Obfuscating DSP Circuits via High-Level Transformations

Yingjie Lao, *Student Member, IEEE*, and Keshab K. Parhi, *Fellow, IEEE*

***Abstract*— This paper presents a novel approach to design obfuscated circuits for digital signal processing (DSP) applications using high-level transformations, a key-based obfuscating finite-state machine (FSM), and a reconfigurator. 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 meaningful from a signal processing point of view, but are functionally incorrect. Examples of such modes include a third-order digital filter that can also implement a sixth-order or ninth-order filter in a time-multiplexed manner. The latter two modes are meaningful but represent functionally incorrect modes. Multiple meaningful modes can be exploited to reconfigure the filter order for different applications. Other modes may correspond to nonmeaningful modes. A correct key input to an FSM activates a reconfigurator. The configure data controls various modes of the circuit operation. Functional obfuscation is accomplished by requiring use of the correct initialization key, and configure data. Wrong initialization key fails to enable the reconfigurator, and a wrong configure data activates either a meaningful but nonfunctional or nonmeaningful mode. Probability of activating the correct mode is significantly reduced leading to an obfuscated DSP circuit. Structural obfuscation is also achieved by the proposed methodology via high-level transformations. Experimental results show that the overhead of the proposed methodology is small, while a strong obfuscation is attained. For example, the area overhead for a $(3l)$th-order IIR filter benchmark is only 17.7% with a 128-bit configuration key, where $1 \leq l \leq 8$, i.e., the order of this filter should be a multiple of 3, and can vary from 3 to 24.**

***Index Terms*— Digital signal processing (DSP), functional obfuscation, hardware security, high-level transformations, intellectual property (IP) protection, obfuscation, reconfigurable design, structural obfuscation.**

I. Introduction

DIGITAL signal processing (DSP) plays a critical role in numerous applications, such as video compression, portable systems/computers, multimedia, wired and wireless communications, speech processing, and biomedical signal processing. However, as electronic devices become increasingly interconnected and pervasive in people's lives, security, trustworthy computing, and privacy protection have emerged as important challenges for the next decade. It is estimated that as much as 10% of all high-tech products sold globally are counterfeit which leads to a conservative estimate of $100 billion of revenue loss [1]. Therefore, DSP system designers have to pay more attention to the security perspective of DSP circuits, since the adversary can easily learn the functionality using massive attacking methods.

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 and 2) obfuscation-based approach. The authentication-based approaches include physical unclonable functions (PUFs)-based authentication [2], digital watermarking [3]–[6], key-locking scheme [7], [8], and hardware metering [9]. The focus of this paper is on obfuscation, 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 [10]. Recently, a number of hardware obfuscation schemes have been proposed that modify the finite-state machine (FSM) representations to obfuscate the circuits [11]–[13].

However, to the best of our knowledge, no obfuscation-based IP protection approach has been proposed specifically 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 even if the adversary can physically tamper the device. In other words, a high level of security is achieved if the functionality of a DSP circuit is designed to be hidden from the adversary.

The key contribution of this paper is a novel approach to design obfuscated DSP circuits by high-level transformations during the design stage. The DSP circuits are obfuscated by introducing an 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. Different variation modes can be inserted into the DSP circuits for obfuscation. While some 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. Other modes would lead to nonmeaningful outputs. The initialization key and the configure data must be known for the circuit to work properly. Consequently, the proposed design methodology leads to a DSP circuit that is both structurally and functionally obfuscated.

Manuscript received July 30, 2013; revised March 10, 2014; accepted April 28, 2014. Date of publication June 4, 2014; date of current version April 22, 2015.

The authors are with the Department of Electrical and Computer Engineering, University of Minnesota, Minneapolis, MN 55455 USA (e-mail: laoxx025@umn.edu; parhi@umn.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TVLSI.2014.2323976

Furthermore, the approach presented in this paper will prevent piracy from overproduction and mask theft, because the manufacturer would not have access to either the initialization key or the configure data. These keys could be programmed by another honest vendor after the chips have been fabricated or provided to the customers by the designer. Therefore, overproduced chips without the correct keys cannot function properly. This paper is an expanded version of [14].

The rest of this paper is organized as follows. In Section II, we present how the high-level transformation techniques can be used to hide the functionalities of DSP circuits, and make observations based on examples which lead to the work in the subsequent sections. Section III presents a detailed implementation of a secure switch, which can be used for obfuscation. In Section IV, a case study is presented to demonstrate the idea of generating variation modes simultaneously with performing high-level transformations during design stage. In Section V, we describe the complete design flow of the proposed DSP circuit obfuscation methodology. Section VI addresses the evaluations of the security aspects. We illustrate the effectiveness of the proposed design methodology by experimental results and analysis in Section VII. In Section VIII, we compare the proposed obfuscation approach with other existing methods. Finally, Section IX presents remarks, conclusions, and future directions.

II. Hiding Functionality by High-Level Transformations

High-level transformations [15] have been known for a long time and have been used in a wide range of applications, such as pipelining [16], interleaving [16], folding [17], unfolding [18], and look-ahead transformations (e.g., quantizer loops [19], multiplexer loops [20], [21], relaxed look-ahead [22], annihilation reordering look-ahead [23]), and have been used in synthesis of DSP systems [24]. 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 [25]. However, the use of high-level transformations from a security perspective has not been studied before. High-level transformations alter the structure of a DSP circuit, while maintaining the original functionality. These 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 different digital filters. Therefore, we can conclude that high-level transformations naturally provide a means to obfuscate DSP circuits both structurally and functionally. Structural obfuscation and functional obfuscation are defined as follows.

1) *Structural Obfuscation*: Any algorithm can be implemented by a family of architectures by using high-level transformations. These architectures enable structural obfuscation where the functionalities of the algorithms can be hidden. This can be considered as a passive model from attacker's perspective.
2) *Functional Obfuscation*: This is realized by encrypting the normal functionality of a DSP circuit with one or more sets of keys. The DSP circuit cannot function correctly without the keys. This corresponds to an active model from attacker's perspective.

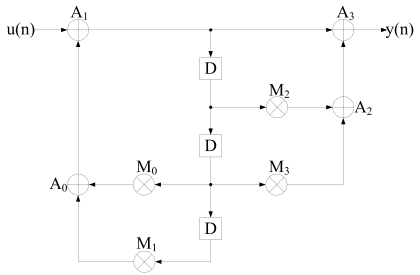


Fig. 1. Third-order IIR filter.

Folding is such an example of high-level transformation, which could be utilized to achieve design obfuscation. The folding transformation generates folded variants based on the folding set, which is the reverse of the unfolding transformation [18]. The choice of folding set is critical to the performance of the folded structure, since an appropriate choice of folding order can lead to an architecture with lower area and power. Folding sets can be designed intuitively to meet the performance requirements or can be obtained from a high-level synthesis system [24]. The details and other examples (e.g., interleaving) of how to hide the functionalities of DSP circuits by high-level transformations are described in [26] and [27]. We can observe that: 1) circuits with different functionalities can have a similar structure, and circuits with the same functionality may have very different structures; 2) structural obfuscation can be achieved by high-level transformations; and 3) if the switch instances are invisible to the adversary, then the DSP systems will be harder to reverse engineer, since the functionality of a DSP circuit is not obvious due to obfuscation achieved by high-level transformations. As a result, 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.

