# Quality Obfuscation for Error-Tolerant and Adaptive Hardware IP Protection

Abdulrahman Alaql, Tamzidul Hoque, Domenic Forte, and Swarup Bhunia
Dept. of ECE, University of Florida, Gainesville, FL 32608
{alaql89, thoque}@ufl.edu, {forte, bhunia}@ece.ufl.edu

***Abstract*—Various attacks on hardware intellectual properties (IPs) have been successful in obtaining design information that can be used to reverse engineer a system, create counterfeits, or insert hardware Trojans. Key-based hardware obfuscation is an attractive solution that helps prevent such attacks. In this paper, for the first time, we propose a key error tolerant obfuscation approach that achieves graceful degradation in output Quality of Service (QoS) as the bit error rate (BER) in obfuscation key increases. The approach, which we refer to it as, "Quality Obfuscation", is applicable to a large variety of IPs, including digital signal processing (DSP) and approximating computing IPs, which are resilient to output QoS degradation. We present a complete obfuscation framework that can be adapted to any error tolerance rate. To demonstrate its robustness, we obfuscate several common DSP IP blocks and observe the performance under various percentages of bit-flips in the key. We show that our approach provides controllability of system quality, as well as the strong protection at low overhead, e.g., average 15% area and 5.9% power overhead to tolerate 10% BER.**

## I. Introduction

Use of hardware intellectual properties (IPs) in system on chip (SoC) design has become a prevalent practice. The security of these IPs, however, has emerged as a major concern, as $250 billion are lost due to IP piracy and the presence of counterfeits in the semiconductor supply chain [1]. Hardware IPs have been targeted by untrusted parties involved in the design and fabrication process to create counterfeits, reverse-engineer a design to steal its secrets, overproduce chips, and insert Trojans [12]. One promising approach to protect IPs from these major attacks is to apply hardware obfuscation, where a physical or a functional locking/hiding mechanism is incorporated into the design. In particular, the functional obfuscation mechanism can take many forms, such as logic locking [9]. Most obfuscation techniques require a key in order for the design to be unlocked, which can be generated or stored internally (e.g., physical unclonable function or non-volatile memory). Alternatively, the key can come from an external source such as a user terminal, a biometric interface [3], an on-board memory or a trusted platform module (TPM), or a remote server. PUF [2] and biometric-based key sources are most promising for hardware obfuscation when applied to field programmable gate arrays (FPGAs) or application-specific integrated circuits (ASICs) with reconfigurable fabric. Traditionally, obfuscation keys are incorporated to ensure maximum possible functional failure in the presence of a wrong key. Therefore, even a small error rate in the key may have a huge impact on the system.

In this paper, we present a quality obfuscation scheme that enables reliable and adaptive hardware IP protection. The approach focuses on DSP IPs, as it aims to relate the percentage of error in the key with the output quality of the system. We propose a lightweight and flexible obfuscation approach that allows the user to control the level of degradation in quality as a function of error rate in the key. This is achieved by functionally partitioning a design into critical (with the highest impact on output quality) and non-critical (lower impact on output quality) ones. DSP IPs are suitable for this approach since most of these IPs consist of non-critical parts. Next, it applies low-overhead conventional logic locking obfuscation to the non-critical part while applying an optional higher overhead redundancy-based obfuscation to the critical blocks. Fig. 1 gives an overview of the conventional and the proposed schemes. In addition, our approach has the following benefits:

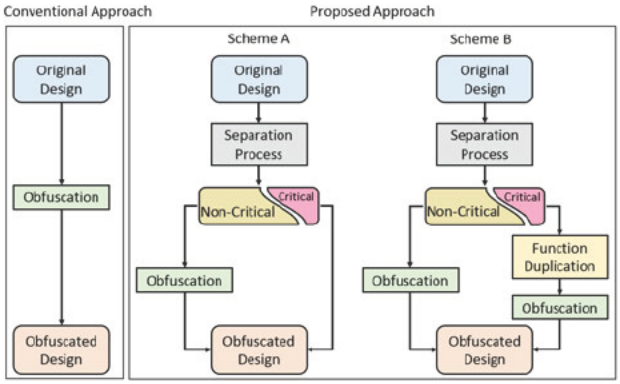


*Fig. 1: Overview of conventional vs. proposed obfuscation schemes.*

1) It provides resilience to bit-flips in the unreliable obfuscation key with graceful degradation in quality. It continues to provide correct functional behavior of an obfuscated design in presence of bit failures in the key.
2) It provides a controllable process that allows different levels of quality to be used for different IP customers. By supplying each customer with a key that has a certain (unique) percentage of errors, the IP owner can control the level of quality (Tier) that each user can have.
3) It protects an IP from untrusted testing facilities by supplying them with an obfuscated design and a partial key that provides a low quality-tier. The quality tier is selected to provide information to test/verify a design while maintaining a lower quality.

