2008-01-01

Structural Division Procedure for Efficient IC Analysis

Michael Brutscheck
Technological University Dublin

St. Berger
University of Applied Sciences, Merseburg

M. Franke
University of Applied Sciences, Mersburg

See next page for additional authors

Follow this and additional works at: <https://arrow.tudublin.ie/engschececon>

Part of the [Electrical and Computer Engineering Commons](#)

Recommended Citation

Brutscheck, M., Berger, S., Franke, M., Schwarzbacher, A., Becker, S.: Structural Division Procedure for Efficient IC Analysis. IET Irish Signals and Systems Conference, (ISSC 2008), pp.18-23. Galway, Ireland, 18-19 June, 2008. doi:10.21427/821p-5873

This Conference Paper is brought to you for free and open access by the School of Electrical and Electronic Engineering at ARROW@TU Dublin. It has been accepted for inclusion in Conference papers by an authorized administrator of ARROW@TU Dublin. For more information, please contact arrow.admin@tudublin.ie, aisling.coyne@tudublin.ie, gerard.connolly@tudublin.ie.

Authors

Michael Brutscheck, St. Berger, M. Franke, Andreas Schwarzbacher, and St. Becker

Structural Division Procedure for Efficient IC Analysis

M. Brutscheck^{*,**}, St. Berger^{**}, M. Franke^{**},
A. Th. Schwarzbacher^{*} and St. Becker^{**}

^{*}*School of Electronic and
Communications Engineering,
Dublin Institute of Technology, Ireland
Kevin Street, Dublin 8, Ireland
email : andreas.schwarzbacher@dit.ie*

^{**}*Department of Computer Science and
Communications Systems,
University of Applied Sciences,
Merseburg, Germany
email: michael.brutscheck@hs-merseburg.de*

Abstract— The efficient and structured analysis of unknown CMOS integrated circuits (ICs) has become a topic of great relevance in recent years. Up until now, different invasive [1], [2] and non-invasive [3], [4] strategies have been developed for procedure of analysis. However, invasive procedures always lead to the destruction of system under investigation. The non-invasive approaches published so far have the disadvantage that ICs are analysed by using complex algorithms. Here, no subdivision exists to avoid extensive analysis times in the case that only simple structures are investigated. Moreover, traditional procedures cannot automatically distinguish between input and output pin types, which is usually required in the investigation of real unknown integrated circuits.

This paper presents an efficient non-invasive procedure to determine binary multi-input multi-output (MIMO) ICs by its input-output behaviour. It was implemented into analysis environment described in [5] and classifies unknown ICs by means of automata theory. A novel separation procedure is proposed in this paper to further minimise the IC analysis. All sections of the classification procedure are simulated and fully tested on ISCAS-85, ISCAS-89 and ISCAS-99 benchmark models of real ICs [6], [7] and the results are presented in this paper.

Keywords – Non-Invasive Reverse Engineering, Digital CMOS IC, Finite State Machines.

I INTRODUCTION

Today's unknown CMOS integrated circuits have very complex structures and are designed using a great variety of functions and behaviours. Since the functions of ICs are not always known it can be essential to determine their correct behaviour. This may be required when the label is lost or it is necessary to find out more about the internal structure of the integrated circuit. Moreover, it is conceivable to use structures of ICs discontinued in new IC designs or to add new functionality to an existing system. Here, finite state machines (FSMs) are the main component in today's integrated circuits. To make a structured analysis of these ICs possible, there is a need to create an abstract model of the real IC. Therefore, the model of automata will be used to abstract the unknown system as in [8]. Automata can be divided into deterministic and

non-deterministic state machines. Real digital integrated circuits work as deterministic state machines. Only in the case of a malfunction of an integrated circuit non-deterministic behaviour may occur. In this research properly operating systems will be considered. That means that only complete and consistent systems will be investigated. Furthermore, it must be distinguished between infinite and finite state machines. Usually, the IC designers develop and realise integrated circuits as finite state machines. Therefore, only finite state machines are considered in this paper. It can be said that for a classification of unknown systems deterministic finite state machines must be considered.

