Non-Invasive Reverse Engineering of CMOS Integrated Circuits

M. Brutscheck, B. Schmidt, M. Franke, A. Th. Schwarzbacher, and St. Becker

***Abstract* — In this paper a novel non-invasive procedure will be presented to determine nonlinear binary multi-input multi-output (MIMO) integrated circuits (ICs) solely by their input-output behaviour. The algorithm identifies unknown CMOS ICs through the abstraction of traditional automata theory. The entire determination procedure was simulated and fully tested on IEEE ISCAS benchmark models as well as user defined models of real ICs. The results obtained will be presented in this paper. For every circuit analysed the function has been successfully determined by the novel identification procedure proposed.**

***Keywords* — Deterministic Automata, Non-Invasive Reverse Engineering, Unknown CMOS ICs.**

I. Introduction

Until now the investigation of unknown CMOS integrated circuits is an important part in reverse engineering. However, several destructive [1], [2] and non-destructive procedures have been developed [3], [4] to identify the internal function and structures. Current ICs consist of very complex structures with a great variety of functions and different behaviours. Since these functions are not always known it can be essential to correctly determine their behaviour. This is required when the label is lost or it is necessary to find out more about the internal structure of the integrated circuit. Furthermore, it is conceivable to use structures of discontinued ICs in new IC designs or to add new functionality to an existing system.

To make a structured analysis of these ICs possible the overall analysis must be divided into different parts. The determination of pin types is the first analysis step which was described in detail in [6]. This is followed by a preliminary investigation of the IC under test which results in combinatorial, sequential linear or sequential nonlinear behaviour as described in [7]. Here, it was demonstrated that a real IC can be abstracted using the model of automaton [8]. A large number of unknown ICs have a nonlinear behaviour. Therefore, this paper will discuss the specific problem of the identification of unknown nonlinear CMOS ICs represented by sequential deterministic finite state machines. The overall identification procedure consists of three parts. The separation into Moore or Mealy automaton will be described in Section II. Afterwards, the preparation algorithm will be explained in Section III, while the identification algorithm will be described in detail in Section IV before the results are presented in Section V.

M. Brutscheck is with the Dublin Institute of Technology, Ireland (e-mail: brutscheck@gmx.net).

II. Separation into Moore or Mealy Automata

The separation into Moore or Mealy is the first step of the overall identification procedure. It is an important improvement of this novel procedure compared to other methods to significantly simplify the following analysis steps. The different behaviour of Moore and Mealy automata is used for a general classification. The output of a Moore automaton only depends on the internal states. Additionally, if the storages and the inputs are connected by combinatorial circuitry the automaton is of type Mealy.

The test run is started while a random input word is applied and then a clock pulse is given to the circuit under test. After this step, the first input word (‘0’) is applied to the automaton and the resulting output word is stored. In the following loop all other input words are applied to the circuit while no clock pulse is applied. The output words which appear are compared to the stored one and in case of any differences the automaton is classified as a Mealy automaton and the procedure is finished. If all output words are identical then the next random step is made until the maximum number of cycles is reached. If the last cycle is executed and no variance in output words was found, the automaton will be considered as a Moore automaton during the following analysis.

III. Preparation Algorithm

After the type of the automaton was determined the identification algorithm is prepared by several process steps. The identification procedure must firstly find one initial state of the automaton. In the simplest case the initial state can be reached by a reset pin control. In other ICs the reset can be carried out by disconnecting the power supply. However, if such a reset capability does not exist an initial state can also be found using suitable process steps by applying input combinations which will be not described in this paper. After an initial state is found the preparation can be carried out.

First, the information about the input-output words (OWs) or input-output word combinations (OWCs) are gathered. These sets are recorded as shown in Equation (1).

$$\begin{array}{l}\{1\}\ \ Moore: OW\,(t-1);IW\,;OW\,(t)\\ \{2\}\ \ Mealy:\ OWC\,(t-1);IW\,;OWC\,(t)\end{array} \qquad (1)$$

If the analysis has identified a Moore automaton set {1} is used. Set {2} is used in the case of a Mealy automaton. The following steps and parts of the algorithm are explained for a Mealy automaton and are carried out in an analogous manner for Moore automata. However, for a

Moore automaton only a single output word is processed instead of all output word combinations.

