Protecting DSP Circuits through Obfuscation

Yingjie Lao and Keshab K. Parhi
Department of Electrical and Computer Engineering, University of Minnesota
Minneapolis, MN, USA
Email: {laoxx025, parhi}@umn.edu

***Abstract*—This paper presents a novel approach to protect digital signal processing (DSP) circuits through obfuscation by using *high-level transformations*. The goal is to design DSP circuits that are harder to reverse engineer. High-level transformations of iterative data-flow graphs have been exploited for area-speed-power tradeoffs. This is the *first* attempt to develop a design flow to apply high-level transformations that not only meet these tradeoffs but also simultaneously obfuscate the architectures both *structurally* and *functionally*. Several modes of operations are introduced for obfuscation where the outputs are either meaningful from a signal processing point of view, but functionally incorrect, or non-meaningful. Experimental results show that the proposed methodology only introduces relatively small overhead, while a high level of obfuscation is achieved. For instance, the area overhead for a $(3l)$th-order IIR filter benchmark is only 17.7% with a 128-bit *configuration key*.**

I. Introduction

The problem of hardware security is a serious concern that has led to a lot of work on hardware prevention of piracy and intellectual property (IP), which can be broadly classified into two main categories: 1) authentication-based approach, or 2) obfuscation-based approach. The authentication-based approaches include physical unclonable functions (PUFs) based authentication [1], digital watermarking [2], [3], key-locking [4], and hardware metering [5]. Obfuscation-based approach is of interest in this paper, which is a technique that transforms an application or a design into one that is functionally equivalent to the original but is significantly more difficult to reverse engineer. Some hardware protection methods are achieved by altering the human readability of the hardware description language (HDL) code, or by encrypting the source code based on cryptographic techniques [6]. Recently, a number of hardware protection schemes have been proposed that modify the finite-state machine (FSM) representations to obfuscate the circuits [7], [8].

However, to the best of our knowledge, no obfuscation-based IP protection approach has been proposed for DSP circuits in the literature. This paper, for the first time, presents design of obfuscated DSP circuits via high-level transformations that are harder to reverse engineer. From this standpoint of view, a DSP circuit is more *secure*, if it is harder for the adversary to discover its functionality. In other words, a high level of *security* is achieved if the functionality of a DSP circuit is designed to be *hidden* to the adversary. Our goal is to design obfuscated circuits by applying high-level transformations during the design phase. The key idea of the proposed work is to generate meaningful design variations by exploiting high-level transformations.

This paper is organized as follows. Section II presents how these high-level transformation techniques can be used for *structural obfuscation* and *functional obfuscation*. In Section III, we propose a novel design methodology for DSP circuit obfuscation via high-level transformations. A case study is presented in Section IV. Finally, we illustrate the effectiveness of the proposed design methodology by experimental results in Section V.

II. Obfuscation via High-Level Transformations

A. Hiding Functionality by High-Level Transformations

High-level transformations have been known for a long time and have been used in a wide range of applications, such as pipelining [9], interleaving [9], folding [10] and unfolding [11], and have been used in synthesis of DSP systems [12]. These techniques can be applied at the algorithm or the architecture level to achieve a tradeoff among different metrics of performance, such as area, speed, and power [13]. However, the use of high-level transformations from a security perspective has not been studied before. In fact, high-level transformations naturally provide a means to obfuscate DSP circuits both *structurally* and *functionally*. *Structural obfuscation* and *functional obfuscation* are defined as follows:

(a) *structural obfuscation*: achieved by structural modification, which is realized by altering the structure of a DSP circuit by using high-level transformations. This is a so-called "passive" technique, which does not directly affect the functionality of the DSP circuit.

(b) *functional obfuscation*: achieved by functional modification, which is realized by encrypting the normal functionality of a DSP circuit with a key. The DSP circuit cannot function correctly without the key. This is an "active" technique, which directly alters the functionality.

