Practical Application of Model Checking – a Taxonomy of Methodologies.

Authors

- Laurent Arditi, Formal Verification Engineer, ARM, France (Laurent.Arditi@arm.com, +33497235100)
- Daryl Stewart, Formal Verification Engineer, ARM, UK (Daryl.Stewart@arm.com, +441223400400)
- Bryan Dickman, Verification Manager, ARM, UK (Bryan.Dickman@arm.com, +441223400400)
- Mike Bartley, Founder, Test and Verification Solutions, UK (mike@testandverification.com, +447796307958)
- Anthony McIsaac, Formal Verification Consultant, Test and Verification Solutions, UK (anthony@tandvsolns.co.uk, +441789299584)
- Lawrence Loh, Vice President of Worldwide Applications Engineering, Jasper Design Automation, USA (lawrence@jasper-da.com, +1 (650) 966-0224)

Introduction

Formal verification applies mathematical concepts and algorithms to the verification process. For example, in combinational equivalence checking we can take two different designs, automatically create a formal mathematical model of each and then automatically prove them equivalent. Thus, for example, one can automatically prove steps in the design process (RTL synthesis, DFT insertion) to be 100% correct. This process can be automated in more circumstances and has therefore lead to a widespread adoption of the technique.

In this paper we consider the formal verification technique known as model checking. Using this technique the requirements, or properties, of a design are expressed in a formal mathematical language and then tools are used to analyze whether there is any way that a given model of the design can fail to satisfy the requirements. The ultimate goal being to prove that the design satisfies the requirements. This technique has not had widespread adoption across the industry for a variety of reasons:

- The tools available to apply the technique have had limited capacity in terms of the size and complexity of the designs they can analyse and the requirements they can prove.
- Identifying suitable requirements and then formalising them into a mathematical language has often been viewed as a specialism out of the reach of most design and verification engineers.

This paper will look at the practical application of model checking within the industry but is particularly focussed on its use at ARM. ARM Holdings is the world's leading supplier of semiconductor intellectual property (IP). The ARM IP portfolio includes processors, graphics processors, video hardware, debug IP, on-chip interconnects, memory controllers and a variety of other system IP. These IP provide a wide variety of complex verification challenges which are compounded by the IP model as ARM must verify their designs as IP components, independently from the target chip or system.

The paper is concerned with the application of formal verification by architects, designers and verification engineers to four main activities:

- ***Bug Avoidance***: In this application of model checking the objective is to avoid adding bugs to designs in the first place. For example, by allowing the designer to perform a formal analysis of possible behaviours of the design during early design specification and capture stages. . Alternatively, the process of trying to rigorously formalise design requirements can often lead to clarifications and avoidance of bugs very early before coding has started.
- ***Bug Hunting***: The objective here is to find bugs in an implemented design rather than try to prove the design satisfies the given requirements. The technique tries to prove simple properties often automatically generated or embedded in the RTL by the designer. Failures are investigated but limited effort is expended to try to exhaustively prove all properties.
- ***Bug Absence***: This is the traditional application of model checking. Specific properties are specified, implemented and checked representing important requirements of a design. This can be done at an early stage on architectural models, or later on RTL implementations. Effort is put into proving the properties as the goal is to prove that the model satisfies the property thus demonstrating the absence of bugs.
- ***Bug Analysis***: Bugs found during the late stages of development or implementation may be hard to reproduce with full analysis in simulation. By writing the symptom of the bug as a property, a formal tool can generate a waveform of minimum length to show the failure. Formal bug analysis also helps to clearly understand the constraints needed for the bug to happen. After a fix is proposed, formal verification can then help prove or disprove that it effectively solves the issue. This technique is required in the case where complete specification in Bug Avoidance were not possible or 100% proofs were not possible and so bugs were not caught earlier. Also it is useful for older designs where formal was not applied but in practice this is not always possible.

The paper will not only describe how the above techniques can be applied by architects, designers and verification engineers but also at what stage of the verification process they are applied in order to maximise their effectiveness. The paper will explain how the methods used in ARM overcome the perceived barriers of limited tool capacity and need for special expertise.

The paper contains supporting data on Bug Hunting from a pilot project to show how ARM has applied these techniques and the results they have achieved.

- The pilot project was a processor design which had 1500 embedded properties written by designers. Bug hunting was applied through a fully automated process. The paper contains data on the design size, the time taken to run the properties through the flow, the number of properties that were fully proven and the numbers of bugs found.
- The reporting of formal results played an important part in the success of the pilot project: bugs were entered in bug tracking tool, all results were logged into a database, regular status were generated along with plots to look at the trends, etc. This was essential to make formal visible, correctly understood and valuable for all the teams and managers.

Data from the application at ARM of the other techniques will also be included, as well as data from the application by other Jasper customers.

In summary, this paper will achieve the following.

- Demonstrate a number of practical applications of model checking besides the traditional approach of demonstrating the absence of bugs. It will discuss who is best able to apply those techniques and when in the design process.
  These applications will be presented along with data generated from application on numerous real projects.
- In the experience of the authors, many companies fail to adopt or realise the full benefits of model checking as they fail to find practical applications that can be performed by existing employees without disrupting existing flows. This paper will explain, through real examples, how that can be achieved.

The above will be explained in a generic way that can be achieved using most commercial model checking tools. However, many of the examples will come from flows using Jasper Gold and there will be some examples that are specific to that tool.