Identification of state registers of FSM through full scan by data analytics

He, Chengkang; Cui, Aijiao; Chang, Chip-Hong

2019

He, C., Cui, A., & Chang, C.-H. (2019). Identification of state registers of FSM through full scan by data analytics. Proceedings of the 2019 Asian Hardware Oriented Security and Trust Symposium (AsianHOST). doi:10.1109/AsianHOST47458.2019.9006677

https://hdl.handle.net/10356/145849

https://doi.org/10.1109/AsianHOST47458.2019.9006677

Identification of State Registers of FSM Through Full Scan by Data Analytics

Chengkang He*, Aijiao Cui* and Chip-Hong Chang†

*School of Electronic and Information Engineering, Harbin Institute of Technology (Shenzhen), China
†School of Electronic and Electrical Engineering, Nanyang Technological University, Singapore

Abstract—Finite-state machine (FSM) is widely used as control unit in most digital designs. Many intellectual property protection and obfuscation techniques leverage on the exponential number of possible states and state transitions of large FSM to secure a physical design with the reason that it is challenging to retrieve the FSM design from its downstream design or physical implementation without knowledge of the design. In this paper, we postulate that this assumption may not be sustainable with big data analytics. We demonstrate by applying a data mining technique to analyze sufficiently large amount of data collected from a full scan design to identify its FSM state registers. An impact metric is introduced to discriminate FSM state registers from other registers. A decision tree algorithm is constructed from the scan data for the regression analysis of the dependency of other registers on a chosen register to deduce its impact. The registers with the greater impact are more likely to be the FSM state registers. The proposed scheme is applied on several complex designs from OpenCores. The experiment results show the feasibility of our scheme in correctly identifying most FSM state registers with a high hit rate for a large majority of the designs.

Keywords—data mining; regression analysis; finite-state machine; decision tree

I. INTRODUCTION

Finite-state machine (FSM) is a mathematical model that characterizes the behavior of transitions among a finite number of states. It is the backbone for the design of controller for digital integrated circuits (ICs). As the core control unit, an FSM inside a chip usually coordinates the activities of various signals and implements specific operations. FSM design is usually implemented at higher architecture design level. After logic synthesis, the overall circuit design including the FSM unit is transformed to a netlist. Different abstractions and transforms applied in each design level forms a natural barrier to access the original FSM structure from the synthesized design or physical design. It is hence claimed to be computationally intractable to retrieve the FSM structure of a reasonably large design from the downstream design, such as the netlist [1]. Many security-oriented techniques rely on this argument to insert confidential copyright information into the original design by modifying the FSM design at the higher design abstraction level. The work in [2-4] proposed FSM-based dynamic watermarking schemes to protect the hard intellectual property (IP) core. In [2], some extra transitions are added into the FSM so that the output sequence of the FSM contains the specific information at designated positions under a given input sequence of excitation. The work in [5, 6] proposed the active hardware metering scheme to protect IC from being overbuilt by a foundry. A lock is inserted into the specification of a design function so that without a unique key to each chip, the chip cannot be unlocked. The lock of [5] is implemented by introducing some extra initial states and transitions between these states and the true initial state into the original FSM. A physical unclonable function (PUF) is adopted to determine which extraneous state is reached upon power-up. Only the designer is able to determine the key to enable the extraneous state to transit to the true initial state. As the untrusted foundry cannot retrieve the FSM structure from the physical design, the key cannot be deduced without knowing the extraneous states of the FSM. Similarly, the work in [7] proposed to combine the FSM design with PUF design to protect the field programmable gates array (FPGA) from being illegally copied. As the FSM design is hard to be extracted from the configuration bitstream, the key to unlock the design is inaccessible.

It is also noted that some fault injection [8, 9] attacks have been proposed to inject faults into the FSM to crack the cipher key that is used to encrypt sensitive information. The success of such attack relies on the fact that the FSM design structure is not kept secret. Although most encryption algorithms are open to the public, the FSM structure underlying each implementation is usually different and inaccessible. Under such situation, the proposed fault injection attack cannot succeed as expected.

