July 2015

Development of a Layout-Level Hardware Obfuscation Tool to Counter Reverse Engineering

Shweta Malik
University of Massachusetts Amherst

DEVELOPMENT OF A LAYOUT-LEVEL HARDWARE OBFUSCATION TOOL TO COUNTER REVERSE ENGINEERING

A Thesis Presented

by

SHWETA MALIK

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL AND COMPUTER ENGINEERING

May 2015

Department of Electrical and Computer Engineering

DEVELOPMENT OF A LAYOUT-LEVEL HARDWARE OBFUSCATION TOOL TO COUNTER REVERSE ENGINEERING

A Thesis Presented

by

SHWETA MALIK

Approved as to style and content by:

Wayne P. Burleson, Chair

Christof Paar, Member

Sandip Kundu, Member

C.V Hollot, Department Head

Department of Electrical and Computer Engineering

Dedicated to my family and mentor Daisaku Ikeda

ACKNOWLEDGEMENT

I would like to thank my advisor prof. Wayne Burleson for his continuous advising and offering critical comments on the thesis work. This immensely helped me in polishing my skill and striving to do even better.

I would like to thank Prof. Christof Paar for sharing his insightful knowledge and lessons on security. The course work offered by prof. Sandip Kundu has helped me sharpen my skills in VLSI design and given me a well rounded perspective on the design processes.

I would really like to thank my Phd. mentor Dr. Georg T. Becker (now post-doc in Ruhr University, Bochum, Germany) without whose help and support this project wouldn't have been possible. Over time it was a great learning experience interacting and getting more insight into the field of hardware security and reverse engineering vulnerabilities. He was always accessible and ready to help, which definitely made working on this thesis a smooth process.

The love and support during the course of my master's has been immense from my parents Rajesh and Sonu Malik, brother Ishank Malik, grandparents Krishan Duggal and Sudesh Duggal. Also, being away from home my host family John Montanari and Karen Tarlow have always offered great encouragement and the spirit to be driven toward my work. I would like to express deep gratitude toward the Soka Buddhist community, my friends Amulya Gullapalli, Chaitra Narayana, Srinivas Santhanam, Swati Khanna, Gesine Hinterwalder and my labmates who believed strongly in my efforts and supported me throughout.

ABSTRACT

DEVELOPMENT OF A LAYOUT-LEVEL OBFUSCATION TOOL TO COUNTER REVERSE ENGINEERING

May 2015

SHWETA MALIK

B.E, MAHARISHI DAYANAND UNIVERSITY

M.S.E.C.E,UNIVERSITY OF MASSACHUSETTS, AMHERST

Directed By: Professor Wayne P Burleson

Reverse engineering of hardware IP block is a common practice for competitive purposes in the semiconductor industry. What is done with the information gathered is the deciding legal factor. [23] Once this information gets into the hands of an attacker, it can be used to manufacture exact clones of the hardware device. In an attempt to prevent the illegal copies of the IP block from flooding the market, layout-level obfuscation based on switchable dopant is suggested for the hardware design. This approach can be integrated into the design and manufacturing flow using an obfuscation tool (ObfusTool) to obfuscate the functionality of the IP core.

The ObfusTool is developed in a way to be flexible and adapt to different standard cell libraries and designs. It enables easy and accurate evaluation of the area, power and delay v/s obfuscation trades-offs across different design approaches for hardware obfuscation.

The ObfusTool is linked to an obfuscation standard cell library which is based on a prototype design created with "Obfuscells" and 4-input NAND gate. The Obfuscell is a standard cell which is created with switchable functionality based on the assigned dopant configurations. The Obfuscell is combined with other logic gates to form a standard cell library, which can replace any number of existing

gates in the IP block without altering it's functionality. A total of 160 different gates are realized using permutated combinations starting with 26 unique gate functions. This design library provide a high level of obfuscation in terms of the number of combinations an adversary has to go through increase to 2^{2000} approximately based on the design under consideration.

The connectivity of the design has been ignored by previous approaches, which we have addressed in this thesis. The connectivity of a design leaks important information related to inputs and outputs of a gate. We extend the basic idea of dopant-based hardware obfuscation by introducing "dummy wires". The addition of dummy wires not only obfuscates the functionality of the design but also it's connectivity. This greatly reduces the information leakage and complexity of the design increases. To an attacker the whole design appears as one big 'blob'. This also curbs the attempts of brute force attacks. The introduced obfuscation comes at a cost of area and power overhead on an average 5x, which varies across different design libraries.

CONTENTS

LIST OF TABLES

Table Page

LIST OF FIGURES

CHAPTER 1

MOTIVATION

1.1 Introduction

The complexity of integrated circuit (IC) is promoting reuse-based design to speed up the development of new products in a system-on-chip environment. A significant amount of time is invested into creating these reusable intellectual property (IP) cores [13] [2]. Violation of this reusable IP core poses a serious threat and exorts a sizable revenue from the IP producer. According to SEMI report [23] approximately $4 billion is lost due to IP infringement, which includes counterfeiting through reverse engineering, theft of trade secrets and trade marks [15].

The given IP block can be reverse engineered based on the compute power and

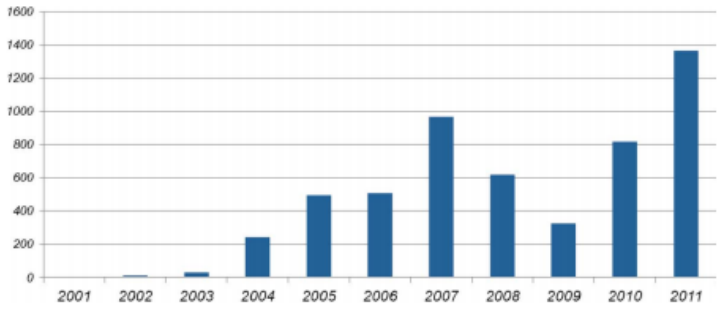


Figure 1.1: Cases of counterfeiting [15]

resource of an attacker. Reverse engineering involves the analysis of the functionality, architecture and technology of a device and representing them in a manner which allows reuse or duplication of the product [27] [25]. Post-manufacturing, an adversary or competitor get access to the product and use techniques to delayer the IP core and photograph each layer using a scanning electron microscope

(SEM) in order to reveal functionality of the circuit [11]. In some cases the goal of this process is not be to copy the IP, but to use it for competitive analysis, redocumentation of legacy systems and design improvements. Whereas an adversary will analyse the design and attempt to replicate the functional details to create an exact clone. This will be based on the assumption that the reverse engineering (RE) process for the adversary will be a short, and less expensive than a full prototype-development project [16]. Then the cloned products flood the market and compete for a share [19] [18]. In addition to the IP block, hardware implementation of security application such as PRESENT [6] and AES block cipher are also at a risk of being reverse engineered to extract key bits.

To curb the attempts of reverse engineering, schemes have been suggested by Cocchi et al [12] to use contact changes to hide the functionality and create lookalike. This approach ignores the fact, that connectivity of the design reveals important information. In this thesis, we aim at creating a low-overhead, piracy-proof obfuscation methodology using an obfuscation tool flow (ObfusTool) which not only hides the functionality, but also obfuscates the connectivity by insertion of "dummy wires". This protects and benefits the IP vendors and chip designers of the IP cores and security implementations. The approach is based on applying obfuscation to the design at the layout level involving dopant configuration changes.

The intent of this obfuscation technique is to hide the actual functionality of the hardware circuit by altering the mask and presenting lookalike gates. Dopant type variation are introduced at the transistor level, can change the functionality of the standard inverter/buffer gates to produce an always 1 or 0 gate. The obfuscation cell(Obfuscell) are combined with logic cells from the Nangate standard Cell Library to form a prototype. This prototype as discussed in chapter 3 and 4, is used to create an obfuscation library with 160 lookalike gates, realized from 26 unique gate functions. The entire library is attached to the obfuscation tool

flow, which is further utlilized to convert any vhdl/verilog behavioral code into an obfuscated mask.

The reverse engineering of the post manufactured obfuscated designs is really hard, due to the underlying assumption that dopant polarity detection is 16x hard under SEM processes [4] [26]. The connectivity of the design is obfuscated by the use of dummy wires, which are randomly selected from the intermediate nets of the design. The entire design appears like a one "blob" to the attacker, which is non-decomposable and hard to reverse engineer.

Major Contributions

1. Design a **Obfuscation standard cell (Obfuscell)** which has different functions based on the dopant type configuration. The Obfuscell will be further integrated with the different logic gates such as NAND4, AND2 to create a library using multiple functionality prototype as a building block.

2. An **Automatic Library Generation Tool** will be designed for the generation of all possible logic cells based on the prototype OBNAND4. This library would contain all the permutations for a given gate alongwith the dummy input information. Just by feeding in the logic gate and other related information, we would get an entire generated OBNAND4 library as discussed in chapter 5. This tool can be flexibly changed for including other logic gates for designing new libraries.

3. Design a flexible and adaptable **Obfuscation tool flow(ObfusTool)** which converts a .vhd description of logic into an obfuscated logic mask which is difficult to reverse engineer using optical inspection methods. There will be an ability to insert "dummy wires" in the cases where of 2 or 3 input gates are constructed using a 4-input gate. Random inputs can be selected for the gates using the supporting features in the ObfusTool. The tool will be flexible to handle different

requirements.

4. Different designs under consideration will be used to experiment across different obfuscation libraries and calculate the overall overhead involved. This will help in deciding the best obfuscation library design based on the trade-off between low overhead and high level of obfuscation.

Organization of Thesis

In the following chapter, we discuss the background work in the field of hardware obfuscation. Following chapter 3 and 4 go into details of constructing the obfuscation cell and library designs. The obfuscation library are designed by using 2 different approaches where the latter uses the designed Obfuscell. Chapter 5 talks about the automation design for creation of the obfuscation library based on prototype OBNAND4 consisting of 160 different gate layouts. Further on, in chapter 6 we discuss design of the obfuscation tool and how it incorporates the obfuscation library at different stages. Ultimately, we implement the Obfuscation tool on various security application including the PRESENT block cipher, AES 8 bit S-box and ITC'99 benchmark circuits.

CHAPTER 2

HISTORY AND PRIOR WORK

2.1 Background

Hardware intellectual property (IP) cores have emerged as an integral part of modern system-on-chip (SoC) designs. However, IP vendors are facing major challenges to protect hardware IPs and to prevent revenue loss due to IP piracy [8]. To prevent the IP violations various techniques have been suggested on both rtl-level and hardware implementation level.In following sections we shed light on these techniques.

2.1.1 Watermarking

Watermarking has been used to provide an ID to each chip to prevent the counterfeiting of the hardware device by malicious adversary. The presence of this unique ID is mostly kept hidden and when required an IP owner can prove his identity to the verifier. In case of an IP violation, and counterfeited good can be easily identified based on the watermark or in certain cases be non-functional. Watermarking can be applied to both hardware and software codes. In instance of reverse engineering, the watermark itself can be reverse engineered and render this countermeasure useless.

2.1.2 Obfuscation

Obfuscation is the process of intentional hiding of the functionality and structure of a device, to make it difficult, if not impossible to reverse engineer [12].

