Mode-based Obfuscation using Control-Flow Modifications

Sandhya Koteswara
University of Minnesota,
Twin Cities
200 Union Street SE,
Minneapolis, MN 55455, USA
kotes001@umn.edu

Chris H. Kim
University of Minnesota,
Twin Cities
200 Union Street SE,
Minneapolis, MN 55455, USA
chriskim@umn.edu

Keshab K. Parhi
University of Minnesota,
Twin Cities
200 Union Street SE,
Minneapolis, MN 55455, USA
parhi@umn.edu

ABSTRACT

Hardware security has emerged as an important topic in the wake of increasing threats on integrated circuits which include reverse engineering, intellectual property (IP) piracy and overbuilding. This paper explores obfuscation of circuits as a hardware security measure and specifically targets digital signal processing (DSP) circuits which are part of most modern systems. The idea of using desired and undesired *modes* to design obfuscated DSP functions is illustrated using the fast Fourier transform (FFT) as an example. The selection of a mode is dependent on a *key* input to the circuit. The system is said to work in its desired mode of operation only if the correct key is applied. Other undesired modes are built into the design to confuse an adversary. The approach to obfuscating the design involves *control-flow* modifications which alter the computations from the desired mode. We present simulation and synthesis results on a reconfigurable, 2-parallel FFT and discuss the security of this approach. It is shown that the proposed approach results in a *reconfigurable* and flexible design at an area overhead of 8% and a power overhead of 10%.

CCS Concepts

•Security and privacy → Hardware reverse engineering; •Hardware → Application specific integrated circuits;

Keywords

Design obfuscation, fast Fourier transforms (FFT), Control-flow modification, Mode-based obfuscation

1. INTRODUCTION

As more design houses are moving away from fabrication, the supply chain for semiconductor integrated circuits is getting more distributed. This has made control over the quality and security at different levels of manufacturing difficult. Attacks on hardware such as hardware trojans, intellectual

property (IP) piracy, overbuilding of integrated circuits (IC), reverse engineering, side channel attacks and counterfeiting can alter device functionality, lead to loss of jobs and even catastrophic situations including loss of human lives [1]; this has led to a high level of importance on hardware security. Hardware obfuscation, which involves hiding the functionality of a design, helps in protection against many of the above mentioned attacks. Therefore, we address the task of hardware obfuscation in this paper.

Several approaches to hardware obfuscation have been proposed in literature including passive methods such as change of code written in a hardware description language (HDL) [2], active methods such as insertion of gates at register transfer level (RTL) [3], combinational logic modifications [4] and netlist-level obfuscation [5]. However, none of the above methods specifically addresses the special challenge of obfuscation of DSP circuits. An approach to hardware obfuscation using high-level transformations has been proposed in [6]. An implementation of this obfuscation technique using counter resetting on a Radix-2 real FFT was shown in [7]. However, no systematic approach to control-flow modifications was presented in this paper.

The motivation behind obfuscating DSP circuits is two fold. First, DSP circuits are highly control driven and hence obscuring the control-flow of such designs is critical. Also, DSP circuits are defined by specific properties such as number of taps in a filter, length of the FFT etc. which dictate important constraints such as performance, area and power. Hence, hiding this type of information is another goal of obfuscation. The approach to obfuscation proposed in [6] and used here consists of using a mode-based method where the entire design can operate in meaningful and non-meaningful modes. A *key* to the system decides its mode of operation by selecting the configuration of the datapath and control-flow output from the controlpath. A block diagram to illustrate the approach is shown in Figure 1. Additional modes that leak partially incorrect information are also added to aid obfuscation.

The obfuscation scheme protects the circuit in two ways. Locking the entire design using keys prevents illegal piracy and overproduction, since the keys are kept secret with the design house and only programmed into the chip after fabrication. The foundry has no access to the key, rendering the circuit useless even if multiple copies are made. The scheme also achieves a second objective by making the chips harder to reverse engineer. The control-flow of the design is well hidden and concept of multiple modes in the design prevents competitors who are trying to make use of the design

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

CS2 '16, January 20 2016, Prague, Czech Republic

© 2016 ACM. ISBN 978-1-4503-4065-6/16/01...\$15.00

DOI: <http://dx.doi.org/10.1145/2858930.2858934>

Figure 1: Design flow of mode-based obfuscation using Control-flow modifications

aspects to lower the cost or power of the circuits.