High-level transformations alter the structure of a DSP circuit, while maintaining the original functionality. For instance, different folding sets lead to a family of folded architectures; this can be exploited for *structural obfuscation*. As a result, circuits with the same functionality may have very different structures. Furthermore, high-level transformations may lead to architectures whose functionalities are not obvious. Take an extreme case for example, many filters can be folded into one multiply-accumulator (MAC), but their functionalities are not the same. In other words, one MAC with proper switches can implement many digital filters. It is important to note this kind of *structural obfuscation* can be applied beyond the architecture level. For example, at the HDL level or the gate-level netlist, high-level transformations can also lead to an obfuscated version of a DSP circuit. Therefore, circuits with different functionalities could have a similar structure by employing high-level transformations. Comparing the folded structures (a) and (b) in Figure 1, it can be observed that

the two structures are exactly the same, except the switch instances. However, their functionalities are different, i.e., the former implements a 1st-order IIR filter, while the latter a 2nd-order IIR filter. In conclusion, if the *switch instances are invisible* to the adversary, the DSP systems will be hard to reverse engineer. The adversary who only has knowledge of the structural information but lacks knowledge of the switch instances cannot easily discover the functionality of a DSP circuit [14].

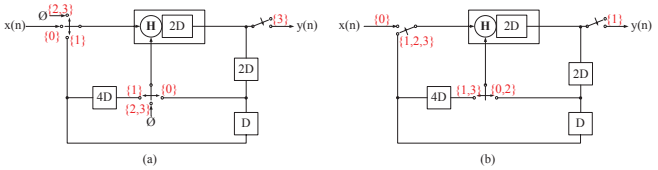

Fig. 1: (a) Folded 1st-order IIR filter: S={A, B, ∅, ∅} (b) Folded 2nd-order IIR filter: S={A, B, C, D}. The switch instance "i" corresponds to clock cycle $4l + i$.

B. Design of Secure Switch

The DSP circuits can be obfuscated via high-level transformations by appropriately designing the switches in a secure manner, which are often modeled as FSMs. Indeed, existing works have demonstrated that *functional obfuscation* can be achieved by embedding a well-hidden FSM (i.e., obfuscating FSM) in the circuit to control the functionality based on a key [8], [15]. In contrast to these existing methods, we propose a novel *functional obfuscation* scheme along with *structural obfuscation* by using high-level transformations, which improves the security of DSP circuits by providing a two-level protection.

The detailed implementation is shown in Figure 2. Note that other secure switch designs, whose detailed switch instances are hidden to the adversary, can also be adopted in the framework. We employ the idea of hardware design obfuscation as an activation sequence required before configuration. A *reconfigurator* is introduced to control the output function G, next-state function F, and the initial state S_0. At the beginning of a successful configuration, the reset signal will be generated to reset the initial state of the FSM. At the same time, the output function and the next-state function are reconfigured based on the signals from the reconfigurator. In our design, the operation of the secure switch is determined by a *configuration key*, which consists of two parts: an L-bit *initialization key* and a K-bit *configure data*. The *initialization key* is used as the input of the obfuscating FSM, while the *configure data* are applied to the *reconfigurator* to control the operation of the switches. As the configuration of the switch is only enabled after receiving a correct *initialization key*, hostile attempts of the *configure data* can be avoided by the obfuscating FSM.

III. Design Flow of the Proposed DSP Circuit Obfuscation Approach

A. Design Methodology

In this section, we propose a novel DSP hardware protection methodology through obfuscation by hiding functionality via high-level transformations. This approach helps the designer to protect the DSP design against piracy by controlling the circuit configuration among the generated variation modes

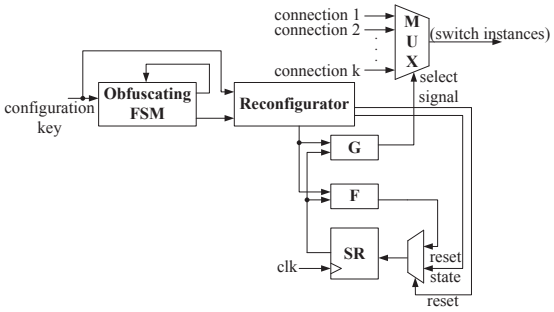

Fig. 2: Proposed secure switch design.

of the original design. The detailed design flow is described below:

Step 1: DSP algorithm. This step generates the DSP algorithm based on the DSP application.