Big data analytics have gained phenomenal successes in predicting trends and behavior of complex systems in recent years. This has motivated us to consider data analytic approach to retrieve the FSM structure based on large amount of data collected from the physical design. The scan chain for design-for-testability (DfT) has been widely adopted in modern digital IC design to facilitate the post-manufacturing IC testing. As all flip-flop outputs in a design including those from the FSM are reachable and their inputs are controllable in full scan design, FSM register related information can thus

This work was supported in part by the National Natural Science Foundation of China under Grant 61672182, the Guangdong Natural Science Foundation under Grant 2016A030313662, and in part by the Singapore Ministry of Education Tier 1 Grant MOE2018-T1-001-131 (RG87/18).

be collected from the scan design in a non-intrusive manner. Given this backdoor, coupled with the rapid development in statistics, artificial intelligence, machine learning and modeling technique, is it feasible to retrieve the FSM structure by data mining the scan data obtained through the scan design? We attempt to answer this question in this paper.

The FSM is composed of discrete states and the transitions between these states. The retrieval of FSM can be divided into two phases, identification of FSM registers and extraction of the state-transition graph (STG). The states information is stored across multiple interdependent registers. This paper focuses primarily on first phase of identifying state registers in FSM from all registers of a physical design. The decision tree method is adopted as the data mining scheme to determine the degree of dependency among all registers. To our best knowledge, this is the first data analytic attempt to recover the FSM design. Its preliminary success will challenge those security-oriented techniques based on the stealth of specific FSM registers, and increase the chance of success or efficiency of fault injection attacks on FSM of real chips in the field.

The rest of this paper is organized as follows. Section II introduces the percept of this proposition. A scheme to identify the FSM-related registers is described in Section III. The experimental results and analysis are presented in Section IV. Finally, Section V concludes this paper.

II. PERCEPT

An overview of the percept of this work is illustrated in Fig. 1. As a circuit design usually contains many registers, it is important to explore an attribute or a figure of merit that can be used to distinguish the FSM state registers from the non-FSM registers. As the core control unit, FSM manages the circuit operation by the transitions among multiple states. When the states vary, the design switches to a new state. In response to the new state, most registers will react directly which triggers the combinational circuits to switch and distribute the updated stimuli and data globally. Thus, the state variation of FSM registers can cause most other registers to switch their output more frequently consequently. On the other hand, the non-FSM registers can affect a fewer or even none of the registers most of the time. Thus, the dependency among all registers in a design can be considered for FSM register identification. Regression analysis is a statistical method which can determine the mutual dependency among two or more variants. It is hence adopted as the data mining method in this exploration.

Full scan design has been regarded as the best discipline of DfT as it can provide high controllability and observability for the circuit under test (CUT) [10]. In a full scan design, all registers, which include the FSM and non-FSM registers, are transformed to scannable flip-flops and connected in a chain, as shown in Fig. 2. With the scan chain, the status of all the registers in a working design can be controlled and observed. When the scan design of a chip can be accessed freely, sufficient data about register status under different input vectors can be collected from scan design for the regression analysis to discriminate FSM registers from most other registers based on the extracted attribute.

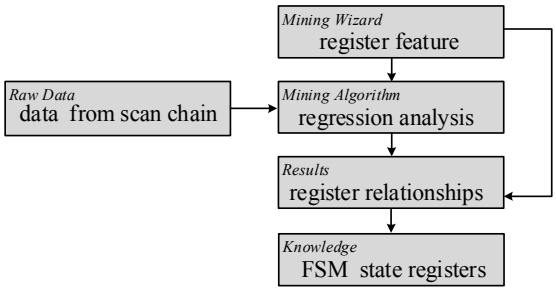


Fig. 1. Identification of FSM state registers by data analytics.

III. IDENTIFICATION OF FSM STATE REGISTERS

There are a few general assumptions in our method for the identification of FSM registers. First, the circuit is assumed to have full scan structure. Second, the scan design is not protected by secure scan design schemes such as [11] and [12]. This means that anyone who has access to the chip can freely perform the scan test with their own input stimuli. Third, except the chip designer, others have no knowledge about the circuit structure or design details.

