資料19－2

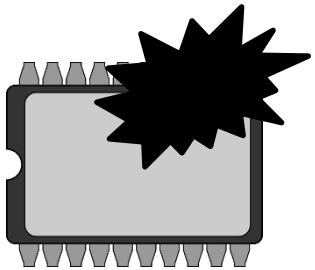
ハードウェアチップ脆弱性検知手法

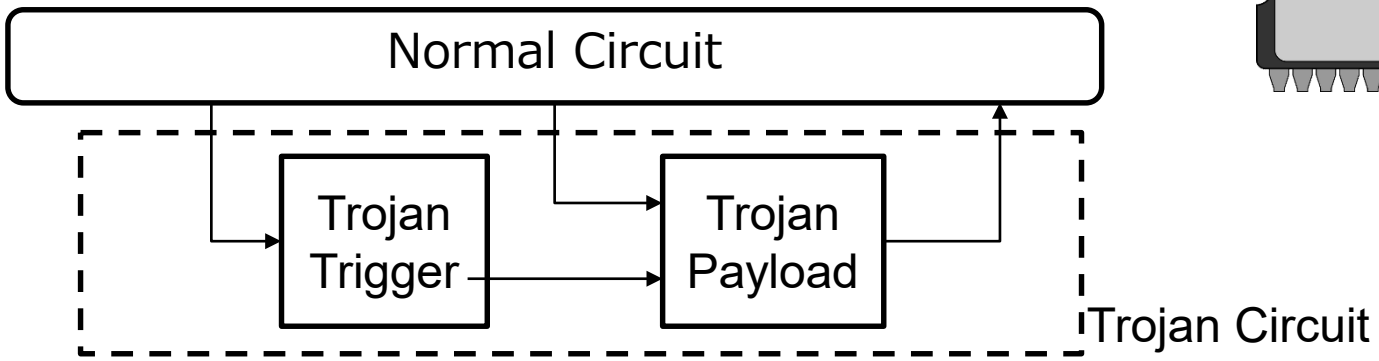
過去10年間のチップ脆弱性検知手法の歩みと現在の進捗状況

戸川 望
早稲田大学理工学術院
ntogawa@waseda.jp

ハードウェアトロイ(1)

- ハードウェアトロイとは
  - ハードウェアに組み込まれた悪意のある機能





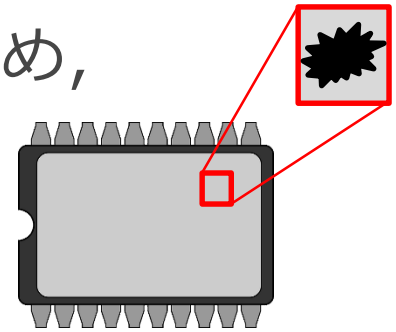
- ハードウェアトロイの特徴

物理的特性に対する攻撃の特徴	ハードウェアトロイ機能有効化のトリガの特徴	ハードウェアトロイの動作の特徴

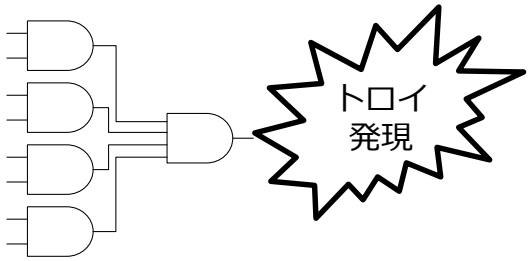
M.Tehranipoor and F.Koushanfar, IEEE Transactions on Design and Test of Computers, vol. 27, no. 1, pp. 10-25, 2010.

ハードウェアトロイ(2)

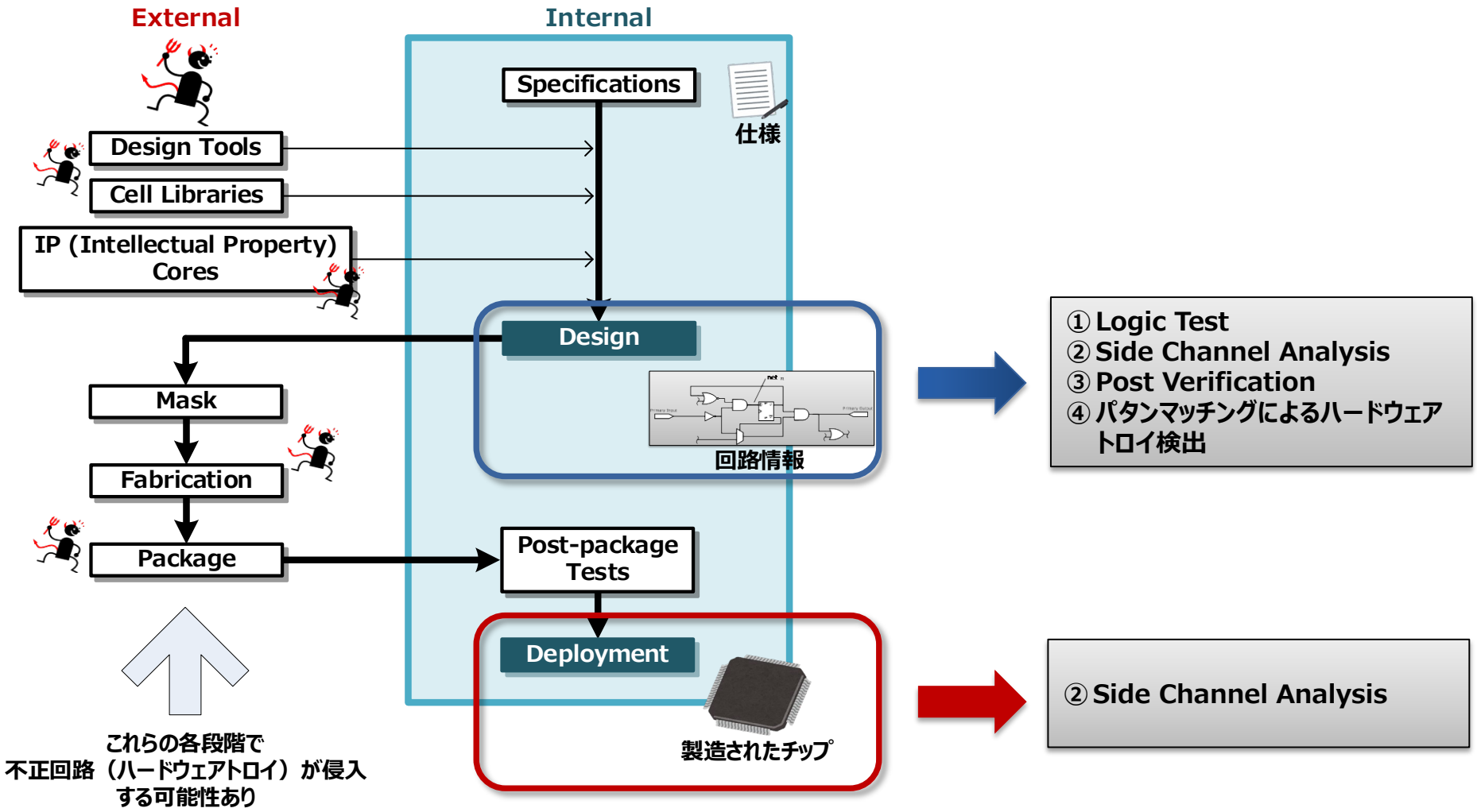
- ハードウェアトロイの特徴
  - 元の回路に比べ**小規模**
    ICベンダやユーザから自身の存在を隠すため，
    回路規模が小さく収められている．



  - ある**一定の条件で**ハードウェアトロイの機能が発現
    普段は正常に動作するのに，ある特定の条件
    （特定の入力や，回路の内部状態）をトリガとして
    悪意ある機能が作動する．
  - **常に動作**するタイプのものもある



ハードウェアトロイ(3)


External
Internal
Specifications
仕様
Design Tools
Cell Libraries
IP (Intellectual Property) Cores
Design
回路情報
Mask
Fabrication
Package
Post-package Tests
Deployment
製造されたチップ
これらの各段階で
不正回路（ハードウェアトロイ）が侵入する可能性あり
① Logic Test
② Side Channel Analysis
③ Post Verification
④ パタンマッチングによるハードウェアトロイ検出
② Side Channel Analysis

