Hardware Watermarking Using Polymorphic Inverter Designs Based On Reconfigurable Nanotechnologies

Shubham Rai[2], Ansh Rupani[2], Pallab Nath[1], Akash Kumar[2]
(1) Indian Institute of Technology, Indore, India
(2) Chair For Processor Design, CfAED, Technische Universität Dresden, Dresden, Germany

***Abstract*—We present here two watermarking techniques as a countermeasure to IC overbuilding and IP piracy by employing an encoding scheme using polymorphic inverter designs based on reconfigurable nanowire technologies. We employ a fabrication method unique to nanowire technologies which enables fixing of a node in the logic network to either 0 or 1. This technique allows fixing implicit *don't care* nodes to drive polymorphic inverters in a predetermined way, thereby contributing to the watermark. For a 64-bit signature, an area overhead of 0.72% and 2.14%, and an extremely low average probability of coincidence of 3.3x10^{-47} and 3.52x10^{-53} are obtained for our watermarking techniques for EPFL and IWLS benchmarks.**

***Index Terms*—reconfigurable RFETs, Schottky FET, tunable polarity**

I. Introduction

Globalization of IC fabrication has led to frequent copyright infringements and illegal ownership claims on IP. According to a report [32], the IC industry, on an average, loses about $4 billion annually due to IP infringement. Proving the ownership of an IP has gathered more attention today than ever before. To prove the ownership of an IP in a court of law, the original designer can show a secretly hidden signature or its watermark in the falsely claimed ICs. This signature should be justified in an unambiguous manner. Watermarking is an effective measure to prevent such false ownership claims.

There have been several works on watermark [42, 17, 40], but most of them deal with adding algorithmic constraints to embed the designer's signature at various stages of logic or physical synthesis. Further, actual overhead in terms of circuit parameters has not been discussed in detail. In the present work, we take a fresh perspective of looking at the problem of adding watermark to circuits using emerging reconfigurable nanotechnologies and present an overall flow for insertion and extraction of watermarks in the electronic circuits.

Recent developments in emerging technologies, especially runtime-reconfigurable nanowire technologies like silicon or germanium nanowire reconfigurable FETs [14, 19, 37] have opened up new avenues for hardware security. Transistors based on these technologies show electrical symmetry and can switch between p and n-type behavior. They have two or more independent gate terminals on a single channel: the control gate (CG) and the program gate (PG) where the difference in the potential bias at PG causes the realization of such behavior.

Using the above reconfigurability, we propose two watermarking techniques. The motivation for our watermarking techniques lies in the fact that the RFET-based inverters are intrinsically polymorphic i.e. they function as an inverter irrespective of the value of the program gate input. Our definition for "polymorphic" is different as compared to the previous works in the context of hardware security. While previous works correlate polymorphism with a single logic gate carrying out multiple logic functionality, our polymorphic inverters have different designs but they exhibit the same functionality.

We demonstrate our first watermarking technique by using an example of an encoding scheme inspired from *Huffman encoding*. This watermarking scheme is meant for attack models like IC counterfeiting and IC overbuilding. The type of polymorphic inverters used defines the unique watermark of the designer. For attacks based on reverse engineering, we propose a second watermarking technique where we encode designer's signature in the form of program gate inputs for our polymorphic inverter. Further, we employ a special mask development technique unique to nanowires to induce imperfections at implicit *don't care* nodes to drive the program gate inputs for the inverters [41]. Since such imperfections do not interfere with the actual functionality of the design, the watermark is extremely stealthy as the layout of the gate remains the same and this imperfection is hidden well within the technology.

Contributions: Following are the major contributions of this work:

- We introduce four polymorphic inverter designs based on reconfigurable transistors ranging from a static design to a totally runtime reconfigurable design and use them in our watermarking techniques.
- For the second watermarking technique, we mark the implicit don't care [15] nodes scattered in the logic network to introduce an imperfection at those nodes so that it is either short to V_{ss} or V_{dd} depending upon designer's watermark. These nodes then feed the program gate terminal of runtime-reconfigurable inverters for the inverting operation.
- An evaluation in terms of area overhead and probability of coincidence is carried for EPFL [3] and IWLS [2] benchmark suites. For a 64-bit signature, area overhead of 0.72% and 2.14% and probability of coincidence 3.3x10^{-47} and 3.52x10^{-53} are obtained for watermarking technique 1 and 2 respectively. This is the lowest as compared to the existing works. The flow is available as an open source tool at *www.updateafterreview.org*.

Further, we discuss various attacks like removal, masking and forging on our proposed watermarks as mentioned in [7] and assess the strength of our watermarking techniques in terms of robustness, unobtrusiveness, universality and unambiguous nature of the signature as suggested in [31].

II. Background and Motivation

A. Reconfigurable Transistors

In the present work, we focus on silicon nanowire based reconfigurable technology[1]. As mentioned earlier, nanowire RFETs have two or more gate terminals on a single channel as shown in Fig. 1a where the **control gate** (CG) receives the normal input to the transistor and controls the creation of a carrier channel. The **program gate** (PG) determines the carrier type in the channel. The uniqueness lies in the fully-symmetrical I-V characteristics in either behavior. This is shown in Fig. 1a. The blue and red lines show the I-V characteristics for p and n-type behavior respectively.

