BooSTER: Speeding Up RTL Property Checking of Digital Designs by Word-Level Abstraction

Peer Johannsen

Siemens AG, Corporate Technology, Design Automation, CT–SE–4
81730 Munich, Germany
peer.johannsen@mchp.siemens.de

Abstract. In this paper we present a tool which operates as a pre- and postprocessor for RTL property checking and simplifies word-level specifications before verification, thus speeding up property checking runtimes and allowing larger design sizes to be verified. The basic idea is to scale down design sizes by exploiting word-level information. BooSTER implements a new technique which computes a one-to-one RTL abstraction of a digital design in which the widths of word-level signals are reduced with respect to a property, i.e. the property holds for the abstract RTL if and only if it holds for the original RTL. The property checking task is completely carried out on the scaled-down version of the design. If the property fails then the tool computes counterexamples for the original RTL from counterexamples found on the reduced model.

1 High-Level Property Checking of Digital Designs

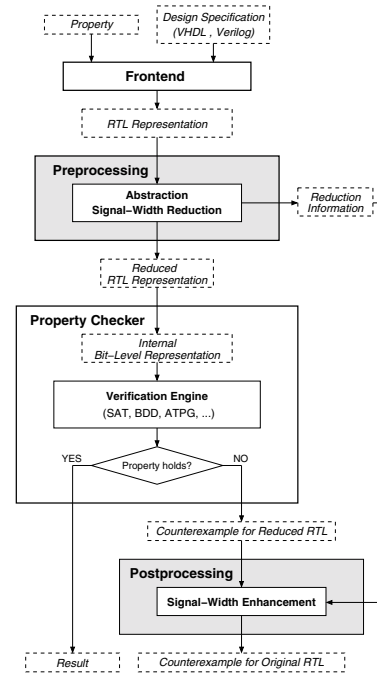
Today's digital circuit designs frequently contain up to several million transistors and designs need to be checked to ensure that manufactured chips operate correctly. Formal methods for verification are becoming increasingly attractive since they confirm design behavior without exhaustively simulating a design. Over the past years, bounded model checking and property checking have increased in significance in electronic design automation [1,9]. Promising approaches to enhance capabilities of hardware verification tools are decision procedures which make use of high-level design information [2,3,4,5,11], and automated abstraction techniques, e.g. using uninterpreted functions and small domain instantiations [6,10].

We consider a property checking flow in which design specifications are given as VHDL or Verilog source code. Properties are specified in a linear time logic used in Symbolic Trajectory Evaluation and describe the intended behavior of the design within a finite bounded interval of time. As a first step, design and property are synthesized into a flattened RTL netlist, including word-level signals, word-level gates, arithmetic units, comparators, multiplexors and memory elements. Each word-level signal $\boldsymbol{x}$ has a fixed width $n \in \mathbb{N}_+$ and takes bitvectors of respective length as values. A property checker, which reads RTL netlists

G. Berry, H. Comon, and A. Finkel (Eds.): CAV 2001, LNCS 2102, pp. 373–377, 2001.

as input, translates such representation of design and property into an internal bit-level representation (i.e. an instance of propositional SAT) and uses SAT, BDD and ATPG methods to either prove that the property holds for the given design or to compute a counterexample. A counterexample is an indication that a circuit does not function in the way intended by the designer and is given in terms of assignments of values to the circuit inputs, such that a violation of the desired behavior, which is described by the property, can be observed.

BooSTER (Boolean String Length Reduction) implements a new word-level abstraction technique developed in [7], which is embedded within the flow. In a preprocessing step prior to the property checker (see fig.), the tool takes the RTL netlist and computes a scaled down RTL model of the design in which each word-level signal $\boldsymbol{x}$ is replaced by a corresponding shrunken signal of width $m_{\boldsymbol{x}} \leq n$, where n is the original width of $\boldsymbol{x}$, while guaranteeing that the property holds for the reduced RTL if and only if it holds for the original RTL. Design and abstract model differ from each other only as far as signal widths are concerned. The reduced RTL is given to the property checker instead of the original RTL. Depending on the degree of reduction, the internal bit-level representation computed from the reduced RTL contains significantly less variables than the one computed when using the non-reduced RTL. If the property does not hold, the counterexample returned by the property checker is taken, which is a counterexample relating to the signals of the reduced RTL. A corresponding counterexample for the original RTL is computed, using information about the applied reduction, gathered during the preprocess.

