Journal of Theoretical and Applied Information Technology
20th April 2013. Vol. 50 No.2
© 2005 - 2013 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

EQUIVALENCE CHECKING OF COMBINATIONAL CIRCUIT USING CHAOTIC PATTERN SIMULATION AND BINARY DECISION DIAGRAMS

ZHONGLIANG PAN, LING CHEN
Department of Electronics, School of Physics and Telecommunications Engineering,
South China Normal University, Guangzhou 510006, China.
E-mail: panz@scnu.edu.cn

ABSTRACT

With the increase of complexity of circuits, guaranteeing the correctness of design becomes extremely important. A new equivalence checking method is presented in this paper for the verifications of combinational circuits; the method uses the chaotic pattern simulation to find a lot of equivalent nodes, which results in that the scale of the composite circuit is reduced. The equivalence checking of two combinational circuits is carried out by constructing a BDD which is corresponding to a circuit being made up of the composite circuit and interface circuit. If the BDD is a constant 0, then the two combinational circuits are functional equivalence, the two rest circuits are not equivalent. The experimental results for a lot of circuits show that the more accurate equivalent nodes can be obtained by using chaotic pattern simulation in this paper than the random pattern simulation, and the equivalence checking method presented in this paper is able to verify the combinational circuits in shorter time.

Keywords: *Combinational circuits, equivalence checking, formal verification, pattern simulation, binary decision diagrams.*

1. INTRODUCTION

In recent years, the advancements in VLSI technology have led to the increased complexity in the circuit hardware design. It is becoming more and more important to ensure the correctness of design and the removal of design errors in the design cycle [1]. The larger sizes of circuits have made the verifying functional correctness to increasing difficult. Therefore, it is in a great need of the equivalence checking technique that can verify the correctness of circuit design. Here, the verifying functional equivalence of combinational circuits is one of basic equivalence checking problems; it is known to be a co-NP complete problem.

In the aspect of circuit verification approaches, Guralnik et al [2] discussed the simulation-based verification methods for floating-point division; the method consists of a comprehensive test plan and a powerful test generator. Vasudevan et al [3] investigated the approach to verify the correctness of arithmetic circuit designs described at the register transfer level, the approach used the stepwise refinement of term rewriting system. Hao et al [4] discussed the state explosion in the verification of timed-circuits by using abstraction directed by the failure model. Chandan et al [5] investigated the equivalence checking approaches for scheduling verification in high-level synthesis; the cut points in the finite state machine with data path were used.

In the aspect of system-on-chip(SoC) verification, Nam et al [6] aimed at various requirements of SoC verification, discussed a universal verification method to build an efficient and structured verification environment, the standardized test-bench architecture was used in this method. Xiaoxi et al [7] discussed the simulation-based verification of SoC, and used transfer-resource graph (TRG) to generate the test cases of resource competitions, and gave an approach that the test cases were structured in event-driven test programs. Strang et al [8] applied the holistic technique to the SoC verification. The hierarchies of signals, color-coding, advanced packet bundling etc. were used in the SoC verification and debug procedure. Chakraborty et al [9] investigated the various timing issues related to the modular SoC verifications, and presented a hierarchical method to verify the system level timing of SoC.

In the applications of verification techniques, such as the verification of microprocessors, Schubert et al [10] discussed the verifications of the Power7 microprocessor and multiprocessor systems, the random-constrained unit verification method and the thread-scaling support method in core verification were used. Ching et al [11] investigated the verification of external interrupt behaviors of microprocessor, and presented a tool of processor exception verification to verify the individual, multiple, and nested interrupts. Wagner et al [12] gave a tool called stress-test for the microprocessor verification, where the stress-test was based on a markov model driven random instruction generator with activity monitors. Madl et al [13] proposed a cross-abstraction real-time analysis framework for the model-based functional verification of chip multiprocessors.

Besides, the Petri nets have been applied in the circuit verification. For example, Little et al [14] used the labeled hybrid Petri nets to the verification of analog/mixed-signal circuits. Poliakov et al [15] made use of a special type of Petri nets to represent and verify the asynchronous circuits. Weinberger et al [16] proposed the workflow Petri nets method that can model the verification processes in the circuit design flows.