As an example, we consider a third-order IIR digital filter given by transfer function $H(z) = (1 + m_2 z^{-1} + m_3 z^{-2})/(1 - m_0 z^{-2} - m_1 z^{-3})$, as shown in Fig. 1. The coefficients m_i correspond to the multiplication M_i. We assume the availability of one 1-stage pipelined adder and one 3-stage pipelined multiplier. The filter is folded with folding factor $N = 4$ using the following folding sets:

$$M = \{M_0, M_1, M_2, M_3\}$$
$$A = \{A_0, A_1, A_2, A_3\}.$$

Folding sets represent the order of operations executed by the same hardware. For a folded system to be realizable, the folding equations, $D_F(U \xrightarrow{e} V) = Nw(e) - P_U + v - u$, must be greater or equal to 0 for all the edges in the diagram, where N is the folding factor, $w(e)$ is the number of delays from U to V, P_U represents the pipelining level of hardware functional unit for operation type U, and u and v represent the folding orders of U and V, respectively. Retiming and pipelining can be used to satisfy this property (or it can be determined that the folding sets are not feasible), as a preprocessing step prior to folding. The folded architecture is shown in Fig. 2. Fig. 3 presents the structure that the switch instances are designed to be invisible. Null operations are incorporated into the switches.

We consider the implementation of another third-order IIR filter given by transfer function $H(z) = (1 + m_2 z^{-1} + m_3 z^{-2})/(1 - m_1 z^{-3})$ as shown in Fig. 4. In order to achieve

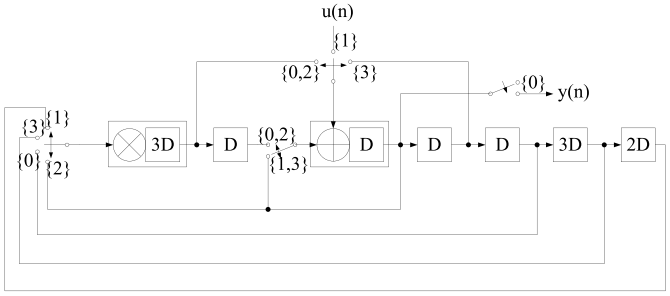


Fig. 2. Folded structure of the third-order IIR filter in Fig. 1. The switch instance i corresponds to clock cycle $4l + i$.

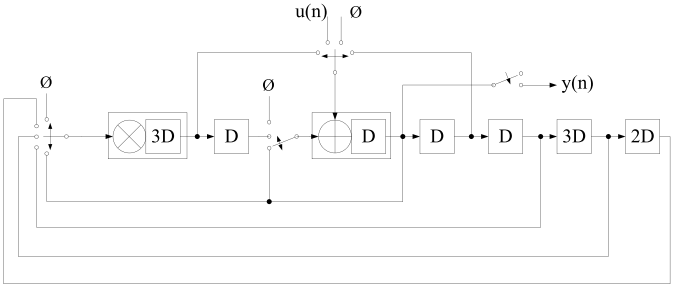


Fig. 3. Folded structure of the third-order IIR filter with invisible switches.

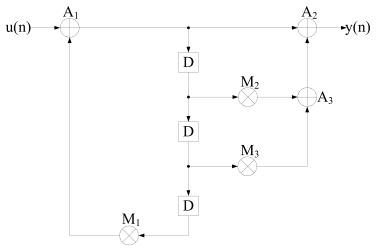


Fig. 4. Another third-order IIR filter.

obfuscation, an architecture can be designed to be configurable as a third-order IIR filter shown in either Fig. 1 or 4. These two modes are considered as meaningful modes.

In fact, a folded architecture of Fig. 4 using the following folding sets can be obtained by assigning different switch instances to the structure in Fig. 3, which is shown in Fig. 5

$$M = \{\emptyset, M_1, M_2, M_3\}$$
$$A = \{\emptyset, A_1, A_2, A_3\}.$$

The folding factor is 4, while there are only three multipliers and three adders in the DSP circuit. Therefore, if we consider the functionality of Fig. 4 as the desired mode, one computation cycle is wasted every four cycles. The latency will also be increased. Note that we could use clock gating techniques to reduce the power for the null operation cycles.

However, we can extend the periodicity of the switches to overcome the hardware underutilization. For instance, we can fold the third-order IIR filter in Fig. 4 by folding factor 3 with the folding sets

$$M = \{M_3, M_1, M_2\}$$
$$A = \{A_2, A_1, A_3\}.$$

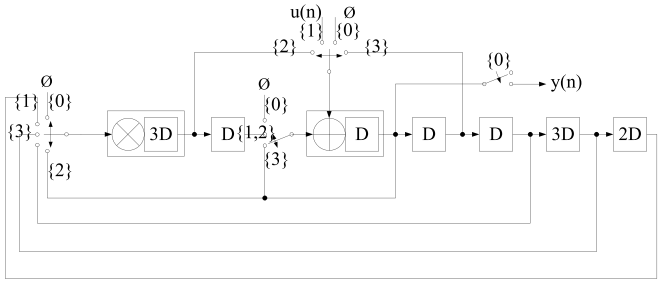


Fig. 5. Folded structure of the third-order IIR filter in Fig. 4. The switch instance i corresponds to clock cycle $4l + i$.

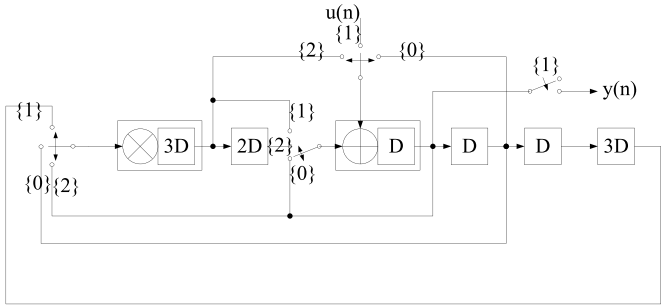


Fig. 6. Another folded structure of the third-order IIR filter in Fig. 4. The switch instance i corresponds to clock cycle $3l + i$.

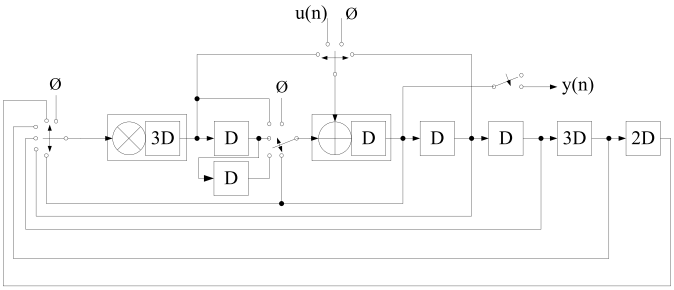


Fig. 7. Obfuscated structure, which can be configurable as a third-order IIR filter shown in either Fig. 1 or 4.

The folded structure is shown in Fig. 6. We can accommodate the two meaningful modes with no increase in latency for the second mode by extending the periodicity of the switches to the least common multiple of the folding factors of the two modes [i.e., lcm(3, 4) = 12]. Similar extensions of switch periods have been considered in design of digit-serial DSP architectures [28]. For example, switch instance $4l + i$ can be rewritten as $12l + i$, $12l + 4 + i$, and $12l + 8 + i$, for i ranging from 0 to 3, in Fig. 2; while switch instance $3l + i$ can be rewritten as $12l + i$, $12l + 3 + i$, $12l + 6 + i$, and $12l + 9 + i$, for i ranging from 0 to 2, in Fig. 6. As a result, for each meaningful mode as the desired mode, the latency remains the same as the original folded structure. This is achieved by increasing the complexity of the switch and the expense of hardware overhead associated with this step. The final obfuscated architecture for these two meaningful modes is shown in Fig. 7. The switch instances are obfuscated and the two correct configurations of the switches correspond to two meaningful modes.