Unlike existing obfuscation techniques, which do not consider the control of output quality and error tolerance, our approach provides similar protection as those approaches, while overcoming the issues of unstable key-bits. Fig. 2 shows an overview of the proposed quality obfuscation approach. In particular, our major contributions are as follows:

- We partition a design in critical and non-critical part in terms of QoS and then propose a bimodal obfuscation approach - a conventional obfuscation for the non-critical part and a redundant obfuscation for the critical part.

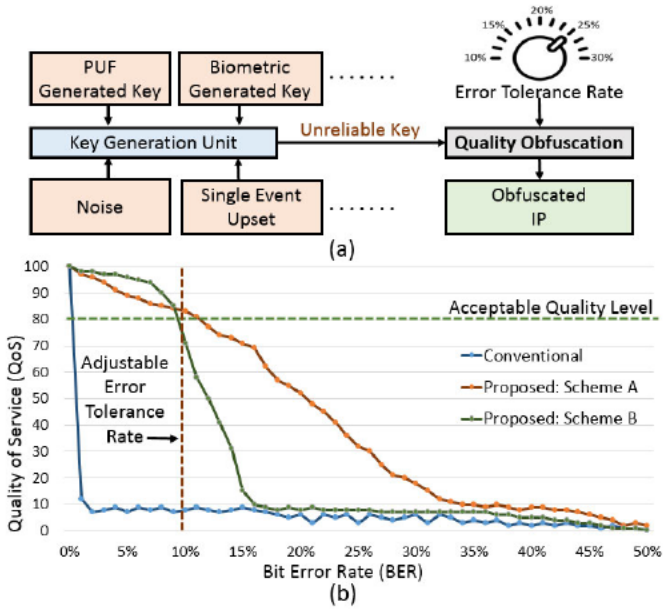


*Fig. 2: (a) Sources of obfuscation key unreliability that our adaptive framework can overcome. (b) A small percentage of error with graceful degradation in output quality followed by a gradual decrease (Scheme A) or a steep drop (Scheme B) in quality.*

These modes control the pattern of output quality degradation as the error rate in the key increases.

- We provide two case studies on common signal processing IPs, namely, finite impulse response (FIR) filter and discrete cosine transform (DCT), to show promising performance in terms of security and design overhead under various levels of output QoS and key error rate.
- We provide a comprehensive security assessment for the proposed obfuscation schemes, under different attack models, such as brute force and SAT attacks.
- We compare our approach to popular error tolerance methods, e.g., error correction codes (ECCs).

The remainder of the paper is organized as follows. Section II provides background on key reliability, existing error correction methods, and the threat model. Section III discusses the methodology for the proposed approach and Section IV presents two case studies. Section V presents security analysis against various known attacks and discusses comparative benefits against existing ECC based approaches. Section VI presents the conclusion and future directions.

## II. Background

### A. Key Reliability and Error Correction

Existing obfuscation techniques tend to maximize the output corruption when a small percentage of the key is incorrect, which can be problematic if the used key is unreliable and an unintended error occurs. Secret keys used for obfuscation may become unstable due to the following:

- Internally generated keys from hardware security primitives, such as PUFs, may suffer from reliability issues due to their high sensitivity to aging, supply voltage drop, and temperature [2].
- Biometric-based keys (e.g., fingerprints, iris, or facial recognition) may suffer from noise from the environment, biometric sensor, or aging-related effects [3].
- Environmental noise and intentional tampering may cause errors in the key during data transmission.
- A storage unit (e.g., ROM, flash, or other non-volatile memory) containing the key can be subject to single event upset (SEU) that causes stored data to be corrupted and provide incorrect values [11].

The above-mentioned causes for errors in the key may not affect all system architectures. For example, PUF based keys can only be used if the design is implemented in a reconfigurable platform (such as FPGAs and ASICs with reconfigurable fabric). Table I shows the causes of errors and which platforms are affected.

