Standard Cell Camouflage Method to Counter Silicon Reverse Engineering

Hector Gomez, Ckristian Duran and Elkim Roa
Integrated Systems Research Group, Onchip - Universidad Industrial de Santander, UIS
Bucaramanga, Santander, Colombia
{hector.gomez,ckristian.duran}@correo.uis.edu.co, efroa@uis.edu.co

***Abstract*—Payment and identification IDs using smart cards technology are vulnerable to a physical attack by reverse engineering the netlist of an embedded integrated circuit. An attacker gains access to a netlist by de-packaging and delayering a chip and processing its micrograph images. We proposed a low performance impact and novel method to obfuscate gates by using a current digital design flow with a layout standard cell generator to obfuscate non-critical paths. Results indicate that at cell and block level, a 15% timing and power overhead is incurred in applied worst-performance benchmarks. After applying the method to a set of benchmark circuits, results shown a 40% average obfuscation, or not detected gates, of the resultant netlist.**

I. Introduction

Reverse engineering integrated circuits has become a profitable industry considering the easy access to imaging tools and the large volume market that this semiconductor area represents. An attacker inspects lower metal layers pictures of a chip to extract a design to get rewarded by reproducing it or by finding vulnerabilities to exploit it. Reproducing a design would decrease development costs and bring unfair pricing compared to original designed chips. An extracted design can also be applied to find security vulnerabilities in order to get access to restricted data.

A common methodology to extract a design, which has been proved wisely, starts by de-packaging and delayering physically the chip. The extraction of the necessary information is achieved by employing imaging tools and CAD tools [1]. These tools store chip images in a database to re-create the circuit netlist by identifying common patterns in a circuit layout. A designer who aims to protect an intellectual property (IP) faces a difficult task due to the physical access of the chip to the attacker.

Developing a successful strategy to protect an IP is a concern of the semiconductor industry. Hardware obfuscation [2] is an alternative with two main approaches to fulfill the necessary protection: structural obfuscation and physical obfuscation. The aim of an structural approach is to hide the true functionality of the device-under-attack through techniques such as randomized placement of additional logic elements, irregular routing or dummy wires. Structural obfuscation is also implemented by using key-gates [3]. The use of key-gates enables the possibility to create unique secret keys such that the logic functionality is wrong without the correct key. Although logic level obfuscation by using key-gates is defeated by using test patterns and observing the output to decipher the keys, its strength against attackers is larger than layout level camouflage.

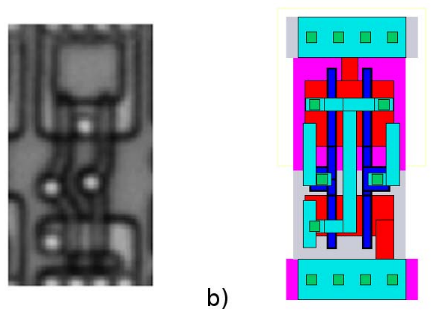


Fig. 1. a) A metal1 microphotgraph of a 2-input NAND from a 9-track standard library, b) and its layout.

Exploiting camouflaging at layout level have been reported by designing look alike gates by using metal and contact dummies as the traditional approach to physical obfuscation [4]. Using dummy structures incur in area and timing overheads that might impede the adoption of camouflaging in high performance applications. Other method of look alike cells, it is to exploit different voltage thresholds to define logic gates with identical layouts [5]. A prominent disadvantage is the impracticality of designing standard cells with multiple voltage threshold which translates to considerable larger areas.

This work reports a camouflage method based on the property that a gate can be drawn in multiple manners. Although the usual chosen layout is a drawing such that the geometric shapes form an optimal structure from the perspective of timing performance and routing accessibility, non-optimal layouts can be placed instead of the chosen regular layout within non-critical timing paths making harder for the attacker to extract a netlist based in well-known gate geometries. The proposed methodology is enabled by the current approach in high performance nodes to generate dynamically cells customized to a specific cell size in order to reduce overall area and power consumption.