In general, the classification procedure determines not the complete internal function of IC investigated, but general behaviour of the FSM, which can be divided into blocks with combinatorial,

linear or nonlinear behaviour using automata theory. The overall research objective of this project is to find non-invasive reverse engineering procedures to extract the function of unknown CMOS integrated circuits.

II FINITE STATE MACHINES

Generally, automata can be divided into deterministic and non-deterministic automata [8]. In this paper, only deterministic finite state machines M will be considered, which are defined by:

$$M = (Q, q_0, \Sigma, \Delta, \delta, \varphi) \quad (1)$$

Here, Q is the set of state symbols: $Q = \{q_1, q_2, \dots, q_r\}$, r is the number of state symbols, $q_0 \in Q$ is the initial state, Σ is the set of input symbols: $\Sigma = \{a_1, a_2, \dots, a_m\}$, m is the number of input symbols, Δ is the set of output symbols: $\Delta = \{b_1, b_2, \dots, b_l\}$, l is the number of output symbols, $\delta: Q \times \Sigma \rightarrow Q$ is the state transition function and $\varphi: Q \times \Sigma \rightarrow \Delta$ is the output function. The input alphabet consists of $\{0, 1\}$ values, which is described as a Galois Field (GF(2)), because only binary automata are handled in the classification procedure.

It will be supposed, that the FSM M operates at unit time intervals. Letting $i(t) \in \Sigma$, $o(t) \in \Delta$ and $q(t) \in Q$ be the input symbol, output symbol and state symbol at time t , respectively, then the FSM M is described by the discrete dynamical system of the form:

$$M \begin{cases} q(t+1) = \delta(q(t), i(t)) \\ o(t) = \varphi(q(t), i(t)) \end{cases} \quad (2)$$

A further division into several types of automata is possible. Finite state machines can be subdivided into combinatorial, linear sequential or nonlinear sequential automata as illustrated in Figure 1.

Figure 1: Basic Classification of State Machines

The combinatorial finite automaton is a special case of FSM, which has only one state. Linear behaviour means, that the automaton only uses modulo-2 operations, time-shifts and constant operations, which are realised by XOR-gates, D-latches and constant values of "0" or "1",

respectively. Nonlinear behaviour includes all remaining functions. Further subdivision of combinatorial automata into linear or nonlinear combinatorial finite state machines has only a theoretical sense, because there is no difference in the final evaluation of combinatorial equations.

The objective of this paper is to discuss the problem of separation of unknown CMOS ICs represented by FSMs described by (1) and (2) to minimise further analysis procedure.

III CLASSIFICATION PROCEDURE

Assumption 1: Every single pin type, such as input, output, voltage supply etc. is already known. Therefore, the number of input symbols Σ and output symbols Δ are known. The determination of pin types was already fully accomplished by the authors and is described in detail in [9]. Due to clarity, this paper will only consider input and output pin types, even so the presented environment is able to cope with all pin types.

Assumption 2: The maximum number of state symbols r is known. Further on the FSM investigated has a synchronous clock line and reset capability to set the FSM into the initial state q_0 after each simulation run.

Assumption 3: The state transition function δ and output function φ of IC are unknown.

In principle the separation procedure works in three steps, the determination of independent blocks, the division into combinatorial or sequential FSMs and the division into linear or nonlinear sequential FSMs, which can be seen in Figure 2.

Figure 2: The Classification of FSMs

In Figure 2 a complete overview of the separation procedure is shown. First the unknown

integrated circuit is divided into independent blocks, where every block has its own behaviour, inputs and outputs. Afterwards, every single block is checked for combinatorial behaviour. If the block has a combinatorial behaviour, the separation procedure is complete and no further classification analysis of the block has to be done. In this case, for every combinatorial block all 2^i bit combinations have to be applied, where i is the number of inputs of combinatorial block. Then the output function can be easily determined and minimised [9].

Otherwise, the block investigated must have a sequential behaviour and then a further division into linear or nonlinear sequential behaviour must be carried out. This is done by checking the block for linear sequential behaviour, because if the block does not have a linear behaviour, its behaviour must be nonlinear.