Obfuscation is different from watermarking which might be used to conceal the identity of owner within the content itself. Major security threats for hardware IP include (a) hardware intellectual property infringement during SoC design; (b) reverse engineering the manufactured ICs or the IC design database (in fabrication facilities) to produce counterfeit or clone ICs; and (c) malicious modifications of an IP through the insertion of hardware Trojan to cause in-field functional failure. [8] [5]. Hardware obfuscation aims at minimizing these threats at IP or chip level by making it difficult for an adversary to comprehend the actual functionality of a design. Hardware obfuscation is approached at 2 level software and hardware level.

2.1.3 Software Obfuscation

Software obfuscation is introducted at the front-end design level, where changes are made to the netlist to make it incomprehensible. In [9], the HDL (RTL) source code is modified by adding some comments and changing internal net-name followed by simple string substitution strategy. Semantic Design is a company which provides Obfuscation solutions to various companies in 20 languages to increase the security of intellectual property by scrambling the source code while allowing applications to run normally [14]. They come up with a source code obfuscator which accepts a source code from the owner and generates a functional equivalent of the original which is harder to reverse engineer. The obfuscator reads the comments, variable, indentation and converts it into nonsense name, making the code hard to read and decipher the actual functionality. Certain netlist level changes made chakraborty et al [9] [10] can be made by adding dummy code, adding always 1 or 0 nodes to the RTL, which makes it more obscure without changing the functionality. Although, there will be an involved trade-off between obfuscation and overall overhead. All these schemes preserve the functionality of

the design.

2.1.4 Hardware Obfuscation

Post-manufacturing the design information is utilized to recover the functionality of the hard IP core by reverse engineering. Cloned products flood the market making it a multi-million dollar business. The fabricated hardware can be for a security sensitive applications such as military and secured medical devices and it become necessary to be protected. Also, the threats of cloned products, IP theft, and copyright infringement necessitate semiconductor designers and manufacturers to implement countermeasures into their products [24]

In the past, several countermeasures have been used to protect the IC. SypherMe-

Figure 2.1: SypherMedia lookalike gates [12]

dia International (SMI) is a company is offering Camouflaging library for physical design which allows the manufacturer and designer to keep the key and strategic aspects of their designs secret from there competitors and counterfeiters [12]. In this approach, a designed library has cells which lookalike to standard gates, but might exhibit multiple functionality. These gates obfuscate the design functionality and make the process of reverse engineering complex. This approach has been widely used to provide secure design systems for pay TV.

A technique suggested by rajendran et al uses a mixture of real and dummy con-

tacts, they camouflage a standard cell whose functionality is one of many. It's hard for an attacker to identify the functionality of camouflaged gates, leading to incorrect netlist extraction [20] [21] [22]. Each standard cell has limited functionality of 3-4 gates. Both the above mentioned approaches, have ignored the connectivity of the design that might leak important information. In our approach, we not only obfuscate the functionality, but also hide this connectivity between the gates by introducing dummy wires. The OBNAND generated library based on an prototype can replicate the behavior of 26 unique gate and 160 permuted combinations which unique layouts. In the next chapter, we will discuss the main idea of dopant obfuscation and design of library based on modified gate.

CHAPTER 3

OBFUSCATION USING MODIFIED GATES

In [4] Becker et al, the dopant type changes are suggested to maliciously insert a hardware Trojan into an existing circuit at mask level without changing the functionality of the existing circuit. Similar approach is used in designing the obfuscation logic OBAOI222 to protect the layout post-manufacturing. OBAOI222 prototype is designed by integrating the modified inverter and buffer gates with the AOI222 gate from the Nangate open cell library. The modification to the inverter and buffer are made by changing the dopant type, which will be discussed in further detail under section 3.2. The difficulty in detection of the dopant is the main motivating factor to use dopant changes. According to Sugawara et al detecting a dopant is 16x harder than detection of a metal mask under specialized SEM process [26]. Especially, when the circuit is scaled to billions of transistors on the chip, it will be next to impossible to find the dopant type under the active region in each of the transistor. In the next section we discuss the obfuscation prototype using AOI222 in detail.

3.1 Overview of Obfuscation Prototype OBAOI222

The fixed configuration is obtained as shown in figure 3.1 has a 6 input AOI222 with input A0 and A1 connected to an inverter, B0, B1 and C0 to a buffer and C1 to an inverter. There will be 4 inputs provided by the user which would connect to 6 input based the gate to be formed. This prototype is used to mimic the behavior of different gates from the standard cell library such as AND, OR, XOR, NOR and NAND gate to name a few. Based on the configuration we will have some

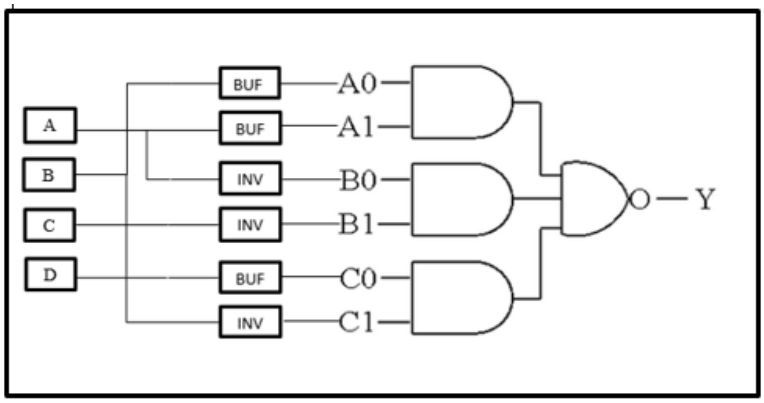


Figure 3.1: Obfuscation logic for the prototype cell using AOI222 gate

dummy inputs alongwith the real inputs. The description of these gates based on the AOI222 prototype is written in verilog and added to the existing standard cell library .v file. For example consider a 2input NAND gate which will have the truth table as in figure 3.2. In this case, we have 2 real inputs A & and 2 dummy

A0(BUF)B	A1(BUF)A	B0(INV)A	B1(INV_0)C	C0(BUF_0)D	C1(INV)A	Y
0	0	1	0	0	1	1
1	0	1	0	0	0	1
0	1	0	0	0	1	1
1	1	0	0	0	0	0

Figure 3.2: Truth table for NAND gate using OBAOI222 prototype

inputs C & D. A and B connect to 2 inputs each, whereas C and D connect to 1 input which would be an always0 or always1 gate. So, the output of this gate takes into account the 2 real inputs and give the output as a 2-input NAND gate. Similar approach is used to design the other gates, which complete the library with a total of 12 gate. AOI222 was selected as a design, since we could obtain 2-input XOR gates which is a common operation in s-boxes. 4-input to 6-input transformation is used to confuse the attacker in attempts of brute force.

Next section will focus on the idea of dopant type changes described in further detail.

3.2 Basic Idea of dopant type variations

Switching the type of dopant under the active region can lead to changed functionality of the transistor. In case of the PMOS, the doping at the drain is changed from p-implant to n-implant which connects the output to VDD. In case of NMOS the output would be shorted to GND.

3.2.1 Dopant Obfuscation by modifying gates

Dopant modified cells are multiple transistor gate which can have varied function, under different doping scenarios. Various 2-transistor gates are considered from the Nangate standard Cell Library to demonstrate the effect of dopant type variation on their functionality. At layout level, all the configurations may look different, but to the eyes of an attacker using optical reverse-engineering these changes would be less visible. This approach will be successful in confusing the attacker of the true functionality of the device under consideration. This confusion can be referred back to the fact that it is difficult to detect the polarity of an active region using optical reverse engineering. Whether an area is doped with p-implant, n-implant or not doped at all, it will be hard for optical reverse engineer to detect the changes. It is also difficult to detect the borders and type of well used [4]. Still detecting a well type, might be easier based on the process technology. In the following section, the existing gates will be modified by changing the n-implant and p-implant mask, which will be suffice for the obfuscation logic. These modifications will be performed on existing inverter and buffer design from the standard cell library.

3.2.2 Modification of the Inverter cell

In this section, we will demonstrate, how the obfuscation techniques can be applied to an inverter logic by manipulating the p-implant and n-implant mask. An

inverter design consist of an PMOS and NMOS transistor and performs the inversion logic as shown in the cross-section of the inverter figure 3.3.The modification is made in the PMOS, which is doped with the n-implant over an n-well, that shorts the output to VDD as shown in figure 3.4b. The unmodified layout of

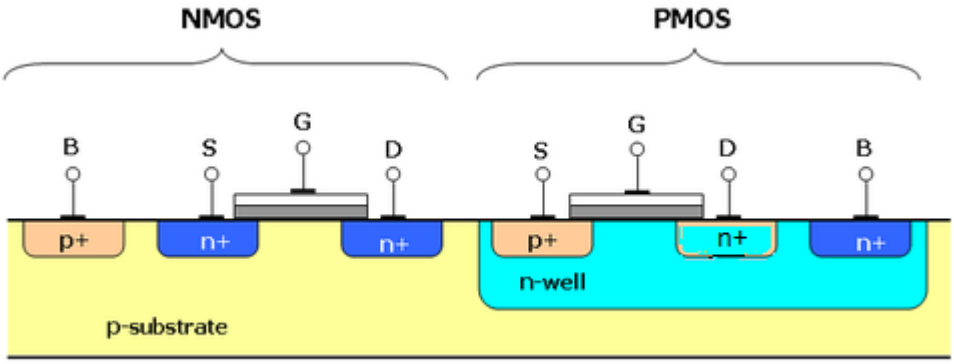
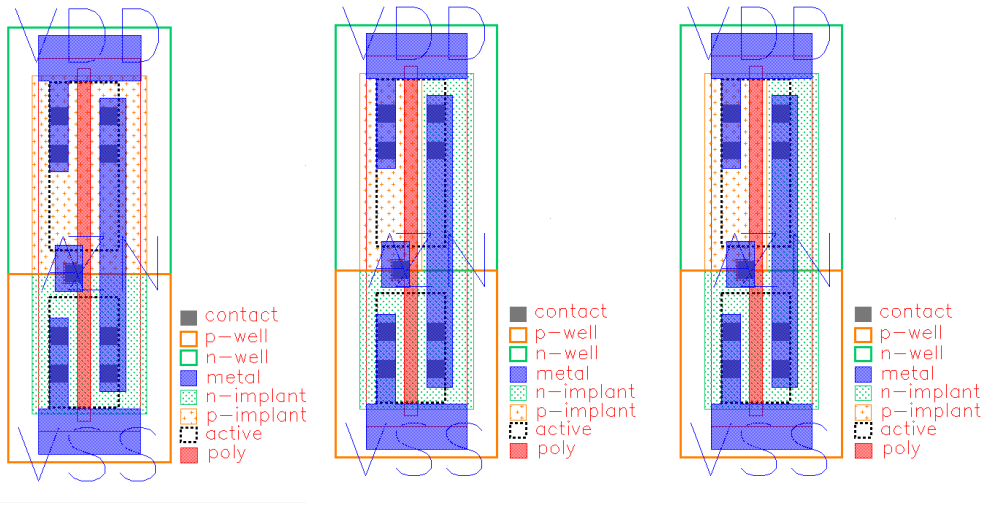


Figure 3.3: Cross-section of a Modified Inverter

the inverter gate taken from the Nangate Open Cell Library in figure 3.4a. The effective width of an PMOS or NMOS is determined by the width of the dopant area. In the modified layout in figure 3.4b, the p-implant on the drain is changed to n-implant at the contact. The source still consist of p-implant connected to VDD. The drain is disconnected as the implant is same as the well type, and the output is shorted to VDD independent of the inverter input.