In this paper, a new equivalence checking method for combinational circuits is presented, the method uses the chaotic pattern simulation to find a lot of equivalent nodes, and construct the BDD of composite circuit to perform the equivalence checking of two combinational circuits. This paper is organized as follows. Section 2 gives the brief description about binary decision diagrams. Section 3 presents the equivalence checking method by chaotic pattern simulation and binary decision diagrams. Section 4 gives the experimental results for a lot of benchmark circuits. Finally, the conclusions are given in Section 5.

2. BINARY DECISION DIAGRAM

The Boolean variables and Boolean functions are widely used in the circuit design. The binary decision diagram (BDD) is a graph representation of logic Boolean functions [17,18]. Suppose the Boolean variable x_i be from {0,1}, the vector $\mathbf{x}=(x_1, x_2, \cdots, x_n)$. Let the Boolean functions h be from $\{0,1\}^n \to \{0,1\}^m$, i.e., the function h is expressed over the variables $x_1, x_2, \cdots, x_n$. Every Boolean function h can be represented by a BDD, where a following Shannon decomposition is performed on each node in the BDD.

$$h = \overline{x}_i \cdot h_0 + x_i \cdot h_1$$

In the above equation, the h_0 represents the value of h at x_i =0, the h_1 represents the value of h at x_i =1. The binary decision diagram is a rooted directed acyclic graph, which has two types of terminal nodes that are referred to as the 0-terminal and the 1-terminal. Each non-terminal node is associated with a primary input variable so that it has two outgoing edges called the 0-edge and 1-edge. The 0-edge corresponds to assigning the variable a 0 value, and the 1-edge corresponds to assigning the variable a 1 value.

For example, the BDD of function $h=x_1x_2x_3+x_4$ is shown in the Fig.1. The 0-edge is shown by a dashed line, the 1-edge is shown by a solid line in the Fig.1.

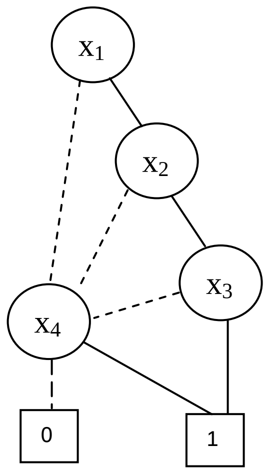

Fig.1 The BDD of function h.

A BDD is called ordered if it satisfied following two aspects: (a) Each variable is encountered at most once on each path from the root node to a terminal node. (b) The variables are encountered in the same order on all such paths. The BDD is called reduced if it has not isomorphic sub-graphs or instances of both edges from a single node pointing to the same node. The reduced and ordered binary decision diagram (ROBDD) is unique for a given variable order of a Boolean function, it can provide compact representations of logic Boolean functions [19]. In the following discussions, the ROBDD is considered, and for briefness these graphs are referred to as BDD.

3. EQUIVALENCE CHECK BY CHAOTIC PATTERN SIMULATION AND BDD

It is necessary to verify the correctness of the synthesis operations during the synthesis and optimization of the combinational circuits. The main verification task of combinational circuits is to carry out the equivalence checking, i.e., verifying the functional equivalence of two combinational circuits, one of which is the circuit before the synthesis steps and the other circuit is the post-synthesis circuit.

3.1 Composite Circuit

A digital circuit is combinational circuit if and only if the circuit does not contain cycles. A combinational circuit can be modeled as a directed acyclic graph. A composite circuit given in the Fig.2 is produced in order to carry out the equivalence checking of two combinational circuits.

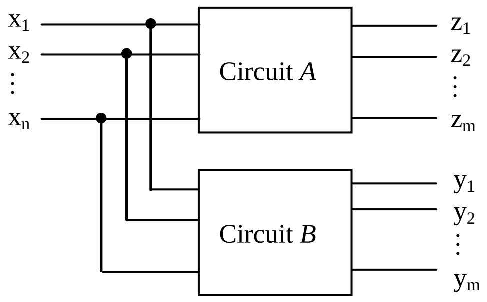

Fig.2 Composite circuit.

