Reducing Logic Encryption Overhead Through Gate Level Key Insertion

Kyle Juretus and Ioannis Savidis

Department of Electrical and Computer Engineering
Drexel University
Philadelphia, Pennsylvania 19104
{kjj39@drexel.edu, isavidis@coe.drexel.edu}

***Abstract*—Integrated circuits (ICs) are used in fields such as banking, transportation, energy, health, and the military. However, the use of ICs in many applications is threatened by an increasing reliance on untrusted third-parties within the IC design flow. The result is a growing concern for IC reliability and security, with threats that include IC counterfeiting, intellectual property (IP) theft, IC overproduction, and the insertion of hardware Trojans. An area of research aimed at ensuring the reliability and security of ICs in critical applications is logic encryption. While the security of an IC is increased when using logic encryption, current methods such as the XOR or look-up table (LUT) techniques have high per-gate overheads in area, performance, and power. A reduction in the per-gate overhead permits the use of logic encryption in a wider range of applications. Novel gate level designs for logic encryption are described in this paper, resulting in reduced overhead in power, area, and performance as compared to the XOR or LUT based techniques. With the proposed gate level technique, encrypting an AND gate results in a power reduction of 43.2%, an estimated area reduction of 19.8%, and a performance increase of 46.9% in comparison to the XOR based implementation of the encrypted AND.**

I. Introduction

The use of integrated circuits (ICs) in many applications that benefit society including banking, transportation, energy, health, and the military, continues to expand. While use in such fields present potential benefits, the increasing reliance on third-parties has introduced security and reliability concerns. One particular concern gaining significant traction is the increasing reliance on third-party foundries, as in-house fabrication facilities in advanced technologies cost in excess of $5 billion US dollars [1]. Since a third-party foundry has access to the IC design, typically in the GDS-II format, and has the required tools and knowledge to reverse engineer a design from the GDS-II file alone [2], utilizing an untrusted third-party foundry presents serious threats to an IC.

An untrusted third-party foundry is capable of intellectual property (IP) theft, IC counterfeiting, IC overproduction, and the insertion of malicious circuitry (hardware Trojans). Counterfeiting and piracy are expected to cause losses of $1.7 trillion dollars in 2015 [3], and a 2008 analysis conducted by SEMI estimated an IP revenue loss of $4 billion dollars to the IC industry alone [4]. While the monetary concerns of utilizing untrusted third-parties are significant, a greater danger is the use of ICs that do not meet the original specifications required for the target application. ICs that do not meet specifications cause increased failure rates and produce logical errors. In addition, there is a potential risk of malicious hardware Trojans being embedded within the ICs. Hardware Trojans inserted into the IC aim to deny service, steal information, and/or cause incorrect functionality [5]. Even with methods to detect hardware Trojans, there is the possibility that the Trojan executes without the knowledge of the end-user of the circuit.

An area of research that has emerged to reduce the security risks of IP theft, IC counterfeiting, IC overproduction, and hardware Trojan insertion is logic encryption[1]. Logic encryption adds additional circuitry (key gates) to a design in order to hide the functionality from an adversary. Without the application of the correct input key, the IC will not function correctly. Essentially, a malicious foundry no longer possesses all of the information to reverse engineer the entire design, making it more difficult to steal the IP, counterfeit the IC, produce extra ICs, and even insert hardware Trojans as the adversary is not certain of the effects the correct functionality of the IC has on the Trojan.

Current logic encryption typically uses XOR gates (or a gate of similar functionality) or replaces the original gate with a look-up table (LUT) utilizing a 4x1 MUX. While XOR or LUT based implementations offer a means to increase security, the per-gate area, power, and performance overhead is high. In order to utilize the security benefits of logic encryption in a wide array of IC applications, the per-gate overhead must be reduced. A novel gate level implementation of logic encryption is proposed in this paper that reduces the circuit overhead that affects the other logic encryption techniques.

The paper begins with an introduction to combinational logic encryption in Section II. Then a per-gate overhead analysis of current logic techniques is provided in Section III, followed by an explanation of the proposed gate topologies in Section IV. A comparison between the proposed topologies and current techniques is described in Section V. Finally, conclusions are offered in Section VI.

