IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version February 10, 2021.

Digital Object Identifier

Logic Encryption for Resource Constrained Designs

DAVID M. LURIA[1], RANGA VEMURI[2]
[1]EECS, University of Cincinnati, Cincinnati, OH 45220 (e-mail: dluria13@gmail.com)
[2]EECS, University of Cincinnati, Cincinnati, OH 45220 (e-mail: vemurir@ucmail.uc.edu)

Corresponding author: David M. Luria (e-mail: dluria13@gmail.com).

DISTRIBUTION STATEMENT A. Approved for public release: distribution unlimited. This Research was Sponsored by the Air Force Research Laboratory Under Prime Contract W911QY-17- C-0114.

ABSTRACT Logic Encryption is a hardware security technique that protects integrated circuit designs that are fabricated at untrusted pure play foundries from being pirated or maliciously modified. In the technique, logic gates are added to the design that are driven by an added key input bus, such that the correct behavior of the circuit is recovered with only the exact correct key input pattern. However, the power, performance, and area (PPA) cost of implementing logic encryption has often been ignored in the literature in favor of increasing the level of security provided. This has proved to be a significant hurdle in transitioning the method to use in commercial-grade designs and a systematic methodology of constraining the cost of logic encryption is needed. In this paper, we propose a generalized Constraint-Directed Logic Encryption (CDLE) methodology. In CDLE, the potential design space of encrypted versions of a circuit is searched to apply logic encryption under PPA constraints. Two example CDLE methods are proposed. The first is a concurrent tree search method which uses commercial tools to sample designs for their PPA cost and determine the optimal encryption strategy. In this method, PPA cost is accurately analyzed at the cost of heavy runtime. The second is a machine learning approach which estimates the PPA cost to predict the optimal encryption strategy. The machine learning model developed in this work is limited, but the results are promising as a direction for study in logic encryption. Detailed experimental results evaluating both methods are presented.

INDEX TERMS Hardware security, integrated circuits, logic encryption, cost metrics, logic synthesis, machine learning.

I. INTRODUCTION

THE rising cost of manufacturing integrated circuits (ICs) has lead many circuit design houses to outsource their fabrication by contracting third party pure play foundries to reduce production costs [1], [2]. However, this has come at the cost of reduced trust in the IC supply chain. The globalization of IC production has opened an opportunity for bad actors at untrusted foundries to maliciously mishandle, steal, or modify the intellectual property (IP) of circuit designers. Some popular foundries are located in countries with weak IP protection policies and enforcement, compounding the issue [2]. This has lead to an estimated annual cost of over $100 billion to the semiconductor industry in piracy and damages [3], [4]. Ways IP can be stolen or modified include IC overproduction [5], reverse engineering [6], hardware Trojan insertion [5], and IC counterfeiting [7].

A. LOGIC ENCRYPTION

Roy, Koushanfar, and Markov [2] proposed a new hardware security technique to prevent the piracy of IC designs known as logic encryption (or logic obfuscation, or logic locking). In their method, a circuit is modified by the insertion of key gates, which are driven by an added input bus, called the key inputs. The key gates are added such that, for at least one input pattern on the key inputs, the original output behavior of the circuit can be recovered (effectively making the key gates dummy gates). However, under any other key input pattern, the output behavior of the circuit is corrupted for one or more primary (non-key) input pattern(s).

The circuit is rendered effectively useless to an attacker unless the correct key pattern, known only to the circuit designer, is asserted on the key input bus. Once the locked circuit is returned to the designer after fabrication, it can be activated before being sold on the market by loading the correct key into non-volatile memory to drive the key inputs.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3059163, IEEE Access

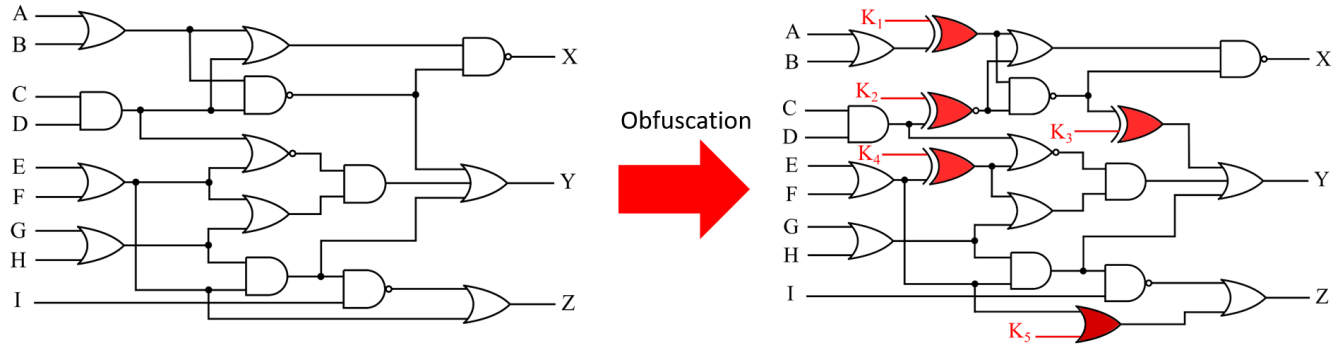


FIGURE 1. Example of logic encryption of a netlist by inserting XOR, XNOR, and OR gates with correct key K=01000.

If an attacker at an untrusted foundry cannot uncover the correct key, then the design is effectively useless and cannot be stolen or mishandled. Therefore, the circuit designer is in control of their design throughout the entire supply chain through logic encryption.

B. CONSTRAINT-DIRECTED LOGIC ENCRYPTION

Logic encryption is an evolving field of research in hardware security. Since Roy et al. proposed the technique, several attacks to uncover the correct key input have been proposed. Numerous techniques have been developed in response to bolster the level of security provided by logic encryption and increase the attack complexity against a locked circuit. However, the cost of improving the level of security provided by logic encryption is not well understood, especially in terms of power, performance, and area (PPA) [1]. A few logic encryption methods individually attempt to constrain area [8] or performance [2] impact, but many methods do not consider cost metrics at all. A cost evaluation of state of the art logic encryption methods is lacking in the field, which is a major hurdle preventing the technology from being implemented in more than a couple commercial designs. Furthermore, a formal, systematic approach to efficiently implementing it is sorely needed to transition the technology to large-scale commercial designs, and only exists in a limited form [9].

We propose a Constraint-Directed Logic Encryption (CDLE) methodology as a systematic approach to efficiently implementing logic encryption in a circuit. The methodology is driven by both security and cost metrics. In CDLE, potential logic encryption implementations are considered, each with an associated PPA cost and a level of security provided by encryption. The most optimal encrypted design is chosen based on user-set cost and security constraints (defined in the design specification). Therefore, designs produced using the methodology are optimized for both security and cost.

CDLE is not a single method; it is a family of methods. In addition to the methodology, we have developed two example methods that follow the CDLE framework. The first is a tree-based search. Multiple threads are harnessed to concurrently estimate the PPA of potential designs using commercial off-the-shelf EDA tools. By sampling the potential encrypted design space, the method converges on the optimal encrypted design. The second is a machine learning approach. In this method, a machine learning model is trained using PPA cost data collected from a set of training designs. Then, the optimal encryption strategy can be predicted based on the training results.

This paper is organized as follows. In Section II, a brief background of logic encryption is given. In Section III, an overview of CDLE is presented as a generalized strategy for implementing logic encryption while responding to design constraints. In Section IV, the CDLE concurrent tree search method is presented along with, in Sections V and VI, the results of using the method to apply logic encryption to a set of commercial-grade test designs. In Section VII, we show a CDLE machine learning method and compare the results of experimentation with those of the concurrent tree search. In Section VIII, potential research directions are suggested based on the overall methodology, the results of each example method, and the comparison between them.

II. BACKGROUND

Logic encryption is a hardware security technique first introduced by Roy, Koushanfar, and Markov in their 2010 paper, Ending Piracy of Integrated Circuits (EPIC) [2]. Their security strategy was devised to combat IP theft. The locking mechanism proposed by Roy, et al. involved the addition of new primary inputs, called key inputs, to the circuit design. If the correct key is applied to these inputs, the output behavior of the circuit will be equivalent to the design specification. Under application of an incorrect key, the output behavior of one or more primary (non-key) input patterns is corrupted. This prevents theft of circuit IP by rendering the design unusable unless the correct key is known, which is only known to the circuit designer. The designer can then activate each chip using the key, stored in secure non-volatile memory, before selling them on the market.

Early logic encryption methods aimed for enough output corruption to make an incorrect key obvious during testing, but not enough to easily eliminate incorrect keys. Thus, 50% hamming distance from normal output behavior under application of an incorrect key was the primary goal [1]. Attacks

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3059163, IEEE Access

TABLE 1. Examples of logic encryption strategies

Encryption Schemes		
SAT-Vulnerable	**SAT-Resilient**	**Sequential**
EPIC [2]	SARLock [10]	Encrypt Flip-Flop [11]
SLL [12]	Anti-SAT [13]	ChainLock [14]
Key Interdependency [15]	TTLock [16]	StateLock [17]
Hardware Enlightening [18]	SFLL [8]	
	Cyclic Modification [19]	
	SRCLock [20]	

against logic encryption were soon devised, necessitating novel and specific logic encryption strategies to responded to the attacks [21], [22].

Attacks can be classified into two types: oracle-guided and oracle-less. An oracle is a working copy of the circuit with which the correct input-output behavior of the circuit can be observed without access to the netlist. This can come in the form of an IC obtained from the market, or a black box simulation of the design. See Table 2 for examples of attacks against logic encryption. One of the most potent attacks proposed among these was an oracle-guided attack proposed by Subramanyan, Ray, and Malik called the satisfiability attack, or SAT attack [23]. This attack leveraged a Boolean satisfiability engine to eliminate entire classes of keys at a time by only observing input-output behavior of the locked circuit and the oracle. This was a vast improvement over other attacks of the time which relied on brute force and/or structural analysis. Unlike other attacks, SAT was guaranteed to find the exact correct key without any brute force required, making it particularly potent.

After its proposal, the focus of the field shifted to SAT. Many logic encryption strategies were devised to specifically thwart SAT [10], [19], [31], new attacks and improvements to SAT were created to combat these methods specifically [24], [26]–[28], and even stronger encryption methods [8], [16], [20] attempted to prevent ever-strengthening attacks. See Table 1 for examples of SAT-vulnerable and SAT-resilient logic encryption strategies. The race of strengthening logic encryption versus increasingly potent attacks has lead to improved security metrics of logic encryption methods being the focus of the field, with cost metrics more or less ignored. However, considering cost metrics is an important step to transitioning this technology to the commercial realm [1], [9].

TABLE 2. Examples of Attacks against logic encryption

Attacks	
Oracle-Guided	**Oracle-Less**
Key Sensitization [21]	Removal: SPS [24]
SAT [23]	FALL [25]
AppSAT [26]	
CycSAT [27]	
Removal: AGR [24]	
Sequential SAT [14]	
BeSAT [28]	
SMT [29]	
KC2 [30]	

For most of the past decade, logic encryption methods and attacks were mostly limited to combinational logic. Sequential behavior was not considered for encryption strategies and attacks generally relied on flip-flops being a part of controllable and observable scan chains [1], [23]. However, recent research has involved developing encryption methods [11], [17], [32] that lock sequential logic and state machines, and attacks that target sequential behavior specifically, including an improvement to SAT that converts sequential logic to a combinational equivalent through an unrolling process [14], [30]. See Table 1 for examples of logic encryption strategies that specifically modify sequential logic .

A. METRICS

Two types of metrics can be considered when applying logic encryption to protect a design [1]. One set of metrics involve the strength of security provided by the applied encryption. Examples of these metrics are:

- **Output Corruption** – The degree to which the IC output is affected by incorrect keys. This is typically quantified by the Hamming Distance (HD) between the correct output behavior and the output behavior under each correct key. A basic ideal value is 50%, large enough such that the application of an incorrect key is noticeable in post-silicon validation testing, but small enough to ensure difficulty in cracking the key.
- **Key Size** – The number of key inputs pins that are added to the design. In general, the larger the key size, the more difficult the encryption is to crack.
- **Attack Resilience** – The known attack strategies against which the encryption strategy is resilient. This depends on the particular key gate placement strategy used.

The second set of metrics involve the cost of applying logic encryption. These are PPA (power, performance, and area) metrics. The addition of key gates have an impact on these metrics for any design:

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3059163, IEEE Access

- **Power Usage** – Additional logic gates consume additional power when operating the IC. This comes in two forms, static power from leakage current, and dynamic power from gate output switching activity.
- **Performance Degradation** – The speed at which the IC can be operated. Despite attempts to lessen the critical timing of a design, adding logic encryption can affect the critical path or introduce a new critical path. If this results in negative timing slack, then the IC must be operated at a lower clock speed with the added encryption.
- **Area Footprint** – The additional area on the die required for the IC with added encryption. Adding key gates increases the logic gate count, which may require additional area if the die area needs to be increased to place and route the additional logic.

Novel logic encryption methods of the past decade often respond directly to security metrics, especially attack resilience. Known attacks present a challenge to the effectiveness of logic encryption and responding to these attacks is an important design consideration. Some encryption methods consider cost metrics, especially area. However, cost metrics are often ignored in favor of improving security metrics.

III. CONSTRAINT-DIRECTED LOGIC ENCRYPTION

The proposed constraint-directed logic encryption (CDLE) methodology balances cost and security to find an optimal encryption strategy. Consider a design C to be encrypted with a desired security goal, S_{opt}, which can include, for example, attacks to protect against and an allowed key size range. Assume that the design specification of C includes a set, PPA_{inc}, which represents the allowed increase in power, performance, and area after encryption. There is a design space of possible designs, D_C, containing all possible encrypted versions of C using known encryption methods. An encrypted version of the design $C_E^i \in D_C$ has its own encryption parameters which include, but is not limited to, an obfuscation scheme and key size, and has its own security and cost metrics. There is a subset of designs that meet S_{opt}, $D_C^S \subseteq D_C$, and another subset of designs that have PPA costs less than PPA_{inc}, $D_C^{ppa} \subseteq D_C$. Consider the set $D_C^{opt} = D_C^S \cap D_C^{ppa}$, which represents the set of designs in D_C that meet the security goal while also staying within the PPA constraints of the design specification. If $D_C^{opt} \neq \emptyset$, then $\exists\ C_E^{opt} \in D_C^{opt}$, the encrypted version of C within D_C^{opt} which maximizes all security metrics.

The goal of CDLE is, first, to identify the subset of encrypted designs D_C^{opt}, and then find the optimized design C_E^{opt} within that set which maximizes each security metric. Therefore, the methodology can be seen as a set of functions $F_i(C, S_{opt}, PPA_{inc})$ that have the design C, security goal S_{opt}, and the maximum allowed PPA increase PPA_{inc} as input, and output an optimally encrypted design C_E^{opt}. Theoretically, there is an exact correct C_E^{opt} such that increasing the security metrics, even slightly, exits the optimized design space D_C^{opt}.

Ideally, all algorithms in the methodology would return the same C_E^{opt}. However, different methods will estimate PPA and security metrics for each design in D_C differently, and may produce a variety of estimations of C_E^{opt} as well. Additionally, methods will have a variety of expected execution times, which can depend on the method itself, as well as design-specific parameters such as post-synthesis gate count. Methods in the methodology should aim to return an accurate C_E^{opt}, but should also have a reasonable execution time, based on the needs of the circuit designer. In general, using more accurate PPA and security estimation methods that produce a more accurate C_E^{opt} could result in longer execution time, creating a trade-off between these two properties of the CDLE methodology. However, this trade-off does not necessarily hold. As methods of metrics estimation are explored, some methods could prove to be a direct improvement to others. Our work begins to explore the methodology with this in mind.

A. EXAMPLE: BINARY SEARCH-BASED ALGORITHM

The constraint-directed logic encryption method of Luria et al. [9], though basic, does fit into the scheme of the methodology proposed in this work. Therefore, it can serve as an introductory example for the methodology. In this design-space search algorithm, the set of encrypted designs that meet the security goal, D_C^S, was determined by grading the available encryption methods in terms of attack resilience. In this method, only SAT-resilient encryption schemes were included in D_C^S, using randomly generated correct key patterns for all encrypted designs. Only combinational benchmarks were tested, so sequential locking methods were ignored. The quality of security provided by a design was based on the encryption scheme grade, and the key size. Encryption schemes used were (in order of decreasing security grade):

1) SFLL-HD
2) SRCLock
3) Cyclic
4) TTLock
5) Anti-SAT
6) SARLock

The search algorithm proposed by Luria, et al. [9] is shown in Algorithm 1. First, the PPA of the original design is measured. Then, for each encryption scheme in the ranking, the design is encrypted at the minimum allowed key size, k_{min}. If $C_E^{k_{min}}$ does not meet the PPA constraint, then the algorithm moves to the next encryption scheme in the rating. If it does meet the constraint, then the design is encrypted at the maximum allowed key size, k_{max}. If $C_E^{k_{max}}$ meets the PPA constraint, then the design is immediately accepted as the optimal encrypted design, C_E^{opt}. Otherwise, the optimally secure design has a key size within k_{min} and k_{max}. The space of potential key sizes between the maximum and minimum key size is searched by binary search, encrypting the design and measuring the PPA of C_E^i at each searched key size. The algorithm terminates when the optimal key size is found, such that increasing the key size by 1 breaks PPA_{max}. If all

Algorithm 1 Constraint-Directed Logic Encryption using Binary Search

Require: Original Circuit C; Key width bounds k_{min}, k_{max}; Cost constraints PPA_{max}; Ranked encryption methods E
Ensure: Encrypted circuit C_E^{opt}

$PPA_o \leftarrow get\_cost(C)$
for $e \in E$ **do**
  $C_e^{k_{min}} \leftarrow encrypt(C, e, k_{min})$
  $PPA_e^{k_{min}} \leftarrow get\_cost(C_e^{k_{min}})$
  if $PPA_e^{k_{min}} \leq PPA_{max}$ **then**
    $C_e^{k_{max}} \leftarrow encrypt(C, e, k_{max})$
    $PPA_e^{k_{max}} \leftarrow get\_cost(C_e^{k_{max}})$
    if $PPA_e^{k_{max}} \leq PPA_{max}$ **then**
      return $C_e^{k_{max}}$
    end if
    $k_{left} = k_{min}, k_{right} = k_{max}$
    while $k_{left} \leq k_{right}$ **do**
      $k_{mid} = floor((k_{left} + k_{right})/2)$
      $C_e^{k_{mid}} \leftarrow encrypt(C, e, k_{mid})$
      $PPA_e^{k_{mid}} \leftarrow get\_cost(C_e^{k_{mid}})$
      if $PPA_e^{k_{mid}} \leq PPA_{max}$ **then**
        $k_{left} = k_{mid}$
      else
        $k_{right} = k_{mid} - 1$
      end if
    end while
    return $C_e^{k_{left}}$
  end if
end for
return $failure$

encryption schemes are tried without finding C_E^{opt}, then the algorithm fails ($D_C^{opt} = \emptyset$).

The authors used the following strategy for constraining PPA for the original design and each encrypted version:

- **Power** – This was constrained as a percentage of the power usage of the original design. Both switching and leakage power were considered in estimations.
- **Performance** – The maximum delay overhead on the critical path was determined by the allowed slack. Any encrypted design should have a slack greater than or equal to 0 on the critical path.
- **Area** – The sum total combinational area and interconnect area was used as the area calculation. A set percentage increase from the original design area was the constraint for encrypted designs (similar to the power constraint).

For this method, the PPA of each design was estimated using both Synopsys Design Compiler (DC) [33] and Synopsys IC Compiler (ICC) [34]. For each of these, the design was simulated using Icarus Verilog [35] to collect switching information to inform and increase the accuracy of DC and ICC PPA estimations. This collection represents the get_cost function of Algorithm 1. For simulation, a testbench of 10000 total test patterns was generated once per design, and used for all encrypted versions of the design. Synopsys Tetramax [36] automated test pattern generator was utilized to stimulate critical paths in the design, and then the rest of the test patterns were randomly generated.

The algorithm had three main steps, the first of which was estimating the PPA of the original design which involves DC execution time, t_{DC}, and ICC execution time, t_{ICC}. It was found that $t_{ICC} >> t_{DC}$, so PPA estimation had an approximate execution time of t_{ICC}. In the next step, PPA are measured at k_{min} for potentially every encryption scheme in the set of encryption schemes. Therefore, step two had a worst-case execution time of $N_E t_{ICC}$ for N_E encryption schemes tried. Then, in step three, the best key size is found by binary search. This had a worst-case execution time of $log_2(k_{max} - k_{min})t_{ICC}$ to run PPA collection at every step of the binary search. As a whole, the algorithm had a worst-case execution time of $t_{exec} = t_{ICC}(1 + N_E + log_2(k_{max} - k_{min}))$, assuming t_{ICC} is approximately equal for all designs.

This constraint-directed logic encryption method was only tested for small, combinational benchmarks with less than 4,000 logic gates. Execution time was not reported in their work. However, place and route is a complicated problem, and increases in complexity with design gate count. The complexity of PPA estimation and execution time limited the data set in this CDLE method. Furthermore, the algorithm's time complexity is directly related to t_{ICC}, resulting in a very heavy-handed, though accurate, solution to the constraint-directed logic encryption methodology. In the next section, improvements to this algorithm will be explored to 1) extend this method to sequential designs, 2) drastically decrease execution time, and 3) improve its applicability to a set of larger, more realistic designs.

