An Abstraction of Multi-Port Memories with Arbitrary Addressable Units

Tomáš Vojnar, Aleš Smrčka, and Lukáš Charvát

Brno University of Technology, FIT, IT4Innovations Centre of Excellence

The paper describes a technique for automatic generation of abstract models of memories that can be used for efficient formal verification of hardware designs. Our approach is able to handle addressing of different sizes of data, such as quad words, double words, words, or bytes, at the same time. The technique is also applicable for memories with multiple read and write ports, memories with read and write operations with zero- or single-clock delay, and it allows the memory to start with a random initial state allowing one to formally verify the given design for all initial contents of the memory. Our abstraction allows large register-files and memories to be represented in a way that dramatically reduces the state space to be explored during formal verification of microprocessor designs as we witnessed in experiments with the approach proposed in [1].

Related Work. Numerous works have focused on memory abstraction, notably within the area of formal verification. Some of the proposed abstractions are tightly coupled with the verification procedure used: for instance, many of them rely on that SAT-based bounded model checking is used [3, 4]. An approach not tailored for a specific verification approach has been presented in [2] which introduces a theory for reasoning about safety properties of systems with arrays. In the work, an automatic algorithm for constructing abstractions of memories is presented. The algorithm computes the smallest sound and complete abstraction of the given memory. This approach does, however, not support addressing of different sizes of data. A recently published work [5] formally specifies and verifies a model of a large memory that supports efficient simulation. The model is tailored for x86 implementations only in order to offer a good trade-off between the speed of simulation and the needed computational resources. The approach assumes starting from the nullified state of the memory, not from a random state.

Our algorithm can generate abstractions of memories that support addressing of arbitrary addressable units with multiple read and write ports, and it allows the memory to start from a random initial state. The algorithm is not bound to any specific verification technique. The generated abstraction can be described in any language for which the user can provide templates specifying (1) how to express declarations of state and signal variables, (2) how to encode propositional logic expressions over state and signal variables, (3) and how to define initial and next states of state variables.

The Main Idea of the Proposed Memory Abstraction. When formally verifying a system which uses memories, a need to deal with large capacity memories causes a state explosion. The technique of memory abstraction proposed in the paper exploits the fact that formal verification often suffices with exploring a limited number of accesses to the available memory, and it is thus possible to reduce

the number of values that are to be recorded as actually stored in the memory (abstracting away the random contents stored at unused memory locations). Our abstraction preserves the interface of the abstracted memory, but the abstract memory effectively remembers only the memory cells which have been accessed. Internally, the memory is implemented as a table consisting of some number of couples of variables storing corresponding pairs of addresses and values[1]. The memory also remembers which of the pairs are in use. If the memory is accessed for reading, the address-value pairs that are in use are searched. If a location is read that has already been accessed in the past, then the value associated with the appropriate address is simply returned. If a location that has never been accessed is read, a corresponding pair is not found in the table, and a new couple is allocated. Its address part will store the particular address that is accessed while the value stays unconstrained. However, the variable representing the value associated with the accessed location stays constant in the future (unless there occurs a write operation to the concerned address). This ensures that subsequent reads from the location return the same value. In case of writing, the address and value are both known. In case of writing to a location that has not been accessed yet, a new address-value pair is allocated to store the given address-value pair. Otherwise, a value associated with the given address is updated.

In order to support dealing with different sizes of addressable data (including reading/writing data smaller than the contents of a single memory cell of the modeled memory), we split our abstract memory into a low level memory and a set of functions mapping accesses to ports of the modeled memory to ports of the low level memory. The low level memory consists of cells whose size equals the size of the least addressable unit of the modeled memory. In order to allow for reading/writing the allowed sizes of data in one step, the number of read and write ports of the low level memory is appropriately increased.

In order to prove usefulness of the proposed model, we used it in a manual, but strictly algorithmic way to derive memory models for the approach of checking correspondence between the ISA and RTL level descriptions of microprocessors proposed in [1]. Our experiments showed significant improvements in the verification time.

References

1. L. Charvát, A. Smrčka, and T. Vojnar. Automatic Formal Correspondence Checking of ISA and RTL Microprocessor Description. To appear in *Proc. of MTV'12*.
2. S. M. German. A Theory of Abstraction for Arrays. In *Proc. of FMCAD'11*, FMCAD, 2011.
3. P. Manolios, S. K. Srinivasan, and D. Vroon. Automatic memory reductions for RTL model verification. In *Proc. of the ICCAD'06*, ACM, 2006.
4. M. K. Ganai, A. Gupta, and P. Ashar. Verification of Embedded Memory Systems using Efficient Memory Modeling. In *Proc. of DATE'05*, IEEE, 2005.
5. W. A. Hunt, Jr. and M. Kaufmann. A Formal Model of a Large Memory that Supports Efficient Execution. In *Proc. of FMCAD'12*, FMCAD, 2012.

[1] When using bounded model checking (BMC) as the verification technique, the needed number of address-value pairs can be easily determined from the depth of BMC. For unbounded verification, this number can be iteratively incremented until it is sufficient. The incrementation is finite since the number of memory cells is finite.