Random Forestを用いたネットリスト特徴選択と機械学習によるハードウェアトロイ識別

長谷川 健人[1,a)]　柳澤 政生[1]　戸川 望[1]

概要：ハードウェアの設計・製造の一部を第三者に外部委託することが増加し，悪意ある第三者によりハードウェアにハードウェアトロイを挿入される危険性が指摘されている．このような現状から，ハードウェア出荷前にハードウェアトロイを検出することが強く求められている．ハードウェアトロイを構成するネット (トロイネット) には，通常の機能を実現するネット (ノーマルネット) とは異なる特徴が存在する．トロイネットとノーマルネットを効果的に識別するには，トロイネットの特徴のうち識別に有効なものだけを適切に抽出し，これらを用いたトロイネットとノーマルネットの識別が必要である．本稿では，まず第一にネットリスト中のそれぞれのトロイネットに対してトロイネットを特徴づける多数の特徴量を算出する．そして Random Forest と呼ばれる機械学習を用い，そこで得られる重要度をもとにトロイネットとノーマルネットを識別する際の F-measure を最大化する 11 の特徴量を抽出する．第二にその 11 の特徴量を用いて機械学習にもとづき与えられたネットリスト中の各ネットがトロイネットかノーマルネットかを識別する．その結果，True Positive Rate と False Positive Rate をそれぞれ最大 100%とし，F-measure を平均 74.3%とした．これは従来の機械学習によるハードウェアトロイ手法に比較して最良の結果である．

1. はじめに

近年 IoT をキーワードに，これまで高度な情報処理が必要とされなかった日常の身の回りのものにも高機能なハードウェアが搭載されるようになってきた．高機能なハードウェアの需要が高まるにつれ市場ではハードウェアを安価に容易に製造することが求められている．このため，ハードウェアベンダはハードウェア設計・製造の一部を外部委託することが増えている．

ハードウェアトロイとは，ハードウェアの設計・製造段階においてハードウェアに組み込まれた悪意のある機能の総称である．ハードウェアの設計・製造には複数の工程があり，いずれの段階でもハードウェアトロイ挿入のリスクが存在する [3]．特にハードウェアの設計工程では，ハードウェア設計情報だけを改変すればハードウェアトロイを挿入できるため，ハードウェアトロイ挿入のリスクが高い．ハードウェアトロイのリスクを回避するため，ハードウェア設計段階においてハードウェアを検出することが強く求められている．本稿ではハードウェア設計段階におけるハードウェアトロイ挿入に着目し，機械学習によるハードウェアトロイ識別を目的に，ハードウェアトロイの識別に効果的な回路の特徴量を抽出し，実際にハードウェアトロイ識別に有効であることを示す．

1.1 ハードウェア設計段階に注目したハードウェアトロイ識別

ハードウェア設計段階におけるハードウェアトロイ検出手法はすでにいくつか提案されている．

UCI [6] は，ハードウェアの設計情報にもとづき回路の論理シミュレーションによりハードウェアトロイを識別する．この手法ではハードウェアトロイが有効化されなければならない問題がある．ハードウェアが有効化される確率は非常に低いため，UCI だけですべての種類のハードウェアトロイを検出することは難しい．

設計段階に注目した別の手法として，FANCI [12] が挙げられる．FANCI は，回路の設計情報にもとづき，それぞれのゲートに対して真理値表を作成し，出力の遷移する確率が低いネットに対して Suspicous Flag を付加する．これにもとづきハードウェアトロイを識別する．この手法ではゲートの真理値表に注目しているため，順序回路を利用したハードウェアトロイを検出することは難しい．

ハードウェアトロイの回路の特徴に注目してネットリストにハードウェアトロイが含まれているか否かを識別する

[1] 早稲田大学大学院基幹理工学研究科情報理工・情報通信専攻
Dept. of Computer Science and Communications Engineering, Waseda University
[a)] kento.hasegawa@togawa.cs.waseda.ac.jp