II. Proposed camouflaging method

There are two digital flows used at the industry. The most common one is to use standard cell libraries with regular gate sizes. For instance, a regular standard library would have

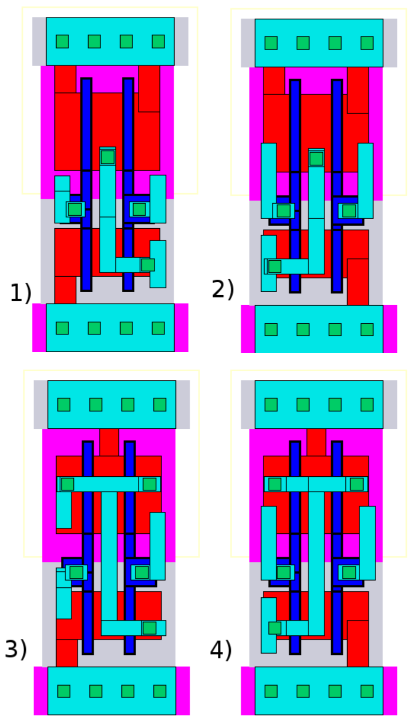


Fig. 2. Four different layout representations of a NAND2D1 layout.

inverters sized as multiples of an inverter capable to drive a load of fan-out of four usually called INVD1. Inverters are usually sized twice to eight times the size of an INVD1. These libraries are generated and tested by foundries and broadcasted to foundry users. The other flow, that it is uncommon to small and mid sized companies, is the flow with dynamic libraries. A library is populated dynamically during synthesis to reduce power and improve timing. This flow generates gates on-fly synthesis to improve timing and power performance. This flow requires standard cell layout generators which are usually restricted to companies focused on high volume consumer electronics silicon.

A flow capable to generate on-fly gates enables the generation of non-optimal cells with different placement and routing compared to an optimal cell which has been drawn to perform timing- and power-wise. By exploiting the capability to get additional geometries for the same cell, we have developed a method to obfuscate digital blocks by the use of multiple cells for the same function. An imaging tool, such as Chipworks ICWorks and Degate [2], trained with a database of regular optimal cells would not be able to identify gates that have a non-optimal geometry shape and therefore a full netlist cannot be extracted disrupting the attack.

Imaging tools are feeded with known gate geometries to train them to detect regular gates [1]. For instance, microphotgraphs of metal 1 and polysilicon layers of NAN2D1 cell are shown at Fig. 1a) with its representative layout shown at Fig. 1b). By training the imaging tool that NAND2D1 at metal 1 and at polysilicon layers looks like the one shown at Fig. 1a). Therefore, the tool would be capable to detect all the NAND2D1 cells from a complex layout by correlating the images from polysilicon and metal 1. However, the use of NAND2D1 gates with different geometry shapes at metal 1 and polysilicon would divert the imaging tool since usually an NAND2D1 has an optimal well known shape.

Different layout geometries can be requested to an automatic cell generation tool considering that the non-optimal cases are candidate results during the generation. For instance, Fig. 2 shows four different NAND2D1 flavors with placement shape in some of them quite far from the optimal placement of Fig. 2-1 but considering routing accessibility is not significantly impacted. Although in this work we are presenting a reduced number of gates, such as the inverter, NOR and NAND gates for one column size, we have demonstrated experimentally that these three cells are a representative sample of the large set of combination gates to validate the proposal.

TABLE I
Basic Gates Comparison for typical case characterization.

	INVD1			
	A	B	C	D
POWER	1	1.04	1.03	1.09
TIMING	1	1.01	1.01	1.04
	NAND2D1			
	A	B	C	D
POWER	1	1	1.15	1.16
TIMING	1	1	1.11	1.12
	NOR2D1			
SPEC	A	B	C	D
POWER	1	0.99	1.14	1.15
TIMING	1	0.99	1.10	1.11

TABLE II
Basic Gates Comparison characterized at low voltage supply, slow-slow process corner, and at 125C.