First several abstract virtual IC models were built in MATLAB [10]. They contain all types of automata the classification procedure handles as represented in Figure 2 to demonstrate the mode of operation.

Figure 3: Example of a Virtual IC Model

The sample consists of three different independent blocks, eight input pins and seven output pins as can be seen above as can be seen in Figure 3. The first block comprehends of the inputs x_1 and x_2 and the outputs y_1 , y_2 and y_3 , is built of simple logic gates only and for this reason it is a combinatorial finite state machine. The second block contains the input pins x_3 , x_4 and x_5 and the output pins y_4 and y_5 , consists of linear logic gates and additional memories and therefore it is a recursive linear sequential system. The third block contains the inputs x_6 , x_7 and x_8 and the outputs y_6 and y_7 . There, instead of a linear XOR gate a nonlinear OR gate is used and thus it is a recursive nonlinear sequential automaton.

Again, Figure 3 is only an example to demonstrate the functioning and can be enlarged or modified. It should be noted that one real IC input may act to multiple blocks, where the blocks may be of the different types (combinatorial, linear or nonlinear sequential). The proposed classification

procedure also supports determination of such problems.

a) Determination of Independent Blocks

Since the unknown integrated circuits usually consist of a very complex structure they may consist of several independent blocks with different characteristics. To avoid a false interpretation of the whole IC the interdependencies between the internal blocks are analysed first. Moreover, the determination of independent blocks is very important for the separation of unknown system, because the complexity of following analysis procedures can be reduced significantly. Figure 4 illustrates the basic structure of an unknown integrated circuit with its different finite state machines. There, the unknown system consists of j independent blocks, i input pins and w output pins. Every independent block is defined by its input and output pins, e.g. block j consists of $(n+1)-i$ inputs and $(y+1)-w$ outputs.

Figure 4: Basic Structure of a Multidimensional Recursive FSM

Generally, the determination of independent blocks is arranged in three steps. First, different combinations of pseudo noise (pn)-sequences are applied at the input, then the output values have to be stored and at last, equations for the interdependencies must be solved. Due to the fact that only binary deterministic automata are considered in the classification procedure for each input a different generator to create pseudorandom binary numbers is used. In contrast to real random number generators, the sequence generated by pn-generators is deterministic as after a finite amount of numbers the sequence starts to repeat itself. The pn-sequence generator block in MATLAB generates a sequence of pseudorandom binary numbers using a linear-feedback shift register (LFSR). The LFSR is implemented using a shift register as described in [8]. For every input pin a completely different pn-sequence has to be generated, as it is necessary to create the amount of different input values as high as possible. Afterwards, i different pn-sequences have to be applied to the inputs in 2^i-1 different

combinations, where i is the number of input pins as illustrated in Table 1.

Number of Combination	Inputs x_i				
	x_1	x_2	x_3	x_4	x_5
1	pn_1	0	0	0	0
2	0	pn_2	0	0	0
3	pn_1	pn_2	0	0	0
...	...	...	...	...	...
30	0	pn_2	pn_3	pn_4	pn_5
31 (2^i-1)	pn_1	pn_2	pn_3	pn_4	pn_5

Table 1: Example of a Data Table Generated

In this research generator polynomials with length $l = 2^{10} - 1 = 1023$ were used. Furthermore, it is important that each sequence consists of uniformly distributed values "0" and "1". The so called binary maximum, m , sequence has the property that its autocorrelation function (ACF) leads to maximum at (0) as illustrated in Figure 5.

Figure 5: ACF of Used m-sequences

Therefore, all generator polynomials used were successfully checked towards this property. Table 2 shows the characteristics of pn-sequences used with its generator polynomials and initial states.