手法 [10] も提案されている．この手法ではハードウェアトロイによく使われる回路の特徴や信号の遷移に着目し，ハードウェアトロイを識別する．

ハードウェアトロイ検出手法が提案される一方で，その検出手法で検出されないようなハードウェアトロイの開発も進んでいる．例えば，FANCI に対抗するハードウェアトロイ DeTrust [13] も提案されており，ハードウェアトロイ検出手法の開発とハードウェアトロイ自体の進歩はイタチごっこになっている．

こうした現状に対応するため，機械学習による検出手法 [4,5] も提案されている．機械学習を用いた検出手法ではハードウェアトロイのデータベースを学習させれば新しく登場したハードウェアトロイも検出可能となる．しかし，ハードウェアトロイの検出性能に課題がある．機械学習の精度を向上させるための一つの手段として，特徴量の改善が挙げられる．[4,5] では [10] で提案された回路の特徴にもとづき機械学習しているが，ここで用いられる特徴が必ずしも機械学習に有効であるとは限らない．機械学習に適した特徴量を抽出できれば，機械学習を用いた識別の性能の向上を見込める．

機械学習を用いたハードウェアトロイ検出の関連研究として，近年 [9] や [7] が提案されている．[9] では複数コアのプラットフォームを対象に動作時のハードウェアトロイ検出を目的としている．[7] では動作時の消費電力を解析することでハードウェアトロイを検出している．[9], [7] の手法はハードウェア動作時に注目しているのに対し本稿で着目するのはハードウェア設計時のハードウェアトロイ識別のための特徴量抽出である．

本稿では，Random Forest [2] [*1] を用いて機械学習に適した回路の特徴量を抽出し，ハードウェアトロイを識別する．

1.2 特徴量抽出手法の概要

本稿ではハードウェアの設計段階における機械学習を用いたハードウェアトロイ識別に着目する．まず，ネットリスト中のネットに対してトロイネットを特徴づける特徴量を算出する．次に，Random Forest を用いて得られる特徴量の重要度にもとづき，F-measure を最大化する 11 の特徴量を抽出する．

1.3 本研究の貢献

本研究による貢献を以下に示す．

(1) トロイネットを特徴づける 51 種類のネットの特徴量を取り上げ，このうち 11 種類の特徴量を抽出し F-measure を最大化する．

(2) 選択した特徴量により，True Positive Rate, True Negative Rate ともに最大 100%を得られたとともに，従来よりも高い F-measure を得られたことを示す．

[*1] 文献 [2] では “Random Forests” の表記となっているが，本稿では “Random Forest” として表記を統一して扱う．

表 1 ネットの特徴の例.

#	Benchmark	Net name	fan_in_5 (f_1)	out_neares_pout (f_2)	in_nearest_flipflop (f_3)	Trojan / Normal
(1)	RS232-T1000	iXMIT_state_1	59	2	4	Trojan
(2)	s35932-T100	Tj_OUT1	24	6	29	Trojan
(3)	s35932-T100	Trojan_SE	28	3	1	Trojan
(4)	RS232-T1600	n84	36	8	2	Normal
(5)	s35932-T100	n8403	7	6	38	Normal

1.4 本稿の構成

本稿は以下のように構成される．2 章で特徴量とトロイネットの識別率の関係を示す．3 章でハードウェアトロイ識別のためにトロイネットを特徴づける 51 種類の特徴量を列挙する．4 章で機械学習において有効なネットの 11 種類の特徴量を，Random Forest を用いて抽出する．5 章で提案手法により抽出した特徴量を用いた機械学習の結果にもとづき，その性能評価と既存手法との比較を示す．最後に 6 章で本稿をまとめる．

2. 特徴量選とトロイネットの識別率

機械学習において特徴量は多過ぎず少な過ぎず，適切な数を抽出する必要がある．特徴量の数が十分にない場合，クラスを識別するために必要な特徴量が不足する．一方，特徴量の数が過剰である場合，計算に必要な時間が大幅に増大するとともに，識別性能を高めるためより多くの学習量を必要とする．これは一般に次元の呪い [1] と呼ばれる問題として知られている．