IV. CDLE CONCURRENT TREE SEARCH

In the binary search constraint-directed logic encryption implementation, PPA measurements are carried out for one design at a time. Therefore, for every design that needs to be produced and have PPA estimated to converge on C_E^{opt}, the overall execution time of that CDLE algorithm is increased by the execution time of IC Compiler place and route, t_{ICC}. In the worst case, the number of times ICC was executed to find the best encryption scheme among N_E schemes is $O(N_E)$, and $O(log(k_{range}))$ times to find the best key size among the range of acceptable key sizes, k_{range}. Using ICC to estimate PPA is very accurate, but also very time consuming. Therefore, effectively reducing the number of times this estimation needs to be performed to arrive at C_E^{opt} is one strategy for decreasing the overall execution time of the algorithm.

One property of the PPA estimation of different encrypted designs in D_C is the *parallelization* property. At each step of the binary search algorithm of CDLE, an encrypted version of the original design, $C_E^i \in D_C$ is produced and PPA is

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3059163, IEEE Access

estimated using Design Compiler and IC Compiler. However, this process is completely independent of doing the same process for a separate encrypted design, C_E^j. Therefore, the PPA of C_E^i and C_E^j can be estimated simultaneously. Due to this property of this PPA collection strategy, the CDLE search method can be done in parallel for different designs. The parallelization property of PPA collection will be the basis upon which the Concurrent Tree Search solution to CDLE is built. Therefore, it can be seen as a direct improvement to the CDLE binary search algorithm. This modification to the algorithm will theoretically return the same C_E^{opt}, but in faster execution time than the binary search algorithm.

A. PPA ESTIMATION

The PPA estimation strategy of the concurrent tree search is almost identical to that of the binary search algorithm, except executed in parallel. In this strategy, there are two versions of the PPA estimation flow: one for the original design, C, and one for encrypted versions that are explored, C_E^i. Each of these PPA estimation flows involves a series of steps to inform and carry out the estimation. This involves interfacing several external commercial off-the-shelf (COTS) electronic design automation (EDA) tools. We have used Synopsys Design Compiler (DC) [33] for logic synthesis, Synopsys IC Compiler II (ICC) [34] for place and route synthesis, Synopsys Tetramax [36] for automated test pattern generation (ATPG), and Icarus Verilog [35] for netlist-level simulation. PPA estimations are based on the following factors:

- **Power** – Power is reported with both cell switching and leakage power. Switching power is determined by annotating cells with user-provided switching information, and leakage power is determined by the cell library.
- **Performance** – The critical timing of the design determines its performance. The maximum delay with primary inputs and flip-flops serving as start points and primary outputs and flip-flops serving as end points is measured as the critical path delay. The allowed performance increase is used to determine the slack on this path.
- **Area** – The area is measured as the total footprint of the core area, including logic cells, net connections, I/O, and filler area in the design.

Each PPA estimation flow is divided into two sub-flows. The first is estimation using Design Compiler, in which the design is simulated for switching information, and then PPA estimated for the netlist after logic synthesis. The second is estimation using IC Compiler, in which the design is simulated again, using the synthesized netlist, and then the IC layout is placed and routed. Then, PPA is estimated more accurately using the produced layout.

1) Pre-Encryption Design

The PPA of C is estimated, first using DC, with the following steps:

1) **Preliminary Logic Synthesis** – The design is synthesized to produce a test strategy for ATPG, and a technology-mapped netlist for simulation.
2) **Test Pattern Generation** – Tetramax fast sequential ATPG is used to produce test patterns that stimulate critical paths of the design. Additional random test patterns are generated to produce a total of 5000 patterns. This was lowered from 10000 patterns used in the binary search method to accommodate larger designs.
3) **Simulation** – The design is simulated using the test patterns generated in the previous step. This produces switching information for nets in the design, which will improve the accuracy of the power estimation.
4) **DC PPA Estimation** – PPA is estimated using logic synthesis. After synthesis is complete, PPA is reported by Design Compiler using the identified metrics.

After a fully synthesized and technology-mapped netlist is produced, the PPA is estimated more accurately using ICC:

1) **Simulation** – The design is simulated again using the same test patterns as the DC estimation.
2) **ICC PPA Estimation** – PPA is estimated using an IC layout. After place and route are complete, PPA is reported by IC Compiler using the same identified metrics.

2) Encrypted Designs

The PPA estimation flow of all C_E^i is similar to that of C, but with certain modifications. One of the main modifications is the removal of test pattern generation. Since under application of the correct key $C_E^i = C$, the same test pattern strategy from the PPA estimation of C can be used for C_E^i, with the addition of the correct key pattern being constantly applied to set of key inputs. Therefore, the test pattern generation step can be skipped for all C_E^i, and the testbench of the simulation step can reuse the test patterns used for C, with the correct key being constantly applied throughout simulation. Therefore, the steps to estimate PPA with DC are:

1) **Encryption** – C is encrypted using the encryption parameters corresponding to the target C_E^i, including encryption scheme and key size. The design is encrypted with a randomly generated correct key pattern, unique to the specific C_E^i being produced.
2) **Preliminary Logic Synthesis** – C_E^i is synthesized to produce a technology-mapped netlist for simulation.
3) **Simulation** – The encrypted design is simulated using the same test patterns as C. This produces switching information for nets in the design, which will improve the accuracy of the power estimation.
4) **DC PPA Estimation** – PPA is estimated using logic synthesis. After synthesis is complete, PPA is reported by Design Compiler using the identified metrics.

After PPA is estimated using DC, the results are compared with the ICC PPA results of the original design. If the PPA of the encrypted design meet the allowed PPA cost set by the designer, then the design is accepted as a part of D_C^{opt},

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3059163, IEEE Access

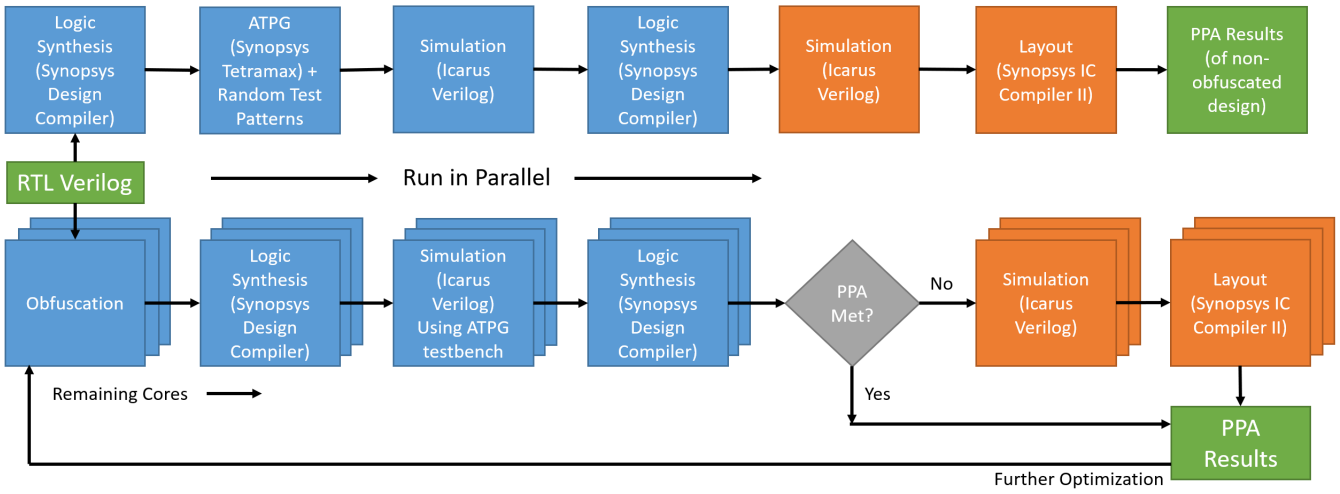


FIGURE 2. Parallel PPA collection strategy

and PPA estimation is concluded for this C_E^i. If the PPA constraints are not met, then PPA estimation continues to ICC estimation for more accurate results:

1) **Simulation** – The design is simulated again using the same test patterns as the DC estimation.
2) **ICC PPA Estimation** – PPA is estimated using an IC layout. After place and route are complete, PPA is reported by IC Compiler using the same identified metrics.

Figure 2 shows the combined PPA estimation strategy for the concurrent tree search CDLE method. The number of PPA estimations for encrypted designs that can be run concurrently is dependent on the number of available threads to execute the estimation flows. One additional piece of the strategy to improve the parallel execution of the method is concurrent execution of PPA estimations for C and the first set of encrypted designs that are explored. Rather than waiting for PPA estimations of C to finish before beginning parallel execution of encrypted designs, the PPA estimation of the C can be added to the first parallel exploration of encrypted designs. This is feasible with the addition of a few locks and considerations:

1) Simulation of any design cannot begin without test patterns. Therefore, the test pattern generation step of the PPA flow of C must complete before any simulation is done for any designs. This can cause execution of PPA estimations of C_E^i to pause until test patterns are available.
2) For any DC PPA estimations of C_E^i, if the post-ICC PPA estimation of C is not completed, then ICC PPA estimation of C_E^i will be executed without checking if the DC PPA meets the constraints.
3) For any ICC PPA estimations of C_E^i, if the post-ICC PPA estimation of C is not completed, then the process must wait until the post-ICC PPA estimation of C is completed before deciding if the PPA constraints have been violated.

This PPA estimation strategy will be used in the concurrent tree method of CDLE as an attempt to improve execution time over the binary search method, while still maintaining the same level of accuracy. This can be seen as an improved version of the *get_cost* function from Algorithm 1 that can handle several designs at once, rather than one. Since PPA estimation is very parallelizable, especially for any C_E^i that are being explored, this should introduce a speedup directly related to the number of parallel computation threads that are available.

B. SECURITY EVALUATION OF ENCRYPTION SCHEMES

Each logic encryption schemes has attack resistance, often by design, that has been experimentally shown by the authors of the scheme. Refer to Tables 3 and 4 for a summary of this theoretical attack resilience per encryption scheme. SMT was excluded from the attacks in this table because behavioral locking mechanisms are outside the scope of this work, and otherwise SMT [29] behaves as SAT does. This table will be the basis with which to measure attack resilience of a particular encryption strategy. The attack resilience will be assumed to be the sum of the attacks against which each encryption scheme applied to each design protects. Therefore, several compound encryption schemes can be formulated to create a security strategy that protects against specific attacks.

For each design, a ranking of possible compound encryption strategies based on their proven resilience to known attack methods is used to quantify the attack resilience security metric. First, encryption methods were chosen based on their resilience to SAT-based and removal attacks:

1) SFLL
2) SRCLock
3) Cyclic Modification
4) SARLock
5) Anti-SAT

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3059163, IEEE Access

TABLE 3. Resiliency table for combinational encryption methods. ✗ denotes resilience, ≈ denotes resilience based on sub-method

	Sensitization	SAT	AppSAT	CycSAT	Removal	FALL	BeSAT
EPIC					✗	✗	
SLL	✗				✗	✗	
Key Int.	✗				✗	✗	
SARLock	✗	✗		✗		✗	✗
AntiSAT	✗	✗		✗		✗	✗
TTLock	✗	✗		✗	✗		✗
SFLL	✗	✗	✗	✗	✗	≈	✗
Cyclic	✗	✗	✗		✗	✗	
SRCLock	✗	✗	✗	✗	✗	✗	

TABLE 4. Resiliency Table for sequential encryption methods. ✗ denotes resilience

	Sequential SAT	KC2
EFF		
ChainLock	✗	
StateLock	✗	✗

Then, methods were ranked in order of resilience to sequential attack methods. StateLock was excluded to reduce encryption implementation overhead:

1) ChainLock
2) Encrypt Flip-Flop

These SAT-resilient and sequential encryption methods were combined to produce the final ranking:

1) ChainLock + SFLL
2) EFF + SFLL
3) ChainLock + SRCLock
4) EFF + SRCLock
5) ChainLock + Cyclic
6) EFF + Cyclic
7) ChainLock + SARLock
8) EFF + SARLock
9) ChainLock + Anti-SAT
10) EFF + Anti-SAT

This set of compound encryption schemes forms the set of acceptable encryption schemes implemented in D_C^S, the set of encrypted versions of the design C which fulfill the security goal, S_{opt}. The rating will serve to find the maximum level of security within D_C^{opt}, the set of designs that meet both the PPA constraint and the security goal. Designs within D_C^{opt} which maximize the encryption scheme rating will maximize the number of attacks against which the design is resilient, maximizing the overall level of security. Then, the key size can be maximized to further maximize security. These are the two security metrics upon which the concurrent tree search-based CDLE method bases D_C^S, and C_E^{opt}.

Authors of several SAT-resilient encryption schemes recommend pairing a SAT-resilient encryption scheme with a SAT-susceptible scheme to ensure an adequate amount of output corruption from the obfuscation strategy, as several SAT-resilient schemes have low corruption on incorrect keys. Randomly placed key gates (as in EPIC [2]) was chosen as the SAT-susceptible scheme in this method due to its straightforward and quick implementation, and since no additional attack resilience is required from the added scheme. For each compound encryption scheme listed, 45% of key inputs are dedicated to each listed obfuscation method, and 10% are dedicated to randomly placed key gate encryption.

C. METHOD ALGORITHM

The algorithm for the concurrent tree search CDLE method is a modified version of the binary search-based method. Refer to Algorithm 2. The original circuit C, minimum and maximum integer key sizes allowed k_{min} and k_{max}, cost constraints PPA_{max}, ranked encryption methods E, and a new input, the integer number of concurrent computation threads T ($T \geq 1$) are provided. In this algorithm, the function $encrypt$ produces a set of encrypted designs of C, rather than one design as in the binary search-based algorithm, using a set of encryption schemes at one key size, or one encryption scheme at a set of key sizes. Also, get_cost becomes get_costs, which returns sets of PPA values for a set of designs, rather than one design. The encrypted design with maximum security within D_C^{opt} is ensured upon successful execution of the algorithm. The algorithm may not succeed if the PPA constraint is too tight for the set of encryption methods and key size range used, in which case $D_C^{opt} = \emptyset$.

This algorithm has the same three basic steps as the binary search-based algorithm. First, the PPA of C is estimated. Concurrently, $T - 1$ encrypted designs at key size k_{min} are produced, with encryption schemes starting at the top of the ranking. The PPA of these designs is estimated using Design Compiler and IC Compiler, and checked for breaking PPA_{max}. The design with the highest ranked encryption scheme that does not break PPA_{max} is used for the next step. If all of the encrypted designs break PPA_{max}, then the next T encryption schemes are attempted at key size k_{min}. If all possible encryption schemes break PPA_{max} at k_{min}, then the algorithm returns a failure, because no design in D_C^S meets the PPA constraints.

If one or more encryption schemes are found to be within PPA_{max}, then the strongest in the ranking among them, E_{max} is chosen for the final step, the concurrent tree search of the key space of potential key sizes. In the first loop of the search, the range of key sizes between and including k_{min} and k_{max} is divided into T sections. The set of key sizes, $kvals$ at the end of each division, including k_{max} but not k_{min}, are selected for encryption. C is encrypted using E_{max} and the selected key sizes, producing T encrypted designs, for which the PPA of each design is estimated. For

Algorithm 2 Constraint-Directed Logic Encryption using Concurrent Tree Search

Require: Original Circuit C; Key width bounds k_{min}, k_{max}; Cost constraints PPA_{max}; Ranked encryption methods E; Computation threads T

Ensure: Encrypted circuit C_E^{opt}

$E_+ \leftarrow null \cup E$ // Encrypting with $null$ does nothing
for $i = 0, ... \frac{|E_+|}{T}$ **do**
  do in parallel
    $D_C^{k_{min}} \leftarrow encrypt(C, \{E_i, E_{i+1}, ..., E_{i+T-1}\}, k_{min})$
    $PPA^{k_{min}} \leftarrow get\_costs(D_C^{k_{min}})$
  end
  for $C_{E_{max}}^{k_{min}} \in D_C^{k_{min}} \mid C_E^{k_{min}} \neq C$ **do**
    if $PPA_{E_{max}}^{k_{min}} \leq PPA_{max}$ **then**
      $k_{left} = k_{min}, k_{right} = k_{max}$
      while $k_{left} < k_{right}$ **do**
        if $k_{right} \neq k_{max}$ **then**
          $k_{step} = floor(\frac{k_{right} - k_{left}}{T+1})$
          $kvals = \{k_{left} + k_{step}, ..., k_{left} + Tk_{step}\}$
        else
          $k_{step} = floor(\frac{k_{max} - k_{min}}{T})$
          $kvals = \{k_i \in \{k_{min} + k_{step}, ..., k_{min} + (T-1)k_{step}, k_{max}\} \mid k_i > k_{min}\}$
        end if
        do in parallel
          $D_C^{kvals} \leftarrow encrypt(C, E_{max}, kvals)$
          $PPA^{kvals} \leftarrow get\_costs(D_C^{kvals})$
        end
        $kvals_{left} = \{k_i \in kvals \mid PPA_i^{kvals} \leq PPA_{max}\}$
        $kvals_{right} = \{k_i \in kvals \mid PPA_i^{kvals} > PPA_{max}\}$
        if $kvals_{left} \neq \emptyset$ **then**
          $k_{left} = max(kvals_{left})$
        end if
        if $kvals_{right} \neq \emptyset$ **then**
          $k_{right} = min(kvals_{right}) - 1$
        end if
      end while
      return $C_{E_{max}}^{k_{left}}$
    end if
  end for
end for
return $failure$

all $k_i \in kvals$ that meet the PPA constraints, the maximum is used as the new minimum of the search space (k_{left}), unless no key sizes met the PPA constraints in which case the minimum of the search space remains the same. For all $k_i \in kvals$ that do not meet the PPA constraints, one less than minimum is used as the new maximum of the search space (k_{right}). If all key sizes met the PPA constraints, then the maximum remains the same. The loop breaks once the minimum of the search space meets the maximum. So, in the first loop, where k_{max} as a part of the tested $kvals$, if k_{max} meets the constraints, then $k_{left} = k_{right} = k_{max}$, so the loop terminates and returns the design with a key size of k_{max}. This way, if the entire key space is within PPA_{max}, then the algorithm will return k_{max} in one iteration of the key size search.

After the first loop, if k_{max} does not meet the PPA constraints, the key size space to be explored shrinks to between the new k_{left} and k_{right}. In iterations after the first, $kvals$ no longer includes k_{right}, so the search space is divided into $T+1$ sections, and C is encrypted with E_{max} and the key sizes at division points, not including k_{left} and k_{right} (so that the number of encrypted designs created is always T). The PPA of each encrypted design is estimated. The space of potential key sizes shrinks to the set between the maximum key size that met the PPA constraint ($max(kvals_{left})$) and before, but not including, the minimum key size that did not meet the PPA constraint ($min(kvals_{right})$). This explores the design space iteratively by ruling out key sizes between sampled ones that fail, accepting those between ones that succeed as a part of D_C^{opt}, and exploring the rest of the design space in the next iteration. This continues until the unexplored design space shrinks to nothing, and the design with the maximum rated encryption scheme with the maximum key size that does not violate PPA_{max} is returned as C_E^{opt}. Because the design space is guaranteed to shrink each iteration, the algorithm is guaranteed to converge on a single C_E^{opt} if an encryption method, E_{max} can be found. The specific C_E^{opt} found, however, may be inaccurate, based on the margin of error of ICC PPA estimation for each design which it is performed.

Since the first and second step of the algorithm are executed concurrently, their worst-case execution time can be combined. It will be assumed that the execution time of PPA collection steps is equivalent among versions of C. Additionally, it will be assumed that the execution time of estimating PPA with ICC eclipses that of DC ($t_{ICC} >> t_{DC}$), so t_{ICC} will be used as the worst case timing for PPA estimation. For the estimation of PPA for C and C_E^i at k_min, the worst case execution time is the number of PPA calculation "batches" necessary for C and to explore the entire ranking, $t_{ICC}\frac{1+|E|}{T}$. In the concurrent tree key size search, the binary search problem is sped up with concurrent sampling points. This converts the time complexity from $O(log_2(k_{range}))$ to $O(log_{T+1}(k_{range}))$ for T concurrent threads and k_{range} potential key sizes. This results in a worst-case execution time of $log_{T+1}(k_{max} - k_{min})$ for the key size search step of the algorithm. The overall worst case execution time is therefore $t_{exec} = t_{ICC}(\frac{1+|E|}{T} + log_{T+1}(k_{max} - k_{min}))$.