	INVD1			
	A	B	C	D
POWER	1	1.03	1.03	1.08
TIMING	1	1.02	1.02	1.05
	NAND2D1			
	A	B	C	D
POWER	1	1.005	1.15	1.16
TIMING	1	1.004	1.12	1.13
	NOR2D1			
	A	B	C	D
POWER	1	0.99	1.14	1.15
TIMING	1	0.99	1.12	1.12

Timing performance of non-optimal gate geometries based on the same sizing for a 9-track height cell can have a 13% overhead compared to the optimal cell as indicated in table I for the typical case and in table II for the worst case. Power performance can have an overhead of 16% in the case of the NAND2D1. Average performance indicates that a large timing penalty can be incurred if the non-optimal cells are placed inside a critical path which forces to limit the placement to non-critical paths.

The procedure to obfuscate a circuit including the proposed obfuscated cells is described in the pseudo-code shown in Algorithm 1. First, the circuit is synthesized using a standard

TABLE III
THERMOMETER DECODER DETECTION RESULTS

	Thermo 1		
	Detected	**Existent**	**Ratio (%)**
INVD1	16	28	57.14%
NAND2D1	4	9	44.44%
NOR2D1	7	7	100%
	Thermo 2		
	Detected	**Existent**	**Ratio (%)**
INVD1	16	28	57.14%
NAND2D1	4	9	44.44%
NOR2D1	4	7	57.14%
	Thermo 3		
	Detected	**Existent**	**Ratio (%)**
INVD1	15	28	53.57%
NAND2D1	7	9	77.78%
NOR2D1	3	7	42.86%

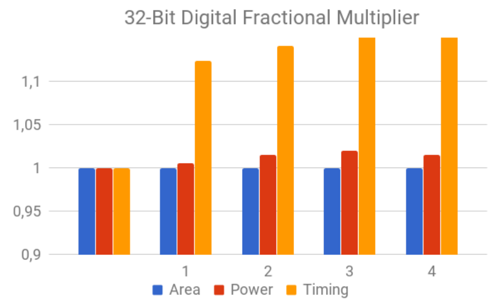
library to obtain an initial netlist and the corresponding critical path. Then, the obfuscated library is included and the circuit is synthesized again maintaining the critical path. If the new circuit presents the same critical path, a first obfuscation level is achieved. The algorithm use the new netlist to increase the obfuscation level and the critical path is analyzed again. If a change is found, the algorithm returns to the previous obfuscation level and perform a different obfuscation with the aim to maintain the required timing until the desired obfuscation level.

Algorithm 1 Obfuscation algorithm

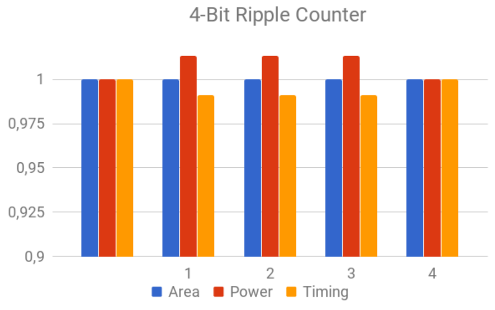
```
1: procedure StdCellOb($LOGIC$,$OBLevels$,$LibLevel$)
2:     $[netlist, critPath] \leftarrow RTLcompiler(LOGIC)$
3:     $save(netlist, "NORMAL");$
4:     for each integer $i$ in $OBLevels$ do
5:         $RTLconfig("InsertLib", LibLevels[i]);$    ▷ Config RTLcompiler for using OBS Lib
6:         $RTLconfig("PreservePath", critPath);$    ▷ Block path for using same cells
7:         $[netlist, newCritPath] \leftarrow RTLcompiler(LOGIC)$
8:         if newCritPath != critPath then
9:             $i \leftarrow i - 1$    ▷ Decreases Obfuscation Level
10:            $RTLconfig("InsertLib", LibLevels[i]);$    ▷ Reconfigure
11:            $[netlist, newCritPath] \leftarrow RTLcompiler(LOGIC)$
12:            if newCritPath != critPath then
13:                continue    ▷ Failed
14:            end if
15:        end if
16:        $name \leftarrow format("OBS[1]", i)$
17:        $save(netlist, name);$
18:    end for
19: end procedure
```