This paper presents an obfuscated design of an FFT that can be configured as a 16/64/256/1024-point FFT working in 2-parallel mode. These functional modes are the meaningful modes of the design. We create non-meaningful modes and the method of selection of these modes is addressed in this paper. A main contribution of this paper is a novel method of modifying the control-flow for obfuscation with low overhead. This paper also introduces the notion of modes that compute partially correct outputs. Security analysis of the design with respect to the contribution of the designed meaningful and non-meaningful modes is presented highlighting the importance of this scheme in securing DSP circuits.

The rest of the paper is organized as follows. Section 2 presents a brief overview of the obfuscation methodology applied to FFT circuits and the design flow. An implementation of the design using a 1024-point folded FFT is presented in Section 3. An analysis of the security of the design considering various obfuscated modes is carried out in Section 4. We present overheads in Section 5 with respect to area and power. Finally, we conclude the paper with a summary and discussion on possible future work.

2. OBFUSCATION METHODOLOGY

In this section, we discuss the basic techniques used to create a mode-based obfuscated design. These methods exploit popularly used transformations and modifiable properties of DSP circuits to affect the outputs in different ways. We use a simple FFT circuit to demonstrate these ideas.

2.1 Folding transformation on FFT circuits

Time-multiplexed architectures can be designed by a high-level transformation referred to as *folding* [8, 9]. For a folding factor N , N algorithm operations can be executed by the same hardware functional units in N clock cycles. The inputs to the functional units in a folded or time-multiplexed circuit are selected by control signals. Altering these control signals alters the functionality of the circuit.

Pipelined, parallel complex FFT architectures can be designed using folding as described in [10]. It is seen that for an FFT of size N , a folding factor of $N/2$ leads to a 2-parallel architecture. To illustrate folding, we consider a 16-point, radix- 2^2 , decimation-in-frequency (DIF) complex FFT circuit shown in Figure 2. Appropriate selection of folding sets results in a 2-parallel structure for this FFT as shown in Figure 3. The folded architecture consists of major blocks such as two-point butterfly (BFI), two-point butterfly which also performs trivial multiplication (by $-j$) and twiddle factor multiplication (BFII), and delay-switch-

Figure 2: 16 point Radix- 2^2 complex DIF FFT flow-graph

Figure 3: 16-point, 2-parallel folded FFT architecture

delay of size n (nDS) structures which are shown in Figure 4. Reconfigurability of the design can be achieved by

Figure 4: Major blocks of the folded FFT architecture

selecting various folding sets and generating structures of 2-parallel operation. We term these folding sets to be *meaningful modes* of the design and a combination of these will be used to create a complete obfuscated system.

2.2 Control-flow changes for obfuscation

After applying folding and obtaining appropriate folded FFT architectures, we can exploit the properties of control flow of folded FFT architectures to generate functionally incorrect modes. We term these *non-meaningful modes*, and show how modifications to signals generated by the control path of the architecture could be used to create them.

For the folded FFT components in Figure 4, the control

inputs t (which corresponds to multiplication by $-j$) of BFII and s of switches are specified by the control path. The delay-switch-delay structures operate correctly only if the appropriate control signals derived from a $\log_2 N$ -bit counter are used. Incorrect patterns applied to these delay-switch-delay structures produce modes which generate random outputs. The randomness of the outputs increases with increase in the number of delay-switch-delay structures with wrong control signals. The correct operation of these structures and the effect of the changes are illustrated in Figure 5.

Figure 5: Modification of control logic of delay-switch-delay structure (input s)

Next, from the flowgraph of Figure 2, it is observed that a radix-2² FFT has alternate columns of twiddle factors and multiplication by $-j$ terms. In the folded architecture, these changes can be embedded in the BF II structure shown in Figure 4. From the folding sets chosen, the scheduling order of each stage can be found. Using this information, we can identify that changes to twiddle factors or $-j$ multiplication will only alter a subset of the outputs. This is an important observation as it leads to generation of non-meaningful modes with partially correct outputs. Such modes provide partial information about the functionality of the system. The role of these modes will be discussed in later sections.

As an example, consider the FFT flowgraph shown in Figure 6. Suppose the system is working in the 2-parallel

Figure 6: Effect of modification of $-j$ multiplication factors on output

mode, the outputs are obtained in the order $X(0)/X(8)$, $X(2)/X(10)$, $X(1)/X(9)$, $X(3)/X(11)$, $X(4)/X(12)$, $X(6)$

$/X(14)$, $X(5)/X(13)$, $X(7)/X(15)$. Now if we modify the first column of $-j$ multiplications such that only the top 50% are correct values, we see that the outputs $X(0)/X(8)$, $X(4)/X(12)$, $X(2)/X(10)$ and $X(6)/X(14)$ are correct and the rest are incorrect. Such a modification corresponds directly to changing the control signal t of the first BFII block of the obfuscated FFT architecture. The details of change in operations that occur for a single block are shown in Figure 7.