Number of pn-sequence	Generator Polynomial	Initial State
1	[1 0 0 1 0 0 0 0 0 0 1]	[0 0 0 0 0 0 0 0 0 1]
2	[1 1 0 1 1 0 0 0 0 0 1]	[0 0 0 0 0 0 0 1 0 0]
3	[1 1 1 0 0 1 0 0 0 0 1]	[0 0 0 0 0 0 0 1 1 1]
4	[1 0 1 1 0 1 0 0 0 0 1]	[0 0 0 0 0 0 1 0 1 0]
5	[1 0 1 0 0 1 1 0 0 0 1]	[0 0 0 0 0 0 1 1 0 1]
6	[1 1 1 1 0 1 1 0 0 0 1]	[0 0 0 0 0 1 0 0 0 0]
7	[1 0 0 0 0 0 0 1 0 0 1]	[0 0 0 0 0 1 0 0 1 1]
8	[1 1 0 1 0 0 0 1 0 0 1]	[0 0 0 0 0 1 0 1 1 0]
9	[1 0 1 0 0 0 1 1 0 0 1]	[0 0 0 0 0 1 1 0 0 1]
10	[1 1 1 0 1 0 1 1 0 0 1]	[0 0 0 0 0 1 1 1 0 1]

Table 2: Generator Polynomial for pn-sequence Generation

Next, for every pn-sequence combination the results of each output have to be recorded and stored in a data table, e.g. for Table 1 31 output sequences have to be recorded. Finally, the dependencies can be solved by forming equations for each output. If an output reacts to a particular combination of applied pn-sequences, the output depends on the appropriate inputs. For example, if an output responds to the 30th combination represented in Table 1, it depends on the inputs x_2 - x_5 . In case the result sequence of the same output with the 31st combination applied is equal to the former combination, the output depends on every input except the first one. Each output, which has the same input dependencies, belongs to the same internal block.

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
y_1	1	1	0	0	0	0	0	0
y_2	1	1	0	0	0	0	0	0
y_3	1	1	0	0	0	0	0	0
y_4	0	0	1	1	1	0	0	0
y_5	0	0	1	1	1	0	0	0
y_6	0	0	0	0	0	1	1	1
y_7	0	0	0	0	0	1	1	1

Table 3: Result Array of Independent Blocks

For the example mentioned in Figure 3 the example result array looks like illustrated in Table 3.

b) Division into Combinatorial or Sequential Finite State Machines

The same pn-sequences as described in the previous section are used for the division into combinatorial or sequential finite state machines. This time no different combinations of the pn-sequences, but many completely different pn-sequences are applied to all inputs of one particular independent block at the same time. The pn-sequences have to be different every time, because identical input patterns must be avoided. This is because various possible internal states have to be considered and otherwise the analysis may lead to false results. In the second step all values, the results of the output pins as well as the values of the input pins of the corresponding time steps, are stored in a data table. An example with 4 inputs and 2 outputs is shown in Table 4.

Time Index t	Inputs x_i				Outputs y_k	
	x_1	x_2	x_3	x_4	y_1	y_2
1	1	0	1	0	0	0
2	0	1	0	0	1	0
3	1	1	0	1	0	1
...						
255	1	1	0	1	0	1
256	0	1	1	0	1	1
...						
510	1	0	0	1	0	1
511	1	1	0	1	0	1
512	0	1	0	0	1	0
...						

Table 4: Example of a Data Table Recorded

As the last step, for each time step t the results are checked by means of the theory of combinatorial logic [8]. According to this theory the same input values must lead to the same output results at any time. In the example the lines 3, 255 and 511 have the same input combination and therefore have to lead to the same output results in case of a combinatorial system. This analysis can easily be done for every time step with the same input patterns. If the results show that the system is a combinatorial finite state machine the classification procedure is complete and the system can further be analysed by using a maximum number of 2^i test vectors as described by the authors in [9] instead of pn-sequences. On the other hand, if only one identical input combination does not lead to the same output results the check can be stopped immediately and the system can be referred as sequential finite state machine. Then, a further subdivision into linear or nonlinear sequential automata is necessary.

c) Division into Linear or Nonlinear Sequential Finite State Machines

In contrast to combinatorial finite state machines the subdivision into linear or nonlinear sequential systems has a convenient sense as the further analysis of sequential automata can be simplified in case of linear behaviour.

