Synthesis of Trustable ICs using Untrusted CAD Tools

Miodrag Potkonjak
Computer Science Department
University of California, Los Angeles
Los Angles, CA 90095
miodrag@cs.ucla.edu

ABSTRACT

We have developed the first technique for synthesis of trustable ICs using untrusted CAD tools. The approach enables the use of CAD tools for difficult synthesis tasks and very simple trusted tools developed by the designer for checking results and modifying specifications. The main idea is to specify the pertinent design in such a way that there is no room for the untrusted tool to add any malicious circuitry.

Categories and Subject Descriptors

K.6.5 [**Security and Protection**]

General Terms

Security.

Keywords

Hardware security, Trusted design.

1. MOTIVATION

Although many hardware and system security problems [1], including digital right management using watermarking [2], intellectual property passive [3] and active [4] metering, evaluation and design of hardware security primitives [5], hardware-based cryptography that is intrinsically resilient against side channel and physical attacks [6], and hardware Trojan horse detection [7, 8], have been successfully addressed, probably the most dangerous hardware attacks facilitated through the use of complex and untrusted synthesis software received little attention.

Our goal is to address the difficult version of this problem: how to design trustable integrated circuits (ICs) and systems using untrusted synthesis software and untrusted reusable cores [1]. The importance of this problem is self evident due to the popularity and profitability of the horizontal semiconductor business model, which includes CAD tool providers, design houses, and foundries.

Our objectives include the following desiderata: (i) The approach should be resilient against any arbitrary feature of CAD synthesis software that adds, deletes, or modifies any aspect of the specification of a design; (ii) We treat CAD software as a black box, i.e. synthesis is conducted under the assumption that synthesis software is too complex and strategically important to CAD to allow it to be inspected in any way; (iii) Secure synthesis is fully transparent to the current and pending synthesis tools and flows. Therefore, only small and simple tools that conduct minor design specification modification and checks should be required to ensure trustability; (iv) Secure synthesis must be applicable at all levels of design abstraction; (v) The approach should induce low or no overhead in terms of delay, power, area, and other design metrics.

DAC'10, June 13-18, 2010, Anaheim, California, USA.

2. NEW PARADIGM: COMPLETELY SPECIFIED DESIGN

Our solution to trustable synthesis using untrusted tools is based on the newly created security paradigm of completely specified design and complete and low latency observability. The new security synthesis paradigm has three dimensions: (i) completely specified design at all levels of abstraction; (ii) full accounting of all resources at each step of the design process; and (iii) complete and low latency observability and controllability of the manufactured design to ensure the trustability of the complete design flow.

The first paradigm allows that the design is easily checked at any point of the synthesis flow, since no freedom is left for untrusted CAD tools to embed hardware Trojan horses (HTHs). A completely specified design is one where the pertinent functionality is such that it uses all hardware resources in all clock cycles. Therefore, no resources are available to the attacker to organize an attack. For example, during scheduling all functional units are used in all control steps; during register assignment all registers are occupied by actually consequently used variables in all clock cycles; all transitions in the finite state machines (FSMs) are fully specified so that there are no don't-care states; and the placement and routing is done so that no white spaces are left for additional component placement or routing.

We explain the essence of the new security approach using a scheduling and assignment synthesis task. The first approach for the synthesis of trustable ICs using untrusted CAD tools can be summarized using the following pseudocode:

1. Perform scheduling and assignment using the untrusted synthesis tool;
2. Check that all operations are properly scheduled and assigned using a simple designer developed tool;
3. Augment the schedule and assignment so that all units are used in all cycles using a simple designer developed tool;
4. Add controllability and observability for testing manufactured ICs.

3. MOTIVATIONAL EXAMPLE

We explain the approach using a 2nd order direct form IIR filter shown in Figure 1. The filter has two states, S_1 and S_2, four multiplications by constants denoted by C_1, C_2, C_3, and C_4, and four additions (A_1, A_2, A_3, and A_4). We denote both the operation and its results (variable) using the same symbol. Finally, the input and output are denoted by In and Out, respectively. We assume the synchronous data flow model of computation. The number of each operation indicates the control step in which it is used.

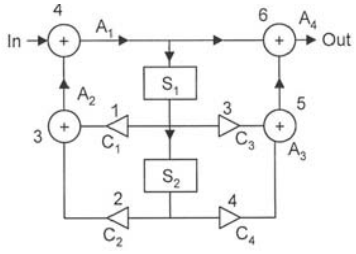

Figure 1. Motivational example, 2nd order IIR filter

Table 1 shows the schedule and assignment under the assumption that the available time is 6 clock cycles and that each operation takes one clock cycle. In general, resource allocation, scheduling, and assignment are difficult optimization problems that are completed using untrusted CAD software. However, checking the correctness of such a schedule is a task of linear complexity for which very simple designer developed software is sufficient. Once we check the correctness of the schedule, we intentionally introduce new operations so that all execution units are used in all control steps by our functionality. Therefore, the attacker does not have room to add any new operations by himself. The additional functionality can be added in several ways. For example, the most energy efficient strategy is to repeat the operations from the previous clock cycle on the same functional unit because in such a way no new switching is introduced. From the security point of view, it is most advantageous to add new operations that can be sent to the output pins and used for checking the correctness of the design as shown in Table 2.