Feasibility of RFET based electronic circuits: While there are several apprehensions on the feasibility aspects for emerging nanotechnologies, there are several reasons which motivate us to look at security features for circuits based on reconfigurable nanotechnology.

- Reconfigurability at the transistor level is a manifestation of ambipolarity observed in various materials like silicon and germanium below 45nm which are commonly used in CMOS manufacturing process [11]. Hence, reconfigurable devices made of such materials can readily be adopted by industries [39]. Works like [20, 35] have shown that they follow the similar fabrication and manufacturing process as CMOS.
- There is no dearth of reconfigurable nanotechnologies based on materials like carbon [18], graphene [13], germanium [37] and WSE_2 [30] showing that reconfigurability at the technology level has many contenders.
- Various works like [11, 38, 25, 26] have demonstrated advanced circuit-level designs and implementations using RFETs followed with detailed evaluation.
- Polymporphism at the transistor level is an important criteria for hardware security as it can inherently support both camouflaging and locking [21]. Security applications using such reconfigurable nanotechnologies have already been demonstrated in works like [28, 5, 8] owing to their unique I-V characteristics which are generally not possible with conventional MOSFETs.

B. Related Works on Hardware Watermarking

One of the earliest work in watermarking was proposed in [42]. The authors introduced a constrained graph partitioning at various stages of CAD synthesis. The constraints are encoded with designer's watermark to be embedded within the hardware system. On similar lines, the authors in [17] used a set of design and timing constraints which encoded the designer's signature during behavioral synthesis. They claimed low hardware overheads while demonstrating the watermarks to be robust, universal and unambiguous. However, for hardware overheads they used number of parity bits for error-correcting as a reference. Similarly, authors in [34] proposed a watermarking scheme for high level synthesis (HLS). They proposed embedding the designer's signature in three phases of high level synthesis: scheduling, hardware allocation and register allocation. Additionally, they claim that since the three phases were independent to each other, it is extremely difficult to identify which HDL phase and constraints are used for watermark insertion.

[1]For this work, we have used silicon nanowire based reconfigurable field effect transistors (SiNW RFETs) because it is one of the most actively researched emerging technologies and has been evaluated with a physical synthesis flow [24]. However, the concepts described here are applicable for reconfigurable nanotechnologies in general.

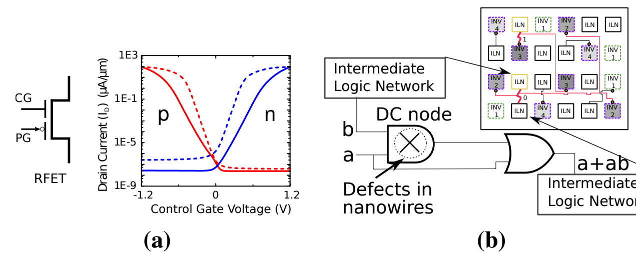

Fig. 1: (a) Symmetrical IV characteristics for SiNW RFET as shown in [26] (b) *Don't Care* node to be used for our polymorphic inverters. ILN stands for intermediate logic node

While the above works used constraint optimization, there are certain works in which the authors used functional polymorphism to realize dynamic watermarks [23, 7]. In dynamic watermarks, the circuit delivers correct functionality during normal operation but in order to detect the watermark, the circuit is subjected to special conditions like different inputs or external controls to reveal the embedded watermarks. In [40], the authors used evolutionary algorithms to generate polymorphic gates for watermarking which behave differently based on external conditions like temperature or pressure. They proposed replacing standard logic gates with their polymorphic gates which hide secret information from the designer. The secret information can be activated using a special mode with specific control factors. One common aspect in all of these works is that they propose watermarking techniques by encoding designer's signature in a list of constraints which then govern specific stages in CAD synthesis. This may lead to sub-optimal circuit parameters in terms of area, power and delay. Moreover, in works related to watermarks based on polymorphic gates, most of them require special inputs and conditions to reveal the designer's signature. Since technology level modifications are difficult to detect [4], we propose techniques to introduce static watermarks in the circuit leveraging unique features of emerging technologies.

C. Implicit Don't Cares

An essential component in our proposed scheme is using *Implicit Don't Care (DC)* nodes. These refer to the input or output patterns which are made impossible because of logic restrictions caused by the structure of the logic network. For example, as shown in Fig. 1b, the function $f = a + ab$ can be easily reduced to $f = a$ with b as a don't care signal. For our watermarking, we do *not* optimize out some of the *don't care* nodes intentionally which are present in the network. Then we introduce the mask defects so as to fix these nodes at V_{ss} or V_{dd}. Since these nodes are *don't care*, the logic circuit before and after this defect remains logically equivalent. These nodes further drive the program gate of our polymorphic inverters for the watermark.