The resulting behavior of the always 1 transistor configuration is as follows: When the input A=0, the source is connected to VDD using p-implant but the drain has n-implant which is same as the well type doping. So, the output becomes independent to the input and is shorted to VDD. Now, when the input A=1, the strong NMOS will try to pull down the output to a value close to GND. Since, the PMOS and NMOS of equal strength are used for this design, the output will fall to a less than VDD, due to the pull-down force of the NMOS transistor. But, irrespective of the input value the output will remain close to VDD. Although, there will a high input dependent power consumption if the input is 1 than 0.

Similarly in the design in figure 3.4c the modifications are made to the NMOS



(a) Simple Inverter (b) Modified Inverter for always 1 (c) Modified Inverter for always 0

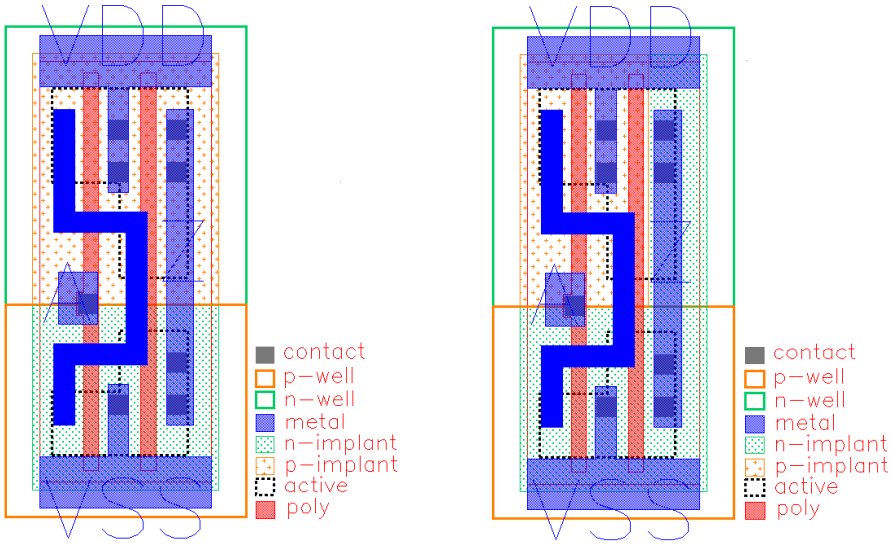
Figure 3.4: The two configurations are obtained for an inverter working as an always 1 and always 0 logic

to get a resulting always 0 gate. The dopant area of the drain at the contact is doped with p-implant. This cuts-off the drain and the output is shorted to GND. When the input A=1, then the output of the NMOS should be 0. Since, the output is shorted to GND it becomes independent to the input condition. In case, when A=0 the PMOS tries to turn on and pull-up, which leads to the output to be not completely grounded to GND.

To solve this problem in the always1 and always0 gate, the NMOS and PMOS are respectively cut-off. This is done by introducing p-implant in the source of the NMOS to cut-off the path from GND to VDD in case of always1 gate. When the NMOS is cut-off, irrespective of the values at the input, the output is always 1. For always0 logic circuit the n-implant is introduced at the source, disconnecting the path from VDD. The output always remains grounded and highly input data dependent power consumption is reduced. The output becomes independent of the input 1 or 0. Using dopant type changes, 3 type of gate functions are obtained from a single logic design.

3.2.3 Modification to the Buffer Cell

A buffer is used to re-inforce a weak signal in a design, with the output is equal to the input. In the buffer layout, similar approach of dopant modifications is used. Consider an unmodified Buffer gate, as in 3.5a, which has staggered inverters that are progressively sized due to loading effect. The two PMOS have an active area, p-implant in an n-well. The dopant near the drain of the PMOS close to the output is changed to n-implant as in figure 3.5b. The output as mentioned



(a) Buffer Design (b) Modified always 1 design

Figure 3.5: The two configurations are obtained for an obfuscation buffer gate and always 1 gate using the layout pattern with switchable dopant type

in the section above is shorted to VDD. The NMOS is cut-off by applying a dopant change near the GND connection to the source. This is done to prevent high input data dependent power consumption, which arises due to the pull-down affect of the NMOS transistor when the input A=1. Hence, the output will be tied to VDD, independent of the input value for an always 1 gate. Similarly, the alway0 gate is obtained by replacing the n-implant in NMOS with p-implant at the drain. Three different logic styles are obtained under varied doping profiles.

CHAPTER 4

OBFUSCATION STANDARD CELL DESIGN LIBRARY

4.1 Overview of prototype OBNAND4

The approach of obfuscation used in this thesis is to create a logic block with multiple functionality based on dopant variation. The logic cell will work like a programmable device which switches the functionality based on the dopant type changes and the applied inputs. The layout of this OBNAND4 logic cell will remain same at the metal and polysilicon mask level for different configurations. The only thing that will change is the dopant type mask. The final experiment design which is placed and routed using obfuscation library(OBNAND4 library). The OBNAND4 design in figure 4.1 uses a 4-input NAND gate from the existing

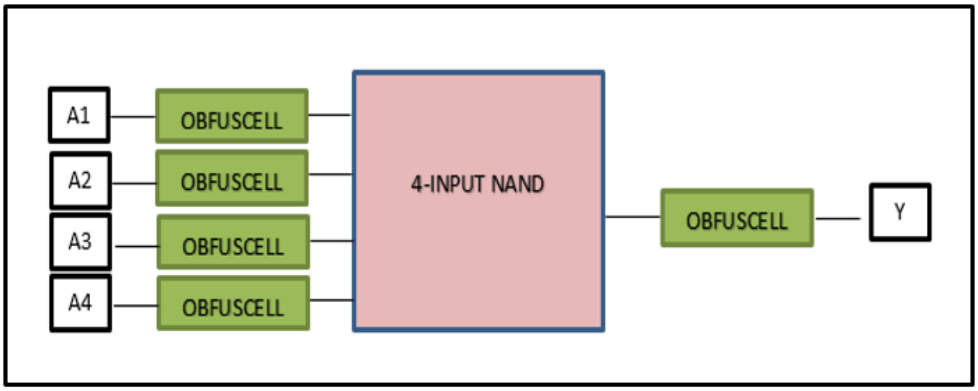


Figure 4.1: Obfuscation logic using Obfuscell and 4-input NAND gate

45nm Nangate standard cell library and integrates it with the obfuscation standard cell (Obfuscell). The Obfuscell which has a common layout is inserted at each of the input and output which can function as a inverter, buffer, always1 and always0 gate. These different functionality arise by altering the dopant in the

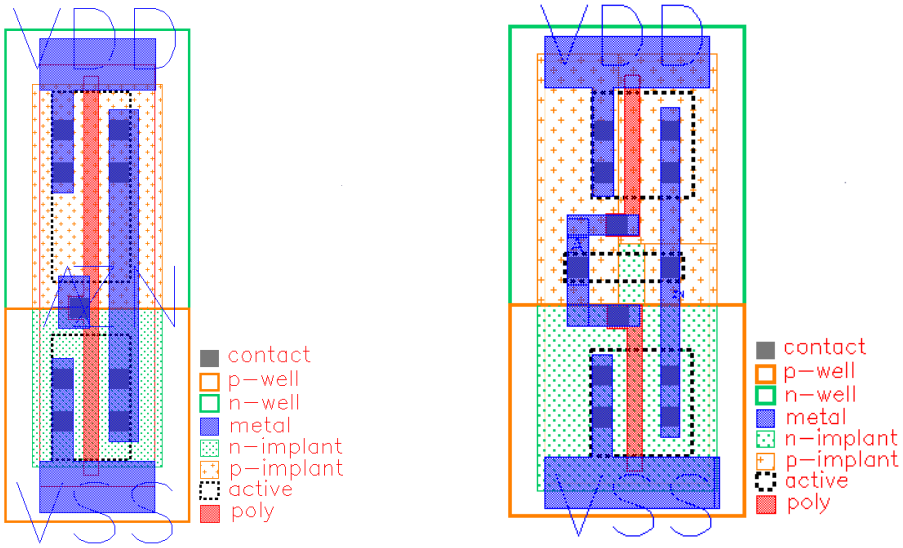
same layout design. This leads to variation in the functionality of the prototype OBNAND4 making it a versatile design. In the next section, we discuss in detail the design of the Obfuscell with it's different configurations.

4.2 Obfuscell standard cell design

In this section, a standard cell design is discussed, which together with other gates will form a prototype OBNAND4 to improve the level of obfuscation and also reduce the overall design area. The custom designed layout is shown, consisting of an inverter design with a dopant variation. The layout is similar to that of an inverter but an active region is introduced between the input and output. The same layout is used with switchable dopant to form 4 different configurations. Just by changing the dopant type the standard cell is made to function differently. In the next section, each of the dopant configurations are discussed in further detail starting with the inverter gate.

4.2.1 Obfuscell layout explanation

The mask of the obfuscation inverter is similar to a conventional inverter, but the differences arise with changes made by adding a doped active region between the input and output. The height of the cell is kept the same as Nangate Open Cell Library. In fig 4.2b is an inverter from the Nangate open cell library and an Obfuscell with equal PMOS and NMOS width almost to the ratio of 1.5:1. The weak PMOS is due to the compromise made to keep the height of the standard cell the same as the rest of the design library and confine to DRC. There is an active region between the input and the output which is switched based on the dopant type and the desired function configuration, a small doped region is introduced between the active region with n-type doping that cuts-off the connection between the input A and output Y.



(a) Simple Inverter (b) Obfuscell Inverter

Figure 4.2: Comparison is shown between a simple inverter layout and an Obfuscation Inverter layout

Now, the device works as a normal inverter with a slower drive of p-devices, since the ratio of P:N device is 1.5:1. The characteristic are shown in figure 4.3.