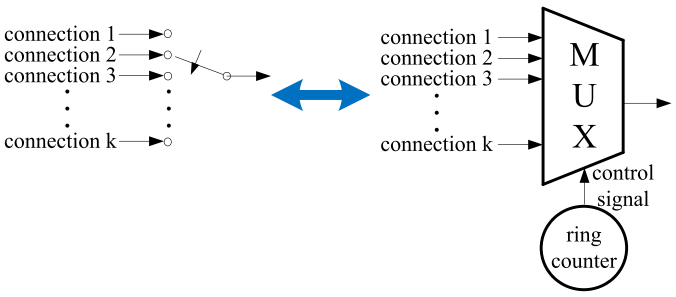


Fig. 8. Switch implementation.

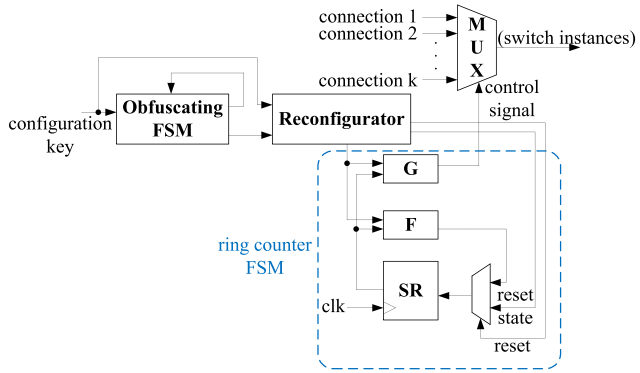


Fig. 9. Complete reconfigurable switch design.

III. Obfuscated Design via High-Level Transformations

A. Secure Switch Design

From Section II, it can be seen that the DSP circuits can be obfuscated via high-level transformations by appropriately designing the switches in a secure manner. The switches generated by high-level transformations are periodic N-to-1 switches. These switches can be implemented as multiplexers, whose control signals are obtained from ring counters (as shown in Fig. 8). Thus, the security of the switch relies upon design of the ring counters such that the outputs of the ring counters can be obfuscated.

A ring counter is often modeled as an FSM. An FSM is usually defined by a 6-tuple (I, O, S, S_0, F, G), where S is a finite set of internal states, I and O represent the inputs and outputs of the FSM, respectively, F is the next-state function, G is the output function, and S_0 is the initial state. However, unlike general FSMs, the FSM of a ring counter is input-independent, such that it always transits to the next state based on the current state. As a result, the control signal of the switches (i.e., output of the FSM) will be periodic.

B. Reconfigurable Switch Design

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 [12], [13], [29]. In order to achieve design obfuscation by using high-level transformations, we propose a reconfigurable switch design. The detailed implementation is shown in Fig. 9, where SR represents the state registers that store the information of the current state. We employ the idea of hardware design obfuscation as an activation sequence

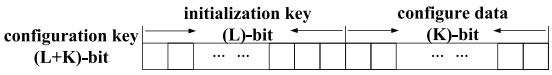


Fig. 10. Configuration key containing an initialization key and a configure data.

TABLE I
Switch Configurations

Mode	*Configure Data*
1	$data_1$, $data_2$
2	$data_3$
...	..., ...
M	$data_{\{2^K-1\}}$, $data_{\{2^K\}}$

required before configuration by inserting an obfuscating FSM. The FSM enables a reconfigurator that controls the functionality mode of the DSP circuit by configuring the output function G, next-state function F, and the initial state S_0. In our design, the configuration key must be known for the circuit to work properly, which consists of two parts: an L-bit initialization key and a K-bit configure data, as shown in Fig. 10. 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 cannot be processed by the reconfigurator as the reconfigurator is not activated. Note that other secure switch designs, whose detailed switch instances are hidden to the adversary, can also be adopted in the framework.

The number of possible variations of ring counters is limited by the length of the configure data, K. We can create M variation modes of the original circuits that have different functionalities, while $\log_2 M$ should be less than or equal to the length of configure data, K. Different configure data can be mapped into the same mode. An example of the mapping between the configure data and the associated modes is illustrated in Table I. Note that this only involves simple combinational logic synthesis.

IV. Generation of Variation Modes

A. Security Perspective of Variation Modes

The cost for an adversary to find the correct key of an obfuscated DSP circuit by adopting the proposed architecture is not only dependent on the length of the configuration key $(L + K)$, but also on the number of required input-vectors for learning the functionality of each variation mode (note that in this paper, the inputs represent the original inputs of the DSP circuits, while the key represents the data to control the switches). For a certain variation mode, the adversary attempts to attack the DSP circuit by generating input-vectors until the functionality could be discovered. The most intuitive way to mask the desired functionality is to modify the switch instances arbitrarily. However, the functionality of the resulting structure may not be meaningful from a signal processing point of view, which would be easier for the adversary to distinguish whether the circuit is operating correctly. The numbers of required input-vectors for these nonmeaningful variation modes would be less than the numbers of required input-vectors for meaningful variation modes. If one mode of a DSP circuit is obfuscated to output a large portion of

invalid values, the adversary can figure out this mode is nonmeaningful with a relatively small number of input-vectors.

The challenge of this problem is how to generate meaningful variation modes from a signal processing point of view such that the DSP circuit can also be operated in a reconfigurable manner. Most often, we perform a high-level transformation based on the design requirements and constraints by taking an existing design and generating the resulting design during high-level synthesis stage. We may continue to add variation modes into the design, as discussed in Section II. Some of the variation modes may correspond to meaningful modes (e.g., different order filters or filters of same order with different coefficients), which can be exploited for different applications, while others correspond to nonmeaningful modes.

Design of obfuscated DSP circuits requires extra efforts during the design phase. In fact, variations of the algorithm (e.g., different folding sets) can also be utilized to produce several obfuscated versions. Therefore, it is possible to generate meaningful variation modes simultaneously with the high-level transformation during design stage instead of modifying the switch instances after performing high-level transformations. As a result, the variations of the structures are indeed obtained from the variations of the selected algorithm. Furthermore, the secure switches can also be designed systematically based on the variations of algorithms. Using this approach, the extra design efforts can be reduced, while reconfigurable design with a number of different meaningful modes is ensured.

Note that meaningful modes are not mandatory for the proposed obfuscated design. While meaningful modes achieve higher security, obfuscation only with nonmeaningful modes could also attain considerable protection of the DSP circuit against reverse engineering.

B. 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 a very large number of existing high-level transformations in this paper. We just present an example in this paper to demonstrate how to generate variation modes. 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 of N operations inside each section to a hardware functional unit, as shown in Fig. 11. Two hierarchical folding algorithms are presented in [26], which include hierarchical interleaved folding (HIF) and hierarchical contiguous folding (HCF). In this paper, we only address HCF, while it is also applicable to HIF. The HCF transformation executes all operations of one section before starting execution of operations of next section. The reader is referred to [26] for further details.