The different states are separated relative to their detected differences in their output behaviour before the identification algorithm is started. Therefore, random input words are applied and a clock pulse is applied afterwards. In regular intervals a reset is applied to resettable automata to restart the algorithm from initial states. The previous as well as the following output word are recorded at each step. Additionally, the input word which has caused the step is stored. Repeating combinations of these three values are not saved. However, all differences are collected using the described procedure. At always the first *OW(t*-1*)* the change of the *OW*(*t*) to multiple applications of different input words is investigated and is used for the result. If the current output word has two or more following output words when the same input word is applied then the number of these output words relates to the minimum number of states which share the first output word. This is valid for deterministic automata.

Each information set as illustrated in Equation (1) is checked if it has already occurred. If it has not it is added to the current list. The number of output words found is stored. The number of states, the output words, the input words and the type of the automaton are the basic information. The combinations are checked for their first output word. Due to their order each alteration implies a new output word. If entries exist where at any time input words and output words are equal but the following states are different, then the list is rearranged. The detection of such entries is proof that a minimum of two states exists with the same output word. The input word, which causes most output words following a particular output word, is labelled as most significant input word (MSIW). Again, there is no targeted search for an output word which means that the gathered output words are caused by the randomly applied input words. Furthermore, this number is the number of detected distinguishable states. With the help of the most significant input words it is possible to separate states that have the same output word and the same input word is applied but the following output word is a different one. For instance, if there would be three such entries there will be at least three states related to this output word.

After all entries are made the information is interpreted. Therefore, the data structure is reduced to the most significant input words and their significances. Here, significance means the number of expected states when applying the related most significant input word. The total number of found differences forms the number of securely distinguishable states. This means, that it is possible to compare two sets of *OW*(*t*-1); *IW*; *OW*(*t*) for a Moore automata or *OWC*(*t*-1); *IW*; *OWC*(*t*) for a Mealy automata there is no difference in the actual sets but only in the result of the investigation. The use of a number of distinguishable states severely reduces the necessary investigation depth of the integrated circuit. Equation (2) shows the calculation of the number of distinguishable states (NoDS). If this number is equal to the real number of states, then the automaton can be identified directly. Otherwise, it is not possible to determine the automaton in only one step.

$$NoDS = \sum_{MSIW} significance \qquad (2)$$

The classification into Moore or Mealy automata as well as the consideration of the number of distinguishable states are important parts of the analysis procedure. The preparation results are used to fully solve such problems through the identification algorithm which will be described in the next section.

IV. Identification Algorithm

After the initial investigation of the unknown IC the identification algorithm is carried out which is the major part of the analysis procedure for nonlinear FSMs. It consists of several blocks and works similar for Moore and Mealy automata. Figure 2 schematically shows the identification algorithm.

First, the required length of the investigation tree is determined. This state tree length is important to record the state transitions and to afterwards correctly identify the unknown IC. After the determination of the tree length the IC under investigation is checked if a reset capability exists or an entry point can be determined. Then the algorithm queries the solution type. These are the fast or the slow identification. Basically, both the fast and the slow analysis are equivalent. Usually, the IC under test is analysed using a fast identification. However, in case of insufficient RAM it is not possible to process the algorithm using the fast identification. Therefore, it automatically switches to the slower solution which uses less memory but requires more evaluation time.

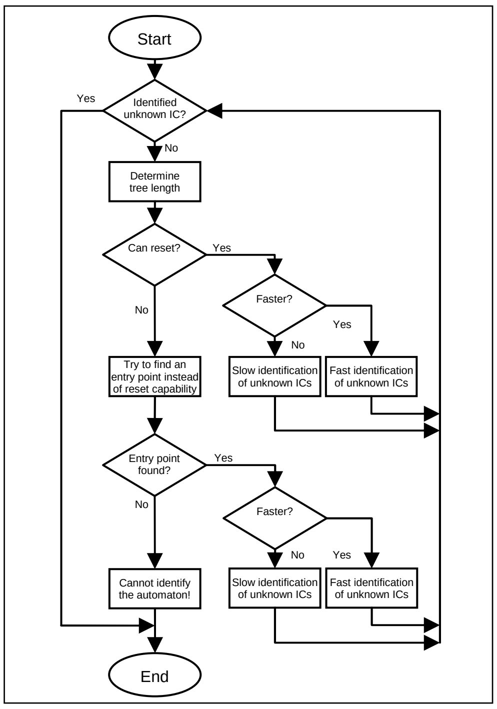
Start
Identified unknown IC?
Yes
No
Determine tree length
Can reset?
Yes
No
Faster?
Yes
No
Try to find an entry point instead of reset capability
Slow identification of unknown ICs
Fast identification of unknown ICs
Entry point found?
Yes
No
Faster?
No
Yes
Cannot identify the automaton!
Slow identification of unknown ICs
Fast identification of unknown ICs
End