Figure 7: Modification of control logic of BFII (input t)

Thus, we have created a partial mode with 50% correct outputs. Various such combinations can be used to generate non-meaningful modes which correctly compute 25% and 50% of the outputs.

3. COMPLETE DESIGN

We now discuss usage of concept of folding and control-flow modifications to create a complete obfuscated design. Since these approaches can be applied in various ways, we can create a reconfigurable and flexible design which can be tailored according to the level of security desired and the acceptable overhead.

3.1 Design of meaningful modes

We make use of the concept of folding on radix-2² FFT flowgraphs to generate two-parallel FFT architectures of varying lengths. For example, we could design a system which can implement four architectures such as 16-point/64-point/256-point and 1024-point FFTs. Various folding sets can be used for this purpose. One of the four architectures obtained as a result of folding is shown in Figure 8.

Figure 8: 256-point, 2-parallel folded FFT architecture

We now proceed to integrate the four architectures to one

Figure 9: Obfuscated FFT with four meaningful modes built on 1024-point, 2-parallel folded FFT

structure which can be programmed to operate in four different modes. We choose architectures such that most of the blocks of individual architectures overlap with the merged architecture. The only modifications necessary are in terms of additional delay-switch-delay elements and muxes. The 16/64/256/1024-point obfuscated FFT after these changes is shown in Figure 9. Corresponding select signals which are derived from the key can then be used to select one among the four modes using the inputs $c0 - c3$. Thus, we have generated a structure which operates in four different functionally correct modes which we call as the *meaningful modes* of the design. It is to be noted, that we can design the obfuscated FFT to work anywhere between 1 to 4 meaningful modes with associated trade-offs as will be discussed later.

3.2 Design of non-meaningful modes

Next, we use the concepts of control-flow modifications on switches and trivial multiplication selection of butterfly units as discussed earlier. From Figure 9, we observe that the control signals available for modification include $s0 - s9$ of switches and $t0 - t4$ of butterfly units. The correct sequences for each of these signals are derived from a 10-bit counter and depend on the location and associated delays of the architecture as described in [10]. However, as part of obfuscation, additional sequences are also derived from the same counter. Since, we have seen that changes in the sequence for these signals affect the output in different ways, the derived incorrect sequences can be used to obfuscate the design. This concept is illustrated in Figure 10.

In Figure 10, $cntr$ represents a 10-bit register and $cntr[i]$ represents its i -th bit. Note that $s0$ has four different sequence combinations but only one value, i.e., $cntr[8]$ is the correct sequence for $s0$ when operated in 1024-point FFT mode. The other sequence combinations are derived to generate either random sequences or bits 0 and 1, all of which affect outputs. In this example, all other signals are also obfuscated with 4 different sequence choices. However, it is to be noted that the number of sequence derivations for each of these signals can be increased and this results in an increase in the size of the mux at the output of the signal. Increase in the size of the mux from 4:1 upto 32:1 results in a flexible design method with associated trade-offs as will be discussed later.

Once these obfuscated signals are set up, the select input for each of the muxes can be derived from the key of the design. A correct key selects all correct control signal combinations. Rest of the keys are mapped to modes which are non-meaningful or partially correct by using various other control signal combinations. Hence, depending on the size

of the key, non-meaningful modes are incorporated into the design. For each of these non-meaningful modes, it is advised to have atleast 50% deviation from the correct control signal combination.

For example, we can create a non-meaningful mode which selects the correct control signals for $s[0], s[2], s[4], s[6], s[8], t[0], t[2]$ and incorrect signals for the rest. This produces a mode with all incorrect outputs. On the other hand, if we select correct control for all signals except $t[0]$ and $t[4]$, we can create a partially correct non-meaningful mode with 25% correct outputs when made to operate in a 256-point, 2-parallel mode. Thus, control information known only to the designer is exploited in designing obfuscated modes.

Figure 10: Obfuscated control-path with various signals derived from a counter

4. ANALYSIS OF OBFUSCATED MODES

The effectiveness of this obfuscation scheme is analyzed by defining an attack model and then considering the role played by modes in obfuscation. It is important to note that several types of attack models exist and the attacker could be using more sophisticated techniques and tools. However, in this paper, we make use of an attack model generally used in design of obfuscation schemes and which reflects the scenario of an IP piracy or reverse engineering based attack [11].