過去10年間のチップ脆弱性検知手法の歩み

ハードウェアトロイ検知のアプローチ

- 設計段階のハードウェアトロイ検知

① Logic Test

② Side Channel Analysis

③ Post Verification

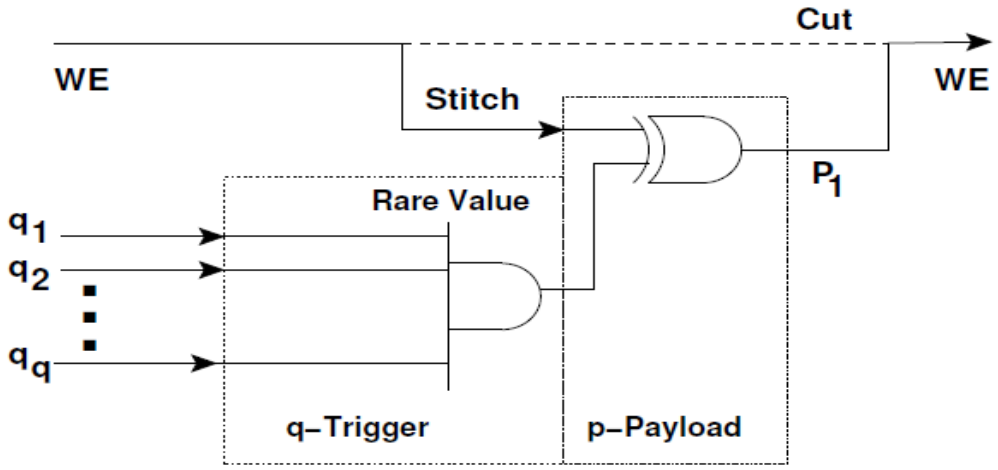
④ パタンマッチングによるハードウェアトロイ検出

- 製造段階のハードウェアトロイ検知

② Side Channel Analysis

① Logic Test

- ハードウェアトロイを活性化させるインプットベクタを生成する

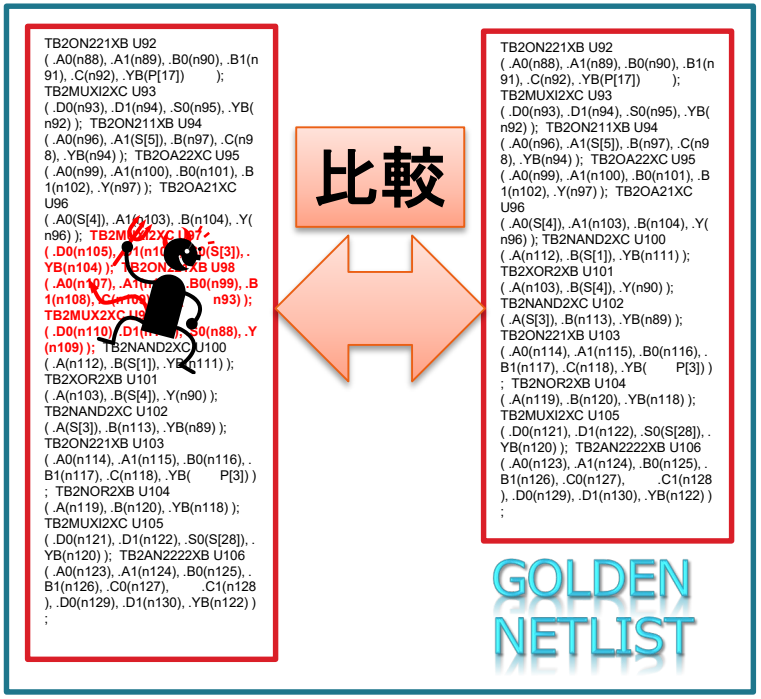


- 問題点:
  - 滅多に活性化しないハードウェアトロイを活性化させる必要あり

F. Wolff, C. Papachristou, S. Bhunia, and R. S. Chakraborty, “Towards trojan-free trusted ics: Problem analysis and detection scheme,” in *Proc. Design, automation and test in Europe (DATE)*, 2008, pp. 1362–1365.

② Side Channel Analysis

- ハードウェアトロイの挿入されていないネットリストのサイドチャネル情報と検査対象のネットリストのサイドチャネル情報を比較

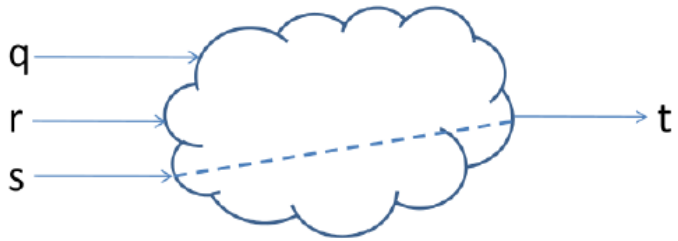


- M. Potkonjak, A. Nahapetian, M. Nelson, and T. Massey, “Hardware trojan horse detection using gate-level characterization,” in *Proc. Design Automation Conference (DAC)*, 2009, pp. 688–693.
- K. Xiao, X. Zhang, and M. Tehranipoor, “A clock sweeping technique for detecting hardware trojans impacting circuits delay,” *IEEE Transactions on Design & Test*, vol. 30, no. 2, pp. 26–34, 2012.
- Y. Cao, C.-H. Chang, and S. Chen, “Cluster-based distributed active current timer for hardware trojan detection,” in *Proc. International Symposium on Circuits and Systems (ISCAS)*, 2013, pp. 1010–1013.
- B. Cha and S. K. Gupta, “Trojan detection via delay measurements: A new approach to select paths and vectors to maximize effectiveness and minimize cost,” in *Proc. Design, Automation & Test in Europe Conference & Exhibi- tion (DATE)*, 2013, pp. 1265–1270.

- Golden Netlist が必要（現実的にはかなり難しい）

③ Post Verification (1)

- UCI
  - 検証中に特定のインプットの値が常に同じ出力をもたらしている信号を特定し，その信号にMUXを挿入してハードウェアトロイを活性化させ，ハードウェアトロイを検出

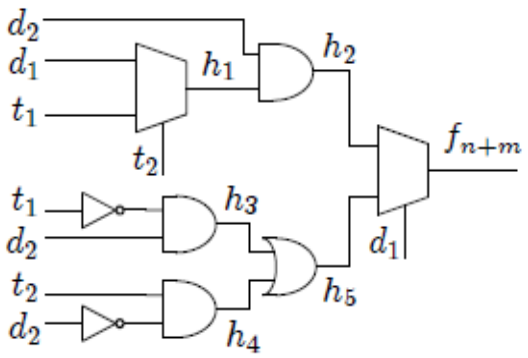


- いかにハードウェアトロイを活性化するか

M. Hicks, M. Finnicum, S. T. King, M. M. 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.

③ Post Verification (2)

- VeriTrust
  - 回路検証中に特定のインプットの値が常に同じ出力をもたらしている信号を特定
  - その信号の論理値をカルノー図で表現し適切なベクタを入力
  - ハードウェアトロイを活性化



K-Map

t_1t_2 \ d_1d_2	00	01	11	10
00	0	0	1	0
01	0	0	1	0
11	0	1	0	0
10	0	0	1	0

J. Zhang, F. Yuan, L. Wei, Z. Sun, and Q. Xu, “Veritrust: verification for hardware trust,” in *Proc. Design Automation Conference (DAC)*, 2013, p. 61.

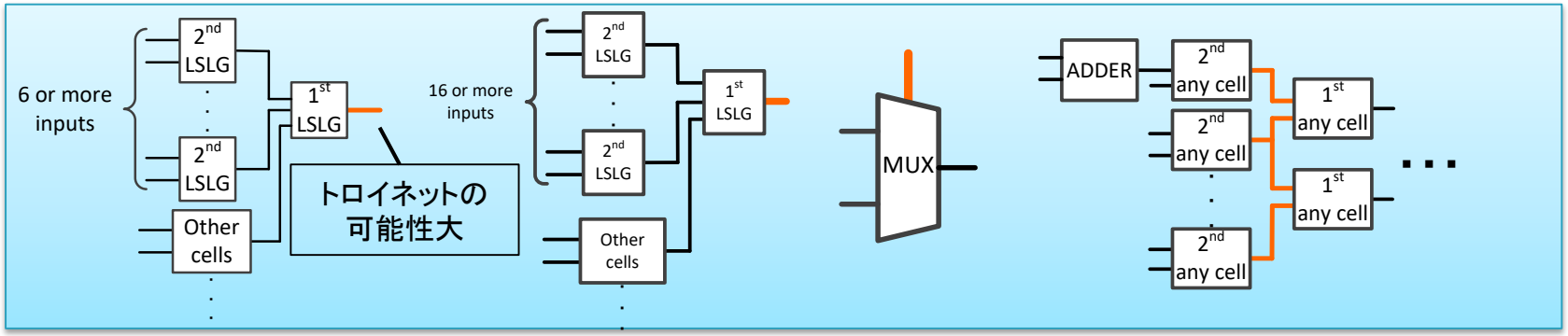
④ パタンマッチングによるハードウェアトロイ検出

- アプローチ：
  - ① 設計データ中の全信号線集合の中から，**トロイネット**「**らしい**」ものを見つけ出す（**弱識別ネット**）
  - ② 弱識別ネット集合から「**確実に**」**トロイネット**となるものを見つけ出す（**強識別ネット**）

