Proving that an ASM model satisfies its requirements does not depend only on states and transition rules but also on some additional assumptions. We provide examples of that during the Timing block's verification. For instance, the verification of the abstract model, i.e., comparing the abstract model against the CTL property stated above, needs a fairness requirement [8]. Fairness conditions are used to constrain the nondeterministic choices present in abstract models.

Moreover, the verification of the Timing block implementation requires an additional assumption about the *environment*. Indeed, we have abstracted the environment by means of external functions in the ASM model. However, and in the next lower level ASM, these functions must be assumed to satisfy some logical properties. These assumptions must be *reasonable* in order to guarantee the correctness of the verification. Finally, we use assume-guarantee paradigm [5] to prove the correctness of our implementation by splitting the proof in two parts. First, we must show that the implementation satisfies our CTL requirement, assuming that the environment behaves like our CTL environment assumption. Then, this assumption is discharged by proving that the actual environment indeed meets the CTL assumption. Using SMV, the first step is proved while we hold the assumption about the environment until we tackle the actual implementation of the environment.

The last step of our verification consists of proving that our abstract ASM model is correctly implemented by our ASM implementation. This is usually done by introducing *refinement maps* [1] that translate the behavior of the abstract model into the behavior of given interfaces and structures in the low design. SMV supports this methodology by including a construct called *layer*. Using this feature, we succeeded the verification of the translated ASM models of the timing block specification and implementation.

5 Conclusions

In this work, we investigated model checking and refinement of a piece of hardware modeled with ASMs. Validation and translation of the ASM models into SMV were done using existing tools. Then, model checking and refinement techniques available in SMV were used to prove that the implementation corresponds to the specification.The specification and implementation of our case study were validated and verified separately.

From our case study, we suggest that the ASM language should be extended with features for expressing fairness assumptions, assume-guarantee paradigm and refinement. Further, it should be equipped with a powerful formal specification language like CTL.

References

[1] M. Abadi and L. Lamport. The existence of refinement mappings. *Theoretical Computer Science*, 81(2):253–284, 1991.

[2] G. Del Castillo. ASM-SL, a Specification Language based on Gurevich's Abstract State Machines: Introduction and Language Definition. *Heinz Nixdorf Institut*, Paderborn, Germany 1999.

[3] G. Del Castillo and K. Winter. Model checking support for the ASM high-level language. *International Conference for Tools and Algorithms for the Construction and Analysis of Systems, TACAS'2000*, April 2000.

[4] E. M. Clarke, Orna Grumberg, and Doron Peled. *Model Checking*. MIT Press, January 2000.

[5] O. Grumberg and D. E. Long. Model Checking and Modular Verification. *ACM Transactions on Programming Languages and Systems*, 16(3):843–871, May 1994.

[6] Y. Gurevich. The Sequential ASM Thesis. *Bulletin of the EATCS*, 67:93–124, 1999.

[7] I.M. Leslie and D.R. McAuley. Fairisle: An ATM Network for the Local Area. *ACM Communication Review*, 19(4):327–336, 1991.

[8] K. L. McMillan. *Symbolic Model Checking*. Kluwer Academic Publishers, 1993.