The folding sets are described as follows:

$$\{X_0^0, X_1^0, \ldots, X_{N-1}^0, X_0^1, X_1^1, \ldots, X_{N-1}^1, \ldots, \ldots X_0^{M-1}, X_1^{M-1}, \ldots, X_{N-1}^{M-1}\}.$$

The operation X_i^j, i.e., the ith operation of jth block, will be executed at time $Nj + i$. The operations in a later section will be executed only after all the operations from the previous section are completed. The algorithm is described in Algorithm 1.

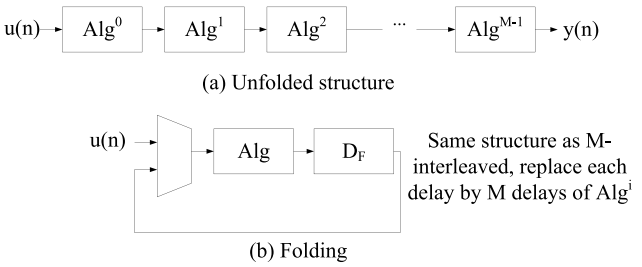


Fig. 11. (a) DSP data-flow graph containing M cascaded stages. Block Alg^i represents ith stage of the cascade. (b) Folded architecture where M stages are folded to same hardware. D_F represents the number of folded delays.

Algorithm 1 Hierarchical Contiguous Folding (HCF) Algorithm 1

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 s by s,$s+N$,$s+2N$,...$s+(M-1)N$.
3) Compute $D_F(Alg^j \xrightarrow{e} Alg^{j+1})$, for $j = 0, 1, 2...M-2$, and use these folded edges to replace the external inputs.

Algorithm 2 Design Obfuscation Algorithm based on the HCF Algorithm 2

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 s by s,$s+N$,$s+2N$,...$s+(l-1)N$, and set 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.

We propose an algorithm for obfuscation that generates 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 shown in Algorithm 2. (the total number of operations is still NM, where $M \geq l$).

If $l = M$, Algorithm 2 reduces to the HCF algorithm. From Algorithm 2, we can generate M meaningful modes correspond to $l = 1, 2, \ldots, M$. 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. Note that this algorithm can be easily extended to other types of DSP systems where the subcircuits are not directly connected.

V. 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. The detailed design flow is described below.

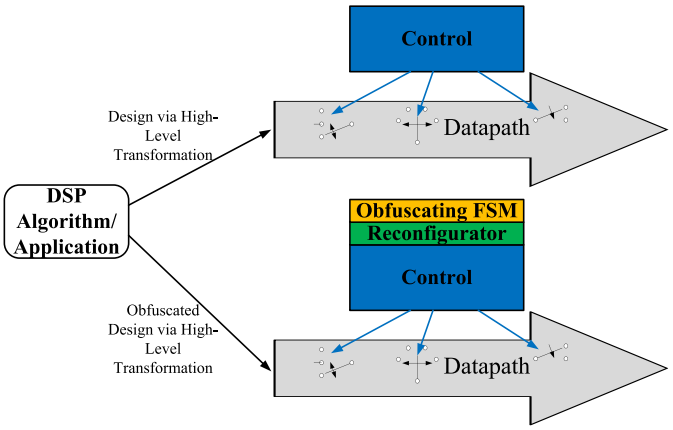


Fig. 12. Relationship between the obfuscated design and the original design.

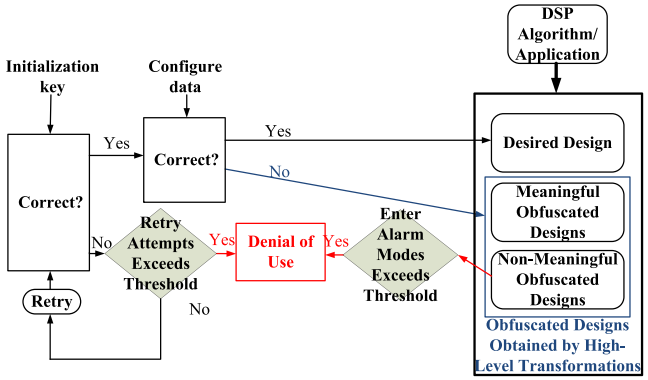


Fig. 13. Architecture of the proposed obfuscated DSP circuit.

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 Fig. 9. 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.

After these design steps, designer sends the obfuscated design to the foundry that manufactures the DSP circuit. By using the proposed design methodology, the manufacturer will not gain access to the desired functionality or the configuration key. Unauthorized copies of the obfuscated DSP circuits would provide little information to the adversary. The relationship between the obfuscated design and the original design via high-level transformation is shown in Fig. 12. The only difference between the obfuscated design and the original design via high-level transformation is the control of the DSP circuit. The main datapath is unaltered. As a result, the critical path would not increase for the obfuscated design. Furthermore, 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 Obfuscated DSP Circuits

The complete system of the proposed obfuscated DSP circuit is shown in Fig. 13. The reconfigurator will be enabled only by the correct initialization key. Only the correct configure data leads to the desired design. A wrong configure data activates an obfuscated mode (either a meaningful or nonmeaningful). The obfuscating FSM and a portion of nonmeaningful variation modes (i.e., we denote as alarm modes) can both be utilized for security check purpose. For example, some undesired modes in Table I can be designed as alarm modes by adding another output signal to the combinational logic. We can improve the security by mapping a larger number of configure data to this alarm mode, while keeping the portion of functional configure data to be relatively small. If the circuit continuously receives wrong initialization key or configure data whose number exceeds the predefined threshold, the adversary is prevented from further attempts of the configuration key by a denial of use block.

VI. Security and Resiliency Against Attacks

A. Attacks and Countermeasures

The goal of the proposed methodology is to ensure the designer's IP would not be stolen against reverse engineering. Generally speaking, a hacker trying to determine the functionality of a DSP circuit can resort to either of the following ways: 1) structural analysis of the netlist to identify and isolate the original design from the obfuscated design or 2) simulation-based reverse engineering to determine functionality of the design.

Our proposed obfuscation methodology protects the hardware against the first type of attack (i.e., structural analysis) from two perspectives: 1) structural obfuscation by high-level transformation and 2) integration with obfuscation modes. As presented in Section II, high-level transformations lead to structural obfuscation at the HDL level or gate-level netlist. Without knowing the correct configuration of the switches, it is hard for the adversary to learn the functionality of the original design. Furthermore, although the obfuscating FSMs could be isolated, the obfuscation of configuration switches cannot be separated from the original functionalities. Since the obfuscation variation modes are integrated to the reconfigurator in the synthesized DSP circuit, the adversary cannot remove the design obfuscation achieved by high-level transformations. In addition, meaningful variation modes also create ambiguity when the adversary performs the structural analysis attacks.

For a simulation-based approach where random key vectors are sequentially applied to take the circuit to the correct mode, the probability of discovering the configuration key sequence is $1/2^{L+K}$ for a circuit with a length-$(L+K)$ configuration key sequence. For example, the probability is only 5.4×10^{-20} for a circuit with a length $L=32$ initialization key sequence and a length $K=32$ configure data sequence.

In addition, the cost of the simulation-based attack approach is also dependent on the size of inputs. Various input patterns need to be tested to determine the complete functionality of a DSP circuit. In practice, most DSP circuits will have considerable size of inputs and the length of configuration keys can be made larger. Therefore, brute-force attack for learning the key sequences and input/output patterns is computationally infeasible.

