誤り修正論理合成を用いた論理暗号化手法

松永 裕介[1,a)]

概要：本稿では論理 IP の剽窃や盗用を防ぐための論理暗号化手法およびそれに対する攻撃手法について概観した上で，現在のところ最も強力な攻撃手法と考えられている SAT 攻撃アルゴリズムに耐性を持つ論理暗号化手法である TTLock の紹介を行う．TTLock の暗号化を実現する上での論理合成における課題を提示したうえで，誤り修正手法に基づく論理暗号化手法について考察する．

A logic encryption method using error correction logic synthesis

YUSUKE MATSUNAGA[1,a)]

Abstract: This paper briefly explains logic encryption algorithms, which prevent IP piracy, and also describes its attack methods. TTlock algorithm, which is known to be robust against the most effective attack method 'SAT attack' is then introduced. The problem on implementing TTLock encryption algorithm using logic synthesis technique is shown, and finaly implementing issue is discussed, which is based on error correnting logic synthesis algorithms.

Keywords: logic encryption, SAT-attack, error correction logic synthesis

1. はじめに

大規模な LSI をゼロから設計することは極めて稀であり，多くの場合は IP(Intellectual Property) と呼ばれるすでに設計された部品を組み合わせ，足りない部分だけを新規設計する手法が用いられている．IP にはレイアウト (マスクパタン) レベル，ゲートレベル，RT レベルなど様々な抽象度のものがあり，抽象度が高いほどその適用可能範囲も広い．一方，抽象度の高い IP は可読性が高く，また改変が容易であることから剽窃や盗用，改変などの著作権侵害が容易に行えるという欠点を持つ．そのため，悪意のある半導体メーカー (あるいは半導体設計者) が他社から IP を取得し，それをあたかも自分の知財であるかのように装って販売もしくは提供することが可能である．マスクパタンレベルの IP には画像として「透かし」(watermark) を入れるなどの手段で盗用を検知することが可能であるが，前述の様に論理 IP の場合は比較的容易にその機能を変えることなく内容を改変することが可能であるため，ゲートレベルや RT レベルの記述に透かしを入れておいたとしても除去されてしまう可能性が高い．

このようなゲートレベルあるいは RT レベルの論理 IP に対する著作権保護の一手法として論理暗号化 (logic enctyption) 手法が提案されている [3]．論理暗号化とは元の論理回路に「鍵入力」と呼ばれる外部入力信号線を追加し，鍵入力に正しい値が与えられた時のみ回路全体が正しく動作するように回路を改変する手法のことである．この論理 IP を入手したとしても鍵入力の正しい値を知らなければ論理 IP を動作させることはできないため，鍵入力を適切に管理することで論理 IP を保護することが可能となる．しかし，論理暗号化に関しては未だ研究・評価が十分ではなく，暗号化手法だけでなく，その暗号化が堅牢であるかどうかの評価手法も未だ定まっているとは言い難い．本稿では，論理合成技術の応用で誤り修正を行う手法を用いて堅牢な論理暗号化が可能かどうかを評価する手法について検討を行う．

以下，2 節で論理暗号化手法および攻撃方法 (評価手法) について説明し，つづく 3 節で従来の攻撃方法に耐性のある論理暗号化手法 (TTLock) の紹介を行う．4 節では TTLock

[1] 九州大学大学院システム情報科学研究院
〒 819–0395 福岡市西区元岡 744
[a)] matsunaga@ait.kyushu-u.ac.jp

の問題点に関して考察を行い，5 節で論理合成の誤り修正技術の観点からその暗号化手法の実現可能性について述べる．

2. 論理暗号化とその攻撃方法

論理暗号化の概念を図 1 に示す．図 1(a) が元の回路であり，暗号化を施した回路が同図 (b) である．暗号化された回路には新たな入力 $K = (k_1, k_2, \ldots)$(鍵入力と呼ぶ) が追加されており，これらは耐タンパ性 (tamper-proof) を持ったメモリと接続される[*1]．鍵入力 K に正しい値が入力された時のみ暗号化された回路は元の回路と等価な動きをするように設計される．このようにすることで，暗号鍵を保持するメモリチップという物理的な部品で論理 IP の動作をロック (lock) することが可能となり，論理 IP のデータをコピーするだけでは回路を動作させることができなくなる．