The execution time of this algorithm is a direct improvement over that of the binary search-based algorithm. At $T = 1$ thread, the algorithm becomes equivalent to the binary search algorithm, both functionally and by execution time, with a worst case of $t_{exec} = t_{ICC}(1 + |E|\, T + log_2(k_{max} - k_{min}))$. Additionally, the execution time of the first two steps of Algorithm 2 is $O(\frac{1}{T})$ with T parallel threads, and of the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3059163, IEEE Access

TABLE 5. Opencores [37] and ITC'99 [38] circuits chosen for CDLE experimentation

CIRCUIT	GATE COUNT	
Stepper Motor	193	Module for controllong 4 or 6 wire stepper motor
SS_PCM	427	PCM Interface
USB Phy	507	USB 1.1
sasc	647	Simple asynchronous serial controller which includes 4 byte receive and a 4 byte transmit FIFO and external baud rate generator
Simple SPI	847	Enhanced version of SPI with a wider operating frequency range, 4deep read and write fifo and 8 bit wishbone interface
caprng	913	Generates complex pseudo random numbers
Hilbert Tranformer	1,630	Approximates the hilbert transform with a digital filter
systemcdes	2,096	Implementation of DES algorithm in System C for low power applications
des_area	2,698	Area optimised single DES IP core working in CBC mode (Cipher block chaining), sequential impelementation requiring 16 cycles to complete one encryption or decryption cycle
des3_area	3,072	Area optimised triple DES IP core working in CBC mode (Cipher block chaining), sequential impelementation requiring 48 cycles to complete one encryption or decryption cycle
TV80	6,221	8 bit Z80 compatible microprocessor core
ac97_ctrl	12,121	AC97 Controller core. It provides an interface to external AC97 audio Codec
USB Func	12,984	USB 1.1 slave/device IP core
aes_cipher	17,153	128 bit AES encryption algorithm
sha256core	17,958	Implementation of SHA 256 hashing algorithm
des_perf	23,430	Performance optimised single DES IP core, working in ECB mode. This is a pipelined architecture having 16 cycle pipeline. So it can perform encryption/decryption every cycle
aes_inv_cipher	27,427	AES decryption
des3_perf	71,991	Performance optimised triple DES IP core, working in ECB mode. This is a pipelined architecture having 48 cycle pipeline. So it can perform encryption/decryption every cycle
vga_lcd	124,350	VGA/LCD Controller core is a wishbone revB.3 compliant embedded VGA core capable of driving CRT and LCD displays.
b19	174,519	Cross-connected Viper and and Intel 80386 microprocessors
cf_rca	196,946	A platform for dynamic reconfigurable computing.

final step is $O(\frac{1}{logT})$, both of which strictly decrease as the number of threads increases (again, $T \geq 1$). Therefore, not only is this method a direct improvement of the binary search-based method, but it is a superset of that method, in which the binary search method is the worst version.

V. CONCURRENT TREE SEARCH EXPERIMENTATION

A. EXPERIMENTAL SETUP

To test the CDLE concurrent tree search algorithm, a group of 21 test circuit designs were selected from the Opencores IP core database [37] and ITC '99. benchmark circuit suite [38]. These designs were chosen as examples of modern IP cores that showcase the possible applications of logic encryption. They were also chosen for their range of post-logic synthesis gate counts. All designs were synthesized using a 90nm technology node. Some small designs were selected for proof-of-concept, and then larger designs will show the limits of execution time and PPA estimations for designs with gate counts over 100,000. See Table 5 for more information for each benchmark circuit.

All experimentation was performed using an AMD Ryzen Threadripper 1950X 3.4GHz 16 core (32 thread) processor and 64GB RAM. The number of threads utilized in each experiment was controlled to 8 ($T = 8$), unless otherwise specified. Circuits were encrypted using the compound encryption schemes mentioned in the previous section, with a minimum key size $k_{min} = 32$, and a maximum key size $k_{max} = 512$. For all circuit simulation, 5,000 input patterns were used between those from automated test pattern generation and additional randomly generated test patterns. All synthesis during experimentation used a 90nm technology node.

The concurrent tree search algorithm was developed for this work in the Go programming language [39] as a part of the Constrained and Obfuscated Design-Space Exploration for Security (CODES) platform. CODES provides many tools and methods for encrypting designs, including this CDLE method, and implements encryption schemes.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3059163, IEEE Access

TABLE 6. Encryption strategies chosen by CDLE concurrent tree search for various PPA constraints as a percentage of the original design PPA. A '-' denotes failure of the algorithm to find an optimally encrypted design

Design Name	50%	25%	10%	5%	1%
steppermotor	CHL+SFL 149	CHL+SFL 67	-	-	-
ss_pcm	CHL+SRC 150	CHL+SRC 54	CHL+SRC 32	-	-
usb_phy	CHL+SFL 133	CHL+CYC 103	CHL+SRC 53	-	-
sasc	EFF+SFL 136	CHL+SFL 38	EFF+SRC 257	CHL+SRC 73	CHL+SRC 82
simple_spi	CHL+SAR 345	CHL+SAR 88	-	-	-
caprng	CHL+SRC 153	CHL+SRC 68	CHL+SRC 55	-	-
hilbert	CHL+SFL 502	CHL+SFL 266	CHL+SFL 88	CHL+SFL 37	-
systemcdes	CHL+SFL 512	CHL+SFL 512	CHL+SFL 444	CHL+SFL 255	EFF+SFL 309
des_area_opt	CHL+SFL 512	CHL+SFL 316	CHL+SFL 151	CHL+CYC 32	-
des3_area_opt	CHL+SFL 512	CHL+SFL 482	CHL+SFL 87	CHL+SRC 180	-

Go includes a straightforward multithreading construct, *goroutines*, which were utilized for parallel design space exploration. For each time CDLE was performed, the final encryption scheme and key sizes chosen for C_E^{opt} (if C_E^{opt} exists), the total execution time, PPA of the final and original designs as well as the PPA cost as a percentage increase from the original PPA, and the algorithm trajectory were collected. CDLE was executed for several sets of PPA constraints for each design. The chosen encryption strategies of all PPA constraint sets tested will be reported for all designs successfully encrypted, as well as which constraint sets resulted in failure of the tree search algorithm (i.e. the PPA constraints were too strict). One design, the Hilbert Transformer, was selected for more detailed execution time data collection. A set of CDLE tree search algorithm tests were selected as case studies for closer examination as a representative group in terms of encryption scheme chosen and algorithm trajectory. The case studies will include full algorithm trajectory and PPA results, as well as more detailed information about the designs being studied.

B. RESULTS

The CDLE Concurrent Tree Search algorithm was attempted for all test designs in Table 5. However, for designs with gate counts over 5,000, execution time of the algorithm became prohibitively long. Therefore, a full results set was obtained for the 10 smallest designs of the set. This already shows one limitation of the algorithm, which will be discussed later. The concurrent tree search algorithm was executed for 5 sets of PPA constraints for each test design. Constraints are represented as a percentage increase from the PPA of the original, pre-encrypted design. For each constraint set, the power, performance, and area were constrained to the same percentage increase. The set of constraints used in these experiments were 50%, 25%, 10%, 5%, and 1% for each power, performance, and area. The results of this experimentation are reported in Table 6. A "-" in the table denotes failure of the algorithm to find a suitable encryption strategy (i.e. the PPA constraints were too tight). The compound encryption scheme and key size selected are reported for each experiment. The schemes correspond to the compound encryption schemes utilized, with the following codes for each encryption scheme:

- SFL: SFLL
- SRC: SRCLock
- CYC: Cyclic
- AST: Anti-SAT
- SAR: SARLock
- EFF: Encrypt Flip-Flop
- CHL: ChainLock

The most chosen encryption scheme was ChainLock+SFLL, which is expected because the algorithm prioritizes encryption scheme over key size for security. For sequential encryption schemes, Encrypt Flip-Flop was rarely selected over ChainLock. In most cases, less secure encryption schemes were used for tighter PPA constraints. However, there are some notable exceptions. For usb_phy, SRCLock was chosen at 10% PPA, while Cyclic was chosen at 25%. Also, for SASC, EFF was chosen for ChainLock at 50%, but not for 25%, and the key size increases from 5% and 1% PPA. These inconsistencies could be due to some randomness in the power estimation based on the test patterns chosen during the test pattern generation step of the PPA estimation strategy. Otherwise, the ranking behaves as expected. SFLL was the most popular combinational scheme, and AntiSAT was never used (likely, SARLock was always the better option at similar cost). Also, as the gate count of a design increases, the more likely it is to meet stricter PPA requirements. There are exceptions though, especially SASC.

In Figure 4, the execution times of each CDLE experiment for which an encrypted design is found are reported vs the pre-encryption gate count of each design. This appears to be a roughly linear relationship, with about 1.55s of execution time added per logic gate. This is likely related to the direct proportionality found between t_{exec} and t_{ICC}. It also appears that the spread of execution time tends to increase with gate count. Additionally, the best-case time levels off for larger designs. The best-case scenario for execution time is the case in which the best possible encryption scheme and key size

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3059163, IEEE Access

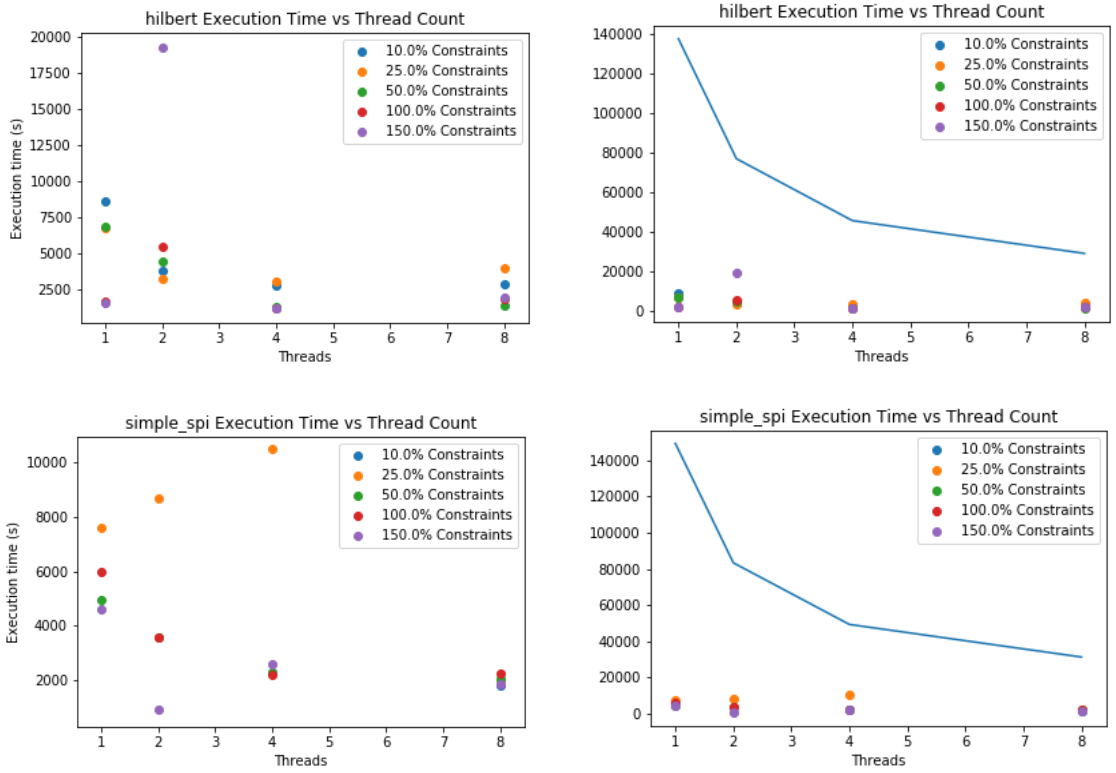


FIGURE 3. Execution time in seconds vs thread count of CDLE encryption of Hilbert Transformer and Simple SPI

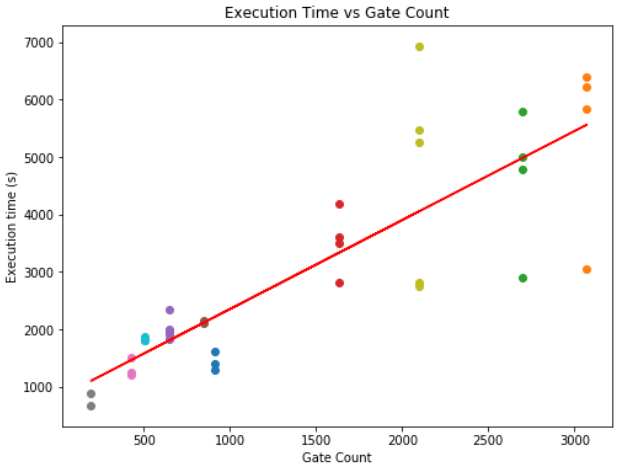


FIGURE 4. Execution time in seconds of each successful CDLE execution vs pre-encryption gate count

are chosen. Since this involves less steps than the worst-case, and less layouts are produced, the result is a smaller increase in execution time vs gate count.

In addition to the CDLE experiments performed with 8 concurrent threads, additional experiments were performed for various thread counts for two selected designs, the Hilbert Transformer and Simple SPI. Data points at 150%, 100%, 50%, 25%, and 10% PPA constraints were collected, for thread counts of 1, 2, 4, and 8 each. The execution time of each of these experiments are plotted in Figure 3. For each, the second plot contains the curve representing the predicted worse-case execution time at each thread count. For the worst case curve, the longest ICC execution time among all of the plotted experiments was used as t_{ICC}. As expected, all of these experiments executed in less time than the predicted worst case. In fact, these experiments did not tend to approach the worst case execution time at all, with very little difference between them compared to the worst case, especially for the Simple SPI design. In general, the execution times do follow the same curve shape as the worst case, with some exceptions such as simple SPI at 25% constraints. The spread of execution time also appears to decrease with increased thread count.

C. CASE STUDY: AREA OPTIMIZED 3-DES

Several individual CDLE experiments have been selected as brief case studies that highlight different behaviors and trajectories of the algorithm and edge cases. One case is presented in this section, and the remaining case studies are in the Appendix. The execution trajectory of the algorithm is represented as a series of bars, from top to bottom. In the first step, encryption schemes are tried in parallel at k_{min} until the best within the PPA constraints is found. In the remaining steps, the key size space is explored until the maximum allowed key size is found. The ticked key

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3059163, IEEE Access

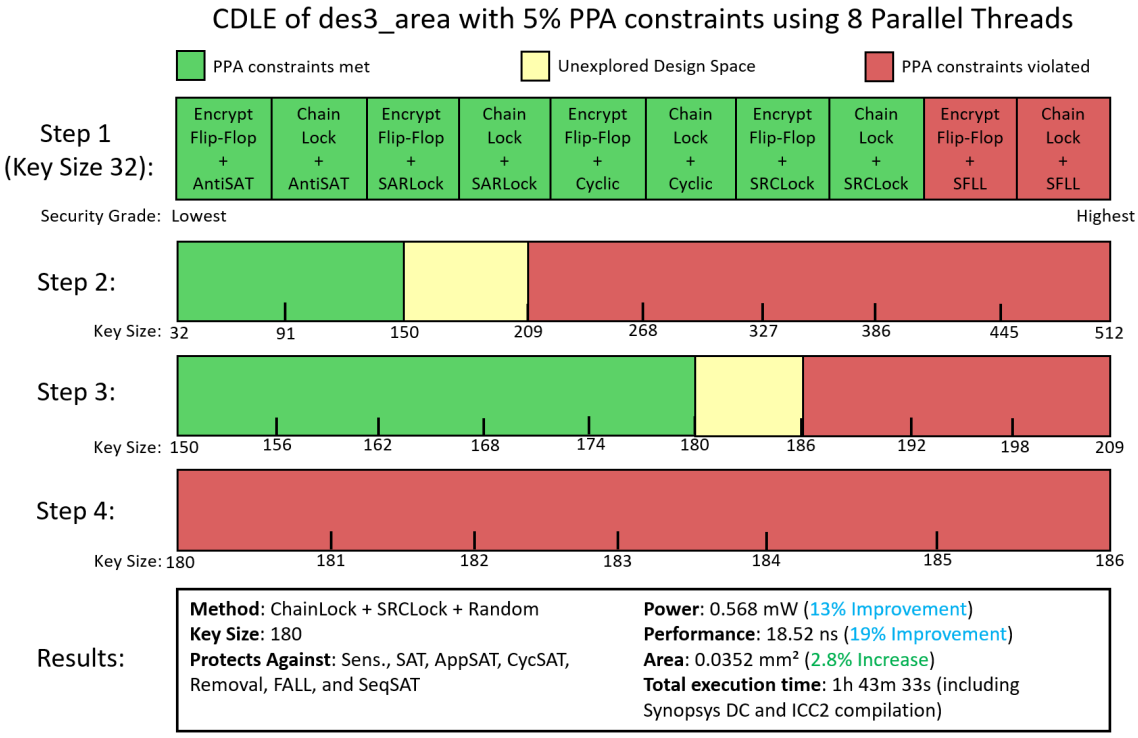


FIGURE 5. CDLE Trajectory and result of des3_area with PPA constrained to 5% increases

sizes between the maximum and minimum of the remaining exploration space are the sampled key sizes at each step, each tried in parallel. The maximum and minimum key sizes inn each step are not sampled, except k_{max}, which is always sampled in the second step. The green "constraints met" area is determined by the maximum sampled key size which met the PPA constraints. The smallest key size that violated the PPA constraints determines the "constraints violated" area. The area between is unexplored, and determines the key size space of the next step. In the final step, the best key size is determined.

The area optimized 3-DES is the largest design among the set of designs used for this set of experiments. The algorithm trajectory is shown in Figure 5. This example shows that even when tight PPA requirements are used on larger designs, a secure encryption scheme can be chosen due to the low impact of logic encryption relative to the size of the design. The chosen encryption scheme was ChainLock+SRCLock at key size 180. In the final step, no key sizes were within the PPA constraints, so the maximum key size of the previous step within the constraints was chosen. The layout produced by ICC2 for the final encrypted design is in Figure 6. The algorithm executed in 1h 43m 33s. The execution time of CDLE quickly began to increase for designs with logic gate counts of more than a few thousand, including this design. The PPA values estimated during this case were:

- Original Power: 0.649 mW
- Original Performance: 22.78 ns
- Original Area: 0.0342 mm^2

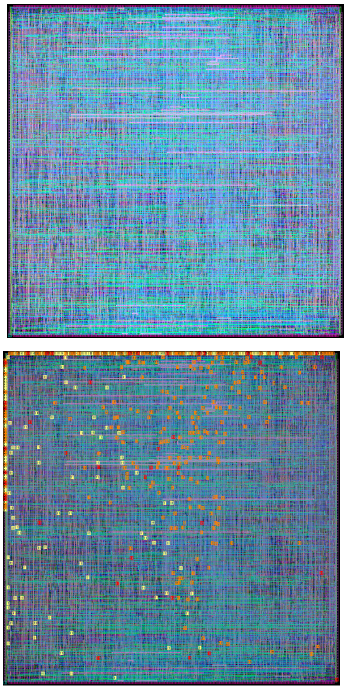


FIGURE 6. Layout of encrypted des3_area. On right, the key inputs and their respective key gates are highlighted, yellow corresponding to ChainLock, orange to SRCLock, and red to randomly placed gates.

- Final Power: 0.568 mW (13% Improvement)
- Final Performance: 18.52 ns (19% Improvement)
- Final Area: 0.0352 mm^2 (2.8% Increase)

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3059163, IEEE Access

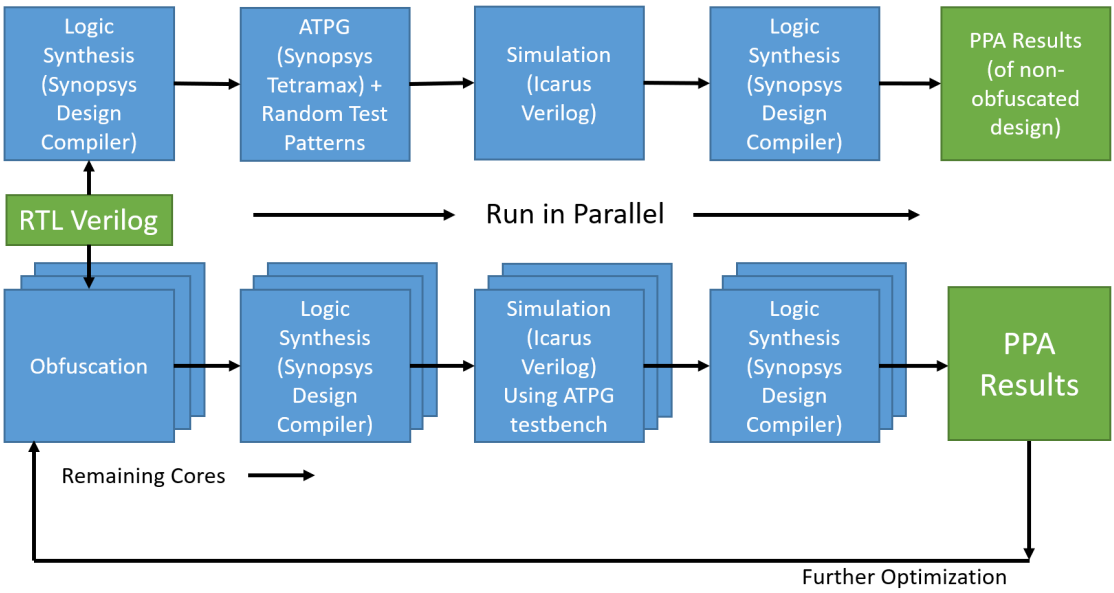


FIGURE 7. Modified PPA collection strategy without IC Compiler II

VI. MODIFIED CONCURRENT TREE SEARCH: DC ONLY

The execution time of the CDLE concurrent tree search algorithm grows linearly with ICC execution time, as seen in Figure 4. For designs with gate counts larger than even 1,000, execution time grew to several hours, and designs with gate counts over 10,000 were skipped in experimentation due to the execution time. Furthermore, Figure 3 shows the predicted behavior of increasing thread count on execution time. That is, increasing the thread count has diminishing returns in that regard for the concurrent tree search algorithms. Therefore, simply increasing the number of threads for execution is not a viable strategy for handling larger designs than 10,000 gates.