B. Measure of Obfuscation Degree

1) Structural Obfuscation Degree: Manual attacks can be performed by visual inspection and structural analysis. In these types of manual attacks, the adversary has to analyze the RTL or gate-level structure as well as the layouts. This is a weak attack, as the adversary has very little chance of figuring out the obfuscation scheme for large DSP circuits.

The obfuscation degree of the structural obfuscation is dependent on the number of independent switches (N_s), the period of switch instances after high-level transformations (P), and the number of connections for each independent switch (C_m). To estimate the obfuscation degree against these manual attacks, we propose a metric called structural obfuscation degree (SOD)

$$\text{SOD} = \prod_{m=1}^{N_s} (C_m + 1)^P$$

where the additional 1 corresponds to the null operation. Note that a higher SOD value implies better obfuscation, as the SOD value indicates the number of possible functionalities generated by the combinations of these switches. A circuit is more secure, if there is more ambiguity in the structure. For example, the SOD value of the structure in Fig. 3 can be calculated as $3^4 \times 4^4 \times 5^4 = 12\,960\,000$ (excluding the output switch), which is already very large.

For a small subcircuit with a small SOD value, it may be feasible to figure out the original functionality from the obfuscated design, as the number of combinations is small and most of these combinations may result in nonmeaningful functionalities. However, the structure of a DSP circuit after high-level transformations usually has a larger SOD value (as illustrated by the example earlier). Thus, it will be extremely hard to distinguish the original functionality from other possible functionalities of these DSP circuits.

2) Functional Obfuscation Degree: The proposed methodology also achieves functional obfuscation by encrypting the correct functionality with a key sequence. The cost for an adversary to discover the correct key is dependent on the bit-length of the key ($L + K$), the number of meaningful modes (N_m), the number of nonmeaningful modes (N_n), and the size of input bits (N_I). To estimate the obfuscation degree against simulation-based attacks, we propose a metric denoted as functional obfuscation degree (FOD)

$$\text{FOD} = f(N_I)2^L[N_m + \alpha N_n + \beta(2^K - N_m - N_n)]$$

where $f(N_I)$ represents the input cost coefficient that is the number of input vectors required for learning the functionality of the DSP circuit, which is proportional to N_I, and α, β represent the cost coefficients of learning a new nonmeaningful variation mode and learning a previously known mode compared with learning a new meaningful mode, respectively. These coefficients are dependent on the particular applications.

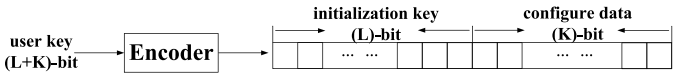


Fig. 14. Key encoding.

As discussed in Section IV-A, it can be expected that they would follow the relation that $0 < \beta < \alpha < 1$. Future work will be directed toward validating the values of α and β through experimental results. The higher the FOD value, more secure the obfuscated design. Note that in practice, the computation cycles and the clock periods of the DSP circuits would also affect the cost of simulation-based attacks.

C. Improving the Security by Key Encoding

In the proposed obfuscation scheme, the key consists of two parts: 1) initialization key and 2) configure data. However, in the scenario that the adversary has found a key that can successfully pass the initialization but is still in an incorrect configuration, the adversary will only try different configure data while fixing the initialization key. This could weaken the scheme. An encoder could be added to improve the security of the system, as shown in Fig. 14. The encoder could be a Hash function, a linear feedback shift register, or a PUF. By incorporating the encoder, the user key and the configuration key are no longer bit-to-bit mapped.

In addition, if we use a PUF as the encoder, key collisions could also be avoided in different chips. The PUFs can be used to give unique user keys for different DSP circuits even though they are all obfuscated with the same configuration key.

D. Security Properties

The main objective of this paper is to protect DSP circuits against reverse engineering. The obfuscated DSP circuits will only operate in the desired mode with a negligible probability that others would be able to find. Thus, the correct functionality is hidden to the adversary even when the adversary can access the DSP circuits. In addition, the proposed obfuscating scheme also satisfies a set of following properties to ensure security and resiliency against attacks.

1) *Unobtrusiveness:* The obfuscation is invisible to the functional DSP circuits. Its presence would not interfere with regular operation of the design.
2) *Unambiguity:* The probability of finding the correct key for a DSP circuit is low by employing our proposed obfuscation scheme. The chance for an adversary to enter a DSP circuit into the correct mode by random guessing is $1/2^{L+K}$, which will be negligible when the bit-length of the key is long. Therefore, the correct key is a strong proof of ownership.
3) *Robustness:* Since the obfuscation modes are generated along with the high-level transformation design phase, all the stages after this phase in the high-level synthesis flow would contain the obfuscation. The embedded obfuscation is extremely difficult to remove, since the variation modes are integrated to the switches. In addition, we could combine PUFs to ensure that the keys of different DSP circuit designers would not collide.
4) *Universality:* The proposed obfuscating methodology can be used for all common DSP designs. In this paper, we have only described a few examples of high-level transformations for hardware obfuscation. However, other types of high-level transformations and other

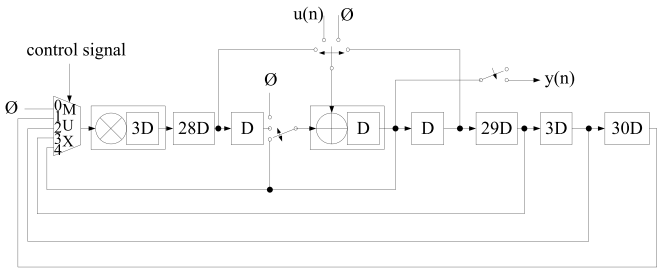


Fig. 15. Obfuscated design of the original 18th-order IIR filter.

transformation algorithms can also be used to achieve hardware obfuscation.

VII. Evaluation of the Proposed Methodology

A. Overhead Impact

Component overhead of the proposed obfuscation design includes: 1) additional control logic of switches; 2) reconfigurator; and 3) obfuscating FSM. These additional circuits only affect the switches of an obfuscated DSP circuit, while the main datapath stays the same as the original design, as shown in Fig. 12. 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 three MACs.

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 third-order IIR filter as shown in Fig. 1. We assume the desired functionality is an 18th-order IIR filter realized as a cascade of six third-order IIR filters. 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 eight meaningful variation modes, the parameters $M = 8$ and $N = 4$ are used to the structure with six sections of third-order IIR filter (i.e., the original 18th-order IIR filter) and two additional sections of null operations. The switch instances of this folded design are periodic with period 32. The eight meaningful modes correspond to $(3l)$th-order IIR filter where $l = 1, 2, \ldots, 8$, respectively. Eight nonmeaningful variation modes are also incorporated. Each secure switch is controlled by the reconfigurator independently. Fig. 15 shows an example of the switch connected to the input of the multiplier in the obfuscated design. This switch has five possible input paths, as the null operations are also integrated to the switches. Based on the algorithm, the control signals of this switch within one period for the intended mode should be

$$(3, 1, 4, 2, \quad 3, 1, 4, 2, \quad 3, 1, 4, 2, \quad 3, 1, 4, 2,$$
$$3, 1, 4, 2, \quad 3, 1, 4, 2, \quad 0, 0, 0, 0, \quad 0, 0, 0, 0).$$

For other generated meaningful variation modes whose functionalities are $(3l)$th-order IIR filters, the control signals should be periodic with a length-32 sequence that consists of l $(3, 4, 1, 2)$ in the beginning and $8 - l$ $(0, 0, 0, 0)$ in the end.