II. Combinational Logic Encryption

Combinational logic encryption alters the logic structure by requiring a key to enable the correct operation of the IC. The two prominent logic encryption techniques are: 1) the utilization of XOR/XNOR gates [6], [8], [9], and 2) the

[1] Researchers have previously referred to logic encryption as logic obfuscation [6], but [7] distinguishes between the two as logic encryption prevents black-box use of the design.

insertion of a LUT as a gate replacement [10]. A technique based on the insertion of 2x1 MUXes was also introduced in [9]. The inputs to the 2x1 MUX include the correct input and a net that carries the negated logical value of the correct input. The key input is the select signal of the 2x1 MUX. Finding a net that consistently negates the desired output is challenging and limits the use of the 2x1 MUX in logic encryption. An overview of the implementation of the XOR and LUT based logic encryption techniques is provided in this section.

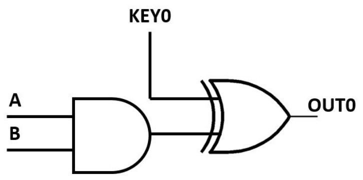

Fig. 1: Logic encryption with the use of an XOR gate.

A. XOR Based Encryption

An illustration of a very simple case of adding an XOR gate to implement a key input is shown in Fig. 1. When KEY0 is 1, the XOR gate behaves as an inverter and when KEY0 is 0, the XOR acts as a buffer. Therefore, when KEY0 is 1, an incorrect value is obtained on OUT0. To deter adversaries from knowing the key based on the implementation of the key gate (i.e. an XOR/XNOR gate), inverters are added in select locations, essentially inverting the key [6]. However, the additional inverters add overhead on top of the overhead already incurred to implement the key gate itself.

TABLE I: Analysis of the propagation delay, power, and area of standard cells from a 180 nm IBM technology.

Standard Cell	Prop. Delay (*ps*)	Power (*nW*)	Area (μm^2)
AND	69.79	70.40	30.73
NAND	36.71	72.67	23.25
OR	92.90	64.52	30.73
NOR	42.09	96.85	23.25
XOR	91.23	148.0	45.69
XNOR	106.7	204.6	41.62

B. LUT Based Encryption

The method proposed in [10] uses a look-up table to implement a key gate. A 4x1 MUX structure that is able to encrypt the functionality of a single gate is shown in Fig. 2. The key values are passed to the inputs of the 4x1 MUX, which sets the functionality of the gate. Using a 4x1 MUX structure allows for the realization of any 2-input function as opposed to the simple inversion provided by the XOR gate. Note that the illustration in Fig. 2 does not include any memory elements attached to the inputs of the 4x1 MUX, as memory is not explicitly required to encrypt the functionality of a gate. If memory is used in a design, the per-gate encryption overhead increases drastically.

An additional benefit of the technique presented in [10] is that the original gate is completely replaced, leaving very little information pertaining to the original functionality. The masked functionality of a gate encrypted with the LUT technique is dissimilar to the XOR based logic encryption method,

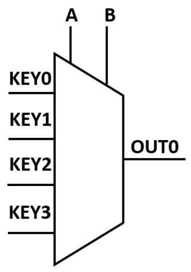

Fig. 2: Logic encryption with the use of a 4x1 MUX.

which requires additional inverters to mask the key value, as described in Section II-A.

TABLE II: Analysis of the propagation delay, power, and area of XOR based logic encryption. Per-gate overheads are provided as percent increases over the standard cell values listed in Table I.

Standard Cell	Prop. Delay (*ps*)	Power (*nW*)	Area (μm^2)
AND	151.3 (116.8%)	150.4 (113.6%)	63.73 (107.4%)
NAND	127.6 (247.6%)	142.2 (95.68%)	63.73 (174.1%)
OR	157.5 (69.54%)	174.6 (170.6%)	63.73 (107.4%)
NOR	134.3 (219.1%)	168.5 (73.98%)	63.73 (174.1%)
XOR	181.8 (99.27%)	219.3 (48.18%)	84.50 (84.94%)
XNOR	201.3 (88.66%)	226.3 (10.61%)	84.50 (101.4%)
Average	**140.2%**	**85.45%**	**124.9%**

III. Overhead of Logic Encryption

The XOR and LUT based logic encryption techniques both hinder the ability of an adversary to maliciously use an IC, but the per-gate overheads of both are high. An analysis of the performance, power, and area penalties associated with encrypting a standard logic gate for both methods is provided in this section. Note that for the cost analysis, no inverters were added after the XOR encryption gates, and no memory elements were used for the LUT based approach. The values provided for the per-gate overhead of each method are therefore highly optimistic as compared to implementations that include the inverters and memory.

