Accepted for Publication in ACM TODAES, May 2026

DECODE: Detecting Structural Obfuscation in HLS based DSP Designs using Machine Learning Framework

Anirban Sengupta
Indian Institute of Technology Indore, India, asengupt@iiti.ac.in

Nabendu Bhui
Indian Institute of Technology Indore, India, phd2401101004@iiti.ac.in

Vishal Chourasia
Indian Institute of Technology Indore, India, vishalchourasia@iiti.ac.in

Structural obfuscation has been a widely used hardware security technique that is used to transform the internal structure of a high-level synthesis (HLS) digital signal processing (DSP) intellectual property (IP) design such that it becomes unobvious and obscure. The intent is to thwart deobfuscation/reverse-engineering (RE) attempts. These transformations in structurally obfuscated DSP designs are elusive and non-intuitive, making them arduous to identify and detect from an attacker's perspective. In this paper, we for the first time, present a novel strategy for detecting structural obfuscation in HLS based DSP designs using machine learning (ML) frameworks. This paper presents a novel technique called *'DECODE'*: ***DE****tecting Stru****C****tural* ***O****bfuscation in HLS based* ***D****SP Designs using Machine L****E****arning Framework*. The presented approach introduces a set of unique/distinct features that can successfully classify between structurally obfuscated HLS DSP designs and its non-obfuscated versions. The paper also presents a novel ML database consisting of several instances of structurally obfuscated and un-obfuscated DSP HLS designs that can be fed into our ML framework (logistic regression and random forest classifier). Results of the proposed approach revealed successful detection of structurally obfuscated DSP designs with high accuracy.

CCS CONCEPTS • DSP design • HLS • Machine learning • Structural obfuscation

1 INTRODUCTION

Digital signal processing (DSP) based hardware IP designs are extensively integrated in system-on-chips, which are used in different areas such as video and image compression, filtering, de-noising, portable computing systems, multimedia systems, wireless and wired communication networks [1] [2] [14] [17]. It contributes significantly to advancements in modern electronic and communication systems. Therefore, the security of DSP-based IP cores is crucial for maintaining the security of modern systems [22] [14] [15]. One standard method to secure DSP design involves altering the design's structure, making it unrecognizable to adversary/attackers and preventing cloning/counterfeiting and Trojan insertion. Generically, this process is known as hardware obfuscation [22] [23] [24] [25] [3]-[7]. Structural obfuscation is a hardware security technique [8] [10] [12] [13] that transforms the internal structure of a DSP IP design into an obscure design, without affecting the functionality and preventing an attacker from detecting/analyzing, understanding and interpreting the DSP design during deobfuscation/reverse-engineering [22] [23] [24] [25]. It conceals

the design functions by altering the datapath connections and overall framework, which results into unclear and hidden structure of the DSP design, while still preserving its functionality. Structural obfuscation technique conceals/obfuscates the DSP IP designs through a series of high-level transformation (HLT) techniques with low design cost, viz. logic transformation, tree height transformation, and loop unrolling [22] [21]. These HLT methods modify the structure of the design by altering logic arrangements, adjusting hierarchical depth, and expanding iterative loops, which makes internal architecture more complex and less recognizable while attempting reverse engineering.

High level synthesis streamlines the DSP design process by automatically producing register-transfer level (RTL) hardware description language (HDL) (or VHDL) code that captures the functional behavior of the DSP IP design [20]. However, with the advancement of design techniques, there also exists reverse engineering tools [9] [11] which can be maliciously employed by an adversary for unethical/illegal purposes. An adversary/attacker may attempt to reverse engineer a design that has been structurally obfuscated. One of the essential steps for successful reverse engineering, would be to detect/identify structural obfuscation logic in HLS based DSP IP design. This is because identifying/detecting structural obfuscation logic may form an essential part of deobfuscation in the reverse engineering process [33], [34]. It has been established in [33] and [34] that automatic deobfuscation of obfuscation logic is an important step toward reverse engineering. Identifying structural obfuscation and its type helps to make the obfuscation logic easier to understand with help of standard deobfuscation/hardware reverse engineering tools [35], [36], [37]. For example, in [37], as part of the deobfuscation/RE attempt, formal equivalence checker was used which requires recovery of original control data flow graph (CDFG). Identifying and detecting structural obfuscation in term of LU/THT/LT helps in faster and accurate recovery of the original CDFG. Nevertheless, detection of such structurally obfuscated logic is complex because such structural obscurity in an elaborate HLS generated DSP RTL design is non-intuitive. As structural obfuscated designs contain several convoluted features and transformations which do not affect the design functionality but subtly change the architecture of the design such that it becomes not recognizable to an attacker/adversary. These features are secretive in nature and appear like a normal interconnected design component in the RTL of the design. Furthermore, different HLTs used in obfuscated design have different individual features which are orthogonal to each other. Hence, accurate detection of structurally obfuscated design would be highly challenging. The proposed approach leverages the power of machine learning to train and classify structurally obfuscated HLS DSP designs using a set of distinct and unique formulated features.

1.1 Novel Contributions

- This paper presents a novel strategy for detecting structural obfuscation in HLS based DSP designs using machine learning framework.
- The paper presents a set of 26 unique and distinct features that can successfully identify and classify between structurally obfuscated DSP design and its non-obfuscated versions.
- The paper also presents a novel ML database comprising of several instances of structurally obfuscated and un-obfuscated DSP HLS designs that can be fed into our ML framework (logistic regression and random forest classifier). Results of the proposed approach revealed successful identification and detection of structurally obfuscated DSP designs with high accuracy.

2 RELATED WORKS

There has been no prior work on identification and detection of structural obfuscation in HLS based DSP designs using machine learning framework. However, there has been few works on structural obfuscation on DSP design. For example, authors in [22], presents an HLS method for creating DSP obfuscated circuits using high-level transformations

such as loop unrolling, tree height transformation, logic transformation etc. The aim was to design DSP circuits that have capability to hinder reverse engineering. Authors in [21], present a structural obfuscation approach with folding based algorithmic transformation, which is controlled with the help of an IP vendor selected initialization key. Further, authors in [26], presents an HLS based obfuscation technique by performing C-way partitioning. It offers strong hindrance against RTL alteration. Additionally, authors in [24], present an HLS based key-obfuscated RTL design with a strong design lockout mechanism. The structural obfuscation methodology is based on algorithmic transformation properties. Moreover, authors in [25], presents a key based structural obfuscation technique along with physical level watermarking to provide dual line of defense. Furthermore, authors in [27], present a structural obfuscation technique using high level transformations for JPEG CODEC hardware IP used in consumer electronics systems. Moreover, authors in [28], presented structural obfuscation technique in integration with DNA watermarking process for DSP designs. Authors in [23], present physical design obfuscation technique that perform alterations of circuit elements which are difficult to observe/analyze. Further, authors in [30] present high-level synthesis obfuscation at the algorithm level. The authors present several key-based transformations applied during component generation to make reverse engineering, with the key later provided to the circuit to unlock its functionality. Moreover, authors in [31], present constraint-based dynamic watermarking technique to generate the owner's signature, integrated with a logic encryption-based hardware obfuscation approach. Author in [32], present an HLS based computation partitioning for obfuscating IP that divides a design's computation into two parts. One representing commonly known operations shared across many designs, and other representing computations unique to the specific design. However, most of the prior structural obfuscation techniques such as [21] – [28] employ HLS based high level transformation mechanisms such as loop unrolling, tree height transformation, logic transformation for achieving architectural strength. Therefore, most of these structural obfuscation methods used for HLS based DSP designs can be detected using machine learning framework with the ulterior motive of deobfuscation. However, none of the prior techniques have discussed the vulnerability of structural obfuscation techniques to machine learning attack. Nevertheless, there has been limited effort on employing machine learning attack for HLS Trojan detection [29]. In this technique, authors applied decision tree classifier-based ML model for detecting various classes of HLS Trojan such as performance degradation hardware Trojan (HT), Denial of service HT, Functional HT etc. Therefore, none of the prior techniques present novel machine learning framework for detecting structural obfuscation in HLS based DSP designs.

3 PROPOSED METHODOLOGY

3.1 Background and Motivation

An HLS framework directly generates hardware IP designs from its high-level specifications (e.g. data flow graph). Structural obfuscation is a hardware security technique that transforms the internal structure/architecture of a DSP design into an unobvious one, without affecting the functionality in order to prevent an attacker from analyzing, interpreting and understanding the design during reverse-engineering. It disguises how the design operates by modifying the interconnections and structure. This makes the DSP design's architecture appear obscure and inconspicuous, while retaining its functionality. Therefore, structural obfuscation serves as a proactive defense mechanism that conceals the underlying architecture and interconnection topology of a DSP design, thereby hindering an attacker's ability to comprehend or analyze its internal organization. By deliberately introducing complexity and ambiguity into the structural design, this technique makes it exceedingly difficult for adversaries to deduce the circuit's true functionality or extract meaningful structure during reverse engineering [9] [11].

Structural obfuscation employs high level transformation (HLT) techniques such as Logic Transformation (LT), Tree Height Transformation (T-HT), and Loop Unrolling (LU) to conceal the true organization of a DSP design [25] [22] [21]. These methods modify the structural representation of the design by altering logic arrangements, adjusting hierarchical depth, and expanding iterative loops, making the circuit's internal architecture more complex and less recognizable to attackers attempting reverse engineering (RE).

Logic Transformation: LT modifies control data flow graph (CDFG) by replacing certain nodes or subgraphs with logically equivalent alternatives to create a structurally transformed graph which on subjecting to HLS converts into an obfuscated RTL that is different from its un-obfuscated version in terms of internal structure, internal connectivity and design components. The core idea is not to change the functionality of the design, but to alter its structural representation, making the design appear different from the original. By doing so, LT makes the DSP design unobvious and thereby increases the difficulty for an attacker to analyze the design, identify key computational paths, during RE attempt [22].

Tree Height Transformation: Another HLT technique used for structural obfuscation is Tree Height Transformation, which targets the CDFG by modifying its hierarchical structure specifically, by increasing or decreasing the height of the graph. The technique works by breaking down critical path dependencies into smaller, temporary sub-computations that can be evaluated in parallel. This restructuring changes the overall shape and depth of the graph, producing a structurally different yet functionally equivalent representation of the original design. The THT transformed graph on subjecting to HLS coverts into an obfuscated RTL that is different from its un-obfuscated version in terms of internal structure, internal connectivity and design components. The final outcome of this process is a THT-based structurally obfuscated design, which not only maintains the intended functionality but also adds a layer of complexity that increases the design's resilience against RE attempt [22] [25].

Loop Unrolling: Another HLT technique used for obfuscation is Loop Unrolling (LU), which targets the Control Data Flow Graph (CDFG) by expanding or "unwinding" loops. In this approach, the iterations of a loop are explicitly replicated, creating multiple copies of the loop body in sequence. This not only transforms the structural appearance of the graph but can also optimize execution timing by enabling parallelism and reducing loop control overhead. The LU graph on subjecting to HLS coverts into an obfuscated RTL that is different from its un-obfuscated version in terms of internal structure, internal connectivity and design components. The final output of this process is a loop-unrolling-based structurally obfuscated design, which preserves the intended behavior of the design while significantly increasing its complexity during RE attempt [21] [22].

