Part 17 IP Core Protection Chapter Obfuscation of Designs

Protecting Your IP Cores – Part I Soft IP, Section Three: Obfuscation of Designs

Hardware obfuscation is a technique by which the description or the structure of electronic hardware is modified to intentionally conceal its functionality, which makes it significantly more difficult to reverse-engineer. In other words, hardware obfuscation modifies the design in such a way that the resulting architecture becomes un-obvious to an adversary. The obfuscated design will exhibit a correct function only when a correct key is applied. These obfuscation techniques can protect the IC from piracy and overbuilding.

Obfuscation can also help to effectively hide security features in an IC and thus enable protection of ICs from counterfeiting and cloning in fabrication facilities.

Obfuscating and tamper-proofing techniques are **not the optimal way for HDL code protection**. First, they make programs less readable and harder (if not impossible) to modify, which are all against the incentive to release soft IPs for better design reuse. Secondly, the continuous push for HDL design standards reduces the power of such protections.

In the past few years, research on logic obfuscation has received considerable attention and has made remarkable progress. This research can be categorized into different hierarchical levels: fabrication level [9, 17, 18, 93], cell level [27, 43, 49, 61], and netlist level [39, 43, 61, 77, 78, 105]. At the fabrication level, the main focus is on the nano-device structures that can hide the circuit's functionality. For camouflaging, this includes using doping levels, dummy-contacts between metal layers, and so on. For locking at the device level, tamper-resistant programmable technologies are needed. At the cell level, the nano-primitives are used to develop different cell designs that can manifest reverse engineering ambiguity. The next step is to transform the original circuit and incorporate these cells with the goal of maximizing security with minimum area/delay/power overhead. The majority of the algorithmic work involved with locking/camouflaging hence focuses on a netlist-level abstraction [69, 99, 100, 103]. With a netlist-level abstraction both types of obfuscation (camouflaging and locking) can be modeled with the same mathematics resulting in attacks that can target both techniques interchangeably.

Types of Obfuscation

Active Hardware Obfuscation Techniques

The **active hardware obfuscation** techniques directly alter the functionality of the system. Often the active hardware obfuscation techniques are "key-based", such that normal functionality of the obfuscated design can only be enabled by the successful application of a single pre-determined key or a sequence of secret keys at the input; otherwise the circuit operates in a mode, which exhibits incorrect functionality. This can be done by embedding a well-hidden **finite state machine (FSM)** in the circuit to control the functional modes based on application of key.

The technique of **key-based, active hardware obfuscation** is similar in principle to private-key cryptographic approaches for information protection, since the "key sequence" for the obfuscated design plays a similar role as the cryptographic key. The technique can be applied at different levels of hardware description, namely gate-level or register transfer level (RTL) design and hence can be used to protect soft, firm and hard IP cores.

Passive Hardware Obfuscation Techniques

The **passive hardware obfuscation** techniques do not directly affect the functionality of the electronic system.

In fact, the passive techniques modify the circuit description in a soft form (e.g. syntactic changes), such that it becomes difficult for a human reader to understand the functionality of the circuit. These approaches typically employ either string-substitution (including variable name change, comment removal, etc.), or structural change in the hardware description language (HDL) description of a circuit (including loop unrolling, register renaming, etc.). A major shortcoming of the passive approaches is that they do not modify the black box functionality of a circuit, and hence cannot prevent potential usage of an IP as black-box in a design. Moreover, the actual strength of such passive obfuscation is debatable, since, in general, black-box obfuscation does not exist, at least for software programs computing certain mathematical functions.

保护您的 IP 核 - 第一部分 软 IP 核，第三章：硬件设计的混淆

硬件混淆是一种通过修改电子硬件的描述或结构以故意隐藏其功能的技术，这使得逆向工程变得更加困难。换句话说，硬件混淆会以这样一种方式修改设计，使得最终的架构对对手来说变得不明显。只有当应用正确的密钥时，混淆设计才会显示正确的功能。这些混淆技术可以保护 IC 免受盗版和过度构建。

混淆还有助于有效地隐藏 IC 中的安全特性，从而保护 IC 在制造设施中免受伪造和克隆。

混淆和防篡改技术并不是 HDL 代码保护的最佳方式。首先，它们降低了程序的可读性和修改难度（如果不是不可能的话），这都违背了发布软 IP 以更好地重用设计的动机。其次，对 HDL 设计标准的不断推动降低了这种保护的能力。

