# Improving Integrated Circuit Security using Mathematical Model Based on Clique Covering Reformulation

Jonathan Fontaine, Mohamed Benazouz, Lilia Zaourar, Roselyne Chotin

# Improving Integrated Circuit Security using Mathematical Model Based on Clique Covering Reformulation

Jonathan Fontaine [1] and Mohamed Benazouz[1] and Lilia Zaourar[1] and Roselyne Chotin[2]

***Abstract*— Integrated Circuits (IC) are increasingly present in our daily lives through various everyday objects. Many third-party companies are involved during the manufacturing process. It introduces many threats to the ICs' manufacturing, such as IP piracy and Hardware Trojans. Strong Logic Locking methodology is generally used to protect from IC piracy, such as counterfeiting or reverse engineering, and against Hardware Trojans insertion; however, the lack of automated tools fully integrated into a CAD flow limits the integration of countermeasures. This paper proposes mathematical models on the Strong Logic Locking method to optimize the security and an automatic security design inserted in an open CAD flow.**

**We implemented an exact algorithm to maximize the security measure while implementing a strategy to minimize the impact of delay and area on the circuit. This algorithm is a custom branch and bound based on the mathematical model developed in this paper. In addition, we propose a strategy to limit the impact of countermeasures on the delay. Furthermore, our approach takes place inside a standard open CAD flow after logic synthesis to be as generic as possible.**

**The experiments carried out that security can be added in a standard open CAD flow with a reasonable computation time and a limited impact on the circuit. Our security measure is more precise than the previous one, with a limited area overhead defined by a user. The increase of the critical path is less than $7\%$ for large benches with a limit of $10\%$ area overhead.**

## I. INTRODUCTION

Integrated Circuits (IC) are increasingly present in our daily lives through various everyday objects (phones, health, games, cars, etc.). It leads to significant complexity in designing them and a challenge to ensure their safety and security. This increasing complexity and globalization have led third-party companies worldwide to specialize in carrying out different stages of the fabrication process, such as Intellectual Property (IP) vendors, design houses, and foundries (Fig.1). Despite the fact that these companies offer a much-needed economic advantage, there are less and less controls in the fabrication process. It introduces many threats in the ICs' manufacturing, such as IP piracy and Hardware Trojans (HT) [1], which must be considered when designing circuits.

Techniques used to counter these threats fall into one of these two categories: prevention and/or detection. We are interested in *logic locking* countermeasure [5] that provides security but requires advanced expertise to implement and might have non negligible impacts on the IC's cost and performance if not well contained. We aim to provide an automated and optimized tool based on *logic locking* that takes into account all the design constraints to be widely used in an open CAD flow. In that case, we need to conduct a study on the optimization problem of this method.

In this paper, we propose a reformulation of the problem to have a more precise security measure in section II-A. Then we propose two mathematical problems to describe it in section II-C - II-D. We developed a dedicated branch and bound algorithm to solve the second model in section IV. Finally, in section V, we conduct a study on ISCAS-85 benchmark.

### A. Logic Locking

There are several approaches to logic locking [3], which all rely on the same principle, obfuscating the circuit's logic by adding a digital key connected to additional logic gates. These additional logic gates are named key gates. Fig.2 is an example of logic locking (in red) where *xor* gates ($K_1$ to $K_7$) are inserted inside a circuit with their corresponding activation digital key bits ($k_1$ to $k_7$). Note that the digital key is stored in a tamper-proof memory that reverse engineering cannot read. Logic locking was first proposed to prevent counterfeiting, but also hardware Trojans insertion [4].

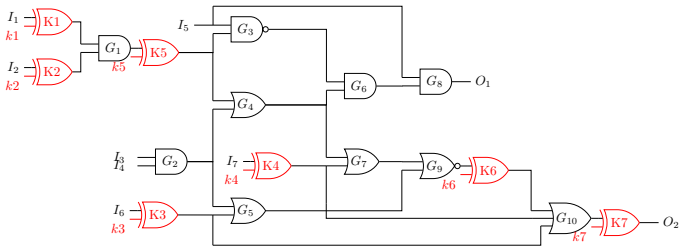


Fig. 2. Locked circuit.

Among many methods, we focus in this work on Strong Logic Locking (SLL) [6], [7] that proposes to measure the resistance to an attack that wants to recover the digital key by arguing that if the key is found, then the circuit is deobfuscated. Even if new methods have been proposed, some propose a hybrid method with SLL that combines the advantage of new methods with a limited impact on the circuit, such as [8].