III. Results

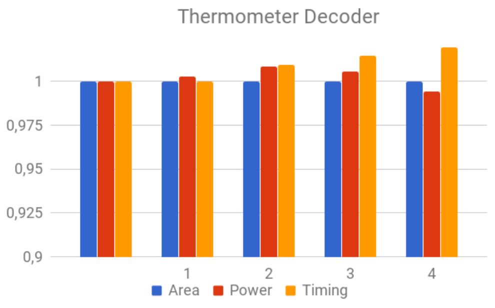
Camouflaging circuits using non-optimal layouts incur in some performance penalties if critical paths are modified.



(a)



(b)



(c)

Fig. 3. Performance summary of camouflaged circuits with the proposed technique.

Applying the design flow to different sequential and combinational circuits helps to figure out how much the circuits are impacted. First, three different circuits were synthesized with a classical standard cell library. Then, the gates with different geometries were included obtaining the power consumption, area and timing. Area is considering but as the camouflaged standard cells were designed to have the same size, results in this specification has not relevance.

Results of three synthesized circuits, a fractional multiplier, a ripple counter and a thermometer decoder from ISCAS benchmark [6], are shown in Fig. 3. Fig. 3a shows the performance of the synthesized multiplier where the camouflaged cells with worst timing were included. The most left bars indicate the results of the synthesized multiplier using the foundry provided standard library. In the first iteration (group of bars number 1), only one camouflaged cell is randomly included, alternating its use with the classical one. Timing penalty is evident due to the modification of critical path.

TABLE IV
4-BIT COUNTER DETECTION RESULTS

	4bitrip 1		
	Detected	**Existent**	**Ratio (%)**
INVD1	7	12	58.33%
NAND2D1	2	8	25.00%
NOR2D1	6	14	42.86%
DFD1	2	4	50.00%
	4bitrip 2		
	Detected	**Existent**	**Ratio (%)**
INVD1	5	12	41.67%
NAND2D1	3	8	37.50%
NOR2D1	11	14	78.57%
DFD1	1	4	25.00%
	4bitrip 3		
	Detected	**Existent**	**Ratio (%)**
INVD1	7	12	58.33%
NAND2D1	4	8	50.00%
NOR2D1	11	14	78.57%
DFD1	1	4	25.00%

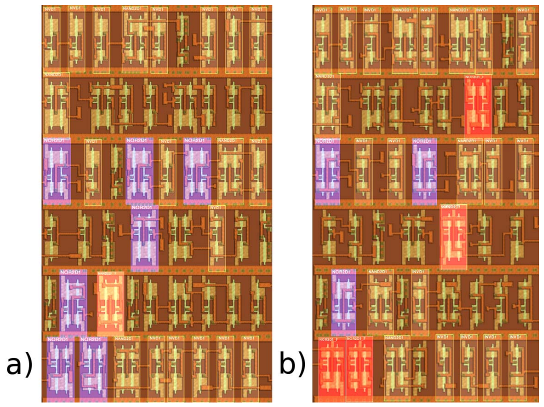


Fig. 4. Obfuscated layouts using camouflaged cells of a thermometer decoder. Detection software uses one type of NOR cell layout for circuit detection. Successfully detected cells are highlighted in blue and not detected ones are in red. a) Layout using one NOR cell, which is mostly detected. b) Layout using multiple camouflaged NOR cells with less than half detected cells.

