Verification of Register Transfer Level Low Power Transformations

C. Karfa, C. Mandal, D. Sarkar
Department of Computer Science and Engineering,
Indian Institute of Technology, Kharagpur 721302, India.
Email: {ckarfa, chitta, ds}@cse.iitkgp.ernet.in

***Abstract*—An automated framework for verification of low power transformations in register transfer level (RTL) designs is presented in this paper. Our verification method consists in two steps. In the first step, the datapath interconnection and the controller finite state machine of both the input RTL and the transformed RTL are analyzed by a rewriting based method to obtain the finite state machine with data paths (FSMDs). In the second step, an FSMD based equivalence checking method is deployed to establish equivalence between the RTLs. Our method is is strong enough to handle most of the RTL low power transformations.**

***Keywords*-Verification; Low power transformations; Register transfer level.**

I. Introduction

The main obstacle of applying low power transformations at RTL designs [1], [2], [3] is the difficulties involved in their verification. In a typical industry scenario, an RTL or architectural low power transformation implies a full cost of dynamic validation, which can extend to many months [4]. Therefore, an automated formal verification method for low power transformations of RTL designs has tremendous practical importance and is the objective of this work.

We present an automated verification method for low power transformations in RTL designs. The inputs to our method are two RTL designs – an input RTL design and the one obtained from the input RTL by applying low power transformations. The input RTL can be either manually generated or synthesized from behavioural specification by high-level synthesis tools. Our verification process consists in two steps. We first construct the finite state machine with datapaths (FSMDs) from both the input and the transformed RTLs using the method given in [5]. In the next step, the equivalence between two FSMDs are established by the method proposed in [6]. The key aspect of our method is that it is independent of the low power transformations. In this paper, we analyze several low power transformations applied on the RTL designs to show that our method can handle most of them.

II. Construction of FSMDs from RTL Designs

Each RTL consists of a datapath and a controller. The controller, represented as an finite state machine (FSM), invokes a control assertion pattern (CAP), in each control step to execute all the required data-transfers and proper operations in the FUs. The data path of the RTLs are captured by a function $f_{mc} : \mathscr{M} \rightarrow \mathscr{A}$, where $\mathscr{M}$ is set of all possible micro-operations in the datapath and $\mathscr{A}$ is the set of all possible control assertion patterns. Let there be n control signals. A control signal assertion pattern needed for any micro-operation is represented as an ordered n-tuple of the form $\langle u_1, u_2, \ldots, u_n \rangle$, where each u_i, $1 \leq i \leq n$, represents the value of the control signal c_i from the domain $\{0, 1, X\}$; $u_i = X$ implies that the control signal c_i is not required (relevant) for a particular micro-operation.

The next task is to obtain the set of micro-operations $\mathscr{M}_A$ ($\subseteq \mathscr{M}$) which are activated by a given control assertion pattern A. The following definition is in order.

Definition 1 (Superposition of assertion patterns:): $\forall i$,

$$\begin{aligned} \pi_i(A_1\ \theta\ A_2) &= \pi_i(A_1),\ \text{if } \pi_i(A_1) = \pi_i(A_2) \\ &= \pi_i(A_1),\ \text{if } \pi_i(A_1) \neq \pi_i(A_2) \wedge \pi_i(A_1) = X \\ &= U(\text{undef}),\ \text{if } \pi_i(A_1) \neq \pi_i(A_2) \wedge \pi_i(A_1) \neq X \end{aligned}$$

We define the set $\mathscr{M}_A$ as $\mathscr{M}_A = \{\mu \mid \mu \in \mathscr{M} \text{and} f_{mc}(\mu)\ \theta\ A = f_{mc}(\mu)\}$.

Construction of FSMD essentially consists in replacing the CAP in each state of the controller FSM with the corresponding RT-operations. A *rewriting based method* is used for this purpose. The micro-operations in which a register occurs in the left hand side (lhs) are found first. Such a micro-operation has the form $r \Leftarrow r\_in$, where r is a register and r_in is its input terminal. Next, the right hand side (rhs) expression "r_in" is rewritten by looking for a micro-operation in $\mathscr{M}_A$ of the form "$r\_in \Leftarrow s$" or "$r\_in \Leftarrow s_1\ \langle op \rangle\ s_2$". So, after rewriting "$r\_in$", we have the rhs expression, either of the form "s" or of the form "$s_1\ \langle op \rangle\ s_2$". In the next step, s (or s_1 *and* s_2 for the latter case) are rewritten *provided they are not registers*. When the expression in hand is of the form "$s_1\ \langle op \rangle\ s_2$" (and s_1, s_2 are not registers), rewriting takes place from left to right in a *depth-first manner*. The process terminates successfully when all s_i's in the expression in hand are registers.

A. Low power RTL transformations

In this subsection, we introduce a set of commonly used low power RTL transformations and then discuss how FSMDs can be constructed by our method when they are applied.

The glitch propagation from control signals through a multiplexer is minimized when its data inputs are highly

This work was supported by Microsoft Corporation and Microsoft Research India under Microsoft Research India Ph.D. Fellowship Award.

978-0-7695-4447-2/11 $26.00 © 2011 IEEE
DOI 10.1109/ISVLSI.2011.73

co-related [1]. This observation can be used to reduce the glitch propagation from control signals feeding a multiplexer network by restructuring it. The datapath structure changes by restructuring the multiplexer architecture. Therefore, the sets of micro-operations are different in the datapaths. However, the controller remains the same and it produces identical CAP for both the datapath to execute certain RT-operations. In this case the rewriting sequence differs for the datapaths. For a given CAP, our rewriting method obtains the same RT-operation(s) in both the datapaths.