### B. Strong Logic Locking

Strong Logic Locking [7] is a method that locks a combinational circuit by adding key gates linked to a digital key. It measures security using a binary relation P between pairs of key gates called *pairwise secure*. The authors of [7] define

This work is funded by the project ANR MOOSIC 18-CE39-0005.

[1] *Université Paris-Saclay, CEA, LIST* F-91120, Palaiseau, France jonathan.fontaine / lilia.zaourar / mohamed.benazouz / @cea.fr

[2] *Sorbonne Universite, CNRS, LIP6* F-75005 Paris, France roselyne.chotin@lip6.fr

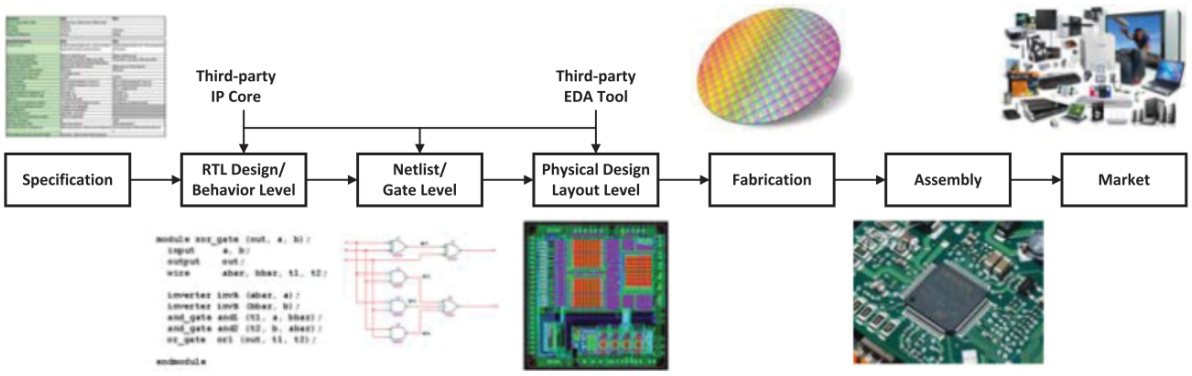


Fig. 1. Semiconductor supply chain [2].

two binary relations to construct the pairwise relation. The first one is the mutable relation that seeks to find an input vector (assignment value of each input signal of the circuit) of the combinational circuit to inhibit one key gate $A$ to observe the influence of another key gate $B$ on the output. In that case, we say that $B$ mutes $A$. The second one is the interference relation, defined by the impossibility of finding an input vector to mute a key gate $A$ with another key gate $B$. In such a case, $A$ is said to interfere with $B$. Finally, if a key gate $A$ interferes with a key gate $B$ and vice versa, $A$ and $B$ are in a pairwise relation. Pairwise relations are then used to construct a graph called an *interference graph* $G_{interference}(K, P)$ with $K$ the set of key gates and $P$ the set of pairwise relations (*see* Fig. 3).

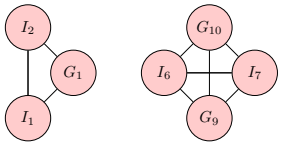


Fig. 3. Interference graph of circuit from Fig.2.

This interference graph is used to compute the security metric. Let $c$ be the size of the biggest clique (complete subgraph) in $G_{interference}$. The security metric is given by $2^c$. A clique $c$ from $G_{interference}$ corresponds to an irreducible part of the key that needs an exhaustive test to be recovered. This problem can be summarized as finding an interference graph that maximizes the maximum clique. The authors propose a greedy algorithm to solve this problem. This approach does not provide any guarantees of optimality. We then want to propose an exact solution to this problem.

## II. MATHEMATICAL DESCRIPTION

### A. Reformulation of Strong Logic Locking

SLL security is based on the largest *clique*. However, we claim that other *cliques* can significantly improve security. In fact, this security measurement may underestimate the quality of some solutions. For instance, in Fig.3, the security is $2^4$, but ignoring the other clique leads to overlook an additional $2^3$ security that is $50\%$ more. In order to take into account other objectives, such as delay, we need a fine-grained security estimation to avoid missing better solutions. We propose then to measure the security by $\sum_{c \in C} 2^{|c|}$ with $C$ a set of disjoint cliques from the interference graph described in [7]. The maximum clique problem is transformed into a weighted clique covering.