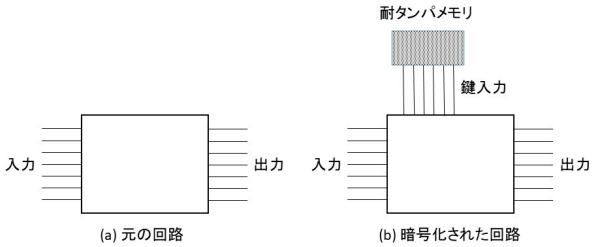


図 1 論理暗号化の概念図

論理暗号化の実際の手法はいくつか提案されている [2], [3], [5]．論理暗号化が提案された当初は暗号化の鍵を探索するアルゴリズムが未知であったため，暗号化の堅牢性を定量的に表す適切な尺度がなかった．その後いくつかの攻撃方法が提案され，特に文献 [4] で提案されている SAT ソルバを用いた攻撃アルゴリズムを用いると既存の暗号化手法の殆どが脆弱であり暗号鍵が容易に特定できることが示された．この攻撃アルゴリズムの概要は以下のとおりである．なお，暗号化された回路の入出力の関係を表す論理関数を $F^{enc}(X, K)$ とする．ここで，X はもともとの回路の入力の値を表す変数，K は鍵入力の値を表す変数である．また，暗号化される前のもとの回路の入出力の関係を表す論理関数を $F^{ref}(X)$ とする．

(1) 適当な方法で (他の攻撃手法を用いても良い) 一つの入力パタン X_0 を求める．

(2) その入力パタンのもとで正解 (攻撃者は正解の出力を知ることができるものと仮定する) を出力する鍵入力の値 (K_1 とする) を SAT ソルバを用いて求める．つまり，$F^{en}(X_0, K_1) = F^{ref}(X_0)$ を満たす K_1 を求める．

(3) X_0(および同様に求められた他の入力パタン) に対しては K_1 と同様の出力を返し，かつ，ある入力 X_1 に対しては K_1 と異なる出力を返すような鍵入力の値 K_2 が存在するかどうかを SAT ソルバを用いて求める．

(4) そのような鍵入力が存在しない場合には K_1 が正解として出力して終わる．

(5) 存在する場合，X_1 を鍵の条件の入力パタンに加える．

(6) 2. へ戻る．

このアルゴリズムの肝は限られた数の入力パタンのもとで，正解と一致する鍵入力の値を求めているところにある．直感的には別の入力パタンのもとではその鍵入力の値が正解とはならない場合があるように思われるが，ステップ (4) において条件を満たす鍵入力の値がただ一つに定まっている場合にはそれ以外に正解の可能性はない．そこで，アルゴリズムを終了し，求められた鍵入力の値を解として出力する．一方，ステップ (5) でそのような鍵入力の値が 2 つ以上あった場合. その 2 つの鍵入力に対して X_1 を入力すると異なる出力が得られることになる．その場合，どちらかの鍵入力は正解ではないことがわかるので，X_1 を鍵条件を作るための入力パタン集合に追加する．SAT 攻撃で以上の処理を繰り返し行うが，ステップ (5) の X_1 を用いた時に正解でないと判明する鍵入力の値が最悪でも 1 つあるので，この繰り返しはいつかは必ず停止する．

基本的に SAT 問題は $\mathcal{NP}$ 完全であり最悪の場合は問題のサイズ (式中の変数の数 ∼ 回路規模) の指数に比例した手間がかかることが知られているが，論理回路の入力値を求めるような SAT 問題は比較的高速に解を求めることができる．そのため，上記のアルゴリズムの効率は鍵入力の候補を 1 つに減らすまでに必要な繰り返し数が多いか少ないかに依存する．実際，文献 [4] によれば既存の論理暗号化手法の多くはこの攻撃アルゴリズムによって比較的容易に暗号鍵の特定が行えることが示されている．

3. SAT 攻撃に耐性を持つ論理暗号化手法

文献 [5] では SAT 攻撃アルゴリズムを用いても殆どの場合に鍵入力のビット数の指数に比例した回数の繰り返しを必要とする暗号化アルゴリズムを提案している (TTLock).