Basically, this separation is based on the linear superposition of linear automata as presented in [8]. Superposition and linearity are mutually dependent. In signal and system theory traditionally the one-dimensional step function is used to determine the transition function of a one-dimensional linear system as in [8]. Here, two problems have to be solved. First, the structure of ICs might be of adverse structures. That means for example, if the circuit has a lot of AND-combinations, the division cannot be solved using the step function. Therefore, the division is also carried out using pn-sequences to increase the internal switching activity of the IC investigated. The same sequences as for the determination of independent blocks are used. However, many real integrated circuits have more than one input and one output. In this case, pn-sequences have to be adapted to multidimensional applications using matrices.

First, the pn-sequence has to be applied to the first input of one particular sequential block, while the other input pins are assigned zeros. With every step the output values have to be recorded to create the corresponding matrix. For example, if the sequential block consists of three inputs, the first pn-sequence pn_1 has to be applied to the first input, while inputs two and three are assigned zeros and the output values are recorded. Then, pn-sequence pn_2 has to be applied to the second input, while the other two input pins are assigned zeros and the output values are recorded. At last the same is done for the third input using pn-sequence pn_3 . Afterwards, the sum function has to be built of the different recorded output values. Therefore, the binary values of each output result, where only one pn-sequence is applied to the input, are added without considering the carry over. Next, all pn-sequences are applied to the inputs at the same time and the output results are recorded.

Finally, the system is analysed for linear sequential behaviour. This is done by comparing the results of the sum function with the result function, where all pn-sequences are applied at the same time. If the sums are equal, the checked output has a linear behaviour, otherwise a nonlinear behaviour. This analysis has to be done for each output of one particular sequential block. Only if the results for the analysis of every output have turned out to be linear, the whole block is considered to be a linear sequential finite state machine, else wise the system is a nonlinear sequential automaton.

IX RESULTS

Next the results of the classification procedure for finite state machines are presented. The three main parts firstly implemented in the test environment [5].

The results of classification procedure exist in form of arrays and vectors with binary values. However, the results have to be converted into a human readable form. Therefore, a text file is created, where the results are written in a suitable form. At the end of the classification procedure all information will be interpreted and printed on computer screen. The result screen as described in (3) contains the simulated results for the IC model presented in Figure 3.

$$\begin{aligned} B_1 &= f(y_1, y_2, y_3, x_1, x_2, \text{combinatorial}) \\ B_2 &= f(y_4, y_5, x_3, x_4, x_5, \text{linear sequential}) \\ B_3 &= f(y_6, y_7, x_6, x_7, x_8, \text{nonlinear sequential}) \end{aligned} \quad (3)$$

The following information can be gathered from (3). First, there exist three independent blocks. Here, block B_1 is of combinatorial type and is interconnected between inputs x_1, x_2 , and outputs y_1, y_2, y_3 . Circuit block B_2 has linear sequential behaviour and is connected between inputs x_3, x_4 and x_5 , and outputs y_4 and y_5 . Internal circuit B_3 has nonlinear sequential behaviour and is connected between inputs x_6, x_7 and x_8 and outputs y_6 and y_7 . The result extracted from the separation procedure is equivalent to the real virtual sample of the integrated circuit. At the end of the classification procedure all information will be interpreted and printed on computer screen. The whole simulation, the preparation, the recording of the simulation results as well as the analysis of the results is executed by functions implemented in the analysis environment.

Figure 6: Analysis Environment

Figure 6 shows the hardware of the analysis environment used in this research. In the next step, HDL-code of benchmark ICs of the ISCAS-85, ISCAS-89 and ISCAS-99 series [6] and [7] were implemented into MATLAB and used to prove the correct mode of operation of the classification procedure. Due to current limitation of the whole separation procedure to maximum of ten inputs and ten outputs only benchmarks with less than ten ins

and outs are used. Table 5 shows the results of IC models implemented into the analysis environment. Here, C stands for combinatorial and NLS for nonlinear sequential, respectively.