From an attacker perspective, an attacker aims to identify structural obfuscation logic in DSP IP design during deobfuscation attempt. Detecting/identifying structural obfuscation logic is the essential part of deobfuscation in the reverse engineering process. However, detecting structural obfuscation is complex HLS generated DSP designs is non-trivial. This is because structural obfuscated DSP designs contain several convoluted features and transformations which do not affect the functionality but significantly alter the structural of the design such that it becomes not interpretable to an attacker. Such features appear like a normal interconnectivity logic in the RTL of the DSP design. Further, different structural transformation used in obfuscation have different features which are orthogonal to each other. Therefore, successful detection of structural obfuscation in complex is highly challenging. Hence, ML techniques can aid in successful detection of structural obfuscated design and classify non-obfuscated DSP design from its structurally obfuscated versions. ML techniques provide a powerful platform for classification/detection of structurally obfuscated designs from its baseline counterparts.

3.2 Proposed Structural Obfuscation Features

The proposed approach introduces a set of distinctive features that are used to identify and detect structural obfuscation in DSP designs using machine learning technique. Successful detection of structural obfuscation in DSP designs help to classify between obfuscated and un-obfuscated versions. As mentioned before, detecting/identifying structural obfuscation logic is the essential part of deobfuscation. The proposed formulated features that help distinguish between an un-obfuscated and structurally obfuscated DSP IP designs are shown in Table 1. Each proposed feature is explained in details in the following sub-section.

Features for detecting structurally obfuscated DSP design:

- Feature: ***Series_of_back_to_back_interconnectivity_demux_o/p_of_adder_connected_to_mux_i/p_of_adder***

This proposed feature detects *loop unrolling based structural obfuscation* employed in several obfuscation techniques [22] [25] [21]. The proposed feature is an indicative of loop unrolling logic where output from adder is connected to the input of adder (same/different depending on the allocation chosen by the IP vendor) in series of back-to-back interconnectivity. More explicitly in HLS generated RTL datapath if there is a series of back-to-back interconnectivity from demux output of adder connected to the mux input of adder, then it is an indicator of loop unrolling based structural obfuscation, that has been implemented by the IP vendor. We exploit this specific feature in our machine learning model to detect *loop unrolling based structural obfuscation* in any DSP design.

TABLE 1
FORMULATED FEATURES OF STRUCTURAL UN-OBFUSCATED AND OBFUSCATED IP DESIGNS

Formulated features from HLS generated RTL IP design (unique 26 features)	**Non-obfuscated / Obfuscated HLS IP design type**
2*1 Mux, 4*1 Mux, 8*1 Mux, 16*1 Mux, 32*1 Mux, 1*2 DeMux, 1*4 DeMux, 1*8 DeMux, 1*16 DeMux, 1*32 DeMux, Latch, Adder, Subtractor, Multiplier	Non-obfuscated
2*1 Mux, 4*1 Mux, 8*1 Mux, 16*1 Mux, 32*1 Mux, 1*2 DeMux, 1*4 DeMux, 1*8 DeMux, 1*16 DeMux, 1*32 DeMux, Latch, Adder, Subtractor, Multiplier, **Series_of_back_to_back_interconnectivity_demux_output_of_adder_connected_to_mux_input_of_adder** (*Obs. Feature 1*)**, Same_register_output_connected_to_same_mux_dual_inputs_corresponding_to_same_FU** (*Obs. Feature 5*)**, Same_register_output_connected_to_mux_inputs_corresponding_to_different_FU** (*Obs. Feature 6*)	Loop Unrolling (LU)
2*1 Mux, 4*1 Mux, 8*1 Mux, 16*1 Mux, 32*1 Mux, 1*2 DeMux, 1*4 DeMux, 1*8 DeMux, 1*16 DeMux, 1*32 DeMux, Latch, Adder, Subtractor, Multiplier, **Two_demux_outputs_of_adder_connected_to_mux_input_of_adder** (*Obs. Feature 2*)**, Two_demux_outputs_of_multiplier_connected_to_mux_input_of_adder** (*Obs. Feature 7*)	Tree Height Transformation (THT)
2*1 Mux, 4*1 Mux, 8*1 Mux, 16*1 Mux, 32*1 Mux, 1*2 DeMux, 1*4 DeMux, 1*8 DeMux, 1*16 DeMux, 1*32 DeMux, Latch, Adder, Subtractor, Multiplier, **Two_demux_outputs_from_same_FU_connected_to_mux_inputs_of_FU** (*Obs. Feature 3*)**, Demux_output_from_adder_connected_to_mux_input_of_adder_and_another_input_from_register** (*Obs. Feature 4*)	Logic Transformation (LT)
Broader_redundancy_padding_patterns, Repeated_look_alike_substructures_and_symmetry, Local_connectivity_density_spikes_and_hub_style_wiring, Reconvergence_patterns, Cycle_feedback_complexity (*Obs. Feature 8 to 12*)	LU/THT/LT

Figure 1 shows an example of scheduled data flow graph (SDFG) of a finite impulse response (FIR) filter for unrolling factor (UF) = 7 scheduled using 4 multipliers and 1 adder. The red encircled portion indicates the non-unrolled (baseline) CDFG of FIR filter which on subjecting to HLS does not provide any structural obfuscation. The SDFG of LU based structurally obfuscated FIR is adopted from [25]. The specific datapath portion (shown in Figure 1) which is structurally obfuscated due to LU is indicated by *Obf. Feature 1*. As evident, a series of back-to-back interconnections (indicated in red) between the demultiplexer (demux) outputs denoted as (y_1, y_2, y_3, …) of the adder (A1) and the multiplexer (mux) input of A1, are clearly visible post LU based structural obfuscation implementation. This configuration reflects implementation of loop unrolling where the output of the adder is recursively fed into its mux

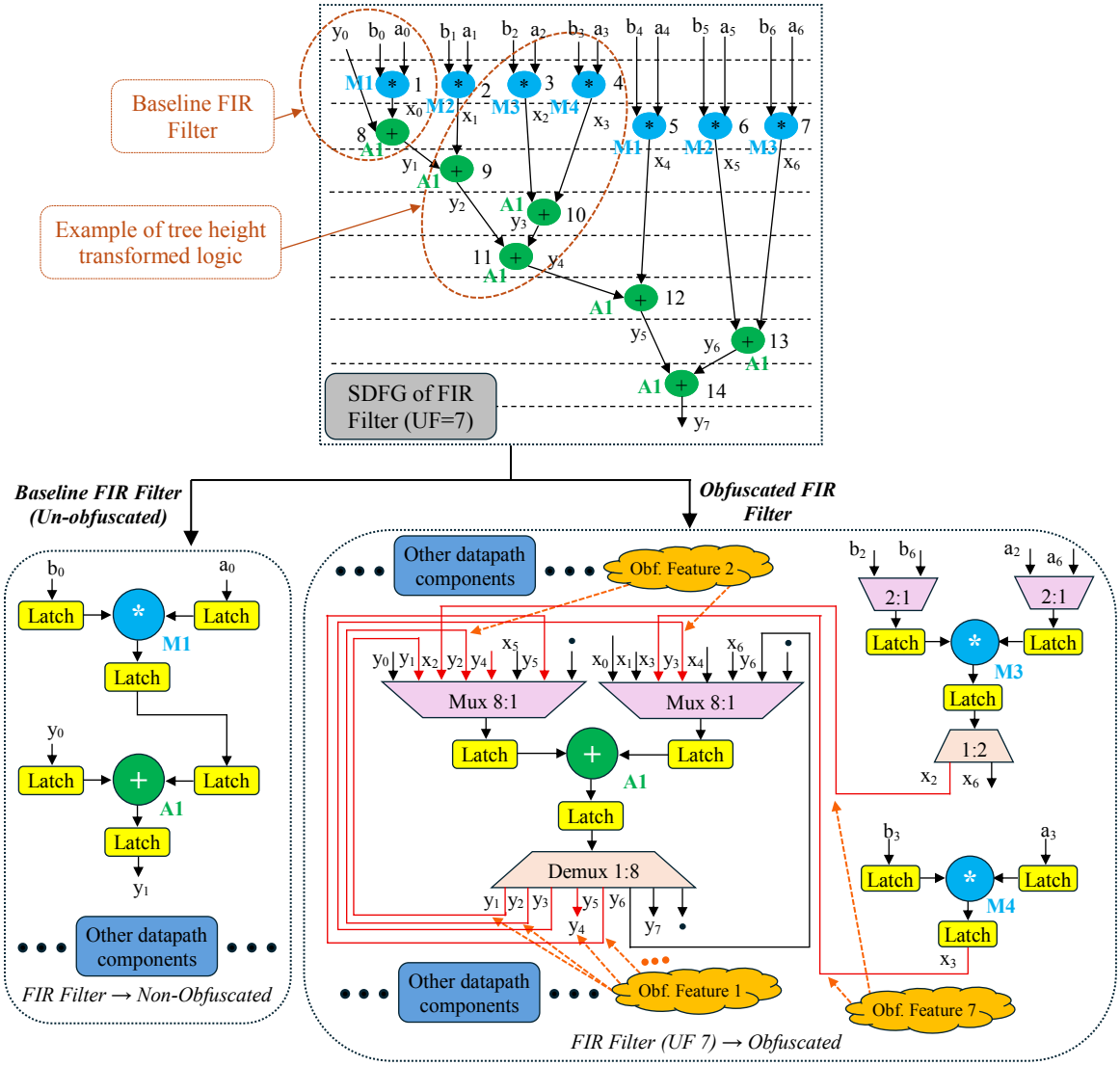


Figure 1: Structural obfuscation feature 1: Loop Unrolling (LU) and
Structural obfuscation feature 2 and feature 7: Tree Height Transformation (THT)

input. Therefore, feature 1 as part of our ML model can successfully detect any *LU based structural obfuscation* containing similar feature.

- Feature: ***"Two_demux_outputs_of_multiplier_connected_to_mux_input_of_adder" & "Two_demux_outputs_of_adder_connected_to_mux_input_of_adder"***

This proposed feature detects *THT based structural obfuscation* employed in several obfuscation techniques [22] [25] [21]. The proposed feature is an indicative of tree height transformed logic where both characteristics satisfy: (a) two demux outputs of multipliers are connected to the multiplexer input of the adder (b) demux outputs of adder is connected to the multiplexer input of the adder. More explicitly in HLS generated RTL datapath if there is two demux outputs from multiplier connected to the mux input of adder as well as demux outputs of adder connected to the multiplexer input of the adder, then it is an indicator of existences of *THT based structural obfuscation*, implemented by the IP vendor. We exploit this specific feature in our machine learning model to detect THT based structural obfuscation in any DSP design.