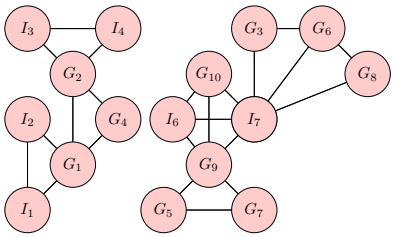


Fig. 4. Complete interference graph of the circuit (without locking) from Fig.2.

In order to solve this new problem, we compute the *Complete Interference Graph (CIG)*, i.e., we determine the *pairwise secure* relation between each possible insertion position, as in Fig.4. Then, we find the subgraph of size $K$ that maximizes security, with $K$ the number of key gates as in Fig.5. This number $K$ is a limit of area overhead defined by the designer. In our model, the impact on the area is a constraint. The vertices of the subgraph are the insertion positions. For instance, the optimal subgraph from Fig.5 states that optimal position insertion to maximize security in integrated circuit (without locking) from Fig.2 is in position $I_1$, $I_2$, $G_1$ $I_6$, $I_7$, $G_9$, $G_{10}$, that is the lock circuit from Fig.2.

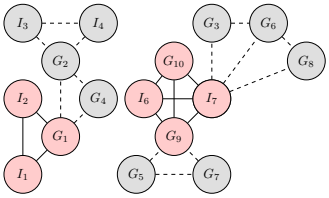


Fig. 5. Optimal Subgraph of CIG with 7 key gates from Fig.4.

### B. Complete Interference Graph

As depicted in Section I-B, the *pairwise secure* relation must satisfy all input patterns. As there is an exponential number of input patterns, it is impossible to check them all. The authors of [7] identified this problem and suggested checking a limited number of input patterns. However, some of them find more mutable relations than others. ATPG (Automatic Test Pattern Generation) is an algorithm that finds interesting input patterns to test the circuit. They are commonly used in the IC community for test and validation. These input patterns effectively find mutable relations as depicted in [7]. To build the *CIG*, each input pattern is tested to determine whether it is possible to mute a key gate with each position pair. Even if this step is time-consuming, it only needs to be done once per circuit.

### C. Mathematical Model

We propose the following mathematical model to represent our problem. Our input is a graph representation of the circuit called *netlist*, where vertices are logic gates, and arcs represent wires. Let $G = (V, E)$ be the circuit's netlist with $V$ the gates and $E$ the wires. Let $G' = (V, E')$ be the *CIG* of the circuit with $E'$ the pairwise secure relations. Let $K$ be the number of inserted key gates, determining the subgraph size. Let $\sigma \in \Sigma = \{1, \dots, K\}$ be a slot to represent a *clique*. Since there is at most $K$ disjoint *cliques*, $|\Sigma| = K$. Let $\lambda_\sigma \in \{0, 1\}$ equal to one iff the clique $\sigma$ is selected. Let $w_\sigma \in \{0, \dots, K\}$ be the size of the clique $\sigma$. Let $x_v \in \{0, 1\}$ be set to one iff vertex $v$ is selected in the subgraph and let $a_\sigma^v \in \{0, 1\}$ be set to one iff $v$ is in the clique $\sigma$.

Our objective is to maximize the security $\sum_{\sigma \in \Sigma} (2^{w_\sigma} \cdot \lambda_\sigma)$. With $\lambda$ multiplication, empty slots do not bias the objective. The knapsack constraint (1) enforces that the subgraph size is limited to $K$.

$$\sum_{v \in V} x_v \leq K \tag{1}$$

The clique constraint (2) requires that two vertices $(v, v')$ in the same slot must be adjacent in the *CIG*. $M[v, v']$ is equal to one if $v$ and $v'$ are adjacent, zero otherwise.

$$a_\sigma^v + a_\sigma^{v'} \leq 1 + M[v, v'] \qquad \forall (v, v') \in V^2,\ \forall \sigma \in \Sigma \tag{2}$$

The disjunction constraint (3) limits a vertex $v$ to be associated with only one slot.

$$\sum_{\sigma \in \Sigma} a_\sigma^v \leq x_v \qquad \forall v \in V \tag{3}$$

Constraint (4) links the size of a slot $\sigma$ with the number of vertices associated with that slot.