4.1 Assumptions of the attack model

The design flow as described earlier is used to create an obfuscated FFT incorporating several different meaningful and non-meaningful modes. Since the FFT is used as a submodule in several applications, we assume that a complete architecture for the specific application is built around the obfuscated FFT. A netlist is then generated and the design is sent for further manufacturing to untrusted foundries. Once the design is manufactured and sent back to the design house, the key is programmed and stored in a safe memory inside the chip. This key selects one of the meaningful modes of operation of the circuit.

We assume that the attacker could get access to the obfuscated netlist through various sources and hence the circuit could be subjected to structural observations and functional simulations. The goal of an attack would be to try to decipher the correct key and in turn get access to the correct control information of the system. A second goal would be to guess what length of the FFT is used in the final design of the integrated circuit. Functionally correct I/O pairs can be assumed to be available to the attacker from a functional IC. However, this is from the application in which the FFT is used and hence does not give any additional information.

4.2 Obscurity of control-flow

Let us assume there are C different control signals necessary for the design. Each of the signals is obfuscated to a degree L using techniques described previously. For this purpose, a $L:1$ mux is used at the output of the control signal. For example, from Figure 10, $L = 4$ and from Figure 9, $C = 15$ corresponds to $s = 10$ and $t = 5$ variables. The strength of this method stems from the observation that it is not possible to derive the correct select signal for this $L : 1$ bit mux unless a complete simulation of the system is performed. Thus, we have L^C different control signal combinations possible out of which only one is correct. For a circuit with a large number of control signals, a smaller value of L can be used to achieve the same level of obfuscation,

At the next level, we consider the selection of these signals to create modes. For each mode, we can use M incorrect signal selections. A value of M equal to 0 would mean a meaningful mode of operation. For each control signal change at the switches or butterfly units, the outputs are deviated from actual outputs by either 100% or 25-50%, respectively. Even though, individual control signal changes cause sufficient difference in Hamming distance between correct and incorrect outputs, we use a value of $M \geq C/2$ to create non-meaningful modes. This ensures at least 50% deviation from the correct control sequence combination making it secure against attacks. Thus, for each M value, we can modify the signal incorrectly in L ways using the mux. This gives us a total of $\binom{C}{M} * L^M$ modes, for every chosen value of M . The

Table 1: Overhead due to meaningful modes

No. of meaningful modes	Total area overhead	Total power overhead
1 (1024 point)	0.2%	0.5%
2 (256/1024 point)	8.5%	10.6%
3 (64/256/1024 point)	38%	15.5%
4 (16/64/256/1024 point)	41.5%	17.3%

number of modes in turn dictates the size of the key and can be increased by using various values for M . Thus, control information is hidden effectively through the correct choice of L and M .

4.3 Protection of length of FFT

Meaningful modes and partially correct modes help in protecting the length of the FFT. Upon observation of an obfuscated netlist, an attacker is able to see the obfuscated datapath consisting of hardware units for four meaningful modes of the design. However, the attacker has no way of understanding which among the four modes has been used in the application. Thus, these modes successfully defend a reverse engineering attack. The partially correct modes give out some information about the system like 25% to 50% correct outputs. This can also be used to mislead the attacker into believing an incorrect length of the FFT is being used. For example, if 25% of the outputs of a 256-point FFT are observed by an attacker in a particular mode, then one might be led to believe that the application uses a 256-point FFT when actually a 1024-point FFT is used. Additionally, the partial control information given out in this mode is that of the incorrect length, i.e., 256-point FFT.

5. AREA AND POWER OVERHEAD

We quantify cost associated with the obfuscation scheme by simulating and synthesizing architectures with various modes and key sizes and calculating area and power overheads. All architecture descriptions are written using Verilog HDL and synthesized using Design Compiler. A 65nm technology library is used for all synthesis and the circuits are clocked at a speed of 100MHz. The overheads are compared with a 1024-point folded FFT design but with no obfuscation.

By varying several parameters of the design, we present three separate sets of results. For the first set of analysis, the datapath is obfuscated to have several meaningful modes ranging from 1 to 4, while keeping the switch size of control-path at a value 4 and key size as 16. It is to be observed that the meaningful modes involve changes in the datapath of the design and hence incur larger overheads with each additional mode. This is tabulated in Table 1. Hence, for practical applications it would be advisable to keep the number of modes at a nominal value of 2 for this FFT architecture. Security critical applications could use higher number of meaningful modes.