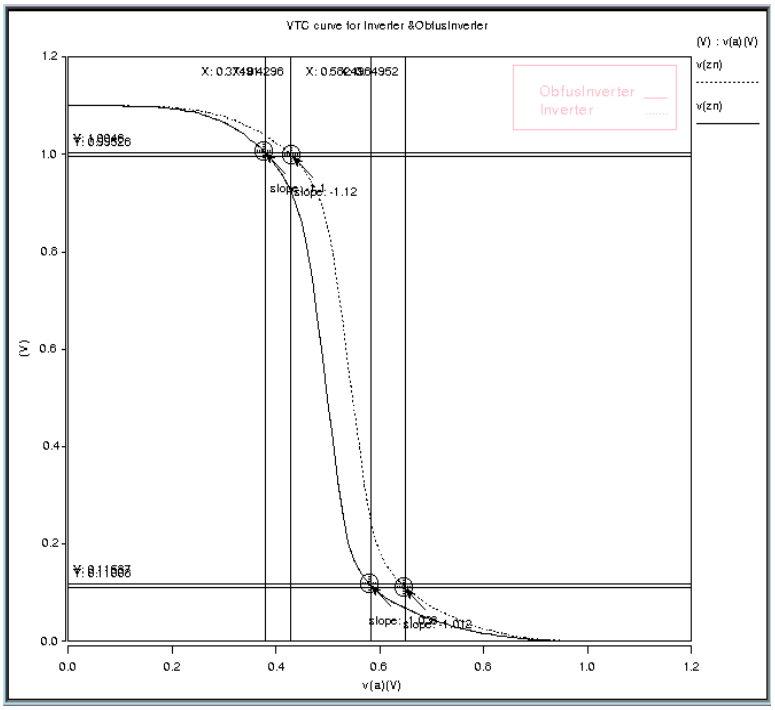
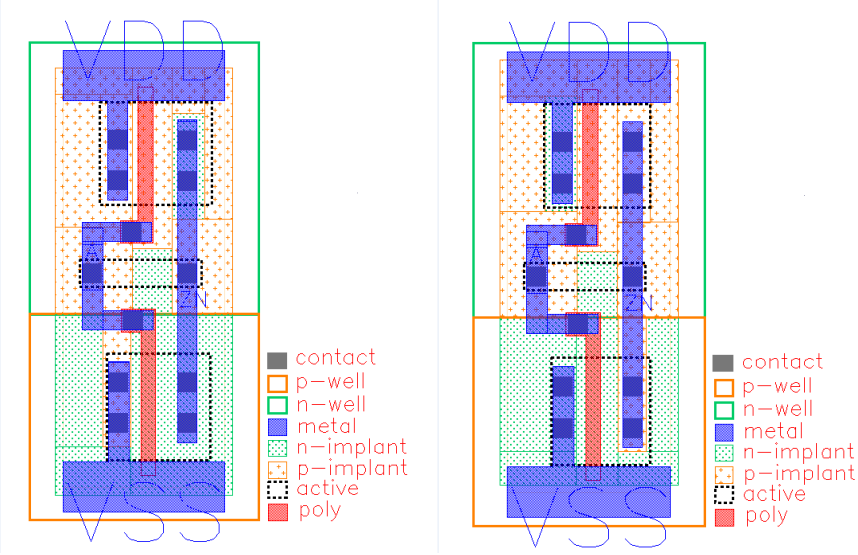


Figure 4.3: VTC curve comparison for Obfuscell and Inverter

4.2.2 Always 0 and always 1 standard cell design

To the existing inverter design standard cell dopant changes similar to chapter 3 are introduced as shown in 4.4b where the drain of the NMOS is cut-off by changing the doping from n-implant to p-type implant near the drain. The source of the PMOS is cut-off from the dopant near the contact. When the input A=0, then the output is 0, as the output Y is shorted to GND. When the input is A=1 ,then also the output Y=0. Hence, irrespective of the input A, the output will always be 0. Same approach is followed while creating an always 1 gate in



(a) Obfuscation always 1 gate (b) Obfuscation always 0 gate

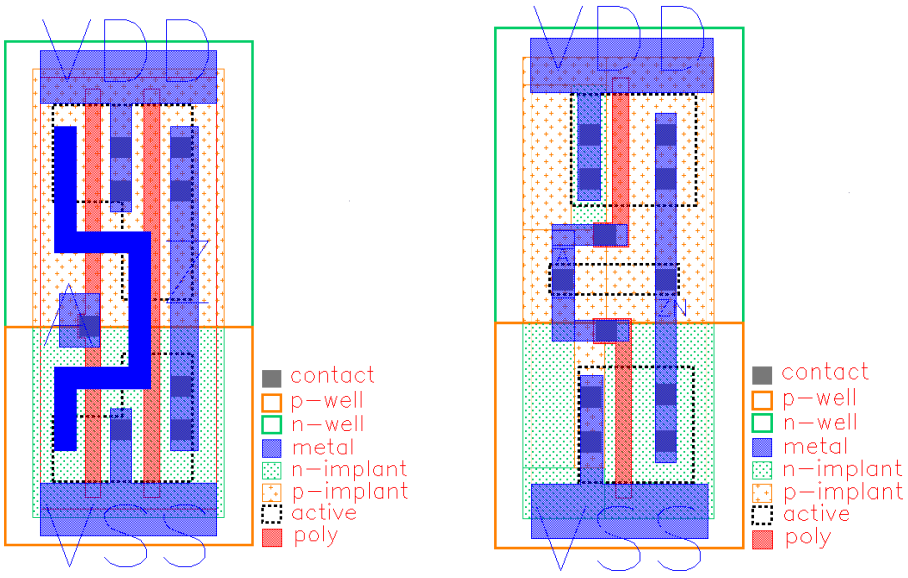
Figure 4.4: The two configurations are obtained for an obfuscation always 1 and 0 gate using the switchable dopant Obfuscell layout

4.4a, where at the drain the dopant is changed to a n-type in order to connect the output Y to VDD as shown in 4.4b. To prevent the noise from the NMOS , P-dopant is introduced at the source to cut-off the transistor. Hence, when the input A=0 , the output is 1 and when the input A=1 the output Y=1 irrespective of the input

4.2.3 Explanation of buffer configuration of Obfuscell

Conventional buffer design as shown in 4.5a consist of 2 back to back inverters where input A is equal to output Y, with differently sized inverters to improve the drive strength of the input. Whereas the design in 4.5b consist of single inverter layout with 1:1 ratio of P and N device. The active region between the input A and output Y is turned on by using the p-type dopant, which conducts the input to the output with A=Y for A=1 and A=0. The dopant changes are introduced where VDD and GND connects to the source of PMOS and NMOS respectively. The small region of contact in the PMOS on the source the dopant type is changed to n-type to disconnect the device.

Similar changes are made to the NMOS device to change cut-off the transistor.



(a) Buffer design (b) Obfuscell Buffer Design

Figure 4.5: The two configurations are obtained for an Obfuscell buffer and Obfuscell always 1 gate using switchable dopant

The resulting gate have both PMOS and Nmos devices cut-off and input conducts to the output through the active region. This is the standard cell design for the buffer design with minor variation in the boundary from the conventional buffer

design. Resulting all layouts are same at mask level and only with variation in dopant type they change to INV, BUF, always0 and alway1 gate.

4.3 Extracting LIB file from the Obfuscell layout using ELC

The custom made layout was designed using cadence virtuoso. Encounter Library characterizer was used to generate the respective output files for the standard cell design. While using ELC 3 input files are required. 1. Hspice netlist of the schematic was extracted from the drawn schematic. 2. Model file 3. ELC setup file as shown in figure 4.6

Model file: The model file for 45nm technology node is used and the spice netlist is extracted based on the inverter schematic in cadence virtuoso is inserted into the setup file.

ELC setup: The file consist of the process parameter such as threshold voltage, temperature, rise and fall time, which are pre-defined for the library creation for specific NMOS and PMOS device. It also contains additional information pertaining to the loading capabilities of the inverter. These files are used for library characterization of the standard cell and give an output format of a .lib file, which is added to the existing library for the Nangate Open Cell Library.

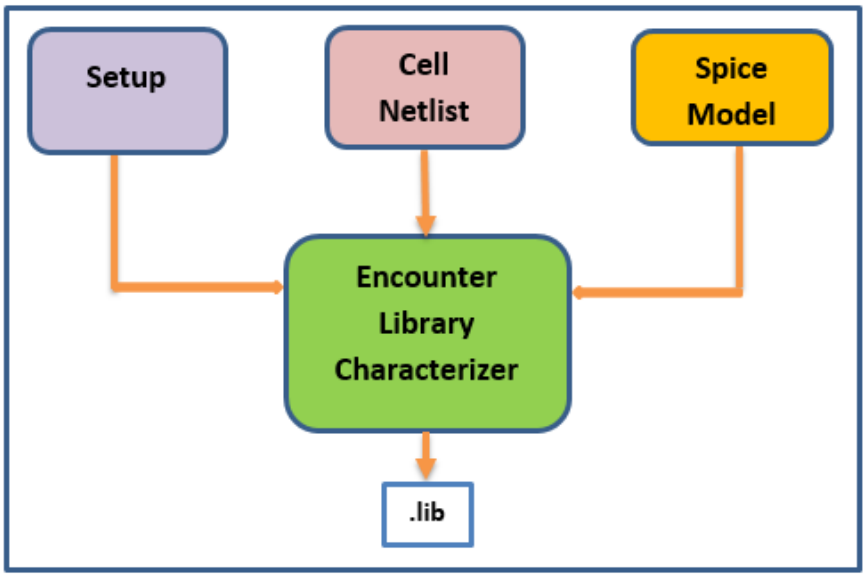


Figure 4.6: Encounter Library Characterization Flow

Once this .lib file is obtained the standard cell description can be included in the library.

Abstract extraction: From the existing layout of the designed cell an abstract is extracted to provide the metal layer information. This process is repeated for all the configurations including BUF, always1 and always0.

Liberty File Creating: For the place and route a liberty file is created, containing the information of the boundary and metal width for the entire standard cell library. Using candence virtuoso, a stream out .lef file is exported out using an input design library including the designed Obfuscation standard cell.

CHAPTER 5

AUTOMATIC LIBRARY GENERATION TOOL

5.1 Basic building block for standard cell Library

The prototype OBNAND4 design 4.1 as discussed in chapter 4 is the basic building block for the obfuscation standard cell library. The logic gates in the library can perform the function of 2-input gate, 3-input gate and dummy wires can be inserted on rest of the inputs.

An example is shown for a 2-input NAND gate in figure 5.1 with it's truth table

A (Obfuscell BUF)	B(Obfuscell BUF)	C(always1)	D(always1)	Y (Output)
0	0	1	1	1
0	1	1	1	1
1	0	1	1	1
1	1	1	1	0

Figure 5.1: Truth table for a 2-input NAND formed using OBNAND4

where all input A and B are the true inputs and the Obfuscell works as a buffer, and the other two inputs are an always 1 Obfuscell. The output Obfuscell behaves like a buffer and OBNAND4 is made to function like a 2-input NAND gate with 2 dummy inputs. The underlying layout of all the Obfuscells are the same and they differ only in the dopant configuration used to program them. Similarly, other gates are realised using the OBNAND4 prototype which are further discussed in section 5.1.

5.2 Creation of obfuscation library using OBNAND4

Using the standard cell library based on OBNAND4 gate combination the obfuscation library database is created with 26 unique gate which include universal gates and additionally include gates such as A+ $\bar{B}$. In addition to the gate name, the database consists of number of inputs, dummy inputs, real inputs and inversions. Inversions are required in case of gates like A+ $\bar{B}$.

5.2.1 Database for unique gates

Unique gate database is used in creating the permuted combinations for each gates and extending the library to 160. The goal of the extension is to reduce the design area and power by bringing down the obfuscation gate count of the final design with minimal performance overhead. The extra inputs in two and three input gates will be deployed in obfuscating the connectivity of the gates in addition to the functionality.