$$w_\sigma = \sum_{v \in V} a_\sigma^v \qquad \forall \sigma \in \Sigma \tag{4}$$

Selection constraint (5) pushes towards discarding empty slots whereas constraint (6) includes every slot $\sigma$ with at least one vertex in the computation of the security metric.

$$\lambda_\sigma \leq w_\sigma \qquad \forall \sigma \in \Sigma \tag{5}$$

$$\lambda_\sigma \geq a_\sigma^v \qquad \forall \sigma \in \Sigma,\ \forall v \in V \tag{6}$$

Constraints (1) - (6) form a first mathematical model ($P_1$) that represents our problem. It contains $\mathcal{O}(K \cdot |V|)$ variables and $\mathcal{O}(K \cdot |V|)$ constraints. However, solving this model is slow in practice due to the size of the exploration space. The following section proposes a bypass approach that reduces the exploration space for a more reasonable computing time.

### D. Maximal Cliques Approach

Since clique covering uses cliques, we compute the list of all maximal cliques (maximal for inclusion). This step takes an exponential time to solve ($\mathcal{O}(3^n)$) [10]. Nevertheless, CIG are sparse graphs, and there are efficient algorithms for listing all maximal cliques on sparse graphs [11].

Let $L$ be the list of all maximal cliques. Let $w_c = |c|$ be the size of the clique $c$. The decision variables of the model are as follows. Let $\lambda_c \in \{0, 1\}$, to be set to one iff the clique $c$ is selected. Let $x_v \in \{0, 1\}$ be set to one iff vertex $v$ is selected in the subgraph and finally, let $a_c^v \in \{0, 1\}$ be set to one iff $v$ is in the clique $c$.

As in model ($P_1$), the objective is to maximize the security $\sum_{c \in L} (2^{w_c} \cdot \lambda_c)$. The difference is that pre-computed cliques replace slots. The knapsack constraint (7) limits the size of the subgraph to a maximum of $K$.

$$\sum_{v \in V} x_v \leq K \tag{7}$$

The disjunction constraint (8) enforces a vertex $v$ to be associated with one clique $c$ at most.

$$\sum_{c \in L} a_c^v \leq x_v \qquad \forall v \in V \tag{8}$$

Constraint (9) links the size of a clique $c$ with the number of vertices associated with it.

$$w_c = \sum_{v \in c} a_c^v \qquad \forall c \in L \tag{9}$$

Selection constraints (10) and (11) enforce selecting cliques with at least one vertex while discarding empty ones.

$$\lambda_c \leq w_c \qquad \forall c \in L \tag{10}$$

$$\lambda_c \geq a_c^v \qquad \forall c \in L,\ \forall v \in c \tag{11}$$

Unlike the model (P1), there is no need for a clique constraint as this is provided by the pre-computed list $L$. Constraints (7) - (11) form a second mathematical model ($P_2$) for our security problem with $\mathcal{O}(|L| \cdot |V|)$ variables and $\mathcal{O}(|L| \cdot |V|)$ constraints.

## III. STATE OF THE ART

As introduced above, in a graph with $n$ vertices, there is an exponential number of maximal cliques: $\mathcal{O}(3^{\frac{n}{3}})$ [9]. Thus, enumerating these cliques might take exponential time. Many algorithms have been proposed in the literature for that. Bron and Kerbosh are the first to reach the theoretical bound and to prove it [10]. Nevertheless, we have sparse graphs in our case, and some authors propose a faster algorithm in such cases [11].

As described in section II-A, our problem can be seen as a *weighted clique covering* where the objective is to maximize

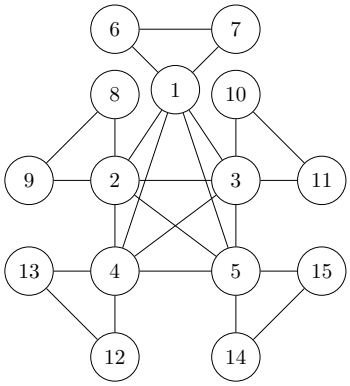


Fig. 6. Counterexample where WCC and CC optimal solution are different

the sum of the weight of each clique $c$ with $w(c) = 2^{|c|}$. The clique covering problem (CC) is one of the 21 Karp's NP-complete problems [12]. The CC problem consists of finding a partition of the graph into disjoint cliques. It can be transformed into the graph coloring (GC) problem by taking the complementary graph. This problem has been widely studied, and many authors have worked on its complexity [13]. It has been proved that the approximate chromatic number to within $n^{\epsilon}$, $\epsilon > 0$ is NP-HARD. Therefore, we know that approximating the optimal solution of the CC problem is also NP-HARD.