The best strategy for decreasing the worst-case execution time, $t_{exec} = t_{ICC}(\frac{1+|E|}{T} + log_{T+1}(k_{max} - k_{min}))$, should be to decrease t_{ICC}. Estimating PPA with IC Compiler II is, in general, very accurate. However, producing a layout for every design explored in D_C before reaching C_E^{opt} is not practical for large designs. As discussed earlier, $t_{ICC} >> t_{DC}$. Logic synthesis executes much more quickly for any given design, and produces PPA estimations based on the resulting netlist. Therefore, to save on execution time, a modified PPA estimation flow is proposed in Figure 7. This flow removes PPA estimation using IC Compiler, as well as the simulation step that informed that estimation.

Effectively, for the new worst case execution time, t_{exec}^{DC}, t_{ICC} is replaced with t_{DC}. It follows that, since $t_{exec}^{ICC} \propto t_{ICC}$ and $t_{ICC} >> t_{DC}$, that $t_{exec}^{ICC} >> t_{exec}^{DC}$. The caveat to basing the algorithm on DC PPA estimation, rather than ICC, is increased inaccuracy of the estimation. During place and route, IC Compiler makes optimizations to the design and routing that change (especially reduce) power usage and area, and improve performance. Therefore, it is expected that improved execution time will be exchanged with quality of results from this CDLE algorithm. However, this opens the possibility of using CDLE with larger designs. In this section, the modified CDLE using only DC will be explored. Other than the PPA collection strategy, the algorithm and design exploration process is unchanged from the one laid out in Algorithm 2.

A. RESULTS

The experimental setup of the concurrent tree search algorithm is equivalent to that of other experimentation for this algorithm. The initial PPA of each unencrypted design was estimated for using both the ICC and DC PPA estimation strategies. The power estimation averages 19% more from DC, and area 26% less, compared to the same unencrypted designs that were estimated with ICC. Performance, however, is much more egregious. The performance for unencrypted designs estimated with DC were, on average, 8.25 times slower than those with ICC. This large disparity shows the importance of estimating these post-layout when accuracy is desired. For CDLE, this may or may not be important, depending on if the disparity between DC and ICC is consistent within the scope of a design to be encrypted. If it is, then CDLE should produce the same results with or without ICC.

The modified design-space search algorithm using only DC estimation was performed for all test benchmarks in Table 5 for 5 sets of PPA constraints. For all constraint sets, the allowed power, performance, and area increase are represented by the allowed percent increase in each of these values compared to those of the original design. For each set of constraints, the same allowed increase was used for each power, performance, and area. For example, 5% denotes a 5% allowed increase in power usage, the same

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3059163, IEEE Access

TABLE 7. Encryption strategies chosen by CDLE concurrent tree search without ICC for various PPA constraints as a percentage of the original design PPA. A '-' denotes failure of the algorithm to find an optimally encrypted design

Design Name	25%	10%	5%	1%	0.10%
steppermotor	CHL+SFL 55	-	-	-	-
ss_pcm	CHL+CYC 38	-	-	-	-
usb_phy	EFF+SFL 78	CHL+AST 45	-	-	-
sasc	CHL+SFL 33	-	-	-	-
simple_spi	EFF+SFL 37	-	-	-	-
caprng	CHL+SRC 85	CHL+SRC 32	-	-	-
hilbert	CHL+SFL 368	CHL+SFL 133	CHL+SAR 74	-	-
systemcdes	CHL+SFL 225	EFF+SFL 74	EFF+SAR 35	-	-
des_area_opt	CHL+SFL 275	-	-	-	-
des3_area_opt	CHL+SFL 368	CHL+SFL 85	CHL+SRC 34	-	-
tv80	CHL+SFL 512	CHL+SFL 90	-	-	-
ac97_ctrl_top	CHL+SFL 512	-	-	-	-
usb_funct	CHL+SFL 512	EFF+SFL 76	-	-	-
aes_cipher	CHL+SFL 512	CHL+SFL 512	CHL+SFL 313	-	-
sha256core	EFF+SFL 444	EFF+SRC 76	-	-	-
des_perf_opt	CHL+SFL 512	CHL+SFL 512	CHL+SFL 489	-	-
aes_inv_cipher	CHL+SFL 512	CHL+SFL 503	CHL+SFL 503	EFF+SAR 86	-
des3_perf_opt	CHL+SFL 512	CHL+SFL 512	CHL+SFL 512	CHL+SFL 35	CHL+AST 50
vga_lcd	CHL+SFL 512	CHL+SRC 254	-	-	-
b19	CHL+SFL 512	CHL+SFL 512	CHL+SFL 512	-	-
cfrca	CHL+SFL 512	CHL+SFL 512	CHL+SFL 512	CHL+CYC 60	-

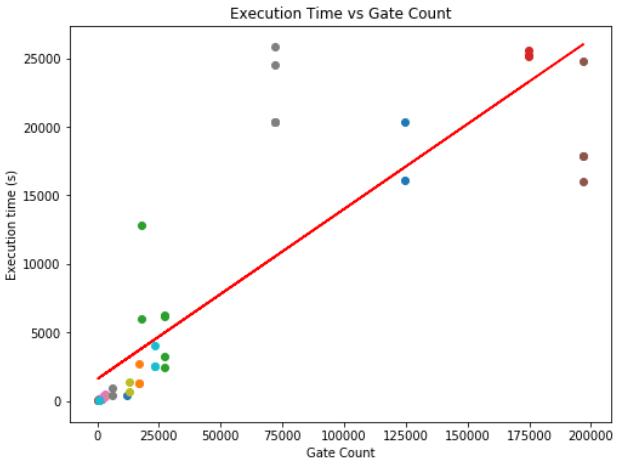


FIGURE 8. Overall execution time in seconds as a function of pre-encryption gate count for each CDLE design experiment

increase for critical path timing, and the same percentage increase for total area. All PPA estimations were collected from Design Compiler, including that of the original design against which the PPA of encrypted designs were compared. For these experiments, 25%, 10%, 5%, 1%, and 0.1% PPA constraints were used, which use smaller values than the ICC experimentation due to the inclusion of larger designs. The encryption strategies, including encryption schemes and key sizes chosen, for each design at each PPA constraint set are shown in Table 7. A '-' denotes failure of the algorithm to find an optimally encrypted design (i.e. the PPA constraints are too strict to find a viable encryption strategy). The three-letter codes for encryption schemes used are the same as those used in previous experiments. Additionally, like in previous encryption strategies, 45% of key inputs are used for each listed encryption strategy, and 10% are reserved to drive randomly placed key gates.

The predominant encryption schemes chosen using the algorithm was ChainLock+SFLL. By design, the process will select the strongest encryption algorithm when possible, to protect against as many attacks as possible. There are several instances, especially for larger designs, where the strongest possible encryption strategy, ChainLock+SFLL at key size 512, was selected. This indicates situations in which the PPA constraints are loose enough to allow any encryption scheme in the security goal S_{opt}. As expected, tightening PPA constraints lowered the optimized security metrics, until an encrypted design within the PPA constraints could not be found.

The execution time of each successful CDLE experiment is plotted as a function of the gate count of the original design in Figure 8. These are the overall execution times of the algorithm, including each DC execution. The execution time of each PPA constraint for each design in Table 7 is reported and therefore each gate count has several data points

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3059163, IEEE Access

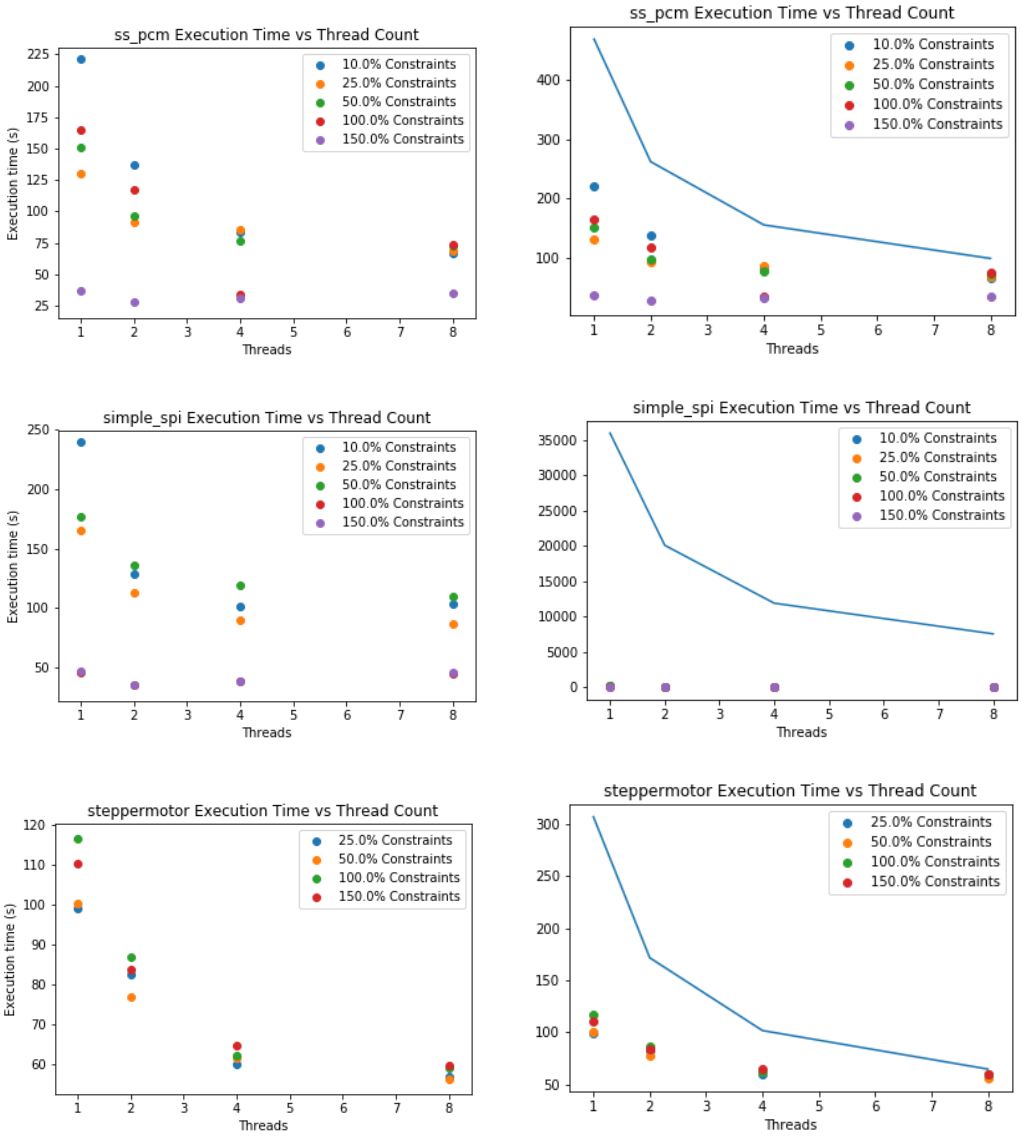


FIGURE 9. Execution time in seconds vs thread count of DC-Only CDLE encryption of SS_PCM, Simple SPI, and Stepper Motor

associated with it. As expected, the overall execution time is much improved compared to that of the concurrent tree search using ICC, having an expected average execution time of about 0.124s per pre-encryption logic gate, compared to 1.55s with ICC. The largest designs, which contain close to 200,000 logic gates, have an execution time on the order of a few hours, which is comparable to that of 10,000 gate designs when ICC is used for PPA estimation. There appears to be a linear increase in average execution time for any given design for gate count. This is in line with the linear relationship between t_{DC}, the worst-case execution time of Design Compiler, and the worst-case overall execution time. However, the maximum execution time for each design appears to level off just above 25,000 seconds. This could be due to the maximum algorithm execution time being a rare case for each experiment.

Similar to the experiments using ICC, execution time was also explored as a function of thread utilization in a separate set of experiments. The results of this experimentation are shown in Figures 9 and 10, which have both a plot containing only the data, and one that also contains the expected worst-case execution time for each design. The designs used for these experiments were the Hilbert Transformer, Simple SPI, SS_PCM, Stepper Motor, and USB Physical designs. However, PPA constraints of 150%, 100%, 50%, 25%, and 10% increases in each metric were used for all designs tested. Higher PPA constraints were used compared to other DC experimentation due to the selected designs being among the smallest in terms of pre-encryption gate count. The worst-case execution time in each case is plotted as well, using the worst case DC execution time among these sets of experiments as t_{DC} for each design. As expected, in all cases,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3059163, IEEE Access

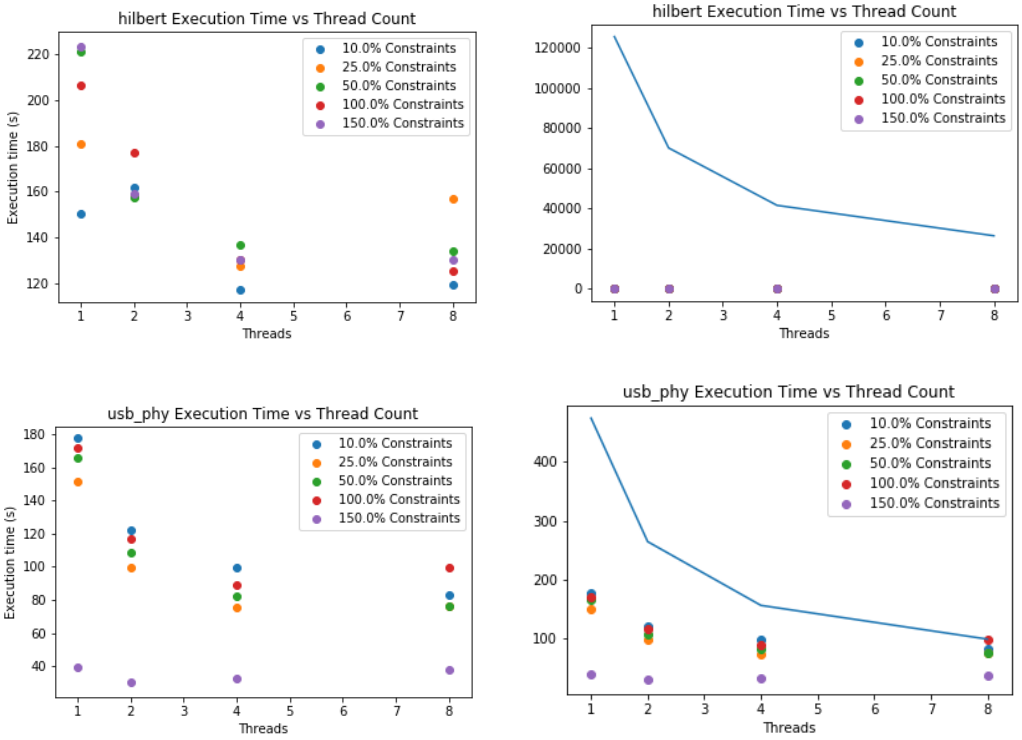


FIGURE 10. Execution time in seconds vs thread count of DC-Only CDLE encryption of Hilbert Transformer and USB Physical

the overall execution time of each experiment was below the worst-case execution time limit, and tended to follow a similar shapes to their respective worst-case curves. Simple SPI and the Hilbert Transformer, however, had execution times well below the worst-case for all tested constraints.

The results of the USB Physical core show an interesting case. All of the data points lie below the worst-case curve, except for one experiment in which the maximum execution time was met for one 8-thread experiment. This data point corresponds to the 10% PPA constraint experiment, in which the resulting encryption strategy was ChainLock+Anti-SAT at key size 45. ChainLock+Anti-SAT is the second lowest ranked encryption scheme. With a total of 10 encryption schemes and 8 threads, this places the encryption scheme in the worst-case group in execution order in the second step of the algorithm. Also, at a key size of 45, the third step of the algorithm represented a full concurrent tree search. Therefore, this experiment represents the worst-case execution of the algorithm. This shows the accuracy of the predicted worst case execution time. Another interesting data point is the 100% PPA constraint experiment of the same design. In this case, an extra step needed to be added to the algorithm to fully explore the key size space. This made the trajectory approach the worst case execution time, but not violate it.

B. CASE STUDY: AREA OPTIMIZED 3-DES

Brief case studies have been selected to highlight specific CDLE experiments using only DC for PPA estimation. Some studies have been selected as interesting cases, and some for comparison of the corresponding case study done for ICC experimentation. In these studies, a layout was produced with ICC2 after the completion of CDLE, and PPA estimations were made from the layout for comparison with the DC estimations. One case study is shown in this section, and other case studies are presented in the Appendix. The figures are organized in the same way as in the previous case studies.

The area optimized 3-DES case was selected as a comparison between the concurrent tree search algorithm using ICC PPA estimation and DC. The algorithm trajectory is shown in Figure 11. In the corresponding ICC case study, ChainLock+SRCLock with key size 180 was the optimal encryption strategy. In this case, the same encryption scheme was chosen, but at a smaller key size of 34. This result is slightly less conclusive, but will definitely be within the PPA constraints based on the ICC result. This case finished executing after 5m 57s, versus the 1h 43m 33s execution time of the same case with ICC.

There was some disparity between the PPA values from ICC versus DC. For the unencrypted des3_area, the power estimation is 27% larger from DC compared to ICC, 296% slower in performance, and 18% smaller in area. This disparity, especially in performance, could degrade the quality of results from CDLE when using only DC to estimate PPA. The PPA estimated in this experiment were:

- Original Power: 0.725 mW
- Original Performance: 90.2 ns

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3059163, IEEE Access

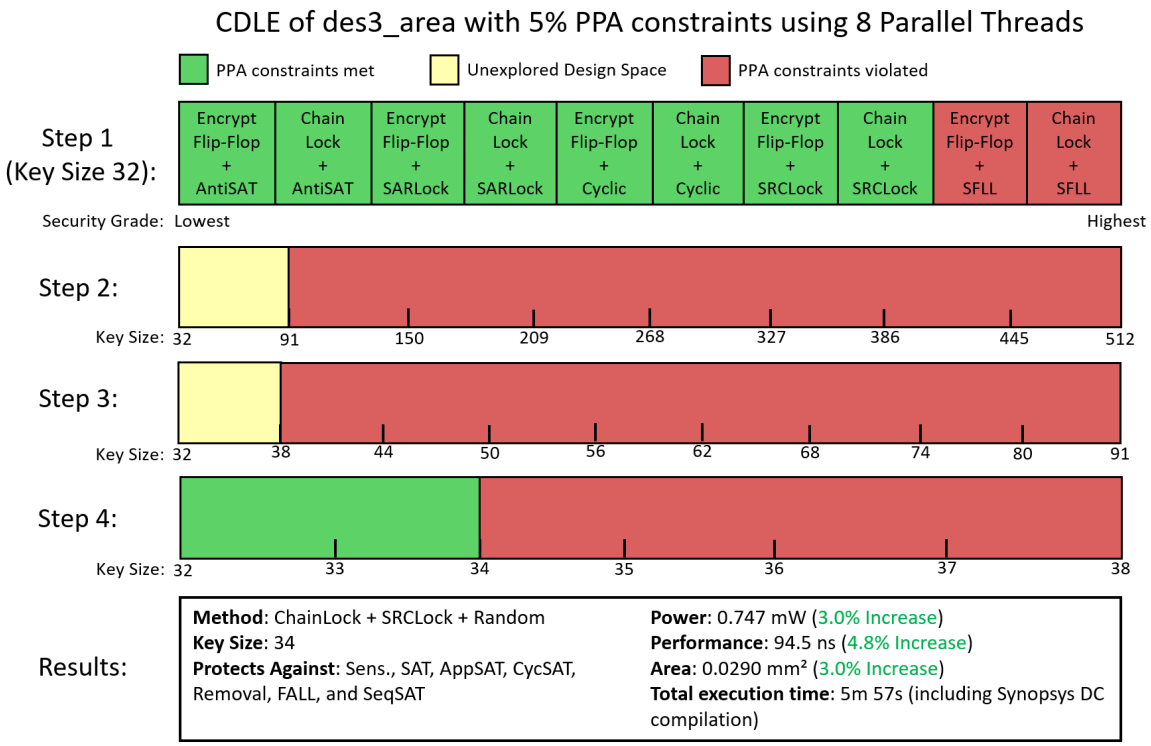


FIGURE 11. CDLE Trajectory and result of des3_area with PPA constrained to 5% increases

- Original Area: 0.0282 mm^2

- Final Power: 0.747 mW (3.0% Increase)
- Final Performance: 94.5 ns (4.8% Increase)
- Final Area: 0.0290 mm^2 (3.0% Increase)

The layout produced with ICC2 after CDLE is shown in Figure 12, with key gates and their respective driving key inputs highlighted on right. The design PPA was also estimated from this layout. The power usage was significantly lower, the performance drastically improved, and area usage was slightly greater compared to the DC estimation:

- Layout Power: 0.605 mw
- Layout Performance: 11.4 ns
- Layout Area: 0.0309 mm^2

C. COMPARING ICC AND DC STRATEGIES

Comparing the execution times of the concurrent tree search of CDLE using the full PPA estimation strategy with IC Compiler II, and the modified strategy using Design Compiler only, there is a clear advantage of using DC only for estimations over ICC. The execution time of the algorithm is greatly reduced in the former case, as anticipated. For all of the cases that can be compared (matching design and PPA constraints), 67.5% of 40 experiments returned matching encryption schemes but different key sizes, which includes cases where no encryption strategy was found for either strategy. Of the 25 cases shown, 48% returned matching encryption schemes. Five cases were able to return a result using ICC estimation, but no result for DC estimation. There

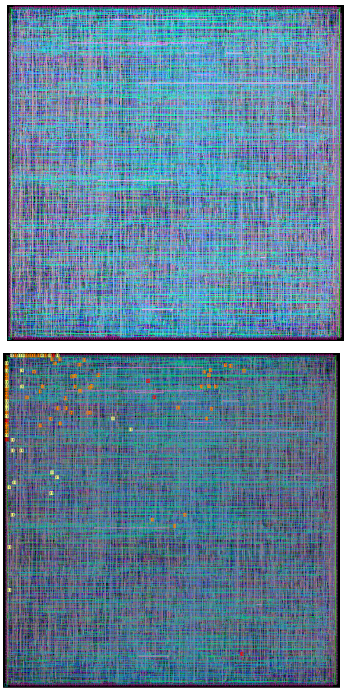


FIGURE 12. Layout of encrypted des3_area after CDLE with DC only. On right, the key inputs and their respective key gates are highlighted, yellow corresponding to ChainLock, orange to SRCLock, and red to randomly placed gates.

are no cases in which DC returned a result, and ICC did not. There are also no cases in which both strategies returned

the exact same encryption scheme and key size. However, a couple of cases returned results with a difference of less than 10 key bits, such as des3_area_opt with 10% constraints, SASC at 25%. When at least one strategy found an encryption strategy, estimation with ICC returned a higher security strategy than estimation with DC in 80% of cases either by scheme or key size, including those in which no strategy was found with DC.