*TABLE I: Key Reliability Issues in Different System Architectures*

| Cause of Key Unreliability | FPGA | Reconfig. ASIC | ASIC |
|---|---|---|---|
| PUF-Generated Keys | ✓ | ✓ | ✗ |
| Biometric-Based Keys | ✓ | ✓ | ✗ |
| Environmental Noise | ✓ | ✓ | ✓ |
| Single Event Upset | ✓ | ✓ | ✓ |

The use of ECCs has been the standard approach when integrating unreliable keys into an obfuscated design [10]. The ECC block generates a checksum and additional data from the initial secret key. This data is then stored in the device and can be used to detect and correct errors in the key. The major limitation of employing ECCs is the large overhead, where a large portion of the design and the memory space will be allocated for the ECC decoder and the checksum data. The overhead increases dramatically as the error correction capability increases, which makes this option not suitable for system-on-chips with many obfuscated hardware IPs. Additionally, ECC has been shown to be vulnerable to side-channel attacks, where power and electromagnetic analysis have been applied to extract the key out of the ECC module [14].

Fuzzy extractors have been widely used to allow unreliable keys to be used for cryptographic systems [4]. Fuzzy extractors compare the original stored key with the input key, and the input is accepted if it is close enough to the stored key. Similar to ECCs, helper data is also needed in this process. However, the produced key is completely uncorrelated to the input key, which makes it more secure than ECC. Fuzzy extractors are suitable for PUF and biometric-based keys, however, they require large resources to be implemented [4].

### B. Quality Obfuscation

The area of quality obfuscation has not been well explored in the literature. Only a few works have focused on obfuscation techniques that reduce the quality of the system rather than the output corruptibility. The proposed approach in [16] applies a delay obfuscation mechanism, where a wrong key forces the system into a slower processing time. The approach only works for sequential circuits, and it does not corrupt the output. Although the system still provides valid outputs when the key is incorrect, the quality of service (processing time in this context) is affected by the used key.

### C. Threat Model

Most adversaries aim to obtain design information of the system under attack in order to make counterfeits. The other goal for adversaries is to modify a system with a back-door or a Trojan that interrupts the functionality at a specified instance. Adversaries apply these modifications to systems that

perform functional failure or information leakage. In this work, we assume Kerckhoffs's principle [15], where attackers have access to the gate-level netlist and have familiarity with the system and its functionality. The only thing the attacker does not have is the secret/obfuscation key.

## III. Methodology

We propose a lightweight approach for quality obfuscation that takes into account the criticality factor of the target function, which leads to a separation of critical and non-critical parts. This factor is observed by analyzing which parts of the function carry the highest impact on the output, and which parts have lower output impact. The number of critical and non-critical parts and their corresponding criticality-factor varies across different designs. A major assumption in this work is that the output of the IP is always a numerical value (signed or unsigned). The proposed obfuscation process flow is shown in Fig. 3. The major proposed steps and details involved in the obfuscation process are as follows:

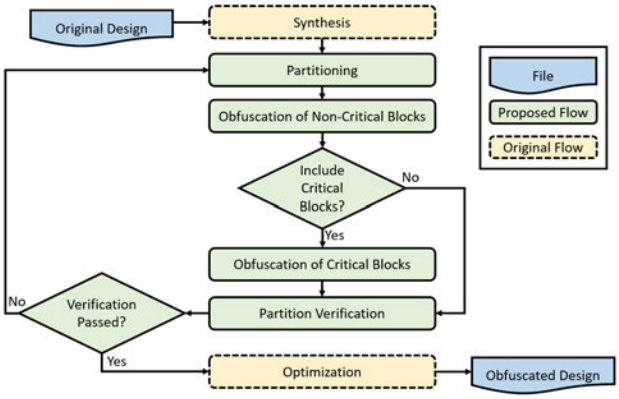


*Fig. 3: Process flow for the error-tolerant obfuscation.*

### A. Partitioning

The partitioning process that separates critical and non-critical blocks of the design is applied in this stage. A block is defined as a gate or the basic logic element of the IP. Identifying critical blocks is subjective, where different factors may determine whether a block is critical or non-critical. These factors include the key size, the selected error tolerance rate, and the accepted quality of service. The maximum acceptable number of bit-flips ($e_{max}$) can be calculated as $e_{max} = e_{rate} \times K_{size}$, where $e_{rate}$ is the user-selected error tolerance rate, and $K_{size}$ is the key size. The minimum acceptable quality of service ($QoS_{min}$) is also determined by the user. Eqn. 1 shows how the quality of service is calculated:

$$QoS = [1 - \frac{|OUT_{original} - OUT_{obfus}|}{|OUT_{max} - OUT_{min}|}] \times 100\% \quad (1)$$

where $OUT_{original}$ is the output of the original unobfuscated design, $OUT_{obfus}$ is the output of the obfuscated design with $e_{max}$ applied. $OUT_{max}$ is the maximum possible value for the function, and $OUT_{min}$ is the minimum possible value. The partitions are considered "accurate" if all combinations of $e_{max}$ bit-flips result in $QoS > QoS_{min}$. The following two methods are used to perform the partitioning. $QoS$ is calculated for one output signal, and to get an accurate overall $QoS$, a set of diverse signals should be applied and the average $QoS$ is used.