The *weighted clique covering* (WCC) problem is a close problem. Maximizing the sum of the clique weights tends to maximize the number of elements in each clique as much as possible. This leads to reducing the number of cliques, and we know it is NP-hard to approach. However, the optimal solution to the WCC may not be the optimal solution to the CC problem. In Fig. 6, the optimal solution to the CC problem is 5 cliques (outsides cliques) with a sum of weights of 40, and the optimal solution to the WCC is 6 cliques (the inner clique + 5× cliques of size 3) with a sum of weights of 52.

Finally, our problem is finding the subgraph of size $K < n$, with the best objective value for the WCC problem. This problem is at least as hard as the WCC.

## IV. RESOLUTION

Algorithm 1 is a Branch and Bound that explores all the possibilities of the $P2$ model. The first call of the algorithm must be $BranchAndBoundP2(L, K, emptyList(), 0)$. In line 2, $findNextClique$ consists of finding the index of the next clique in the list $L$ with at least one vertex that is not in the current solution. It avoids computing the same solution multiple times and ensures that all cliques are disjoint. In line 5, we branch for all parts of the maximal clique into $L[newIndex]$, even the empty set. The test at line 6 ensures that the knapsack constraint (7) is satisfied.

The list $L$ contains an exponential number of cliques: $|L| < 3^{\frac{n}{3}}$. In order to compute solutions, we can compute sub-optimal solutions using a subset of this list, repeatedly increasing until we obtain an optimal solution. Since large cliques have more impact, the subset is fulfilled by cliques in decreasing order according to their size. There are three stopping criteria, firstly, when the subset contains all the maximal cliques, secondly, when the objective reaches the upper bound and thirdly, when the smallest clique in the solution is larger than all the remaining cliques.

**Algorithm 1:** BranchAndBoundP2(L, K, current, index, lowerBound)

**Input:** L: List of maximal cliques
K: Int to limit the size of the subgraph
current: List of disjoint cliques
index: Int that represents the current position in the list L
lowerBound: Best solution found so far
**Result:** List of disjoint cliques that cover the subgraph and the objective

1 **if** *upperBound(current)* $\leq$ *lowerBound* **then**
  /* Bounded solution */
2 **end**
3 vertices $\leftarrow \bigcup_{c \in current} c$
4 newIndex $\leftarrow$ findNextClique(L, current, index)
5 solution $\leftarrow$ current
6 security $\leftarrow$ objective(solution)
7 **forall** $c \in \mathcal{P}($ *L[newIndex]* $\setminus$ *vertices* $)$ **do**
8  **if** $|current \cup c| \leq K$ **then**
9  lsol, lsec $\leftarrow$ BranchAndBoundP2(L, K, append(current, c), newIndex, lowerBound)
10  **if** *lsec* $>$ *security* **then**
11  solution $\leftarrow$ lsol
12  security $\leftarrow$ lsec
13  **if** *security* $>$ *lowerBound* **then**
14  lowerBound $\leftarrow$ security
15  **end**
16  **end**
17  **end**
18 **end**
19 **return** solution, security

So far, the model only takes security as an objective. Nevertheless, the subgraph size corresponds to the number of added key gates. It is, therefore, related to the overhead area and is limited by the constraint (7). In order to reduce the impact on the delay, we propose an additional strategy in which we prevent key gates insertion on critical paths. This can be achieved by removing the vertices of these critical paths from the CIG. This, as we will see, will not prevent the delay from increasing. In fact, the insertion of key gates may still transform some paths into critical ones with even higher lengths.