Figure 1 shows an example of THT employed SDFG of FIR filter. Non-THT version of FIR SDFG does not yield the impact of structural obfuscation due THT. The non-THT version of FIR SDFG is shown in [25] [18]. The specific datapath portion (shown in Figure 1) which is structurally obfuscated due to THT is indicated by *Obf. Feature 2 and 7*. As evident, the demux outputs from multipliers (M3 and M4) denoted as (x_2, x_3) are connected as inputs to the adder (A1) as well as the demux outputs of adder (A1) denoted as (y_2, y_3) are connected as inputs to the adder (A1) again (indicated in red), are clearly visible post THT based structural obfuscation realization. This configuration reflects implementation of *THT based structural obfuscation*.

- Feature: ***Two_demux_outputs_from_same_FU_connected_to_mux_inputs_of_FU***

This proposed feature detects logic transformation based structural obfuscation employed in several obfuscation techniques [22] [21] [25]. The proposed feature is an indicative of logic transformation where two demux outputs from same FU is connected to the multiplexer input of another FU. More explicitly in HLS generated RTL datapath if there is two demux outputs from same FU connected to the mux input of another FU, then it is an indicator of existences of logic transformation based structural obfuscation, implemented by the IP vendor. We exploit this specific feature in our machine learning model to detect logic transformed based structural obfuscation in any DSP design.

Figure 2 shows an example of SDFG of a cardiac pacemaker filter scheduled using 1 multiplier and 1 adder. The red encircled portion indicates the post LT (baseline) CDFG of cardiac pacemaker filter which on subjecting to HLS does not provide any structural obfuscation. The SDFG of LT based structurally obfuscated cardiac pacemaker is adopted from [14] [15]. The specific datapath portion (shown in Figure 2) which is structurally obfuscated due to LT is indicated by *Obf. Feature 3*. As evident, the demux output from multiplier (M1) denoted as (J^8) is connected as inputs to the adder (A1) (indicated in red), are clearly visible post LT based structural obfuscation realization. This configuration reflects implementation of *LT based structural obfuscation*.

- Feature: ***Demux_o/p_from_adder_connected_to_mux_input_of_adder_and_another_i/p_from_register***

This proposed feature detects logic transformation based structural obfuscation employed in several obfuscation techniques [22] [25] [21]. The proposed feature is an indicative of logic transformation where demux output from an FU is connected to the multiplexer input of same FU while another input comes from register output. More explicitly in HLS generated RTL datapath if there is demux output from adder connected to the mux input of same adder and another input comes from register output, then it is another indicator of existences of logic transformation based structural obfuscation, implemented by the IP vendor. We exploit this specific feature in our machine learning model to detect logic transformed based structural obfuscation in any DSP design.

Figure 3 shows an example of SDFG of a test case scheduled using 1 multiplier and 3 adders. The red encircled portion indicates the post-LT CDFG of test case. The SDFG of LT based structurally obfuscated test case is adopted from [25]. The specific datapath portion (shown in Figure 3) which is structurally obfuscated due to LT is indicated by *Obf. Feature 4*. As evident, the demux output from adder (A1) denoted as (x1´) are connected as mux input to the same A1 while another input comes from register (u) (indicated in red), are clearly visible post LT based structural obfuscation realization. This configuration reflects another implementation of *LT based structural obfuscation.*

- Feature: ***Same_register_output_connected_to_same_mux_dual_inputs_corresponding_to_same_FU***

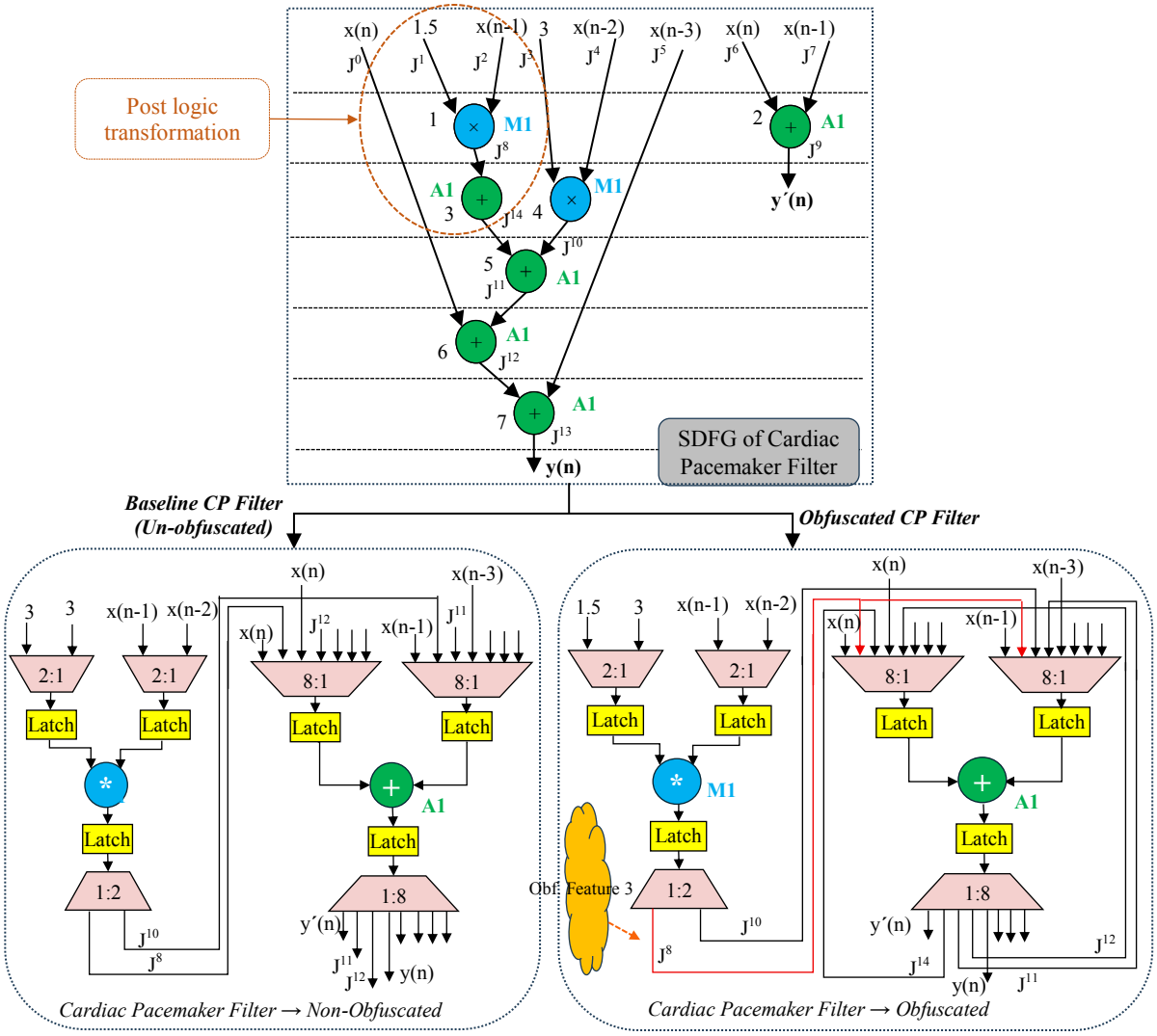


Figure 2: Structural obfuscation feature 3: Logic Transformation (LT)

This proposed feature detects loop unrolling based structural obfuscation employed in several obfuscation techniques [21] [22] [25]. The proposed feature is an indicative of LU where same register outputs are connected to the dual inputs of mux corresponding to same FU. More explicitly in HLS generated RTL datapath if there is same register outputs connected to the mux inputs of adder, then it is another indicator of existences of loop unrolling based structural obfuscation, implemented by the IP vendor. We exploit this specific feature in our machine learning model to detect

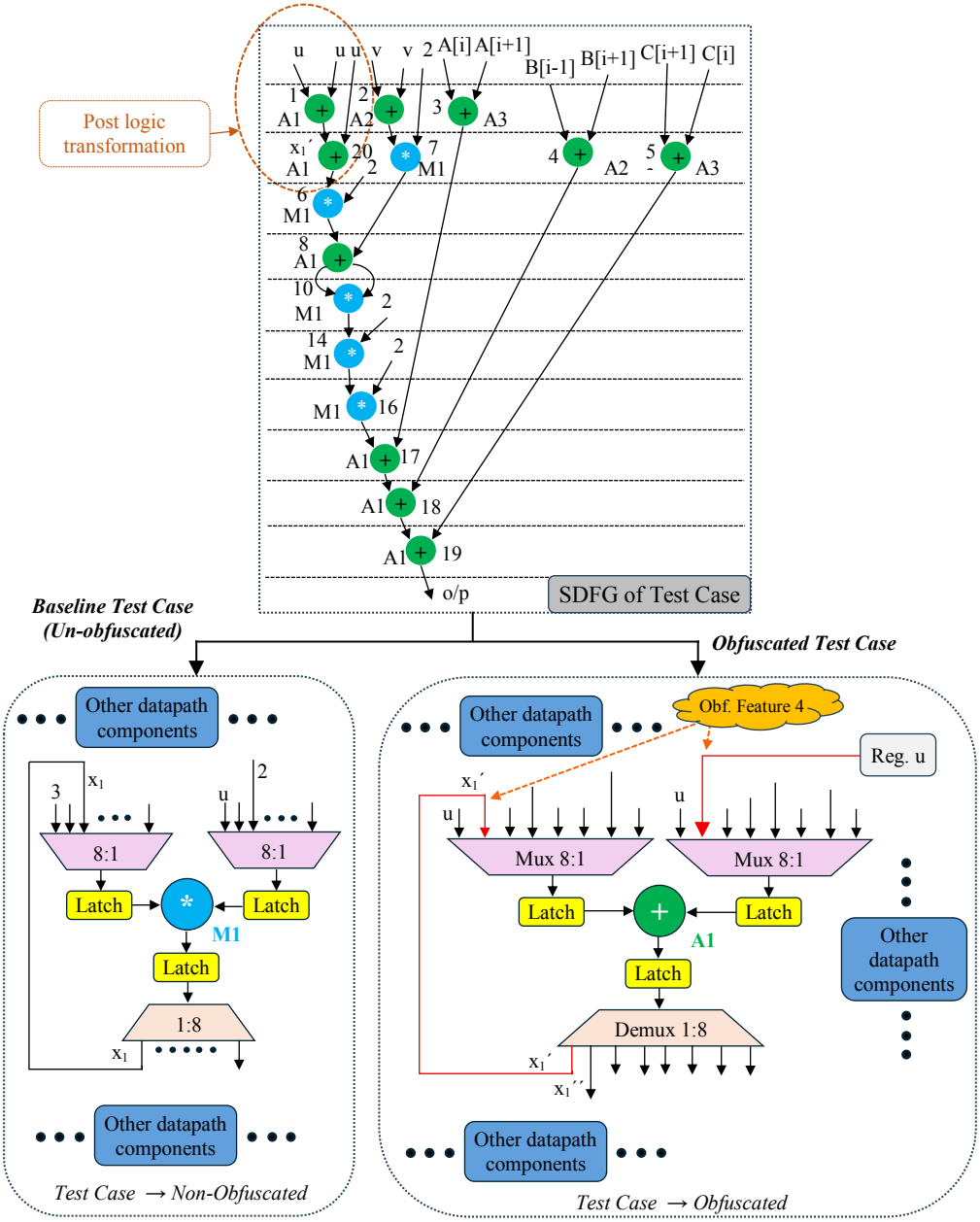


Figure 3: Structural obfuscation feature 4: Logic Transformation (LT)

logic transformed based structural obfuscation in any DSP design.