実際，ベンチマークから具体例を取り出しハードウェアトロイ識別のために適切な数の特徴量を抽出する必要があることを示す．

表 1 に Trust-HUB [14] から選択したネットの特徴量の例を示す．ここで識別に用いる特徴量は fan_in_5 (入力側 5 段手前の論理ゲートのファンイン数), out_nearest_pout (最も近いプライマリ出力までの段数), in_nearest_flipflop (入力側で最も近いフリップフロップまでの段数) である．これらをそれぞれ f_1, f_2, f_3 とおく．表 1 において，(1)–(3) のネットはトロイネット，(4), (5) のネットはノーマルネットである．表 1 に示した f_1, f_2, f_3 にもとづくトロイネット識別を考える．簡単化のため，各特徴量に一つの閾値を設定するものとする．

まず，特徴量が 1 つ，f_1 だけを用いてトロイネットを識別することを考える．例えば閾値を 20 として $f_1 > 20$ のときをトロイネットと識別する場合，ネット (4) はトロイネットとして識別される (表 1 下線__部)．したがって，特徴量 f_1 だけを用いてトロイネットを正しく識別することはできない．

次に，特徴量が 2 つ，f_1, f_2 を用いてトロイネットを識別することを考える．それぞれの特徴量に対して閾値を設定して識別のための条件を考えると，$f_1 > 20$ かつ $f_2 < 7$ の

ときをトロイネットとする，という条件が考えられる (表 1 において $f_2 < 7$ を下線＝部で示す)．この条件でトロイネットとノーマルネットを正確に識別できる．

最後に，特徴量が 3 つ，f_1, f_2, f_3 を用いてトロイネットを識別することを考える．特徴量 f_1, f_2 だけを用いて閾値を設定した場合に正しく識別できたが，これに f_3 を加え f_3 に閾値を設定するとトロイネットを正しく識別することができなくなる．例えば $f_1 > 20$ かつ $f_2 < 7$ かつ $f_3 < 10$ をトロイネットとみなすとき，ネット (2) はノーマルネットとして識別される (表 1 において $f_3 < 10$ を下線～部で示す)．したがって，正しく識別することはできない．

以上より，表 1 の例では識別のための特徴量として f_1, f_2 を抽出するのが適当であるといえる．特徴量を増しても必ずしもトロイネットの識別率は上がらない．トロイネットの識別率を最大化するためには識別に用いる特徴量を抽出する必要がある．

3. ハードウェアトロイを識別するためのネットの特徴量

本章では，Trust-HUB [14] で公開されているゲートレベルのネットリストのうち表 3 に示す 15 種類を参照し，ハードウェアトロイと関係する可能性が高いネットの特徴量を取り上げる．

3.1 論理ゲートファンイン数

RS232-T1000 のように，ハードウェアトロイの中には，ある特定の条件 (トリガ) を満たした場合にだけ動作するものが存在する．これは，ハードウェアの動作テストやユーザの通常の利用においてハードウェアトロイが動作することを防ぐためである．これによりハードウェア製造者やユーザはハードウェアにトロイが挿入されていることに気付かない．

トリガ回路の中でも組合せ回路に着目する場合，ある特定の条件を満たした場合にのみ信号が遷移する回路を構成するため，複数の論理ゲートを組み合わせる必要がある．条件を厳しく設定した場合，論理ゲートのファンインの数は増大する．特にハードウェアトロイは条件が厳しく設定される場合が多いため，ノーマルネットと比較して近傍の論理ゲートのファンイン数が多くなる．

そこで，ネット n から入力側 x 段までの論理ゲートのファンインの合計 (fan_in_x) を特徴量とする．本稿では，トロイネットが小規模に構成されることを踏まえ 5 段を “近いネット” と考え，$x = 1, 2, 3, 4, 5$ とする．

3.2 フリップフロップ

RS232-T1200 のように，順序回路のトリガ回路を持つハードウェアトロイも存在する．ハードウェアトロイはノーマルネットに比べ小規模に，かつまとまった場所に構成されるため，トロイネットからトリガ回路のフリップフロップまでの段数は小さくなる．

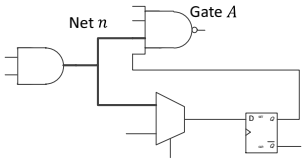

図 1 回路内のループの例.