## V. IMPLEMENTATION AND NUMERICAL RESULT

We propose to add a security extension module to an open CAD flow, as shown in Fig.7. Yosys [14] performs the logical synthesis that transforms the RTL into a netlist. Then, our

TABLE I

EXECUTION TIME AND RESULT OF THE ALGORITHM 1.

| Circuit information | | | Pre-processing | | No delay strategy | | | Delay strategy | | |
|---|---|---|---|---|---|---|---|---|---|---|
| circuit | #nodes | #key gates | CIG time (s) | Cliques time (s) | Optimization time (s) | Security | Delay (%) | Optimization time (s) | Security | Delay (%) |
| c499 | 205 | 10 (5%) | 21.29 | 12.22 | 1.21 | $2^{10}$ | 12.26 | 0.67 | $2^{10}$ | 23.39 |
| | | 20 (10%) | | | 1.51 | $2^{20}$ | 23.39 | 1.11 | $2^{20}$ | 23.39 |
| | | 31 (15%) | | | 1.48 | $2^{31}$ | 23.39 | 0.89 | $2^{31}$ | 23.39 |
| c1355 | 205 | 10 (5%) | 14.89 | 7.46 | 0.36 | $2^{10}$ | 12.27 | 0.36 | $2^{10}$ | 22.25 |
| | | 20 (10%) | | | 0.45 | $2^{20}$ | 22.26 | 0.37 | $2^{20}$ | 22.25 |
| | | 31 (15%) | | | 0.44 | $2^{31}$ | 23.4 | 0.43 | $2^{31}$ | 34.53 |
| c3540 | 493 | 25 (5%) | 48.91 | 0.039 | 1.40 | $2^{25}$ | 3.76 | 1.39 | $2^{25}$ | 0.44 |
| | | 49 (10%) | | | 1.72 | $2^{29}+2^{10}+...$ | 4.46 | 1.50 | $2^{29}+2^{10}+...$ | 4.46 |
| | | 74 (15%) | | | 40.15 | $2^{29}+2^{10}+...$ | 12.88 | 42.86 | $2^{29}+2^{10}+...$ | 12.88 |
| c5315 | 752 | 38 (5%) | 807.62 | 0.27 | 1.70 | $2^{38}$ | 0.0 | 1.75 | $2^{38}$ | 0.0 |
| | | 75 (10%) | | | 2.083 | $2^{46}+2^{29}$ | 0.0 | 1.96 | $2^{46}+2^{29}$ | 0.0 |
| | | 113 (15%) | | | Timed Out | $2^{46}+2^{12}+...$ | 21.97 | Timed Out | $2^{46}+2^{12}+...$ | 21.97 |
| c6288 | 1480 | 74 (5%) | 498.28 | 0.25 | 7.96 | $2^{40}+2^{8}+...$ | 9.6 | 4.49 | $2^{39}+2^{7}+...$ | 6.79 |
| | | 148 (10%) | | | 16.30 | $2^{40}+2^{9}+...$ | 14.96 | 4.35 | $2^{39}+2^{9}$ | 7.86 |
| | | 222 (15%) | | | 5.49 | $2^{40}+2^{9}+...$ | 22.9 | Timed Out | $2^{39}+2^{9}+...$ | 11.02 |
| c7552 | 953 | 48 (5%) | 3010.72 | 0.59 | 2.32 | $2^{48}$ | 0.0 | 2.13 | $2^{48}$ | 0.0 |
| | | 95 (10%) | | | 3.39 | $2^{95}$ | 4.25 | 2.10 | $2^{95}$ | 3.26 |
| | | 143 (15%) | | | 3.15 | $2^{143}$ | 13.71 | 2.72 | $2^{143}$ | 11.38 |

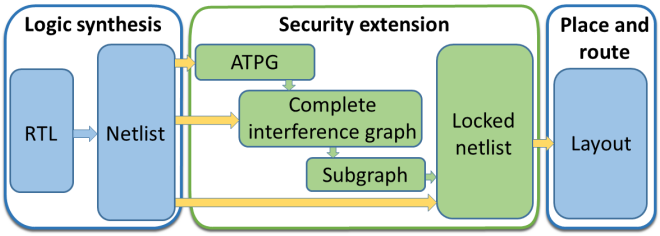


Fig. 7. Design flow with the security extension module.