Figure 4 shows an example of SDFG of a test case scheduled using 1 multiplier and 3 adders. The SDFG of LU based structurally obfuscated test case is adopted from [22]. The specific datapath portion (shown in Figure 4) corresponding to the blue encircled portion which is structurally obfuscated using LU, is indicated by *Obf. Feature 5*. As evident, the

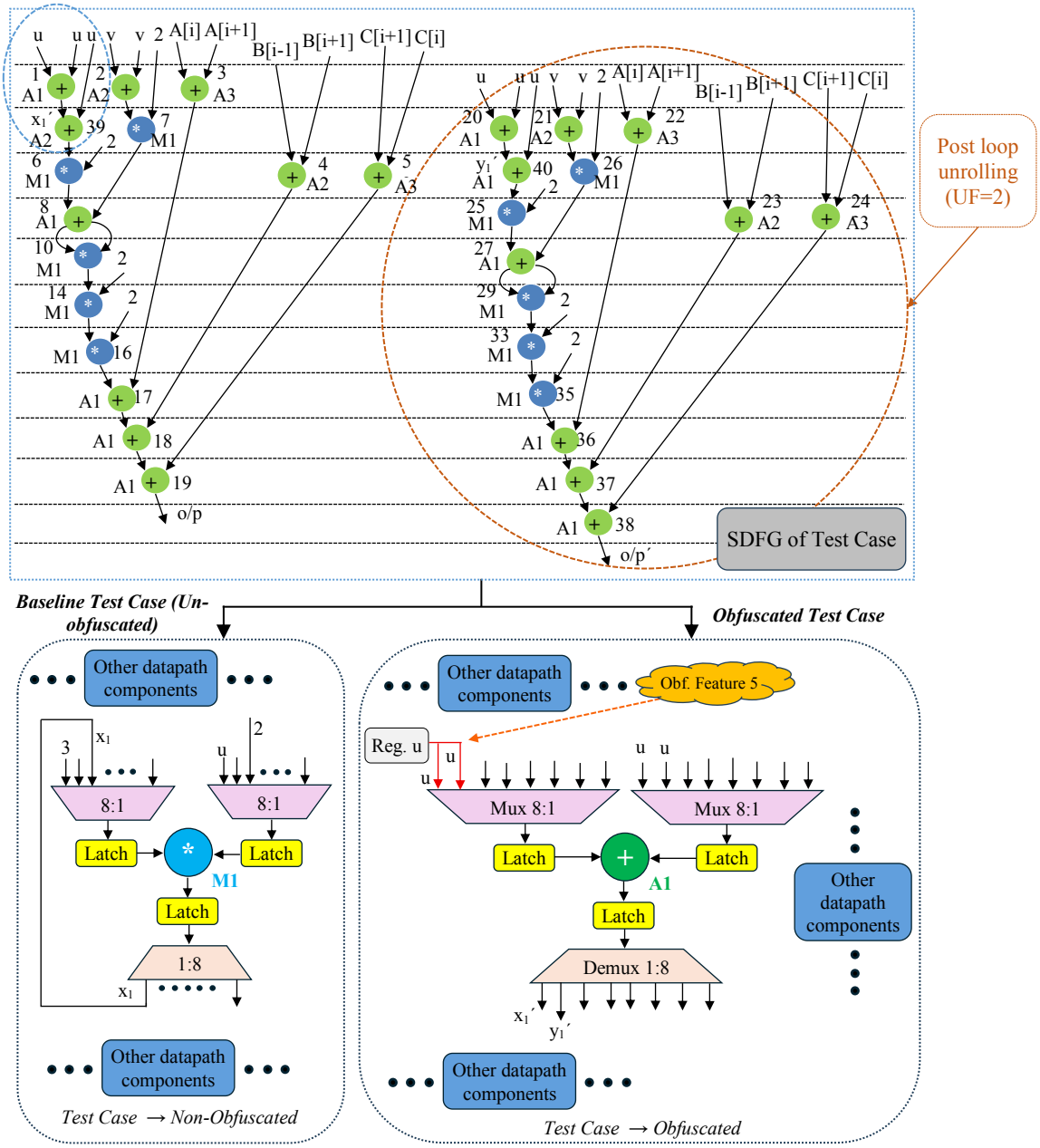


Figure 4: Structural obfuscation feature 5: Loop Unrolling (LU)

Note: The partial datapath shown above is reflecting the encircled portion of implementing partial LU logic post LT (in blue)

output of register u is connected to the same mux dual inputs of adder A1 (indicated in red), are clearly visible post LU based structural obfuscation realization. This configuration reflects another implementation of *LU based structural obfuscation*.

- Feature: ***Same_register_output_connected_to_mux_inputs_corresponding_to_different_FU***

This proposed feature detects loop unrolling based structural obfuscation employed in several obfuscation techniques [22] [21] [25]. The proposed feature is an indicative of LU where same register output is connected to the inputs of mux corresponding to different FU. More explicitly in HLS generated RTL datapath if there is same register output connected to the mux inputs of different adders, then it is another indicator of existences of loop unrolling based structural obfuscation, implemented by the IP vendor. We exploit this specific feature in our machine learning model to detect logic transformed based structural obfuscation in any DSP design.

Figure 5 shows an example of SDFG of a test case. The SDFG of LU based structurally obfuscated test case is adopted from [22]. The specific datapath portion (shown in Figure 5) which is structurally obfuscated due to LU is indicated by *Obf. Feature 6*. As evident, the output of register u is connected to the mux input of the A1 and A2 (indicated in red), are clearly visible post LU based structural obfuscation realization. This configuration reflects another implementation of *LU based structural obfuscation*.

- Feature: ***Two_demux_outputs_of_multiplier_connected_to_mux_input_of_adder***

This proposed feature detects THT based structural obfuscation employed in several obfuscation techniques [22] [21] [25]. The proposed feature is an indicative of THT where two demux output of different FU is connected to mux input of a different FU. More explicitly in HLS generated RTL datapath if there is two different demux outputs of multipliers connected to the mux input of adders, then it is another indicator of existences of THT based structural obfuscation, implemented by the IP vendor. We exploit this specific feature in our machine learning model to detect logic transformed

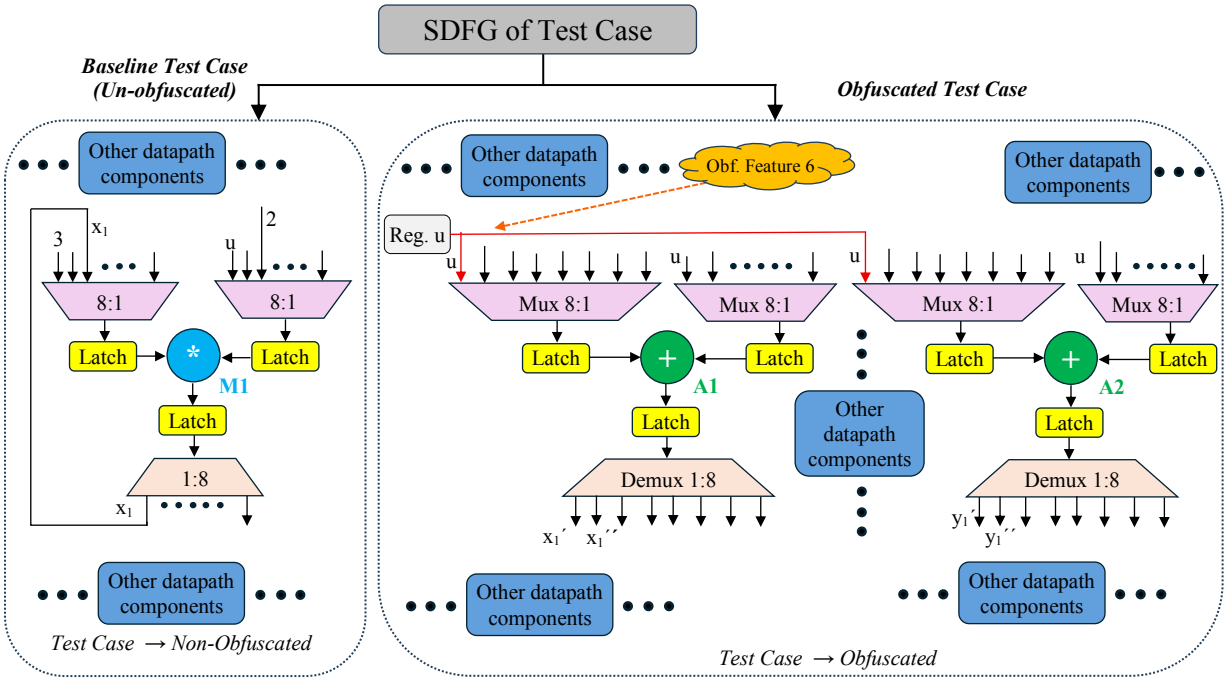


Figure 5: Structural obfuscation feature 6: Loop Unrolling (LU)
Note: (Addition operations '1' and '20' scheduled in same CS but allocated through different adders (A1 and A2))

based structural obfuscation in any DSP design.