TABLE III: Analysis of the propagation delay, power, and area of 4x1 MUX based logic encryption. Per-gate overheads are provided as percent increases over the standard cell values listed in Table I.

Standard Cell	Prop. Delay (*ps*)	Power (*nW*)	Area (μm^2)
AND	122.5 (75.53%)	146.0 (107.4%)	90.58 (194.8%)
NAND	124.6 (239.4%)	157.0 (116.0%)	90.58 (289.6%)
OR	120.8 (30.03%)	142.2 (120.4%)	90.58 (194.8%)
NOR	126.8 (201.3%)	158.8 (63.96%)	90.58 (289.6%)
XOR	124.0 (35.92%)	191.8 (29.59%)	90.58 (98.25%)
XNOR	124.6 (16.78%)	191.7 (6.310%)	90.58 (115.9%)
Average	**99.82%**	**73.95%**	**197.2%**

The IBM 180 nm process design kit was used for all of the results presented in this paper. A standard drive strength was used for each gate to test the overhead associated with each technique. The drive strength of the encrypted and original gate were matched, and each drove a load capacitance of 5fF. The per-gate overheads in power, performance, and area of the XOR based logic encryption technique are listed in Table II.

Similarly, the overheads of the LUT based logic encryption method are listed in Table III.

The large overheads in terms of power, area, and performance observed in Tables II and III limit the use of such techniques in many applications. The large overhead for per-gate encryption limits the acceptable signal paths to place gates without decreasing performance. In addition, the total number of encrypted gates placed in a circuit is limited by power and area constraints. It is therefore imperative to reduce the per-gate encryption overhead.

IV. Encrypted Gate Topologies

The motivation for gate level encryption stems from the inefficiencies in the circuits depicted in Figs. 1 and 2. Consider the circuit depicted in Fig. 1; to encrypt the AND gate, an additional XOR gate is added to the output of the AND. As described in Section III, the additional XOR gate increases the delay, area, and power consumption of the circuit. Similarly, logic encryption that uses 4x1 MUXes adds a large area overhead and creates additional levels of logic which decreases performance, as listed in Table III. Instead of inserting an extra gate in the logic path, as the XOR encryption method does, or utilizing a large area, as the 4x1 MUX technique does, novel gate level designs with embedded security capabilities are proposed to reduce the overhead associated with previous logic encryption implementations.

A. Stack-based Topology

The first topology is termed *stack-based*, and is shown in Fig. 3. The functionality depicted in Fig. 3 is dependent on the value of KEY0. When KEY0 is 0, the PMOS stack is activated allowing the gate to behave as a NAND. When KEY0 is set to 1, the NMOS stack is activated allowing the gate to behave as a NOR. The benefit of such a topology is that one of the stacks is essentially disconnected from OUT, limiting the capacitance connected to OUT during execution, which reduces power consumption and limits the degradation in performance.

Truth Table

KEY0	A	B	OUT
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

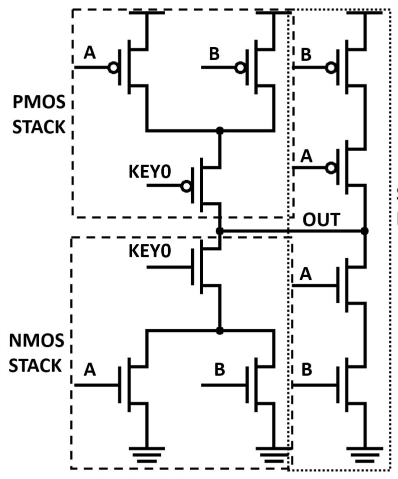

Fig. 3: Stack-based topology implementing a NAND or NOR gate depending on the value of KEY0.

An additional benefit of the stack configuration is the shared functionality between the implemented logic of the gates. For example, a NAND and NOR gate share the same output for two of the four input combinations of *A* and *B*. Having shared functionality eliminates the need for an additional transistor with a key input, such as when *A* and *B* are 0 in the box labeled *shared func.* (fine dashed) of the circuit shown in Fig. 3. Eliminating a key transistor for certain input combinations reduces the area and power consumption, as well as reducing the number of transistors in series in the stack, which further degrades performance. The NAND/NOR stack-based gate also has another important characteristic; the ability to forgo negated input logic. While the NAND/NOR configuration does not require negated logic, most designs implementing the stack-based topology do, as the ability to turn-on both PMOS and NMOS logic requires both logic high and logic low.