*1) **Weighted Observability Analysis**:* The output observability for a specific gate is the probability of how likely for the gate's value to propagate to the primary output. The weighted observability can determine how critical a gate is by relating the observability to the bit order of the output pin, where most significant output pins yield larger effect. Various design tools can generate the observability ($OB_g[n]$) for every gate in the netlist, where $n$ is the output pin order (output size in bits). Eqn. 2 shows how to compute the weighted observability ($WOB_g$) for gate $g$ and output size of $b_{max}$:

$$WOB_g = \sum_{n=1}^{b_{max}} OB_g[n] \times 2^{n-1} \quad (2)$$

$2^{n-1}$ indicates the value of the output pin with the bit-order of $n$. The weighted observability can help determine which gates belong to the critical partition, as higher $WOB_g$ values indicate higher impact to the overall output. Eqn. 3 shows how to calculate the total acceptable weighted observability threshold $WOB_{th}$:

$$WOB_{th} = (1 - QoS_{min}) \times (|OUT_{max} - OUT_{min}|) \quad (3)$$

Identifying critical and non-critical gates is based on both the maximum acceptable bit-flips $e_{max}$, and the maximum tolerable deviation in output quality ($WOB_{th}$). Gates are considered non-critical if they meet the condition expressed in Eqn. 4:

$$\sum_{n=1}^{e_{max}} Random(WOB_g) < WOB_{th} \quad (4)$$

The condition for the non-critical partitioning elaborated in Eqn. 4 indicates that the sum of $WOB_g$ of all combinations of $e_{max}$ gates should be less than $WOB_{th}$. Algorithm 1 shows how the critical and non-critical gates are identified.

**Algorithm 1** Partitioning algorithm

```
1:  procedure WEIGHTED OBSERVABILITY ANALYSIS
2:  Input: e_max: Maximum number of bit-flips
3:  Input: WOB_th: Weighted observability threshold
4:  Input: gates[k]: List of all gates
5:  Input: WOB_g[k]: Weighted observability for all gates
6:      Sort WOB_g[k] and gates[k] (descending)
7:  for all gates in WOB_sorted[k]
8:      total_weight = sum_{n=k}^{k+e_max} WOB_sorted[n]
9:      if total_weight < WOB_th then
10:         critical = gates_sorted[0 : k − 1]
11:         noncritical = gates_sorted[k : end]
12:         break
13:     end if
14: end for
15: Output: critical = gates_sorted[0 : k − 1]
16: Output: noncritical = gates_sorted[k : end]
17: end procedure
```

Algorithm 1 performs the sum of a sliding window with the size of $e_{max}$ and checks if the sum ($total_{weight}$) is less than the threshold $WOB_{th}$. Once the condition in line 9 is satisfied, all gates inside and beyond the sliding window are considered non-critical, and the rest are considered critical.

*2) **Tracing Signal Weight**:* When available, prior knowledge about the design is helpful to identify which parts carry most of the signal's weight, and therefore isolate critical parts. Since the basic functionality of common DSP IPs is well-known, identifying critical parts can help improve the accuracy of the partitioning process. Table II shows several common DSP IPs and their respective critical parts.

*TABLE II: List of Common DSP IPs and Their Critical Parts*

| DSP IP | Critical Part |
|---|---|
| Fast Fourier Transform (FFT) | High Order Windows [5] |
| Finite Impulse Response (FIR) | Middle Stages [6] |
| Discrete Wavelet Transform (DWT) | First Approximation and Detail [5] |
| Discrete Cosine Transform (DCT) | First Stages [7] |
| Approximation Functions | Most Significant Bits [8] |
| Lossy Compression | Most Significant Bits [8] |

## B. Obfuscation

Our proposed approach consists of two different schemes that the user can choose from. Fig. 4 shows an overview of the proposed obfuscation schemes. The following list elaborates on how the obfuscation process is applied in each scheme:

*1) **Scheme A: Non-Critical Only Obfuscation**:* In this scheme, the conventional obfuscation process is applied to the design. However, the obfuscation process is not allowed to select any nodes to obfuscate in the critical partition. Errors in the key in this scheme are not corrected, this is because the impact on the output is negligible when few errors in the keys to the non-critical blocks occur. The critical part is not obfuscated in this scheme. In this scheme, the output quality gradually decreases as the error rate increases.