A. Impact of register on state-transition

Suppose a design has K primary inputs $X = \{x_i\}_{i=1}^{K}$ and N registers $R = \{r_i\}_{i=1}^{N}$, of which M registers are state registers of the FSM. We can express the dependence of the input d_i of each register $r_i \in R$ on the present states of a set S_i of registers, where $S_i \subseteq R$ such that $d_i = F_i(S_i, Y_i)$, $Y_i \subseteq X$ and F_i is a combinational function. Since there are N registers in R, there will be N such sets $\{S_i\}_{i=1}^{N}$. A binary matrix $V = \{v_{i,j}\}$ can be constructed where $v_{i,j} = 1$ if $r_i \in S_j$ and $= 0$ otherwise. An impact factor can then be defined for each register $r_i \in R$ according to the number of registers in S_j that have their inputs influenced by the state of r_i.

$$\Im(r_i) = \sum_{j=1}^{N} v_{i,j} \tag{1}$$

The *impact* of a register expresses its strength of influence on the state transitions of other registers in a design. Therefore, a register with a greater *impact* is more likely to be a state register of an FSM. The set of M FSM registers can hence be distinguished from other registers of R based on

their *impacts*. The objective of data mining to determine the sets S_i for all $i = 1, 2, \ldots, N$.

B. Collection of scan data

Data mining begins from the collection of state values of all registers of the scan design continuously under normal working mode. As shown in Fig. 2, the full scan design consists of K primary inputs (*PIs*), L primary outputs (*POs*), a serial scan input (*SI*), a serial scan output (*SO*), a test control (*TC*) input and the clock input. The N flip-flops in the design are all scannable (SFF_1 to SFF_N) based on our assumption. To ensure that current states of the registers scanned out will be input to the combinational circuit to obtain the next states, *SO* is connected to *SI*. The timing diagram of some of these signals is illustrated in Fig. 3. Upon reset (*/RST* = '0'), the FSM is initialized to the starting state S_0. Then, with *TC* = '1' and the primary input $X = X_0$, the design advances to the next state S_1 on the first clock cycle after reset, and the states of its registers are updated accordingly. *TC* is then set low to scan out the state S_1 of all N registers from *SO*. Meanwhile, the new state S_1 is looped back into the registers of the scan chain through *SI*. After N clock cycles, *TC* is set high again with the values of the primary inputs X changed to X_1 for one cycle to advance the design to the next states S_2. The contents of all registers are updated accordingly. Then, *TC* is set low again to scan out the new states of the registers. This process is repeated until sufficient data are collected. To guarantee the completeness of the collected data, each time *TC* = '1', the *PIs* may be kept the same or updated with new set of values so that the FSM in the design experiences a sufficiently large number of state transitions under the same and different primary input combinations.

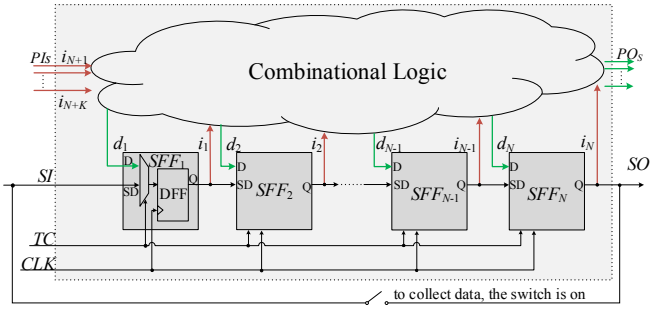


Fig. 2. Collection of scan data for regression analysis.

C. Regression analysis

1) *Regression model*

Regression analysis is adopted to explore the relationship among all registers. With reference to Fig. 2, when *TC* is high, the combinational logic of the design will compute according to the logic states of the set of input signals $\{i_i\}_{i=1}^{N+K}$ at that time, where $\{i_i\}_{i=1}^{N}$ are the present outputs $\{Q_i\}_{i=1}^{N}$ of all *SFFs*, and $\{i_i\}_{i=N+1}^{N+K}$ are the values of the K primary inputs $\{x_i\}_{i=1}^{K}$. The outputs of combinational logic are fed to the L *POs*, and the D inputs of the *SFFs*, denoted as d_i for all $i = 1, 2, \ldots, N$ in Fig. 2. With sufficiently large number of (D_i, I_i) tuples for all i, where $D_i = \{d_i\}_{i=1}^{N}$ and $I_i = \{i_i\}_{i=1}^{N+K}$, collected from the scan data, the impact $\Im(r_i)$ of each register $r_i \in R$ can be deduced to distinguish the FSM register and infer its input *function* $d_i = F_i(S_i, Y_i)$.