VII. CDLE MACHINE LEARNING APPROACH

The CDLE Concurrent Tree Search method showed a direct speedup compared to the binary search-based method proposed by Luria, et al. [9]. By utilizing parallel computation threads, the design space was more efficiently explored. However, there were diminishing returns in execution time with respect to thread count. Even with additional threads, the method could have a prohibitively long execution time for large designs. This is primarily due to the PPA estimation strategy, which heavily relies on many instances of logic synthesis and place and route to accurately estimate PPA. The limitations of the concurrent tree search method highlight a need for accurate and quick CDLE methods. One potential strategy to achieve this is by taking a machine learning approach to CDLE. Building a model with which to predict PPA costs would front-load the execution time to a training loop. Comparatively, predicting a CDLE outcome would take a negligible amount of time. Therefore, we will explore a machine learning CDLE method.

The machine learning implementation used for CDLE in this work will involve two steps. The first is the training loop. In the training loop, several designs will be selected as a training set. The gate pre-encryption post-logic synthesis gate count of each design will be collected. Each design will be encrypted with the individual encryption schemes that make up the compound encryption schemes identified for implementation earlier in this work. Therefore, each training design is encrypted with the encryption methods below individually, at key sizes ranging from 32 to 256 at steps of 32. Therefore, a prediction for each compound encryption scheme can be built at any key size between 32 and 512. The individual encryption schemes used in training are:

- SFLL [8]
- SRCLock [20]
- Cyclic [19]
- Anti-SAT [31]
- SARLock [10]
- Encrypt Flip-Flop [11]
- ChainLock [14]
- Random (EPIC [2])

The power, performance, and area of each encrypted design will be estimated using the full PPA estimation strategy using IC Compiler II estimations to build the machine learning model. Each PPA estimation will be compared to that of the unencrypted design, so each PPA data point will be a percent cost from the origianl design. The data from PPA estimation will be analyzed so an accurate regression of the data can be built. The regression model will use encryption scheme, key size, and design gate count as independent variables to predict each power, performance, and area cost of any encrypted design. It is expected that three submodels will be built for each PPA metric. Furthermore, due to the potential difficulty of quantifying the encryption scheme variable, a regression will be made for each encryption scheme, with key size and design gate count as independent variables. This will be done for each PPA submodel, so the total number of expected regressions created is $3 * |E|$, where E is the set of training encryption schemes. Additionally, a strategy for building predictions for compound encryption schemes with the model will be devised. This should provide a very accurate model with which to predict the PPA of designs within a design space of potential encrypted designs in the context of a CDLE flow.

The second step of the machine learning implementation is the experimentation step. In this step, the regression model built in the training loop will be utilized to make predictions for a set of experimental designs separate from the training set. The same designs that were used to experiment with the CDLE Concurrent Tree Search will be used for this step. To search the design space, Algorithm 3, which is a modification of the binary search-based Algorithm 1, will be used. In this algorithm, the $encrypt$ and get_cost functions are replaced with $predict\_cost$, which uses the regression model to predict the PPA of an encrypted version of C based on the given encryption scheme and key size without needing to encrypt. The Concurrent Tree Search algorithm could also be used, but the main expected complexity of the experimentation is calculating the regression from training data, not in the execution of the algorithm itself. Therefore, the expected overall speedup from parallel threads is negligible.

One goal of the machine learning approach is for it to be extensible to designs outside of the training set, so the training set will not overlap with this experimentation step at all to test the generalization limits of the machine learning model. This experimentation step should give a look into the capability of the machine learning approach to CDLE, and ideas for improvements to the approach so it can become a robust solution to the CDLE problem.

A. TRAINING

A set of 22 designs from the Altera Synthesis Cookbook [40] were selected to train the CDLE machine learning model. There is no overlap between this training set and the test designs used to perform experiments for the concurrent tree search-based algorithm. Instead, the latter will be used to perform experiments using the machine learning model. The training set designs, their post-logic synthesis gate count, and functions are listed in Table 8.

Several designs were generated by varying the input/output bus, and therefore the size of the overall circuit in terms of gate count. This was done to ensure a predictable spread of gate counts among the designs, to increase coverage of the potential use cases for the prediction. The gate counts of the overall set was restricted to below 30,000 to reduce training time.

Algorithm 3 Constraint-Directed Logic Encryption Driven by Machine Learning

Require: Original Circuit C; Key width bounds k_{min}, k_{max}; Cost constraints PPA_{max}; Ranked encryption methods E
Ensure: Encrypted circuit C_E^{opt}

for $e \in E$ **do**
  $PPA_e^{k_{min}} \leftarrow predict\_cost(e, k_{min})$
  if $PPA_e^{k_{min}} \leq PPA_{max}$ **then**
    $PPA_e^{k_{max}} \leftarrow predict\_cost(e, k_{max})$
    if $PPA_e^{k_{max}} \leq PPA_{max}$ **then**
      return $C_e^{k_{max}}$
    end if
    $k_{left} = k_{min}, k_{right} = k_{max}$
    while $k_{left} \leq k_{right}$ **do**
      $k_{mid} = floor((k_{left} + k_{right})/2)$
      $PPA_e^{k_{mid}} \leftarrow predict\_cost(e, k_{mid})$
      if $PPA_e^{k_{mid}} \leq PPA_{max}$ **then**
        $k_{left} = k_{mid}$
      else
        $k_{right} = k_{mid} - 1$
      end if
    end while
    return $C_e^{k_{left}}$
  end if
end for
return $failure$

There are two main limitations to this training set. One, the size of the training set is lacking. Ideally, training data for hundreds or even thousands of designs would be collected to ensure adequate coverage of potential logic patterns and netlist constructions in the estimation. The second limitation is the size of the designs. Due to the large amount of data collection being attempted, the gate size was limited compared to the test set. The purpose of this exploration into a machine learning approach is to offer a possible alternative to the methods presented so far in this work. This machine learning model will serve as a limited example and impetus for machine learning in logic encryption.

For each design in the training set, encryption was attempted using the individual (non-composite) encryption schemes at key sizes between 32 and 256 with a step size of 32 (up to 64 total encrypted versions of each design). At times, encryption can fail, especially for small designs using encryption schemes that have a requirement on input space and/or non-flop gate count, or if the circuit structure otherwise does not support the encryption scheme. For each encrypted design, the PPA was estimated using both the full ICC PPA estimation strategy as well as the modified one using only DC, and compared to that of the pre-encrypted design. The percentage increases in PPA were recorded as one data point for each (3 total data points per encrypted design). A total of 8,808 data points were collected for all PPA for all designs, representing a total of 1,468 ICC/DC PPA estimations.

TABLE 8. Designs from the Altera Synthsis Cookbook [40] selected to train the CDLE machine learning model

CIRCUIT	GATE COUNT
Approx FP Sqrt	92
CRC32 DAT48	378
Stream MUX	382
UART	383
VGA	474
64-bit Descrambler	683
64-bit Scrambler	699
128-bit Descrambler	1,088
128-bit Scrambler	1,211
Gearbox 32x33	1,319
64-bit Divider	1,853
256-bit Descrambler	1,888
256-bit Scrambler	2,452
512-bit Descrambler	3,493
128-bit Divider	3,733
256-bit Divider	7,788
512-bit Scrambler	7,971
AES	10,139
1024-bit Descrambler	13,972
512-bit Divider	14,919
1024-bit Scrambler	19,745
1024-bit Divider	29,543

Several machine learning models were considered which each utilize separate regressions for power, performance, and area for each encryption scheme, a total of 24 regressions per model. Two basic features were considered for each regression model: key size and pre-encryption gate count. The gate count feature should be seen as a course feature to estimate more subtle circuit features that effect the PPA impact of encryption. The exploration of such features is left to future research. The relationship between key size and PPA cost was strongly linear in most cases. See the examples in Figure 13. Area and power costs both showed strong positive linear correlation. Performance, on the other hand, tended to be constant in key size in most cases. Simple linear regression was chosen for power, performance, and area to model the key size behavior, if gate count is held constant. To determine the correlation between gate count and PPA cost, the costs were plotted as a function of pre-encryption gate count for all training designs, controlling for key size and encryption scheme. See the examples in Figure 14. For most cases, a positive inverse correlation fit best for pre-encryption gate count. With these correlations in mind, three candidate multivariable regression equations to relate PPA cost to both key size and design gate count tested were considered:

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3059163, IEEE Access

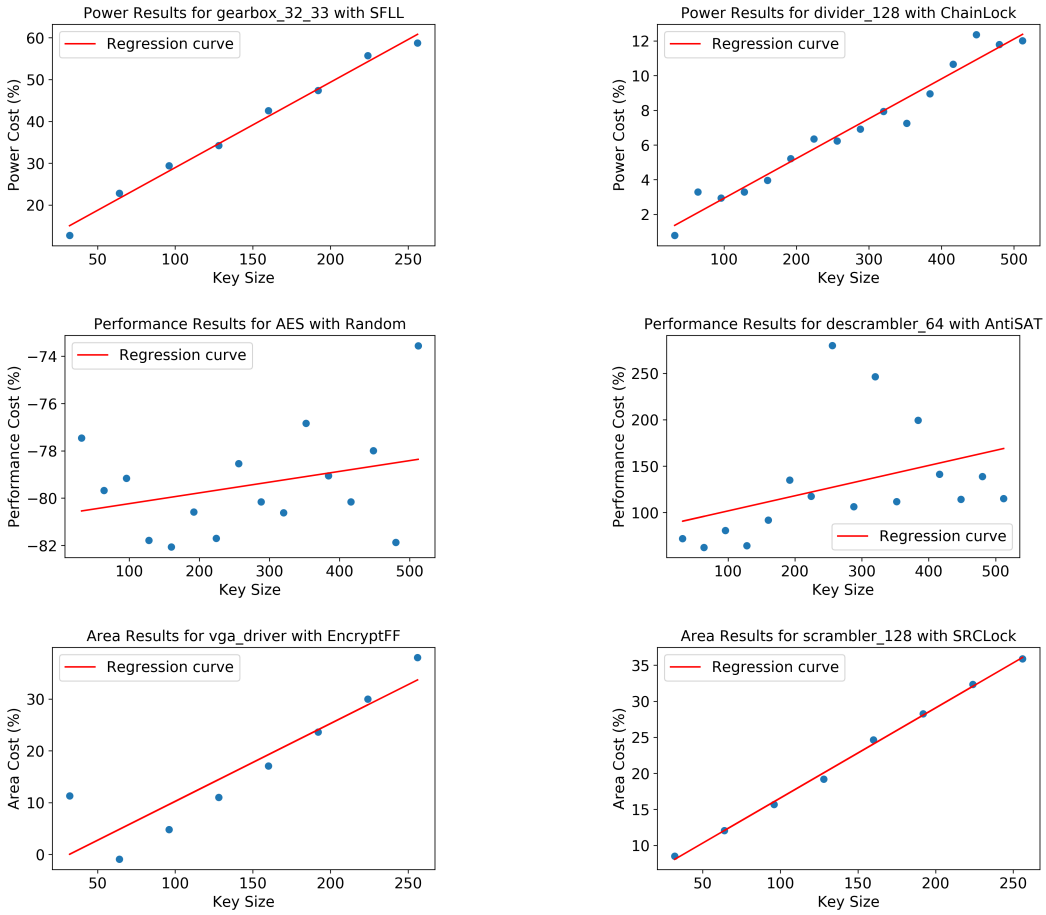


FIGURE 13. Selected examples of regression for key size vs PPA cost

- **Linear**: $PPA = a + bk + \frac{c}{g}$
- **Interaction**: $PPA = a + b\frac{k}{g}$
- **Linear + Interaction**: $PPA = a + bk + \frac{c}{g} + d\frac{k}{g}$

where PPA is power, performance, or area, k is the key size, g is the gate count, and all other variables are regression coefficients. Each of these models could potentially result in the single-variable behavior seen during training. Therefore, the results of models using each regression equation will be compared.

Each regression model was created using Python 3.6.3 [41] using the SciPy 1.5.2 [42] least squares method. Three candidate robust least squares regression methods implemented in SciPy were chosen for comparison of quality of results. Each least squares method varies the loss function of residuals when minimizing the cost function, as described in the SciPy optimize least squares function documentation [42]:

- **Soft_l1**: $rho(z) = 2(\sqrt{1+z} - 1)$
- **Cauchy**: $rho(z) = ln(1+z)$
- **Arctangent**: $rho(z) = arctan(z)$

The normalized root mean square error (NRMSE) was collected for each linear fit model and averaged over the different cost functions, reported in Table 9 for ICC training data and Table 10 for DC data. The RMSE of each model was normalized using the average PPA cost of the model. These NRMSEs show a highly linear relationship for power and area in most cases, but performance is not well fit by a linear regression. Models with low NRMSE confirm the behavior seen in the training data in Figures 13 and 14. Therefore, linear regression works for this preliminary CDLE example. However, better regression models may exist and should be explored in future research.

Each of these least squares methods was tested using each regression equation for all necessary regressions to build the models. This resulted in a total of 9 candidate regression models for CDLE, with two submodels each using ICC and DC training estimation. Each submodel used 24 regressions. Each regression model was tested and the results of each will be compared to determine the model that provides the best quality of results.

The last attribute of the regression model to consider is how to build predictions for compound encryption schemes using those for individual schemes. When applying a compound encryption scheme, the schemes are applied in order, and a fraction of the key space is dedicated to each encryption scheme. Considering the linear relationship between cost and

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3059163, IEEE Access

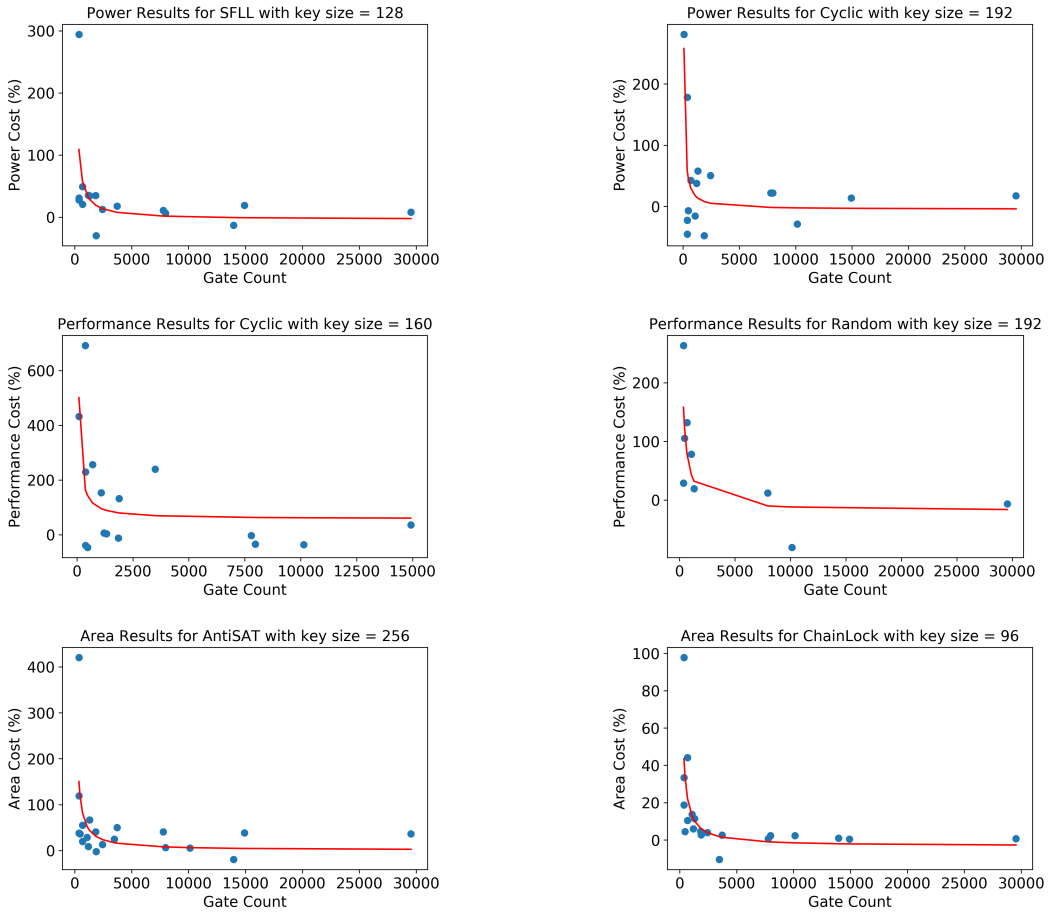


FIGURE 14. Selected examples of regression for gate count vs PPA cost

TABLE 9. Normalized root mean square error, averaged among cost functions for each fit model using results with ICC PPA estimation

Model	Average NRMSE - ICC		
	Power	Performance	Area
Linear	0.236	1.22	0.0271
Interaction	0.156	0.627	0.0271
Linear+Inter	0.195	0.898	0.0271

TABLE 10. Normalized root mean square error, averaged among cost functions for each fit model using results with DC PPA estimation

Model	Average NRMSE - DC		
	Power	Performance	Area
Linear	0.0445	0.595	0.0286
Interaction	0.0445	0.515	0.0286
Linear+Inter	0.0445	0.707	0.0286

key size found during training for power and area, it will be assumed that the costs of encryption schemes in a compound strategy are simply additive. So, power and area costs will be predicted for each encryption scheme for the key size equivalent to the size of the key space dedicated to them, and then summed. For performance, which is constant for key size, the cost of each scheme will be predicted in a similar manner, and the maximum among them will be accepted as the final performance cost prediction. The performance cost being based on longest timing path lends itself to this sort of prediction model as well.

B. TEST RESULTS

To test the effectiveness of the 9 candidate CDLE machine learning models, predictions of C_E^{opt} were made for all test designs for each regression model. The experimental setup used is the same as that for the concurrent tree search, except using the machine learning models to make predictions for C_E^{opt} of each test design, for several sets of PPA constraints. The models were implemented into the CODES platform and predictions were made using the regression equation pertaining to the model under test, and the regression coefficients determined by least squares. Predictions were made using both the models built from PPA estimations using IC Compiler II, as well as the model using Design Compiler PPA estimations with PPA constraints of 25%, 10%, 5%, and 1%, and 0.1%. These values were chosen to match the experimentation done

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3059163, IEEE Access

TABLE 11. Encryption scheme and key size predicted by the regression with the highest quality of results using ICC training estimations, interaction regression with a Cauchy cost function.

