Technology-Independent Silicon Camouflage using Virtual Camouflage Front-End Cells

Bryan J. Wang, Lap Wai Chow, James P. Baukus, and Ronald P. Cocchi
Inside Secure
Westminster, CA 92683

***Abstract*— Silicon camouflage, a highly effective obfuscation technique to protect hardware IP from reverse engineering and analysis, is typically implemented in a technology-specific camouflaged cell library and inserted into an IC's gate-level netlist. A new camouflaged IC design flow using virtual camouflage cells enables the detailed specification of silicon camouflage features in a design's technology-independent register transfer level (RTL) models. Using virtual camouflage cells, RTL designers can specify how their hardware IP will utilize camouflaged circuitry while maintaining the models' technology independence. This paper introduces the concept of virtual camouflage cells and describes a camouflaged IC design flow that enables camouflage features to be designed in RTL, including the methods by which the technology-independent camouflaged RTL is synthesized to technology-specific cell libraries and implemented at a given technology node.**

Keywords—silicon camouflage; circuit camouflage; hardware obfuscation; technology independence; obfuscation; reverse engineering; trusted electronics; hardware security; camouflage design flow

I. Introduction

Integrated Circuit (IC) designs are vulnerable to IP theft from reverse engineering, unauthorized cloning and over-production, and device corruption due to Trojan insertion. The risks to the IC industry have been steadily increasing as reverse engineering capabilities increase, and as worldwide IC production capabilities consolidate into a small number of foreign entities. IC designers can protect their circuit designs from these attacks using circuit camouflage. Normally, circuit camouflage is inserted into a circuit at the gate level netlist stage of the design flow, or even later (during or after layout). This IC design flow is illustrated in Figure 1.

A new method to secure an IC using virtual camouflaged circuit components is presented here. This method allows hardware designers to use technology-independent virtual camouflage cells to specify how circuit camouflage will protect their IC. Because the virtual camouflage cells are technology-independent, they can be specified in the IC's high-level design models such as at the Register Transfer Level (RTL). A camouflage-inserted high-level design model can then be synthesized to a target technology, and technology-dependent camouflaged circuit components will be utilized as specified by the designer in the IC's high-level models.

There are several benefits of using virtual camouflage cells to incorporate camouflaged features into the IC's high-level design models. First is to allow designers greater accessibility in selecting useful nodes in the design for camouflage insertion. Because much of a design's nomenclature (module names, hierarchy, signal and register names) is lost during the synthesis process, it is easier to identify specific nodes in the design's RTL code than in the design's synthesized gate-level netlist. Second is to improve reuse of the camouflage insertion parameters across multiple technologies. Since the camouflage insertion parameters are specified by the design's RTL code's instantiation of virtual camouflage cells, reuse across technology libraries requires no additional effort from designers.

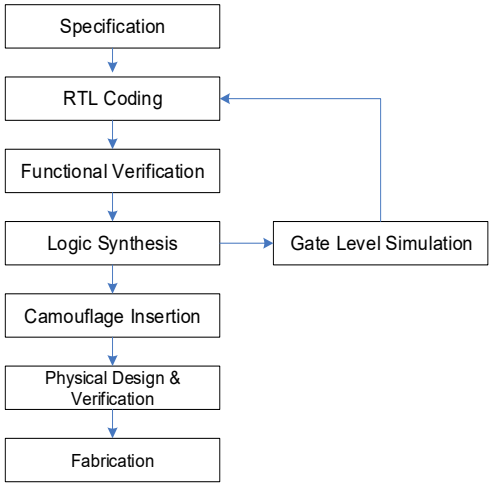

Fig. 1. Typical technology-dependent camouflaged IC design flow

II. Designing with Virtual Camouflage Cells

A. New Camouflaged ASIC Design Flow

A new type of camouflaged ASIC design flow is described here and illustrated in Figure 2. This flow differs from a conventional camouflaged IC design flow because it allows for camouflage features to be designed into an IP block at the RTL level. Insertion of circuit camouflage components can be done once for an ASIC or an IP block, and then reused many times across many technologies. The "Camouflage Insertion" step from Figure 1 is merged with the "RTL Design" step, and a new "Camo Cell Replacement" step is added. The "RTL Coding with Virtual Camouflage Cells" step involves a designer inserting virtual camouflage cells into the design's RTL code. The "Camo Cell Replacement" step is a fully

DISTRIBUTION STATEMENT A.
Approved for public release: distribution is unlimited.

automated step that replaces the virtual camouflage cells with technology-dependent camouflage components.