混淆类型

主动硬件混淆技术

主动硬件混淆技术直接改变系统的功能。通常，主动硬件混淆技术是“基于密钥的”，因此混淆设计的正常功能只能通过在输入端成功应用单个预定密钥或一系列秘密密钥来启用; 否则, 电路会在显示不正确功能的模式下运行。这可以通过在电路中嵌入一个隐藏良好的有限状态机（FSM）来实现，以根据密钥的应用来控制功能模式。

基于密钥的主动硬件混淆技术在原理上类似于用于信息保护的私钥加密方法，因为用于混淆设计的“密钥序列”起着与加密密钥类似的作用。该技术可应用于不同级别的硬件描述，即门级或寄存器传输级（RTL）设计，因此可用于保护软核、固核和硬核 IP。

被动硬件混淆技术

被动硬件混淆技术不直接影响电子系统的功能。

事实上，被动技术以软形式修改电路描述（例如句法变化），使得人们难以理解电路的功能。这些方法通常采用字符串替换（包括变量名更改、注释删除等）或电路的硬件描述语言（HDL）描述中的结构更改（包括循环展开、寄存器重命名等）。无源方法的一个主要缺点是它们不会修改电路的黑盒功能，因此无法防止 IP 在设计中用作黑盒。此外，这种被动混淆的实际强度值得商榷，因为通常不存在黑盒混淆，至少对于计算某些数学函数的软件程序而言。

说明：

本文是由 陈明华 编写的“硅半导体 IP – 不仅仅是设计”一书（原文英文）“第 17 部分： IP 核保护”的一部分。此次发表的中文由英文机器翻译过来。您还可以在他的网站上找到更多信息。

Structural or Layout-Level Hardware (IP) Obfuscation

Structural obfuscation is a process of concealing the identity of an integrated circuit (IC) through architectural transformations without disturbing the functional behavior. In other words, Structural Obfuscation transforms the circuit of an IC in such a way that it becomes un-obvious to an adversary. Structural Obfuscation aims to transform the circuit structure in such a way that it results into significant modification at gate-level/RT-level. Structural obfuscation is performed to increase the complexity against reverse engineering (RE) attacks i.e. it makes the RE based attack difficult (makes it highly time consuming) to launch. Structural Obfuscation does not make use of encryption or logic locking through application of key sequences such as in Functional Obfuscation. This type of obfuscation performs certain high level transformation on the data flow graph representation to convert it into an unknown form that reflects a un-obvious architecture at RTL or gate level. More the change at gate-level due to structural obfuscation more un-obvious is the circuit structure to an adversary. Thus more stronger is the obfuscation resulting into higher security against RE attacks.

Structural obfuscation can be applied in the context of digital signal processing (DSP) cores/Multimedia cores/IP cores which are usually complex in size with over 30K-50K gates, as well as to combinational/sequential circuits. In the context of DSP cores Structural Obfuscation has been successfully performed using compiler-driven transformations such as loop folding, loop unrolling, loop shifting, loop compaction, redundant operation removal, logic transformation, resource transformation, tree height transformation, loop invariant code motion etc. Almost all of these high-level transformations (HLTs) potentially have the capability to significantly transform the circuit structure of a DSP/Multimedia cores/IP cores while preserving the original functionality. For example, multimedia core such as gray-scale JPEG CODEC on application of tree height transformation may result into transformation of more than 10,000 gates. This makes the resulting gate architecture highly un-obvious to an adversary.

Structural Obfuscation can be considered as a passive model from attacker's perspective.

Structural obfuscation offers a means to effectively secure the contents of an intellectual property (IP) cores used in a system-on-chip (SoC).

Algorithmic Transformation Based Structural Obfuscation

- Multi-stage HLT (high-level transformation) driven structural obfuscation methodology
- Loop-based high-level transformations to enhance the obfuscation complexity.

This novel structural obfuscation methodology is developed specifically for protecting digital signal processor (DSP) IP core at the architectural synthesis design stage. The obfuscation methodology specifically targets at protection of IP cores that involve complex loops.

Multi-stage HLT (high-level transformation) driven structural obfuscation methodology during algorithmic synthesis is achieved through five different compiler-based HLT techniques. They are (1) Redundant Operation Elimination (ROE) (2) Logic Transformation (LT) (3) Tree Height Transformation (THT) (4) Loop Unrolling (LU) (5) Loop Invariant Code Motion (LICM). Each of these can yield camouflaged functionally equivalent designs. This approach takes the input of an application in the form of a loop-based CDFG and applies each of the aforementioned HLT to obfuscate it.