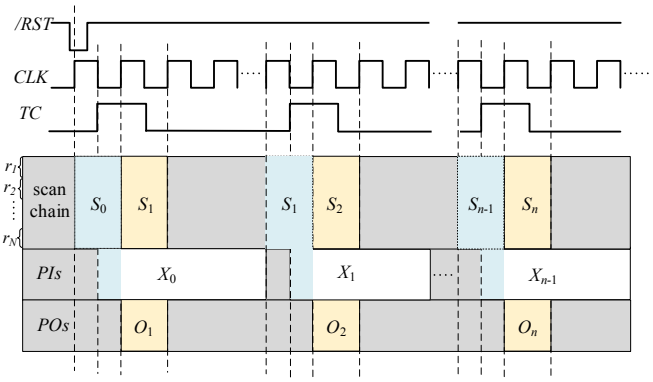


Fig. 3. Timing waveforms of data collection process.

2) *Decision tree algorithm*

To identify the subset of inputs $S_i \subseteq I_i$ for the input function d_i of each register r_i, decision tree algorithm is used as a regression analysis tool for mining the scan data. A decision tree is a data structure in which each internal node represents a "test" on an attribute, each branch represents the outcome of the test, and each leaf node represents a class label. In this paper, the terms tree and decision tree are used interchangeably. A tree is a collection of nodes. It has one *root* node, as shown in Fig. 5(a). The root node is located at *level* 1. Each node of a binary tree can be split into two children nodes according to a selected *attribute*. The level of a child node is one greater than that of its parent node. The node that has no child node is called a *leaf*. The maximum level of all leaves of a tree is called its *depth*. Fig. 4 shows the proposed decision tree algorithm for the mining of scan data.

```
DecisionTreeAlgorithm(data, strategy) {
    decisionTree= InitialDecisionTree(data);
    while (not strategy.meetEndConditions()) {
        splitOneNode(decisionTree, strategy);
    }
    return decisionTree
}
```

Fig. 4. The decision tree algorithm.

In the decision tree algorithm of Fig. 4, information *entropy* is used as the main criteria for the splitting operation. *Entropy* is a statistical measure of the uncertainty of information. A

smaller *entropy* indicates a more deterministic information. In this scheme, the *original entropy* due to a signal *sig* in a data set *dat* can be defined as:

$$E(sig, dat) = -[p \log_2 p + (1-p)\log_2(1-p)] \quad (2)$$

where p denotes the probability that *sig* = '1' in *dat*. The data set *dat* can be split into two data subsets *dat*(*attr* = 0) and *dat*(*attr* = 1) according to a selected attribute *attr*.

The reduction of entropy on splitting a node can be evaluated by comparing the *entropy* before and after a node is split by the selected attribute *attr*. The post-entropy of splitting a node can be computed by:

$$E^*(sig, dat, attr) = \frac{\lambda_0}{\lambda} E(sig, dat(attr = 0)) + \frac{\lambda_1}{\lambda} E(sig, dat(attr = 1)) \quad (3)$$

where $\lambda_0 = |dat(attr = 0)|$, $\lambda_1 = |dat(attr = 1)|$ and $\lambda = \lambda_0 + \lambda_1$.