*2) **Scheme B: Function Duplication for Critical Blocks**:* This scheme applies the same obfuscation process used in Scheme A to the non-critical part. However, the critical part is run through a redundant obfuscation process that can maintain a similar or higher error tolerance rate than what is specified in the partitioning process. The process duplicates critical blocks and supplies each duplicate with its own key gate and key input. Following the duplicates, a majority voting function is added. This function provides the correct output if most duplicates have a valid key. The number of duplicates can determine how many bit-flips are tolerated per critical block. Additionally, outputs of critical blocks are not allowed to flip in order to maintain a valid primary output. By controlling the number of obfuscated critical blocks, the drop in output quality can be easily controlled. Obfuscating more critical blocks lead to sharper fall in the output quality after the tolerance region, at the expense of additional area overhead. Although critical blocks can tolerate a bit-flip or a few errors in the key, there is always a probability that more key bits in a critical block are flipped and the output is corrupted, we call this a failure. Eqn. 5 shows the probability of failures:

$$P_{failure} = e_{rate}^{k} \times 100\% \tag{5}$$

where $P_{failure}$ is the probability of failure, $e_{rate}$ is the error rate in the key, and $k$ is the number of duplicates per critical gate. It is clear from Eqn. 5 that increasing the number of duplicates reduces the failure rate, at the cost of additional overhead. For example, under a 10% of error rate of the obfuscation key, critical blocks that have 3, 5, and 7 duplicates have a failure probability $P_{failure}$ of 0.9%, 0.074%, and 0.005% respectively.

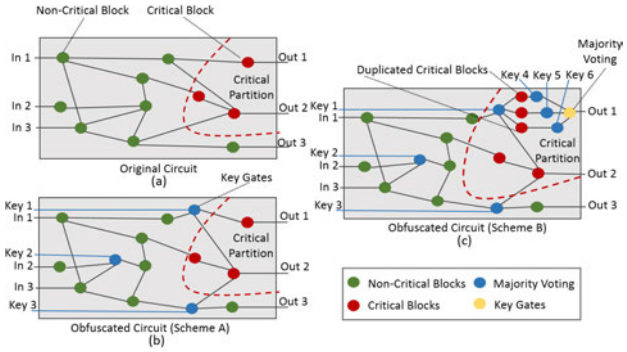


*Fig. 4: (a) Original partitioned design where critical blocks (red) and non-critical blocks (green) are identified. (b) Conventional obfuscation is applied to the non-critical partition. (c) Redundant obfuscation is applied to duplicates of a critical block followed by majority voting.*

## C. Partitioning Verification

After the obfuscation key gates are inserted, a verification process is applied to ensure that the necessary quality obfuscation and error tolerance are provided. The process applies a set of inputs and key error rates to test the effectiveness of the obfuscation. If the verification fails, the partitioning is applied with a reduced $WOB_{th}$, which can lead to a more conservative non-critical partition.

The obfuscated design is then re-synthesized, and the optimization process is applied again while accounting for the original performance constraints.

# IV. Case Study: DSP IPs: FIR and 2D-DCT

To investigate the effectiveness of the proposed error-tolerant obfuscation approach, the method has been applied to two DSP IPs, a Finite Impulse Response (FIR) filter, and a 2-Dimensional Discrete Cosine Transform (2D-DCT). The FIR filter is chosen in this case study because it is an essential part of most DSP systems. Additionally, the efficiency and the reduced computational complexity of FIR filters make them suitable for hardware implementation [5]. The 2D-DCT IP is an essential stage in every image processing system, where the output of the transform is a compressed version of the input array. 2D-DCT is used in both still imaging systems and video processing systems. The 2D-DCT is widely used as a compression stage in most modern imaging devices due to its simple implementation, low processing delay, high compression quality, and high efficiency compared to other compression methods [5]. Both IPs are well-suited for demonstrating the proposed obfuscation approach over a range of different scenarios because of the large presence of non-critical elements in their computation processes [6]. The separation process of critical and non-critical parts is based on the assumption that the obfuscation key has a 10% error rate.

Both IPs are implemented on a general purpose field programmable gate array (FPGA) using ISE Design Suite and ModelSim. The target device used in this implementation is a Xilinx Spartan6 XC6SLX75 FPGA, where the generated netlist consists of a set of look-up tables (LUTs) that are configured to implement the basic functions of the design. The obfuscation method used in this case study is a low-overhead approach introduced in [13] that utilizes unused parts of the FPGA to insert key inputs and dummy functions.