このため，ネット n から入力側，出力側 x 段までのフリップフロップの数 ({in, out}_flipflop_x) と，最も近いフリップフロップまでの段数 ({in, out}_nearset_flipflop) を特徴量とする．本稿では，同様に $x = 1, 2, 3, 4, 5$ とする．

3.3 マルチプレクサ

ハードウェアトロイの中には，トリガ回路からのトリガ信号をマルチプレクサで受け，信号を切り替えるものが存在する．例えば，内部信号をそのままプライマリ出力に流出させるハードウェアトロイでは，マルチプレクサを用いて内部信号とプライマリ出力を接続する．このようにマルチプレクサとトリガ回路が組合せで利用されることが多い．

そこで，ネット n から入力側，出力側 x 段までのマルチプレクサの数 ({in, out}_multiplexer_x) と，最も近いマルチプレクサまでの段数 ({in, out}_nearest_multiplexer) を特徴量とする．本稿では，同様に $x = 1, 2, 3, 4, 5$ とする．

3.4 回路内のループ

ハードウェアトロイのトリガ回路には順序回路が利用される場合がある．このとき，前述のようにフリップフロップは特徴量の一つであるが，それに加え，フリップフロップを用いた回路ではループを構成することがある．本稿ではネット n のファンイン側・ファンアウト側に接続されるゲート A について，m 段目に再び同じゲート A に到達したとき，これを m 段のループと定義し特徴量とする．ループを構成する回路はハードウェアトロイの順序回路を用いたトリガ回路に加え，リングオシレータを意図的に挿入するハードウェアトロイにも対応する．

図 1 に回路内のループの例を示す．ネット n に着目したとき，ゲート A はネット n に直接接続される．ネット n の下側の経路に着目すると，3 段目でゲート A に到達する．このとき，ネット n の出力側 3 段までのループは 1 つ存在することになる．

ネット n から入力側，出力側 x 段までのループの数 ({in, out}_loop_x) を特徴量とする．本稿では，同様に $x = 1, 2, 3, 4, 5$ とする．

3.5 回路内の定数

文献 [10] において，フリップフロップの入力が定数 (ネットの一端が 0 または 1 に固定されている状態) であるとき，

表 2　ネットリストから抽出したネットの特徴.

Feature	Description
fan_in_x	ネット n の入力側 x ($1 \leq x \leq 5$) 段手前に接続される論理ゲートの数
in_flipflop_x	ネット n の入力側 x ($1 \leq x \leq 5$) 段手前に接続されるフリップフロップの数
out_flipflop_x	ネット n の出力側 x ($1 \leq x \leq 5$) 段手前に接続されるフリップフロップの数
in_multiplexer_x	ネット n の入力側 x ($1 \leq x \leq 5$) 段手前に接続されるマルチプレクサの数
out_multiplexer_x	ネット n の出力側 x ($1 \leq x \leq 5$) 段手前に接続されるマルチプレクサの数
in_loop_x	ネット n の入力側で x ($1 \leq x \leq 5$) 段それぞれでループを構成する数
out_loop_x	ネット n の出力側で x ($1 \leq x \leq 5$) 段それぞれでループを構成する数
in_const_x	ネット n の入力側 x ($1 \leq x \leq 5$) 段手前に接続される定数 (ネットの手前が 0，1 に固定されているもの) の数
out_const_x	ネット n の出力側 x ($1 \leq x \leq 5$) 段手前に接続される定数 (ネットの手前が 0，1 に固定されているもの) の数
in_nearest_pin	ネット n から最も近いプライマリ入力の段数
out_nearest_pout	ネット n から最も近いプライマリ出力の段数
{in, out}_nearest_flipflop	ネット n から入力/出力側で最も近いフリップフロップの段数
{in, out}_nearest_multiplexer	ネット n から入力/出力側で最も近いマルチプレクサの段数