TABLE II
Switch Configurations Example

Mode	*Configure Data*	Functionality
1	0000	3rd-order IIR filter
2	0001	6th-order IIR filter
3	0010	9th-order IIR filter
4	0011	12th-order IIR filter
5	0100	15th-order IIR filter
6	0101	18th-order IIR filter
7	0110	21st-order IIR filter
8	0111	24th-order IIR filter
9	1000	non-meaningful
10	1001	non-meaningful
11	1010	non-meaningful
12	1011	non-meaningful
13	1100	non-meaningful
14	1101	non-meaningful
15	1110	non-meaningful
16	1111	non-meaningful

TABLE III
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

In our experiment, eight nonmeaningful variation modes are also incorporated. As a result, the bit-length of the configure data is at least four, since the number of variation modes $N_m + N_n$ should be less than 2^K. For instance, a simple design example of the reconfigurator with configure data $K = 4$ is presented in Table II. Note that multiple configure data can be mapped to the same mode, if we increase K.

An obfuscating FSM is also added into the secure switch design to provide the second-level protection of the obfuscated DSP circuit. The number of the states of the obfuscating FSM should be less than or equal to 2^L, where L is the length of initialization key.

1) Area Overhead: We present the area overhead for the two DSP circuit benchmarks as shown in Tables III and IV, respectively. Note that the overhead percentages presented in Tables III and IV are computed based on the folded designs instead of the original circuits. The results include average 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.

It can be seen from Tables III and IV 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. However, a strong obfuscation is achieved,

TABLE IV
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

as the chance for an adversary to enter the DSP circuit into the desired mode is only $1/2^{L+K} = 1/2^{128} = 2.94 \times 10^{-39}$. 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 under the assumption that we want to create a certain degree of obfuscation (i.e., maintain an approximately same number of switches to obfuscate, even though there are more switches).

In addition, when we compare the effects between L and K, it can be seen that the overhead increases more significantly with the increase of L. For the (3l)th-order IIR filter example, the overhead is 3.8% when $L = 4$ and $K = 4$. If we fix $K = 4$, the overhead is 15.4% when $L = 64$, which is increased by 305%. However, if we fix $L = 4$, the overhead is only increased by 55% when $K = 64$. 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. Indeed, this is another advantage of our proposed methodology, as obfuscating DSP circuits through secure switches incur smaller overhead, compared with the methods only based on obfuscating FSMs.

2) Timing Overhead: As the main datapath in the obfuscated design stays unaltered as the original design via high-level transformations, the critical path will not be increased. The obfuscating FSM and the reconfigurator can be pipelined such that the critical path is only dependent on the main datapath.

There can be a degradation of the performance with respect to latency if null operations are inserted into the obfuscated design. However, obfuscation does not require introduction of null operations. The parameters of an obfuscated design should be selected based on the application requirements and constraints. In addition, at the expense of the control complexity, the latency of the obfuscated design can be guaranteed to be the same as the original design by expanding the periodicity of the switches as discussed in Section II.

3) Power and Energy Overhead: We present the power consumptions for different meaningful modes of the (3l)th-order IIR filter benchmark in Fig. 16.

When comparing the obfuscated design to the original folded design without these additional circuits (i.e., folded design of the 18th-order IIR filter), the power is only slightly increased by 1.1%. As a result, if we compare the obfuscated design running in the desired mode and the folded design, the power overhead is 3.9%. In addition, if an obfuscated DSP circuit is designed with no increase of latency as presented in Section II, the energy overhead will also be small.

4) Number of Meaningful Modes: As we pointed out above, the number of modes is bounded by 2^K. In addition, if we apply the design obfuscation algorithm based on the HCF

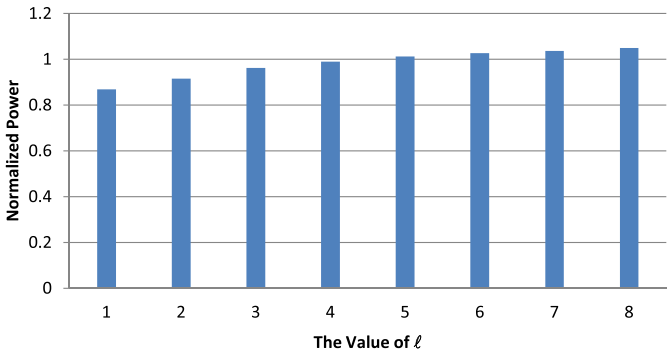


Fig. 16. Normalized power consumption for different meaningful modes (%) of the (3l)th-order IIR filter benchmark (normalized to the power consumption when considering l is a variable).

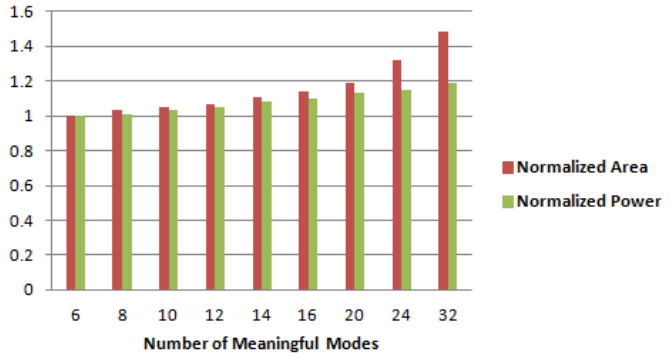


Fig. 17. Normalized area and power cost for different numbers of meaningful modes.

algorithm, the number of meaningful modes is also limited by the selected number of sections of the DSP circuits. For instance, we could only implement at most eight meaningful modes by choosing the parameter $M = 8$ in the (3l)th-order IIR filter benchmark example. In general, the maximum number of meaningful modes is dependent on the number of subcircuits and the number of inserted null sections.

Therefore, in order to increase the number of meaningful modes, we need to add more null sections. We still use the 18th-order IIR filter as an example. If we fold the 18th-order IIR filter directly without any null section, we could only implement six meaningful variation modes in the obfuscated design, i.e., third-order, sixth-order, ninth-order, twelfth-order, fifteenth-order, and eighteenth-order IIR filter. If we want to further increase the number of meaningful modes, we have to insert null operations into the original circuit, but at the price of additional power consumption and computation cycles. Consequently, the area and power of the folded design will also increase. Note that the length of configure data K should be always greater than or equal to $\log_2(N_m + N_n)$. In our experiments, we set $K = 16$ to ensure all the possible meaningful modes can be realized. The measurements of the area and power for the (3l)th-order IIR filter benchmark are shown in Fig. 17.

From Fig. 17, it can be seen that area and power do not increase very significantly with the increase of the number of meaningful modes (e.g., for the obfuscated circuit with 32 meaningful modes, the area and the power area

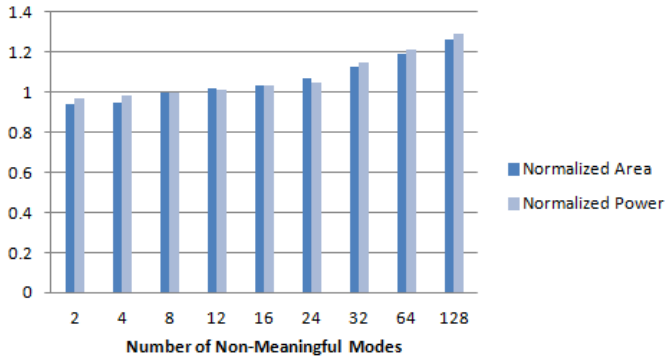


Fig. 18. Normalized area and power cost for different numbers of nonmeaningful modes.