In the Fig.2, the circuit *A* is the circuit before the synthesis steps, the circuit *B* is the circuit after synthesis steps. The circuit *A* and circuit *B* are connected to same primary inputs $x_1, x_2, \cdots, x_n$. The equivalence checking is the problem to check whether corresponding primary outputs pairs z_i and y_i (i=1,2,⋯,m) in a composite circuit are equivalent.

Besides, an interface circuit is constructed, which consists of *m* XOR gates with two inputs and an OR gate with *m* inputs that is the outputs of the *m* XOR gates. For every XOR gate, one of its inputs is the z_i, and the other is the y_i. The interface circuit is shown in the Fig.3.

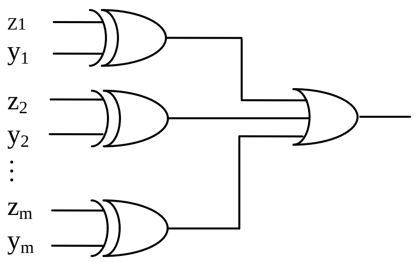

Fig.3 Interface circuit.

Therefore, verifying the functional equivalence of two combinational circuits can be implemented by checking whether the output of the interface circuit is 0 for all values of the primary inputs x_1, $x_2, \cdots, x_n$.

3.2 Equivalence Checking Algorithm

In the following, a new equivalence checking method for combinational circuits is presented; the method uses the chaotic pattern simulation to find a lot of equivalent nodes, which reduces the structure and the number of signal lines in the composite circuit. The equivalence checking method is implemented by constructing the BDDs of the composite circuit and interface circuit. The method consists of following five steps:

Algorithm 1

Step 1. Compute the structure level of each node (signal line) in the circuit *A* and circuit *B*.

Step 2. For these nodes whose structure level being less than a given positive integer L_0, search a lot of possible equivalent nodes in the circuit *A* and circuit *B* of composite circuit by using the chaotic pattern simulation. Let the set N={(n_{A1}, n_{B1}), (n_{A2}, n_{B2}), ⋯, (n_{As}, n_{Bs})} is the set of all the possible equivalent node pairs, where an element in the N is a node pair, for example (n_{A1}, n_{B1}), which shows that the node named as n_{A1} is possible functional equivalent to the node named as n_{B1}.

Step 3. Each node pair in the N is the related to the primary inputs x_1, x_2, ⋯, x_n. Therefore, the logic function of each node can be expressed by these primary inputs. Construct the BDD of each node pair in the N, i.e., construct the BDD of the n_{Ai} and n_{Bi} for i=1,2, ⋯, s. Go to step further, check the functional equivalence of each node pair n_{Ai} and n_{Bi}. Obtain a new set M of node pairs by discarding the not equivalent node pair in the N.

Step 4. For the composite circuit and the set M, all the equivalent nodes being obtained in the circuit *B* are reconnected to the nodes in the circuit *A*.

Step 5. Construct the BDD of the circuit that consists of the composite circuit and the interface circuit. If the BDD is a constant 0, then the two circuits *A* and *B* are functional equivalence, the two rest circuits are not equivalent.

The detail implementation of the Algorithm 1 is given in the following.

In the Step 1 of the Algorithm 1, the structure level of a node is defined as follows: The structure levels of all primary input lines are 0. For all non-primary input lines, for instance, for line P, the structure level of P is defined by $L(P)$, the $L(P)$=

max(L(S)) +1, the line S belongs to the fanin of line P. If there is a path from line S to line P, then line S is called in the transitive fanin of line P.

3.3 Chaotic Pattern Simulation

In the Step 2 of the Algorithm 1, the L_0 is a constant, for example, the L_0=145. The task of this step is to search the possible equivalent nodes whose structure levels are less than the L_0. Here, the following chaotic pattern simulation is used, which has three steps.

First of all, define the following chaotic system which is the map given by the equation:

$$d_{k+1} = \sin(2/d_k) \qquad k = 0,1,2, \cdots \cdots. \qquad (1)$$

Where the d_k is variable d at the k-th iteration, the value of d_k belongs to (0, 1). The chaotic map (1) can generate a large number of uncorrelated, random like and deterministic data sequences. A small difference in the initial value d_0 can lead to a vast change of the chaotic sequence.