The verilog gate description is required for these 160 obfuscation gates alongwith the connectivity to the dummy gates. To complete out obfuscation library with a large number of gate combinations, an automation environment is being setup which is further elaborated in the next section.

5.3 Automating the obfuscation library generation

To extend the obfuscation library based on OBNAND4 configuration an automation will be written in **perl** to generate all the possible combinations from the prototype design. As, shown in figure 5.2 the tool starts with the Obfuscell and the logic under considerations. In this thesis, I have considered a 4-input NAND gate to design the obfuscation library. Using a 4 input NAND gate gives us the maximum number of gate combinations alongwith sufficient number of dummy wires. The list of 26 unique gates is fed into the Automated Library generation

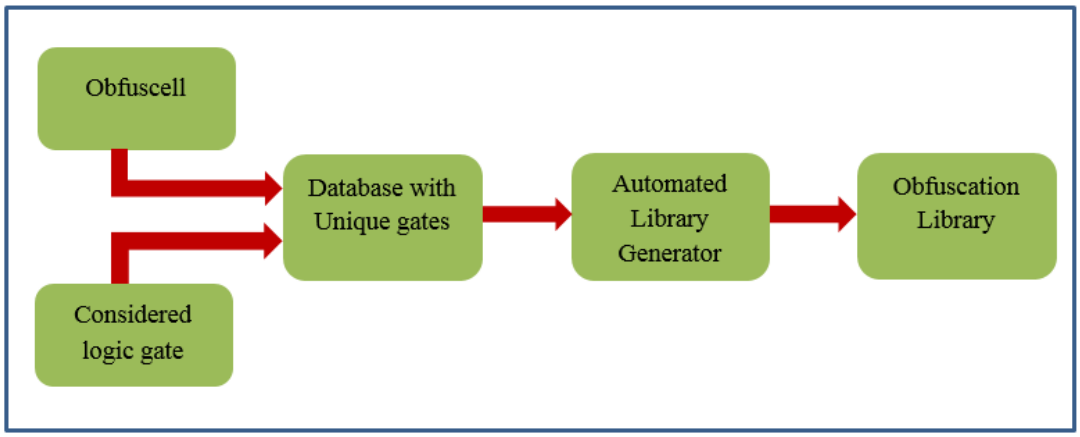


Figure 5.2: Automated Library Generation Flow

tool. There are 2 parts to this tool which create an extended database followed with the verilog description for each gate.

5.3.1 Creating database for complete library

With the information provided by the 26 unique gates is read into the tool. For each gate the real input and dummy wire information is associated with the final gate name. Example if the gate name in the unique gate list is OBNAND4_NAND2, then the final name will be interpreted in terms of 4 inputs and will be OBNAND4_NAND2_real_real_dum_dum.

Each of these 4 inputs will have permuted combinations. What we mean here by permuted combinations is that all possible input combinations for the 4-input gate will be created. In case of the two real and two dummy inputs for 2 input NAND gate we get 6 different unique layout configuration as in fig 6.3. The number of combinations change based on the type of unique gate considered and is included in the database. Any 2 input NAND gate in the design under consideration can be replaced by any of the combinations to obfuscate the inputs. The final generated .csv file consists of the new gates,real and dummy inputs, inversions and position of each of the inputs.

5.3.2 Verilog description generation for library

The .csv file is read into an automation for generating the verilog description for each of the gates matching the inputs with the instances using a hashing function. One of the module gate descriptions is shown in figure 5.3, generated as the output from the automatic library generation tool.

The two real inputs A and C are connected to the Obfuscell buffer and the

```
module OBNAND4_NAND2_realA_realC_dumD_dumB (A,B,C,D,ZN);
input A,B,C,D;
output ZN;

wire n1,n2,n3,n4,n5;
wire A,B,C,D;
Obfuscell_BUF U1 (.A(A), .ZN(n1));
Obfuscell_BUF U3 (.A(C), .ZN(n2));
Obfuscell_always1 U2  (.A(D), .ZN(n3));
Obfuscell_always1 U4  (.A(B), .ZN(n4));
NAND4_X1 U5 (.A1(n1),.A2(n2),.A3(n3),.A4(n4),.ZN(n5));
Obfuscell_BUF U6 (.A(n5), .ZN(ZN));
endmodule
```

Figure 5.3: Verilog Code for 2-input NAND gate

instances are matched accordingly using a hash table. The dummy inputs are connected to B and D using a Obfuscell always 1 gate. This gate gives an output 1 irrespective of the input. A buffer is connected at the output for the specified gate function. Similar description is automatically written for all the other 160 gates and used by the place and route step in the obfuscation tool flow.

5.3.3 Flexible feature of library generation tool

The tool has the capability to be extended to other logic gates from the Nangate Open Cell Library, with minor changes, making it a more generalized. At the moment, I have only experimented with two standard cell library, but the research can be extended to different library and used alternatively to achieve highest

obfuscation with an area efficient approach.

5.3.4 Testing different gates

5.3.5 Overhead analysis of Obfuscell based standard cell library

The some of the basic gates formed using 4.1 are compared in terms of their area overhead in 5.4. There is atleast a 400% increase in the area of the obfuscation prototype while creating an 2-input AND and OR gate. The maximum area

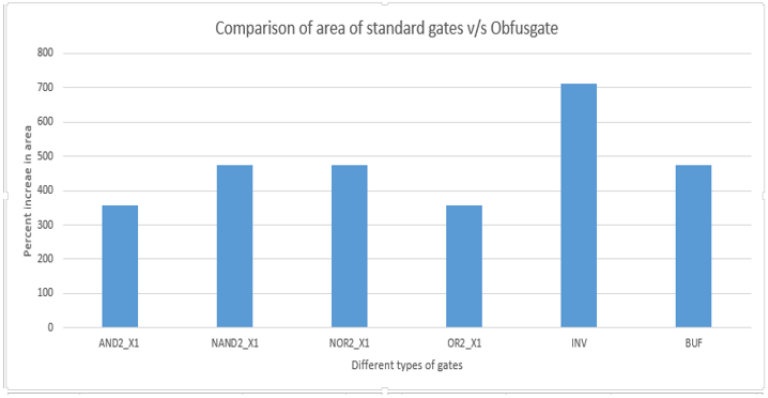


Figure 5.4: Area overhead for different cell compared to NAND4 obfuscation gate configuration

overhead is seen in case of inverter which is 700% increase over a conventional inverter design. This increase in area is due to the fact that Obfuscell uses 6 gates compared to 1 gate used to design the conventional design cell.

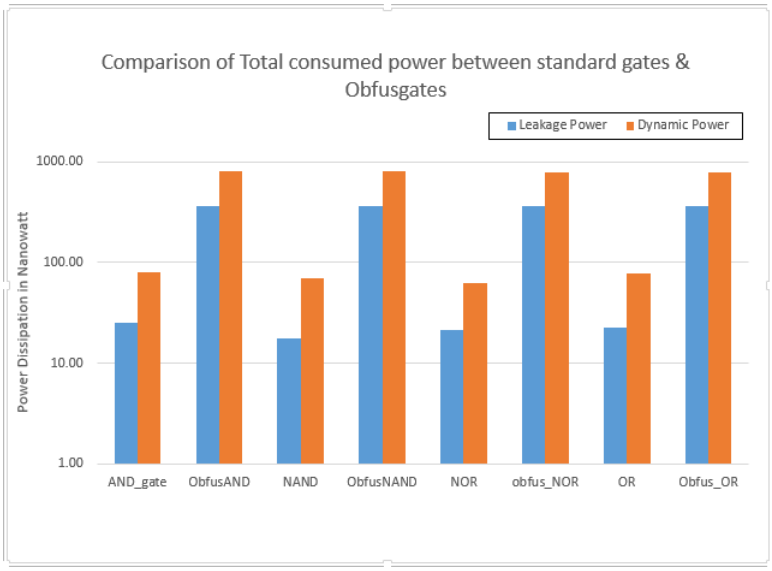


Figure 5.5: Power overhead for different cell compared to NAND4 obfuscation gate configuration

Comparing the universal gates for their power consumption based on figure 5.5, it is mostly uniform in case of obfuscation gate,since they have the same layout pattern. In case of universal gates the AND and OR gates higher leakage and dynamic power due to the required inversion. Roughly the power overhead is about 9x in case of dynamic and 10x for leakage power. Leakage is high due to the use of dummy inputs which are used to confuse the attacker.

CHAPTER 6

OBFUSCATION TOOL FLOW

6.1 Obfuscation tool needs

Implementations of obfuscation in hardware requires a working tool flow to convert the existing design with obfuscating logic. The obfuscation is achieved by using the prototype standard cell replicating the function of different gates based on the selected inputs and configuration of dopant type used. These changes can be introduced in the entire circuit or a part of the IP block depending on the degree of obfuscation required and the amount of money one is willing to spend. The tool will be able to manipulate these variations depending on the response from the designer. The various stages of the tool are discussed in the next section starting with the RTL-level netlist and ending with the generation of an obfuscated hardware circuit.

6.2 Steps in obfuscation tool flow

The design under consideration as in 5.2.1 will go through the ASIC tool flow along with additional in between step for introducing obfuscation. In figure 6.1, the various steps are mentioned, which will be discussed step wise.

6.2.1 Designs under consideration

This work will focus on two major areas of applications. One would focus on the importance of obfuscation in cryptographic designs to prevent safety critical application from being reverse engineered. The other application area will be

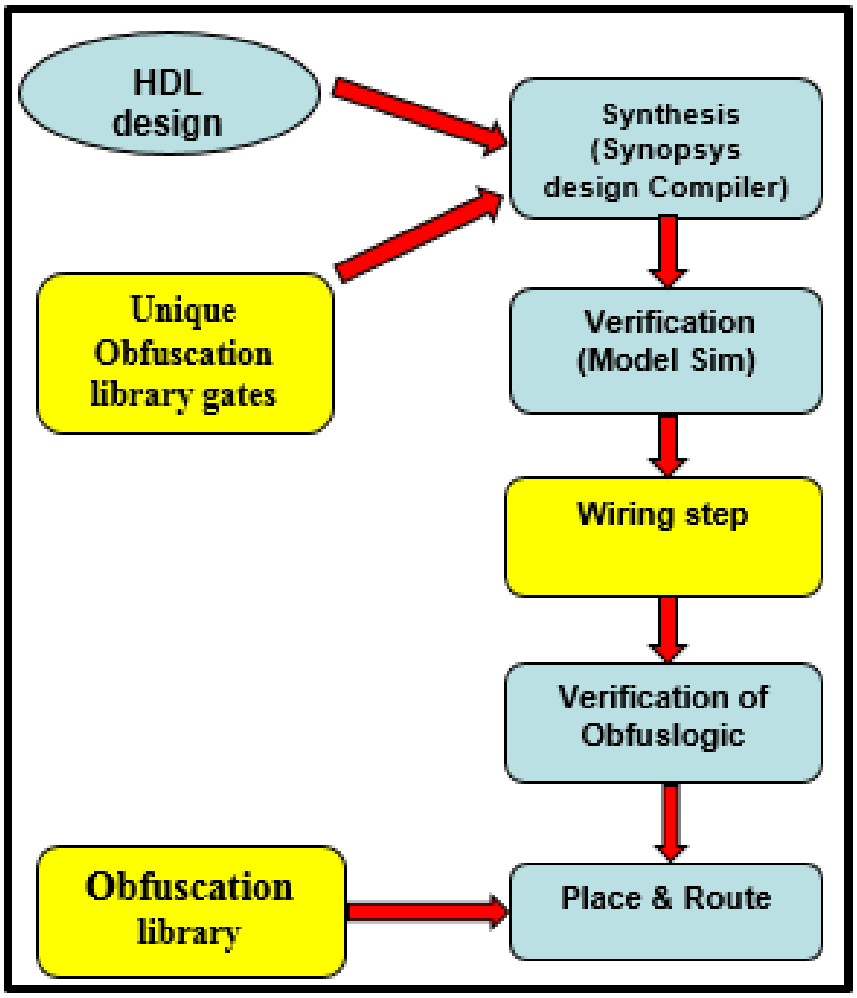


Figure 6.1: Obfuscation Tool Flow

protection of IP blocks from counterfeiting.