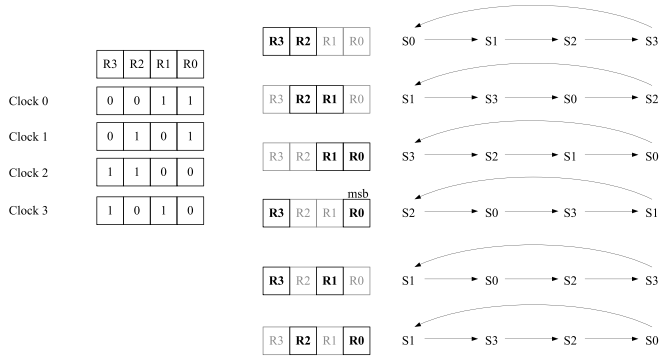


Fig. 19. State registers sharing.

are increased by 48% and 19%, respectively, compared with the example with six meaningful modes). Therefore, in the proposed methodology, we can increase the number of meaningful modes to improve the security while maintaining relatively low power and area overheads.

5) Number of Nonmeaningful Modes: As discussed previously, if the number of meaningful modes increases to a value that the total number of modes is greater than 2^K, we need to increase the length of configure data as well to realize all the possible meaningful variation modes. In this scenario, however, there is another feasible solution that is to reduce the number of nonmeaningful modes, if we want to maintain the same length of configure data.

We present the experimental results of the normalized area and power for different numbers of nonmeaningful modes in Fig. 18. All of the results are based on the $(3l)$th-order IIR filter benchmark with eight meaningful modes and $K = 16$.

Furthermore, since the latency is not affected by the obfuscating FSM and the reconfigurator, the performance of energy has the same trends as the power with the increase in number of variation modes and key length.

B. Discussion

Note that all the experimental results presented previously are only based on the design obfuscation algorithm based on the HCF algorithm for the two particular benchmarks. These results are just aimed to provide an example of the performance for the proposed methodology. Null operations are inserted in this example. However, as discussed in Section II, the obfuscated circuit can also be designed to maintain the same latency as the original structure obtained by performing certain high-level transformation based on the application requirements and constraints. In this case, the overheads of timing, power, and energy would be minimal, which is suitable for the application where diminished runtime performance is not acceptable.

The actual performance of an obfuscated DSP circuit may vary significantly according to multiple design parameters, which include the target DSP algorithm/application, the specific variation of the algorithm, the relation between the desired mode and the obfuscated design, the number of modified switches, the key length, the numbers of meaningful and nonmeaningful modes, and so forth. In the design of obfuscated DSP circuits, the designer should carefully select these parameters according to the specific application requirements and constraints.

C. Overhead Reduction

State register sharing is one possible approach to reduce the area consumption by managing the relation among different switches in a subcircuit. For example, as shown in Fig. 19, four flip-flops can be used to implement six four-state ring counters, in contrast to 12 flip-flops if no sharing of state registers is exploited. The saving could be significant for a large circuit. In Fig. 19, the registers marked in bold represent the active state registers for corresponding ring counters.

Furthermore, exploiting the relation among the state registers is more effective than managing the output function or the next-state function to control the relations among different switches in a subcircuit. However, note that by using these design optimization techniques, the SOD will be degraded, as the number of independent switches (N_s) is decreased.

VIII. Comparison With Existing Obfuscation Methods

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. Therefore, we have introduced two metrics to analyze the security, which are discussed in Section VI.

Most of the hardware obfuscation techniques in this paper 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. In addition, other circuit locking techniques only achieve protection at one-level (i.e., encrypt the normal functionality by a key), while our proposed methodology provides a two-level protection (i.e., structural obfuscation and functional obfuscation). 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. Other existing methods, such as [6], [7], are not specific to DSP circuits, which would not be able to ensure meaningful variation modes from a signal processing point of view. In addition, meaningful variation

modes enable our proposed design methodology to be adaptable to reconfigurable applications.

Finally, when considering the metrics of the design performance, our proposed methodology is also superior. While our proposed approach only alters the logic of switches, most of the existing methods are based on explicit FSM modifications (e.g., the technique proposed in [13]), which are not scalable since the construction of the FSM is not practical for even moderate-sized circuits, not to mention that the number of added obfuscation states can be relatively large as compared with the original FSM. In our proposed methodology, area consumption is slightly increased due to the increased cost of the control logic for the obfuscated switches.

IX. Conclusion

This paper presents a novel low-overhead solution to design DSP circuits that are obfuscated both structurally and functionally by utilizing high-level transformation techniques. It is shown that verifying the equivalence of DSP circuits by employing high-level transformations will be harder if some switches can be designed in such a way that are difficult to trace. A secure reconfigurable switch design is incorporated into the proposed design scheme to improve the security. A complete design flow is presented. In the proposed obfuscation methodology, the variation modes and the additional obfuscating circuits could also be designed systematically based on the high-level transformations. Compared with other existing obfuscation methods, another 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 correct functionality from other variation modes. Experimental results have demonstrated the effectiveness of the proposed methodology.

This paper, for the first time, considers the security perspective of high-level transformations. Future work will explore the algorithmic aspect of different high-level transformations for design obfuscation. Ongoing work includes the validation of the security performances of meaningful modes and nonmeaningful modes. We are also interested in addressing the attack methods of DSP circuits. We intend to exploit the security perspective of the proposed methodology by performing various attacks to the obfuscated DSP circuits. Future work will be directed toward developing a complete design flow which can generate the target structure and obfuscation variation modes automatically based on the specific application performance requirement. The ultimate goal is to develop an electronic design automation synthesis tool which can incorporate large number of design obfuscation algorithms based on high-level transformations for DSP system design.

References

[1] J. Guajardo, S. S. Kumar, G.-J. Schrijen, and P. Tuyls, "Brand and IP protection with physical unclonable functions," in *Proc. IEEE Int. Symp. Circuits Syst.*, May 2008, pp. 3186–3189.

[2] G. E. Suh and S. Devadas, "Physical unclonable functions for device authentication and secret key generation," in *Proc. 44th Annu. Design Autom. Conf.*, Jun. 2007, pp. 9–14.

[3] A. L. Oliveira, "Techniques for the creation of digital watermarks in sequential circuit designs," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 20, no. 9, pp. 1101–1117, Sep. 2001.

[4] D. Kirovski, Y.-Y. Hwang, M. Potkonjak, and J. Cong, "Intellectual property protection by watermarking combinational logic synthesis solutions," in *Proc. Int. Conf. Comput.-Aided Design*, Nov. 1998, pp. 194–198.

[5] A. B. Kahng *et al.*, "Watermarking techniques for intellectual property protection," in *Proc. 35th Annu. Design Autom. Conf.*, Jun. 1998, pp. 776–781.

[6] F. Koushanfar and Y. Alkabani, "Provably secure obfuscation of diverse watermarks for sequential circuits," in *Proc. Int. Symp. Hardw.-Oriented Security Trust*, Jun. 2010, pp. 42–47.