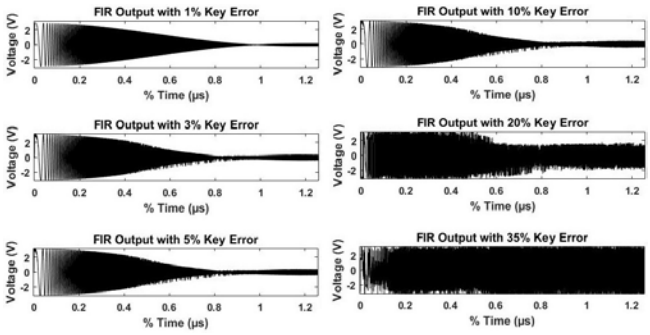


*Fig. 5: Output for the obfuscated FIR filter with when different key error percentages. The filter provides a valid output when the key error percentage is up to 10%.*

### A. Identifying Critical Blocks

The FIR filter is partitioned as critical and non-critical by analyzing the weighted observability described in Algorithm 1. For the 2D-DCT, the applied partitioning approach (elaborated in section 3.2.2) is done by tracing the parts that carry most of the signal's weight. The DCT design consists of an 8x8 coefficients matrix, and first 20 coefficients are considered critical since they hold over 85% of the signal's energy, while the rest of the coefficients are considered non-critical. Table III shows the obfuscation results of both IPs.

*TABLE III: Obfuscation results of the conventional and the proposed schemes, the key sizes, and the key space size for brute force attacks.*

| IP | Obfuscation Scheme | Total LUTs | Key Size | Key Space Size (Brute force attack) |
|---|---|---|---|---|
| FIR | Conventional | 1433 | 1433 | $2^{1433}$ |
| | Scheme A | 1433 | 901 | $2^{810}$ |
| | Scheme B | 1724 | 1627 | $2^{1464}$ |
| 2D-DCT | Conventional | 809 | 809 | $2^{809}$ |
| | Scheme A | 809 | 372 | $2^{334}$ |
| | Scheme B | 1442 | 1231 | $2^{1107}$ |

### B. Error Tolerance and Quality of Service (QoS)

The outputs of the obfuscated FIR filter with different key error percentages are shown in Fig. 5. When the key error is at 1%, the filter output is almost identical to the original output. This goes up to 10% of key error with only graceful degradation in the output's quality. Fig. 6 shows the output of the error-tolerant 2D-DCT obfuscated design with the Lena image as the input and different BERs in the key.

Quality of service ($QoS$) for both IPs are observed across different error rates using Eqn.1. The coherence index of the FIR filter is obtained by comparing the original filter implementation's frequency response with the obfuscated design. The approach is to obtain the magnitude-squared coherence between the original and the obtained frequency response. The quality of the output of 2D-DCT is measured by checking the amount of lost signal energy in the compressed output compared to the input image. This energy is measured by looking at the Peak Signal to Noise Ratio (PSNR), and a "good" quality image is obtained when its PSNR is above a threshold of 25 dB. Fig. 7 shows $QoS$ analysis for both IPs. The conventional obfuscation seems to be sensitive to any minor error in the key, whereas the proposed scheme shows a more error-tolerant manner when BER is less than 10%.


*Fig. 6: Outputs for the obfuscated 2D-DCT with key error percentage set at (a) 1%, (b) 10%, (c) 12%, and (d) 20%. The quality drops off steeply after 10% error.*

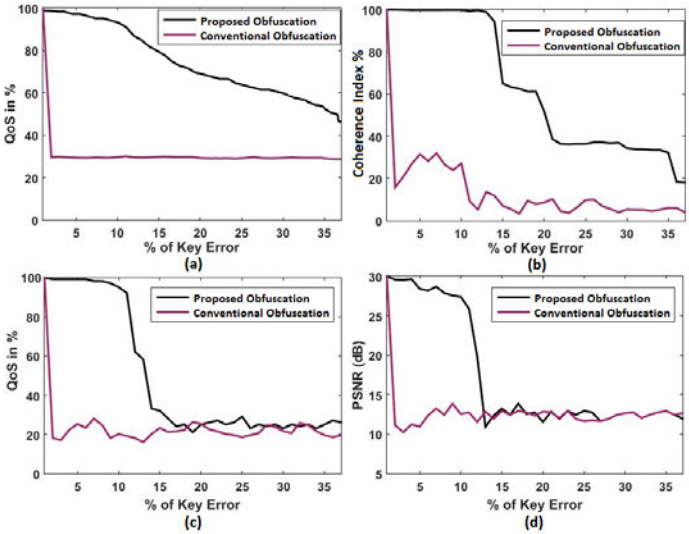


*Fig. 7: (a) QoS and (b) coherence index between the original FIR Filter and obfuscated Output (conventional and Scheme A). (c) QoS and (d) PSNR index between the original 2D-DCT and obfuscated output (conventional and Scheme B).*