その入力線がトロイネットであると指摘されている．定数を含む回路もハードウェアトロイ識別のための有用な目印となりうる．

ネット n から入力側，出力側 x 段までに存在する定数の数 ({in, out}_const_x) を特徴量とする．本稿では，同様に $x = 1, 2, 3, 4, 5$ とする．

3.6　プライマリ入出力との距離

ハードウェアのプライマリ入力はハードウェアトロイのトリガ条件の一つとして利用される場合がある．例えばデータ入力をトリガの一つとして利用したり，テスト信号をトリガに利用したりすることが多い．このため，トロイネットの近くにはハードウェアのプライマリ入力が接続されることがある．

ハードウェアのプライマリ出力はハードウェアトロイが内部信号を流出させるために用いられる場合がある．内部信号をそのまま外部に出力するハードウェアトロイの場合，必ずトロイネットはプライマリ出力に接続されることになる．

そこで，ネット n から最も近いプライマリ入力，プライマリ出力までの段数 (in_nearest_pin, out_nearest_pout) を特徴量とする．

3.7　回路から抽出する特徴量

以上の議論をもとに，回路から抽出する特徴量を表 2 にまとめる．ハードウェアトロイに関連するネットの特徴量として，全部で 51 種類となる．4 章ではここに示した特徴量にもとづき，ハードウェアトロイ識別に有効となる特徴量を抽出する．

4. Random Forest を用いたハードウェアトロイ識別のための特徴量の抽出

本章では Random Forest を Trust-HUB [14] ベンチマークに適用することでハードウェアトロイ識別に有効なネットの特徴量を抽出する．

表 3　実験で使用した Trust-HUB ベンチマーク.

Data Name	Number of normal nets	Number of Trojan nets	Data Name	Number of normal nets	Number of Trojan nets
RS232-T1000	283	36	s35932-T100	6,407	15
RS232-T1100	284	36	s35932-T200	6,405	12
RS232-T1200	289	34	s35932-T300	6,405	37
RS232-T1300	287	29	s38417-T100	5,798	12
RS232-T1400	273	45	s38417-T200	5,798	15
RS232-T1500	283	39	s38417-T300	5,801	44
RS232-T1600	292	29	s38584-T100	7,343	19
s15850-T100	2,429	27			

3 章では，ハードウェアトロイと関連があると考えられるネットの特徴量を 51 種類列挙した．ハードウェアトロイを含むベンチマークは最大で 7,000 以上のネットが含まれ，ハードウェアトロイを識別するためこれら 51 種類の特徴量に対し手動で閾値を設定することは現実的ではない．たとえ何らかの閾値を設定できたとしても，新種のハードウェアトロイが出現したときにそれを考慮して新たに閾値を設定することも難しい．また，2 章の議論より，ハードウェアトロイを識別するための特徴量の数は多すぎても少なすぎても良くない．そこで，本稿では Random Forest [2] を用いて特徴量を選択する．Random Forest は機械学習アルゴリズムの一つであり，特長としてクラス識別に使用した特徴量の重要度を算出できる．この重要度を利用することで，ハードウェアトロイ識別に有効な特徴量を抽出して最適な数に絞り込む．

本稿ではハードウェアトロイベンチマーク Trust-HUB [14] 上の 15 種類のデータを用いる．使用したネットリストを表 3 に示す．表 3 に示したベンチマークはハードウェア記述言語 Verilog HDL で記述されたゲートレベルのネットリストであり，どのネットがノーマルネットでどのネットがトロイネットかも示されている．このベンチマークにもとづき，ハードウェアトロイ識別のための特徴量を抽出する．

特徴量を選択する流れは大きく Step 1 と Step 2 に分かれる．Step 1 でネットリストから特徴量を算出し，Step 2 で実際に特徴量を抽出する．

4.1　Step 1：ネットリストから特徴量算出

Step 1 ではネットリストに対してそれぞれのネットの特徴量を算出する．ここで算出する特徴量はハードウェアトロイ識別に有用であるか否かに関係なく，表 2 にもとづいたネットの特徴を表す特徴量であり，それぞれのネット n に対して複数個の値が得られる．これらの特徴量をベクトル $\boldsymbol{v}$ とする．入力として与えるのは，表 3 の Trust-HUB で公開されているハードウェアトロイのベンチマークである．ここでは 3 章で説明した，表 2 に示す 51 種類の特徴量をそれぞれのネットに対して算出した．