Figure 6 shows an example of SDFG of an 8-Point DCT scheduled using 4 multipliers and 1 adder. The red encircled

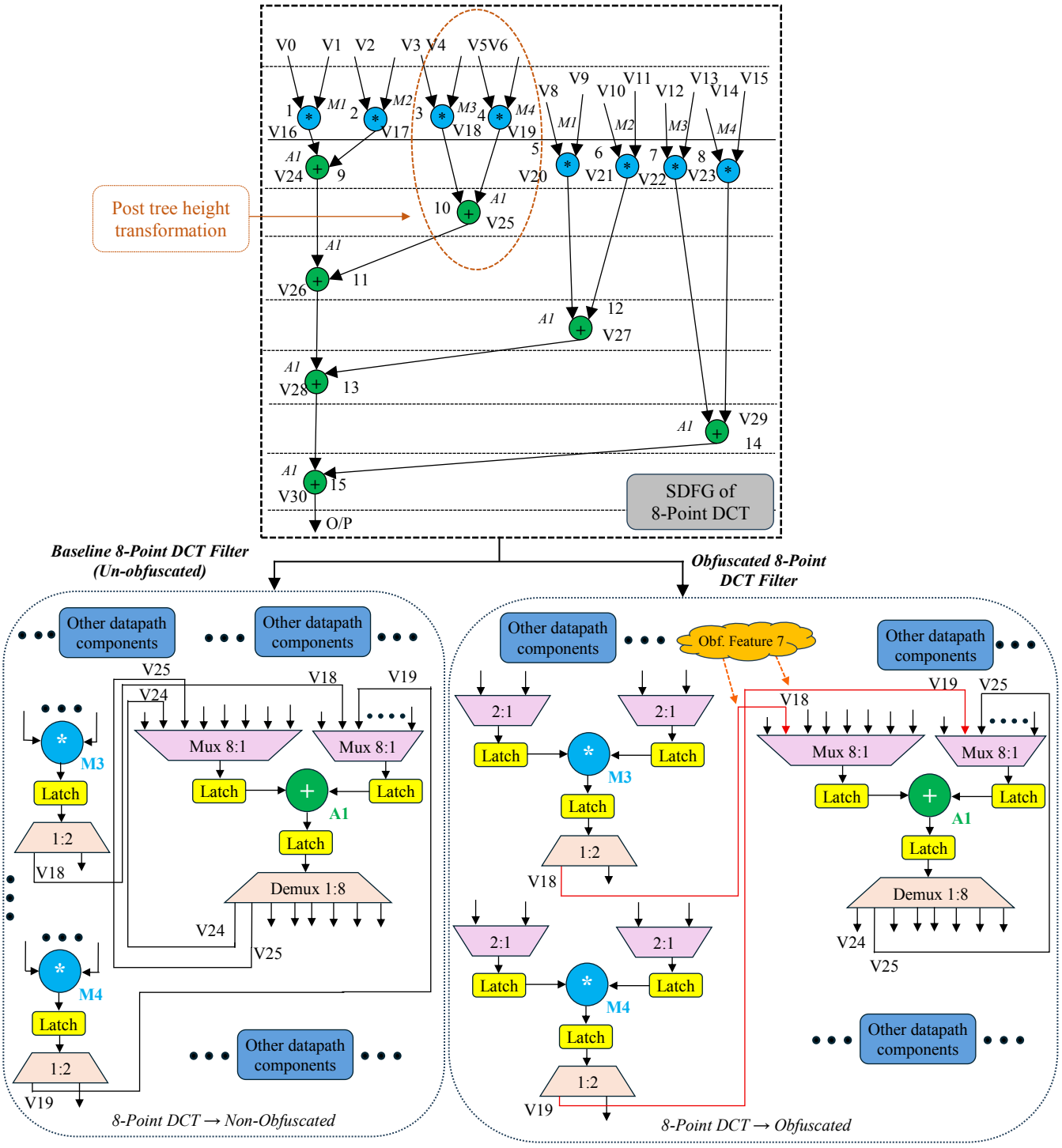


Figure 6: Structural obfuscation feature 7: Tree Height Transformation (THT)

portion indicates the post THT of 8-Point DCT. The SDFG of THT based structurally obfuscated 8-Point DCT is adopted from [17] [27]. The specific datapath portion (shown in Figure 6) which is structurally obfuscated due to THT is indicated by *Obf. Feature 7*. As evident, two demux outputs of multipliers (M3 and M4) denoted as (V18 and V19) connected to the mux inputs of adder (A1) (indicated in red), are clearly visible post THT based structural obfuscation realization. This configuration reflects another implementation of *THT based structural obfuscation*.

- ***Broader structural obfuscation features***

As indicated in Table 1 earlier, besides the specific structural obfuscation features *Obf. Feature 1- Obf. Feature 7*, there are additional broader structural obfuscation features *Obf. Feature 8- Obf. Feature 12* that can be considered. The list of features is enumerated below:

Feature: ***Broader_redundancy_padding_patterns*** - The proposed feature detects when unused redundant input ports are generated in the input mux architecture corresponding to a specific FU. The presence of this structural obfuscation feature can be observed in the obfuscated datapath.

Feature: ***Repeated_look_alike_substructures_and_symmetry*** – The proposed feature detects when similar substructures, such as same size input mux architectures (look alike) exist symmetrically corresponding to a specific FU. The presence of this structural obfuscation feature is observed in Figure 1 in case of FIR filter (UF=7), where repeated 8:1 mux architecture is symmetrically present corresponding to FU A1.

Feature: ***Local_connectivity_density_spikes_and_hub_style_wiring*** – The proposed feature detects when a specific input causes connectivity density spike resulting into a hub style wiring architecture of the mux logic in the datapath. For example, in Figure 3, input *Reg. u* is used multiple times in the mux logic causing connectivity density spike in the mux architecture.

Feature: ***Reconvergence_patterns*** - The proposed feature detects when a convergent pattern of architecture exists in the IP datapath.

Feature: ***Cycle_feedback_complexity*** – The proposed feature detects when single or multiple outputs emanate from demux architecture of a specific FU and fed back cyclically as input into the mux architecture of the same FU. The presence of this obfuscation feature is observed in Figure 1 in case of FIR filter (UF=7).

3.3 Proposed Machine Learning Technique

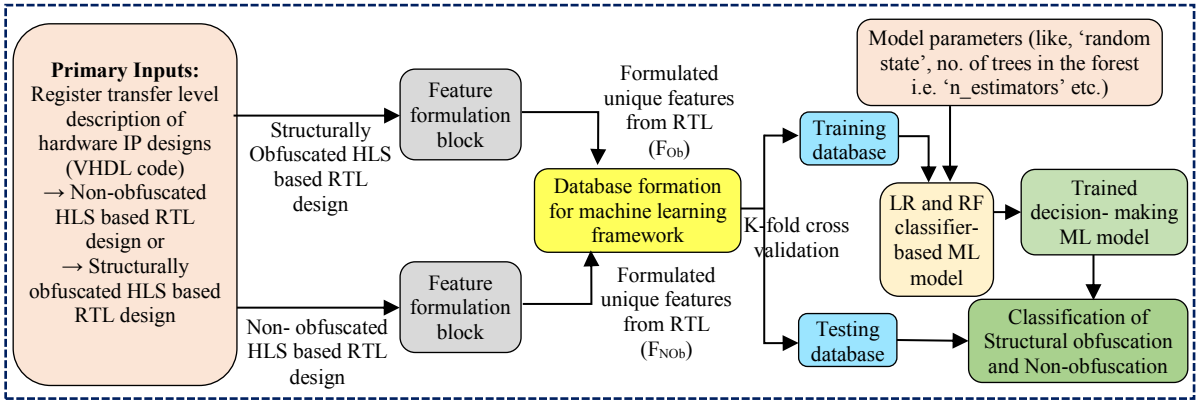


Figure 7: Proposed ML-based detection methodology of HLS driven structurally obfuscated design

TABLE 2.A
AUTOMATED EXTRACTION OF STRUCTURALLY OBFUSCATED FEATURES FROM OBFUSCATED VHDL DESIGNS USING PROPOSED FEATURE EXTRACTION ALGORITHM (IMPLEMENTED IN PYTHON)

Obfuscated DSP Designs\Structurally Obfuscated Features	**Obs. Feature 1**	**Obs. Feature 2**	**Obs. Feature 3**	**Obs. Feature 4**	**Obs. Feature 5**	**Obs. Feature 6**	**Obs. Feature 7**
FIR_Filter_UF7_Ob	✓	✓	×	×	×	×	✓
Cardiac_Pacemaker_Filter_Ob	×	×	✓	×	×	×	×
4Point_DCT_Ob	×	✓	×	×	×	×	✓
TestCase_CDFG_Ob	×	×	×	✓	✓	✓	×
IIR_Butterworth_Filter_Ob	×	✓	×	×	×	×	×
8Point_DCT_Ob	×	✓	×	×	×	×	✓
Image_Filter_3x3_Ob	×	✓	×	×	×	×	✓
Image_Filter_5x5_Ob	×	✓	×	×	×	×	✓
Sharpening_Filter_Ob	×	×	✓	×	×	×	×
Blur_Filter_Ob	×	×	×	×	×	×	×
JPEG_DCT_Core_Ob	×	✓	×	×	×	×	✓

Figure 7 illustrates the details of the proposed machine learning classifier (identification and detection mechanism) using logistic regression - LR and random forest - RF for HLS based structurally obfuscated DSP design. The primary inputs of the detection approach are the register transfer level (RTL) description of hardware IP designs (in the form of VHDL code), which includes non-obfuscated and structurally obfuscated RTL design. Initially, RTL description is processed through the feature formulation block which formulates unique non-obfuscated (F_{NOb}) and structurally obfuscated (F_{Ob}) features. The list of obfuscated features is provided below (please refer to Table 1 which maps the following features to respective structural obfuscation technique employed).

Obs. Feature 1: Series_of_back_to_back_interconnectivity_demux_o/p_of_adder_connected_to_mux_i/p_of_adder

Obs. Feature 2: Two_demux_outputs_of_adder_connected_to_mux_input_of_adder

Obs. Feature 3: Two_demux_outputs_from_same_FU_connected_to_mux_inputs_of_FU

Obs. Feature 4: Demux_o/p_from_adder_connected_to_mux_input_of_adder_and_another_input_from_register

TABLE 2.B
AUTOMATED EXTRACTION OF STRUCTURALLY OBFUSCATED FEATURES FROM OBFUSCATED VHDL DESIGNS USING PROPOSED FEATURE EXTRACTION ALGORITHM (IMPLEMENTED IN PYTHON)

Obfuscated DSP Designs\Structurally Obfuscated Features	**Obs. Feature 8**	**Obs. Feature 9**	**Obs. Feature 10**	**Obs. Feature 11**	**Obs. Feature 12**
FIR_Filter_UF7_Ob	✓	✓	✓	✓	✓
Cardiac_Pacemaker_Filter_Ob	×	×	✓	×	×
4Point_DCT_Ob	×	×	×	×	×
TestCase_CDFG_Ob	✓	✓	✓	×	×
IIR_Butterworth_Filter_Ob	×	×	×	×	×
8Point_DCT_Ob	×	×	×	×	×
Image_Filter_3x3_Ob	✓	✓	✓	✓	✓
Image_Filter_5x5_Ob	✓	✓	✓	✓	✓
Sharpening_Filter_Ob	×	×	×	×	×
Blur_Filter_Ob	×	×	×	×	×
JPEG_DCT_Core_Ob	×	×	×	×	×

Obs. Feature 5: Same_register_output_connected_to_same_mux_dual_inputs_corresponding_to_same_FU
Obs. Feature 6: Same_register_output_connected_to_mux_inputs_corresponding_to_different_FU
Obs. Feature 7: Two_demux_outputs_of_multiplier_connected_to_mux_input_of_adder
Obf. Feature 8: Broader_redundancy_padding_patterns
Obf. Feature 9: Repeated_look_alike_substructures_and_symmetry
Obf. Feature 10: Local_connectivity_density_spikes_and_hub_style_wiring
Obf. Feature 11: Reconvergence_patterns
Obf. Feature 12: Cycle_feedback_complexity

Subsequently, these formulated unique features are merged into a database, which is further passed through the k-fold cross validation method to divide the training and testing datasets, where 'k' defines the number of folds/groups that the dataset is divided into. The proposed approach uses LR and RF classifier independently to classify between non-obfuscated and structurally obfuscated DSP RTL designs. Finally, the trained decision-making LR/RF classifier-based ML model is able to detect the structurally obfuscated designs.