### C. Area, Power, and Latency Overhead

The area overhead of our proposed schemes is caused by key-gate insertion and the function duplication of critical parts. The amount of area overhead varies depending on the nature and the size of the target design. The overhead analysis is done using the ISE Timing Analyzer and ISE XPower Analyzer. The overhead results (compared to the original design) for both IPs at 1%, 5%, and 10% of error tolerance are shown in Table IV.

Due to the time-sensitive nature of DSP IPs, the latency overhead for all schemes is set to be 0% as we constrained the latency to match the original design. This leads to higher area and power overheads as shown in Table IV. Scheme A has a 0% area overhead in both IPs as it does not add any additional LUTs in the design. However, as the error tolerance rate decreases, power overhead increases. The reason for this increase is due to the critical/non-critical separation process which labels more LUTs to be critical. And since Scheme A avoids obfuscating critical LUTs, fewer LUTs are obfuscated when increasing the error tolerance rate. On the other hand, Scheme B applies a triple modular redundancy to all critical LUTs, and since additional LUTs are added, the area overhead increases with the error tolerance rate.

## V. Security Analysis

### A. Brute Force Attacks

Due to the large key space of the proposed obfuscation approach, and even with the error tolerance of 10%, the

*TABLE IV: A comparison between our approach and the use of ECC with 1%, 5% and 10% of error correction.*

| IP | Obfuscation Scheme | Error Tolerance: 1% | | Error Tolerance: 5% | | Error Tolerance: 10% | |
|---|---|---|---|---|---|---|---|
| | | Area Overhead | Power Overhead | Area Overhead | Power Overhead | Area Overhead | Power Overhead |
| FIR | Proposed (Scheme A) | 0% | 15.3% | 0% | 13.8% | 0% | 12.5% |
| | Proposed (Scheme B) | 16.9% | 7.1% | 20.1% | 7.1% | 27.3% | 7.4% |
| | Conventional With ECC | 218% | 215% | 846% | 653% | 1588% | 1612% |
| 2D-DCT | Proposed (Scheme A) | 0% | 4% | 0% | 3% | 0% | 0% |
| | Proposed (Scheme B) | 28.6% | 1% | 39.4% | 1% | 48.2% | 2% |
| | Conventional With ECC | 238% | 119% | 947% | 324% | 1715% | 453% |

minimum amount of correct key guesses required to get the IP to work properly cannot be achieved through brute force attacks. For example, for an obfuscated design using a 400-bit key with 10% of error tolerance, a minimum of 360 bits have to be set correctly in order for the system to provide the correct functionality. Table III shows the maximum key space search size for each scheme when applying a brute force attack. The number of trials is based on the key size after reducing the error tolerance rate.

In the case of supplying an untrusted testing facility with a partially correct key, the key search space is also too large to brute-force. For example, if the obfuscated FIR filter using a 1000-bit key is given to the untrusted party with a 70% correct key, the key search space to obtain a good quality (to reach a 90% of key correctness) is still too large to brute-force.

### *B. Boolean Satisfiability (SAT) Attacks*

The boolean satisfiability attack is done by applying an iterative algorithm that can break a combinational logic locking technique. In order for the attack to be successful, an obfuscated netlist is required as well as a functional device with the correct key embedded inside. The key is extracted by searching for distinguishing input patterns (DIPs) that can reduce the key search space and eventually extract the key [9]. We have tested our approach against SAT attacks by applying the resolving algorithm to the obfuscated FIR filter. The sequential parts have been removed and the SAT resolving algorithm has been run using a 4-core 2.8GHz processor with 20GB of RAM. The SAT attack tool has not been able to obtain the key from the obfuscated FIR filter even after running the SAT tool for more than 5 days. The SAT attack failed to get the key of the obfuscated design due to the presence of multiplications and other arithmetic operations, which are inherently hard for SAT solvers to resolve. Hence, SAT attacks should not raise any security concerns to DSP IPs.