III. Threat Model

In the present work, we propose our watermarking scheme to thwart IP piracy, IC counterfeiting and overbuilding attacks. The attacker does not intend to degrade the design quality. We assume that the adversary can be at the SoC integration, or at the foundry or can be an end user, and he/she has access to the complete specification of the design (this includes logic netlist and even physical synthesis design GDSII). He/She has access to reverse engineering techniques where he/she can identify the components of the circuit as well. He/She can also run redundancy removal methods to find any use of redundant

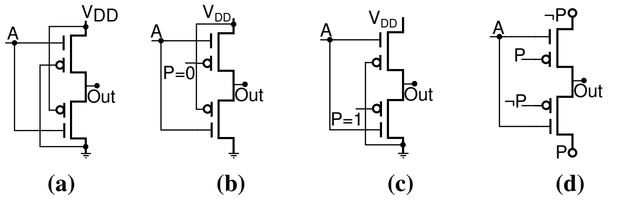

Fig. 2: Different design for inverters (a) Static design. The drain, source and program gate are hard fixed to V$_{dd}$ and V$_{ss}$ as proposed in [25] (b) One of the transistor's program terminal is connected to 0 (c) One of the transistor's program terminal is connected to 1 (d) Fully reconfigurable design

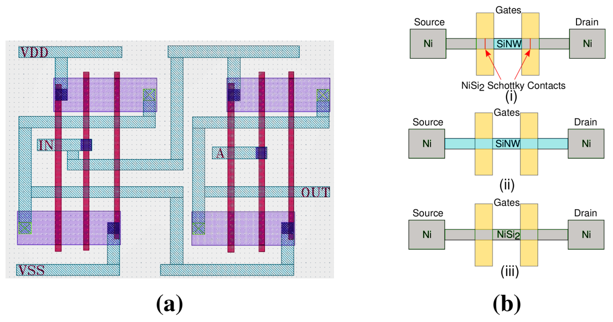

Fig. 3: (a) Layout for fully polymorphic inverter design as shown in Fig. 2d (b) Defect introduction into an RFET. (i) Conceptual layout of a working SiNW RFET (ii) Omitted formation of Schottky junctions, disabling the transistor (iii) Fully silicided channel turning the transistor into a metallic wire

nodes. He/She can also make use of SAT based attacks [36] to either remove or forge the watermarks. He/She also has access to side-channel attacks where he/she can measure current or power profiles of the circuits. SEM PVC (Scanning Electron Microscope in the passive voltage contrast) [9] techniques can potentially figure out the watermark but these techniques are extremely costly and time consuming. However, there are other measures proposed in literature which can prevent SEM attacks [22], which can be applied in combination with our technique. In our threat model, we assume that the attacker does not have this capability.

IV. Polymorphic Inverter Design Using Reconfigurable FETs

In this work, we use four designs of inverters for our watermarking techniques. These designs are presented in Fig. 2. It shows a SiNW RFET based inverter, as inspired from designs in [24] and [38]. The design shown in Fig. 2a is a static design. The P and $\overline{P}$ terminals are pre-connected to V_{dd} and V_{ss} respectively. Fig. 2b shows an inverter which is connected in a way that this cell only functions as an inverter when the value of P is 0. This is so because the bottom transistor is already configured to be an n-type FET. Therefore, for this logic cell to function as an inverter, the upper transistor has to be configured as a p-type FET which requires P to be 0. Similarly, the inverter design shown in Fig. 2c functions as an inverter only if the program gate input to this design is 1.

Fig. 2d shows the completely invariant SiNW RFET based inverter. Its layout is shown in Fig. 3a. The layout shows two inverters where the left inverter is driving the right. The input to the left inverter comes from the *don't care* (explained in the coming sections). This design functions as an inverter irrespective of the program input value. This happens because

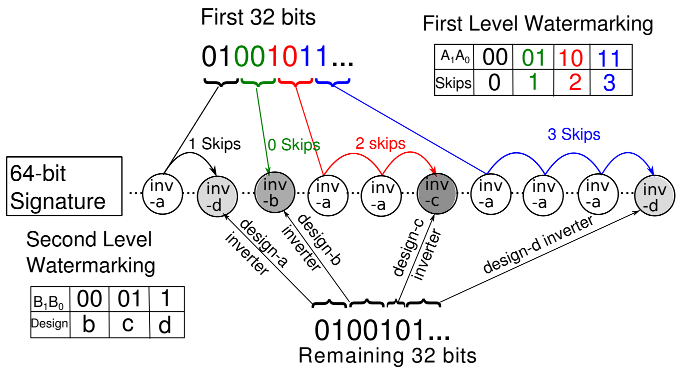

Fig. 4: Encoding Scheme For First Watermarking Technique

depending on the value of the P input, the pull-up and pull-down networks change in compliance with the power and ground terminals. From now on, we refer to the inverter designs shown in Fig. 2a, 2b, 2c and 2d as designs a, b, c and d respectively.

V. Proposed Watermarking Scheme

In the present section, we first discuss the steps to introduce the defects in nanowires and how it can be employed for our watermarking techniques.

A. Introduction of Defects

Fixing the value of the program gate input 'P' for the polymorphic inverters is an essential step in our watermarking techniques. We feed the input P of the inverters by fixing the *don't care* nodes to a particular value. Since these special nodes are *don't care* nodes and are basically redundant, they don't affect the logic functionality.

The unique function of an RFET is enabled by sharp metal-to semiconductor heterojunctions. These junctions are controlled by the two individual gates as shown in Fig. 3b(i) [43]. Typically nickel silicide is used as a contact material. A straight forward and convenient way to build a defect into this type of device is on-purpose misalignment of those heterojunctions by the layout of the device. If no silicide is formed below the gates, the transistor will be always "off" (Fig. 3b(ii)). The other way round, a contact silicidation through the whole channel will turn the transistor into a metallic wire, which always conducts a current (Fig. 3b(iii)). This full silicidation can be enabled by a number of process changes, like a higher amount of deposited Ni or a smaller Source-Drain distance in the device layout as proposed in [41, 37]. Alternatively the ungated area in the centre of the channel can be opened on purpose during fabrication also leading to a fully silicided channel.