Design Name	25%	10%	5%	1%	0.10%
steppermotor	CHL+SFL 35	-	-	-	-
ss_pcm	CHL+SFL 80	-	-	-	-
usb_phy	CHL+SFL 94	-	-	-	-
sasc	CHL+SFL 122	CHL+SFL 32	-	-	-
simple_spi	CHL+SFL 157	CHL+SFL 42	-	-	-
caprng	CHL+SFL 166	CHL+SFL 46	-	-	-
hilbert	CHL+SFL 297	CHL+SFL 81	-	-	-
systemcdes	CHL+SFL 414	CHL+SFL 118	-	-	-
des_area_opt	CHL+SFL 503	CHL+SFL 145	CHL+SAR 35	-	-
des3_area_opt	CHL+SFL 512	CHL+SFL 168	CHL+SAR 47	-	-
tv80	CHL+SFL 512	CHL+SFL 301	CHL+SFL 47	-	-
ac97_ctrl_top	CHL+SFL 512	CHL+SFL 512	CHL+SFL 106	-	-
usb_funct	CHL+SFL 512	CHL+SFL 512	CHL+SFL 97	-	-
aes_cipher	CHL+SFL 512	CHL+SFL 512	CHL+SFL 136	-	-
sha256core	CHL+SFL 512	CHL+SFL 512	CHL+SFL 136	-	-
des_perf_opt	CHL+SFL 512	CHL+SFL 512	CHL+SFL 171	-	-
aes_inv_cipher	CHL+SFL 512	CHL+SFL 512	CHL+SFL 203	-	-
des3_perf_opt	CHL+SFL 512	CHL+SFL 512	CHL+SFL 512	-	-
vga_lcd	CHL+SFL 512	CHL+SFL 512	CHL+SFL 512	-	-
b19	CHL+SFL 512	CHL+SFL 512	CHL+SFL 512	-	-
cfrca	CHL+SFL 512	CHL+SFL 512	CHL+SFL 512	-	-

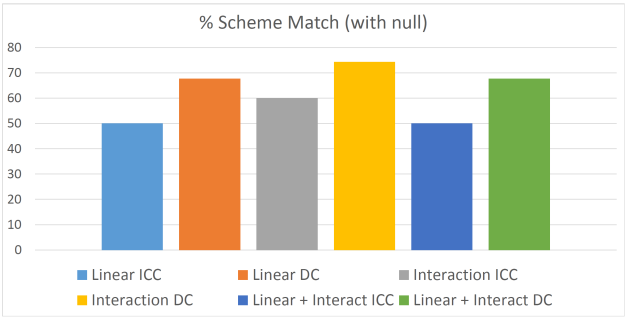


FIGURE 15. Percent of CDLE experiments that matched chosen encryption scheme when compared with the corresponding DC or ICC concurrent tree search results for each machine learning model tested. Experiments in which no encryption scheme could be found were counted as matches. Each cost function produced the same results, so these are not shown.

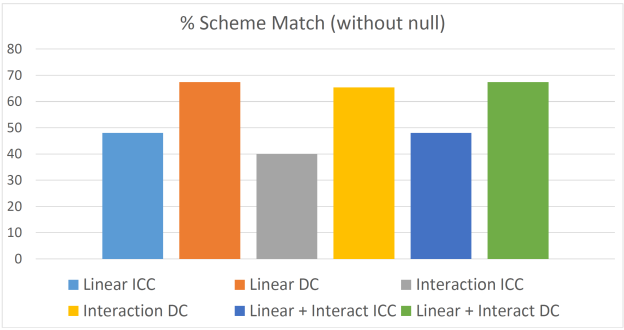


FIGURE 16. Percent of CDLE experiments that matched chosen encryption scheme when compared with the corresponding DC or ICC concurrent tree search results for each machine learning model tested. Experiments in which no encryption scheme could be found were ignored. Each cost function produced the same results, so these are not shown.

in the Concurrent Tree Search DC experiments, since the full set of experimental designs was utilized for each model. After CDLE results were predicted for each model, they were each compared to the results of the CDLE concurrent tree search experiments for quality of results. The regression models using ICC PPA estimations were compared to the Concurrent Tree Search ICC experimentation, and likewise for regression models using DC estimations and the DC Tree Search results for all test cases that overlap.

In Figures 15 – 17, the quality of results of the regression models tested are compared to the control data, which are the respective concurrent tree search ICC and DC results. The three metrics used to determine quality of results were the percent of overlapping test cases that produced the same encryption scheme as the control including cases in which no valid encryption strategy could be found (called "null" cases), the same metric but excluding all null cases, and the percentage of cases in which the machine learning model returned a result, but the concurrent tree search algorithm returned no result. The final metric is referred to as "false positives". The models for each individual cost functions are grouped

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3059163, IEEE Access

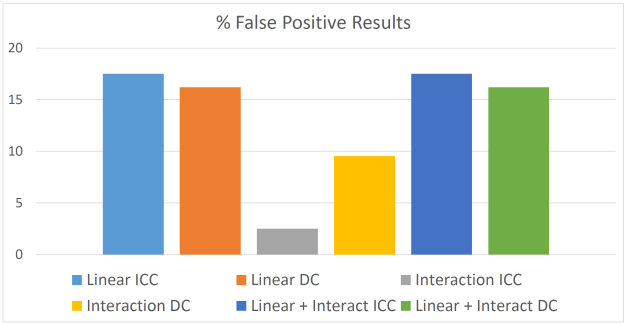


FIGURE 17. Percent of CDLE experiments in which no encryption strategy could be chosen for DC or ICC concurrent tree search results, but one was found for the machine learning model in question. Each cost function produced the same results, so these are not shown.

together under their overarching regression equation, because the results are equivalent among different cost functions in terms of these metrics. The only variation in results among the cost functions was some change in key sizes.

When considering null results as matching between the control and machine learning cases, the interaction models performed significantly better than the purely linear and combination models. However, when the null matches are not considered, the interaction model performed slightly worse than the other models. However, when considering false positive cases, the interaction model performed much better than the others. In fact, the interaction model using ICC training estimations showed almost no false positive cases. For the linear and combination models, cases in which an encryption strategy could be found for both machine learning and concurrent search tended to be more accurate than those for the interaction model. However, this comes at the cost a large number of false positive cases.

The linear and combination models did not vary in result for increasing gate count. However, the interaction model followed the expected behavior the closest, that is, as the design gate count increased encryption strategies could be found using tighter PPA constraints. Therefore, the interaction model showed the highest quality of results. Since the results among cost functions were similar, the cost function with the lowest average relative error, the Cauchy function, was chosen as the highest quality result. The experimentation results for the interaction model using Cauchy cost are shown in Table 11 using ICC training estimation. The results in each case tended to be dominated by ChainLock+SFLL encryption. While this was the most chosen scheme in the concurrent tree search algorithm as well, there were a significant amount of cases in which other schemes were chosen. However, in these results, only a couple of cases show other obfuscation schemes used. Additionally, the variation in the chosen schemes with gate count is not as strong as in the concurrent tree search results. In the following section, adjustments are made to the interaction Cauchy regression model to better fit the results from concurrent tree search.

C. MODEL ADJUSTMENTS

The regression model fitting the interaction equation and using the Cauchy cost equation showed the highest quality of results among the 9 regression equation and cost function combinations tested. When compared to the results of the concurrent tree search algorithm, this model matched the results the most closely among the models tested, especially considering cases in which other models found a "false positive" and returned a result when the concurrent tree search did not. However, even considering matches of encryption scheme chosen between machine learning and concurrent tree search, the best machine learning model still showed very little variation in encryption scheme chosen, with an overabundance of cases in which Chainlock and SFLL were chosen. Additionally, the relation between gate count and encryption strategy was not as strong in the interaction and Cauchy regression model as it was in the concurrent tree search. With these differences in mind, several adjustments were made to the highest quality machine learning model in order to better fit the concurrent tree search results. These adjustments were made under the assumption that the limitations of the training set caused these inconsistencies and that the adjustments merely account for the training set. The adjustments made were:

1) All regression coefficients for SFLL were scaled up by a factor of 10. It is possible that the training designs chosen happened to favor the SFLL block architecture, which includes Hamming distance calculation. Many training designs were communication modules, which include comparators that could also be found in SFLL Hamming distance hardware. This could have caused significant underestimation of the cost of SFLL.
2) The intersection term of all regressions were multiplied by a factor of 0.01. A key size of 0 should result in no added cost, regardless of overall gate count. Therefore this term was effectively eliminated from each model.
3) For all regressions, the coefficient of the interaction term was scaled by a factor of 5. This adjustment was made to compensate for the reduction of the intersec-

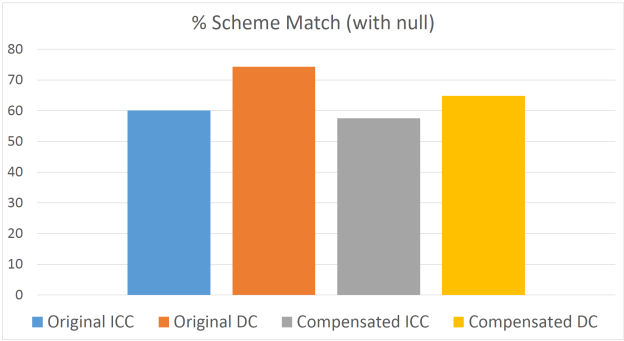


FIGURE 18. Percent of CDLE experiments that matched chosen encryption scheme when compared with the corresponding DC or ICC concurrent tree search results for the adjusted and original versions of the highest quality machine learning model. Experiments in which no encryption scheme could be found were counted as matches.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3059163, IEEE Access

TABLE 12. Encryption scheme and key size predicted for each test design and PPA constraint set by the regression with the highest quality of results using ICC training estimations, interaction regression with a Cauchy cost function, after adjustments were made.

Design Name	25%	10%	5%	1%	0.10%
steppermotor	-	-	-	-	-
ss_pcm	EFF+SFL 32	-	-	-	-
usb_phy	CHL+SRC 62	-	-	-	-
sasc	EFF+SFL 32	CHL+SAR 34	-	-	-
simple_spi	EFF+SFL 35	EFF+SRC 32	-	-	-
caprng	CHL+SRC 79	CHL+SRC 35	-	-	-
hilbert	CHL+SFL 92	CHL+SFL 32	CHL+SRC 39	-	-
systemcdes	CHL+SFL 48	CHL+SRC 77	CHL+SRC 39	-	-
des_area_opt	CHL+SFL 167	CHL+SRC 91	CHL+SRC 48	-	-
des3_area_opt	CHL+SFL 78	CHL+SRC 111	CHL+SRC 58	-	-
tv80	CHL+SFL 134	CHL+SFL 56	CHL+SFL 65	EFF+CYC 32	-
ac97_ctrl_top	CHL+SFL 512	CHL+SFL 152	CHL+SFL 182	EFF+SFL 32	-
usb_funct	CHL+SFL 333	CHL+SFL 115	CHL+SFL 122	CHL+SRC 46	-
aes_cipher	CHL+SFL 376	CHL+SFL 212	CHL+SFL 212	CHL+SRC 61	-
sha256core	CHL+SFL 409	CHL+SFL 512	CHL+SFL 82	CHL+SFL 32	-
des_perf_opt	CHL+SFL 509	CHL+SFL 204	CHL+SFL 332	CHL+SRC 82	-
aes_inv_cipher	CHL+SFL 512	CHL+SFL 240	CHL+SFL 272	EFF+SFL 49	-
des3_perf_opt	CHL+SFL 512	CHL+SFL 512	CHL+SFL 395	CHL+SFL 152	CHL+SAR 36
vga_lcd	CHL+SFL 512	CHL+SFL 512	CHL+SFL 512	CHL+SFL 117	CHL+SRC 43
b19	CHL+SFL 512	CHL+SFL 512	CHL+SFL 502	CHL+SFL 311	CHL+SRC 39
cfrca	CHL+SFL 512	CHL+SFL 512	CHL+SFL 512	CHL+SFL 181	CHL+SRC 77

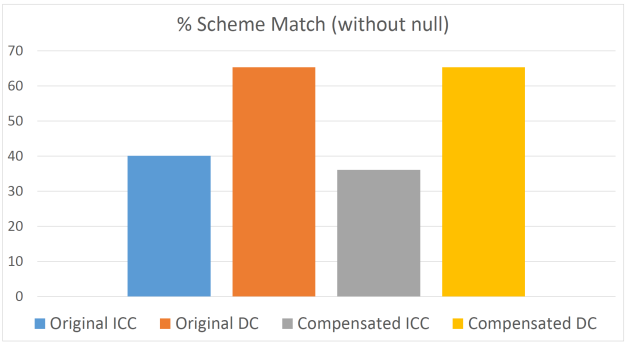


FIGURE 19. Percent of CDLE experiments that matched chosen encryption scheme when compared with the corresponding DC or ICC concurrent tree search results for the adjusted and original versions of the highest quality machine learning model. Experiments in which no encryption scheme could be found were ignored.

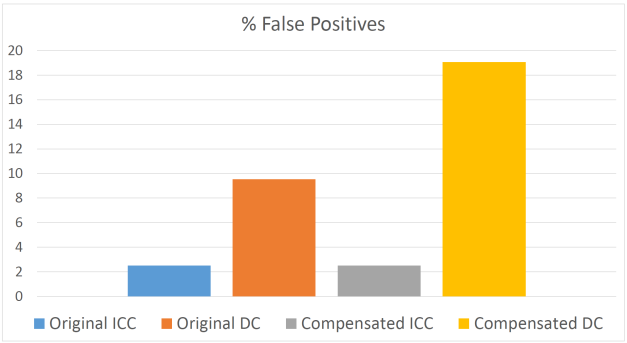


FIGURE 20. Percent of CDLE experiments in which no encryption strategy could be chosen for DC or ICC concurrent tree search results, but one was found for the machine learning model in question.

tion term.

After the identified adjustments were made to each regression in the interaction model using Cauchy cost, predictions were made for each test design again, as shown in Table 12 using ICC training data. As expected, the behavior of the machine learning model now more closely resembles that of the concurrent tree search. ChainLock and SFLL is still the most chosen scheme, but does not dominate the results as it did before the model was adjusted. Now, other schemes have been chosen as the PPA constraints are tightened. Additionally, the increasing amount of successful results with gate count much more closely resembles the concurrent tree search results for both ICC and DC estimations.

Adjusting the model shows overall behavior closer to the concurrent tree search, as seen by comparing Tables 11 and 12 to similar results from the concurrent tree search experimentation, such as Table 7. However, the metrics used do not confirm this. The comparisons of the adjusted, or compensated, model and the original model using these metrics are shown in Figures 18 – 20. The adjusted machine learning model showed some loss of accuracy in choosing

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3059163, IEEE Access

encryption schemes overall. More egregiously, the number of false positives almost doubled for the model using DC estimations. The ICC model, however, performed just as well after adjustment. By these metrics, the adjusted model did not show any improvement. However, further adjustment could improve the edge behavior of successful cases in choosing an encryption scheme. There are more cases in which ChainLock and SRCLock are chosen in the machine learning model, than in the concurrent tree search. These cases tend to happen on the edge of tight PPA constraints, but this edge of SRCLock cases is tighter in the concurrent tree search. Additionally, there are designs in the concurrent tree search which did not follow the trend of increasing gate count shown in this machine learning model. A more advanced machine learning model could take circuit structure into account in more detail than it was for this model.

VIII. CONCLUSIONS AND FUTURE WORK

In this work, a generalized Constraint-Directed Logic Encryption (CDLE) methodology was presented. Since the inception of logic encryption, research related to it has been driven by finding new potential attacks and developing new encryption methods with security metrics in mind. However, cost metrics have not been considered in the literature in general. A major issue in the field as it stands is implementing the technique in designs with stringent cost requirements, and one of the hurdles for achieving that is the lack of analysis of the PPA costs of adding logic encryption to a design. We have presented CDLE, a methodology developed to respond to the need for a holistic, extensible approach to applying logic encryption to any given design using known encryption methods in a way that responds to both security and cost requirements. Methods that fit in the CDLE framework converge on an encrypted design that optimally responds to both the designer's security goal and PPA constraints. Along with the methodology, we have proposed two example methods that follow the methodology, a concurrent tree search method and a machine learning approach.

The CDLE Concurrent Tree Search method explores the encrypted design space of a circuit in parallel by sampling encryption strategies and estimating their associated PPA cost using commercial off-the-shelf EDA tools. The algorithm requires a ranking of available encryption strategies and constraints the allowed PPA costs as percentages of that of the original design constrained the acceptable design space and returns the most optimally encrypted design based on the estimations made through the exploration. Experimentation on encrypted circuits using this strategy showed the expected speedup of increasing the number of computation threads. However, the returns of increasing thread count were diminishing. Therefore, this method cannot be significantly improved by the use of more threads. Additionally, relying on accurate PPA estimations using repeated place and route procedures lead to prohibitively long execution time for large designs. Using logic synthesis only to estimate PPA lead to a trade-off between execution time improvement and accuracy of PPA results. Therefore, the CDLE Concurrent Tree Search method can be extended to large, commercial-grade designs, especially if only logic synthesis is used for PPA estimation. Place and route PPA estimation can be used as well if more accurate results are desired, or if the design in question is small.

A machine learning approach to CDLE was also presented in this work in order to reduce the time of producing an encryption strategy while maintaining the level of accuracy of the Concurrent Tree Search method. A group of 22 designs were identified as a small training set, for which the PPA cost of implementing several logic encryption strategies was estimated. From these estimations, three candidate models were constructed from multivariable linear regression to estimate the PPA costs of different encryption schemes as functions of key size and gate count of the unencrypted design. Of the three candidate models, the one that produced the best results was an interaction model of the form $PPA = a + b\frac{k}{g}$ for key size k, gate count g, and power, performance, or area cost PPA. However, the model needed to be adjusted to recover similar behavior to that of the Concurrent Tree Search method for varying PPA constraints. Considering the very limited training design set and small amount of training done, the results are promising and some of the original behavior of the tree search method was able to be modeled. Therefore, the machine learning direction is a promising one for CDLE, although the process needs to be matured past the model used in this work.

FUTURE WORK

Since considering cost constraints for logic encryption is a new direction for the field, there is a lot of room for future work. CDLE is a methodology, not a single method for driving the application of logic encryption. Therefore many new methods can be developed under the CDLE umbrella. Improvements to the CDLE methodology itself could be made as well. Using commercial tools like Synopsys IC Compiler II and Design Compiler proved to have a trade-off between execution time and accuracy of cost metrics estimation. More lightweight solutions to estimating the cost of applying different logic encryption schemes to a design need to be considered. Such methods could be tied into logic synthesis, or perform RT-level analysis so no synthesis is required. The security metrics in the CDLE methods proposed in this work were assumed based off of theoretical attack resilience. A more robust CDLE approach could tie decryption attempts into the design-space exploration processes, using known state of the art attack methods. This would improve the trust of the result from CDLE.

In this work, a limited example of machine learning was shown. There is much room to grow in this direction in logic encryption, and the method would benefit greatly from a much more mature machine learning model and a larger training data set. More subtle model features should be considered for future models as only two features, key size and design gate count, were considered in this work.

Many other circuit features, such as logic depth or sequential element count, can effect the PPA impact of encryption and should be explored in future work. Though the models in this work were limited, they still showed promising results and machine learning should be considered as a direction for future work. More methods can be developed for CDLE that improve on the ones presented in this work, or take the methodology in new directions. The methodology is intended to be flexible and evolve with the field of logic encryption. More logic encryption methods driven by design constraints need to be explored for this hardware security method to transition to production, and CDLE provides a framework for these methods.