The chaotic map (1) is used to generate the input vectors of the circuit A and circuit B, i.e., the vector $\mathbf{x}$=(x_1, x_2, ···, x_n). The component x_i of a vector $\mathbf{x}$ is 0 if the d_k being produced by chaotic map (1) belongs to (0, 0.5). The component x_i of a vector $\mathbf{x}$ is 1 if the d_k being produced belongs to [0.5, 1]. By using this approach, we can generate a set T, which is made up of K input vectors of the circuit A and circuit B, where the K is a given positive integer.

Secondly, the values of each signal line (whose structure level being less than a given positive integer L_0) in the circuit A and circuit B are computed when the K input vectors in the set T are applied to the primary input lines of the circuit A and circuit B.

Thirdly, when the K input vectors in the set T are applied to the primary input lines, for the circuit A, count the amounts of the value 0 and value 1 for each signal line whose structure level being less than a given positive integer L_0. Perform similar this counting for the circuit B.

If a signal line L_A in the circuit A has the same amounts of both the value 0 and value 1 as the signal line L_B in the circuit B, then the signal lines L_A and L_B are considered as a possible equivalent node pairs (L_A, L_B). Therefore, the set N of equivalent node pairs is obtained, Let the N={(n_{A1}, n_{B1}), (n_{A2}, n_{B2}), ···, (n_{As}, n_{Bs})}, which is made up of s node pairs.

3.4 Construction of BDD

In the Step 3 of the Algorithm 1, the task of this step is to check the functional equivalence of each node pair n_{Ai} and n_{Bi} in the set N, and obtain a new set M of node pairs by discarding the not equivalent node pair in the N.

The node pair n_{Ai} and n_{Bi} is related to the primary inputs x_1, x_2, ···, x_n. The logic functions of nodes n_{Ai} and n_{Bi} can be expressed by these primary inputs. Therefore, the BDDs corresponding to the nodes n_{Ai} and n_{Bi} are constructed respectively. Whether the logic functions of node n_{Ai} is equivalent to the node n_{Bi} or not, it can be checked by comparing the BDDs that corresponding to the nodes n_{Ai} and n_{Bi}. The logic functions of nodes n_{Ai} and n_{Bi} are equivalent if the two BDDs are isomorphic graphs; the other logic functions are not equivalent.

In the Step 4 of the Algorithm 1, the structure of the composite circuit is modified by the following mode: all the equivalent nodes being obtained in the circuit B are reconnected to the nodes in the circuit A. An example is given as follows.

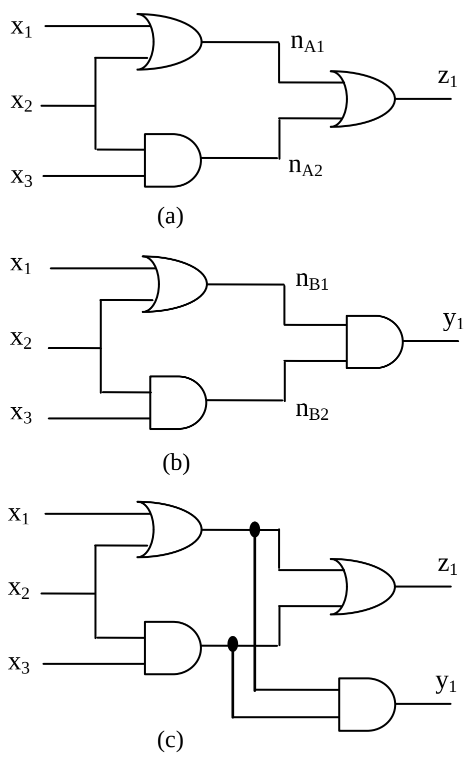

Fig.4 The composite circuit being modified.

In the Fig.4, there are two equivalent node pairs (n_{A1}, n_{B1}) and (n_{A2}, n_{B2}) in the circuits shown in the Fig.4(a) and Fig.4(b). The composite circuit being modified is shown in the Fig.4(c), where the n_{B1} is directly connected the n_{A1}, the n_{B2} is directly connected the n_{A2}, respectively. Thus, the number of signal lines in the composite circuit is reduced.