B. First Watermarking technique

In this scheme, we use various designs of polymorphic inverters to encode the designer's watermark. We use an encoding scheme inspired from *Huffman coding*[2]. We divide the n-bit watermarking signature into two parts: the first $n/2$ bits determine the number of inverters to be skipped between the two earmarked inverters which would be modified and the second $n/2$ bits determine the type of inverter to be used at that specific position. The *skip* here means that for the skipped

[2]While we use *Huffman coding* due to its simplicity and ease of understanding, the use of polymorphic inverters can be applied to other coding schemes as well.

inverter locations, we use the design a inverters (Fig. 2a) to have a lower area overhead.

Our encoding scheme is easily understood with the help of an example. Suppose the signature to be embedded is "KLMNJINC" if the designer is Kilimanjaro Incorporation. The 64-bit binary equivalent of the ASCII values here is "01001011 01001100 01001101 01001110 01001010 01001001 01001110 01000011". As shown in Fig. 4. the first 32 bits are divided into sets of two bits where the numerical value of each set gives the number of skips. We use inverter design b (Fig. 2b), c (Fig. 2c) and d (Fig. 2d) and encode them with bit values '00', '1' and '01' respectively for this technique[3]. The last 32 bits are similarly divided into sets of either two bits (when the immediate bit start from '0') or 1 bit (when the first accessed bit is '1'). Based on the value of each set, the respective type of inverter is used. As shown in Fig. 4, the first set of the first 32 bits is 01 which means there is one skip. So the first inverter is "skipped" which implies that it is of design a. The subsequent inverter to be earmarked is determined by the first set from the second half of the signature, which in this case is '01', thereby implying the usage of a design d inverter (Fig. 2d). Similarly, as we continue, the following set in the first 32 bits is '00', which implies that there are no skips and the next immediate inverter's design is decided by the second set of the last 32 bits. As the value of the second set is '00', an inverter based on design b is utilized as shown in Fig. 4. One might notice that the first-level watermark provides the skipping information for a maximum of 16 earmarked inverters (where all the last 32 bits are 1), in case of 64-bit (or even more) signature. In cases where we land up employing more than 16 inverters in the second level of watermarking, our tool starts re-using the skipping pattern as decided at the first-level of watermarking from the beginning.

This simple watermarking scheme is meant for designers who want to protect their IC from adversaries who are at the foundries and who intend to use counterfeiting and overbuilding as attack measures. This watermarking scheme is ineffective in cases where the attacker uses RE based attacks as he/she can easily detect the difference in layouts of the different types of inverters.

C. Second Watermarking Technique

For making the watermark RE resistant, we modify the above technique by only using the 4th polymorphic inverter design (Fig. 2d). The 4th inverter design functions as an inverter irrespective of the value of the program input. While executing the implementation of the first-level watermark, instead of using design 1 inverters to implement skips, we use design 4 inverters which get activated with any value of the program gate input, i.e. P=DC. This means that P can be fed from any intermediate logic node. For the last $n/2$ bits, the value of each individual bit defines the program gate input of the 4th inverter. When the value of the signature bit is 0, the P value of the earmarked inverter is fixed to 0 and vice-versa. This technique is different from the technique proposed above because the actual watermark at the second level in this case lies in the program gate input value rather than the type of inverter being used. For the attacker, since the inverters used in the watermark are the same (which means they have the same layout), such watermarking technique further secures the circuit from reverse engineering attacks [29]. It is the

[3]Type b and c inverters have least area overhead and we want to have the maximum of these inverters in our watermark signature to minimize the total area. The single bit assignment can be done to either of these designs.

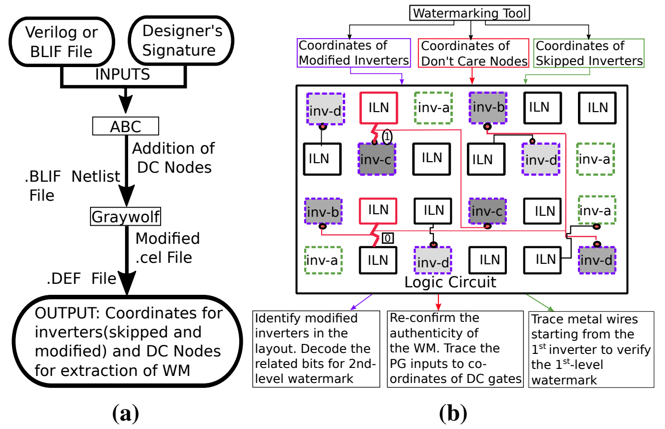
Verilog or BLIF File
Designer's Signature
INPUTS
ABC
Addition of DC Nodes
.BLIF Netlist File
Graywolf
Modified .cel File
.DEF File
OUTPUT: Coordinates for inverters(skipped and modified) and DC Nodes for extraction of WM
(a)
Watermarking Tool
Coordinates of Modified Inverters
Coordinates of Don't Care Nodes
Coordinates of Skipped Inverters
inv-d
ILN
inv-a
inv-b
inv-c
inv-d
1
0
Logic Circuit
Identify modified inverters in the layout. Decode the related bits for 2nd-level watermark
Re-confirm the authenticity of the WM. Trace the PG inputs to co-ordinates of DC gates
Trace metal wires starting from the 1st inverter to verify the 1st-level watermark
(b)

Fig. 5: (a) Complete Flow to Insert the Watermark (b) Proposed Watermark Extraction Mechanism from a logic circuit

designer who knows exactly the earmarked inverters and the exact values of their program gate inputs at runtime.