図 2 に TTLock の暗号化の概念図を示す．まず，鍵入力のビット数を k とし，鍵の値を $P = p_1, p_2, \ldots, p_k$ とする．また，元の論理回路 $\mathcal{C}$ の外部入力のうち k 個を選びだしそれを $X = x_1, x_2, \ldots, x_k$ とする．TTLock による暗号化回路は，変更された論理回路 $\mathcal{C}'$ と復元回路 (restore logic) と XOR ゲートから構成される．$\mathcal{C}'$ は入力 X の値が $x_1 = p_1, x_2 = p_2, \ldots x_k = p_k$ の時だけ元の回路 $\mathcal{C}$ の出力値と異なるように変更を行う．一方，復元回路は鍵入力の各ビットと $X = x_1, x_2, \ldots, x_k$ の値を比較し，全てのビットが等しい時のみ '1' を出力する．もしも鍵入力に正しい値

[*1] 鍵入力の提供方法は他にもさまざまな実装方法が考えられるが本稿の対象ではないので詳細は省略する．

が与えられなかった場合，その値を $Q = q_1, q_2, \ldots, q_k$ とすると，この暗号化回路は $X = P$ の時には $\mathcal{C}'$ が誤った値を出力し，復元回路が $X \neq Q$ であるので '0' を出力するため，全体としても誤った値を出力する．また，$X = Q$ の時には $\mathcal{C}'$ は正しい値を出力するが，復元回路が '1' を出力するので全体としてやはり誤った値を出力する．鍵入力に正しい値 $(= P)$ が与えられた場合，$X = P$ の時に $\mathcal{C}'$ は誤った値を出力するが，復元回路も '1' を出力するので全体として正しい値となる．$X \neq P$ の時は $\mathcal{C}'$ が正しい値を出力し，復元回路は '0' を出力するのでやはり全体として正しい値を出力する．

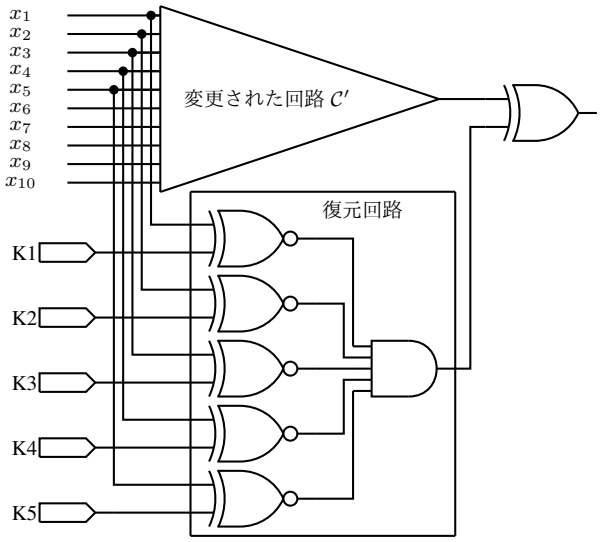


図 2　TTLock 暗号化回路の概念図

この暗号化方式で暗号化された回路に前述の SAT 攻撃アルゴリズムを適用することを考える．今，説明を簡単にするために元の回路の外部入力数も鍵入力のビット数 k と等しいものと仮定する．表 1 に鍵入力の値 (K_i) と外部入力の値 (X_j) による暗号化回路の出力応答を示す．表中の ○は正しい出力が得られることを示しており，×は誤った出力が得られることを示している．ここで正しい鍵の値は

表 1　TTLock の暗号化結果

	K_1	K_2	K_3	K_4	K_5	$\cdots$
X_1	○	×	×	×	×	$\cdots$
X_2	○	×	○	○	○	$\cdots$
X_3	○	○	×	○	○	$\cdots$
X_4	○	○	○	×	○	$\cdots$
X_5	○	○	○	○	×	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