Security application

Security sensitive applications deploy some form of encryption with the on-chip storage of a master key to protect the information from an adversary. The hardware circuit storing the key is susceptible to reverse engineering attacks post manufacturing. Obfuscation can be applied to a part of this circuit that stores the key making the attempt to reverse engineer difficult. In this thesis, we have used various cryptographic circuits for the purpose of experiment. Mostly lookup

table based implementations of substitution-box or sboxes is considered which operates on the plaintext and key to get the resulting encrypted ciphertext. Taking an example of an 8-bit lookup table based sbox, the netlist is converted into a gate level description using Nangate standard cell library. From the attacker's perpective they are looking at set of gates while delayering the chip and learning about the functionality of the design. The design under consideration is further discussed in detail in the section below.

Sboxes

Sboxes or substitution boxes are basic component of symmetric key algorithms which performs substitution. They are used to obscure the relationship between the key and the cipher text. They take a number of input bits m, and transform them into some number n based on the substitution table, where n is not necessarily equal to m. An 8 bit m*n lookup table based sbox implementation has been considered with 2m words of n bits each. In the Figure 6.2, 16 bit s-box has been shown, which take a 16 bit input with 2^{16} possibilities and give an output of 16 bit. An 8-bit s-box consisting of two 4-bit s-boxes is considered for experiment purposes.

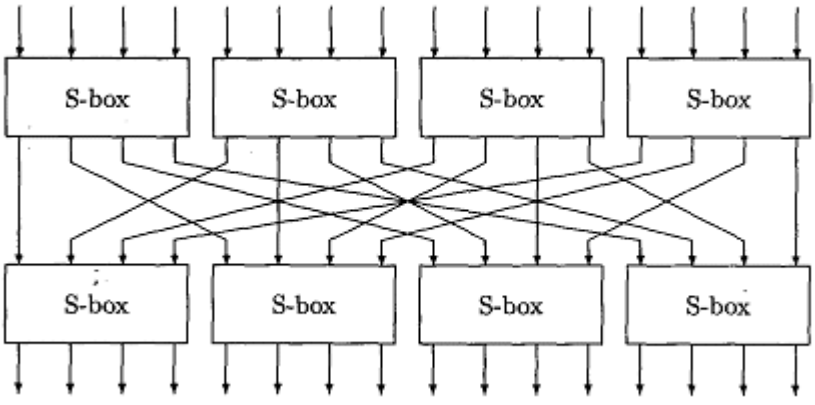


Figure 6.2: A 16-bit s-box design

Post-manufacturing an IP block can get into the hands of untrusted third party, who can use optical reverse engineering to learn the functionality of the circuit and replicate it. Hence, it is becoming important on the part of the design units to introduce some protection to prevent illegal clones of the IP block from being created. Obfuscating this logic using dopant changes can confuse the attacker and make the attempt of reverse engineering difficult. To demonstrate and test the obfuscation flow we would consider different ITC'99 benchmark circuits. The entire logic will be passed to the tool which would select the obfuscation gates from the library to synthesize it and then further add "dummy wires" to the corresponding obfuscation cells. The tool will changes the appearance of the logic at layout level*mask* and additionally obfuscate the connectivity, but the functionality will remain same.

6.2.2 Logic synthesis and simulation

The first step in the flow is the logic synthesis of the behavioral netlist, which would output a structural netlist with the design described in terms of logic gates. In process of synthesis, different constraints are to be specified to the design compiler. A perl based script is used to pass the command and constrains to the compiler for logic synthesis. The input file is a .v or .vhd file of the design under consideration and a obfuscation library with unique set of gates at 45nm technology node is used to map the design to a gate level description. Each of the unique obfuscation gate information will be included in the Nangate Open Cell having the same information for area parameter. This will let the tool make an unbaised decision on gate selection for the given design. Since, all the unique gates are based on the OBNAND4 prototype, so all of them will have the same area, irrespective of the functionality. The mapping of gates is constrained by

other factors such as the availability of logic gates in the technology library, the drive strength, delay and power. These constrains help in calculating the overall efficiency of the IP block.

6.2.3 Random selection of Inputs and dummy wires

After the IP block is verified, we perform the replacement of the gates in the in the logic with the obfuscation gates. Each of the obfuscation gate will perform the function of multiple gates based on the switching configuration of the dopant type. The obfuscation gate prototype will be replaced for all gates in the design, but will add an overhead in terms area and power. In the following 2 chapter we will be following 2 different approaches to form the basic building block for the obfuscation logic.

Using both these approaches we will add dummy wires, in case there are extra inputs for the gate to be created. In addition to this, different inputs can be selected to form the same gate with different layouts (permuted combinations) as shown in figure 6.3. At this step, the tool will randomly select the obfuscation gate that will replace the original gate with the permutated combinations from a 160 gate library.

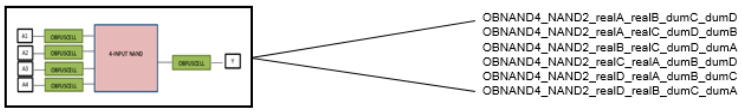


Figure 6.3: Permuted combination for 2-input NAND

For example, in case of an NAND gate depending on which dummy input is selected from the prototype gate, 6 different arrangements will be achieved 6.3. We can either replace the part of the design or the entire logic based on the tool setup. Once the replacement is complete the resulting design needs to be verified using model sim. The design is checked to make sure that the random

replacement has not caused any changes to its functionality.

Dummy wires: can be selected from a set of intermediate nets in the design under consideration. These will result in hiding the connectivity of the design and hence increasing the complexity of exhaustive brute force.

6.2.4 Place and Route

The final step in this flow is to place and route (P&R) the design to achieve the GDSII file, which is the final design file format which is sent out to the fabrication unit and contain the mask information. [7] Once, the synthesized netlist is received, the P&R tool is used to define the floorplan and place the design from gate level at transistor level in a block arrangement. The Obfuscell has been incorporated into the custom cell library and cane be used to places and route the design. Constraints can be specified based on the power, area, delay and clocking requirements.

CHAPTER 7

TOOL IMPLEMENTATION AND RESULTS

7.1 Integrating the Obfuscation cells in different Application

The design under consideration is synthesized using design compiler and the a gate level netlist is obtained by using this. After this the design is simulated in Modelsim to verify the functionality of the logic. Once the functionality is verified, the gate connectivity and random inputs are inserted as explained 6.2.3 OBNAND4 gate based obfuscation logic.

All the changed gates lookalike at the layout level, irrespective of being a AND, OR , NAND and NOR gates with different input configurations. This makes the detection of dopant type in each transistor infeasible. So, when the attacker delayers the chip and photographs each layer to decipher the functionality, it will be infeasible to do so. Also, another level of complexity is added by using differing input configurations. Similarly, the design can be placed and routed using the OBNAND4 approach which currently provided with 160 gates. Now the security applications will be discussed in further detail in the next sections.

7.1.1 Obfuscating PRESENT block cipher implementation

Implementation of PRESENT block cipher:

An low-cost cryptography solution PRESENT has been proposed by Bogdanov et al [6] to design an ultra-light weight block cipher. AES is an accepted cipher with excellent performance in hardware for widespread use, but PRESENT provides a hardware optimized solution targetting moderate security applications [6]. The

main focus of this design is to be come up with a more area and power efficient design for specific lightweight application such as RFID tags.

A simple top-level algorithm of the PRESENT is shown in fig. 7.1

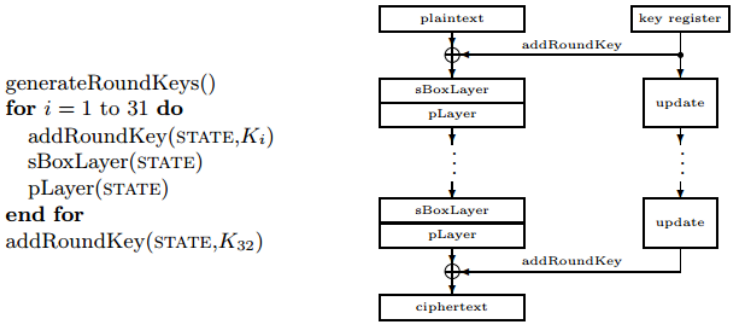


Figure 7.1: A top-level algorithm of PRESENT [6]

The PRESENT basically has a SP (substitution and permutation) network [17] consisting of 31 rounds as shown in fig 7.2 The block length is 64 bits with two key length of 80 and 128 bits are supported. For the purpose of testing the obfuscation tool we will consider the 80 bit key length. Each of the 31 rounds consists of an xor operation to introduce a roundkey Ki for $1 \leq i \leq 32$, a linear bitwise permutation and a non-linear substitution layer. [6]

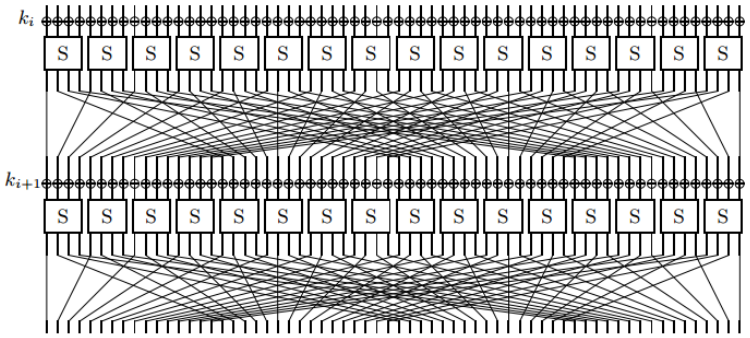


Figure 7.2: The S/P network for PRESENT [6]

In the thesis, for the purpose of experiment and to prove the point of hiding the connectivity in addition to hiding the functionality we implement the the

S-layer and P-layer.