However, using special RE techniques, the attacker can identify various components of the circuit. He/She can further figure out that the inputs to these polymorphic inverters are coming from *don't cares*, and hence he/she can remove these *don't care* nodes as well to create a fresh mask. Therefore in order to have an additional level of protection and use the benefits of these defects, we intentionally add nanowire defects at other locations in the logic circuit. The combination of these defects along with polymorphic nature of RFETs based logic gates can be used in user defined ways to implement watermarks. For example, XOR logic gate as demonstrated in [27] is a simple example which can be employed to replace either intermediate wire or inverting logic. In this scenario, such modification provides an additional security as the new mask may not carry our intentionally introduced defects at the *don't care* nodes connected to some other functionally relevant logic gate. This ensures that the new circuit (from the new mask) will not function correctly. One can argue that the attacker can use redundancy removal techniques to remove all *don't care* nodes but redundancy removal techniques do not guarantee removal of all *don't care* nodes [15]. Similarly, recently demonstrated SAT based attacks [36] can also be used to find these *don't care* nodes. We reason that since RFET based circuits can be potentially made 100% camouflaged with low hardware overheads [25, 38], SAT based attacks will consume large amount of time to detect all such points in the circuit as demonstrated in [21]. Hence, even if the attacker figures out all the *don't cares*, multiple iterations are required which adversely affect his/her *time-to-market* for the final product.

D. Tool Flow

The tool flow to implement the above watermarking technique is shown in Fig. 5a. The two inputs are the Verilog (or BLIF) circuit description file along with the n-bit designer's signature. We have used an open source tool, *QFLOW*, for the implementation [10]. In the first stage, we use the ABC [6] tool to get the *don't care* nodes and the mapped netlist. The mapped netlist with the information of additional DC nodes is converted into a .cel file which is used by graywolf [12] for the floor planning and placement stages. This .cel file is modified according to the designer's signature. The skipping information and the inclusion of modified inverter designs is carried by making the required changes in this file. These changes are forwarded to graywolf and all the standard cells are placed. Special care has to be taken to avoid using DC gates which lie in the critical path to prevent side channel attacks [16]. The coordinates of the modified inverters, skipped inverters and DC nodes are extracted and provided to the designer as the final output which are useful during extraction of watermarks.

E. Extraction of Designer's Signature in IC under test

According to the authors in [7], extraction of watermark should be easy and practical. For the extraction of the embedded signature in an IC under test, the tool provides the designer with the coordinates of modified and skipped inverters as well as the don't care nodes in the circuit as shown in Fig. 5b. For the first watermarking technique, after the identification of the modified inverters in the correct sequence, the designer can decode the second half of the signature. By identifying the DC nodes in the layout and tracing the program gate inputs of the modified inverters up to these DC nodes, the second half of the signature can be confirmed. Also, by tracing the connections of the skipped inverters in the correct sequence starting from the first inverter, the first half of the signature is decoded, as shown in Fig. 5b.

For the second watermarking technique, by identifying the *don't care* nodes in the layout and tracing the program gate inputs of the inverters up to these DC nodes, the designer can confirm the second half of the signature (by noting the value of the P inputs which is supposedly constant due to the defects added). After the detection of the second level watermark in this case, the first level watermark (related to inverter skips) can be detected by observing the P inputs of the skipped inverters which have to be traced and proven to be arriving from random intermediate nodes (and not the nodes with defects), unlike the modified inverters.

As the introduction of defects have caused these *don't care* nodes to be either at V_{ss} or V_{dd}, these points in the circuit will have different side-channel fingerprint [16] as compared to the case when there are no such defects. Since these nodes are known to the IP designer, he/she can use the coordinates and the side-channel measures to prove presence of defects at *don't care* nodes [7] and hence strengthening his/her claim of ownership.

VI. Results and Discussion

A. Evaluation on Benchmarks

We evaluate our watermark techniques on EPFL [3] and IWLS [2] benchmark suites and calculate the probability of coincidence, P_c [17] and area overhead. The complete analysis is shown in TABLE I where IWLS benchmarks are shown with a gray background to differentiate from the EPFL benchmarks. The technique proposed in this work is targeted at large ICs, with an adequate number of inverters already present in the design.

Probability of Coincidence: It is the probability that the same watermark can be inserted by any other designer. As stated in [17], P_c does not follow its exact meaning from mathematics in a rigorous sense, rather it is used as an "approximation" for the actual probability. For the first watermarking technique, in case of a 64-bit signature, the maximum and minimum number of inverters which can be changed with our encoding scheme are 32 and 16 respectively. While the calculations shown here are for 64 and 128 bits, the same procedure can be followed for a larger signature. We take 24 as the average number of inverters which are affected. For a total of n number of inverters in a design, P_c is calculated as: $P_c = \frac{1}{\binom{n}{24} * 3^{24}}$. Similarly for second watermarking technique, the P_c is calculated as: $P_c = \frac{1}{\binom{n}{32} * 2^{32}}$. From TABLE II, we can see that the probability decreases if more bits are used. As shown in TABLE II, the values of probability of coincidence for 64 and 128 bit signatures for our techniques are $3.3 \text{x} 10^{-47}$ and $3.72 \text{x} 10^{-53}$ which are the least as compared to other works.