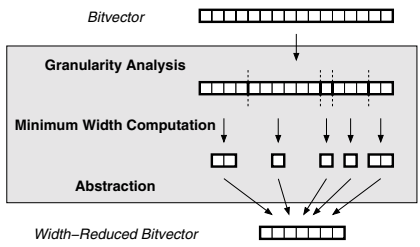


2 Scaling Down RTL Designs by Signal Width Reduction

BooSTER reads an RTL representation of a design and a property and generates a system E of equations over a theory of fixed-size bitvectors based on [7], which is an extension of the core theory of bitvectors presented in [5]. Our theory features high-level operators like bitwise Boolean operations, arithmetics (cf. [2]) and if-then-else, and allows complete RTL designs to be modeled. E is satisfiable if and only if the property does not hold for the RTL. Word-level signals in the RTL correspond to bitvector variables in E, thus the information, which bits belong to the same signal, is kept. A satisfying solution of E yields a counterexample for the RTL. For each bitvector variable occurring in E the smallest possible number

of bits is computed, such that a second system E' of bitvector equations, which differs from E solely in the manner that variable widths are shrunken to these computed numbers, is satisfiable if and only if E is satisfiable. E' is generated using these minimum signal widths and then retranslated into a netlist, which is output by the tool and represents a scaled down version of the original RTL.

The process of scaling down signal widths is separated into two subsequent phases. The high-level operators occurring in the equations of E impose structural and functional dependencies on the bitvector variables. Thereby, variables typically have non-uniform data dependencies, i.e. different dependencies exist for different chunks of a signal. Our method analyzes such dependencies and, for each variable, determines contiguous parts in which all bits are treated uniformly in the exact same manner with respect to data dependencies. Such decomposition of a variable into a sequence of chunks is called a granularity. For each such chunk of a signal, the necessary minimum width is computed, as required by dynamical data dependencies. According to these computed minimum chunk widths, the reduced width for the corresponding shrunken signal is reassembled (see [7,8] for further details on the reduction).

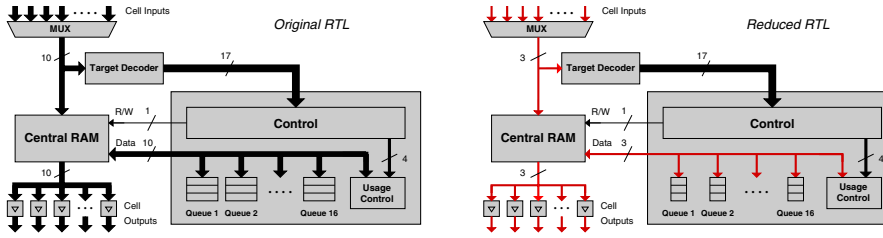


3 Experimental Results

BooStER is implemented in C++ and was tested in several case studies at the EDA department of Siemens Corp. in Munich and at Infineon Techn. in San Jose. Test cases were run on a PII 450 Mhz Linux PC with 128 MB. The tool operated as a preprocessor to the property checker used at Siemens and Infineon. All runtimes on reduced models were compared to those achieved on the original designs without preprocessing. As an example, we here consider the management unit of an ATM switching element. The design consists of 3.000 lines of Verilog code, the netlist synthesis has approx. 24.000 gates and 35.000 RAM cells. The RTL incorporates 16 FIFO queue buffers and complex control logic. Data packages are fed on 33 input channels to the management unit, stored in the FIFOs and upon request are output on one of 17 output channels, while the cell sequence has to be preserved and no package must be dropped from the management unit.

	Property	Original design	Shrunken model
Computation times for pre- and postprocessing	`nop`		2.96 secs
	`read`		6.53 secs
	`write`		3.24 secs
FIFO sizes on RTL	`nop`	160 cells × 10 bit	160 cells × 2 bit
	`read / write`	160 cells × 10 bit	160 cells × 3 bit
Overall number of bits in all signals in cone of influence of property	`nop`	20925	5034 (24.0 %)
	`read`	31452	10592 (33.6 %)
	`write`	14622	5163 (35.3 %)
Overall number of gates in synthesized netlist	`nop`	23801	5661 (27.9 %)
	`read / write`	23801	7929 (33.3 %)
Number of state bits	`nop`	1658	362 (21.8 %)
	`read / write`	1658	524 (31.6 %)
Property checker runtimes	`nop`	23:33 min	37.96 secs (2.7 %)
	`read`	42:23 min	3:27 min (8.1 %)
	`write_fail`	2:08 min	25.66 secs (19.5 %)
	`write_hold`	27:08 min	1:08 min (4.2 %)