APPENDIX: ADDITIONAL CDLE CASE STUDIES

In the following appendix pages, several additional brief case studies are presented to supplement the results and case studies shown in this work. The case studies highlight edge behaviors of the concurrent tree search algorithm and interesting cases from the experimentation results. The figures follow the same format as previous case study sections. For all DC results, a layout was produced after the completion of CDLE using ICC2 and PPA was estimated from the layout for comparison with DC PPA estimations.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3059163, IEEE Access

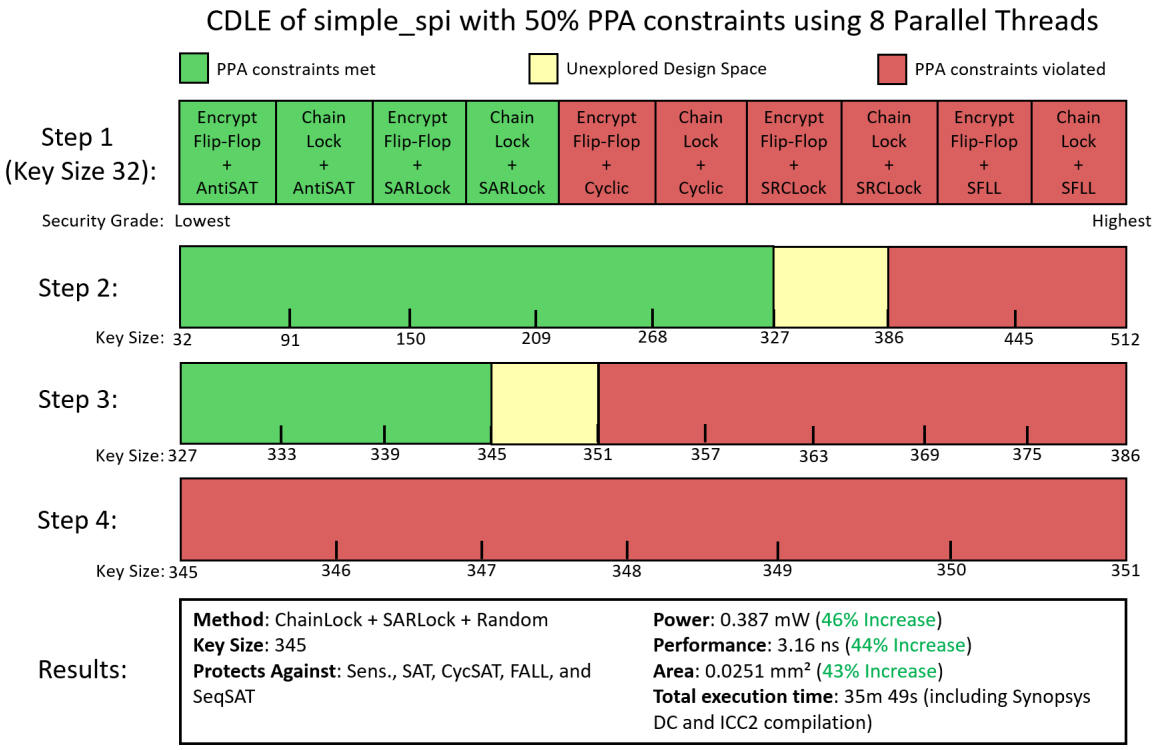


FIGURE 21. CDLE Trajectory and result of simple_spi with PPA constrained to 50% increases

A. CONCURRENT TREE SEARCH WITH ICC

1) Simple SPI

In this case, the Simple SPI design was encrypted using CDLE with PPA constraints of 50% each. The algorithm trajectory is shown in Figure 21. The chosen encryption strategy was ChainLock+SARLock with added Random encryption at key size 345. This shows a usual execution trajectory of the algorithm. In the final step, none of the tried key sizes met the PPA constraints. Therefore, the maximum accepted key size from the previous step was used as the best key size. The layout produced for the final encrypted design is in Figure 22. The algorithm executed in 35m 49s total, including all logic synthesis and layout place and route. The original and final PPA values estimated in this experiment were:

- Original Power: 0.265 mW
- Original Performance: 2.2 ns
- Original Area: 0.0176 mm^2
- Final Power: 0.387 mW (46% Increase)
- Final Performance: 3.16 ns (44% Increase)
- Final Area: 0.0251 mm^2 (43% Increase)

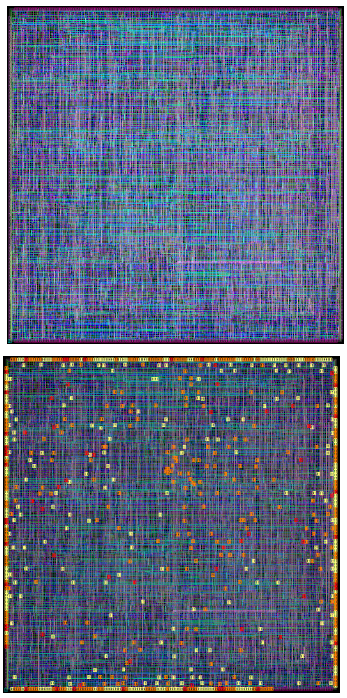


FIGURE 22. Layout of encrypted Simple SPI after CDLE. On right, the key inputs and their respective key gates are highlighted, yellow corresponding to ChainLock, orange to SARLock, and red to randomly placed gates.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3059163, IEEE Access

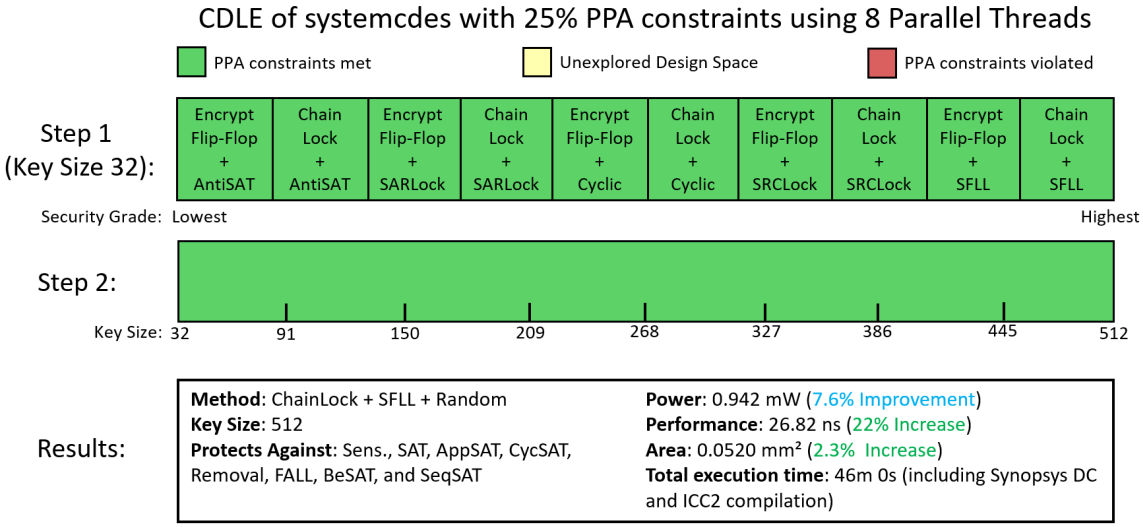


FIGURE 23. CDLE Trajectory and result of systemcdes with PPA constrained to 25% increases

2) SystemC DES

This trajectory is an example of the edge case in which the PPA constraints are loose enough to allow the maximum possible security provided by the possible encryption strategies. The algorithm trajectory is shown in Figure 23. In the first step, the encryption scheme that protected the most attacks was within the PPA constraints, and then in the second step, the maximum key size met the PPA constraints. Therefore, it was immediately accepted in this step. This way, the algorithm executes more quickly if the PPA constraints are loose compared to the cost of encryption. Therefore, the chosen encryption strategy was ChainLock+SFLL at key size 512. Additionally, power usage was improved by synthesis with encryption in this case. The layout produced for the final encrypted design is in Figure 24. The total execution time of the algorithm was 46m. The original and final PPA values were estimated at:

- Original Power: 1.02 mW
- Original Performance: 21.92 ns
- Original Area: 0.0508 mm^2
- Final Power: 0.942 mW (7.6% Improvement)
- Final Performance: 26.82 ns (22% Increase)
- Final Area: 0.0520 mm^2 (2.3% Increase)

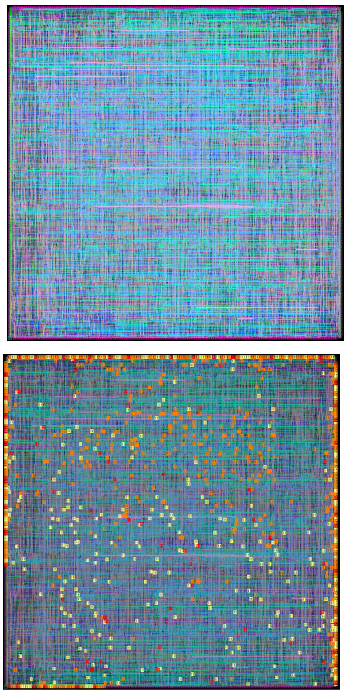


FIGURE 24. Layout of encrypted systemcdes after CDLE. On right, the key inputs and their respective key gates are highlighted, yellow corresponding to ChainLock, orange to SFLL, and red to randomly placed gates.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3059163, IEEE Access

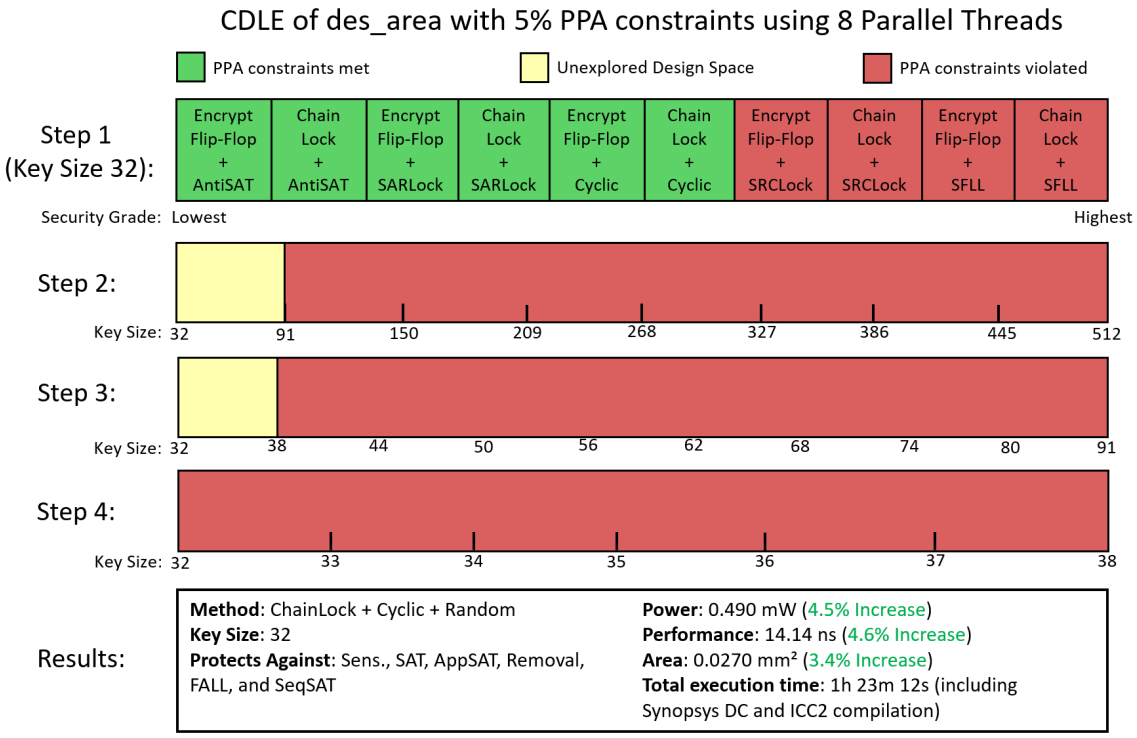


FIGURE 25. CDLE Trajectory and result of des_area with PPA constrained to 5% increases

3) Area Optimized DES

This case shows an edge case in which the algorithm executes successfully, but the final key size is k_{min}. The algorithm trajectory is shown in Figure 25. The encryption strategy chosen in the first step was ChainLock+Cyclic, and in the remaining steps, none of the sampled designs met the PPA constraints. Therefore, at each step, the unexplored design space shrunk at the small end of the key size space, until it collapsed to key size 32. Since 32 passed the PPA constraints in step 1, it was accepted as the best key size. This still happens in a normal amount of time compared to other key sizes. The layout produced for the final encrypted design is in Figure 26. The total execution time was 1h 23m 12s, and the estimated PPA values were:

- Original Power: 0.469 mW
- Original Performance: 13.52 ns
- Original Area: 0.0261 mm^2
- Final Power: 0.490 mW (4.5% Increase)
- Final Performance: 14.14 ns (4.6% Increase)
- Final Area: 0.0270 mm^2 (3.4% Increase)

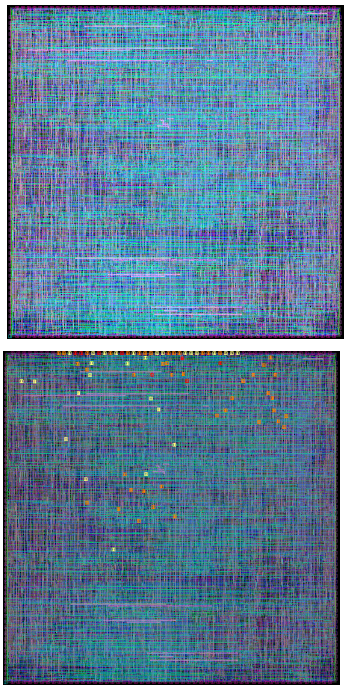


FIGURE 26. Layout of encrypted des_area after CDLE. On right, the key inputs and their respective key gates are highlighted, yellow corresponding to ChainLock, orange to Cyclic, and red to randomly placed gates.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3059163, IEEE Access

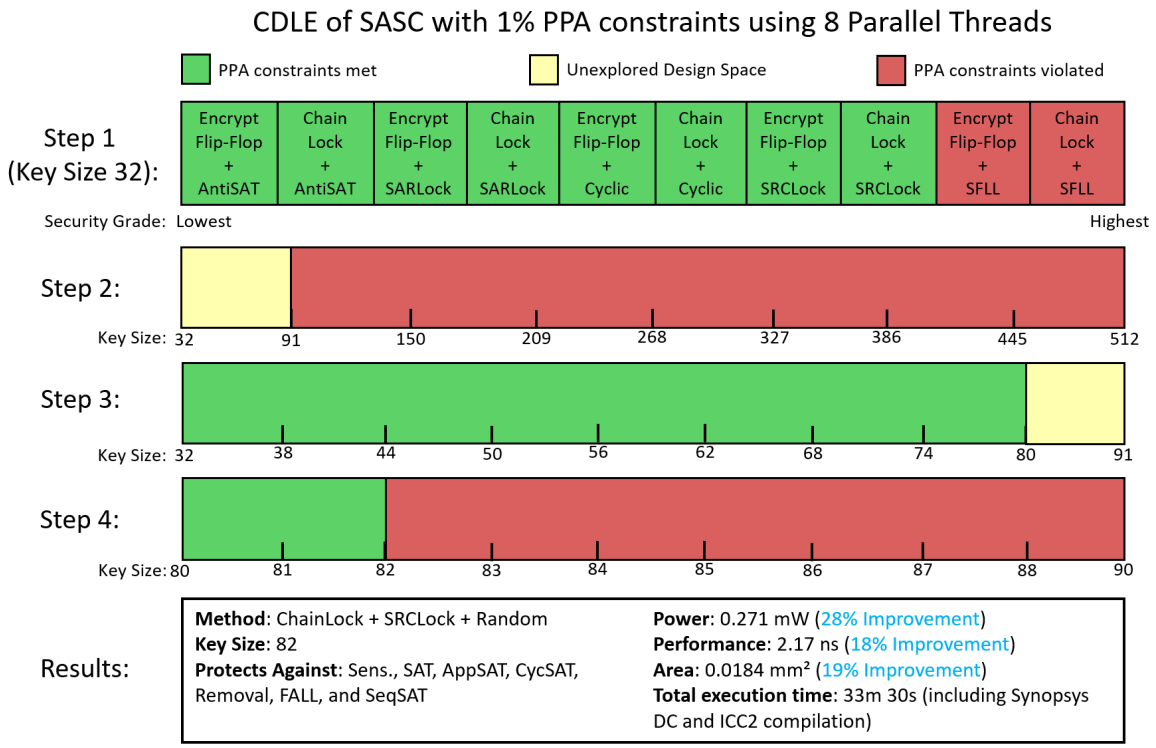


FIGURE 27. CDLE Trajectory and result of SASC with PPA constrained to 1% increases

4) SASC

The SASC CDLE experiments show peculiar behavior. The algorithm trajectory is shown in Figure 27 For one, the security metrics of the final designs found by the algorithm from 10% to 1% PPA constraints increase, which the opposite of the expected behavior, which is to decrease the security metrics and meet the PPA requirements. This can be explained by randomness in the power estimation with random test patterns. However, in this specific SASC case, all PPA were improved by synthesis. SASC is a simple design, containing only a 4 byte data reciever and FIFO. Therefore, even a slight optimization in a cost metric can make a large impact in the overall cost of the design. The final encryption scheme chosen in this case was ChainLock+SRCLock at key size 82, found in 33m 30s. The layout produced for the final encrypted design is in Figure 28. The PPA values estimated were:

- Original Power: 0.380 mW
- Original Performance: 2.65 ns
- Original Area: 0.0226 mm^2
- Final Power: 0.271 mW (28% Improvement)
- Final Performance: 2.17 ns (18% Improvement)
- Final Area: 0.0184 mm^2 (19% Improvement)

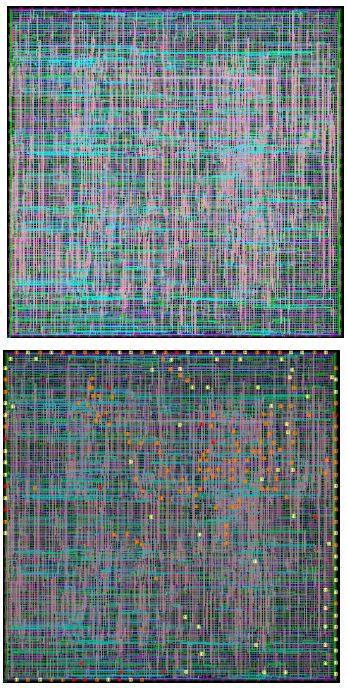


FIGURE 28. Layout of encrypted SASC after CDLE. On right, the key inputs and their respective key gates are highlighted, yellow corresponding to ChainLock, orange to SRCLock, and red to randomly placed gates.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3059163, IEEE Access

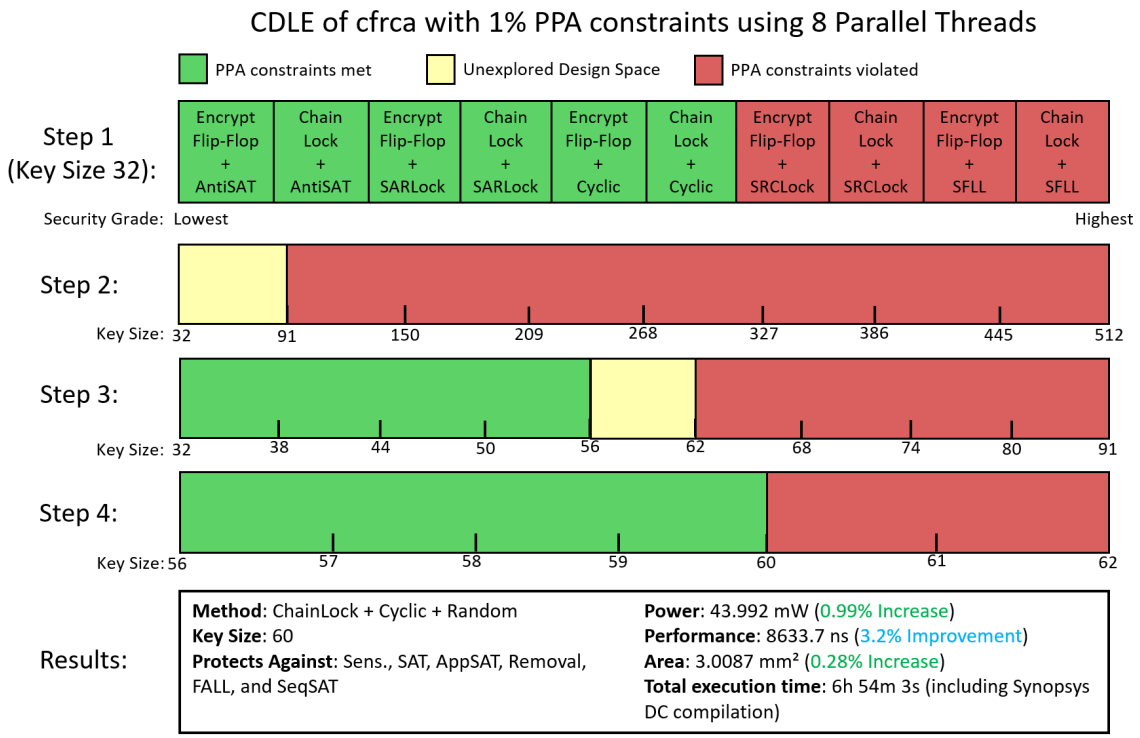


FIGURE 29. CDLE Trajectory and result of cfrca with PPA constrained to 1% increases

B. CONCURRENT TREE SEARCH WITH DC

1) CF RCA

This is a case of the largest design in the test set by gate count, encrypted using CDLE with 1% constraints for each power, performance, and area. The algorithm trajectory is shown in Figure 29. In the first step, ChainLock+Cyclic was chosen as the best encryption scheme within the PPA constraints. In the second step, none of the sampled key sizes were within the constraints, so the smallest key sizes were tested. By the final step, key size 60 was found to be the largest within the constraints. The algorithm was executed in 6h 54m 3s. So, even with only DC, the execution time can grow quickly for gates with large gate counts. The estimated PPA values in this experiment for the original and optimally encrypted designs were:

- Original Power: 43.561 mW
- Original Performance: 8920.9 ns
- Original Area: 3.0003 mm^2

- Final Power: 43.992 mW (0.99% Increase)
- Final Performance: 8633.7 ns (3.2% Improvement)
- Final Area: 3.0087 mm^2 (0.28% Increase)

The layout produced with the final encrypted design is shown in Figure 30. The PPA estimated from this layout showed significantly reduced power and area usage, and much improved performance compared to the DC PPA estimation.