TABLE I: Area overhead for 64 and 128 bits watermark signature.

Benchmarks	DC Gates	Inv.	%Area Ov. 64 bits WM1	%Area Ov. 128 bits WM1	%Area Ov. 64 bits WM2	%Area Ov. 128 bits WM2
arbiter	217	512	0.46	0.93	1.38	2.75
div	26919	16063	0.08	0.17	0.25	0.50
hyp	3542	46601	0.02	0.04	0.07	0.13
log2.	1013	4375	0.16	0.33	0.49	0.98
mem_ctrl	3851	8176	0.11	0.22	0.32	0.65
multiplier	1407	2906	0.21	0.42	0.62	1.24
sin	323	992	0.92	1.85	2.74	5.47
sqrt	13094	5295	0.17	0.34	0.50	1.00
square	2060	3915	0.27	0.54	0.80	1.60
voter	5814	3974	0.35	0.69	1.02	2.05
des_perf	9175	15291	0.07	0.13	0.20	0.40
ethernet	29003	16509	0.07	0.15	0.22	0.43
pci_ 27413	5284	5045	0.22	0.44	0.66	1.32
usb_funct	2353	3181	0.33	0.66	0.97	1.94
ac97_ctrl	4105	2666	0.36	0.71	1.05	2.11
mem_ctrl	7049	2384	0.34	0.68	1.01	2.03
systemcaes	3720	2435	0.42	0.83	1.23	2.47
systemcdes	436	653	1.74	3.48	5.16	10.32
pci_spoci_ctrl	612	285	3.72	7.44	11.03	22.07
i2c	277	238	4.41	8.81	13.06	26.12
Average			**0.72**	**1.44**	**2.14**	**4.28**

TABLE II: Comparison with Previous Works

Methodology	64-bit Signature	128-bit Signature	Avg. Pc 64 bit	Avg. Pc 128 bit
Koushanfar et al. [17]	10.12	20.24	$2.34 \text{x} 10^{-3}$	$5.48 \text{x} 10^{-6}$
Sengupta et al. [33]	9.29	18.59	$2.34 \text{x} 10^{-3}$	$5.48 \text{x} 10^{-6}$
Wang et al. [40]	4.21	8.42	Not Provided	Not Provided
Sengupta et al. [34]	3.37	6.75	$9.56 \text{x} 10^{-18}$	$9.13 \text{x} 10^{-35}$
Present Work WM1	**0.72**	**1.44**	$\mathbf{3.33 \text{x} 10^{-47}}$	$\mathbf{7.62 \text{x} 10^{-80}}$
Present Work WM2	**2.14**	**4.28**	$\mathbf{3.72 \text{x} 10^{-53}}$	$\mathbf{1.11 \text{x} 10^{-87}}$
Fully Camouflaged WM1/2	11.01	12.22	$1/2^n$	$1/2^n$

'n' refers to the total number of inverters present in the design

Overhead and Comparison: For area overhead calculations, we use the standard cell library provided in [25]. We use the basic logic gates available: *XOR, NAND,NOR, INV and BUF* in the library and measure the area in μm^2. Area overhead for the DC Nodes is included with the assumption that one DC node drives one inverter. For DC area overhead calculation, we consider the weighted average of the area of all the logic gates in a particular benchmark. The area overhead for 64 and 128-bit signatures for both the watermarking techniques is shown in TABLE I. We calculate the upper bound in terms of area overhead for both the watermarking techniques. For the first watermark, we assume that for a 64-bit signature, maximum area overhead would be incurred when design d inverters are employed throughout the second level of watermarking. In those cases, a maximum 16 such inverters would be employed.

For the case of second watermarking technique, more number of inverters are used and that reflects in TABLE I. This is because in the first part, for each '2 bits' in the first 32-bits, the maximum skip is 3-'design d' inverters and for each single bit in the second part, we are again using the design d inverter. We compare our results with polymorphic gates based watermarking shown in [40], the behavioral synthesis watermarking employed in [17], and the high-level synthesis based watermarking schemes in [33] and [34]. Results for

[17], [33] and [34] have been obtained from the analysis presented in [34]. The results have been interpolated (or extrapolated in some cases) to a 64-bit or 128-bit signature and shown in TABLE II. Our work (shown in light background in TABLE II) has the least area overhead and probability of coincidence as compared to previous works for 64 and 128 bit signatures.

A special scenario arises when all the inverters in a circuit are design d inverters. This will lead to a fully camouflaged circuit [25, 38] with our proposed watermarking technique. Obviously, this has a higher area overhead, as shown in the last row of TABLE II. In this case, the area overhead differs in the two columns due to the number of *don't care* nodes employed. The probability of coincidence is also very high in this case.

B. Attack Analysis

The authors in [7] list three main attacks on watermarks: removal, masking and forging. Removal is a special type of masking where the attacker is successful in completely removing the original watermark. We analyze these attacks for our watermarking technique.