### *C. Proposed Approach vs. ECC*

The conventional obfuscation approach has been applied to both IPs with an ECC block. The used ECC is a BCH algorithm [10] with 1%, 5% and 10% error tolerance. The designs for both IPs have been mapped using the same settings and target device as the proposed obfuscation approach. The ECC encoder is added to the key generation stage of the obfuscation process and the resultant checksum is stored in the design. Overhead analysis for the conventional obfuscation with ECC is shown in Table IV. Area and power overheads are dramatically increased when increasing the error correction percentage. The key size required has also increased as the ECC error correction rate reaches 10%. The overall comparison between the proposed approach and the use of ECC shows that our approach provides the same level of protection with less power and area overheads.

## VI. Conclusion

In this paper, we have presented a novel hardware obfuscation approach that can protect DSP IPs against piracy and reverse engineering, while providing a high level of resilience to key bit errors at low hardware overhead. The approach enables trading off correctness of key bits with output QoS and provides graceful degradation in QoS with increasing BER in the key. We have presented a complete methodology and developed a CAD tool for automatic obfuscation of a design. We have shown that the resultant obfuscated design is protected against all major known attacks. Implementation results indicate that the overhead of our approach is significantly lower than ECC. Future work on this topic will focus on tolerating higher failure rate in critical parts, studying the effectiveness of obfuscation to general designs, and further reduction in overhead.

## VII. Acknowledgment

This work is supported in part by National Science Foundation grants: CNS 1651701, CNS 1603483, DGE 1623310.

## References

[1] OECD, "The Economic Impact of Counterfeiting and Piracy: Executive Summary," in *Org. for Economic Co-operation and Development,* 2007.
[2] F. Koushanfar, "Provably Secure Active IC Metering Techniques for Piracy Avoidance and Digital Rights Management," in *IEEE Transactions on Information Forensics and Security,* pp. 51-63, 2012.
[3] N. Karimian, Z. Guo, F. Tehranipoor, D. Woodard, M. Tehranipoor, and D. Forte, "Secure and Reliable Biometric Access Control for Resource-Constrained Systems and IoT, in *Computing Research Repository,* 2018.
[4] Y. Dodis, R.l Ostrovsky, L. Reyzin, and A. Smith, "Fuzzy Extractors: How to Generate Strong Keys from Biometrics and Other Noisy Data, in *Computing Research Repository,* 2006.
[5] K. Parhi, "VLSI Digital Signal Processing Systems: Design and Implementation," Wiley, NY 1999.
[6] N. Banerjee, J. H. Choi, and K. Roy, "A Process Variation Aware Low Power Synthesis Methodology for Fixed-point FIR Filters," in *ISLPED,* Portland, OR, pp. 147-152, 2007.
[7] N. Banerjee, G. Karakonstantis and K. Roy, "Process Variation Tolerant Low Power DCT Architecture," in *2007 Design, Automation & Test in Europe Conference & Exhibition,* pp. 1-6, 2007.
[8] A. Timan, "Theory of approximation of functions of a real variable," 1963.
[9] P. Subramanyan, S. Ray and S. Malik, "Evaluating the Security of Logic Encryption Algorithms," in *2015 IEEE International Symposium on Hardware Oriented Security and Trust (HOST),* pp. 137-143, 2015.
[10] Y. Lao, "Authentication and Obfuscation of Digital Signal Processing Integrated Circuits," in *Information and Control,* 2015.
[11] R. Baumann, "The impact of technology scaling on soft error rate performance and limits to the efficacy of error correction," in *IEDM Tech. Dig.,* pp. 329-332, 2002.
[12] R. S. Chakraborty, and S. Bhunia, "Hardware Protection and Authentication through Netlist Level Obfuscation," in *ICCAD,* pp. 674-677. 2008.
[13] R. Karam, T. Hoque, S. Ray, M. Tehranipoor, and S. Bhunia, "Robust Bitstream Protection in FPGA-based Systems through Low-Overhead Obfuscation," in *ReConFig,* Cancun, Mexico, pp. 1-8, 2016.
[14] L. Tebelmann, M. Pehl, and G. Sigl, "EM Side-Channel Analysis of BCH-based Error Correction for PUF-based Key Generation," in *ASHES,* 2017.
[15] C. Shannon, "Communication Theory of Secrecy Systems". in *Bell System Technical Journal,* pp. 656-715, 1949.
[16] L. Li, and H. Zhou, "Structural Transformation for Best-Possible Obfuscation of Sequential Circuits," in *2013 IEEE International Symposium on Hardware Oriented Security and Trust (HOST),* pp. 55-60, 2013.