In addition, low cost obfuscated design is generated through the use of multi-stage algorithmic transformation and particle swarm optimization (PSO)-drive design space exploration (DSE). It accepts the obfuscated design of the application in the form of loop-based CDFG, module library, terminating criteria of PSO, control parameters (like inertia weight, social factor, cognitive factor etc.) and user given area-delay constraints as inputs and generates a low-cost optimized obfuscated IP design as output, which satisfies the user-given area-delay constraints.

The above approach hides the functionality of the application (IP) through transformations and randomized placement of logic elements from an adversary during algorithmic synthesis. The output of algorithmic synthesis is structurally obfuscated RTL description (datapath and controller designs) of a reusable IP core.

保护您的 IP 核 - 第一部分 软 IP 核，第三章：硬件设计的混淆

结构或布局级硬件 (IP) 混淆

结构混淆是在不干扰功能行为的情况下通过架构转换隐藏集成电路 (IC) 身份的过程。换句话说，结构混淆以一种对对手来说变得不明显的方式来转换 IC 的电路。结构混淆旨在以这样一种方式转换电路结构，使其在门级/RT 级进行重大修改。执行结构混淆以增加针对逆向工程 (RE) 攻击的复杂性，即它使基于 RE 的攻击难以启动（使其非常耗时）。结构混淆不通过应用密钥序列（例如功能混淆）来使用加密或逻辑锁定。这种类型的混淆对数据流图表示执行某些高级转换，以将其转换为反映 RTL 或门级不明显架构的未知形式。由于结构混淆导致的门级变化更不明显的是对手的电路结构。因此，更强大的是混淆，从而提高了对 RE 攻击的安全性。

结构混淆可以应用在数字信号处理 (DSP) IP 核/多媒体 IP 核/IP 核的上下文中，这些内核通常在机构上很复杂，具有超过 30K-50K 的逻辑门，以及组合/顺序电路。在 DSP IP 核的上下文中，已经使用编译器驱动的转换成功地执行了结构混淆，例如循环折叠、循环展开、循环移位、循环压缩、冗余操作删除、逻辑转换、资源转换、树高转换、循环不变代码运动等. 几乎所有这些高级转换 (HLT) 都可能具有显着转换 DSP/多媒体 IP 核的电路结构同时保留原始功能的能力。例如，灰度 JPEG CODEC 等多媒体核心对树高变换的应用可能会导致超过 10,000 个门的变换。这使得由此产生的门架构对对手来说非常不明显。

从攻击者的角度来看，结构混淆可以被认为是一种被动模型。

结构混淆提供了一种有效保护片上系统 (SoC) 中使用的知识产权 (IP) 核内容的方法。

基于算法转换的结构混淆

- 多阶段 HLT（高层转换）驱动的结构混淆方法论
- 基于循环的高级转换，以提高混淆复杂度。

这种新颖的结构混淆方法是专门为在架构综合设计阶段保护数字信号处理器 (DSP) IP 核而开发的。混淆方法专门针对保护涉及复杂循环的 IP 核。

算法合成期间的多阶段 HLT（高级转换）驱动结构混淆方法是通过五种不同的基于编译器的 HLT 技术实现的。它们是 (1) 冗余操作消除 (ROE) (2) 逻辑变换 (LT) (3) 树高变换 (THT) (4) 循环展开 (LU) (5) 循环不变代码运动 (LICM)。这些中的每一个都可以产生伪装的功能等效设计。这种方法以基于循环的 CDFG 形式获取应用程序的输入，并应用上述每个 HLT 对其进行模糊处理。

此外，通过使用多阶段算法转换和粒子群优化 (PSO) 驱动设计空间探索 (DSE) 生成低成本的模糊设计。它以基于循环的 CDFG、模块库、PSO 的终止标准、控制参数（如惯性权重、社会因素、认知因素等）和用户给定的区域延迟约束的形式接受应用程序的混淆设计作为输入和生成一个低成本优化的混淆 IP 设计作为输出，它满足用户给定的区域延迟约束。

上述方法通过在算法综合期间对对手进行逻辑元素的转换和随机放置来隐藏应用程序 (IP) 的功能。算法综合的

输出是可重用 IP 内核的结构模糊 RTL 描述（数据路径和控制器设计）。

说明：

本文是由 陈明华 编写的“硅半导体 IP – 不仅仅是设计”一书（原文英文）“第 17 部分： IP 核保护”的一部分。此次发表的中文由英文机器翻译过来。您还可以在他的网站上找到更多信息。

Functional Hardware (IP) Obfuscation

Another type of DSP Core Obfuscation method is called 'Functional Obfuscation' - It uses a combination of AES and IP core locking blocks (ILBs) to lock the functionality of the DSP core using key-bits. Without application of correct key sequence, the DSP core produces either wrong output or no output at all

Functional Obfuscation: This is realized by encryptingthe normal functionality of a DSP circuit with one ormore sets of keys. The DSP circuit cannot functioncorrectly without the keys. This corresponds to an activemodel from attacker’s perspective.