HWトロイを構成する信号線

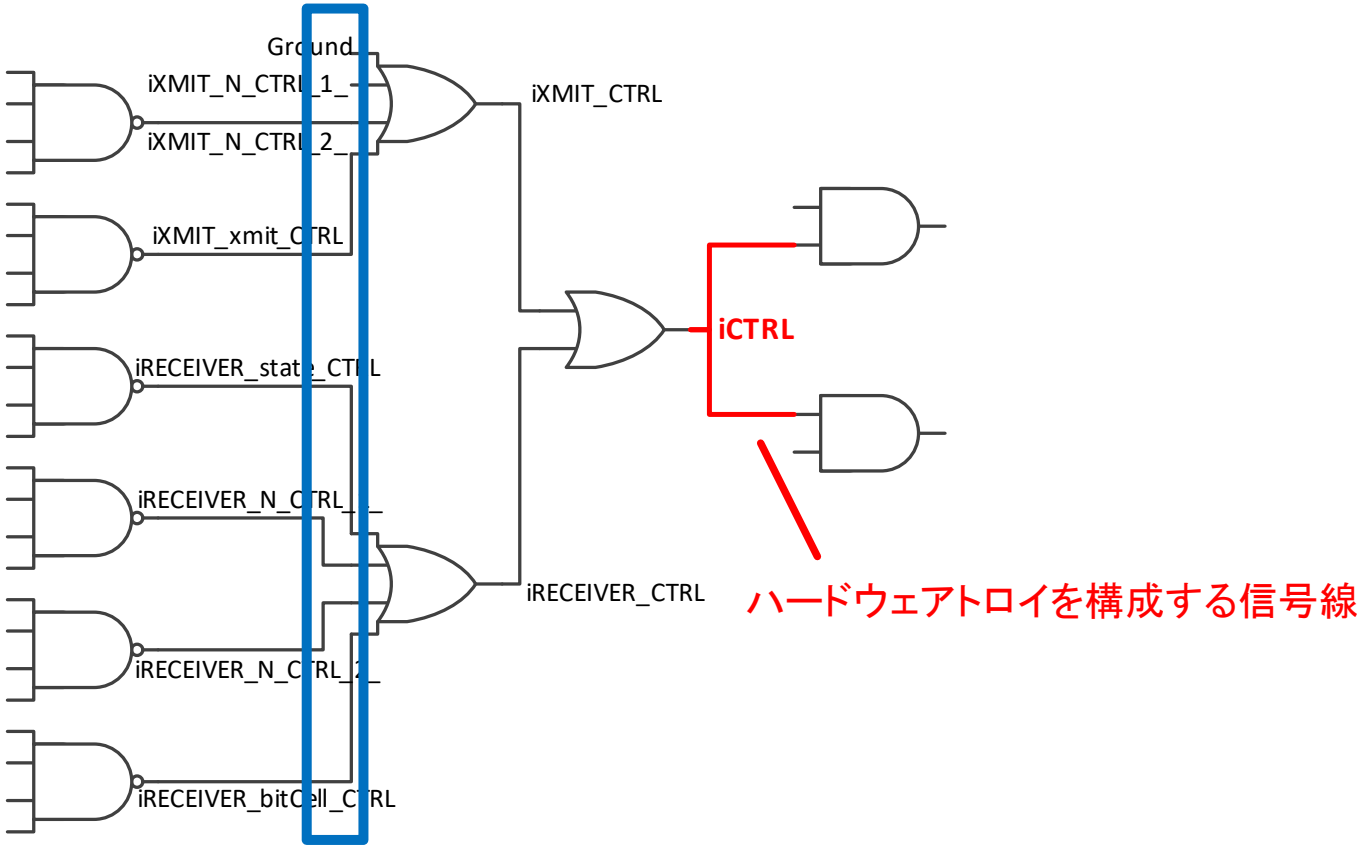
強識別ネットが1つでも発見されれば，ハードウェアトロイが侵入

① **弱識別**：ハードウェアトロイを構成する信号線の9つの特徴
  - 特徴にもとづき，各信号線にスコア付けする



Masaru Oya, Youhua Shi, Masao Yanagisawa, Nozomu Togawa, DATE 2015: 465-470.

トロイネットの例



信号線iCTRLの前々段の入力数=8 ⇒ iCTRLがハードウェアトロイである

④ パタンマッチングによるハードウェアトロイ検出

② **強識別**：2つの基準により強識別する

- **基準1**：スコアの合計≧3の信号線 → 「確実に」トロイネット
- **基準2**：基準1を満たさなくても，最大スコア信号線の数が5以下であり，一定値を出力するクロックサイクルが99.999%以上の信号線 → 「確実に」トロイネット

強識別された信号線が1つでもあれば，**ハードウェアトロイ侵入**と判定

識別結果（一部）：

Benchmark	トロイ有無	開発技術による検出/非検出の結果
b19-T100	有	検出
b19-T200	有	検出
EthernetMAC10GE-T700	有	検出
EthernetMAC10GE-T710	有	検出
EthernetMAC10GE-T720	有	検出
EthernetMAC10GE-T730	有	検出
RS232-T1000	有	検出
RS232-T1100	有	検出
RS232-T1200	有	検出
RS232-T1300	有	検出
RS232-T1400	有	検出
RS232-T1500	有	検出
RS232-T1600	有	検出
s15850-T100	有	検出
s35932-T100	有	検出
s35932-T200	有	検出
s35932-T300	有	検出
s38417-T100	有	検出
s38417-T200	有	検出
s38417-T300	有	検出
s38584-T100	有	検出
s38584-T200	有	検出
s38584-T300	有	検出
vga_lcd-T100	有	検出
wb_conmax-T100	有	検出
AES-T1	有	検出
AES-T2	有	検出

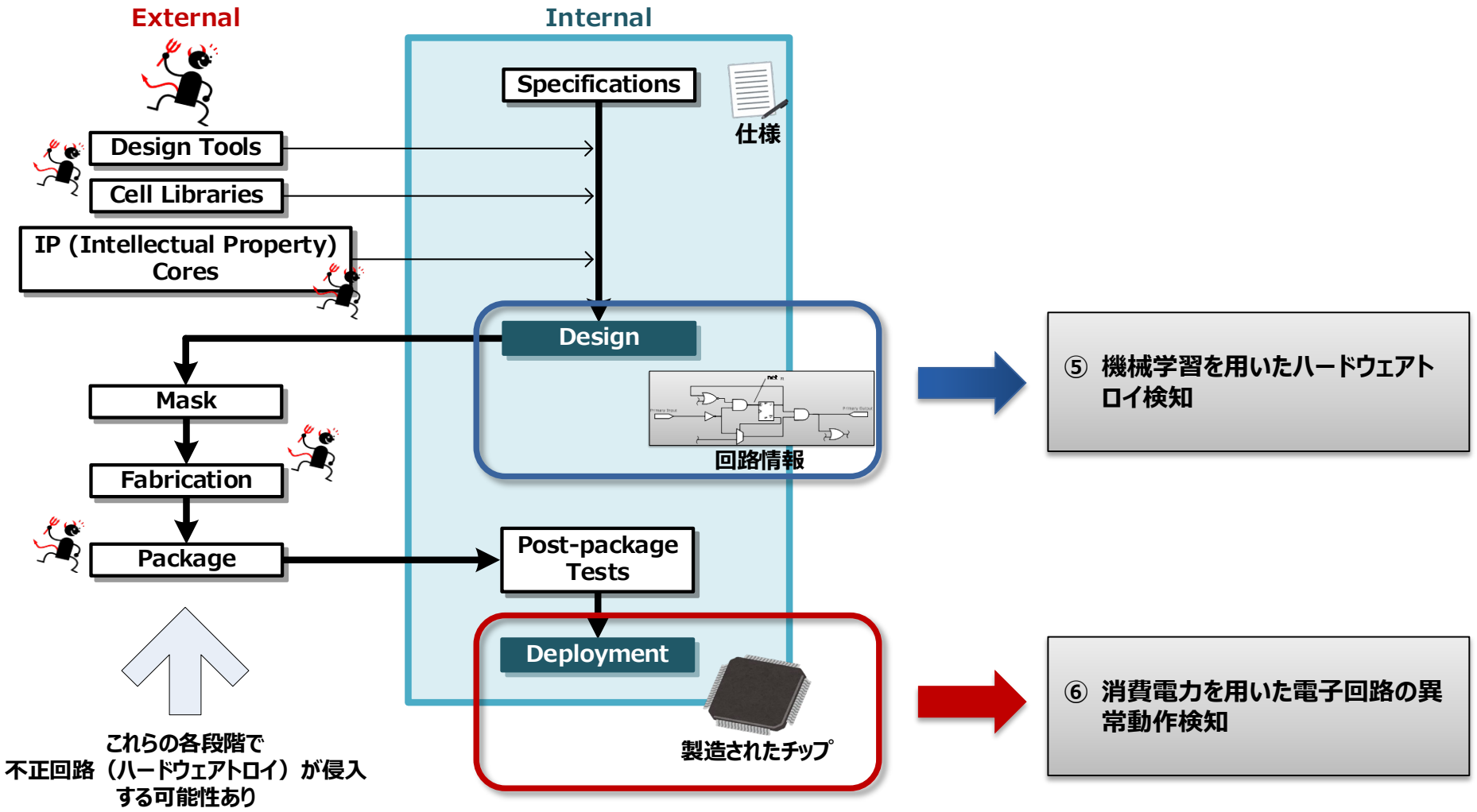
Benchmark	トロイ有無	開発技術による検出/非検出の結果
b19	無	非検出
EthernetMAC10GE	無	非検出
RS232	無	非検出
s15850	無	非検出
s35932	無	非検出
s38417	無	非検出
s38584	無	非検出
vga_lcd	無	非検出
wb_conmax	無	非検出
AES	無	非検出
c17	無	非検出
c432	無	非検出
c499	無	非検出
c880	無	非検出
c1355	無	非検出
c1908	無	非検出
c2670	無	非検出
c3540	無	非検出
c5315	無	非検出
c6288	無	非検出
c7552	無	非検出