K_1 とする．明らかに鍵入力に K_1 が入力された場合にはどのような外部入力 X に対しても回路は正しい出力を返す．しかし，それ以外の鍵入力の値の場合も 2 つの入力値 (前述の説明における P と Q) 以外は正しい出力を返す．さらに，そのうちの 1 つは正しい鍵に対応した入力 (表中の X_1，前述の説明における P) であるが，残りの 1 つはそれぞれの鍵入力の値に対応しているため，それぞれ異なっている (前述の説明における Q)．もしも SAT 攻撃アルゴリズムのステップ 1 において幸運にも X_1 を選ぶことができればこの条件を満たしているのは K_1 だけなので 1 回で正解を求めることができる．しかし，それ以外の値を選んだ場合には唯一つの鍵候補を除外することしかできない．たとえば X_2 を入力として選んだ場合，K_2 が正解でないことがわかるだけで残りの鍵候補は依然として候補のままである．SAT 攻撃アルゴリズムのステップ 4 において 2 つの鍵候補の出力が異なるような入力を求めているが，この時 2 つの鍵候補に K_1 が含まれていれば $\frac{1}{2}$ の確率で X_1 が選ばれるが，2 つの鍵候補の一つとして K_1 が選ばれる確率は $O(\frac{1}{2^k})$ である．このように，SAT 攻撃アルゴリズムでは正しい鍵を特定するために $O(2^k)(k$ は鍵入力のビット数) のパタンを必要とするため，$k = 64$ 程度の鍵長であればほぼ現実的な時間では解読することは難しい．

4. TTLock の問題点

4.1 回路構造からの情報漏えい

前節で紹介した TTLock は SAT 攻撃に対する耐性という意味で優れているように思われる．果たしてそれは本当だろうか？もしも攻撃者が表 1 における入力 X_1 を何らかの方法で知ることができたら，わずか一回の SAT 処理で正しい鍵を解読することが可能である．文献 [5] ではそのような入力を知る術はない，と論じているが変更された論理回路 $\mathcal{C}'$ をどのようにして合成するかを具体的にしなければ，回路の構造から鍵情報が漏洩する可能性がないとは言い切れない．図 3 に単純な回路変更を用いた TTLock の論理暗号化の例を示す．この例では $\mathcal{C}'$ を $\mathcal{C}$ の出力と X と P の

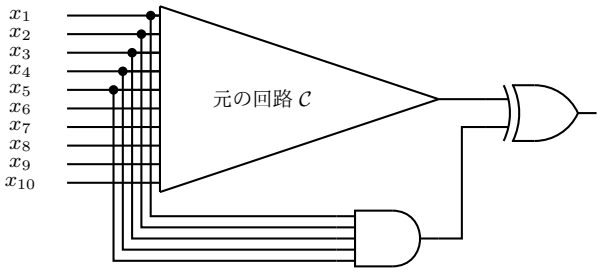


図 3　単純な回路変更

比較結果との XOR ゲートで作っている．P は '0' か '1' の定数のビットベクタなので X と P の比較結果は実際には k 入力の (巨大な)AND ゲートで実現される[*2]．これではこの回路構造をトレースするだけで鍵が P であることがすぐに明らかになってしまう．図 3 中の XOR ゲートの 2 番めの入力となっている信号線が '1' になる時の入力の値が鍵

[*2]　この例では鍵の値 $P = (1, 1, \ldots, 1)$ の場合を示している．

の値に等しいからである．このような鍵情報の漏洩を防ぐためには C' の作り方を工夫する必要があるが，回路全体を真理値表や積和形論理式で表して合成する以外でただ 1 つの入力に対してだけ出力を反転させる回路を作ることは難しい．文献 [5] ではこの点に関しては全く具体的な方法には触れないまま，無責任に既存の論理合成技術を使えばうまく行くとだけ書いている．この点に関してより具体的な観点で考察を行う必要があると思われる．

4.2 多出力回路への一般化の問題

今までは話を簡単にするために対象の回路を 1 出力と仮定してきたが，実際には多出力回路を扱う必要がある．一般に，多出力の回路であっても各出力ごとに独立した回路構造を持つことはまれで，内部の中間論理を表す部分回路は複数の出力で共有されている場合が多い．そのため，特定の入力パタン (= P) においてある出力の値だけを反転させるような回路変更を行うことは極めて難しい．それを避けるためにはもとの回路からある出力に関係する部分回路を抜き出して複製したあとで，その複製した回路に対して回路変更を行うやり方が考えられるが，複製した回路の面積オーバーヘッドが増える問題がある．また，複製元の回路との差分を調べることで回路変更に関するヒントが漏れるという問題もある．