Table 1. Original schedule

A	M
-	$C_1 = C_1 \cdot S_1$
-	$C_2 = C_2 \cdot S_2$
$A_2 = C_1 + C_2$	$C_3 = C_3 \cdot S_1$
$A_1 = In + A_2$	$C_4 = C_4 \cdot S_2$
$A_3 = C_3 + C_4$	-
$Out = A_4 = A_1 + A_3$	-

Table 2. Schedule after we added operation for fully specified design

A	M
$Out' = In + C_5$	$C_1 = C_1 \cdot S_1$
$Out'' = C_1 + C_2$	$C_2 = C_2 \cdot S_2$
$A_2 = C_1 + C_6$	$C_3 = C_3 \cdot S_1$
$A_1 = In + A_2$	$C_4 = C_4 \cdot S_2$
$A_3 = C_3 + C_4$	$C_5 = C_5 \cdot S_4$
$Out = A_4 = A_1 + A_3$	$C_6 = C_6 \cdot S_4$

Once we have the schedule, it is easy to calculate the lifetime of each variable as shown in Table 1. The assignment of variables to registers is again an NP-complete problem for which untrusted CAD tools are used. However, checking the correctness of the variable-to-register assignment can be easily done using designer developed simple software of linear complexity. All that is required is to check that no two variables with overlapping lifetimes are assigned to the same register.

Finally, it is also easy to create fully specified FSMs. For the sake of brevity, we show only the FSM that sets the opcode for the adder. The FSM is built on top of a modulo-8 counter and has four don't-care outputs, denoted by "-". We use standard opcodes for adders.

Table 3. Variables to register assignment. Register R1 is always occupied. Registers R2, R3, R4, and R5 are free in control steps 4-5, 4-6, 3, and 1 & 6. The transfer for state S_1 to state S_2 is done in control step 6. Note that state S_1 is alive across two iterations. (V: Variable, L: Life-time, R: Register)

V	In	A_1	A_2	A_3	A_4	C_1	C_2	C_3	C_4	S_1	S_2
L	1-3	4-5	3	5	6	1-2	2	3-4	4	4-2	1-2
R	R1	R1	R2	R5	R1	R2	R5	R5	R2	R4	R3

Table 4. Column 2 shows initial FSM, Column 3 indicates FSM after the application of the new approach

	NS , $f_2f_1f_0$	NS , $f_2f_1f_0$
CS_1	CS2 , -	CS2 , 001
CS_2	CS3 , -	CS3 , 001
CS_3	CS4 , 001	CS4 , 001
CS_4	CS5 , 001	CS5 , 001
CS_5	CS6 , 001	CS6 , 001
CS_6	CS7 , 001	CS7 , 001
CS_7	CS8 , -	CS8 , 001
CS_8	CS1 , -	CS1 , 001

4. CONCLUSION

We have developed the first technique for the synthesis of trusted ICs using untrusted CAD tools. The approach uses the concept of a fully specified design that prevents CAD tools from adding any new malicious functionality. It leverages the observation that while designing ICs is very difficult, checking them is usually of linear complexity. In a sense, it is similar to the case when we are solving NP-complete problems: it is very difficult to solve them, but it is easy to check their solutions.

5. ACKNOWLEDGMENT

This research is partially supported by the Center for Domain-Specific Computing (CDSC) funded by the NSF Expedition in Computing Award CCF-0926127.

6. REFERENCES

[1] F. Koushanfar, M. Potkonjak, CAD-based security, cryptography, and digital rights management, DAC, pp. 268-269, 2007.

[2] G. Qu, M. Potkonjak, Intellectual property protection in VLSI designs: theory and practice, Springer Kluwer, 2003.

[3] Y. Alkabani, F. Koushanfar, N. Kiyavash, and M. Potkonjak, "Trusted integrated circuits: A nondestructive hidden characteristics extraction approach," Information Hiding, pp. 102-117, 2008.

[4] Y. Alkabani, F. Koushanfar, M. Potkonjak, Remote activation of ICs for piracy prevention and digital right management, ICCAD, pp. 674-677, 2007.

[5] M. Majzoobi, F. Koushanfar, M. Potkonjak, Techniques for Design and Implementation of Secure Reconfigurable PUFs, ACM TRETS, v.2 n.1, p.1-33, March 2009.

[6] N. Beckmann, M. Potkonjak, Hardware-Based Public-Key Cryptography with Public Physically Unclonable Information Hiding, pp. 206-220, 2009.

[7] M. Potkonjak, et al., Hardware Trojan horse detection using gate-level characterization, DAC, pp. 688-693, 2009.

[8] M. Tehranipoor, F. Koushanfar, A Survey of Hardware Trojan Taxonomy and Detection, IEEE Design and Test of Computers, vol. 27, no. 1, pp. 10-25, Jan./Feb. 2010.