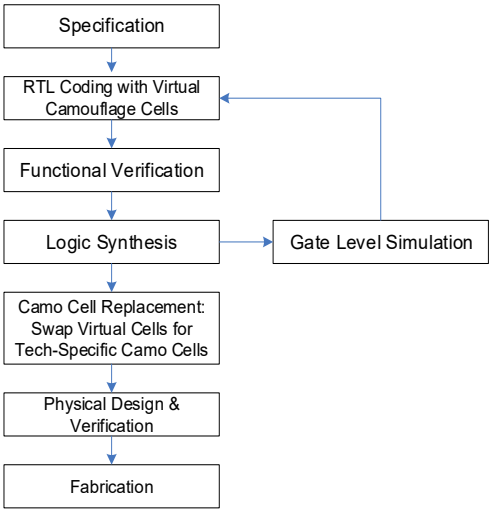

Fig. 2. New camouflaged IC design flow using virtual camouflage cells

The use of virtual camouflage cells in a design's RTL code does not imply a requirement that the design must undergo the camouflage cell replacement process. An RTL design containing virtual camouflage cells may also be synthesized to a non-camouflaged netlist, containing only traditional logic cells. Synthesis to a non-camouflaged netlist is required at technology nodes for which a technology-specific camouflaged cell library is not available. Virtual camouflage is also compatible with and may be used with other available automated camouflage insertion methodologies [1-2].

B. *Virtual Camouflage Library (VCL)*

Virtual camouflage cells are technology-independent logic cells that are organized into a cell library, which shall be called a Virtual Camouflage Library (VCL).The cell library contains all models required to support the Frontend portions of an ASIC design flow (the steps up to and including Logic Synthesis). Designers may instantiate components from the VCL in RTL designs in the same way that they instantiate any other IP block.

Virtual camouflage cells, referred to as VCL Cells, are modeled as logic cells. Driven VCL Cells have outputs which may be stuck-at VDD or VSS, or may be a logical function of cell inputs. VCL cells with stuck-at outputs or outputs that are the logic functions of cell inputs may be used in the design to specify the design's behavior. Undriven VCL Cells have no outputs and one or more camouflage inputs. A camouflage input may be connected to a node (wire) in the circuit to specify that node for camouflage protection. During the Camo Cell Replacement step, these virtual camouflage cells will be replaced with compatible (functionally equivalent) technology-specific camouflaged circuits. VCL cells may have one or more don't-care inputs, which do not affect the output of the VCL cell or the technology-specific camouflage circuit that will eventually replace the VCL cell. Don't-care inputs are optionally used by the designer to specify connections to the camouflage circuit to mislead reverse engineers to believe that the camouflage circuit is a logic function of specific nodes in the circuit.

The instance name of the VCL cell may be used to store camouflage insertion parameters for use during the camouflage cell replacement process. Potentially useful parameters include but are not limited to specification of a set of cells whose nodes must receive the same camouflaged circuit or a functionally-equivalent camouflaged circuit, and the specification of the likelihood that a VCL cell with an unspecified output value will be replaced by a camouflaged circuit with a specified output polarity (i.e. define a 40%/60% chance of VDD/VSS output).

C. *Design and Verification using the VCL*

Camouflaged components can be specified in RTL code by instantiating the desired component from the VCL and connecting the virtual camouflage cell I/O to the desired nodes in the design. The VCL components will not affect the logic function of the design, and RTL verification will not be affected. VCL cells may be conditionally instantiated with preprocessor directives (i.e. *`ifdef*).