5. TTLock 用の論理合成のアプローチ

5.1 極大ファンアウトフリーコーンに基づく回路分割

以上を踏まえ，既存の論理合成技術で対応可能な TTLock の実装方法について検討を行う．詳細な議論の前にいくつかの用語の定義を行う．

組み合わせ論理回路は内部にフィードバックループを含まないので，その構造は DAG(Directed Acyclic Graph) で表される．回路中の各ゲートが節点に対応し，ゲート間の接続 (ネット) が枝に対応する．ゲート v を始点，ゲート u を終点とする枝 $e = (v, u)$ が存在するとき，ゲート v をゲート u の**ファンイン**と呼び，ゲート u をゲート v の**ファンアウト**と呼ぶ．ゲートの順序列 $v_1, v_2, \ldots, v_n$ に対して，隣接するすべてのゲート間 $v_i, v_{i+1} : 1 \leq i \leq (n-1)$ に対応する枝 $e = (v_i, v_{i+1})$ が存在する場合，そのようなゲートの順序列を**経路**と呼ぶ．ゲート v からいずれかの外部出力へ至るいかなる経路もゲート u を含むとき，ゲート u をゲート v の**ドミネータ**と呼ぶ．今，共通のドミネータ u を持つようなゲート v_i の集合を考える．詳細は省略するが，そのようなゲートの集合は u を出口とする連結した部分回路を構成する．このような部分回路を**ファンアウトフリーコーン (Fanout Free Cone: FFC)** と呼ぶ．ファンアウトフリーコーンの出口のゲートをファンアウトフリーコーンの**根**と呼ぶ．また，自身はそのファンアウトフリーコーンに属さずに，そのファンアウトフリーコーン内のゲートにファンアウトしているゲートをそのファンアウトフリーコーンの**葉**と呼ぶ．ファンアウトフリーコーンの葉の要素数をそのファンアウトフリーコーンの**サイズ**と呼ぶ．自分自身以外には自身を含むより大きなファンアウトフリーコーンを持たないファンアウトフリーコーンを**極大ファンアウトフリーコーン (Maximal Fanout Free Cone:MFFC)** と呼ぶ．こちらも詳細は省略するが，すべてのゲートは正確に唯一の MFFC に属している．つまり，もとの組み合わせ回路は MFFC に従って重複なく分割される．

多出力回路に関する問題を避けるため，TTLock の暗号化を回路全体に対して適用するのではなく，MFFC に対して適用することを考える．前述の様に MFFC は唯一つの出口（ドミネータ）を持つ．そこで，このドミネータへ至る部分回路を変更し，最後に復元回路を付加することで MFFC 単位でオリジナルの回路と等価な回路を構成する．こうすることによって多出力回路に関する問題を回避することができる．一つの MFFC でそのサイズに等しい鍵入力を持った暗号化回路を作ることができる．

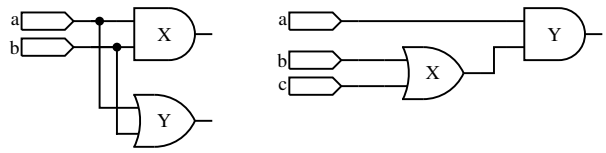
5.2 極大ファンアウトフリーコーンの選択

MFFC のサイズは回路構造によって千差万別であるが，数十のものは滅多になく，多くが 10 以下である．そのため，複数の MFFC を選択して全体で必要な鍵入力の数を満たすようする必要がある．ここではその MFFC の選択方法に関して考察を行う．また，同時にその MFFC の回路変更の仕様 (前述の C' に対する変更パタン P) の決定方法についても検討する．