Step 2: High-level transformation selection. Based on the specific application, appropriate high-level transformation should be chosen according to the performance requirement (e.g., area, speed, power or energy).

Step 3: Obfuscation via high-level transformation. Selected high-level transformations are applied simultaneously with obfuscation where variation modes, and different configurations of the switch instances are designed.

Step 4: Secure switch design. The secure switch is designed based on the variations of high-level transformations. Note that different *configure data* could be mapped into the same mode, which only involves simple combinational logic synthesis.

Step 5: Two-level FSM generation. The reconfigurator and the obfuscating FSM are incorporated into the DSP design as shown in Figure 2. The *configuration key* is generated at this step.

Step 6: Design specification. This step includes the HDL and netlist generation and synthesis of the DSP system.

The proposed design methodology does not require significant changes to established verification and testing flows. In fact, the obfuscated DSP circuit with the correct key behaves just like the original circuit.

B. Architecture of the Proposed Obfuscated DSP Circuit

The complete system of the proposed obfuscated DSP circuit is illustrated in Figure 3. The DSP circuits are obfuscated by introducing a FSM whose state is controlled by a key. The FSM enables a reconfigurator that configures the functionality mode of the DSP circuit. High-level transformations lead to many equivalent circuits and all these create ambiguity in the structural level. High-level transformations also allow design of circuits using same datapath but different control circuits. For example, a datapath may implement a 3rd-order or a 6th-order digital filter, or in general a $(3l)$th-order filter, where l is a positive integer. These correspond to different modes. While these modes generate outputs that are functionally incorrect, these may represent correct outputs under different situations, since the output is meaningful from a signal processing point of view. Finally, other modes lead to non-meaningful outputs. The *initialization key* and the *configure data* must be known for the circuit to work properly. Consequently, the circuit behaves as an obfuscated circuit.

The obfuscating FSM and a portion of non-meaningful variation modes (i.e., we denote as *alarm modes*) can both be utilized for security check purpose. If the circuit continuously receives wrong *initialization key* or *configure data*, the adversary is prevented from further attempts of the *configuration key* by the permanent denial of use block.

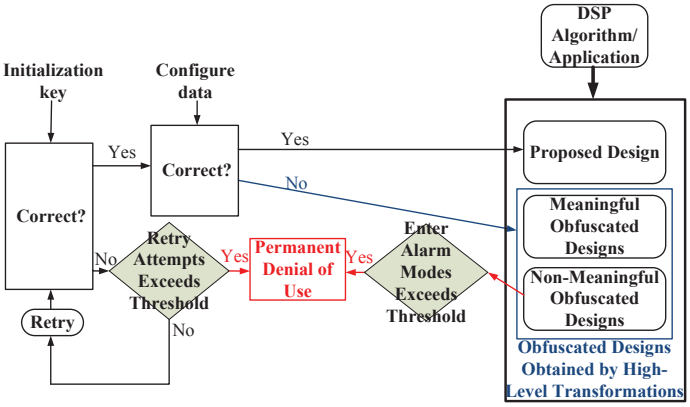

Fig. 3: Architecture of the proposed obfuscated DSP circuit.

IV. A Case Study: Hierarchical Contiguous Folding Algorithm

The variation modes are generated based on the selected transformation algorithm, which are different for various high-level transformations. It is difficult to cover very large number of existing high-level transformations in this paper. We just present an example in this paper for demonstration. The proposed design methodology can also be extended to other high-level transformations.