D. *Logic Synthesis using the VCL*

Logic synthesis converts an RTL model into a gate-level netlist, which is typically technology-specific. An RTL model that contains VCL cells will undergo the same synthesis process. However, the VCL cells shall be modeled to contain a "don't-touch" attribute, meaning that the VCL cells shall not be subject to logic synthesis, restructuring, or optimization. If the RTL model contains components from a VCL, all logic that is not "don't-touched" shall be synthesized into gate-level components, and any "don't-touched" logic (i.e. VCL components) will be preserved. The result is a technology-specific gate-level netlist that still contains some VCL cells. This netlist shall be referred to as the Pre-Replacement Netlist. The VCL cells will be connected to components of the pre-replacement gate-level netlist in the same way that they were connected to signals of the RTL model.

E. *Camouflage Cell Replacement*

The Camo Cell Replacement step is an automated process that replaces technology-independent VCL components with technology-specific camouflage circuits. Each VCL cell in the Pre-Replacement Netlist is automatically replaced with a functionally-equivalent technology-specific camouflaged circuit. The technology-specific camouflaged circuit will be connected to the gate-level netlist in a similar way to how the VCL cell was connected to the RTL model, and subsequently to how the VCL cell was connected to the pre-replacement gate level netlist. Additional signals may be selected to be connected to the technology-specific camouflage circuit.

A formal logic equivalence check (LEC) should confirm that the camouflaged netlist is logically equivalent to the original uncamouflaged circuit.

F. *Physical Design & Verification*

After Camo Cell Replacement, the netlist contains technology-specific logic gates and technology-specific

DISTRIBUTION STATEMENT A.
Approved for public release: distribution is unlimited.

camouflage cells. Physical design and verification may proceed in the standard fashion, using technology-specific physical and logical models for the conventional logic cell library and the camouflaged logic cell library. To prevent camouflaged cells from being optimized out of the design, technology-specific camouflage cells should be don't-touched (prevent tools from modifying) and black-boxed (prevent tools from knowing the cell's logical functions) for the rest of the design flow.

III. Examples

Examples of the usage and the technology-specific cell replacement of undriven and driven VCL cells are shown here. Examples of hardware description language code are shown in Verilog HDL. The following examples cover the design flow steps up to obtaining a camouflaged netlist.

A. Undriven VCL Cell

Illustrated in Figure 3 is an example of the camouflaged ASIC design flow using an undriven virtual camouflage cell. The steps of RTL Design, Logic Synthesis, and Camouflage Cell Replacement are shown below along with an example circuit, specified in RTL, and synthesized into technology-specific camouflaged netlist. The example circuit is a registered half-adder, and the RTL designer inserted an undriven VCL cell (VCL_INST1) to protect the Carry output of the half-adder. The logic synthesis step synthesized the RTL into technology-specific gates, but it did not touch the VCL_INST1 component. The resulting pre-replacement netlist contains the VCL cell. In the camouflage cell replacement step, a netlist editing program performs a *Swap & Merge* technique, described below for undriven VCL cells. A technology-specific camouflaged circuit replaces the VCL cell and has an additional don't-care input I1, which is connected to the Sum pin of the half-adder, a node which was chosen at random. In this example, the technology-specific camo circuit drives a constant VSS output although it appears to have a different logic function. A newly-instantiated merge function gate, an XOR gate in this example, merges the output of the camouflage circuit with the original driver's output, the CO pin of the half adder. The technology-specific camouflaged circuit drives a constant value of logic-0, but its layout suggests that it performs a different logic function. The output of the merge function gate (the camouflaged signal) is now protected from reverse engineering by the camouflage circuit because the correct function of the camouflage circuit must be extracted for the camouflaged signal to have the correct value. The resulting camouflaged netlist is ready to undergo physical design and verification. The *Swap & Merge* technique is described below.

1) Locate the net, or *original_driver*, that connects to the undriven VCL cell's input, and locate that net's *original_loads*. Disconnect the *original_driver* from the *original_loads*.

2) Swap the undriven VCL cell with a technology-specific camouflaged circuit that drives a constant VDD or VSS, but appears to have a different logic function.