大まかにはサイズの大きな MFFC を選べば良いと思われるが，回路構造に起因するドントケア条件を考慮する必要がある．組み合わせ回路の外部入力の場合，任意の値にセットすることは可能であるが，組み合わせ回路の内部ゲートの場合，他のゲートの値との兼ね合いで自由には設定できない場合がある．例えば図 4(a) の回路においてゲート X, Y の出力は $(1, 0)$ になることはない．これは $X = a \wedge b$ より，X を 1 にするためには a, b ともに 1 でなければならず，すると $Y = a \vee b$ より Y の 1 になってしまうためである．このように複数の内部ゲートの値の取りうる組み合わせの偏りを充足性ドントケア (Satisfiable Don't Care) と呼ぶ．

また，回路構造によっては内部ゲートの値の変化の影響が外部出力まで伝搬しない場合がある．例えば図 4(b) の回路において $a = 0$ のときは $Y = 0$ となって X の値の影響を受けない．このように，他の信号線の値によって自身の値の変化が外部出力の値に影響を与えなくなることを観測性ドントケア (Observability Don't Care) と呼ぶ．

さて，ある MFFC に着目した場合，その葉のゲート間に充足性ドントケアがある場合があり，また，その根のゲートの値が観測されない観測性ドントケアがある場合もあ



(a) SDC の例　　(b) ODC の例

図 4　ドントケアを生じる回路の例

る．そのようなドントケアに含まれている値に関して回路変更を行ってもその変更は外部に影響を与えないので暗号化の効果がなく，誤った鍵でも正しく動いてしまう．そこで，MFFC の回路変更に用いるパタンはその影響が必ず外部出力で観測されるものでなければならない．これは表 1 における X_1 に該当する．一方，正しい鍵でない場合のパタンもドントケアで観測できなければ SAT 攻撃の際に考慮する必要がないので SAT 攻撃を容易にしてしまうことになる．このように MFFC の選択および回路変更に用いるパタンを選択する際には回路構造に起因するドントケア条件を考慮する必要がある．

とはいえ，回路各部のドントケア条件を正確に計算し，記憶することは現実的ではない．そこで，ランダムパタンのサンプリングを用いる．ここで，各 MFFC の葉のゲートの値の組み合わせ (パタン) のうち，その値の影響が外部出力に伝搬するものを**有効パタン**と呼ぶことにする．以下のアルゴリズム (1) ではランダムサンプリングを行って有効パタンの計算を行う．

input : 暗号化対象の回路 $\mathcal{G}$
output: 各 MFFC の有効パタンの集合
while 十分な回数 **do**
  $X \leftarrow$ ランダムに生成した入力パタン;
  X を入力として $\mathcal{G}$ の論理シミュレーションを行う．;
  for $\mathcal{G}$ の *MFFC* m **do**
    m の根のゲートの値の変化が外部出力で観測されるか調べる (故障シミュレーション);
    if 観測された **then**
      そのときの葉のゲートの値の組み合わせを有効パタンとして記録;
    end
  end
end

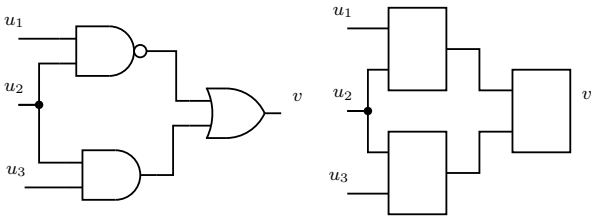
Algorithm 1: 有効パタンの計算アルゴリズム

上記のアルゴリズムで求めた有効パタンが多い MFFC を暗号化することでそれだけ多くの鍵パタン候補を持つことになる．

5.3　誤り修正論理合成

暗号化対象の MFFC および回路変換に用いる入力パタンが選択されたら，実際に回路変換を行う必要がある．通常，MFFC のサイズはそれほど大きくないので，回路変換後の仕様 (論理関数) に基づいて積和形論理式の最適化やファクタードフォームの最適化などの通常の論理合成を行ってもよいが，もとの回路に比べて面積や遅延時間などが大幅に変わってしまう可能性がある．そこで，誤り修正 (error correction) 論理合成の手法を用いることを考える．これは論理回路 $\mathcal{C}$ と満たすべき仕様 $\mathcal{S}$ が与えられた時に，もとの論理回路 $\mathcal{C}$ の構造をなるべく再利用しながらその機能 (論理関数) が $\mathcal{S}$ と等しくなるように回路を変更するものである．ある意味 TTLock の回路変更はこの誤り修正の一種とみなすことができる．