Hierarchical folding approach is a novel folding technique that combines folding of M cascaded stages to one hardware block, and folding N operations inside each section to a hardware functional unit. Two hierarchical folding algorithms are presented in [16], which include *hierarchical interleaved folding (HIF)* and *hierarchical contiguous folding (HCF)*. In this paper, we only address *hierarchical contiguous folding*, while it is also applicable to *hierarchical interleaved folding*. *Hierarchical contiguous folding* transformation executes all operations of one section before starting execution of operations of next section. The details are referred to [16].

We propose a design obfuscation algorithm for generating variation modes by varying the number of sections in the cascade structure based on the HCF algorithm. For example, if the number of sections for a DSP system is l, then the algorithm can be described as (the total number of operations is still NM, where $M \geq l$):

Design Obfuscation Algorithm based on the HCF Algorithm

1) Fold Alg^i by factor NM, with the folding set $\{X_0, X_1, ...X_{N-1}, \emptyset, \emptyset, ...\emptyset\}$, where the number of null operations corresponds to $(NM - N)$.
2) Replace each switch instance s by s,$s+N$,$s+2N$,...$s+(l-1)N$, and assign switch instances from lN to $MN-1$ to null operations.
3) Compute $D_F(Alg^j \xrightarrow{e} Alg^{j+1})$, for $j = 0, 1, 2...l-2$, and use these folded edges to replace the external inputs.

If $l = M$, this algorithm reduces to the *hierarchical contiguous folding* algorithm. From this algorithm, we can generate M meaningful modes correspond to $l = 1, 2, ..., M$. Furthermore, the reconfigurator can also be designed based on the variations of the HCF algorithm, which is a simple 2^K-to-M combinational logic design problem. Figures 1(a) and 1(b) illustrate examples of two variation modes generated by this modified obfuscation algorithm with a parameter $M = 2$. Note that this algorithm can be easily extended to other types of DSP systems where the sub-circuits are not directly connected.

V. Evaluation of the Proposed Methodology

Component overhead of the proposed obfuscation design includes: (a) additional control logic of switches, (b) reconfigurator, and (c) obfuscating FSM. These additional circuits only affect the switches of an obfuscated DSP circuit. In this paper, we present the area overhead results of the proposed obfuscating methodology for two DSP benchmark circuits: $(3l)$th-order IIR filter and $(12l)$-tap FIR filter. All circuits were synthesized using Synopsys Design Compiler with optimization parameters set for minimum area and mapped to a 65 nm standard cell library. We employ the *design obfuscation algorithm based on the HCF algorithm* to obfuscate the circuits. In our experiments, the $(3l)$th-order IIR filter is folded to 1 multiplier and 1 adder, while the $(12l)$-tap FIR filter is folded to 3 multiply-accumulators.

We take the $(3l)$th-order IIR filter benchmark as an example to illustrate the obfuscated design approach. Here, one section of the $(3l)$th-order IIR filter is a 3rd-order IIR filter as shown in Figure 4. We assume the desired functionality is an 18th-order IIR filter realized as a cascade of six 3rd-order IIR filter. In our experiment, the proposed *design obfuscation algorithm based on the HCF algorithm* is applied to the original 18th-order IIR filter to obfuscate this DSP circuit. In order to generate 8 meaningful variation modes, the parameters $M = 8$ and $N = 4$ are used to the structure with 6 sections of 3rd-order IIR filter (i.e., the original 18th-order IIR filter) and 2 additional sections of null operations. The switch instances of this folded design are periodic with period 32. The 8 meaningful modes correspond to $(3l)$th-order IIR filter where $l = 1, 2, ..., 8$, respectively. 8 non-meaningful variation modes are also incorporated. Each secure switch is controlled by the reconfigurator independently. Figure 5 shows an example of the switch connected to the input of the multiplier in the obfuscated design. This switch has 5 possible input paths, as the null operations are also integrated to the switches. Finally, an obfuscating FSM is also added into the secure switch design to provide the second-level protection of the obfuscated DSP circuit.