Fig. 2. The Identification Algorithm.

The maximum number of states needs not to be known to proceed with the algorithm. In most cases the number of states can be calculated using several iterations. The initial value can be either given by the user or is determined from the number of distinguishable states (NoDS). If the real number of states is not known the number of distinguishable state for the initial value can be calculated as $NoS = NoDS + 2$. The added two is based on the fact that it is the number of guessed and not identified states. This number can be chosen in a free range. If a higher value is chosen the likelihood increases to find distinguishable states in each identification cycle which would previously have not been detected. At the same time the investigation complexity increases. If the addend 2 is too high this advantage could be lost. The iteration to the real number of states is carried out after each cycle of the identification algorithm. Again, the addend two is a selectable value which function was previously described. The idea behind the identification of unknown ICs is similar to the general classification of states using the data prepared in the previous determination of the number of distinguishable states. Using deterministic automata a state has to have the same response at the output caused by the same input word which means a jump to the following state. If two states differ in their internal bit combination but always respond equally at the outputs then the algorithm identifies these states as only one state. With this, the automaton is not only identified but also reduced. Several states can share the same output word. This is valid for all output words. Therefore, it is possible that all output words of two states are identical without any redundancy. For a final distinction of states their state trees are investigated. A state tree contains information of particular output words which are causes by the respective input word applied. As previously described the evaluation of the following output word is an adequate further distinctive feature. Therefore, all following output words (FOWs) of the previous final points are also gathered. This classification is continued until the significance of the trees is sufficient to clearly separate occurring states. Traditional solutions require the knowledge of the maximum number of states [3]. This is an essential disadvantage as the maximum number of states is not available in practice. However, the restriction to resettable automata or automata with a definable entry point provides the possibility to determine the number of states using an iterative approximation without any knowledge of the real number of states. More precisely if the initial value is predefined then the solution of the investigated unknown automata is found faster. As previously described the initial value can be either given by the user or is derived from the number of distinguishable states. In this case the number of distinguishable states represents the minimum number of states. A predefined number of states is added to this number of distinguishable states. From this predefined number it is expected that many other similar states exist, which are not distinguishable by only one step. The length of the state trees is calculated as illustrated in Equation (3).

$$length_{tree} = NoS - NoDS + 2 \qquad (3)$$

The added two is based on the fact that two output words form a pair at the rough classification of states. Here, a preliminary reduction is possible because the states are compared in relation to their current output word as well as their following output word. For a general discrimination two output words are required at time t and at $t + 1$ after one step. Here, the maximum tree length is detected if either a difference occurs or the states are identical or redundant. Each state found can be reached again, because the path from the initial state to the related state is recorded if the state is identified as a new state. The states in these trees will only together cause the same output word if the states which the trees belong to are identical. If they are not identical then there will be discrepancies in the output words that render it possible to identify the different states. Once again the following output words of the last branch of both trees are compared and the discrepancies are found. Two states are not redundant if they differ in at least one following state regardless if this is distinguishable from the output word or not. Passing through the complete state trees a difference to all other states will occur due to this condition. Each state found can be retrieved because the path from the initial state to the considered state is saved if it is recognised as a new state. If the state has to be reached again the automaton must be reset. From the initial state the desired state is retrieved. The state tree is recorded while passing from the current investigated state to the end of the tree. During this process all output words are saved. If the end of the branch is reached the automaton is reset and the next branch is investigated. By applying all possible input words step-by-step a tree is subsequently generated. Each tree consists of a number of branches, where each branch differs in a minimum of one input word. Beginning with the initial state each state is investigated towards its following states. Whenever a new state is found the related output word, the tree and its position relative to its initial state is recorded. For each state and for each input word the following state is determined. The different output words of each state tree are stored in an array.