Motivation: In this approach, LR and RF classifier-based machine learning framework is utilized to detect the presence of structural obfuscated features in the RTL IP design. The LR is recognized as a lightweight classification model, making it highly suitable for a variety of learning tasks. Its primary advantage lies in its low computational overhead, which enables efficient processing of both small and large datasets. They are capable of capturing non-linear dependencies and feature interactions without requiring complex preprocessing procedures. On the other hand, RF is an ensemble classification technique that takes the advantage and strength of various decision trees to make the model more stable. RF can take care of both categorical and numerical data in efficient manner. It is also able to detect non-linear patterns among the features without any significant data change/transformation. In contrast, support vector machines (SVMs) employ kernel functions to address non-linear decision boundaries but tend to be less effective for categorical data. Unlike SVMs and multilayer perceptron, LRs and RFs do not require normalization or feature scaling, thereby simplifying data preparation. These classifiers are not much dependent on the hyperparameter tuning values like SVM, artificial neural network, and deep learning-based models. Moreover, deep learning-based model requires more resources, and they are computationally expensive. Therefore, considering the above facts, both of these ML classifiers are well suited for the classification problem targeted in this paper.

4 RESULTS AND ANALYSIS

The proposed combined non-obfuscated and structurally obfuscated feature database comprises of multiple instances of various DSP IP designs [14] [15] [18] [22] [25]. The details of all the IP designs along with their transfer functions and CDFGs are available publicly in [16] [17] [18]. The proposed feature formulation code and generated database is also available in [16]. In the proposed work, structurally obfuscated designs corresponding to LU, THT, and LT are considered. Subsequently, the proposed database includes 26 unique features (F_{NOb} and F_{Ob}) formulated corresponding to non-obfuscated and structurally obfuscated IP designs (shown in Table 1). The proposed 26 feature are encoded in binary format (0/1) in the database for various instances. In the database [16], 1st row defines the unique features (F_{NOb} and F_{Ob}) and remaining 61 row defines multiple instances related to different DSP IP designs. The overall database is divided into 40 instances of structurally obfuscated DSP IP designs and 21 instances of non-obfuscated DSP IP designs. The database comprises of 34 instances of LU based structurally obfuscated DSP design, 37 instances of THT based structurally obfuscated DSP design, and 4 instances of LT based structurally obfuscated DSP design. Here, 32 instances indicate the once which have both LU and THT based structural obfuscation present in the DSP design. Further, one instance

indicates a case which has both LU and LT based structural obfuscation present in the DSP design. Moreover, one instance indicates a case which has both THT and LT based structural obfuscation present in the DSP design. Further, the final column displays the label (non-obfuscated → 0, and Structurally Obfuscated → 1) of each instance.

TABLE 3
EVALUATION OF THE PROPOSED DETECTION USING LOGISTIC REGRESSION TECHNIQUE IN TERMS OF STANDARD PARAMETERS

K-fold validation	k=6	k=7	k=8	k=9	k=10
Accuracy	0.93	0.94	0.94	0.94	0.94
Precision	1.00	1.00	1.00	1.00	1.00
Recall	0.90	0.91	0.90	0.90	0.92
F1-score	0.95	0.95	0.95	0.94	0.95
Error rate	0.07	0.06	0.06	0.06	0.06

TABLE 4
EVALUATION OF THE PROPOSED DETECTION USING RANDOM FOREST CLASSIFICATION TECHNIQUE IN TERMS OF STANDARD PARAMETERS

K-fold validation	k=6	k=7	k=8	k=9	k=10
Accuracy	0.95	0.94	0.95	0.95	0.95
Precision	1.00	1.00	1.00	1.00	1.00
Recall	0.92	0.91	0.92	0.92	0.93
F1-score	0.96	0.95	0.95	0.95	0.96
Error rate	0.05	0.06	0.05	0.05	0.05

TABLE 5
STRUCTURAL OBFUSCATION DETECTION STATUS AND DETECTION TIME

HLS IP designs	FIR Filter, Cardiac Pacemaker (CP) Filter, Test Case CDFG, IIR Butterworth Filter, 4-Point DCT, Sharpening filter (SF), 8-Point DCT, Image Filter 3x3, Image Filter 5x5, Blur filter, Vertical Embossment Filter (VE), Convolutional layer (CNN), JPEG DCT Core, MESA, ECG GLRT, Pacemaker QRS
HLS structural obfuscation types	Loop Unrolling (LU), Tree Height Transformation (THT), and Logic Transformation (LT)
Detection status	Yes
Detection time	~ 505 ms

TABLE 6
EVALUATION OF THE PROPOSED DETECTION USING LOGISTIC REGRESSION TECHNIQUE IN TERMS OF MEAN AND STANDARD DEVIATION

	Mean					Standard Deviation				
K-fold validation	**k=6**	**k=7**	**k=8**	**k=9**	**k=10**	**k=6**	**k=7**	**k=8**	**k=9**	**k=10**
Accuracy	0.93	0.94	0.94	0.94	0.94	0.08	0.09	0.07	0.10	0.11
Precision	1.00	0.98	1.00	1.00	1.00	0.00	0.06	0.00	0.00	0.00
Recall	0.90	0.91	0.90	0.90	0.92	0.12	0.12	0.11	0.18	0.15
F1-score	0.95	0.95	0.95	0.94	0.95	0.07	0.07	0.06	0.11	0.09
Error rate	0.07	0.06	0.06	0.06	0.06	0.08	0.09	0.07	0.10	0.11

TABLE 7
EVALUATION OF THE PROPOSED DETECTION USING RANDOM FOREST CLASSIFICATION TECHNIQUE IN TERMS OF MEAN AND STANDARD DEVIATION

	Mean					Standard Deviation				
K-fold validation	**k=6**	**k=7**	**k=8**	**k=9**	**k=10**	**k=6**	**k=7**	**k=8**	**k=9**	**k=10**
Accuracy	0.95	0.94	0.95	0.95	0.95	0.08	0.09	0.07	0.10	0.11
Precision	1.00	1.00	1.00	1.00	1.00	0.00	0.00	0.00	0.00	0.00
Recall	0.92	0.91	0.92	0.92	0.93	0.12	0.12	0.12	0.18	0.14
F1-score	0.96	0.95	0.95	0.95	0.96	0.07	0.07	0.07	0.12	0.09
Error rate	0.05	0.06	0.05	0.05	0.05	0.08	0.09	0.07	0.10	0.11

The proposed structural obfuscation detection mechanism is implemented on Intel(R) Core (TM) i5-8250U CPU with

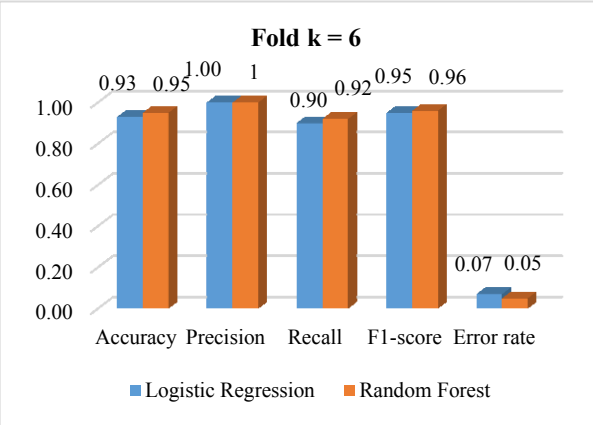


Figure 8: Variation in results for LR and RF classifier with respect to k = 6

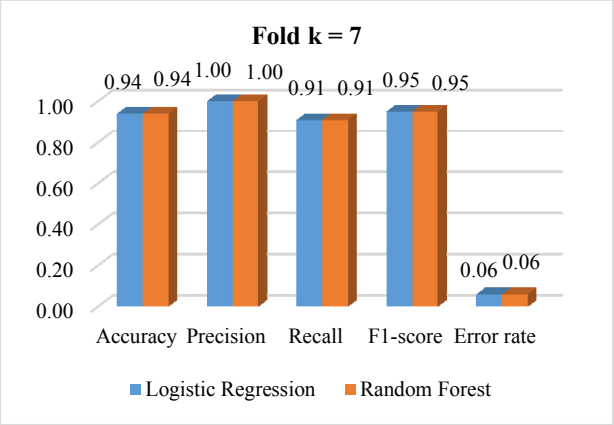


Figure 9: Variation in results for LR and RF classifier with respect to k = 7

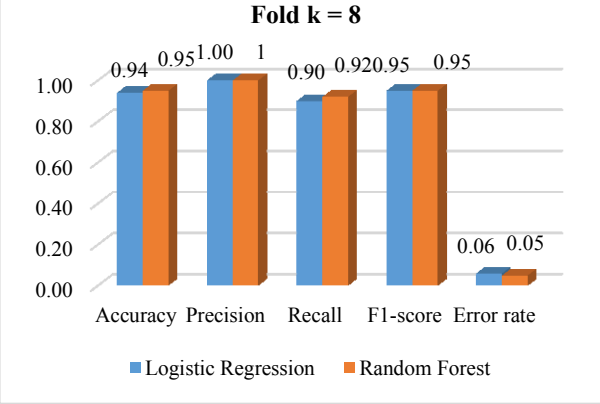


Figure 10: Variation in results for LR and RF classifier with respect to k = 8

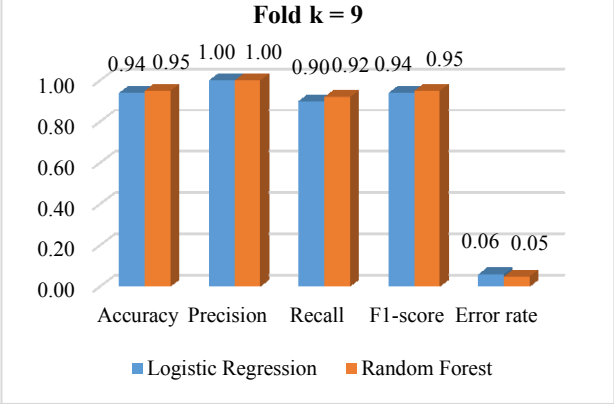
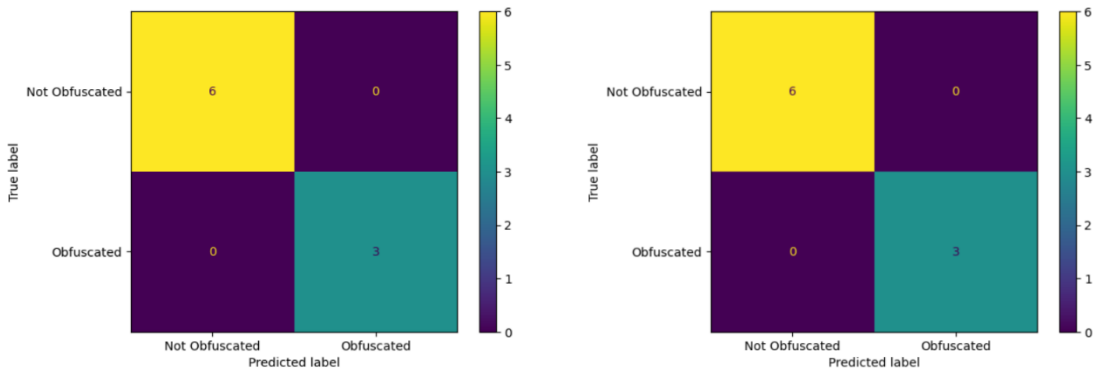