We present the area overhead for the two DSP circuit benchmarks as shown in Table I and Table II, respectively. The overhead percentages are computed based on the folded designs instead of the original circuits. The results are averaged area overheads over a number of different implementations. For certain lengths of *initialization key* and *configure data*, the patterns of the state transition graph in the design of obfuscating FSM and the input-output mappings in the design of reconfigurator would also affect the design overhead of the proposed obfuscated DSP circuit.

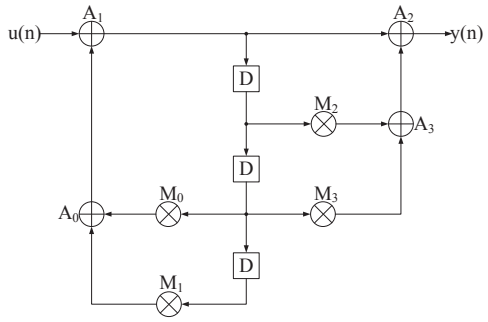

Fig. 4: A 3rd-order IIR filter.

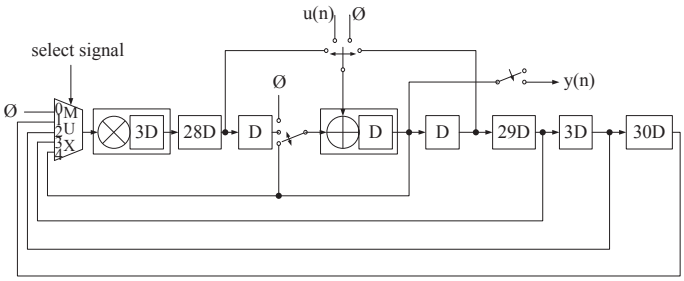

Fig. 5: The obfuscated design of the original 18th-order IIR filter.

It can be seen from Tables I and II that the overall overhead is about 17.7% for the $(3l)$th-order IIR filter with a 128-bit (64+64) *configuration key*, while the overhead is only about 7.1% for the $(12l)$-tap FIR filter also with a 128-bit *configuration key*. In the meanwhile, a high level of obfuscation is achieved, as the chance for an adversary to enter the DSP circuit into the desired mode is only $\frac{1}{2^{L+K}} = \frac{1}{2^{128}} = 2.94 \times 10^{-39}$. Note that these two DSP circuit benchmarks are both small circuits. In practice, as the DSP circuits are more complex, the overhead percentage would be even smaller. Moreover, when we compare the effects between L and K, it can be seen that the overhead increases more significantly with the increase of L. Thus, in order to achieve lower overhead, we should employ a longer *configure data* in designing obfuscated DSP circuits when the total length of the *configuration key* is bounded.

Discussion. As this paper is the first attempt to develop a methodology to obfuscate DSP circuits by utilizing high-level transformations, it is hard to compare with other existing obfuscation methods which are general to a wide variety of designs. Most of the hardware obfuscation techniques in the literature can also be applied to DSP circuits. However, the use of high-level transformations from a security perspective has not been incorporated into any of these prior hardware obfuscation techniques. The main advantage of the proposed methodology is the generation of meaningful variation modes from a signal processing point of view, since the meaningful modes create ambiguity to the adversary such that it is hard for the adversary to distinguish the desired functionality from other variation modes. While considering the metrics of the design performance, our proposed methodology is also superior. Area consumption is only slightly increased as the additional control logic is only built on the switches, instead of inserting additional states based on the entire circuit.