The required length of the array is calculated as in (4).

$$length_{treearray} = \sum_{n=0}^{length_{tree}-1} NoIW^{\,n} \qquad (4)$$

Each output word in this array is controlled by a sequence which is stored in another path array. The position of the output word can be determined from Equation (5).

$$\begin{aligned} position_0 &= 0 \\ position_n &= \left[patharray(n) + 1 \right] + position_{n-1} \times NoIW \end{aligned} \qquad (5)$$

At the position where n is equal to the length of the path array the wanted output word is located. Because of this structure the input words need not to be stored. Therefore, its tree is recorded and compared with all already recorded trees. If the tree is identical then the previously recognised state is entered as the following state. Otherwise, a new state is generated and recorded as the following state. As soon as the last found state is investigated towards its last following state the automaton is fully determined.

V. Results

The theory presented in this paper was verified using both simulation and real hardware tests. The IC models [5] were analysed having unknown as well as known number of internal states using MATLAB [9]. The following tables will show the results of the simulation and the hardware analysis of the nonlinear identification procedure. Furthermore, for each model the result with unknown as well as known number of states is shown.

TABLE 3 first shows the results where NoFS is the number of states found. Moreover, the state transition table is STT and the output function is represented by OF. Here, the number of internal states of the IC models to be investigated is unknown.

TABLE 3: RESULTS OF HARDWARE ANALYSING USING UNKNOWN NUMBER OF STATES.

IC Name	Type of FSM	FSM Found	NoS	NoFS	STT Found?	OF Found?	Evaluation Time
EC1	Mealy	Mealy	1	1	yes	yes	7713,6s = 2,14h
ELS1	Mealy	Mealy	8	8	yes	yes	62110,0s = 17,25h
ENLS1	Mealy	Mealy	8	8	yes	yes	62127,0s = 17,26h
S27	Mealy	Mealy	5	5	yes	yes	615580,0s = 7,12d
B06	Moore	Moore	13	13	yes	yes	11456s = 3,18h

As can be seen in TABLE 3 for each benchmark circuit the presented algorithm found the correct type of IC. The evaluation time in the right column shows that about one week is needed to identify the complex benchmark S27. The other IC models can be determined in less than a day. However, it is even possible to identify the expected state transition table as well as the correct output function. TABLE 4 presents the simulation results using unknown number of internal states of the IC models to be investigated.

TABLE 4: RESULTS OF SIMULATION USING UNKNOWN NUMBER OF STATES.

IC Name	Type of FSM	FSM Found	NoS	NoFS	STT Found?	OF Found?	Evaluation Time
EC1	Mealy	Mealy	1	1	Yes	yes	13,6s
ELS1	Mealy	Mealy	8	8	Yes	yes	73,4s
ENLS1	Mealy	Mealy	8	8	Yes	yes	70,7s
S27	Mealy	Mealy	5	5	Yes	yes	636,2s
B06	Moore	Moore	13	13	Yes	yes	17,2s
C17	Mealy	Mealy	1	1	Yes	yes	2378,7s = 39,6min

TABLE 4 shows that the algorithm found the correct type for all unknown ICs under investigation. Moreover, the correct number of states was also always found. Hence, the correct state table as well as the correct output function were in all cases successfully determined. However, the algorithm introduced was developed to analyse nonlinear FSM. As can be seen from TABLE 4 combinatorial as well as linear sequential FSM can also be identified using the novel algorithm. Furthermore, the evaluation time in the right column shows that the simulation is accomplished within less than an hour for even complex circuits. In the case that the exact number of states is known both the hardware analysis shown TABLE 3 as well as the simulation as presented TABLE 4 was used. Therefore, the same implementations were analysed with known number of states instead the initial number of states equal to zero. First, TABLE 5 presents the hardware models analysed using the known number of states. From TABLE 5 it can be seen that in each case the nonlinear detection algorithm found the correct type of the unknown IC.