Three different properties (`nop`, `read`, `write`) had to be verified, which specified the intended behavior within a range of 4 timesteps (`nop`, `write`) and 6 timesteps (`read`). Results and CPU times are are shown above. As can be seen, in all cases



the data path signals could be scaled. This is illustrated in the block diagrams above, showing the original design and the reduced model for the `read` property. We encountered a significant reduction in the sizes of the design models and a tremendous drop in the runtimes of the property checker. It turned out that the `write` property did not hold due to a design bug in the Verilog code. A counterexample for the reduced model was found (`write_fail`) from which the tool computed a counterexample for the original design, whereupon the bug was fixed and the property was again checked on the corrected design (`write_hold`).

4 Conclusions

Reducing runtimes and the amount of memory needed in computations is one requirement in order to match today's sizes of real world designs in hardware verification. We have presented a tool that efficiently simplifies word-level circuit specifications for RTL property checking by scaling down the widths of input, output and internal signals. A linear reduction from n bits down to m bits, $m < n$, causes an exponential reduction of the induced state space of the signal from 2^n to 2^m, while reduced state space sizes coincide with increased verification performance. Our method provides a one-to-one RTL abstraction,

which interprets all RTL operators and which strictly separates the pre- and postprocessing of design and counterexample, and the property checking process itself. Thus, the proposed method is independent of the concrete realization of the property checker and can be combined with a variety of existing techniques which take RTL netlists as input. Due to providing a one-to-one abstraction, postprocessing of counterexamples is straightforward, false-negatives cannot occur. Moreover, if preprocessing yields that no reduction is possible for a given design and a property, then abstract model and original design are identical, so the verification task itself is not impaired by using the proposed abstraction, and in all case studies pre- and postprocessing runtimes were negligible. Test cases showed that the tool cooperated particularly well with a SAT and BDD based property checking multi-engine, because the complexity of those techniques often depends on the number of bits occurring in a design. Furthermore, experiments revealed that the proposed abstraction seems to be well qualified for hardware verification of memories, FIFOs, queues, stacks, bridges and interface protocols.

References

1. A. Biere, A. Cimatti, E.M. Clarke, M. Fujita, Y. Zhu. Symbolic Model Checking Using SAT Procedures instead of BDDs. *DAC'99*, pages 317–320. 1999.
2. C.W. Barrett, D.L. Dill, J.R. Levitt. A Decision Procedure for Bit-Vector Arithmetic. *DAC'98*, pages 522–527. 1998.
3. N. Bjørner, M.C. Pichora. Deciding Fixed and Non-fixed Size Bit-vectors. *TACAS'98*, pages 376–392. 1998.
4. C.Y. Huang, K.T. Cheng. Assertion checking by combined word-level ATPG and modular arithmetic constraint-solving techniques. *DAC'00*, pages 118–123. 2000.
5. D. Cyrluk, M.O. Möller, H. Ruess. An Efficient Decision Procedure for the Theory of Fixed-Sized Bit-Vectors. *CAV'97*, pages 60–71. 1997.
6. R. Hojati, A.J. Isles, D. Kirkpatrick, R.K. Brayton. Verification Using Uninterpreted Functions and Finite Instantiations. *FMCAD'96*, pages 218–232. 1996.
7. P. Johannsen. Scaling Down Design Sizes in Hardware Verification. *Ph.D. Dissertation at the Christian-Albrechts-University of Kiel*, to appear in 2001.
8. P. Johannsen. Computing One-to-One Minimum-Width Abstractions of Digital Designs for RTL Property Checking. *Intern. Report, Siemens AG, CT-SE-4*, submitted to ICCAD'01.
9. J.P. Marques da Silva, K.A. Sakallah. Boolean satisfiability in electronic design automation. *DAC'00*, pages 675–680. 2000.
10. A. Pnueli, Y. Rodeh, O. Shtrichman, M. Siegel. Deciding Equality Formulas by Small Domains Instantiations. *CAV'99*, pages 455–469. 1999.
11. Z. Zeng, P. Kalla, M. Ciesielski. LPSAT: A Unified Approach to RTL Satisfiability. *DATE'01*. 2001.