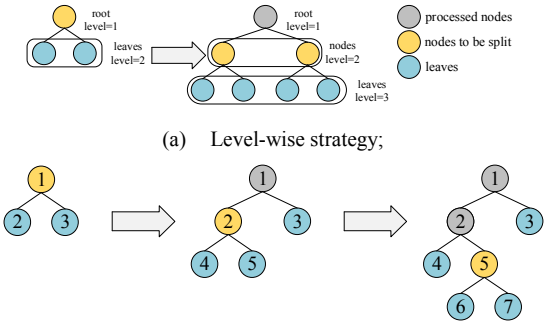
The *gain function*, which is the entropy reduction due to splitting *dat* by *attr* can be computed by

$$G(sig, dat, attr) = E(sig, dat) - E^*(sig, dat, attr) \quad (4)$$

An arbitrary register $r_i \in R$ is considered for the illustration of the construction of its decision tree. Starting from a root node that holds all collected data D, the decision tree algorithm iteratively constructs a regression model by finding the most likely node splitting attributes to approximate the arguments of the register input function $d_i = F_i(S_i, Y_i)$, where $S_i \cup Y_i \subseteq I_i$. The information entropy $E(d_i, D)$ of d_i in D before splitting can be computed by (2). Each input $i_j \in I_j$ is a possible attribute for splitting the node. The data held by the node can be split into two subsets of data, $D(i_j = 0)$ and $D(i_j = 1)$. The *post entropy* $E^*(d_i, D, i_j)$ of each test attribute i_j is computed by (3), and its *gain* $G(d_i, D, i_j)$ can be computed according to (4). After all possible attributes have been tested, the input, assume i_k, with the maximum *gain* of g_{max}, is selected as the attribute of the root node, which has a *node gain* g_{max} . Then, the root node is split into two children nodes that hold the data sets, $D(i_k = 0)$ and $D(i_k = 1)$, respectively.

After splitting the root node, there are two general strategies, level-wise and leaf-wise, to further split the nodes. In level-wise strategy, both children in the current level will be selected for splitting. As shown in Fig. 5(a), according to the above process, the root node is first split into two nodes and then the two nodes in level 2 are split into four nodes in level 3. However, for those nodes that have much lower *node gain*, their splits contribute marginally to a good regression model. Thus, level-wise strategy tends to incur higher redundant computations. In leaf-wise strategy, priority is given to the leaf node with the larger *node gain*. Leaf-wise strategy searches all leaves and select the one with the maximum *node gain* to split. As shown in the example of Fig. 5(b), node 2 has a greater *node gain* than node 3 and it is selected for splitting into node 4 and node 5. Similarly, node 5 has the highest *node gain* among the leaf nodes 3, 4 and 5 and it is chosen to split further. Compared with level-wise strategy, leaf-wise strategy constructs a more precise regression model for the same number of splitting operations. It is also noted that each register input function is dependent on a few inputs of I_i, i.e., S_i is a smaller subset of I_i. It means that most inputs of I_i lead to zero *gain* in splitting the leaf nodes in this process. For these reasons, leaf-wise strategy is used for node splitting.

This process of splitting selects node based on an attribute that gives the maximum gain is repeated until no more leaf nodes can be split. When all leaves have zero node gain, none of the leaves can be split further. If the tree is terminated by this conservative criterion, the constructed model has a tendency to be overfitted. An overfitted model increases the *impact* of non-FSM registers more than they should reasonably possess, which leads to spurious FSM registers. To overcome overfitting, some pruning strategy [13] is applied in the process of tree construction. Four parameters, namely the maximum depth (h_{max}), the maximum number of leaves (l_{max}), the minimum amount of data held by a leaf (D_{min}) and the minimum gain (G_{min}), are used for pruning and they are determined empirically. Aggressive pruning can result in underfitting as some qualified attributes may be pruned along with the non-qualified ones. In comparison, underfitting provides a set of attributes that can better approximate the true relationship than overfitting. Hence, the parameters for pruning are selected to avoid overfitting than underfitting if a trade-off is inevitable. The leaf node cannot be split when the specified limit of any of these parameters is reached. When no more leaf node can be split, the decision tree for the register r_i is completed. The attributes of all the non-leaf nodes in this tree can be identified as the arguments of F_i for the input of register r_i.



(a) Level-wise strategy;

(b) Leaf-wise strategy.

Fig. 5. Examples of level-wise and leaf-wise strategies.