In the Step 5 of the Algorithm 1, a BDD is constructed for the circuit that consists of the composite circuit and the interface circuit. If the BDD is a constant 0, then the two circuits *A* and *B* are functional equivalent, the other two circuits are not equivalent.

4. EXPERIMENTAL RESULTS

The equivalence checking method proposed in this paper has been implemented in C++ language, and the method has been applied to carry out the verifications of combinational circuits in ISCAS'85 benchmark circuits. A lot of experiments have been carried out on a personal computer with 3.0GHz and 512MB memory under Windows operation system. The total numbers of gates in these ISCAS'85 benchmark circuits are shown in the Table 1.

Table 1 The ISCAS'85 benchmark circuits.

Circuit	Ninputs	Noutputs	Ngates	Nfaults
C499	41	32	202	758
C880	60	26	383	942
C1355	41	32	546	1574
C1908	33	25	880	1879
C2670	233	140	1193	2747
C3540	50	22	1669	3428
C7552	207	108	3512	7550

In the Table 1, the column "Circuit" gives the names of benchmark circuits. The columns "Ninputs" and "Noutputs" show the numbers of primary inputs and primary outputs in the circuits, respectively. The column "Ngates" shows the total number of gates in a circuit. The column "Nfaults" denotes the size of the simplistically reduced equivalent single stuck-at fault set for a circuit.

The BDDs in the Algorithm 1 are constructed by following approach. In general, a circuit is made up of many circuit blocks. The logic function of whole circuit can be expressed by a sequence of operations on the logic Boolean functions being realized by these circuit blocks. The BDD of whole circuit can be obtained by using these BDDs of all circuit blocks. The procedure of building BDD is shown as follows: start from the circuit primary inputs, each gate output is expressed in terms of its inputs, and then these BDDs corresponding to the gate outputs are constructed. Repeat this operation, until the BDDs of the circuit primary outputs are constructed.

The following operator *ite* is used for the building BDD. For given three logic functions f, g, and h, *ite*(f, g, h) = f·g + $\bar{f}$ ·h. The *ite* operator can realize all Boolean operations with two variables.

Let the F, G, and H are the BDDs of logic functions f, g and h, respectively. The Shannon decomposition of the F is expressed by the following equation:

$$F = w \cdot Fw + \overline{w} \cdot F\overline{w} \qquad (2)$$

The variable *w* belongs to $\{x_1, x_2, \cdots, x_n\}$. The F_w denotes the $F|_{w=1}$, the $F\overline{w}$ denotes the $F|_{w=0}$. The F_w and $F\overline{w}$ are F being evaluated at *w*=1 and *w*=0, respectively. The following equation can be obtained by the equation (2):

$$ite(F,G,H)=ite(v,\ ite(F_v,G_v,H_v),\ ite(F\overline{v}, G\overline{v}, H\overline{v}))$$

The variable *v* belongs to $\{x_1, x_2, \cdots, x_n\}$. Therefore, the BDD of ite(F,G,H) can be constructed by using the three BDDs: the F, G, and H.

For the parameter L_0 in the Step 2 of Algorithm 1, its value is set to less than $\lambda \cdot L_{\max}$, where the $L_{\max}$ is the maximal structure level of signal lines in a circuit, the λ is a constant, it is set to 2/3, i.e., λ=2/3.

In these experiments, the Algorithm 1 is used to carry out the equivalence checking of two types of combinational circuits: The ISCAS'85 circuits and their non-redundant versions, for example, C499 vs. C499nr, where the C499nr is a non-redundant version of C499. The time being needed for the Algorithm 1 is less than one minute for the equivalence checking of the circuits C499, C880 and C1355, and is greater than one minute but less than two minutes for the equivalence checking of the circuits C1908, C2670, C3540 and C7552.

The experimental results also demonstrate that the chaotic pattern simulation in this paper can get more accurate equivalent nodes than the random pattern simulation, the number of nodes in the BDD corresponding to whole circuits are reduced greatly.