世界で初めて，標準ベンチマークTrust-HUB中の全ゲートレベル回路に対して，誤りなくハードウェアトロイの有無を識別

現在の進捗状況

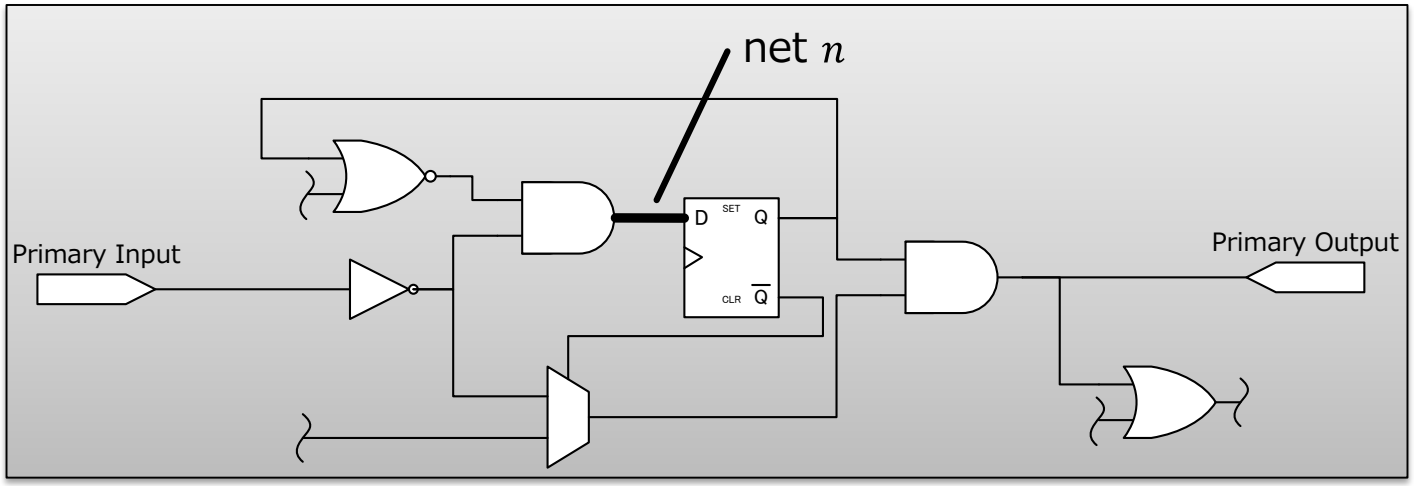
⑤機械学習を用いたハードウェアトロイ検知
⑥消費電力を用いた電子回路の異常動作検知

ハードウェアトロイ検知ー現在の進捗状況



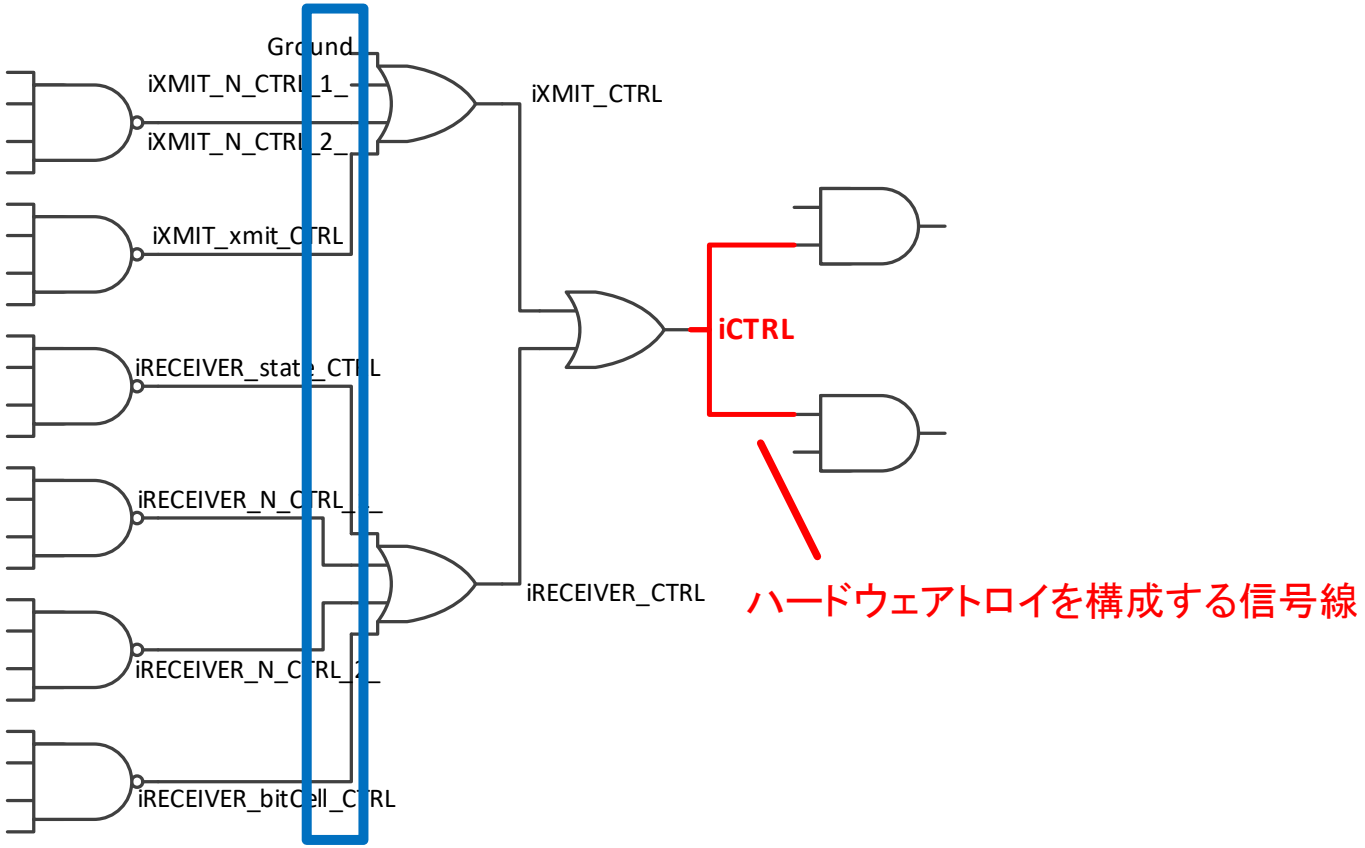
⑤機械学習を用いたハードウェアトロイ検知ー最適な特徴量(1)