3) *Identification of FSM registers*

With the decision trees of all registers constructed, a table of correlation analogous to the adjacency matrix in graph theory can be constructed as shown in Fig. 6. Each

column in this table denotes a unique input and each row denotes a register input function $d_i = F_i(S_i, Y_i)$. According to the tree constructed for register r_i, the attribute i_j of every non-leaf node is an element of the input arguments of F_i, $i_j \in S_i \cup Y_i$. Hence, the entries in row d_i and column i_j are equal to '1' and all other entries in this row are '0'. For example, in Fig. 6, for the tree of register r_1, node 1 is split by the attribute i_N and node 2 is split by i_2, so the entries in columns i_N and i_2 of row d_1 are filled with '1' and all other columns of row d_1 are filled with '0'. The column sum i_j for $j \in [1, N]$ in the correlation table is the *impact* for the register r_j according to (1). The registers can then be sorted in descending order of *impact*. The first $w = \alpha + \beta N$ highest *impact* registers will be shortlisted to further evaluate their input function before confirming if they are FSM registers, where α is an integer and β is a fraction of correlated registers. They can be estimated based on the knowledge of the design functionality. α can be set a little larger if design functionality is unclear. Typically, $\beta N >> \alpha$. As a useful rule of thumb, α and β can be set to be smaller than 10 and 0.05, respectively for a reasonably large design.

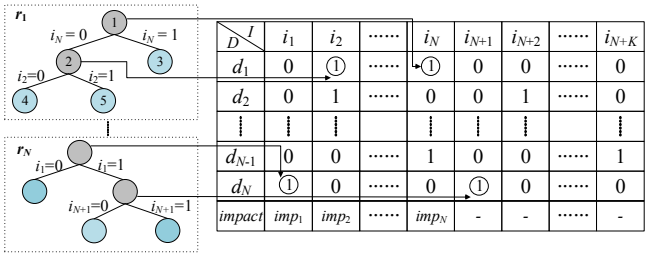
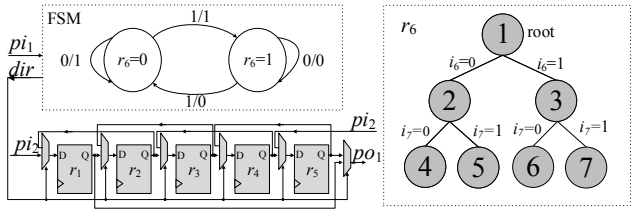


Fig. 6. The correlation table for computing the *impact* of each register.

4) *An illustrative example*



(a) The example design; (b) The built tree for register r_6.

Fig. 7. Building decision tree for an example design.

A simple design in Fig. 7(a) is used to numerically demonstrate the above calculation. The design contains two *PIs*, pi_1 and pi_2, a bidirectional shift register with five registers, $\{r_1, r_2, ..., r_5\}$, and an FSM with one state register r_6. The FSM is used to control the direction of shift. Suppose all six registers are initialized to '0'. 50 groups of scan data are collected from the scan chain according to Section III.B. According to the Section III.C, 50 pairs of $\{i_i\}_{i=1}^{8}$ and $\{d_i\}_{i=1}^{6}$ are obtained. The parameters for pruning, h_{max}, l_{max}, D_{min} and G_{min} are set to be 4, 6, 2 and 0, respectively. The decision tree algorithm of r_6 is constructed as an example. Based on the procedure described in Section III.C, the *gains* due to splitting each node by different attributes are listed in Table I. For the first node, the attribute i_6 has a maximum *gain* of 0.49 and hence node 1 is split by i_6. Similarly, node 2 and node 3 are split into node 4 and 5, and node 6 and 7 respectively by i_7. As the maximum *gain* for the nodes 4, 5, 6, and 7 are 0, G_{min} is reached for all leaf nodes. The final tree for r_6 is shown in Fig. 7(b) and the *node gain* are highlighted in red in Table 1.

TABLE I. COMPUTATION OF GAIN FOR EACH NODE WITH RESPECT TO EACH ATTRIBUTE

G \ node	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8
1	0.00	0.00	0.00	0.02	0.01	0.49	0.01	0.04
2	0.03	0.03	0.02	0.02	0.02	0	0.16	0.02
3	0.05	0.01	0.06	0.2	0.03	0	0.47	0.2
4	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0

The decision tree for the other five registers can be similarly built. The correlation table can be constructed from these six trees as shown in Table II. From the column sums of table II, r_6 has the greatest *impact*. As the example design is small, r_6 is identified as the only register.