TABLE 5: RESULTS OF HARDWARE ANALYSING USING KNOWN NUMBER OF STATES

IC Name	Type of FSM	FSM Found	NoS	NoFS	STT Found?	OF Found?	Evaluation Time
EC1	Mealy	Mealy	1	1	Yes	yes	282,7s = 4,7min
ELS1	Mealy	Mealy	8	8	Yes	yes	39124,0s = 10,9h
ENLS1	Mealy	Mealy	8	8	Yes	yes	39117,0s = 9,9h
S27	Mealy	Mealy	5	5	Yes	yes	466562,0s = 5,4d
B06	Moore	Moore	13	13	Yes	yes	2758,0s = 46,0min

In comparison to the hardware analysis using an unknown number of states the identification using known number of states is 1,3 times faster. Here, no approximation of the number of states is necessary. Furthermore, the analysis identified the expected state transition table as well as the output function. TABLE 6 shows the simulation results using known number of states.

TABLE 6: RESULTS OF SIMULATION USING KNOWN NUMBER OF STATES

IC Name	Type of FSM	FSM Found	NoS	NoFS	STT Found?	OF Found?	Evaluation Time
EC1	Mealy	Mealy	1	1	Yes	Yes	0,355s
ELS1	Mealy	Mealy	8	8	Yes	Yes	34,3s
ENLS1	Mealy	Mealy	8	8	Yes	Yes	34,0s
S27	Mealy	Mealy	5	5	Yes	Yes	8,74s
B06	Moore	Moore	13	13	Yes	Yes	2,94s

From TABLE 6 it can be seen that the nonlinear algorithm firstly investigates the type of automaton. Afterwards, the state transition table as well as the output function of the unknown IC were analysed. From all result tables presented TABLE 6 exhibits the lowest evaluation times of the IC models analysed. This can be explained that in this case the number of states was known prior to simulation.

VI. CONCLUSIONS

All analysis steps described were implemented into the MATLAB. The correct operation was verified through the implementation of several IEEE benchmark ICs as well as user defined IC models. The procedure described successfully solves the identification problem for the first time. Therefore, in conclusion this paper has presented a novel non-invasive reverse engineering procedure for structured analysis of deterministic sequential finite state machines in unknown CMOS ICs.

REFERENCES

[1] St. Jarzabek, T.P. Keam, "Design of a generic reverse engineering assistant tool," in *Proceedings of the 2nd Working Conference on Reverse Engineering*, Toronto, Canada, pp. 61-70, July 14-16, 1995.

[2] S. Blythe, B. Fraboni, S. Lall, H. Ahmed and U. de Riu, "Layout reconstruction of complex silicon chips," *IEEE Journal of Solid-State Circuits*, vol. 28, issue 2, no. 2, Feb. 1993.

[3] D. Lee, M. Yannakakis, "Principles and methods of testing finite state machines – a survey," *Proceedings of the IEEE*, vol. 84, no. 8, pp. 1090-1123, Aug. 1996.

[4] Y. Kuroe, "Learning and identifying finite state automata with recurrent high-order neural networks," *SICE Annual Conference*, pp. 2241-2246, Sapporo, Aug. 2004.

[5] M. C. Hansen, H. Yalcin and J. P. Hayes, "Unveiling the ISCAS-85 benchmarks: a case study in reverse engineering," *IEEE Design and Test of Computers*, vol. 16, issue 3, pp. 72-80, July-Sept. 1999.

[6] M. Brutscheck, M. Franke, A. Th. Schwarzbacher, St. Becker, "Determination of pin types and minimisation of test vectors in unknown CMOS integrated circuits," *13th Electronic Devices and Systems IMAPS CS International Conference*, Brno, Czech Republic, pp. 64-69, Sept. 2006.

[7] M. Brutscheck, St. Berger, M. Franke, A. Th. Schwarzbacher, St. Becker, "Structural division procedure for efficient IC analysis," *Irish Signals and Systems Conference*, Galway, Ireland, pp. 18-23, June 2008.

[8] Z. Kohavi, *Switching and Finite Automata Theory*, 2nd Edition, New York, NY: McGraw-Hill Book Company, 1978.

[9] The MathWorks Incorporation [Online]. Available: http://www.mathworks.com, [Accessed: September 21, 2009].