- 何がハードウェアトロイの特徴か？



- プライマリ入出力までの段数
- マルチプレクサまでの段数
- フリップフロップまでの段数　など

⑤機械学習を用いたハードウェアトロイ検知ー最適な特徴量(2)

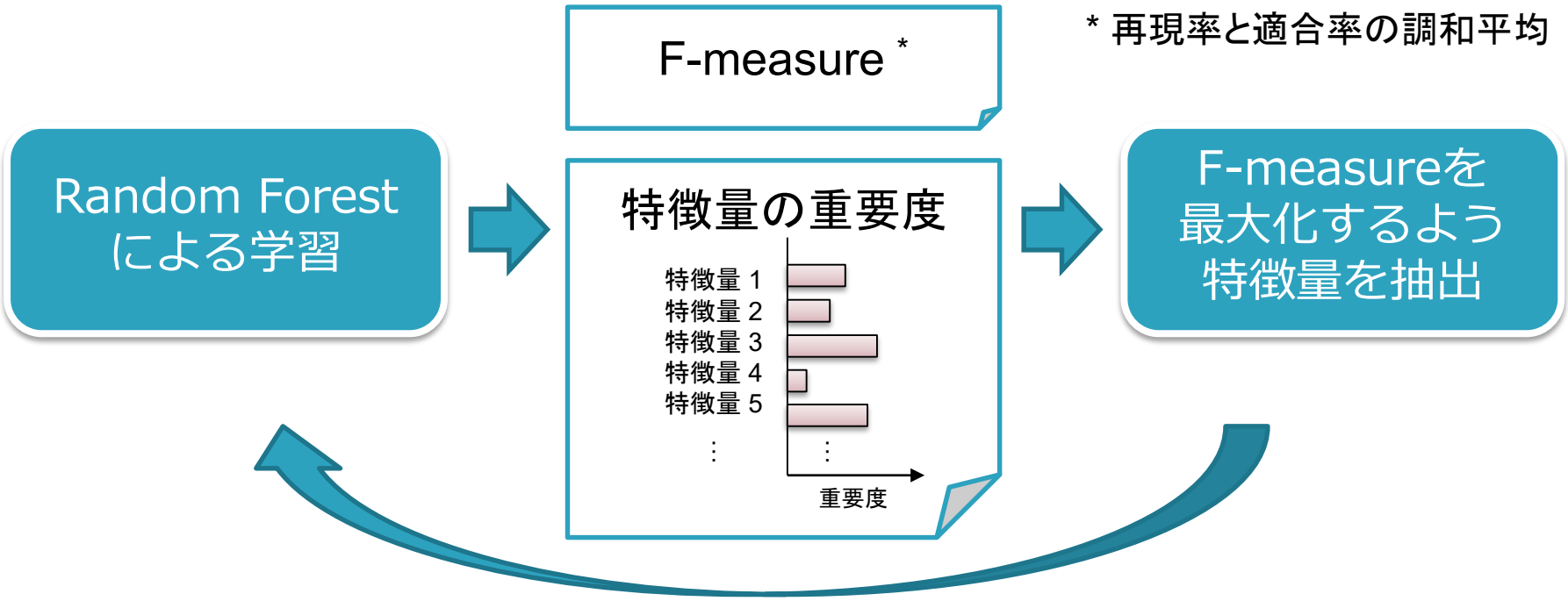


信号線iCTRLの前々段の入力数=8 ⇒ iCTRLがハードウェアトロイである

Trust-HUB: https://www.trust-hub.org/

Random Forest を用いた特徴量抽出

- Trust-HUBベンチマークを用いてRandom Forestにより学習，評価



Random Forestにより「重要」と判断された特徴量を抽出

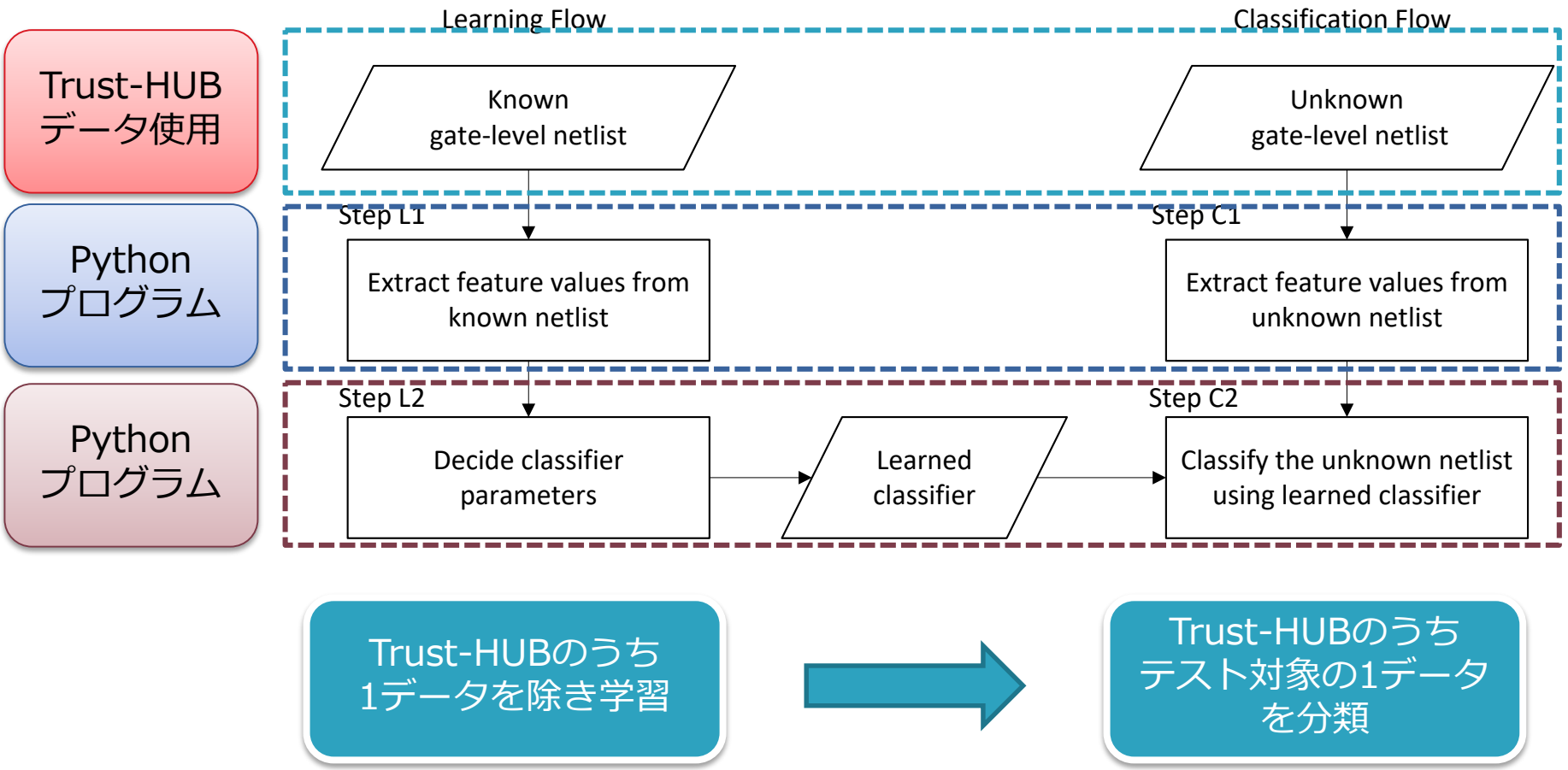
11種類の特徴量の抽出

- 指標選択結果

Feature	Type	Importance
fan_in_5	ファンイン数	0.102
in_flipflop_4	フリップフロップ数	0.040
in_flipflop_5	フリップフロップ数	0.065
out_flipflop_3	フリップフロップ数	0.125
in_loop_4	ループ数	0.062
in_loop_5	ループ数	0.070
out_loop_5	ループ数	0.104
in_nearest_pin	直近プライマリ入力の段数	0.108
out_nearest_pout	直近プライマリ出力の段数	0.207
out_nearest_flipflop	直近フリップフロップの段数	0.048
out_nearest_multiplexer	直近マルチプレクサの段数	0.069

ハードウェアトロイ検出のフロー

- 実験手法



ベンチマークセットの交差検証(1)