Removal and Masking of Watermarks: The first watermark is easy to remove or mask as it caters to simple security threats of IC counterfeiting and IC overbuilding where the adversary does not intend to create a new mask. For the RE resistant watermark, the attacker may use image-processing based techniques to identify the inverters and remove the connected don't care nodes and create a new mask thereby either removing or masking the actual watermarking. As mentioned earlier, the new mask will not contain the additional security layer caused due to defects which are driving functionally relevant nodes, rendering the circuits from the new mask to be non-functional. Further, if the attacker is successful in either removing or masking the designer's watermark, and adding his/her own watermark, the time at which the watermark was inserted becomes the deciding factor in proving rightful ownership [7, 1].

Forging of Watermarks: In case of forging attacks, if the adversary adds his own watermark in the original IP then the owners can prove their rightful ownership by providing an IP with their own watermark while the copied IP has both the watermarks.

C. Evaluating our watermarking techniques

According to the requirements of a good watermarking technique, as stated in [31], a watermarking technique should be robust, unobtrusive, unambiguous as well as universal. So it is imperative to evaluate these properties for our technique.

Robustness: For any watermarking technique to be robust, it is necessary that it is extremely difficult to remove the watermark. In case of attacks due to overbuilding, piracy and counterfeit ICs, our watermark signature will be carried further to the rogue ICs as well. Only SEM based attacks can truly figure out the imperfections in the nanowires but as mentioned before they are very costly and time consuming.

Attackers can also make use of side-channel attacks to probe and analyze the circuit. Due to the intended defects, the nanowire channels are connected to V_{dd} or V_{ss}, thus, they will have a different current trace. But to distinguish whether they are a part of the circuit or a part of watermark is difficult for the attackers considering there is no such *gold* version of the unwatermarked circuit to compare the power or current footprints with [16].

Unobtrusiveness: Thanks to the invariant functionality of the inverters in RFET based technology, our watermarks are very unobtrusive. The functionality of the inverter and henceforth the functionality of the circuit does not get affected by the program gate input if the inverter is the one shown in Fig. 2d. The functionality of other inverter designs is already predecided. We have taken care of identifying specific don't care/redundant nodes which would act as the points to drive a constant 1 or 0. These nodes, being don't care nodes, would not affect the functionality of the circuits, while driving our inverters secretly.

Unambiguity: Our watermarking scheme is unambiguous because it provides a decisive proof of ownership by deploying a watermark related to the design house, as shown in Section V. Since the defects are hidden at the technology level [4], it is generally very difficult to remove them in fake ICs thereby making our watermark unambiguous.

Universal Nature: Likewise, our technique is universal as long as all IC manufacturers keep on using inverters in their designs, the most common logic cell in logic networks.

VII. Conclusions and Future Work

In the present work, we leverage the unique properties of reconfigurable emerging nanotechnologies for hardware security. We propose watermarking techniques which use an encoding scheme to embed the designer's signature in his IC. The techniques exploit the natural presence of inverters in an RFET based ASIC design as well as its fabrication technique to introduce below-gate-level defects to embed our watermark signature. Critical path analysis plays an important role during optimization of don't care nodes. However, since it is orthogonal to the current work, it will be investigated in our future work. Our proposed techniques show the least probability of coincidence of $3.3\text{x}10^{-47}$ and $3.52\text{x}10^{-53}$ for 64 and 128 bit watermark signatures. Additionally, minimal area overhead of 0.72% and 2.14% respectively has been incurred as compared to previous works.

Acknowledgment

The authors would like to thank Dr. Jens Trommer (Namlab gGmbH, Dresden, Germany) for his valuable inputs. This work is supported in part by the German Research Foundation (DFG) within the Cluster of Excellence "Center for Advancing Electronics Dresden" (CfAED) at the Technische Universität Dresden.

References

[1] Amr T. Abdel-Hamid, Sofiéne Tahar, and El Mostapha Aboulhamid. "A Survey on IP Watermarking Techniques". In: *DAES* (2004).
[2] C. Albrecht. *IWLS 2005 Benchmarks*. Tech. rep. June 2005.
[3] L. Amarù, P.E. Gaillardon, and G. De Micheli. "The EPFL Combinational Benchmark Suite". In: *IWLS* (2015).
[4] G. T. Becker et al. "Stealthy dopant-level hardware trojans". In: *IWCHAES*. 2013.
[5] Yu Bi et al. "Emerging Technology-Based Design of Primitives for Hardware Security". In: *J. Emerg. Technol. Comput. Syst.* (2016).
[6] R. Brayton and A. Mishchenko. "ABC: An academic industrial-strength verification tool". In: *Computer Aided Verification*. Springer. 2010.
[7] C. H. Chang, M. Potkonjak, and L. Zhang. "Hardware IP Watermarking and Fingerprinting". In: *Secure System Design and Trustable Computing*. Springer International Publishing, 2016.
[8] A. Chen et al. "Using emerging technologies for hardware security beyond PUFs". In: *DATE*. 2016.
[9] F. Courbon, S. Skorobogatov, and C. Woods. "Direct charge measurement in floating gate transistors of flash EEPROM using scanning electron microscopy". In: *Proc. ISTFA*. 2016, pp. 1–6.
[10] T. Edwards. *A Digital Synthesis Flow using Open Source EDA Tools*. 2017. URL: http://opencircuitdesign.com/qflow/index.html.
[11] P. E. Gaillardon et al. "Vertically-stacked Double-gate Nanowire FETs with Controllable Polarity: From Devices to Regular ASICs". In: *DATE*.
[12] Graywolf. *Graywolf - Professional-grade placement tool*. 2017. URL: https://github.com/rubund/graywolf.