3) Instantiate a logic gate to serve as a *merge function*. Appropriate logic functions for the merge function include but are not limited to AND2 or XNOR2 for a camouflaged circuit that drives VDD, and include but are not limited to OR2 and XOR2 for a camouflaged circuit that drives VSS.

4) Connect the output of the technology-specific camouflaged circuit to one input of the *merge function*.

5) Connect the *original_driver* to the other input of the merge function.

6) Connect the *merge function*'s output, now protected by camouflaged circuitry, to *original_loads*.

7) Choose additional nodes within the design and connect them to don't-care inputs of the camouflaged circuit.

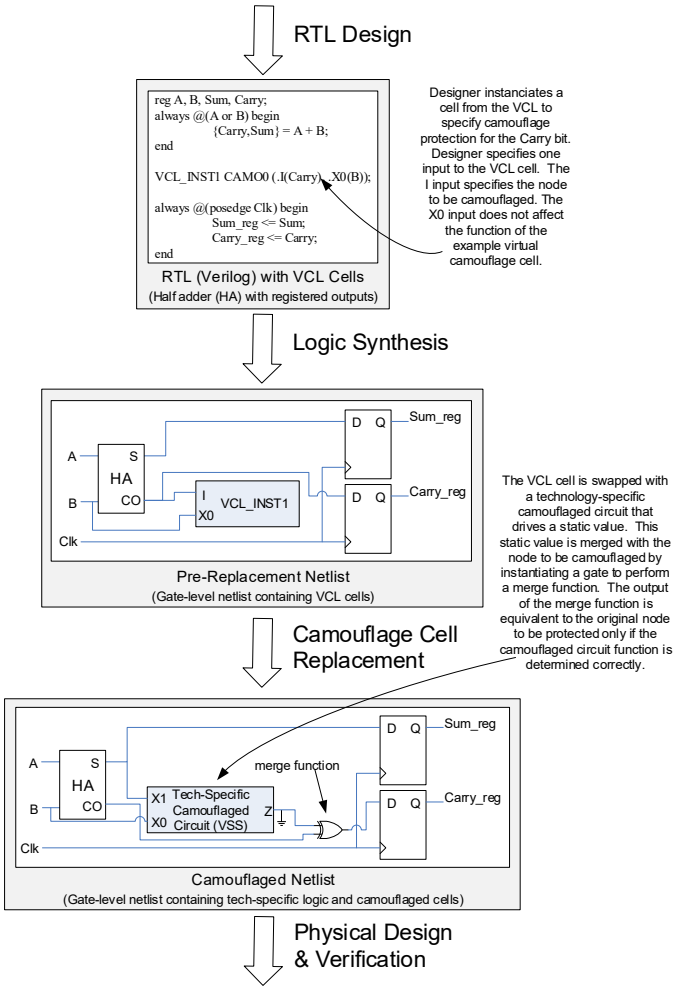

Fig. 3. An example of relevant steps of a camouflaged ASIC design flow using an undriven VCL cell.

B. Driven VCL Cell

Illustrated in Figure 4 is an example of the camouflaged ASIC design flow using a driven virtual camouflage cell. The steps of RTL Design, Logic Synthesis, and Camouflage Cell Replacement are shown below along with an example circuit, specified in RTL, and synthesized into technology-specific camouflaged netlist. The example circuit is a registered half-adder with the carry bit XORed with an additional input, and the RTL designer specified the XOR2 function to be a

DISTRIBUTION STATEMENT A.
Approved for public release: distribution is unlimited.

camouflaged function by instantiating the VCL cell (VCL_XOR2) perform the logical XOR. The logic synthesis step synthesized the RTL into technology-specific gates, but it did not touch the VCL_XOR2 component. The resulting pre-replacement netlist contains the VCL cell. In the camouflage cell replacement step, the VCL cell undergoes the *Swap* technique described above for driven VCL cells. A compatible technology-specific camouflaged circuit performing and XOR2 function replaces the VCL XOR2 cell. The technology-specific circuit has an additional don't-care input I0, which is connected to the Sum pin of the half-adder, a node which was chosen at random. The technology-specific camouflaged circuit performs an XOR2 logic function, but its layout suggests that it performs a different logic function. The output of the tech-specific camouflaged circuit is now protected from reverse engineering by the camouflage circuit because the correct function of the camouflage circuit must be extracted for the camouflaged signal to have the correct value. The resulting camouflaged netlist is shown near the bottom of the diagram and is ready to undergo physical design and verification. The *Swap* technique is described as follows:

1) Swap the driven VCL cell with a functionally-equivalent technology-specific camouflaged circuit.