- ベンチマークセット
  - Trust-HUBベンチマーク回路の1つを既知のものとして学習
  - 残った1つのベンチマーク回路を未知のものとして識別

Table: Used benchmarks in the experiments

Benchmark	# of normal nets	# of Trojan nets
RS232-T1000	266	45
RS232-T1100	300	12
RS232-T1200	305	10
RS232-T1300	298	9
RS232-T1400	299	12
RS232-T1500	302	12
RS232-T1600	298	9
s15850-T100	2429	27
s35932-T100	6423	15
s35932-T200	6419	16
s35932-T300	6423	37
s38417-T100	5807	12
s38417-T200	5807	15
s38417-T300	5807	44
s38584-T100	7390	9
s38584-T200	7380	200
s38584-T300	7380	1730

ベンチマークセットの交差検証(2)

- ベンチマークセット
  - Trust-HUBベンチマーク回路の1つを既知のものとして学習
  - 残った1つのベンチマーク回路を未知のものとして識別

Table: U…nents

Benchmark	nets	Trojan nets
RS232-T1000	266	45
RS232-T1100	300	12
RS232-T1200	305	10
RS232-T1300	298	9
RS232-T1400	299	12
RS232-T1500	302	12
RS232-T1600	298	9
s15850-T100	2429	27
s35932-…	……	15
s35932-…	……	16
s35932-…	……	37
s38417-T100	5807	12
s38417-T200	5807	15
s38417-T300	5807	44
s38584-T100	7390	9
s38584-T200	7380	200
s38584-T300	7380	1730

Unknown netlist

Known netlists

ランダムフォレストによるハードウェアトロイ信号線の検出結果

Test Data	TN	FP	FN	TP	TPR	TNR	Accuracy
RS232-T1000	280	3	0	36	100.0%	98.9%	99.1%
RS232-T1200	289	0	4	30	88.2%	100.0%	98.8%
RS232-T1300	287	0	0	29	100.0%	100.0%	100.0%
RS232-T1400	273	0	1	44	97.8%	100.0%	99.7%
RS232-T1500	282	1	2	37	94.9%	99.6%	99.1%
s15850-T100	2,418	1	6	21	77.8%	100.0%	99.7%
s35932-T100	6,407	0	4	11	73.3%	100.0%	99.9%
s35932-T300	6,404	1	7	30	81.1%	100.0%	99.9%
s38417-T100	5,798	0	8	4	33.3%	100.0%	99.9%
s38417-T200	5,798	0	8	7	46.7%	100.0%	99.9%
s38417-T300	5,801	0	11	33	75.0%	100.0%	99.8%
s38584-T100	7,341	2	18	1	5.3%	100.0%	99.7%
RS232-T1100	279	5	18	18	50.0%	98.2%	92.8%
RS232-T1600	289	3	2	27	93.1%	99.0%	98.4%
s35932-T200	6,405	0	11	1	8.3%	100.0%	99.8%
s38584-T300	7,343	1	143	1,001	87.5%	100.0%	98.3%
s38584-T200	7,340	3	60	67	52.8%	100.0%	99.2%
s35932-free	6,405	0	0	0	100.0%	100.0%	100.0%
s38417-free	5,798	0	0	0	100.0%	100.0%	100.0%
s38584-free	7,343	0	0	0	100.0%	100.0%	100.0%
Average					73.3%	99.8%	99.2%

ランダムフォレストによるハードウェアトロイ信号線の検出結果

ハードウェアトロイを含む回路から，ハードウェアトロイ信号線を正しく識別することに成功

			N	TP	TPR	TNR	Accuracy
			0	36	100.0%	98.9%	99.1%
			4	30	88.2%	100.0%	98.8%
			0	29	100.0%	100.0%	100.0%
RS232-T1400	273	0	1	44	97.8%	100.0%	99.7%
RS232-T1500	282	1	2	37	94.9%	99.6%	99.1%
s15850-T100	2,418	1	6	21	77.8%	100.0%	99.7%
s35932-T100	6,407	0	4	11	73.3%	100.0%	99.9%
s35932-T300	6,404	1	7	30	81.1%	100.0%	99.9%
s38417-T100	5,798	0	8	4	33.3%	100.0%	99.9%
s38417-T200	5,798	0	8	7	46.7%	100.0%	99.9%
s38417-T300	5,801	0	11	33	75.0%	100.0%	99.8%
s38584-T100	7,341	2	18	1	5.3%	100.0%	99.7%
RS232-T1100	279	5	18	18	50.0%	98.2%	92.8%
RS232-T1600	289	3	2	27	93.1%	99.0%	98.4%
s35932-T200	6,405	0	11	1	8.3%	100.0%	99.8%
s38584-T300	7,343	1	143	1,001	87.5%	100.0%	98.3%
s38584-T200	7,340	3	60	67	52.8%	100.0%	99.2%
s35932-free	6,405	0	0	0	100.0%	100.0%	100.0%
s38417-free	5,798	0	0	0	100.0%	100.0%	100.0%
s38584-free	7,343	0	0	0	100.0%	100.0%	100.0%
Average					73.3%	99.8%	99.2%

⑥消費電力を用いた電子回路の異常動作検知
ー典型的な悪意のある機能のモデル化(1)

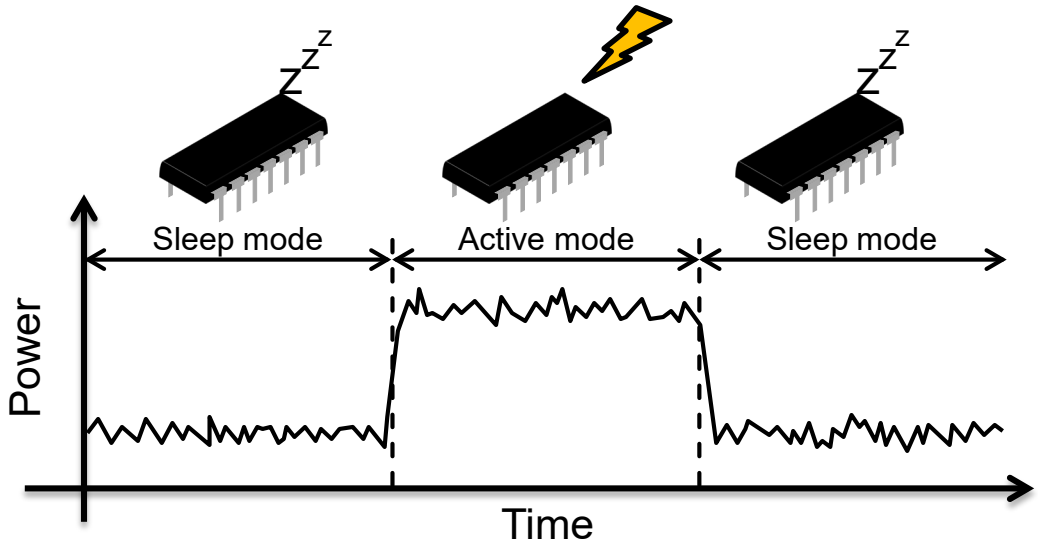
- 組込みマイコンの動作状態

通常状態(Active mode)

- 通常の処理を実行する状態

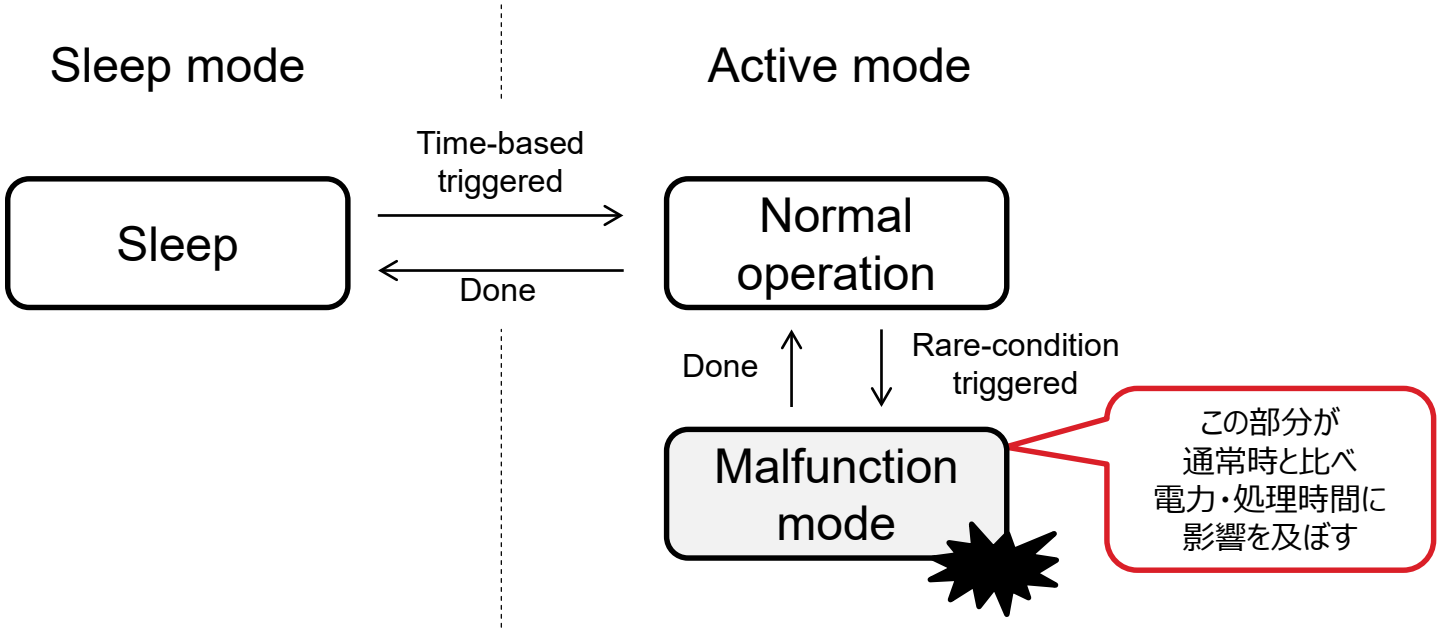
スリープ状態(Sleep mode)