The glitches of control signal propagate through the datapath, consequently, consuming a large portion of the glitch power of the entire circuit. So, it is desired to restructure the multiplexer network in such a way that as few of the glitchy select signals as possible are used to reduce the power consumption of the circuit. Since some of the control signals are removed from the select lines of the multiplexer network, the functionality of the control signals change for the multiplexer network. So, the controller may have to generate different CAPs for these two networks to execute the same RT-operation in them. Also, the set of micro-operations of the transformed network is different from the original one. So, both the datapath and controller of the transformed RTL may differ with that of in the input RTL in this case. Our method is able to construct the same RT-operation from different CAPs over different datapaths.

When multiplexer restructuring schemes do not work to reduce the effect of glitches on control signals, clocking control signals can be used to kill glitches on control signals [1]. This modification does not make any changes in the datapath as well as in the controller. Therefore, our method works identically for both the behaviours.

Alternative datapath architectures (even with more resource) of a given datapath may be available with low power consumption. The controller's functionality would be same even though the datapath architecture is changed. Therefore, the controller of this RTL shall generate the same control assertion pattern (CAP) for both the datapaths to perform certain RT-operation(s) in the datapaths. Since the datapath architecture is changed, the micro-operations of the datapath are different. Therefore, the rewriting sequence would be different. Our rewriting method can find the same RT-operations in both the datapath for a given CAP.

The data signals to a circuit block can also be glitchy. The propagation of glitches from data signals can be minimized by inserting delay elements in the select lines of that circuit block. The delay elements are constructed using either a series of buffers or inverters. Adding a delay element or a clock to a control signal does not results in any structural changes in datapath. Also, it does not affect the functionality of the controller. Clearly, the our method works identically for both the behaviours.

A large part of the register power consumption is due to the transitions on the clock inputs to registers. Clock gating is a scheme to avoid this. Clock gating works by taking the enable conditions attached to registers, and uses them to gate the clocks. In the source datapath, the micro-operation relating to writing to registers does not depend on any control signal. However, the same would be dependent on the control signal when gated registers are used. The controller functionality remains same in this case. Therefore, the rewriting method works identically for both the datapaths.

III. Equivalence of FSMDs

We incorporate the equivalence checking method of FSMDs from [6] in our framework. This method decomposes the FSMD by introducing cutpoints in one FSMD, visualizing its computations as concatenation of paths from cutpoints to cutpoints, and identifying equivalent finite path segments in the other FSMD; the process is then repeated with the FSMDs interchanged. We choose this method for following two reasons: (i) We have seen above that the control structure of the FSMD is not modified due to application of low power transformations. In such a case, this FSMD decomposition based method works efficiently. (ii) It may be noted that the effects of some RTL low power transformations at the behavioural level are nothing but some arithmetic transformations. A normalization technique of arithmetic expressions are incorporated in [6] to handle several such arithmetic transformations. This normalization technique is particularly helpful during equivalence checking of FSMDs in our problem. The details of FSMD equivalence checking method are omitted here and can be found in [6].

IV. Experimental Results

The verication method described in this paper has been implemented in C and tested on several benchmark circuits. We take the output Verilog RTL codes of an existing high-level synthesis tool SAST [7] and apply a combination of low power transformations on these RTL designs. The number of registers, FUs, switches, micro-operations, control states in the FSM, control signals varies between 4 to 57, 2 to 4, 13 to 190, 26 to 292, 8 to 120, 19 to 198, respectively, for these benchmarks circuits. Our method successfully established the equivalence in less than three seconds in all the cases.

References

[1] A. Raghunathan, S. Dey, and N. Jha, "Register transfer level power optimization with emphasis on glitch analysis and reduction," *IEEE Transactions on CAD of ICS*, vol. 18, no. 8, pp. 1114 –1131, Aug. 1999.

[2] S. Ahuja, W. Zhang, A. Lakshminarayana, and S. K. Shukla, "A methodology for power aware high-level synthesis of co-processors from software algorithms," in *Proc. of the VLSID '10*, 2010, pp. 282–287.

[3] A. Chandrakasan, S. Sheng, and R. Brodersen, "Low-power cmos digital design," *IEEE Journal of Solid-State Circuits*, vol. 27, no. 4, pp. 473 –484, Apr. 1992.

[4] V. Viswanath, S. Vasudevan, and J. Abraham, "Dedicated rewriting: Automatic verification of low power transformations in rtl," in *22nd International Conference on VLSI Design, 2009*, 2009, pp. 77 –82.

[5] C. Karfa, D. Sarkar, and C. Mandal, "Verification of datapath and controller generation phase in high-level synthesis of digital circuits," *IEEE Transactions on CAD of ICS*, vol. 29, pp. 479–492, 2010.

[6] C. Karfa, D. Sarkar, C. Mandal, and P. Kumar, "An equivalence-checking method for scheduling verification in high-level synthesis," *IEEE Transactions on CAD of ICS*, vol. 27, pp. 556–569, 2008.

[7] C. Karfa, J. Reddy, C. R. Mandal, D. Sarkar, and S. Biswas, "Sast: An interconnection aware high-level synthesis tool," in *Proc. 9th VLSI Design and Test Symposium, Bangalore*, Aug. 2005, pp. 285–292.