Then, the circuit is synthesized increasing the amount of camouflaged cells up to 4 resulting in more delay penalty.

Fig. 3b shows the results for the ripple counter. In this circuit, the camouflaged cell is manually included resulting in an improved performance. The improvement indicates the camouflaged cell is better than the classical one. The different tests show the performance penalty can be avoided maintaining the size if critical path is not affected.

The test with the thermometer decoder is shown in Fig. 3c. It can be noticed that performance penalty can be avoided only in some cases. However, the performance overhead is worth it if the block can be protected.

As many imaging processing tools, the user need to provide an input pattern in order to train the extractor. However, if there is more than one pattern per gate function the attacker would get insufficient results to extract the full netlist. This increases the necessary resources to extract the complete netlist and in more complex chips, the difficultness of the extraction task for the attacker increase exponentially.

Different versions of place-and-routed circuits using an standard digital flow were analyzed with imaging tools. Different obfuscation levels were given to each version of circuits. Obfuscation level one uses two different layouts from the standard digital cell, level two uses two but at least with one different layout, and level 3 uses three different layouts. The existent cells correspond to total number of cells with the same logic function. The resulting layouts, after circuits were placed-and-routed, were the input for the imaging tool choosing only one layout as a known pattern, as an attacker usually did for cell detection. Table III shows the results for the different versions of the thermometer decoder extracting the total number of cells and the detected ones for every camouflaged cell. The detection ratio decreases every time a circuit is obfuscated at a higher level resulting in detection ratios lesser than 50%.

Table IV shows the corresponding results for the 4-bit counter used to prove the concept in a sequential circuit. Obfuscation process is still consistent and reduce the detection ratio to less than 50% in many cases.

Some pictures are added to express graphically the concept. In the layouts of Fig. 4a), only one different kind of NOR2D1 is used layout and the software is not able to identify all of actuals NORs in the circuit. A prominent result is obtained by increasing the amount of different layouts for a same NOR as shown in Fig. 4b). In this case, the imaging extractor identify only 3 NOR cells leaving 4 NOR cells unidentified proving that the method can obfuscate the hardware resulting in a detection ratio less than 50%.

IV. Summary

Obfuscation techniques always imply performance penalties in digital circuits. These penalties need to be taken into account in order to optimize the performance while the obfuscation level is maintained. Our proposed method induces a maximum performance penalty per cell of 16% and the obfuscated circuits present a performance overhead of 20% whereas the detection ratio is less than 50%. We proved that using the same function cell with different placements and routings will induce an obfuscation level enough to protect intellectual property blocks.

References

[1] K. Nohl and Starbug, “Silicon chips: No more secrets,” in *11th PACific SECurity annual conference (PACSEC)*, 2009, pp. 1–27.

[2] A. Vijayakumar, V. C. Patil, D. E. Holcomb, C. Paar, and S. Kundu, “Physical design obfuscation of hardware: A comprehensive investigation of device and logic-level techniques,” *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 1, pp. 64–77, Jan 2017.

[3] J. Rajendran, Y. Pino, O. Sinanoglu, and R. Karri, “Security analysis of logic obfuscation,” in *DAC Design Automation Conference 2012*, June 2012, pp. 83–89.

[4] J. Rajendran, O. Sinanoglu, and R. Karri, “VLSI testing based security metric for IC camouflaging,” in *2013 IEEE International Test Conference (ITC)*, Sept 2013, pp. 1–4.

[5] B. Erbagci, C. Erbagci, N. E. C. Akkaya, and K. Mai, “A secure camouflaged threshold voltage defined logic family,” in *2016 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, May 2016, pp. 229–235.

[6] F. Brglez, D. Bryan, and K. Kozminski, “Combinational profiles of sequential benchmark circuits,” in *IEEE International Symposium on Circuits and Systems,*, May 1989, pp. 1929–1934 vol.3.