TABLE II. RELATIONSHIP TABLE FOR THE EXAMPLE DESIGN

D \ I	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8
d_1	0	1	0	1	0	1	0	1
d_2	1	0	1	0	0	1	0	0
d_3	0	1	0	1	0	1	0	0
d_4	0	0	1	0	1	1	0	0
d_5	0	0	0	1	0	1	0	1
d_6	0	0	0	0	0	1	1	0
impact	1	2	2	3	1	6	-	-

IV. EXPERIMENTAL RESULTS AND EVALUATION

In the experiment, we apply our proposed scheme on seven complex designs from OpenCores [14], such as the CPU design "mips789", encryption modules "apbtoaes128" and "aes_hl", system controllers "sdram_16bit", etc. We use the *modelsim* or *iverilog* [15] simulation tools to simulate the testbench file of each source design to collect the test data. The collected data is fed to the Python program of the decision tree algorithm. The experimental results are shown in Table III.

In Table III, the columns '*#PIs*' and '*N*' denote the number of *PIs* and registers in the original design, respectively. The column '*M*' denotes the actual number of FSM registers. The column '*CDR* (%)' denotes the correct detection rate in percentage. CDR refers to the ratio of the number of correctly identified FSM registers to actual number of FSM registers. The column '*HR*(%)' denotes the hit rate which is computed as the percentage of shortlisted registers

that are actual FSM registers. Ideally, both HR and CDR should be 100% but this is impractical to achieve. A more feasible target is to achieve a high CDR with a high hit rate. Our current strategy is to keep α small and β large for small design an vice versa for large design. For the five smaller designs, α and β are set as 4 and 0.05, respectively. For larger designs, they are set as 8 and 0.01, respectively.

TABLE III. THE EXPERIMENTAL RESULTS

design	#PIs	N	M	$Q = \alpha + \beta N$		M/N (%)	HR (%)	CDR (%)
				$\alpha = 4$, $\beta = 0.05$	$\alpha = 8$, $\beta = 0.01$			
mips789	131	756	4	-	15	0.53	26.7	100
apbtoaes128	41	910	7	-	16	0.77	43.8	100
aes_hl_key_exp	261	410	2	-	12	0.49	16.7	100
sdram_16bit	89	116	4	9	-	3.44	44.4	100
simple_spi	16	140	2	12	-	1.43	16.7	100
sdr_ctrl_xfr	146	144	2	9	-	1.39	22.2	100
sdr_ctrl_bank	64	80	3	8	-	3.75	37.5	67
sdr_ctrl_req	50	84	2	8	-	2.38	25.0	50
pid_ctrl(oh)	53	397	10	-	12	2.52	83.3	70
pid_ctrl(bin)	53	391	4	-	12	1.02	33.3	100

We can see that for the first six designs, our method can achieve 100% *CDR*, which means that all the FSM registers are identified successfully. Not all the FSM registers in "sdr_ctrl_bank", "sdr_ctrl_req" and "pid_ctr(oh)" are identified. From the collected scan data, it is found that the outputs of one FSM register in "sdr_ctrl_bank" and "sdr_ctrl_req" and two FSM registers in "pid_ctr(oh)" have no switching. These registers have zero *gain* and they will never become the decision attributes of any decision tree. Their *impacts* are zero and then therefore cannot be identified. Due to these registers, the three design has a *CDR* of only 67%, 50% and 70%. In reality, if the fraction of '1's or '0's outputs collected from a register is smaller than 0.01, the *impact* of the register is very low and cannot be identified successfully. It is also noted that the FSM of "pid_ctr(oh)" uses one-hot encoding, which means that each register of an FSM represents one unique independent state. By re-encoding the 10 states of this FSM design in binary using 4 registers instead, 100% CDR can be achieved, as shown in the last row "pid_ctr(bin)" of Table III.