2) Choose additonal nodes within the design to connect ny additional don't-care inputs of the camouflaged circuit.

IV. Conclusions

Using virtual camouflage cells from a virtual camouflage library (VCL) in one's design allows for the specification of camouflage features in a technology-independent way. The technology independence makes the VCL a suitable solution for incorporating silicon camouflage into RTL designs. The VCL facilitates the RTL designer's involvement in the hardware obfuscation process to obtain a higher quality-of-result, and enables reuse of the camouflaged design at different technology nodes for which a camouflaged cell library is available.

V. References

[1] "Circuit Camouflage Technology, SMI IP Protection and Anti-Tamper Technologies", www.smi.tv/SMI_SypherMedia_Library_Intro.pdf.

[2] Cocchi, R., Baukus, P., Chow, L.W., Wang, B., "Circuit Camouflage for Hardware IP Protection", Proceedings of the 51st Annual Design Automation Conference, 2014.

[3] "VLSI/ASIC Design Flow", https://ishwaki.wordpress.com/2016/10/30/vlsiasic-design-flow/

[4] "Introduction to ASIC Design Flow", https://anysilicon.com/introduction-asic-design-flow/.

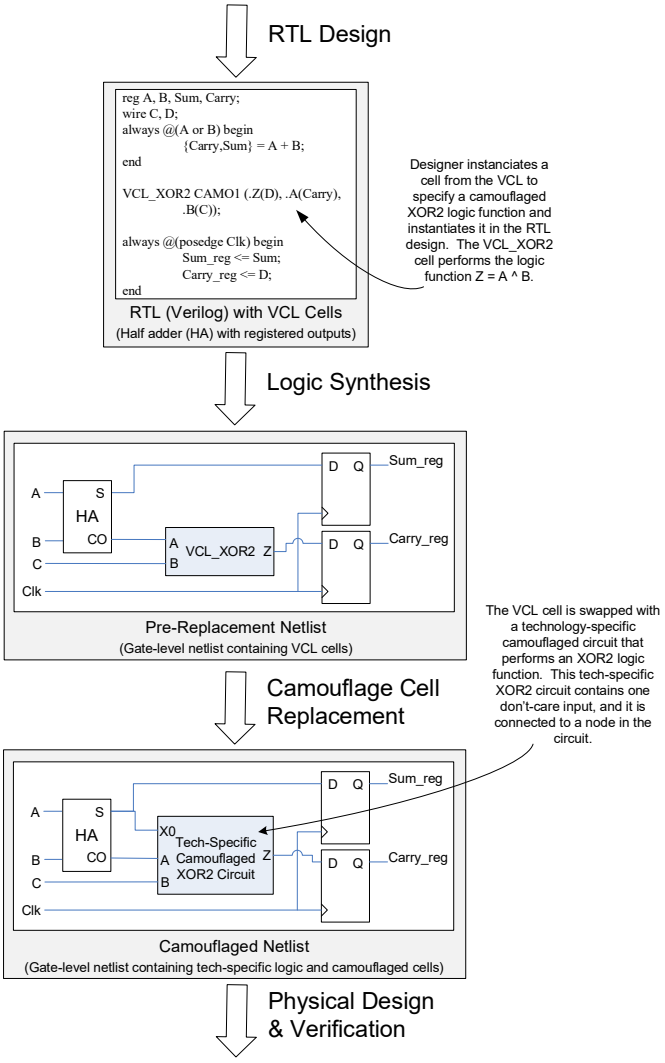

Fig. 4. An example of relevant steps of a camouflaged ASIC design flow using a driven VCL cell.

DISTRIBUTION STATEMENT A.
Approved for public release: distribution is unlimited.