S-box Layer: The non-linear layer uses a single 4-bit S-box S which is applied 16 times in parallel in each round for all the 64 bit. Each s-box used in PRESENT is 4-bit to 4-bit S-box S. The lookup table for each of the S-box as in fig 7.3 has hexadecimal notation.

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$S[x]$	C	5	6	B	9	0	A	D	3	E	F	8	4	7	1	2

Figure 7.3: Lookup table for the 4bit to 4bit S-box for PRESENT [6]

P-layer: The P-layer is a non-linear bit permutation used in PRESENT bit i of STATE is moved to bit Position P(i). The hexadecimal notation is shown for each of the 0 to 63 bits in the table below 7.4 assigning a position.

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$P(i)$	0	16	32	48	1	17	33	49	2	18	34	50	3	19	35	51
i	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
$P(i)$	4	20	36	52	5	21	37	53	6	22	38	54	7	23	39	55
i	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
$P(i)$	8	24	40	56	9	25	41	57	10	26	42	58	11	27	43	59
i	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
$P(i)$	12	28	44	60	13	29	45	61	14	30	46	62	15	31	47	63

Figure 7.4: Bit permutation for the each of the 64 bits [6]

Using obfuscation tool on PRESENT:

The considered implementation as described fig 7.2 is S-layer connected to P-layer through a set of non-linear permutations. The obfuscation tool is used on the hardware implementation of the PRESENT. Starting with the 16 set of S-boxes in parallel connected to the P-layer. The lookup table based .vhdl description is synthesized using the unique set of 26 obfuscation gates. At this stage the functionality of all the gates is hidden, since they all look alike at gate level, but function differently. After this stage, the gates are replaced by the permutated input combinations and dummy wires are introduced.

Input Selection

Example if the chosen gate at synthesize is OBNAND4_NAND2, which is a two input gate, hence we have 2 inputs where we can tie dummy wires. Also, any two inputs can be selected as the real input from a combination of A, B, C, D which would be selected from ${}^4C_2$. The selection of inputs is done randomly from the library database of 160 gates.

Dummy input selection

The remaining 2 dummy inputs have to be connected, but they will not affect the functionality of the logic. In the tool these dummy inputs are randomly selected and assigned from a pool of intermediate "nets". The randomization will not be truly random. Now, the connectivity between the gates is also obfuscated. The connectivity between the S-layer and P-layer will be hidden and it will appear like non-decomposable 'blob' as in figure 7.5 after place and route. It would just show the OBNAND4 cells connected to each other in a way the 1st cell could connect to the last or the next cell. This would be in a completely random fashion and would not affect the functionality of the design. In case of past approach used by SypherMedia, the S-boxes will be obfuscated but the connectivity between the S-boxes will be visible, which is obfuscated with the use of randomly selected dummy wires.

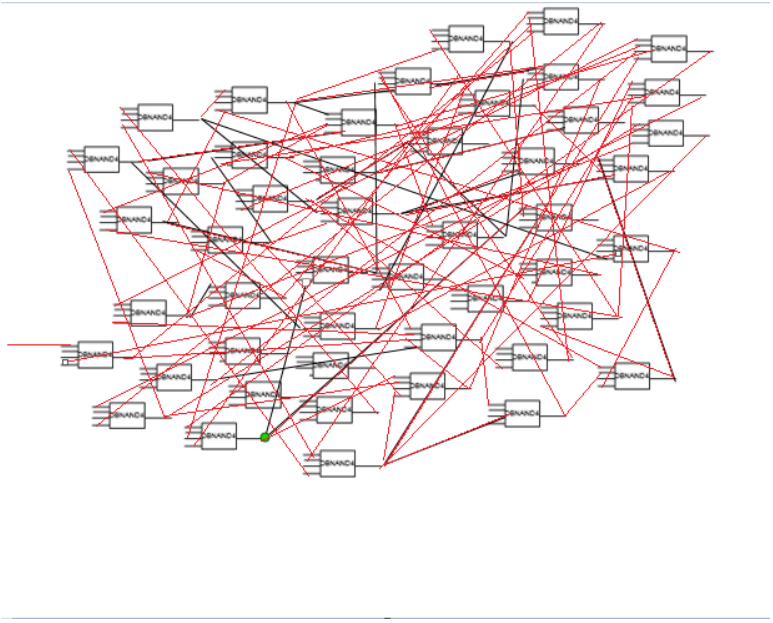


Figure 7.5: The blob

Each simulation through wiring step will result in a unique netlist. Hence, we can get multiple instances of PRESENT 7.6 with the functionality and connectivity equally complex in both the cases. It can be extended to N different netlist. The attacker is going to be confused with each of the implementation and hence prevent exhaustive brute force attack.

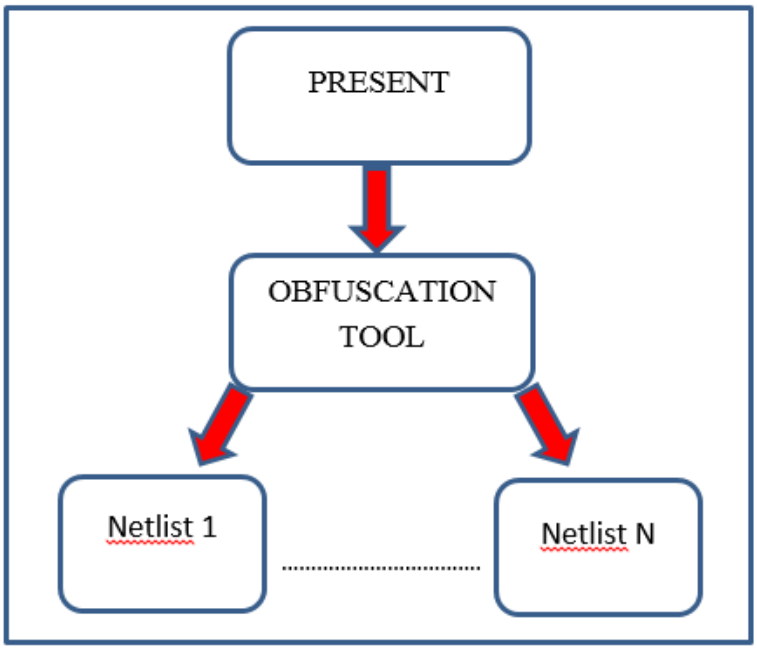


Figure 7.6: Multiple netlist generation for PRESENT

Overhead and performance across different libraries for PRESENT S/P layer

Different library design are analysed in this thesis starting with using AOI222, NAND4 and then AND2 in 7.2 .The experiment is based on an OBAOI222 library design has the highest overall overhead in terms of area and power. It has a low level of obfuscation with only 12 gate combinations that can be obtained from the fixed configuration used in Chapter 4 shown in 3.1. Obfuscell based approach

Parameter \ Library	AOI222	NAND4	AND2
No. of combinations	12	160	12
No. of gates	671	626	511
Brute force complexity	2^{2402}	2^{2241}	2^{3741}
Area	6.23	4.25	3.2
Power	8.61	5.65	2.59
Delay	3.23	2.51	2.12

Table 7.1: Overhead for different Obfuscation Libraries for PRESENT

seems like a winner in case obfuscation and the overhead. With the complete

library of 160 gates using the 4-input NAND the overhead is seen to reduce compared to incomplete library. This includes gates such as A+$\bar{B}$=Y which are not present in the Nangate Open Cell Library. Such combination logic can be used in synthesizing the 8-bit s-box design and improve the design overhead. All libraries are compared against a non-obfuscated design place and routed using anycell in the Nangate Open Cell Library. The level of obfuscation achieved in terms of number of combination an adversary will have to go through to reverse engineer the device goes up exponentially.

7.1.2 Obfuscating AES 8 bit S-box

The hardware implementation of AES 8 bit S-box is considered for the purpose of experiment. The Sboxes substitute a 8-bit input for an 8-bit output and are based on the arithmetic operation of finite field GF(2_8) [28].It operates on 128 bit block with a key size of 128, 192 and 256. The building block of the AES algorithm are non-linear Sboxes and Mix column operation. They are based on finite field arithmetic and have inverse functions used for decryption. The vhdl implementation is used to which is passed through the tool to get various results

Parameter \ Library	AOI222	NAND4	AND2
No. of combinations	12	160	12
No. of gates	882	865	650
Brute force complexity	2^{3127}	2^{3096}	2^{4758}
Area	10.14	7.09	6.03
Power	8.82	6.45	4.59
Delay	4.12	3.12	2.5

Table 7.2: Overhead and performance for different Obfuscation Libraries for AES 8-bit S-box

based on die-size (area) and power. The Sboxes transform 8-bit input to 8-bit output making this design much bigger than the PRESENT block cipher. AES is a widely used and accepted block cipher for high security applications. The

overhead for AES S-box is higher than PRESENT. With the increase in the area of the hardware implementation of S-box the area utilized by place and route increases due to the connectivity of the dummy wires. The power consumption also goes up but doesn't change as much as the area. The delay in this case doesn't change considerably, since we use standard cell 4-input NAND and the Obfuscell creates shorts in certain cases reducing the performance overhead by creating low resistance paths. An average delay of 2.5x has been noticed with a range of 1.5x to 4x.

7.2 VLSI applications

The main goal of the obfuscation tool is to have least area overhead with higher level of obfuscation which is seen to be achieved by using the OBNAND4 approach in case of the cryptographic hardware implementation of S-boxes. There is a need for proper estimation of area across different designs to get an average approximation. Based on the results across different benchmark circuits we see

Benchmarks	OBNAND4 area $um\hat{2}$	Area(Anycell) $um\hat{2}$	Area Overhead
b01	324.74	77.532	4.18
b02	177.66	48.272	3.6
b03	1103.81	324.912	3.39
b04	5639.94	1035.496	5.44
b05	5319.13	895.67	5.93
b06	316.42	101.47	3.11
b07	3961.80	653.68	6.06
b08	1035.34	273.27	3.7
b09	1261.05	318.44	3.96
b11	7056.84	890.76	7.7

Table 7.3: Overhead for different ITC'99 benchmark circuits

an increase in area with the increase in the size of the design. This is due to the fact that the routing for dummy is done for bigger design adds to the area overhead. In some case, we also notice a non-linear trend which is due to the

complexity of the design which might be using a higher number of nets. On an average across

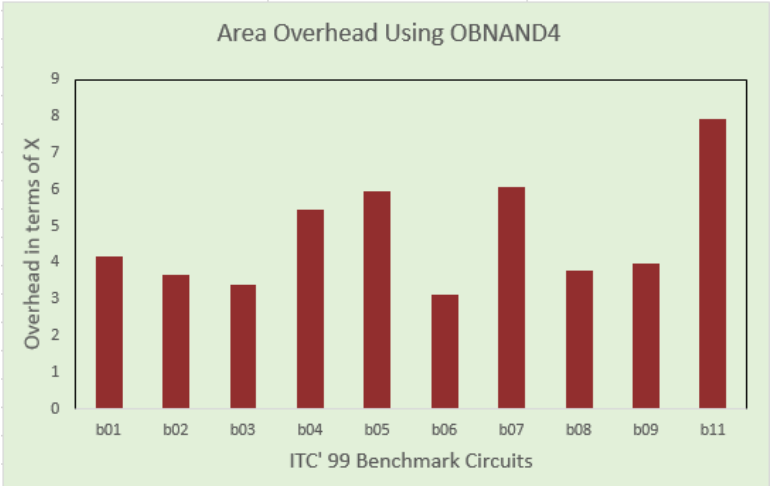


Figure 7.7: Area overhead for different ITC'99 benchmark circuits

different benchmarks the area overhead is in the range of 3.2x to 7.5x with an average of 5.87x. Hence, we gain an obfuscation with each gate had 160 possibilities with random nets selected as dummy inputs each time, which further adds to the complexity.