Next, we vary the switch at the control path outputs added as part of obfuscation. It was seen that these switches mainly contribute to the obscurity of control-flow of the de-

Table 2: Overhead due to size of mux at controlpath

Mux size of controlpath (L)	Controlpath overhead		Total overhead	
	Area	Power	Area	Power
2	2%	0.7%	8.2%	10.1%
4	5.5%	1.7%	8.35%	10.5%
8	22%	4.2%	8.5%	11.4%
16	41%	6.7%	9.1%	12.1%

Table 3: Overhead for different key sizes

Key size (Modes)	Total Overhead	
	Area	Power
4 (16)	8.29%	10.53%
8 (256)	8.33%	10.55%
16 (65536)	8.35%	10.59%
20 (1048576)	8.42%	10.63%
28 (268435456)	8.47%	10.66%

sign and hence its security. From Table 2 it is seen that the size of muxes ranging from 2 to 16 cause an overall overhead increase of around 2%. Hence, it can be used as a parameter to adjust levels of obfuscation as necessary. For these simulations, a meaningful mode value of 2 and key size 16 was used.

Finally, for the last set of results we keep the meaningful modes as 2, a switch size of 4 and vary the length of the key. For each of the keys, modes are created by using signal combinations from the obfuscated design. The number of incorrect signals is kept at a nominal value of 50%. It is seen that the area and power overheads do not vary much for key sizes upto 28 and hence it is easy to build the architecture to accommodate large key sizes.

Comparison of overheads with general encryption or obfuscation schemes cannot be made, since these schemes do not specifically address the goals of DSP circuit obfuscation. This includes protection of important properties such as length of FFT. However, still with a nominal meaningful mode value of 2, an area overhead of 8% and power overhead of 10%, comparable to general obfuscation or encryption schemes is obtained [5, 11]. An automated method could be developed which takes as input the key size, switch size and number of meaningful modes and tries to meet a specified area and power constraint.

6. CONCLUSION

In this paper, we have successfully demonstrated the mode-based method of obfuscation of circuits using a complex FFT architecture. We also illustrated a new approach to design modes, specifically using control-flow changes to design non-meaningful modes. The role of various modes with respect to security of the circuit and obfuscation achieved was also analyzed. Finally we showed that despite the method serving as a strong obfuscated FFT architecture, it does not increase the overhead of the design significantly which makes it practical for use in current designs. Future work would involve developing metrics to measure the level of obfuscation of a design. Such a metric would not only be a measurement tool but would also serve as a comparison tool for several approaches to obfuscation existing today.

7. ACKNOWLEDGMENT

This research has been supported in part by the National Science Foundation under grant number CNS-1441639 and the Semiconductor Research Corporation under contract number 2014-TS-2560.

8. REFERENCES

- [1] M. Rostami, F. Koushanfar, J. Rajendran, and R. Karri, "Hardware security: Threat models and metrics," in *Proceedings of the International Conference on Computer-Aided Design*, 2013, pp. 819–823.
- [2] M. Brzozowski and V. N. Yarmolik, "Obfuscation as intellectual rights protection in VHDL language," in *Proceedings of the 6th International Conference on Computer Information Systems and Industrial Management Applications (CISIM)*, 2007, pp. 337–340.
- [3] R. S. Chakraborty and S. Bhunia, "RTL hardware IP protection using key-based control and data flow obfuscation," in *Proceedings of the 23rd International Conference on VLSI Design*, 2010, pp. 405–410.
- [4] J. A. Roy, F. Koushanfar, and I. L. Markov, "EPIC: Ending piracy of integrated circuits," in *Proceedings of the conference on Design, automation and test in Europe*, 2008, pp. 1069–1074.
- [5] R. S. Chakraborty and S. Bhunia, "HARPOON: an obfuscation-based SoC design methodology for hardware protection," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 28, no. 10, pp. 1493–1502, 2009.
- [6] Y. Lao and K. K. Parhi, "Obfuscating DSP Circuits via High-Level Transformations," *IEEE Transactions on VLSI Systems*, pp. 819–830, May 2015.
- [7] G. N. Shanmugam, Y. Lao, and K. K. Parhi, "An obfuscated radix-2 real FFT architecture," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2015, pp. 1056–1060.
- [8] K. K. Parhi, *VLSI digital signal processing systems: design and implementation*. John Wiley & Sons, 1999.
- [9] 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.
- [10] M. Ayinala, M. Brown, and K. K. Parhi, "Pipelined parallel FFT architectures via folding transformation," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 20, no. 6, pp. 1068–1081, 2012.
- [11] J. Rajendran, Y. Pino, O. Sinanoglu, and R. Karri, "Security analysis of logic obfuscation," in *Proceedings of the 49th Annual Design Automation Conference*. ACM, 2012, pp. 83–89.