TABLE I: Overhead $(\%)$ of the $(3l)$th-order IIR filter benchmark

L \ K	4	8	16	32	64
4	3.8	4.0	4.3	4.8	5.9
8	4.9	5.0	5.4	5.9	6.9
16	6.6	6.7	7.0	7.6	8.7
32	9.7	9.8	10.2	10.8	11.9
64	15.4	15.7	16.0	16.6	17.7

TABLE II: Overhead $(\%)$ of the $(12l)$-tap FIR filter benchmark

L \ K	4	8	16	32	64
4	1.6	1.7	1.8	1.9	2.1
8	2.0	2.1	2.2	2.4	2.5
16	2.8	2.9	3.0	3.2	3.5
32	4.1	4.2	4.3	4.4	4.7
64	6.6	6.7	6.8	6.9	7.1

ACKNOWLEDGMENT

The authors gratefully acknowledge numerous discussions on this topic with Prof. Chris Kim.

REFERENCES

[1] G. E. Suh and S. Devadas, "Physical unclonable functions for device authentication and secret key generation," in *Proceedings of the 44th annual Design Automation Conference*, 2007, pp. 9–14.

[2] D. Kirovski, Y.-Y. Hwang, M. Potkonjak, and J. Cong, "Intellectual property protection by watermarking combinational logic synthesis solutions," in *Proceedings of International Conference on Computer Aided Design (ICCAD)*, 1998, pp. 194–198.

[3] F. Koushanfar and Y. Alkabani, "Provably secure obfuscation of diverse watermarks for sequential circuits," in *Proceedings of International Symposium on Hardware-Oriented Security and Trust (HOST)*, 2010, pp. 42–47.

[4] J. A. Roy, F. Koushanfar, and I. Markov, "EPIC: Ending piracy of integrated circuits," in *Proceedings of Design, Automation and Test in Europe (DATE)*, 2008.

[5] Y. M. Alkabani and F. Koushanfar, "Active hardware metering for intellectual property protection and security," in *Proceedings of the USENIX Security Symposium*, 2007, pp. 1–16.

[6] T. Batra, "Methodology for protection and licensing of HDL IP," [Online]. http://www.design-reuse.com/articles/12745, 2005.

[7] R. S. Chakraborty and S. Bhunia, "Hardware protection and authentication through netlist level obfuscation," in *Proceedings of International Conference on Computer Aided Design ICCAD*, 2008, pp. 674–677.

[8] R. S. Chakraborty and S. Bhunia, "RTL hardware IP protection using key-based control and data flow obfuscation," in *Proceedings of 23rd International Conference on VLSI design*, 2010, pp. 405–410.

[9] K. K. Parhi and D. G. Messerschmitt, "Pipeline interleaving and parallelism in recursive digital filters, part I: Pipelining using scattered look-ahead and decomposition," *IEEE Transactions on VLSI systems*, vol. 37(7), pp. 1099–1117, 1989.

[10] K. K. Parhi, C.-Y. Wang, and A. P. Brown, "Synthesis of control circuits in folded pipelined DSP architectures," *IEEE Journal of Solid State Circuits*, vol. 27, no. 1, pp. 29–43, 1992.

[11] K. K. Parhi and D. G. Messerschmitt, "Static rate-optimal scheduling of iterative data flow programs via optimum unfolding," *IEEE Transactions on Computers*, vol. 40, no. 2, pp. 178–195, 1991.

[12] C.-Y. Wang and K. K. Parhi, "High-level DSP synthesis using concurrent transformations, scheduling, and allocation," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 14, no. 3, pp. 274–295, 1995.

[13] K. K. Parhi, *VLSI Digital Signal Processing Systems: Design and Implementation*. John Wiley and Sons, 1999.

[14] K. K. Parhi, "Verifying equivalence of digital signal processing circuits," in *Proceedings of Conference on Signals, Systems and Computers (ASILOMAR)*, 2012, pp. 99–103.

[15] Y. Alkabani, F. Koushanfar, and M. Potkonjak, "Remote activation of ICs for piracy prevention and digital right management," in *Proceedings of International Conference on Computer-Aided Design (DAC)*, 2007, pp. 674–677.

[16] K. K. Parhi, "Hierarchical folding and synthesis of iterative data flow graphs," *IEEE Trans. Circuits and Systems-II: Transactions Briefs*, vol. 60, no. 9, pp. 597–601, 2013.