CHAPTER 8

CONCLUSION

In order to curb the problem of IP violation and protect cryptographic hardware implementations from reverse engineering a dopant based obfuscation system is developed. An Obfuscell(standard cell) based on switchable dopant configuration has been designed in an area efficient manner with a area almost equal to that of a single inverter ($0.798um^2$). The switchable dopant configuration enables the Obfuscell to function as inverter, buffer, always1 and always0 gate without making changes to the metal or polysilicon. What makes the use of switchable dopant interesting is it's ability to be detected is 16X [26] harder than metal or polysilicon.

Using the Obfuscell alongwith the 4-input logic cell NAND4 a prototype is designed, which can function as 26 unique gates and can be further extended to 160 permutated gates. The underlying layout for all these 160 gates remains the same and only thing that changes is the dopant. This confuses the attacker, as all the cells look alike but function differently. Previously proposed schemes, have only dealt with a library of 4-12 cells, but in this work we have been able to generate 160 different gates based on the protoype OBNAND4. The description of these gates is written in verilog using a automatic library generation tool. It's a flexible tool that can be used to generate libraries based on different logic cells. The OBNAND4 library is the best solution with the right balance of real and dummy inputs for the purpose of obfuscation.

A major focus of this thesis has been to successfully developing a layout-level obfuscation tool which converts a rtl-level netlist into highly secure and complex

obfuscated layout(mask) which is extremely hard to reverse engineer. For example, in a design of 500-700 gates as in case of PRESENT S/P layer the complexity goes up, but the functionality of the design remains same. In addition, to varying the dopant in the OBNAND4 prototype based library, the use of dummy wires makes it next to impossible to reverse engineer using brute force methods. Each implementation of the design under consideration will look different and basically resemble one big 'blob'. The main goal of the ObfusTool was to conduct this tranformation in an area and power efficient manner and try to reduce the die-size and cost of obfuscation. The area overhead involved has been considerably seen reduced across different library designs from 300-400% based on the number of gates used and the complexity of the logic. The power overhead is roughly 3x to 8x. All the designs were tested across 3 different libraries, making OBNAND4 the winner in obfuscation v/s area and power consumption trade-off. The delay across different design doesn't seem to greatly varies and is roughly 1.5x to 4x for different security applications. This was a later addition to the results, hence it is an approximate analysis based on the time at hand. Hence, at this cost of delay an higher level of obfuscation can be provided.

The different ITC'99 benchmarks of varied sizes were checked using the OBNAND4 library, showed a variation in the area overhead. Reasons for these variations are linked to the use of dummy wires which are randomly selected from all the intermediate nets in the design. On an average the overhead was 5x with a variation of 3x to 7.9x. The bigger designs showed a higher overhead due to the random connection of dummy inputs. Hence, requiring a large place and route area.

8.1 Future Research Directions

The ObfusTool has been designed and can be used for for future research in the field of hardware obfuscation. The tool can flexibly incorporate different design for Obfuscell and libraries. Since, this tool has been designed it's gives a start point to build upon and scope for improvement. The next section talks about the different areas that can be taken up for future research.

8.1.1 Smart wiring for design under consideration

The current wiring approach of the obfuscation tool selects the dummy wires randomly from a pool of intermediate nets. This costs a huge area overhead and scale with the size of the design. The obfuscation for a certain design goes up at the cost of area, power and to an extend delay. In order to make this approach more area efficient, the dummy wiring could be placed using a smart algorithm which take the proximity of the different nets into consideration. Constraints can be given to the tool to consider a certain distance from an OBNAND4 cell to route the dummy wires. This with greatly reduce the area, as net A would be restricted to net H and not connect at the way to net Z. Also, this transformation needs to be done in a way, so as to not compromise on the security of the overall design.

8.1.2 New design for Obfuscell

Currently, the designed Obfuscell has 4 different configurations that can be achieved by switching the dopant mask. Sugawara et al where able to reverse engineer a single dopant in the design of an inserted stealthy trojan [4] [26]. According to Sugawara et al the process of reverse engineer the dopant mask is 16x harder compare to the detection of polysilicon and metal. Since, in our approach the obfuscation cell is replicated in the entire design which would make the dopant

detection harder, if not impossible. The connectivity additionally add obfuscation and related overhead. Since, the dopant mask have been reversed, a newer design approach could be used for creating the Obfuscell.

In the Obfuscell contacts can be used to connect of disconnect the VDD and ground which would make the design more area efficient. To make it more secure a design is under progress which will use to back-to-back inverters to decide and always 1 or always 0 gate.

8.1.3 Attacker's Perspective: Side channel reverse engineering

Now, since this new scheme has been designed, we could use it evaluate the attacker's perpective. The future research is going to focus on using side channel information to reverse engineering the design under consideration. This would give us an estimate of the security and vulnerabilities of the designed approach. A better evaluation can be provided for the attack model for the obfuscated design.

BIBLIOGRAPHY

[1] Mehdi-Laurent Akkar and Christophe Giraud. An implementation of des and aes, secure against some attacks. In *Cryptographic Hardware and Embedded Systems—CHES 2001*, pages 309–318. Springer, 2001.

[2] C. Bao, D. Forte, and A. Srivastava. On application of one-class svm to reverse engineering-based hardware trojan detection. In *Quality Electronic Design (ISQED), 2014 15th International Symposium on*, pages 47–54. IEEE, 2014.

[3] Luis Basto. First results of itc'99 benchmark circuits. *IEEE Design & Test of Computers*, 17(3):54–59, 2000.

[4] G. T Becker, F. Regazzoni, C. Paar, and W. P Burleson. Stealthy dopant-level hardware trojans. In *Cryptographic Hardware and Embedded Systems-CHES 2013*, pages 197–214. Springer, 2013.

[5] Swarup Bhunia, Miron Abramovici, Dakshi Agrawal, Paul Bradley, Michael S Hsiao, Jim Plusquellic, and Mohammad Tehranipoor. Protection against hardware trojan attacks: Towards a comprehensive solution. *IEEE Design & Test*, 30(3):6–17, 2013.

[6] Andrey Bogdanov, Lars R Knudsen, Gregor Leander, Christof Paar, Axel Poschmann, Matthew JB Robshaw, Yannick Seurin, and Charlotte Vikkelsoe. *PRESENT: An ultra-lightweight block cipher*. Springer, 2007.

[7] SOC Cadence. Encounter user's manual.

[8] Rajat S. Chakraborty and S. Bhunia. Hardware protection and authentication through netlist level obfuscation. In *Proceedings of the 2008 IEEE/ACM International Conference on Computer-Aided Design*, pages 674–677. IEEE Press, 2008.

[9] Rajat S. Chakraborty and S. Bhunia. Harpoon: an obfuscation-based soc design methodology for hardware protection. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 28(10):1493–1502, 2009.

[10] Rajat S. Chakraborty and S. Bhunia. Rtl hardware ip protection using key-based control and data flow obfuscation. In *VLSI Design, 2010. VLSID'10. 23rd International Conference on*, pages 405–410. IEEE, 2010.

[11] Lap-Wai Chow, William M Clark Jr, and James P Baukus. Integrated circuit with reverse engineering protection, May 24 2005. US Patent 6,897,535.

[12] R. P. Cocchi, J. P. Baukus, L. W. Chow, and B. J Wang. Circuit camouflage integration for hardware ip protection. In *Proceedings of the The 51st Annual Design Automation Conference on Design Automation Conference*, pages 1–5. ACM, 2014.

[13] A. Cui, Chip H. Chang, and Sofiène Tahar. Ip watermarking using incremental technology mapping at logic synthesis level. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 27(9):1565–1570, 2008.

[14] Semantic Designs. Thicket family of source code obfuscators.

[15] Ujjwal Guin, Ke Huang, D DiMase, JM Carulli, M Tehranipoor, and Y Makris. Counterfeit integrated circuits: A rising threat in the global semiconductor supply chain. *Proceedings of the IEEE*, 102(8):1207–1228, 2014.

[16] A. Iqbal. Understanding ic security threats. 2013.

[17] Alfred J Menezes, Paul C Van Oorschot, and Scott A Vanstone. *Handbook of applied cryptography*. CRC press, 1996.

[18] Carla Meninsky. Locked out: The new hazards of reverse engineering. *J. Marshall J. Computer & Info. L.*, 21:591, 2002.

[19] G. Perera. *Purposefully manufactured vulnerabilities in US government technology microchips: risks and homeland security implications*. PhD thesis, Monterey, California. Naval Postgraduate School, 2012.

[20] Jeyavijayan Rajendran, Michael Sam, Ozgur Sinanoglu, and Ramesh Karri. Security analysis of integrated circuit camouflaging. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 709–720. ACM, 2013.

[21] Jeyavijayan Rajendran, Ozgur Sinanoglu, and Ramesh Karri. Vlsi testing based security metric for ic camouflaging. In *Test Conference (ITC), 2013 IEEE International*, pages 1–4. IEEE, 2013.

[22] Jeyavijayan Rajendran, Ozgur Sinanoglu, and Ramesh Karri. Regaining trust in vlsi design: Design-for-trust techniques. *Proceedings of the IEEE*, 102(8):1266–1282, 2014.

[23] SEMI. Intellectual property (ip) challenges and concerns of the semiconductor equipment and materials industry. *SEMI ® WHITE PAPER—S PONSORED AND FUNDED BY SEMI*.

[24] Mitsuru Shiozaki, Ryohei Hori, and Takeshi Fujino. Diffusion programmable device: The device to prevent reverse engineering. *IACR Cryptology ePrint Archive*, 2014:109, 2014.

[25] P. Subramanyan, N. Tsiskaridze, W. Li, Adria Gascón, W. Tan, A. Tiwari, N. Shankar, S. Seshia, and S. Malik. Reverse engineering digital circuits using structural and functional analyses. 2013.

[26] T. Sugawara, D. Suzuki, R. Fujii, S. Tawa, R. Hori, M. Shiozaki, and T. Fujino. Reversing stealthy dopant-level circuits. In *Cryptographic Hardware and Embedded Systems–CHES 2014*, pages 112–126. Springer, 2014.

[27] R. Torrance and D. James. The state-of-the-art in ic reverse engineering. In *Cryptographic Hardware and Embedded Systems-CHES 2009*, pages 363–381. Springer, 2009.

[28] Johannes Wolkerstorfer, Elisabeth Oswald, and Mario Lamberger. An asic implementation of the aes sboxes. In *Topics in Cryptology—CT-RSA 2002*, pages 67–78. Springer, 2002.