V. CONCLUSION

FSM has been widely applied in most modern IC designs. Large FSM with the number of states increases exponentially with the number of registers can be readily implemented at the higher level of design abstraction but recovering its state transition graph from the physical design is computationally intractable. This assumption has been leveraged by many security-oriented designs. This work proposes a data mining technique to analyze the scan data to identify as many FSM registers as possible and hence approximate a large part of their input function (a.k.a next state decoder). By exploring the correlation among registers by a decision tree construction, it is shown that it is feasible to discriminate most FSM registers from other registers based on their stronger influences on the output states of other registers. Despite using a rudimentary thresholding, our experimental results show that the proposed method can effectively shortlist a small fraction of registers and correctly identify most FSM state registers. Our future work will focus on a more analytical criterion to shortlist candidates instead of based primary on number of registers in a design to increase the hit rate. A more structured, precise and efficient retrieval of the state transitions from the correctly identified registers will also be explored.

REFERENCES

[1] M. Pandey, A. Jain, R. E. Bryant, D. Beatty, G. York and S. Jain, "Extraction of finite state machines from transistor netlists by symbolic simulation," in *Proceedings of ICCD '95 International Conference on Computer Design. VLSI in Computers and Processors*, Austin, TX, USA, 1995, pp. 596-601.

[2] A. Cui, C. H. Chang and S. Tahar, "A robust FSM watermarking scheme for IP protection of sequential circuit design," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 30, no. 5, May 2011, pp. 678-690.

[3] A. L. Oliveira, "Techniques for the creation of digital watermarks in sequential circuit designs," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 20, no. 9, Sep. 2001, pp. 1101-1117.

[4] I. Torunoglu and E. Charbon, "Watermarking-based copyright protection of sequential functions," *IEEE J. Solid-State Circuits*, vol. 35, no. 3, Feb. 2000, pp. 434-440.

[5] Y. Alkabani and F. Koushanfar, "Active Hardware Metering for Intellectual Property Protection and Security," in *Proc. 16th USENIX Security Symposium*, Boston, USA, Jan. 2007, pp. 291-306.

[6] Y. Alkabani, F. Koushanfa, and M. Potkonjak, "Remote activation of ICs for piracy prevention and digital right management," in *Proc. IEEE/ACM Int. Conf. on Computer-Aided Design (ICCAD)*, San Jose, CA, USA, Nov. 2007, pp. 4-8.

[7] J. Zhang, Y. Lin, Y. Lyu, and G. Qu, "A PUF-FSM binding scheme for FPGA IP protection and pay-per-device licensing," *IEEE Transactions on Information Forensics and Security*, Vol. 10, no. 6, 2015, pp. 1137-1150.

[8] A. Nahiyan, K. Xiao, K, Yang, Y. Jin, D. Forte, M. Tehranipoor, "AVFSM: A framework for identifying and mitigating vulnerabilities in FSMs," in *2016 53nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, Austin, TX, USA, June 2016, pp. 1-6.

[9] V. S. Rathor, B. Garg, G. K. Sharma, "An Energy-Efficient Trusted FSM Design Technique to Thwart Fault Injection and Trojan Attacks," in *2018 31st International Conference on VLSI Design and 2018 17th International Conference on Embedded Systems (VLSID)*, Pune, India, Jan. 2018, pp. 73-78.

[10] M. L. Bushnell and V. D. Agrawal, Essentials of Electronic Testing for Digital, Memory and Mixed-signal VLSI Circuits. Norwell, MA, USA: Kluwer, 2000, pp. 456-485.

[11] A. Cui, C. H. Chang, W. Zhou, Y. Zheng, "A New PUF Based Lock and Key Solution for Secure In-field Testing of Cryptographic Chips," *IEEE Transactions on emerging topics in computing*, accepted.

[12] A. Cui, Y. Luo and C. H. Chang, "Static and dynamic obfuscation of scan data against scan-based side-channel attacks," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 2, Feb. 2017, pp. 363-376.

[13] P. Salembier and L. Garrido, "Connected operators based on region-tree pruning strategies," in *Proceedings 15th International Conference on Pattern Recognition. ICPR-2000*, Barcelona, Spain, 2000, pp. 367-370.

[14] https://opencores.org/projects

[15] http://iverilog.icarus.com/