- Layout Power – 21.0 uW
- Layout Performance – 185.28 ns
- Layout Area – 1.8723 mm^2

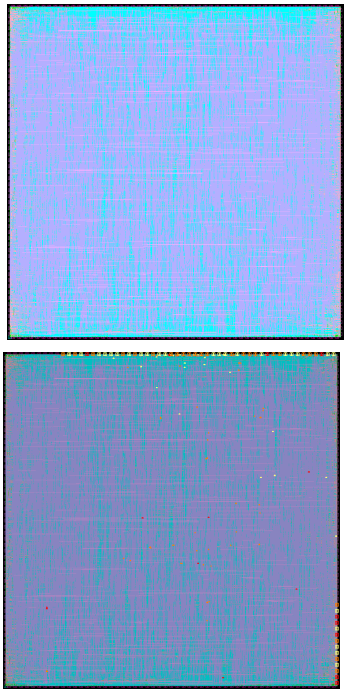


FIGURE 30. Layout of encrypted cfrca after DC-only CDLE. On right, the key inputs and their respective key gates are highlighted, yellow corresponding to ChainLock, orange to Cyclic, and red to randomly placed gates.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3059163, IEEE Access

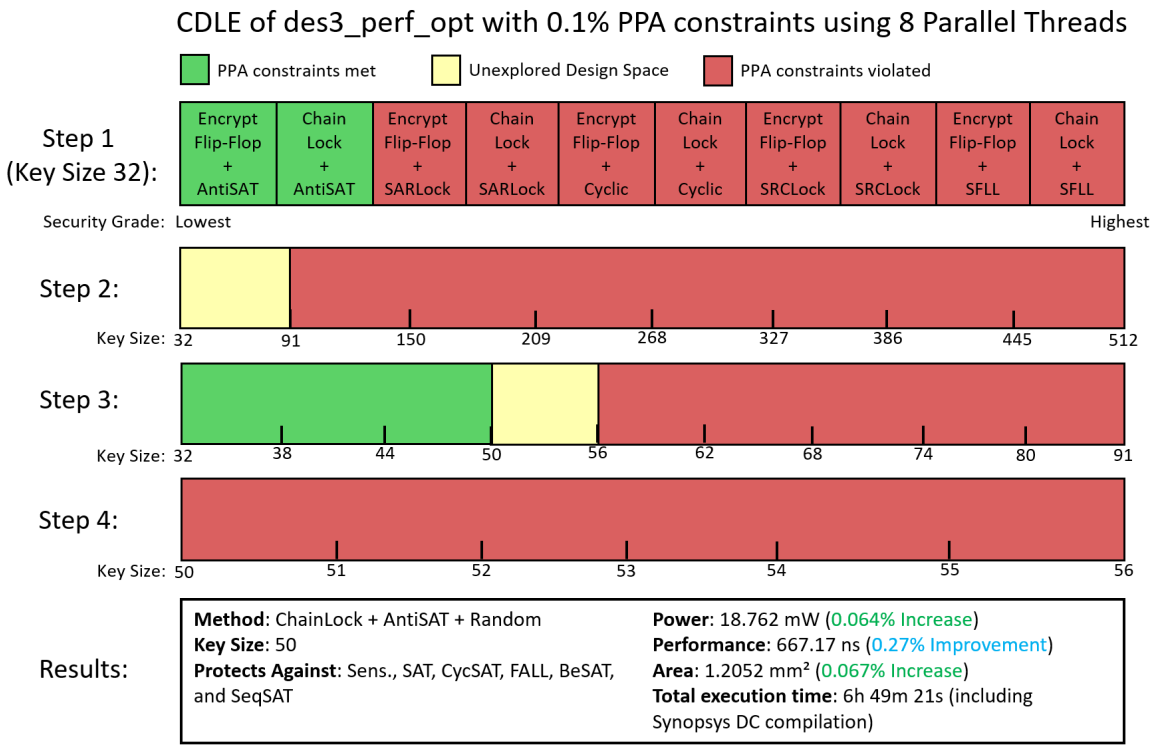


FIGURE 31. CDLE Trajectory and result of des3_perf with PPA constrained to 0.1% increases

2) Performance Optimized 3-DES

This is a case of one of the larger designs of the test set by gate count, encrypted with very tight PPA constraints of 0.1%. The algorithm trajectory is shown in Figure 31. As expected, an encryption strategy with weaker security metrics was selected in response. ChainLock+AntiSAT was chosen as the encryption scheme, which was rarely chosen in other experiments. The PPA constraints must be relatively strict, but not too strict, for this too happen. For large designs, this breakpoint is very close to 0. During the key size space search, 50 was the maximum key within the PPA constraints. The toal execution time was 6h 49m 21s, and the PPA values estimated for the original and encrypted designs were:

- Original Power: 18.750 mW
- Original Performance: 668.98 ns
- Original Area: 1.2043 mm^2

- Final Power: 18.762 mW (0.064% Increase)
- Final Performance: 667.17 ns (0.27% Improvement)
- Final Area: 1.2052 mm^2 (0.067% Increase)

The layout produced with the final encrypted design is shown in Figure 32. The PPA estimated from this layout showed increases in power and area, and much improved performance compared to the DC PPA estimation.

- Layout Power – 28.600 mW
- Layout Performance – 165.29 ns
- Layout Area – 1.5225 mm^2

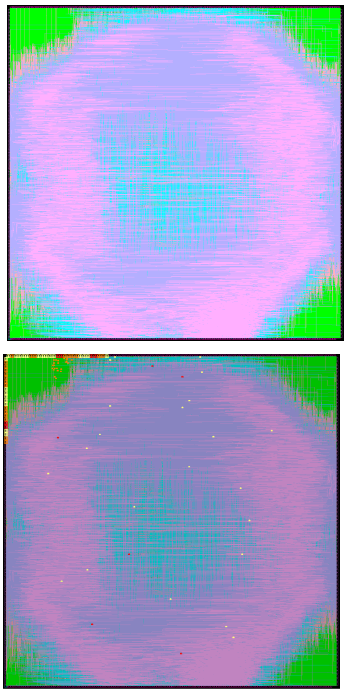


FIGURE 32. Layout of encrypted des3_perf after DC-only CDLE. On right, the key inputs and their respective key gates are highlighted, yellow corresponding to ChainLock, orange to AntiSAT, and red to randomly placed gates.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3059163, IEEE Access

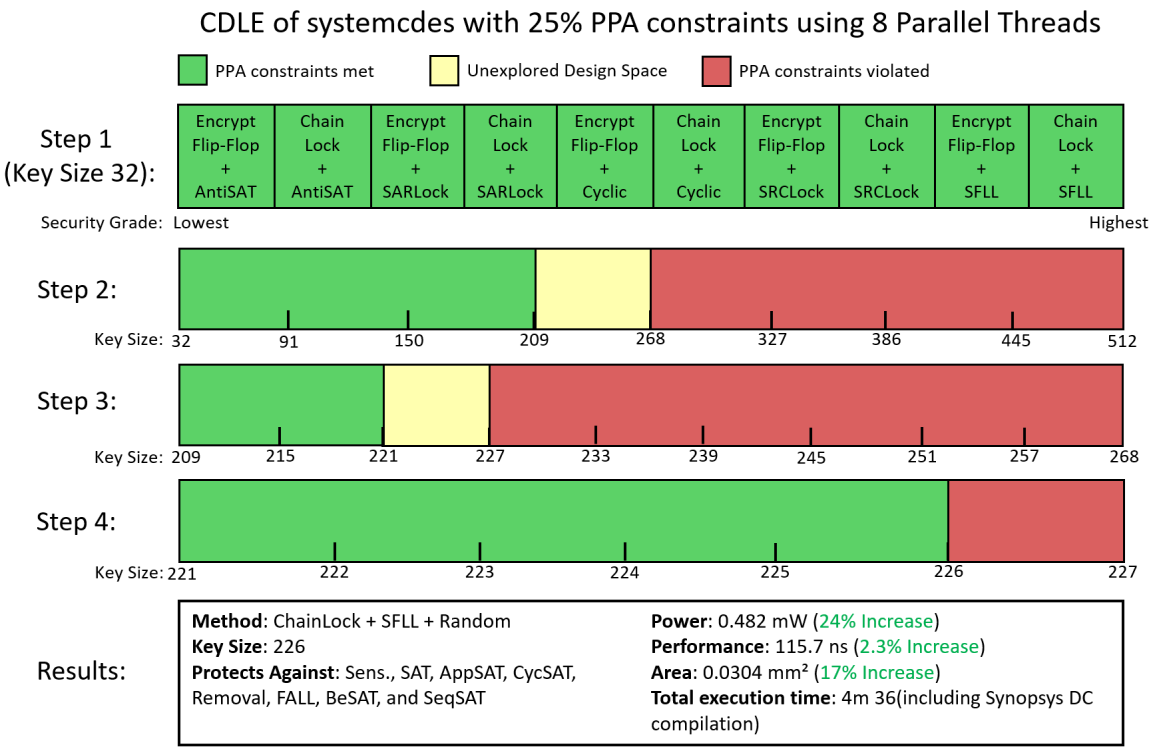


FIGURE 33. CDLE Trajectory and result of systemcdes with PPA constrained to 25% increases

3) SystemC DES

A comparison case study between this CDLE method using ICC and DC PPA estimations encrypts systemcdes with PPA constraints of 25%. The algorithm trajectory is shown in Figure 33. The corresponding experiment using systemcdes with ICC used ChainLock+SFLL at key size 256 for the best encryption strategy. In this case, the same encryption scheme was used, but at key size 226. The execution time with ICC estimation was 46m, while this case finished in 4m 36s. This is also considering that the ICC case executed in only two steps, choosing the maximum security encryption strategy quickly, while this case required 4 steps to close on a key size.

Like the case of des3_area, there was a large disparity between the DC and ICC PPA estimations for the unencrypted design, and therefore the encrypted design as well. For the unencrypted systemcdes, the power estimation is 61% smaller from DC compared to ICC, 416% slower in performance, and 49% smaller in area. The PPA values estimated during this experiment were:

- Original Power: 0.388 mW
- Original Performance: 113 ns
- Original Area: 0.026 mm^2

- Final Power: 0.482 mW (24% Increase)
- Final Performance: 115.7 ns (2.3% Increase)
- Final Area: 0.0304 mm^2 (17% Increase)

The layout produced with the final encrypted design is shown in Figure 34. The PPA estimated from this layout

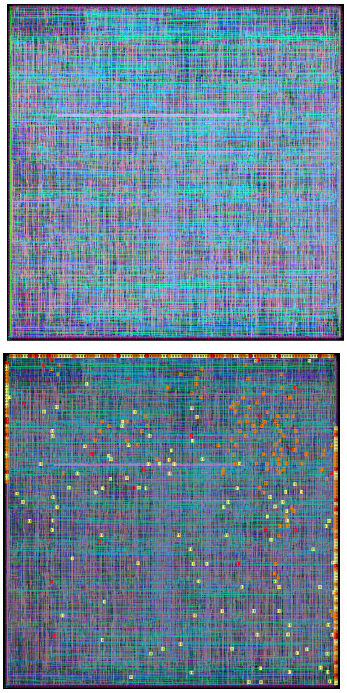


FIGURE 34. Layout of encrypted systemcdes after DC-only CDLE. On right, the key inputs and their respective key gates are highlighted, yellow corresponding to ChainLock, orange to SFLL, and red to randomly placed gates.

showed, as expected, a significant increase in power, increased area, and much improved performance compared to the DC PPA estimation.

- Layout Power – 0.753 uW
- Layout Performance – 25.78 ns
- Layout Area – 0.03949 mm^2

IX. ACKNOWLEDGEMENTS

We thank the CODES team made up of Y. Kasarabada, J. K. Meka, H. Chakraborty, N. Saxena, V. Muralidharan, and S. Venkatesh, for their work in implementing various software modules that apply encryption methods, and made CDLE possible. We specifically thank S. Venkatesh for her help in performing some of the CDLE experiments.

REFERENCES

[1] S. Dupuis and M. Flottes. Logic locking: A survey of proposed methods and evaluation metrics. Journal of Electronic Testing, 35:273–291, 2019.

[2] Jarrod A Roy, Farinaz Koushanfar, and Igor L Markovs. EPIC: Ending piracy of integrated circuits. In Proceedings of the conference on Design, automation and test in Europe, pages 1069—-1074, 2008.

[3] M. Pecht and S. Tiku. Bogus: electronic manufacturing and consumers confront a rising tide of counterfeit electronics. IEEE Spectrum, 43(5):37–46, 2006.

[4] IHS Technology Press Release. Top 5 most counterfeited parts represent a $169 billion potential challenge for global semiconductor industry. https://technology.informa.com/405654/top-5-most-counterfeited-parts-represent-a-169-billion-potential-challenge-for-global-semiconductor-market, 2012. Accessed: August 8, 2020.

[5] Masoud Rostami, Farinaz Koushanfar, and Ramesh Karri. A primer on hardware security: Models, methods, and metrics. Proceedings of the IEEE, 102(8):1283–1295, 2014.

[6] Randy Torrance and Dick James. The state-of-the-art in ic reverse engineering. In Christophe Clavier and Kris Gaj, editors, Cryptographic Hardware and Embedded Systems - CHES 2009, pages 363–381, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[7] Ujjwal Guin, Ke Huang, Daniel DiMase, John M. Carulli, Mohammad Tehranipoor, and Yiorgos Makris. Counterfeit integrated circuits: A rising threat in the global semiconductor supply chain. Proceedings of the IEEE, 102(8):1207–1228, 2014.

[8] Muhammad Yasin, Abhrajit Sengupta, Mohammed Thari Nabeel, Mohammed Ashraf, Jeyavijayan JV Rajendran, and Ozgur Sinanoglu. Provably-secure logic locking: From theory to practice. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, pages 1601–1618. ACM, 2017.

[9] David Luria, Yasaswy Kasarabada, and Ranga Vemuri. Constraint-directed logic encryption methodology. In Proceedings of GOMACTech 2020 Conference. GOMACTech, 2020.

[10] Muhammad Yasin, Bodhisatwa Mazumdar, Jeyavijayan Rajendran, and Ozgur Sinanoglu. SARLock: SAT attack resistant logic locking. IEEE International Symposium on Hardware Oriented Security and Trust (HOST), pages 1–1, 2016.

[11] Rajit Karmakar, Santanu Chatopadhyay, and Rohit Kapur. Encrypt Flip-Flop: A novel logic encryption technique for sequential circuits. arXiv preprint arXiv:1801.04961, 2018.

[12] Muhammad Yasin, Jeyavijayan JV Rajendran, Ozgur Sinanoglu, and Ramesh Karri. On improving the security of logic locking. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 35(9):1411–1424, 2016.

[13] Yang Xie and Ankur Srivastava. Anti-SAT: Mitigating SAT attack on logic locking. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 38(2):199–207, 2018.

[14] Yasaswy Kasarabada, Suyuan Chen, and Ranga Vemuri. On SAT-based attacks on encrypted sequential logic circuits. In 20th International Symposium on Quality Electronic Design (ISQED), pages 204–211. IEEE, 2019.

[15] Rajit Karmakar, Harshit Kumar, and Santanu Chattopadhyay. On finding suitable key-gate locations in logic encryption. In 2018 IEEE International Symposium on Circuits and Systems (ISCAS), pages 1–5. IEEE, 2018.

[16] Muhammad Yasin, Abhrajit Sengupta, Benjamin Carrion Schafer, Yiorgos Makris, Ozgur Sinanoglu, and Jeyavijayan Rajendran. What to lock? functional and parametric locking. In Proceedings of the on Great Lakes Symposium on VLSI 2017, pages 351–356, 2017.

[17] Yasaswy Kasarabada and Ranga Vemuri. StateLock: State transition based logic locking for sequential circuits. In 2020 33nd International Conference on VLSI Design and 2020 19th International Conference on Embedded Systems (VLSID). IEEE, 2020.

[18] Mohammad Saleh Samimi, Ehsan Aerabi, Zahra Kazemi, Mahdi Fazeli, and Ahmad Patooghy. Hardware enlightening: No where to hide your hardware trojans! In 2016 IEEE 22nd International Symposium on On-Line Testing and Robust System Design (IOLTS), pages 251–256. IEEE, 2016.

[19] Kaveh Shamsi, Meng Li, Travis Meade, Zheng Zhao, David Z Pan, and Yier Jin. Cyclic obfuscation for creating SAT-unresolvable circuits. In Proceedings of the on Great Lakes Symposium on VLSI 2017, pages 173–178. ACM, 2017.

[20] Shervin Roshanisefat, Hadi Mardani Kamali, and Avesta Sasan. SRCLock: SAT-resistant cyclic logic locking for protecting the hardware. In Proceedings of the 2018 on Great Lakes Symposium, pages 153–158. ACM, 2018.

[21] Jeyavijayan rajendran, Youngok Pino, Ozgur Sinanoglu, and Ramesh Karri. Security analysis of logic obfuscation. DAC '12: Proceedings of the 49th Annual Desgin Automation Conference, pages 83–89, 2012.

[22] Muhammad Yasin and Ozgur Sinanoglu. Evolution of logic locking. In 2017 IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC). IEEE, 2017.

[23] Pramod Subramanyan, Sayak Ray, and Sharad Malik. Evaluating the security of logic encryption algorithms. In Hardware Oriented Security and Trust (HOST), 2015 IEEE International Symposium on, pages 137–143. IEEE, 2015.

[24] Muhammad Yasin, Bodhisatwa Mazumdar, Ozgur Sinanoglu, and Jeyavijayan Rajendran. Removal attacks on logic locking and camouflaging techniques. IEEE Transactions on Emerging Topics in Computing, pages 1–1, 2017.

[25] Deepak Sirone and Pramod Subramanyan. Functional analysis attacks on logic locking. IEEE Transactions on Information Forensics and Security, 15:2514–2527, 2020.

[26] Kaveh Shamsi, Meng Li, Travis Meade, Zheng Zhao, David Z Pan, and Yier Jin. AppSAT: Approximately deobfuscating integrated circuits. In Hardware Oriented Security and Trust (HOST), 2017 IEEE International Symposium on, pages 95–100. IEEE, 2017.

[27] Hai Zhou, Ruifeng Jiang, and Shuyu Kong. CycSAT: SAT-based attack on cyclic logic encryptions. In Proceedings of the 36th International Conference on Computer-Aided Design, pages 49–56. IEEE Press, 2017.

[28] Yuanqi Shen, You Li, Amin Rezaei, Shuyu Kong, David Dlott, and Hai Zhou. BeSAT: behavioral SAT-based attack on cyclic logic encryption. In Proceedings of the 24th Asia and South Pacific Design Automation Conference, pages 657–662, 2019.

[29] Kimia Zamiri Azar, Hadi Mardani Kamali, Houman Homayoun, and Avesta Sasan. SMT attack: Next generation attack on obfuscated circuits with capabilities and performance beyond the SAT attacks. IACR Transactions on Cryptographic Hardware and Embedded Systems, pages 97–122, 2019.

[30] Kaveh Shamsi, Meng Li, David Z Pan, and Yier Jin. KC2: Key-condition crunching for fast sequential circuit deobfuscation. In 2019 Design, Automation & Test in Europe Conference & Exhibition (DATE), pages 534–539. IEEE, 2019.

[31] Yang Xie and Ankur Srivastava. Mitigating SAT attack on logic locking. In International conference on cryptographic hardware and embedded systems, pages 127–146. Springer, 2016.

[32] Seetal Potluri, Aydin Aysu, and Akasj Kumar. SeqL: Secure scan-locking for sequential circuits. https://eprint.iacr.org/2019/656, 2019. Accessed: 23-February-2020.

[33] Synopsys. Design compiler. https://www.synopsys.com/, 2018.

[34] Synopsys. IC compiler II. https://www.synopsys.com/, 2020.

[35] Icarus Verilog. Icarus verilog. http://iverilog.icarus.com/, 2015.

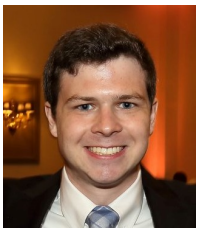
[36] Synopsys. Tetramax. https://www.synopsys.com/, 2018.

[37] Oliscience. Opencores IP cores. https://opencores.org. Accessed: March 15, 2020.

[38] Fulvio Corno, Matteo Sonza Reorda, and Giovanni Squillero. RT-level ITC'99 benchmarks and first ATPG results. IEEE Design & Test of computers, 17(3):44–53, 2000.

[39] Google. Go language, version 1.14.2. https://golang.org/, 2020.

[40] Altera Corporation. Altera advanced synthesis cookbook. https://github.com/thomasrussellmurphy/stx_cookbook. Accessed: August 27, 2020.
[41] Python Software Foundation. Python 3.6.3. https://www.python.org/, 2020. Accessed: September 15, 2020.
[42] SciPy Developers. Scipy. https://docs.scipy.org/doc/.
[43] David Luria. Logic encryption for resource constrained designs. Master's thesis, University of Cincinnati, 11 2020.



DAVID M. LURIA was born in Cincinnati, OH, USA in 1995. He received his B.S. in Physics and B.S. in Astrophysics from the University of Cincinnati in 2018 and his M.S. in Computer Engineering from the University of Cincinnati in 2020. In his M.S. studies, he was a Research Assistant at the Digital Design Environments Laboratory. He was a recipient of the UC Godown Family Fellowship in 2018. His research interests include hardware security, logic encryption, VLSI, and design automation. He now works as a Research and Development Software Engineer.

RANGA VEMURI has been on the faculty of Electrical and Computer Engineering at University of Cincinnati since 1989 and is currently a Professor. His interests span various topics within Hardware Security and Trust, Correctness and Security; VLSI Design and Architectures; Embedded Systems; Formal Methods and Formal Verification; Electronic Design Automation, Logic and Physical Synthesis; and Reconfigurable Computing. He and his students have published over 300 papers and received 13 Best Paper Awards or nominations. Dr. Vemuri graduated over 40 PhD and 80 MS students. His research has been funded by AFRL, DARPA, NSF, SRC, State of Ohio and various industries including EDAptive Computing Inc. to the tune of $12M over the past 25 years. He was an Associate Editor of the IEEE Transactions on VLSI and a Guest Editor of the IEEE Computer.