The stack-based topology is of most use when all the input combinations do not require the generation of an incorrect output when an incorrect key is applied. If, however, applying an incorrect key requires the generation of an incorrect output for all input combinations, such as the case when using the XOR based technique, the stack-based topology is not as beneficial as more power and area are required.

Truth Table

KEY0	KEY1	A	B	OUT
0	0	X	X	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	X	X	1

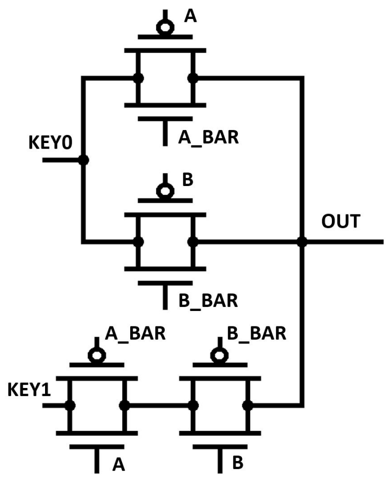

Fig. 4: Transmission gate based topology implementing an AND/NAND depending on the values of KEY0 and KEY1.

B. Transmission Gate Topology

The key value is passed through the transmission gates to reduce the impact on the area and performance of the replicated logic. Therefore, the total number of transistors and the transistors the key values pass through is reduced. With the key value passed through the transmission gates, the circuit shown in Fig. 4 is better suited to replicate the XOR based method. If logic low is applied to either input *A* or input *B* the value of KEY0 is passed to OUT. Similarly, when input *A* and input *B* are both logic high the value of KEY1 is passed to OUT.

A similar circuit is used to produce an OR/NOR gate. The only required modification is to invert the *A* and *B* inputs to the transmission gates. Such a modification passes the value of KEY0 when input *A* or *B* is logic high, and passes the value of KEY1 when inputs *A* and *B* are both low.

TABLE IV: Propagation delay, power, and area analysis of the AND/OR implemented with stack-based encryption. Percent improvements over XOR and LUT logic encryption are listed.

Comp.	Standard Cell	Prop. Delay (ps)	Power (nW)	Area (μm^2)
XOR	AND	119.8 (20.82%)	80.72 (46.33%)	41.62 (34.77%)
	OR	116.7 (25.90%)	79.87 (54.26%)	
LUT	AND	119.8 (2.204%)	80.72 (44.71%)	41.62 (54.05%)
	OR	116.7 (3.394%)	79.87 (43.83%)	

Note that two keys are not necessary, as KEY0 and KEY1 are simply the inversion of one another when implementing a AND/NAND gate or OR/NOR gate. However, using two distinct keys allows for a new concept proposed in the paper that is referred to as key expansion. Key expansion relies on the constant 0 and 1 functions that are generated from the transmission gate topology shown in Fig. 4 when KEY0 and KEY1 are both 0 or 1, respectively. Generating a constant 0 or 1 on the output line allows for the use of either value (0 or 1) as a key input while connecting inputs A and B to arbitrary nets within the circuit. An adversary now has a more difficult task to determine the key nets in a circuit, and must apply more key combinations to decipher the correct key.

The encryption of an XOR or XNOR gate is more complex. The logic of an XOR or XNOR gate does not include the same logic minimization that an AND, OR, NAND, or NOR gate possess. While AND, OR, NAND, and NOR gates all include simplifications that reduce the number of transistors required to implement the function, the only simplification that is possible for the XOR or XNOR gates is to eliminate one of the key inputs. The number of keys is therefore reduced for a single gate from four to three by tying KEY1 and KEY2 together to form a single key input, as shown in Fig. 2. Reducing the number of key bits required per gate is important if there is a strict area and power overhead requirement.

TABLE V: Propagation delay, power, and area analysis of the NAND/AND implemented with transmission gate encryption. Percent improvements over XOR and LUT logic encryption are listed.