4.2 Step 2

Step 2 では，Step 1 で算出した特徴量にもとづき，Random Forest を用いて学習する．

4.2.1 F-measure

機械学習では，識別性能を示す指標がいくつかある．機械学習の識別結果は，2 クラス分類問題の場合，機械学習による予測値と正解値との組み合わせにより 4 通り存在する．真のノーマルネットのうち正しくノーマルネットと識別したものの数を TN，誤ってトロイネットと識別したものの数を FP とする．真のトロイネットのうち正しくトロイネットと識別したものの数を TP，誤ってノーマルネットと識別したものの数を FN とする．

以上にもとづき，さらに 4 通りの指標が導出される．TPR はトロイネットを正しくトロイネットとして識別した割合であり，TP/(TP + FN) で表される．Recall (R) とも呼ばれる．TNR はノーマルネットを正しくノーマルネットとして識別した割合であり，TN/(TN + FP) で表される．Precision (P) はトロイネットとして識別されたもののうち，真にトロイネットであるものの割合であり，TP/(TP + FP で表される．F-measure (F) は R と P の調和平均である．F-measure が高いほど，Recall と Precision の両方が高いといえる．$F = 2PR/(P + R)$ で表される．本稿では F-measure を機械学習の指標として F-measure に注目する．

4.2.2 Random Forest によるトロイネットの識別

表 2 の 51 種類の特徴量にもとづき Random Forest を用いて学習し，トロイネットとノーマルネットを識別する．交差検証として Leave-one-out 法 [8] を適用する．トロイネットの数 (N_t) はノーマルネットの数 (N_n) と比較して非常に少ないので，本稿ではトロイネットを N_n/N_t 回重ねて学習する．機械学習の結果として表 3 の 15 個のベンチマークのそれぞれについて F-measure が得られ，この算術平均 (F-measure) を識別結果とする．このとき，同時に 51 種類の特徴量に対して重要度が得られる．

4.2.3 特徴量の最適化

特徴量の重要度にもとづき，上位半数となる 25 種類の特徴量を次の試行で用いる．このように試行を繰り返し，識別結果が前回を下回った時点で，それまでの試行で識別結果が最大であったときの特徴量を用いて，一つずつ特徴量を変化させ，同様に識別結果が最大となる特徴量の組み合わせを見つける．

特徴量の数を 51, 25, 12, 6 としたときの平均 F-measure を表 4 に示す．表 4 より，特徴量が 12 種類のとき平均 F-measure が最も高いことがわかる．この 12 種類の特徴量について，次に 1 種類ずつ特徴量を減らして調べる．特徴量の数が 13, 12, 11, 10 のときで実験した．このときの結果を表 5 に示す．表 5 より，特徴量が 11 種類のとき平均 F-measure が最も高い結果を得た．

表 4 特徴量選択 (1).

# of features	F-measure
51	63.3%
25	66.3%
12	70.1%
6	58.5%

表 5 特徴量選択 (2).

# of features	F-measure
13	70.5%
12	70.1%
11	71.7%
10	70.5%

表 6 選択した特徴量とその重要度.

Feature	Feature importance
fan_in_5	0.102
in_flipflop_4	0.040
in_flipflop_5	0.065
out_flipflop_3	0.125
in_loop_4	0.062
in_loop_5	0.070
out_loop_5	0.104
in_nearest_pin	0.108
out_nearest_pout	0.207
out_nearest_flipflop	0.048
out_nearest_multiplexer	0.069

表 7 機械学習による識別結果.