[7] J. A. Roy, F. Koushanfar, and I. L. Markov, "EPIC: Ending piracy of integrated circuits," in *Proc. Conf. Design, Autom. Test Eur.*, Mar. 2008, pp. 1069–1074.

[8] W. P. Griffin, A. Raghunathan, and K. Roy, "CLIP: Circuit level IC protection through direct injection of process variations," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 20, no. 5, pp. 791–803, May 2012.

[9] Y. M. Alkabani and F. Koushanfar, "Active hardware metering for intellectual property protection and security," in *Proc. USENIX Security Symp.*, Aug. 2007, pp. 291–306.

[10] T. Batra. (2005). *Methodology for Protection and Licensing of HDL IP* [Online]. Available: http://www.design-reuse.com/articles/12745

[11] R. S. Chakraborty and S. Bhunia, "Hardware protection and authentication through netlist level obfuscation," in *Proc. Int. Conf. Comput.-Aided Design*, Nov. 2008, pp. 674–677.

[12] R. S. Chakraborty and S. Bhunia, "RTL hardware IP protection using key-based control and data flow obfuscation," in *Proc. 23rd Int. Conf. VLSI Design*, Jan. 2010, pp. 405–410.

[13] R. S. Chakraborty and S. Bhunia, "HARPOON: An obfuscation-based SoC design methodology for hardware protection," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 28, no. 10, pp. 1493–1502, Oct. 2009.

[14] Y. Lao and K. K. Parhi, "Protecting DSP circuits through obfuscation," in *Proc. IEEE Int. Symp. Circuits Syst.*, Jun. 2014.

[15] K. K. Parhi, "Algorithm transformation techniques for concurrent processors," *Proc. IEEE*, vol. 77, no. 12, pp. 1879–1895, Dec. 1989.

[16] K. K. Parhi and D. G. Messerschmitt, "Pipeline interleaving and parallelism in recursive digital filters. I. Pipelining using scattered look-ahead and decomposition," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 37, no. 7, pp. 1099–1117, Jul. 1989.

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

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

[19] K. K. Parhi, "Pipelining in algorithms with quantizer loops," *IEEE Trans. Circuits Syst.*, vol. 38, no. 7, pp. 745–754, Jul. 1991.

[20] K. K. Parhi, "Low-energy CSMT carry generators and binary adders," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 7, no. 4, pp. 450–462, Dec. 1999.

[21] K. K. Parhi, "Design of multigigabit multiplexer-loop-based decision feedback equalizers," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 13, no. 4, pp. 489–493, Apr. 2005.

[22] N. R. Shanbhag and K. K. Parhi, "Relaxed look-ahead pipelined LMS adaptive filters and their application to ADPCM coder," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 40, no. 12, pp. 753–766, Dec. 1993.

[23] J. Ma, K. K. Parhi, and E. F. Deprettere, "Annihilation-reordering look-ahead pipelined CORDIC-based RLS adaptive filters and their application to adaptive beamforming," *IEEE Trans. Signal Process.*, vol. 48, no. 8, pp. 2414–2431, Aug. 2000.

[24] C.-Y. Wang and K. K. Parhi, "High-level DSP synthesis using concurrent transformations, scheduling, and allocation," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 14, no. 3, pp. 274–295, Mar. 1995.

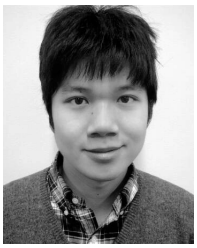
[25] K. K. Parhi, *VLSI Digital Signal Processing Systems: Design and Implementation*. New York, NY, USA: Wiley, 1999.

[26] K. K. Parhi, "Hierarchical folding and synthesis of iterative data flow graphs," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 60, no. 9, pp. 597–601, Sep. 2013.

[27] K. K. Parhi, "Verifying equivalence of digital signal processing circuits," in *Proc. 46th Asilomar Conf. Signals, Syst. Comput.*, Nov. 2012, pp. 99–103.

[28] K. K. Parhi, "A systematic approach for design of digit-serial signal processing architectures," *IEEE Trans. Circuits Syst.*, vol. 38, no. 4, pp. 358–375, Apr. 1991.

[29] Y. Alkabani, F. Koushanfar, and M. Potkonjak, "Remote activation of ICs for piracy prevention and digital right management," in *Proc. Int. Conf. Comput.-Aided Design*, Nov. 2007, pp. 674–677.



Yingjie Lao (S'09) received the B.S. degree in information engineering from Zhejiang University, Hangzhou, China, in 2009. He is currently working toward the Ph.D. degree at the Department of Electrical and Computer Engineering, University of Minnesota–Twin Cities, Minneapolis, MN, USA.

His current research interests include hardware security, digital signal processing, and reliable and fault-tolerant VLSI architectures.

Keshab K. Parhi (S'85–M'88–SM'91–F'96) received the B.Tech. degree from IIT Kharagpur, Kharagpur, India, in 1982, the M.S.E.E. degree from the University of Pennsylvania, Philadelphia, PA, USA, in 1984, and the Ph.D. degree from the University of California at Berkeley, Berkeley, CA, USA, in 1988.

He has been with the University of Minnesota, Minneapolis, MN, USA, since 1988, where he is currently a Distinguished McKnight University Professor and an Edgar F. Johnson Professor with the Department of Electrical and Computer Engineering. He has published over 500 papers, has authored the textbook *VLSI Digital Signal Processing Systems* (New York, NY, USA: Wiley, 1999), and co-edited the reference book *Digital Signal Processing for Multimedia Systems* (New York, NY, USA: Marcel Dekker, 1999). His current research interests include the VLSI architecture design and implementation of signal processing, communications and biomedical systems, error control coders and cryptography architectures, high-speed transceivers, stochastic computing, secure computing, and molecular computing. He is also currently working on intelligent classification of biomedical signals and images, for applications such as seizure prediction and detection, schizophrenia classification, biomarkers for mental disorder, brain connectivity, and diabetic retinopathy screening.

Dr. Parhi has served on the Editorial Boards of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS PART I AND PART II, *VLSI Systems*, *Signal Processing*, the IEEE SIGNAL PROCESSING LETTERS, and the IEEE SIGNAL PROCESSING MAGAZINE, and served as the Editor-in-Chief of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS PART I from 2004 to 2005. He currently serves on the Editorial Board of the *Springer Journal of Signal Processing Systems*. He has served as the Technical Program Co-Chair of the 1995 IEEE VLSI Signal Processing Workshop and the 1996 Application-Specific Systems, Architectures, and Processors Conference, and as the General Chair of the 2002 IEEE Workshop on Signal Processing Systems. He was a Distinguished Lecturer of the IEEE Circuits and Systems Society from 1996 to 1998. He served as a Board of Governors Elected Member of the IEEE Circuits and Systems Society from 2005 to 2007. He was a recipient of numerous awards, including the 2013 Distinguished Alumnus Award from IIT Kharagpur, the 2013 Graduate/Professional Teaching Award from the University of Minnesota, the 2012 Charles A. Desoer Technical Achievement Award from the IEEE Circuits and Systems Society, the 2004 F. E. Terman Award from the American Society of Engineering Education, the 2003 IEEE Kiyo Tomiyasu Technical Field Award, the 2001 IEEE W. R. G. Baker Prize Paper Award, and the Golden Jubilee Medal from the IEEE Circuits and Systems Society in 2000.