Figure 11: Variation in results for LR and RF classifier with respect to k = 9

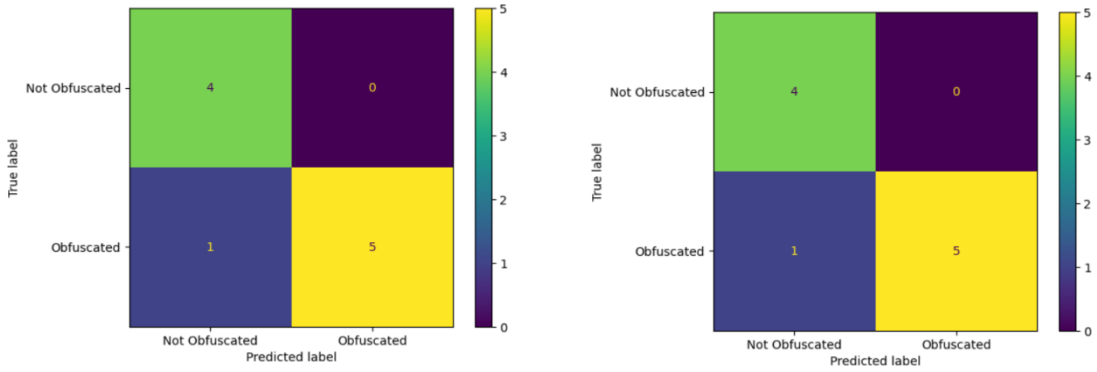
Figure 12: Variation in results for LR and RF classifier with respect to k = 10

8GB RAM and processor frequency of 1.60GHz. To implement LR and RF classifier-based ML model in the proposed



(a) RF classifier (b) LR

Figure 13. Confusion matrix for train-test split ratio of 85:15 for RF classifier and LR



(a) RF classifier (b) LR

Figure 14. Confusion matrix (with FN>0% indicating misclassification) for RF classifier and LR when the training and testing instances comprise of variants of the same underline IP design during train: test split ratio of 85:15

TABLE 8
EVALUATION OF THE PROPOSED DETECTION USING RF CLASSIFIER FOR DIFFERENT VALUES OF N_ESTIMATORS AS HYPERPARAMETER TUNING

	n_estimators=50					n_estimators=75					n_estimators=100				
K-fold validation	**k=6**	**k=7**	**k=8**	**k=9**	**k=10**	**k=6**	**k=7**	**k=8**	**k=9**	**k=10**	**k=6**	**k=7**	**k=8**	**k=9**	**k=10**
Accuracy	0.95	0.94	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.95
Precision	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Recall	0.92	0.91	0.92	0.92	0.93	0.92	0.93	0.92	0.92	0.93	0.92	0.93	0.92	0.92	0.93
F1-score	0.96	0.95	0.95	0.95	0.96	0.96	0.96	0.95	0.95	0.96	0.96	0.96	0.95	0.95	0.96
Error rate	0.05	0.06	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05

TABLE 9
ABLATION STUDY OF THE PROPOSED DETECTION USING RF CLASSIFIER

Parameters	**Obs. Feature 1-12**	**Obs. Feature 1-7 (removing Obs. Features 8-12)**	**Obs. Feature 3-7 (removing Obs. Features 1, 2, 8-12)**	**Obs. Feature 3-12 (removing Obs. Features 1, 2)**
Accuracy	1.00	1.00	0.90	0.90
Precision	1.00	1.00	1.00	1.00
Recall	1.00	1.00	0.66	0.83
F1-score	1.00	1.00	0.80	0.90
Error rate	0.00	0.00	0.10	0.10

mech anism, we have used jupyter notebook within the Anaconda distribution framework and scikit-learn [19] along with the python environment. The feature (F_{NOb} and F_{Ob}) formulation process is also performed automatically using python programming language (feature formulation time is ~25 ms).

4.1 Binary Classification

Table 2.A and 2.B shows the corresponding automated feature extraction from structurally obfuscated VHDL codes. Here, Table 3 shows the results/performances of the proposed detection mechanism in terms of standard evaluation parameters with k-fold cross validation (k = 6 to 10) for LR-based ML model. As evident from Table 2.A and 2.B, the accuracy of detecting structurally obfuscated design is ~0.95 for three structurally obfuscated design types discussed here. Similarly, Table 4 shows the results for RF classifier-based ML model. As evident from Table 3, the accuracy of detecting structurally obfuscated design ranges from ~0.91 to ~0.94. *Note: Here, the number of trees in the forest (i.e., 'n_estimators')=50, which is one main hyperparameter in RF classifier.* Furthermore, Table 5 depicts the detection status and the detection time of the proposed approach corresponding to various HLS DSP IP designs, with respect to the proposed LR/RF classifier-based ML method. The detection time of the proposed mechanism is approximately 505 milliseconds. Note: The standard hyperparameters utilized in the LR and RF classifier-based ML model are random_state=0, ccp_alpha=0.0, criterion='gini', min_impurity_decrease=0.0, min_samples_leaf=1, min_samples_split=2, splitter='best', n_estimators=50, shuffle=True, and n_splits=k (where, k = 6 to 10). Figure 8 –

TABLE 10
EVALUATION OF THE PROPOSED DETECTION FOR MULT-LABEL CLASSIFICATION USING RF CLASSIFIER IN TERMS OF STANDARD PARAMETERS (*HYPERPARAMETER RANDOM_STATE =0*)
NOTE: NON-OBFUSCATED → 0, LU-BASED STRUCTURALLY OBFUSCATED → 1, THT-BASED STRUCTURALLY OBFUSCATED → 2

Multi-label classification	Accuracy	Precision	Recall	F1-score
0	0.90	0.80	1.00	0.89
1		1.00	0.80	0.89
2		1.00	1.00	1.00

TABLE 11
EVALUATION OF THE PROPOSED DETECTION FOR MULT-LABEL CLASSIFICATION USING RF CLASSIFIER IN TERMS OF STANDARD PARAMETERS
(*HYPERPARAMETER RANDOM_STATE =60*)

Multi-label classification	Accuracy	Precision	Recall	F1-score
0	1.00	1.00	1.00	1.00
1		1.00	1.00	1.00
2		1.00	1.00	1.00

TABLE 12
EVALUATION OF THE PROPOSED DETECTION FOR MULT-LABEL CLASSIFICATION USING RF CLASSIFIER IN TERMS OF STANDARD PARAMETERS
(*HYPERPARAMETER RANDOM_STATE =75*)

Multi-label classification	Accuracy	Precision	Recall	F1-score
0	1.00	1.00	1.00	1.00
1		1.00	1.00	1.00
2		1.00	1.00	1.00

Figure 12 shows the variation of the proposed LR vs. RF classifier for parameters accuracy, precision, recall, F1-score and error rate with respect to k = 6 to 10. The mean and standard deviation across folds is reported in Tables 6 and 7. Table 8 reports the evaluation of the proposed detection using RF classifier for different values of *n_estimators* as hyperparameter tuning. Table 9 reports the ablation study of the proposed detection using RF classifier highlighting the importance of obfuscation features. As evident from Table 9, Obs. Features 1, 2, 8-12 are very critical in providing accurate classification as removal of these obfuscation features results in degraded accuracy and other parameters. Similarly, Obs. Features 1-2 are very important, as removal of these feature also results into degraded performance.

Figure 13 shows the confusion matrix for train-test split ratio of 85:15 for RF classifier and LR where no variance of the same design is appearing in both train and test instances. As evident from the results, the FP and FN rate is 0%. Figure 14 shows confusion matrix (with FN>0% indicating misclassification) for RF classifier and LR when the training and testing instances comprise of variants of the same underline IP design during train: test split ratio of 85:15.

4.2 Multi-label Classification

For multi-label classification of structurally obfuscated DSP IP designs, the database (the final column) is labelled as follows: non-obfuscated → 0, LU-based Structurally Obfuscated → 1, THT-based Structurally Obfuscated → 2, and LT-based Structurally Obfuscated → 3 for various instances.

TABLE 13
EVALUATION OF THE PROPOSED DETECTION FOR MULT-LABEL CLASSIFICATION USING LR IN TERMS OF STANDARD PARAMETERS
(*HYPERPARAMETER RANDOM_STATE =0*)
NOTE: NON-OBFUSCATED → 0, LU-BASED STRUCTURALLY OBFUSCATED → 1, THT-BASED STRUCTURALLY OBFUSCATED → 2

Multi-label classification	Accuracy	Precision	Recall	F1-score
0	0.90	0.80	1.00	0.89
1		1.00	0.80	0.89
2		1.00	1.00	1.00

TABLE 14
EVALUATION OF THE PROPOSED DETECTION FOR MULT-LABEL CLASSIFICATION USING LR IN TERMS OF STANDARD PARAMETERS
(*HYPERPARAMETER RANDOM_STATE =42*)

Multi-label classification	Accuracy	Precision	Recall	F1-score
0	1.00	1.00	1.00	1.00
1		1.00	1.00	1.00
2		1.00	1.00	1.00

TABLE 15
EVALUATION OF THE PROPOSED DETECTION FOR MULT-LABEL CLASSIFICATION USING LR IN TERMS OF STANDARD PARAMETERS
(*HYPERPARAMETER RANDOM_STATE =75*)

Multi-label classification	Accuracy	Precision	Recall	F1-score
0	1.00	1.00	1.00	1.00
1		1.00	1.00	1.00
2		1.00	1.00	1.00

TABLE 16
ABLATION STUDY OF THE PROPOSED DETECTION USING RF CLASSIFIER FOR MULTI-LABEL CLASSIFICATION

	Obs. Feature 1-12				Obs. Feature 1-7 (removing Obs. Features 8-12)			
Multi-label classification	**Accuracy**	**Precision**	**Recall**	**F1-score**	**Accuracy**	**Precision**	**Recall**	**F1-score**
0	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
1		1.00	1.00	1.00		1.00	1.00	1.00
2		1.00	1.00	1.00		1.00	1.00	1.00

	Obs. Feature 3-7 (removing Obs. Features 1, 2, 8-12)				Obs. Feature 3-12 (removing Obs. Features 1, 2)			
Multi-label classification	**Accuracy**	**Precision**	**Recall**	**F1-score**	**Accuracy**	**Precision**	**Recall**	**F1-score**
0	0.90	0.88	1.00	0.93	0.90	0.83	1.00	0.91
1		0.00	0.00	0.00		1.00	1.00	1.00
2		1.00	1.00	1.00		0.00	0.00	0.00

Tables 10, 11 and 12 show the evaluation of the proposed RF classifier-based detection for multi-label classification in terms of standard parameters (with different hyperparameter (random_state) tuning). Similarly, Tables 13, 14 and 15 show the evaluation of the proposed LR based detection for multi-label classification in terms of standard parameters (with different hyperparameter (random_state) tuning). *Note*: Since the database has a single instance of LT-based structurally obfuscated design (label 3) which is included in the training database by the ML frameworks, hence it is not displayed in the tables as part of the testing database.