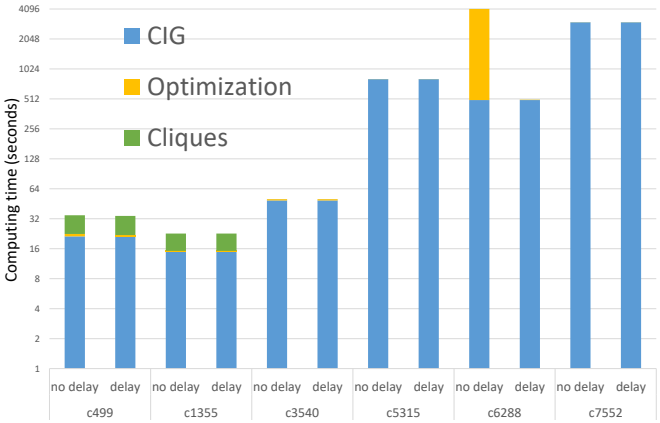


Fig. 8. Computation time of the extension module with an area overhead limit of 5%.

extension performs the automated and optimized insertion of key gates to maximize security while minimizing the delay described in section IV. Finally, Coriolis [15] performs the place and route.

The security extension is written in Julia 1.8. All calculations were performed on a server Scientific linux 7 64bits, intel 2x Xeon E5 – 2640$v4$ with: 20 cores, 40 threads, 128 GBytes of memory. The computation time was limited to 3600 seconds. The benchmarks used are ISCAS-85 [16]. We use a 350nm symbolic library named sxlib. Input patterns have been generated using Synopsys TetraMax ATPG tool [17].

As we are using 350nm technology, we can compute the sum of the gate delay in each path and thus get a close estimate of the delay. We count the number of logic gates added to estimate the area overhead. Every computation is done at the post-synthesis level, which allows us to be as generic as possible.

In this experiment, we wanted to carry out two points. The first point is the computation time of the pre-processing and the optimization algorithm. The second point is how increasing the limited area will affect the computation time and delay overhead. Table I summarizes these two points. It is separated into four parts. The first part summarizes the circuit information and the number of key gates to be added ($5\% - 15\%$ of the nodes). The second part is the pre-processing, which is the transformation of the netlist into a graph (*CIG*) used in the optimization algorithm and the computation for listing all maximal cliques. The third and fourth parts are respectively the results of Algorithm 1 with or without the proposed delay strategy.

This work aims to get the best security with a limited area overhead given by a designer as input. The limit of 5% area overhead is common in the literature. By increasing area overhead, we also want to show that our algorithm is still relevant.

Fig.8 shows that computing *CIG* takes most of the computing time. Although the algorithm to list all maximal cliques takes an exponential time in theory, it only requires few seconds of computation on sparse graphs in practice. Thus, our algorithm finds the optimal solution in few seconds. The computing time for circuit $c6288$ corresponds to a near-optimal solution, but the algorithm could not prove it is the best.

Fig.9 indicates that forbidding the critical path from increasing the delay overhead works in most cases. However, in some cases, such as circuit $c499$, the strategy has increased

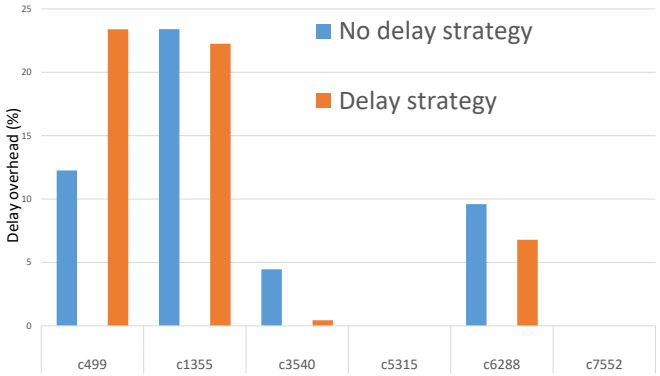


Fig. 9. Delay overhead of Algorithm 1 with 5% area overhead without and with the delay strategy.

the delay, with many key gates inserted in a near-critical path instead of a few on the critical path. As required, the delay overhead decreases with the size of the circuit. There are more insertion positions outsides the critical path. As expected, using this strategy does not increase the solution time. As it reduces the CIG, the problem becomes smaller.

Compared to [7] best algorithm (out of the 5 ones presented in their work) *SLJI*, globally, our technique is faster than their technique in terms of computing time but slower than the Time Improved (TI) version. Even if we did not implement their time improved technique based on an identified property of pairwise secure, we plan to implement this part to improve computation time of our *CIG* graph in future work.

We extracted from curves presented in [7] security, area as well as delay overhead values and reported them in Table II for benchmarks $c5315$ and $c7552$ and algorithms *SLJI*.