Test Data	TN	FP	FN	TP	TPR	TNR	Precision	F-measure
RS232-T1000	278	5	0	36	100.0%	98.2%	87.8%	93.5%
RS232-T1100	280	4	15	21	58.3%	98.6%	84.0%	68.9%
RS232-T1200	288	1	5	29	85.3%	99.7%	96.7%	90.6%
RS232-T1300	285	2	1	28	96.6%	99.3%	93.3%	94.9%
RS232-T1400	272	1	0	45	100.0%	99.6%	97.8%	98.9%
RS232-T1500	280	3	2	37	94.9%	98.9%	92.5%	93.7%
RS232-T1600	287	5	3	26	89.7%	98.3%	83.9%	86.7%
s15850-T100	2,410	9	18	9	33.3%	99.6%	50.0%	40.0%
s35932-T100	6,407	0	7	8	53.3%	100.0%	100.0%	69.6%
s35932-T200	6,405	0	11	1	8.3%	100.0%	100.0%	15.4%
s35932-T300	6,405	0	3	34	91.9%	100.0%	100.0%	95.8%
s38417-T100	5,798	0	7	5	41.7%	100.0%	100.0%	58.8%
s38417-T200	5,798	0	10	5	33.3%	100.0%	100.0%	50.0%
s38417-T300	5,800	1	1	43	97.7%	100.0%	97.7%	97.7%
s38584-T100	7,337	6	16	3	15.8%	99.9%	33.3%	21.4%

最終的に，抽出した特徴量とそのときの特徴量の重要度を表 6 に示す．特徴量の重要度は，重要な特徴量であるほど数値は高くなる．結果より，ファンイン数，前後のフリップフロップの数と出力側直近の段数，前後のループの数，プライマリ入出力までの段数，出力側マルチプレクサまでの段数を，ハードウェアトロイ識別のための特徴量として抽出した．

5. 機械学習によるハードウェアトロイ識別結果

本節では，4 節で提案した特徴量を用いて Random Forest で機械学習したときのハードウェアトロイ識別結果を示す．解析プログラムは Python で記述し，機械学習のライブラリである scikit-learn [11] の Random Forest Classifier を用いて実装した．

5.1 機械学習による識別結果

表 3 に示したベンチマークに対し，表 6 の特徴量を学習したときの結果を表 7 に示す．

TPR に着目すると，RS232-T1000 と RS232-T1400 で 100%を得た．これは，すべてのトロイネットを正しくトロイネットと識別できたことを示す．TNR に着目すると，ほとんどのベンチマークで 98%を超える結果を得た．ノーマルネットの数はトロイネットの数に比べて非常に多いため，

表 8　既存手法との Precision, F-measure の比較.

	Precision			F-measure		
Data	[4]	[5]	Ours	[4]	[5]	Ours
RS232-T1100	53.3%	9.6%	84.0%	59.3%	17.5%	68.9%
RS232-T1200	33.3%	4.6%	96.7%	45.2%	8.6%	90.6%
RS232-T1400	38.1%	7.6%	97.8%	48.5%	14.1%	98.9%
RS232-T1500	45.0%	7.2%	92.5%	56.3%	13.0%	93.7%
s15850-T100	16.1%	3.9%	50.0%	26.5%	7.4%	40.0%
s35932-T100	58.8%	1.5%	100.0%	62.5%	2.9%	69.6%
s38417-T100	10.5%	0.7%	100.0%	18.2%	1.5%	58.8%
Mean	36.5%	5.0%	88.7%	45.2%	9.3%	74.3%

FP がわずかに検出されたとしても TNR は高くなる傾向にある．しかし，s15850-T100 から s38417-T200 までのベンチマークで FP が 0 になっていることから，すべてのノーマルネットを正しくノーマルネットとして識別できたことを示す．Precision に着目すると，15 個中 13 個のベンチマークで 80%を超えており，10 個のベンチマークで 90%を超えている．さらに，5 個のベンチマークでは 100%となっている．Precision が 100%となった 5 つのベンチマークにはそれぞれ 5,000 を超えるネットが存在しており，この中からトロイネットだけを正しく識別することは難しい．しかし，提案手法により選択された特徴量を用いた識別によって，トロイネットの一部を正しく指摘することができた．特に s35932-T300 と s38417-T300 では TPR も 90%を超えており，識別によりトロイネットであると指摘されたトロイネットが，トロイネット全体の 90%であった．