Table 16 reports the ablation study of the proposed detection using RF classifier for multi-label classification highlighting the importance of obfuscation features. As evident from Table 16, Obs. Features 1, 2 are very critical in providing accurate classification as removal of these obfuscation features results in degraded accuracy and other parameters. However, Obs. Features 8-12 are generic and broader by nature, hence removal of them don't impact the classification accuracy, as evident from Table 16.

5 CONCLUSION AND FUTURE WORK

This paper presented a novel strategy for detecting structural obfuscation in HLS based DSP IP designs using LR and RF classifier-based ML framework. The proposed approach formulated set of 26 unique and distinct features that can successfully classify between structurally obfuscated design and its non-obfuscated versions, which fed into our ML framework. The accuracy of detecting structurally obfuscated DSP designs obtained is found to be high.

One of our future works would aim to expand the feature set for structurally obfuscated DSP designs such that it can encompass more structural transformation techniques beyond LU, THT and LT. We also would like to expand our ML database with more instances of structurally obfuscated DSP designs with different functionally equivalent structural transformation techniques.

REFERENCES

[1] H. Yang, N. Basutkar, P. Xue, K. Kim, and Y. H. Park, "Software defined DVT-T2 demodulator using scalable DSP processors," IEEE Trans. on Consumer Electronics, vol. 59, no. 2, pp. 428–434, May 2013.

[2] P. J. Lobo, E. Juarez, F. Pescador, G. Maturana, and M. C. Rodrguez, "A DVB-H receiver and gateway implementation on a FPGA- and DSP based platform," IEEE Trans. on Consumer Electronics, vol. 57, no. 2, pp. 372–378, May 2011.

[3] L. Li and H. Zhou, "Structural transformation for best-possible obfuscation of sequential circuits," Proc. IEEE Int. Symp. Hardw. Oriented Security Trust (HOST).

[4] M. Hashemi, S. Mohammadi and T. E. Carlson, "TOP: A Combined Logical and Physical Obfuscation Method for Securing Networks-on-Chip Against Reverse Engineering Attacks," in *IEEE Access*, vol. 13, pp. 133438-133456, 2025.

[5] Y. Zhang *et al.*, "A Pay-Per-ISE RISC-V Processor With Hardware-Assisted Orthogonal Obfuscation," in *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 33, no. 4, pp. 1157-1161, April 2025.

[6] Y. Sao and S. S. Ali, "Security Analysis of Scan Obfuscation Techniques," in *IEEE Transactions on Information Forensics and Security*, vol. 18, pp. 2842-2855, 2023.

[7] B. Olney and R. Karam, "WATERMARCH: IP Protection Through Authenticated Obfuscation in FPGA Bitstreams," in *IEEE Embedded Systems Letters*, vol. 13, no. 3, pp. 81-84, Sept. 2021.

[8] M. Rostami, F. Koushanfar and R. Karri, "A Primer on Hardware Security: Models, Methods, and Metrics," in *Proceedings of the IEEE*, vol. 102, no. 8, pp. 1283-1295, Aug. 2014.

[9] R. Torrance and D. James, ‘‘The state-of-the-art in semiconductor reverse engineering,’’ in Proc. IEEE/ACM Design Autom. Conf., 2011, pp. 333–338.

[10] M. Rostami, F. Koushanfar, J. Rajendran, and R. Karri, ‘‘Hardware security: Threat models and metrics,’’ in Proc. Int. Conf. Comput.-Aided Design, 2013, pp. 819–823.

[11] Chipworks, ‘‘Reverse engineering software.’’ [Online]. Available: http://www.chipworks. com/en/technical-competitive-analysis/ resources/reverse-engineering-software.

[12] N. N. Anandakumar; M. Janke; J. Knechtel; I. Levi; X. Zhao; K. -S. Chong; S. Kose; B. -H. Gwee; J. Chang; L. Paul; Y. Wang, "Hardware Security Attack Landscape and Countermeasures," in *Hardware Security Attack Landscape and Countermeasures* , vol., no., pp.1-31, 19, Nov. 2024.

[13] P. Mishra and R. S. Chakraborty, "Tutorial T2B: Hardware Intellectual Property (IP) Security and Trust: Challenges and Solutions," *2018 31st International Conference on VLSI Design and 2018 17th International Conference on Embedded Systems (VLSID)*, Pune, India, 2018, pp. xxxix-xl, doi: 10.1109/VLSID.2018.21.

[14] A. Uma, P. Kalpana. (2021). Area efficient folded undecimator based ECG detector. Sci Rep 11,3756.

[15] Y. -J. Min, H. -K. Kim, Y. -R. Kang, G. -S. Kim, J. Park and S. -W. Kim, “Design of Wavelet-Based ECG Detector for Implantable Cardiac Pacemakers,” IEEE Trans. Biomed. Circuits Syst., vol. 7, no. 4, pp. 426-436, Aug. 2013.

[16] GitHub Repository, Available [Online]: https://github.com/vishbaba89/Detection-of-HLS-based-Structurally-Obfuscated-DSP-Designs-using-ML-Framework.

[17] DSP benchmarks for HLS, Indian Institute of Technology Indore, https://www.anirban-sengupta.com/HLS%20Benchmarks%20Details.pdf, accessed on November 2025.

[18] Express Benchmark Suite, *University of California Santa Barbara (UCSB)*, http://www.ece.ucsb.edu/EXPRESS/benchmark/, November 2025.

[19] Scikit-learn, Available [Online]: https://scikit-learn.org/stable/, Accessed on: November 2025.

[20] N. Pundir, F. Farahmandi and M. Tehranipoor, "Secure High-Level Synthesis: Challenges and Solutions," *2021 22nd International Symposium on Quality Electronic Design (ISQED)*, Santa Clara, CA, USA, pp. 164-171, 2021.

[21] Y. Lao and K. K. Parhi,“Obfuscating DSP Circuits via High-Level Transformations,” *IEEE Trans. Very Large Scale Integration Sys.*, vol. 23, no. 5, pp. 819–830, May 2015.

[22] A. Sengupta, D. Roy, S. P. Mohanty and P. Corcoran, "DSP design protection in CE through algorithmic transformation based structural obfuscation," IEEE Trans. Consum. Electron., vol. 63, no. 4, pp. 467-476, 2017.

[23] A. Vijayakumar, V. C. Patil, D. E. Holcomb, C. Paar, and S. Kundu,“Physical Design Obfuscation of Hardware: A Comprehensive Investigation of Device and Logic-Level Techniques,”*IEEE Trans. On Information Forensics and Security*, vol. 12, no. 1, pp. 64–77, Jan 2017.

[24] S. A. Islam, L. K. Sah, and S. Katkoori. 2020. High-Level Synthesis of Key-Obfuscated RTL IP with Design Lockout and Camouflaging. ACM Trans. Des. Autom. Electron. Syst. 26, 1, Article 6, 35 pages, 2021.

[25] A. Sengupta and M. Rathor, "Enhanced Security of DSP Circuits Using Multi-Key Based Structural Obfuscation and Physical-Level Watermarking for Consumer Electronics Systems," IEEE Trans. Consum. Electron., vol. 66, no. 2, pp. 163-172, 2020.

[26] A. Sengupta and R. Chaurasia, "Secure FFT IP Using C-Way Partitioning-Based Obfuscation and Fingerprint," in *IEEE Design & Test*, vol. 41, no. 5, pp. 55-64, Oct. 2024, doi: 10.1109/MDAT.2024.3395981.

[27] A. Sengupta, D. Roy, S. P. Mohanty and P. Corcoran, "Low-Cost Obfuscated JPEG CODEC IP Core for Secure CE Hardware," in *IEEE Transactions on Consumer Electronics*, vol. 64, no. 3, pp. 365-374, Aug. 2018, doi: 10.1109/TCE.2018.2852265.

[28] A. Sengupta and R. Chaurasia, "Securing IP Cores for DSP Applications Using Structural Obfuscation and Chromosomal DNA Impression," in *IEEE Access*, vol. 10, pp. 50903-50913, 2022.

[29] A. Anshul and A. Sengupta, "HLS Trojan Detection Using Machine Learning Technique," in *IEEE Embedded Systems Letters*, 2025, doi: 10.1109/LES.2025.3614187.

[30] C. Pilato, F. Regazzoni, R. Karri and S. Garg, "TAO: Techniques for Algorithm-Level Obfuscation during High-Level Synthesis," 2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC), San Francisco, CA, USA, pp. 1-6, 2018.

[31] P. Bagul, V. Inamdar, “Hardware Obfuscation Based Watermarking Technique for IPR Ownership Identification, International Journal of Reconfigurable Computing, 2023, 4550758, 13 pages, 2023.

[32] J. Chen, M. Zaman, Y. Makris, R. D. S. Blanton, S. Mitra and B. C. Schafer, "DECOY: DEflection-Driven HLS-Based Computation Partitioning for Obfuscating Intellectual PropertY," 2020 57th ACM/IEEE Design Automation Conference (DAC), San Francisco, CA, USA, 2020.

[33] S. K. Udupa, S. K. Debray and M. Madou, "Deobfuscation: reverse engineering obfuscated code," 12th Working Conference on Reverse Engineering (WCRE'05), Pittsburgh, PA, USA, 2005, pp. 10 pp.-54, doi: 10.1109/WCRE.2005.13.

[34] A. Dasgupta, S. Paria, C. Sozio and S. Bhunia, "LATTE: Library Attack for Evaluating Hardware IP Protections Against Reverse Engineering," IEEE Design & Test, vol. 42, no. 6, pp. 127-137, Dec. 2025.

[35] T. Gordon, E. Kilgore, N. Wylds and M. Nowatkowski, "Hardware Reverse Engineering Tools and Techniques," 2019 SoutheastCon, Huntsville, AL, USA, 2019, pp. 1-6.

[36] A. Dhavlle “Reverse Engineering of Integrated Circuits: Tools and Techniques” Cornell university. https://doi.org/10.48550/arXiv.2208.08689.

[37] J.Rajendran, A. Ali, O. Sinanoglu, R. Karri, “Belling the CAD: Toward Security-Centric Electronic System Design”, IEEE Trans. Comput-Aided Design Integr. Circuits Syst., Volume 34, Issue 11, 2015, pp 1756–1769.