Bench	In	Out	Gates	Expected Behaviour	Found Behaviour?
C17	5	2	6	C	yes
S27	4	1	8	NLS	yes
B01	2	2	39	NLS	yes
B02	1	1	24	NLS	yes
B03	4	4	144	NLS	yes
B06	2	6	49	NLS	yes
B09	1	1	146	NLS	yes
B11	7	6	622	NLS	yes

Table 5: Result Table of Benches Analysed

To demonstrate the operation of the classification procedure several benchmarks models were combined and implemented into MATLAB. Table 6 shows the results of the circuits investigated.

Bench-Mix	Expected Behaviour	Found Behaviour?
2x C17	4xC	yes
C17, S27	2xC, 1xNLS	yes
2x S27	2xNLS	yes
S27, 2xB03	3xNLS	yes
B01, 2xB03	3xNLS	yes
B01, B06	2xNLS	yes
B03, B06	2xNLS	yes
B01, B11	2xNLS	yes
B03, B11	2xNLS	yes

Table 6: Result Table of Mixed Benches Analysed

It has to be noted, that one IC-model C17 consists of two independent blocks. Therefore, the expected behaviour for two models C17 in Table 6 is four, which conforms with the found behaviour.

As can be seen in Table 5 and Table 6, all IC models used were successfully solved due to its independent blocks and its behaviour, respectively.

IX CONCLUSION

This paper has presented the first non-destructive classification procedure for finite state machines in unknown binary MIMO CMOS integrated circuits. The analysis was performed using statistical investigation of its input-output behaviour. First, it was shown that the analysis described uses the abstraction of theoretical FSMs, which are a part of most of today's ICs. Here, the theoretical assumptions were verified with various circuit combinations. For this purpose all analysis steps described were implemented into the software MATLAB [10] and verified using the analysis environment as in [5]. It was demonstrated that the simulation results obtained fully agree with the theoretical assumptions made.

Secondly, several independent structures may exist in one integrated circuit. It has been presented, how the identification of independent automata can successfully be carried out in relation to the number of independent blocks. Furthermore, it was shown that unknown ICs can be classified with the

procedure described. The correct operation was verified through the implementation of several benchmark ICs of the ISCAS-85, ISCAS-89 and ISCAS-99. Here, the procedure described successfully solves the separation problem for the first time.

This paper has demonstrated that the non-invasive classification procedure presented is furthermore capable to separate FSMs for a further analysis of unknown CMOS integrated circuits. Therefore, in conclusion this paper has presented a novel non-invasive reverse engineering procedure for structured analysis of unknown CMOS ICs.

REFERENCES

- [1] J. Suhua, N. Jingfeng and W. Jiaji, "Investigation on properties of focused ion beam enhanced etching," *IEEE Proceedings of 6th International Conference on Solid-State and Integrated-Circuit Technology*, vol. 2, pp. 1091-1094, Shanghai, China, October 22-25, 2001.
- [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, no. 2, February 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, August 1996.
- [4] Y. Kuroe, "Learning and identifying finite state automata with recurrent high-order neural networks," *SICE Annual Conference*, pp. 2241-2246, Sapporo, August 2004.
- [5] M. Brutscheck, A. Th. Schwarzbacher and St. Becker, "A test environment for analysing unknown integrated circuits," *7th New Generation Scientist Conference*, Wernigerode, Germany, pp. 223-224, January 2006.
- [6] 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*, pp. 72-80, 1999.
- [7] ISCAS Benchmark Circuits, www.fm.vslib.cz/~kes/asic/iscas, March 2008.
- [8] Z. Kohavi, *Switching and Finite Automata Theory*, McGraw-Hill Book Company, Second Edition, New York, 1978.
- [9] M. Brutscheck, M. Franke, A. Th. Schwarzbacher and St. Becker, "Determination of pin types and minimisation of test vectors in unknown CMOS integrated circuits," *Electronic Devices and Systems IMAP CS International Conference*, Brno, Czech Republic, pp. 64-69, September 2006.
- [10] Mathworks, www.mathworks.com, March 2008.