Comp.	Standard Cell	Prop. Delay (ps)	Power (nW)	Area (μm^2)
XOR	AND	80.19 (46.99%)	85.41 (43.21%)	51.10 (19.81%)
	NAND	98.54 (22.77%)	101.0 (28.97%)	
LUT	AND	80.19 (34.54%)	85.41 (41.50%)	51.10 (43.58%)
	NAND	98.54 (20.91%)	101.0 (35.67%)	

V. Evaluation of Power, Area, and Performance

The improvements in power, propagation delay, and area for both the stack and transmission gate topologies are provided in this Section. As with the simulations described in Section III, the drive strengths of the encrypted cells were matched with those of the XOR and LUT based logic encryption techniques. In order to match the drive strength, an inverter was added to the output of both the stack and transmission gate topologies. The functionality of the stack-based design is therefore an AND/OR gate as opposed to a NAND/NOR. The improvements over the XOR and LUT based encryption techniques are listed in Tables IV, V, and VI. The improvement in performance is described as a reduction in propagation delay. The power is based on the average power of each gate. The area improvement is based on an area estimate determined from a non-optimized layout of the encryption gates as compared with layouts of the XOR and LUT encryption methods.

The results demonstrate the ability of gate level logic encryption to lower the overheads in area, power, and performance. The reductions in overhead provide more flexibility in the potential placement and quantity of encrypted gates within an IC design.

TABLE VI: Propagation delay, power, and area analysis of the NOR/OR implemented with transmission gate encryption. Percent improvements over XOR and LUT logic encryption are listed.

Comp.	Standard Cell	Prop. Delay (ps)	Power (nW)	Area (μm^2)
XOR	OR	86.99 (44.77%)	91.44 (47.63%)	51.10 (19.81%)
	NOR	96.77 (27.94%)	88.00 (47.78%)	
LUT	OR	86.99 (27.99%)	91.44 (35.70%)	51.10 (43.58%)
	NOR	96.77 (23.68%)	88.00 (44.58%)	

VI. Conclusions

Two different topologies for gate level logic encryption were described in the paper. By utilizing gate level logic encryption, substantial reductions in the per-gate encryption overhead were achieved. Encrypting an AND gate with the transmission gate topology results in a power reduction of 43.2%, an estimated area reduction of 19.8%, and a performance increase of 46.9% as compared to an XOR based implementation. Similar improvements are achieved with the stack-based topology, demonstrating that gate level logic encryption is an effective technique to mask circuit functionality while lowering the implementation cost of securing an IC from IP theft and attack.

References

[1] DigiTimes, "Trends in the global IC design service market," http://www.digitimes.com/news/a20120313RS400.html?chid=2, March 2012.

[2] R. Torrance and D. James, "The State-of-the-Art in Semiconductor Reverse Engineering," *Proceedings of the IEEE Design Automation Conference*, pp. 333 – 338, June 2011.

[3] International Chamber of Commerce, "Impacts of counterfeiting and piracy to reach US $1.7 trillion by 2015," http://www.iccwbo.org/News/Articles/2011/Impacts-of-counterfeiting-and-piracy-to-reach-US$1-7-trillion-by-2015/, Feb. 2011.

[4] Semiconductor Equipment and Materials Industry, "Intellectual Property (IP) Challenges and Concerns of the Semiconductor Equipment and Materials Industry," http://www.semi.org/cms/groups/public/documents/web_content/p043701.pdf, April 2008.

[5] M. Tehranipoor and F. Koushanfar, "A Survey of Hardware Trojan Taxonomy and Detection," *IEEE Design and Test of Computers*, Vol. 27, No. 1, pp. 10 – 25, Feb. 2010.

[6] J. Roy, F. Koushanfar, and I. Markov, "EPIC: Ending Piracy of Integrated Circuits," *Proceedings of the IEEE/ACM Design, Automation and Test in Europe*, pp. 1069 – 1074, Oct. 2008.

[7] J. Rajendran, O. Sinanoglu, and R. Karri, "Regaining Trust in VLSI Design: Design-for-Trust Techniques," *Proceedings of the IEEE*, Vol. 102, No. 8, pp. 1266 – 1282, July 2014.

[8] J. Rajendran, Y. Pino, O. Sinanoglu, and R. Karri, "Logic Encryption: A Fault Analysis Perspective," *Proceedings of the IEEE/ACM Design, Automation and Test in Europe*, pp. 953 – 958, Oct. 2012.

[9] J. Rajendran, H. Zhang, C. Zhang, G.S. Rose, Y. Pino, O. Sinanoglu, and R. Karri, "Fault Analysis-Based Logic Encryption," *IEEE Transactions on Computers*, Vol. 64, No. 2, pp. 410 – 424, Feb. 2015.

[10] A. Baumgarten, A. Tyagi, and J. Zambreno, "Preventing IC Piracy Using Reconfigurable Logic Barriers," *IEEE Design and Test of Computers*, Vol. 27, No. 1, pp. 66 – 75, Feb. 2010.