- 必要最小限の機能だけ有効にして電力消費を抑える状態



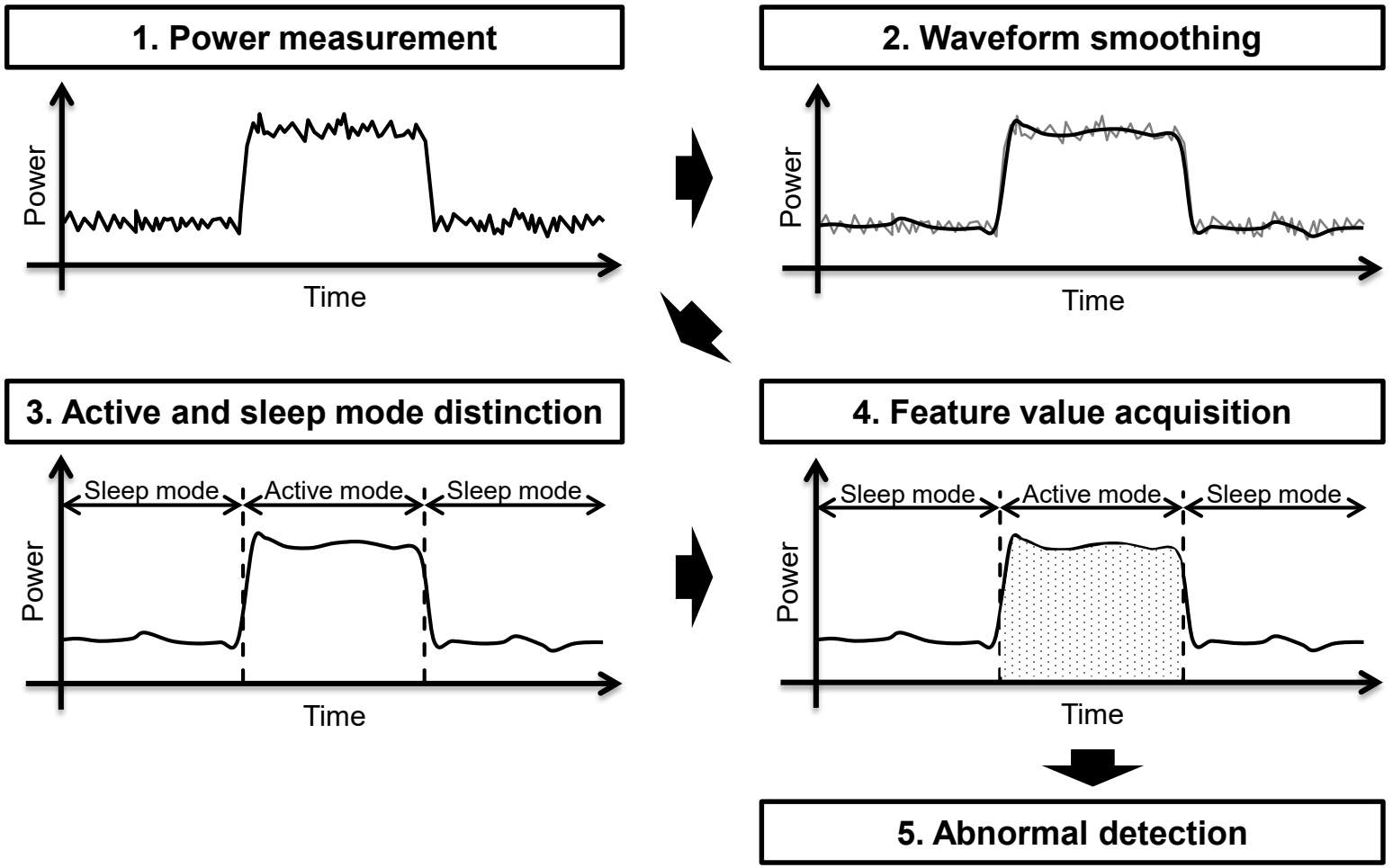
⑥消費電力を用いた電子回路の異常動作検知
ー典型的な悪意のある機能のモデル化(2)

- 悪意のある機能の発現による影響



通常モード時の消費電力や継続時間をもとに
悪意のある機能の発現を検知

悪意のある機能発現検知ー全体フローー



Arduino UNO を用いた検証(1)

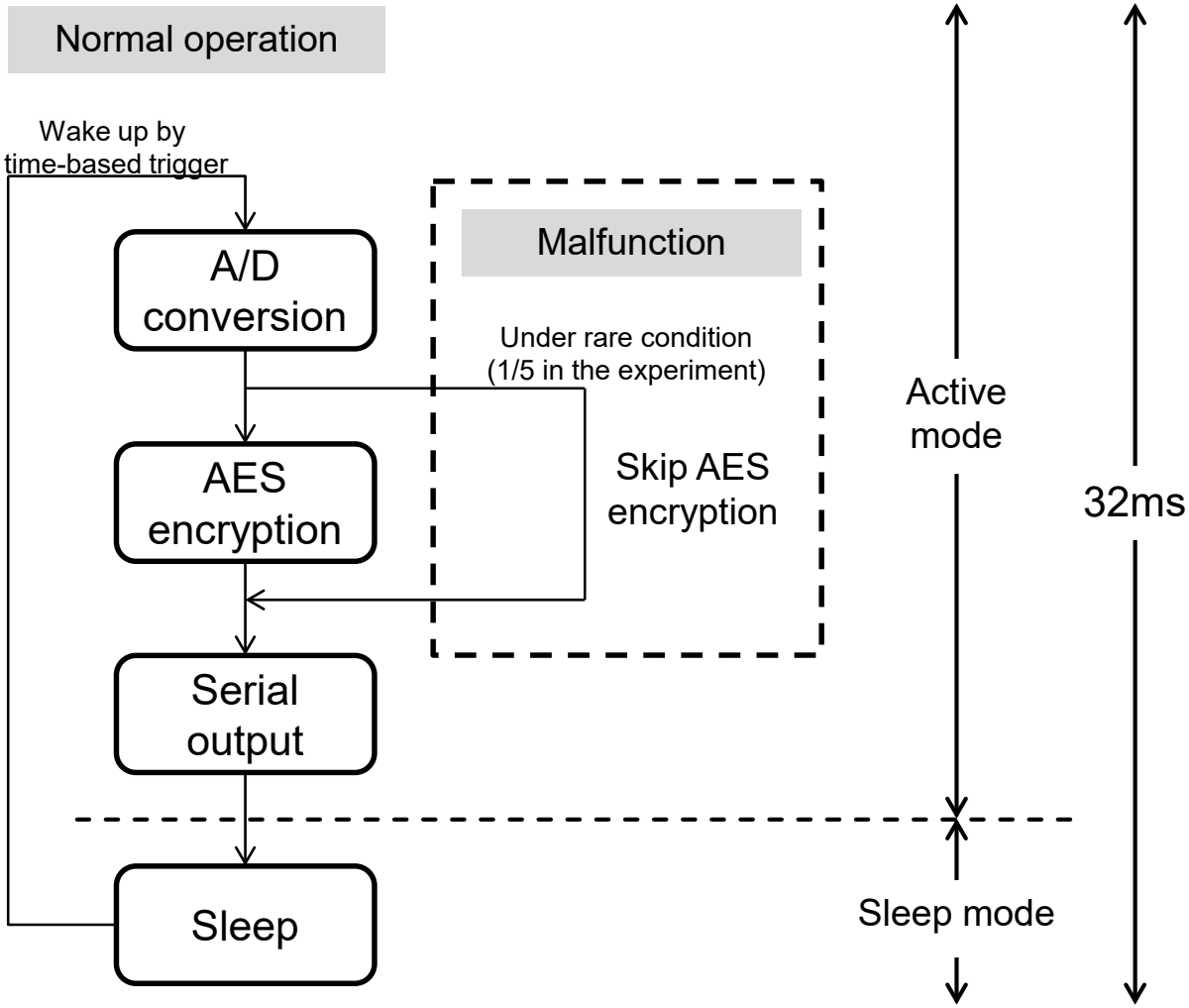
- 本発表で実験に使用するデバイス・機能
  - 組込みマイコン
    Arduino UNO