TABLE II
RESULT COMPARISON BETWEEN OUR ALGORITHM 1 AND [7] ALGORITHM.

| Circuit | Algorithm 1 | | | *SLJI* [7] | | |
|---|---|---|---|---|---|---|
| | security | delay (%) | area (%) | security | delay (%) | area (%) |
| c5315 | $2^{15}$ | 0 | 2 | $\approx 2^{16}$ | 0 | 2 |
| c7552 | $2^{200}$ | 0 | 21 | $\approx 2^{126}$ | 0 | 21 |

As depicted in Table II, we limit the area to their limit to compare security values. Our algorithm 1 finds a better security on bench $c7552$. However, our algorithm finds less security than their algorithm on bench $c5315$, even if we obtain our optimal security. 2% of area overhead allows us to add 15 key gates. The authors of *SLJI* got a better result because our circuit has fewer gates after logic synthesis. This difference comes from the limitation of area. Indeed, fixing a budget of 21% for area, our algorithm finds a security of $2^{200}$, with a delay overhead of 13.69%.

## VI. CONCLUSIONS

Due to the globalization and specialization of the IC supply chain, IP Piracy and HT insertion threats increased. It must be taken into account when designing circuits. In order to reduce this risk, we propose a security extension module integrated automatically into an open CAD flow to be widely used by designers.

This security module is based on an extension of the security measure proposed in SLL with a novel approach for its computation. We propose a mathematical formulation of the problem and different strategies to solve it. Our algorithm provides an optimal insertion position to maximize security with a limited area overhead defined by the designer. It contains the impact on delay to around 7% for large benches when considering a limit of 10% area overhead.

This security extension is providing great results on ISCAS 85 benchmarks, and using our proposed strategy to limit the impact of delay is effective. In future works, we plan to test our CAD flow with $ITC99$ and larger instances. In addition, we want to use an open-source ATPG algorithm to be fully open.

As depicted in section V, the delay strategy is efficient, but a multi-objective exploration approach could find better trade-offs between maximizing security and minimizing area and delay overhead.

In addition to this work, we are currently working on a proof for the approximation complexity of the WCC problem.

## REFERENCES

[1] Bloomberg Businessweek (2018). The big hack: How China used a tiny chip to infiltrate US companies. Bloomberg, New York, NY, USA.
[2] Xiao, K, et al. "Hardware trojans: Lessons learned after one decade of research." ACM Transactions on Design Automation of Electronic Systems (TODAES), 2016.
[3] Yasin, Muhammad, and Ozgur Sinanoglu. "Evolution of logic locking." International Conference on Very Large Scale Integration, 2017.
[4] S. Dupuis, et al. "A novel hardware logic encryption technique for thwarting illegal overproduction and hardware trojans" IOLTS, 2014.
[5] J. A. Roy, et al. "EPIC : ending piracy of integrated circuits," DATE, 2008.
[6] J. Rajendran, et al. "Security Analysis of Logic Obfuscation," Design Automation Conference (DAC), 2012.
[7] M. Yasin, et al. "On improving the security of logic locking." Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2015.
[8] B. Shakya, et al. "Cas-lock: A security-corruptibility trade-off resilient logic locking scheme." IACR Transactions on Cryptographic Hardware and Embedded Systems 2020.
[9] Moon, J. W., and L. Moser. "On cliques in graphs" Israel journal of Mathematics 3 (1965).
[10] Bron, C., Kerbosch, J. "Algorithm 457: finding all cliques of an undirected graph" Communications of the ACM, 1973.
[11] D. Eppstein, et al. "Listing all maximal cliques in sparse graphs in near-optimal time." International Symposium on Algorithms and Computation, 2010.
[12] R. M. Karp, "Reducibility among combinatorial problems," in Complexity of computer computations, 1972.
[13] Pardalos Panos M., Thelma Mavridou, and Jue Xue. "The graph coloring problem: A bibliographic survey." Handbook of Combinatorial Optimization, 1998.
[14] Claire Wolf. Yosys Open SYnthesis Suite. https://yosyshq.net/yosys/
[15] Alexandre, Christophe, et al. "Tsunami: An integrated timing-driven place and route research platform." DATE, IEEE, 2005.
[16] Brglez F, Fujiwara H. "Special Session on ATPG (Also introducing'A Neutral Netlist of 10 Combinational Benchmark Circuits')." InInt. Symp. On Circuits and Systems 1985.
[17] TetraMAX ATPG User Guide. Synopsys Inc., 2005.