5.2　既存手法との比較

既存の機械学習を用いた手法 [4,5] と，本稿での識別結果を比較する．[4] では，Support vector machine を用いて機械学習によりハードウェアトロイを識別している．[5] では，Neural network を用いて機械学習によりハードウェアトロイを識別している．文献に示されているうち，すべてに共通して提示されている値にもとづき Precision と F-measure を比較した．本稿の識別結果と，文献 [4], [5] に関して，F-measure の比較を表 8 に示す．

表 8 において，本稿で抽出した特徴量を用いた識別による Precision, F-measure が最も高い F-measure を示している．これらの結果より，本稿で得られた結果は F-measure が高いので機械学習の識別性能が良い，ということが言える．

6.　まとめ

本稿では，まずネットリストからトロイネットを特徴づける 51 種類の特徴量を算出した．次に，Random Forest で得られる重要度にもとづき，F-measure を最大化する 11 種類の特徴量を抽出した．抽出した特徴量を用いたハードウェアトロイ識別の実験では TPR, TNR ともに最大 100%となり，F-measure は既存の機械学習を用いたハードウェアトロイ識別手法と比較して最も高い結果となった．

今後の課題として，個別のハードウェアトロイに対する TPR, TNR 向上が挙げられる．

謝辞

本研究開発は一部，総務省 SCOPE (受付番号 141303001) の委託を受けた．

参考文献

[1] C. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[2] L. Breiman, “Random Forests,” *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[3] J. Francq and F. Frick, “Introduction to hardware trojan detection methods,” in *Proc. Design, Automation and Test in Europe (DATE)*, 2015, pp. 770–775.

[4] 長谷川健人, 大屋優, 柳澤政生, 戸川望, “SVM を利用したネットリストの特徴に基づくハードウェアトロイ識別,” **信学技報**, vol. 115, no. 338, VLD2015-58, pp. 135–140, 2015.

[5] 長谷川健人, 柳澤政生, 戸川望, “ニューラルネットを利用したネットリストの特徴にもとづくハードウェアトロイ識別,” **信学技報**, vol. 116, no. 93, CAS2016-1, pp. 1–6, 2016.

[6] M. Hicks, M. Finnicum, S. T. King, M. M. K. Martin, and J. M. Smith, “Overcoming an untrusted computing base: detecting and removing malicious hardware automatically,” in *Proc. Symposium on Security and Privacy (SP)*, 2010, pp. 159–172.

[7] T. Iwase, Y. Nozaki, M. Yoshikawa, and T. Kumaki, “Detection technique for hardware Trojans using machine learning in frequency domain,” in *2015 IEEE 4th Global Conference on Consumer Electronics (GCCE)*. IEEE, oct 2015, pp. 185–186.

[8] R. Kohavi, “A study of cross-validatio and bootstrap for accuracy estimation and model selection,” in *Proc. International Joint Conference on Artificial Intelligence*, vol. 2, 1995, pp. 1137–1143.

[9] A. Kulkarni, Y. Pino, and T. Mohsenin, “SVM-based real-time hardware Trojan detection for many-core platform,” in *2016 17th International Symposium on Quality Electronic Design (ISQED)*. IEEE, mar 2016, pp. 362–367.

[10] M. Oya, Y. Shi, M. Yanagisawa, and N. Togawa, “A score-based classification method for identifying hardware-trojans at gate-level netlists,” in *Proc. Design, Automation and Test in Europe (DATE)*, 2015, pp. 465–470.

[11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and É. Duchesnay, “Scikit-learn: machine learning in python,” *The Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[12] A. Waksman, M. Suozzo, and S. Sethumadhavan, “FANCI: identification of stealthy malicious logic using boolean functional analysis,” in *Proc. ACM SIGSAC Conference on Computer and Communications Security (ACM-CCS)*, 2013, pp. 697–708.

[13] J. Zhang, F. Yuan, and Q. Xu, “DeTrust: defeating hardware trust verification with stealthy implicitly-triggered hardware Trojans,” in *Proc. ACM SIGSAC Conference on Computer and Communications Security (ACM-CCS)*, 2014, pp. 153–166.

[14] “Trust-HUB.” http://www.trust-hub.org