  - アプリケーション
    AES暗号処理 *
    （AD変換結果を32msごとに暗号化して送信）
    - AES暗号処理はGPLv3のオープンソース「AESLib」使用
  - 悪意のある機能
    5回に1回の割合でAES暗号処理を無効化

* https://github.com/DavyLandman/AESLib
* http://akizukidenshi.com/catalog/g/gM-07385/

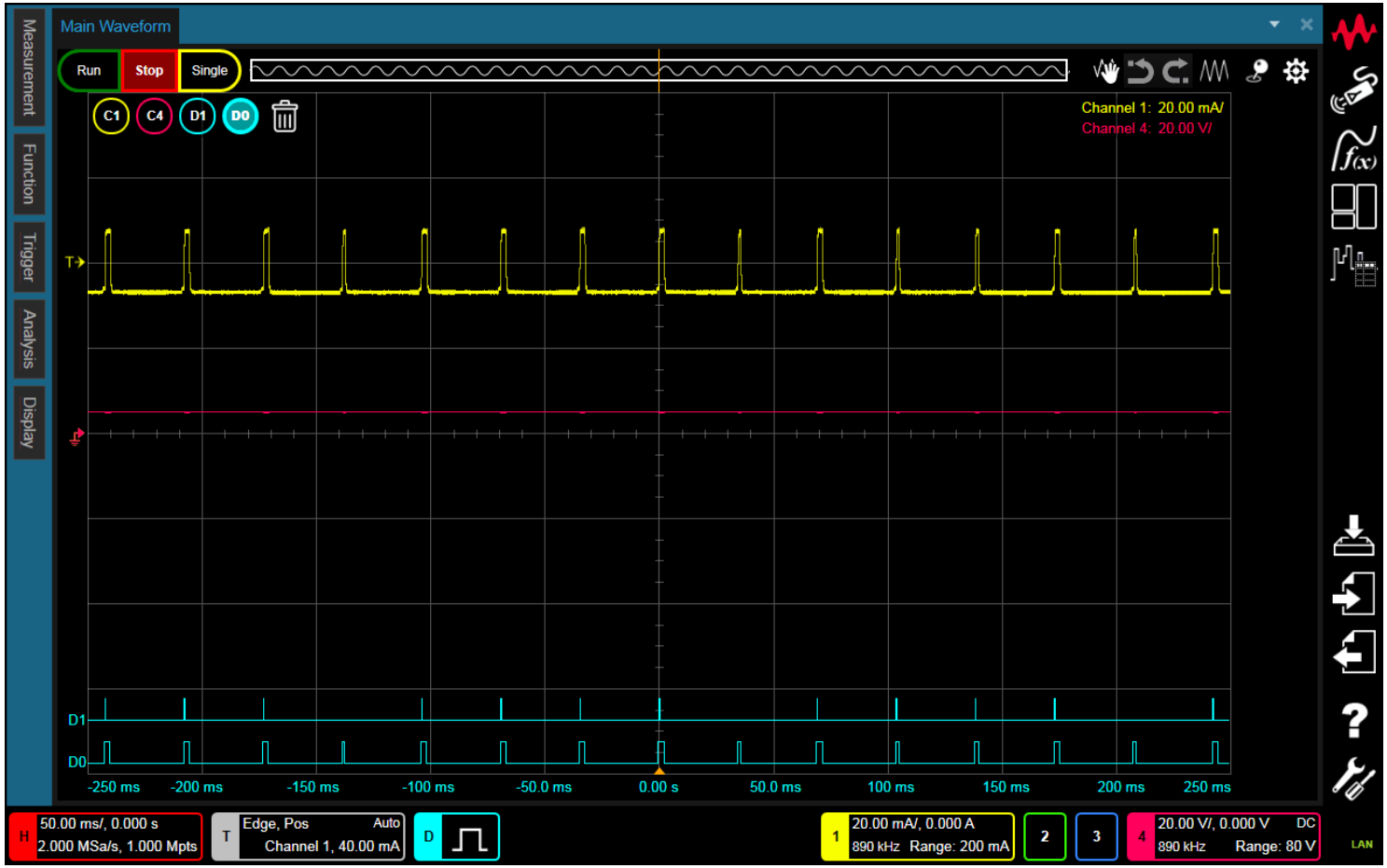
Arduino UNO を用いた検証(2)

- 悪意のある動作



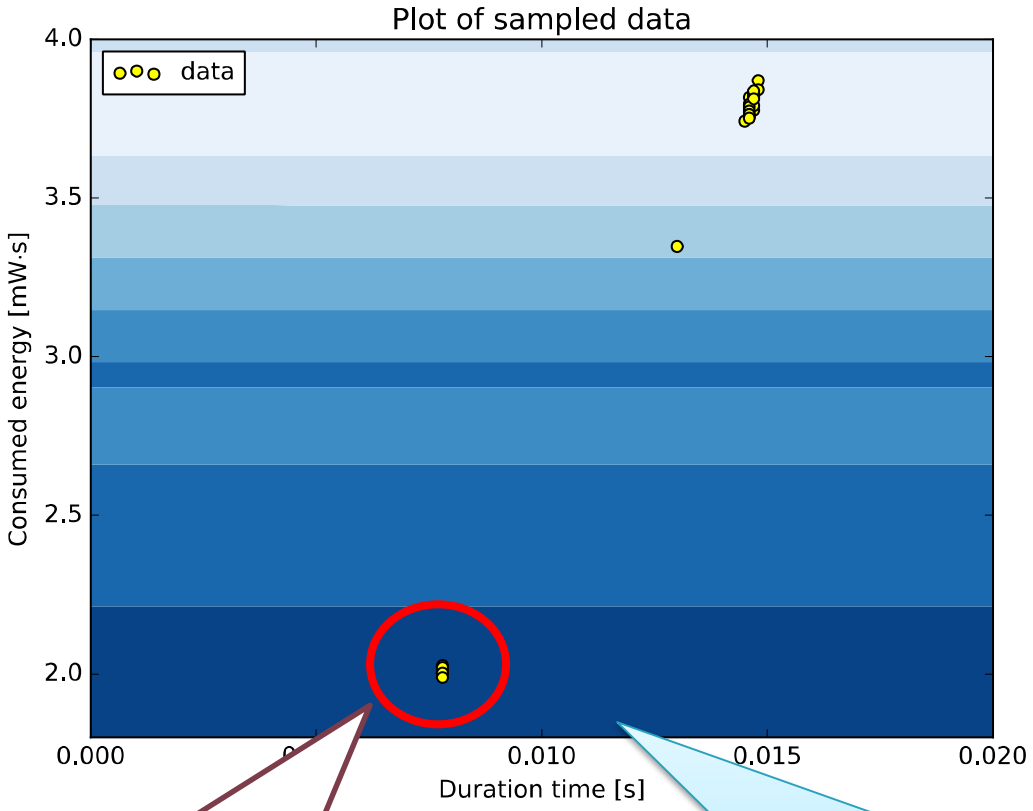
Arduino UNO を用いた検証(3)

- Step 1: 測定結果



Arduino UNO を用いた検証

- Step 5: 外れ値識別



濃い青の部分が外れ値

消費電力値と継続時間を使って，異常動作（暗号なし）を検知成功

ハードウェアチップ脆弱性検知の展望

- チップ脆弱性の検知には，ハードウェア（チップ）とセキュリティの知見の双方が必要
- チップ脆弱性の認知

  → ハードウェアは必ずしも安全でない

●継続的な**設計段階・製造段階のチップ脆弱性の検知の研究開発**が必要

- チップ脆弱性検知技術の早期の確立と実用化
  - 未知のハードウェアトロイや不正動作にいかに対応するか
  - 機械学習を用いることが解の一つか
- 検証フローの確立（認証フローも必要）