[13] Naoki H. et al. “A polarity-controllable graphene inverter”. In: *Applied Physics Letters* (2010).
[14] A. Heinzig et al. “Reconfigurable Silicon Nanowire Transistors”. In: *Nano Letters* (2012).
[15] W. Hu et al. “Why you should care about don’t cares: Exploiting internal don’t care conditions for hardware Trojans”. In: *ICCAD*. 2017.
[16] F. Koeune and F. X. Standaert. “A tutorial on physical security and side-channel attacks”. In: *Foundations of Security Analysis and Design III*. Springer, 2005.
[17] F. Koushanfar, I. Hong, and M. Potkonjak. “Behavioral Synthesis Techniques for Intellectual Property Protection”. In: *TDAES* (2005).
[18] Y.M. Lin et al. “High-performance Carbon Nanotube Field-effect Transistor with Tunable Polarities”. In: *IEEE Trans. Nanotechnol.* (2005).
[19] M. De Marchi et al. “Polarity control in double-gate, gate-all-around vertically stacked silicon nanowire FETs”. In: *IEDM*. 2012.
[20] T Mikolajick et al. “The RFET-a reconfigurable nanowire transistor and its application to novel electronic circuits and systems”. In: *SST* (2017).
[21] S. Patnaik et al. “Advancing hardware security using polymorphic and stochastic spin-hall effect devices”. In: *DATE*. 2018.
[22] S. Patnaik et al. “Obfuscating the interconnects: Low-cost and resilient full-chip layout camouflaging”. In: *ICCAD*. 2017.
[23] Gang Qu and Miodrag Potkonjak. *Intellectual property protection in VLSI designs: theory and practice*. Springer Science & Business Media, 2007.
[24] S. Rai et al. “A physical synthesis flow for early technology evaluation of silicon nanowire based reconfigurable FETs”. In: *DATE*. 2018.
[25] S. Rai, M. Raitza, and A. Kumar. “Technology mapping flow for emerging reconfigurable silicon nanowire transistors”. In: *DATE*. 2018.
[26] M. Raitza et al. “Exploiting transistor-level reconfiguration to optimize combinational circuits”. In: *DATE*. 2017.
[27] J. Rajendran et al. “Fault Analysis-Based Logic Encryption”. In: *IEEE TC* (2015).
[28] J. Rajendran et al. “Nano Meets Security: Exploring Nanoelectronic Devices for Security Applications”. In: *Proceedings of the IEEE* (2015).
[29] J. Rajendran et al. “Security Analysis of Integrated Circuit Camouflaging”. In: *ACM SIGSAC Conference on Computer*. 2013.
[30] G. V. Resta et al. “Polarity control in WSe2 double-gate transistors”. In: *Scientific Reports* (2016).
[31] M. Rostami, F. Koushanfar, and R. Karri. “A Primer on Hardware Security: Models, Methods, and Metrics”. In: *Proceedings of the IEEE* (2014).
[32] SEMI. *Innovation is at risk as semiconductor equipment and materials industry loses up to $4 billion annually due to IP infringement*. 2012.
[33] A. Sengupta, S. Bhadauria, and S. P. Mohanty. “Embedding low cost optimal watermark during high level synthesis for reusable IP core protection”. In: *ISCAS*. 2016.
[34] A. Sengupta, D. Roy, and S. P. Mohanty. “Triple-Phase Watermarking for Reusable IP Core Protection During Architecture Synthesis”. In: *TCAD* (2018).
[35] M. Simon et al. “Top-Down Technology for Reconfigurable Nanowire FETs With Symmetric On-Currents”. In: *IEEE Trans. Nanotech.* (2017).
[36] P. Subramanyan, S. Ray, and S. Malik. “Evaluating the security of logic encryption algorithms”. In: *HOST*. 2015.
[37] J. Trommer. *Towards Reconfigurable Electronics by Functionality-Enhanced Circuits and Germanium Nanowire Devices*. BoD–Books on Demand, 2017.
[38] J. Trommer et al. “Reconfigurable nanowire transistors with multiple independent gates for efficient and programmable combinational circuits”. In: *DATE*. 2016.
[39] Jing Wan et al. *Integrated circuits with nanowires and methods of manufacturing the same*. US Patent 9,306,019. 2016.
[40] T. Wang et al. “Polymorphic gate based IC watermarking techniques”. In: *ASP-DAC*. 2018.
[41] W. M. Weber. “Silicon to Nickel Silicide Longitudinal Nanowire Heterostructures: Synthesis, Electrical Characterization and Novel Devices”. PhD thesis. Technische Universität München, 2008.
[42] G. Wolfe, J. L. Wong, and M. Potkonjak. “Watermarking graph partitioning solutions”. In: *DAC*. 2001.
[43] E. Zschech, C. Whelan, and T. Mikolajick. *Materials for information technology: devices, interconnects and packaging*. Springer Science & Business Media, 2006.