誤り修正論理合成の具体的な手法はいくつか考えられるがここでは SAT ソルバを用いた手法を紹介する．まず，回路変更後の論理関数を F とする．つぎに対象の論理回路の各ゲートを同じ入力数の LUT(Look-Up Table) に置き換えた仮想的な回路 G を考える (図 5)．k 入力 LUT は任意の k 入力論理関数を実現することが可能なので，各 LUT を適切に設定することで全体で F という論理関数を実現することができれば，もとの回路構造を維持したまま回路変更を行えることとなる．もちろん，同じ入力数のゲートでも機能が異なれば面積や遅延時間は異なるが，回路全体としてはそれほど大きな差にはならないと考えられる．与えられた論理関数を実現するための LUT の設定を求める問題は複合型 LUT のブーリアンマッチング問題と同様である [1].



(a) 回路例　　(b) LUT への置き換え

図 5　回路変更の例

今，各 LUT のコンフィグレーションビット (LUT の各エントリの値) を $c_1, c_2, \ldots, c_m$ とし，MFFC の入力 (葉) の値を表す変数を $x_1, x_2, \ldots, x_n$ とする．すると，複合型 LUT のブーリアンマッチング問題とは論理式 (1) を満たす値の組み合わせを求める問題となる．

$$\exists c_1, c_2, \ldots, c_m \forall x_1, x_2, \ldots, x_n$$
$$G(x_1, x_2, \ldots, x_n, c_1, c_2, \ldots c_m) = F(x_1, x_2, \ldots, x_n) \quad (1)$$

ここで，G および F は実際には複数の CNF 式で構成されている[*3]．式 (1) は限定作用子付き論理式でそのままでは SAT ソルバで解くことはできないが，CEGAR(counter

[*3] 中韓変数も導入されている

example guded abstraction refinement) 法で複数回 SAT 問題を解くことで解を求めることができる [1].

6. おわりに

本稿では SAT 攻撃に耐性を持つ論理暗号化手法として TTLock を取り上げ，その現実的な実装方法についての考察を行った．極大ファンアウトフリーコーンに着目し，その有効パタンを用いて回路変換を行うことで実効的な論理暗号化が行えるものと思われる．また，実際の回路変換の方法として誤り修正論理合成の適用に関しても検討を行った．今後，提案手法を実装し，実際の論理暗号化の攻撃に対する耐性の評価を行う予定である．

謝辞

本研究の一部は科研費基盤 (C)「論理 IP の盗用を防ぐ堅牢な論理暗号化アルゴリズムの研究」による．

参考文献

[1] Matsunaga, Y.: Accelerating SAT-based Boolean Matching for Heterogeneous FPGAs using One-hot encoding and CEGAR technique, *Proceedings of of Asia and South Pacific Design Automation Conference 2015*, pp. 255–260 (online), DOI: 10.1109/ASPDAC.2015.7059014 (2015).

[2] Rajendran, J., Zhang, H., Zhang, C., Rose, G., Pino, Y., Sinanoglu, O. and Karri, R.: Fault Analysis-Based Logic Encryption, Vol. 64, No. 2, pp. 410–424 (2015).

[3] Roy, J. A., Koushanfar, F. and Markov, I. L.: Ending Piracy of Integrated Circuits, *Computer*, Vol. 43, No. 10, pp. 30–38 (online), DOI: 10.1109/MC.2010.284 (2010).

[4] Subramanyan, P., Ray, S. and Malik, S.: Evaluating the security of logic encryption algorithms, *2015 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pp. 137–143 (online), DOI: 10.1109/HST.2015.7140252 (2015).

[5] Yasin, M., Sengupta, A., Schafer, B. C., Makris, Y., Sinanoglu, O. and Rajendran, J.: What to Lock? Functional and Parametric Locking, *Great Lake Symposium on VLSI*, pp. 351–356 (2017).