Logic Obfuscation, Logic Locking, Logic Encryption, Key-Based Obfuscation, Logic Masking

Logic obfuscation can be classified into two types: a) sequential, and b) combinational. In **sequential logic obfuscation**, additional invalid or blocking states are inserted into the finite state machine (FSM) of a design. The FSM is constructed in such a way that the design executes properly and reaches a valid state if the correct key is applied. In **combinational logic obfuscation** additional XOR/XNOR gates are introduced into the circuit to protect the IP core. All these **key-based obfuscation** techniques increase the chance of higher design overhead due to insertion of additional components. Furthermore, do not apply any optimization technique to achieve low-cost functionally obfuscated design.

Sequential Logic Obfuscation

In **sequential logic obfuscation**, additional invalid or blocking states are inserted into the finite state machine (FSM) of a design. The FSM is constructed in such a way that the design executes properly and reaches a valid state if the correct key is applied.

Sequential encryption schemes often focus their efforts on the implicit logical FSM. The straightforward approach increases the FSM's state space thereby reducing access of the original FSM. In addition access to the circuit's true logic can require a particular sequence of input vectors. Other methods incorporate special locking states in the updated FSM. These methods select a subset of states that can be accessed by the new reset state but cannot reach the states of the original FSM.

Two main methods for increasing the state space exist. The first method changes the logic of the registers to utilize previously unreachable states. The other method involves inserting additional registers that usually but not necessarily act as flags for the FSM's behavior. Additional registers tend to be very appealing since the state space increase exponentially with the number of inserted registers. The major detriments to a large number of register insertions is the time to unlock, area, and power overhead.

An example of sequential circuit encryption, **HARPOON**, inserts additional state elements (SE) and combinational logic that adversely affects the behavior of the netlist while the circuit is locked. The inserted SEs control the activation of the inserted combinational modules, that have the potential to corrupt parts of the netlist. Moreover HARPOON's FSM's state space is partitioned into three general sections (modes): obfuscation, authentication, and original. The obfuscation mode, the first part of the obfuscation mode, corrupts parts of the netlist. The authentication mode simply watermarks the netlist. The original mode, as it sounds, does not corrupt the netlist's internal signals and allows for normal execution.

Combinatorial Logic Obfuscation

In **combinational logic obfuscation** additional XOR/XNOR gates are introduced into the circuit to protect the IP core. This type of obfuscation performs changes to the gate level structure of the circuit itself.

Logic Encryption

Logic encryption hides the circuit functionality by inserting a number of key gates which are controlled by key bits. Only when the correct key vector is applied, the circuit can operate correctly. On applying incorrect key vectors, the circuit cannot generate correct outputs. Consequently, even if attackers obtain the encrypted netlist by reverse-engineering, they cannot know the correct netlist unless they know the key.

The key gates are a combination of AND/OR or XOR/XNOR or multiplexers (MUX), look Up Tables. These key gates can be either inserted randomly or inserted selectively at specific locations of the circuit. The determination of selective insertion key gate locations requires an algorithm.

In encryption using gates, one input of the key gate is connected to any net inside the circuit and the other input is the key input. When the correct key input is given then the functionality of the circuit is correct, else the outputs are corrupted.

In encryption using MUXes, MUXes inserted are of size 2x1. One input of the MUX is correct value; other input is taken from any net in the circuit. The select bit of the MUX is the key bit. MUX based encryption can also be carried out by using an Obfuscation Cell (OC). An obfuscation cell is a combination of MUX and an inverter. In both the methods, the select line of the MUX acts is the key bit. For correct key input, the circuit gives the correct behavior, whereas if incorrect key bit is given then output obtained mismatches from ideal output.

In these methods, each key bit could be sensitized to the output of the circuit and the information regarding the design could be obtained.

Logic Locking

Logic locking is based on adding some form of programmability to the design and ensuring that
the circuit cannot function properly without the circuit being programmed with a secret string
of configuration data referred to as the “key”