5. CONCLUSIONS

The equivalence checking of the combinational circuits is one important aspects in circuit design, it is known to be a co-NP complete problem. In this paper, the chaotic pattern simulation and BDD are used to perform the equivalence checking of combinational circuits; a lot of equivalent nodes can be found to reduce the number of nodes in the composite circuit. Further work needs to be done such as acquire more equivalent nodes by using pattern simulation.

ACKNOWLEDGEMENTS

This work was supported by National Natural Science Foundation of China (No.61072028), the Project of Department of Education of Guangdong Province (No.2012KJCX 0040), Guangdong Province & Chinese Ministry of Education Cooperation Project of Industry, Education and Academy (No.2009B090300339).

REFERENCES:

[1] C.Karfa, D.Sarkar, C.Mandal. "Verification of datapath and controller generation phase in high-level synthesis of digital circuits". *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol.29, no.3, 2010, pp.479-492.

[2] E.Guralnik, M.Aharoni, A.J.Birnbaum, A.Koyfman. "Simulation-based verification of floating-point division". *IEEE Trans. on Computers*, vol.60, no.2, 2011, pp.176-188.

[3] S.Vasudevan, V.Viswanath, R.W.Sumners, J.A.Abraham. "Automatic verification of arithmetic circuits in RTL using stepwise refinement of term rewriting systems". *IEEE Trans. on Computers*, vol.56, no.10, 2007, pp.1401-1414.

[4] Z.Hao, C.J.Myers, D.Walter, S.Little, T.Yoneda. "Verification of timed circuits with failure-directed abstractions". *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol.25, no.3, 2006, pp.403-412.

[5] K.Chandan, D.Sarkar, C.Mandal, P.Kumar. "An equivalence-checking method for scheduling verification in high-level synthesis". *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol.27, no.3, 2008, pp.556-569.

[6] Y.Y.Nam, K.J.Beom, K.N.Do, M.Byeong. "Beyond UVM for practical SoC verification". *Proceedings of International SoC Design Conference*, Jeju (Korea), Nov.17-18, 2011, pp.158-162.

[7] X.Xiaoxi, L.C.Chew. "Using transfer-resource graph for software-based verification of system-on-chip". *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol.27, no.7, 2008, pp.1315-1328.

[8] A.Strang, D.Potts, S.Hemmady. "A holistic approach to SoC verification", *Proceedings of International Symposium on Quality Electronic Design*, San Jose (USA), March 18-19, 2008, pp.417-422.

[9] R.Chakraborty, D.Chowdhury. "A hierarchical approach towards system level static timing verification of SoCs". *Proceedings of IEEE International Conference on Computer Design*, Squaw Creek (USA), Oct.4-7, 2009, pp.201-206.

[10] K.D.Schubert, W.Roesner, J.Ludden, J.Jackson. "Functional verification of the IBM Power7 microprocessor and Power7 multiprocessor systems". *IBM Journal of Research and Development*, vol.55, no.3, 2011, 101-107.

[11] Y.F.Ching, H.W.Kai, Z.J.Kun, H.I.Jer. "Automatic verification of external interrupt behaviors for microprocessor design". *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol.27, no.9, 2008, pp.1670-1683.

[12] I.Wagner, V.Bertacco, T.Austin. "Micro-processor verification via feedback-adjusted markov models". *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol.26, no.6, 2007, pp.1126-1138.

[13] G.Madl, S.Pasricha, N.Dutt, S.Abdelwahed. "Cross-abstraction functional verification and performance analysis of chip multiprocessor designs". *IEEE Trans. on Industrial Informatics*, vol.5, no.3, 2009, pp.241-256.

[14] S.Little, D.Walter, C.Myers, R.Thacker, S.Batchu, T.Yoneda. "Verification of analog/ mixed-signal circuits using labeled hybrid petri nets". *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol.30, no.4, 2011, pp.617-630.

[15] I.Poliakov, A.Mokhov, A.Rafiev, D.Sokolov, A.Yakovlev. "Automated verification of asynchronous circuits using circuit petri nets". *Proceedings of IEEE International Symposium on Asynchronous Circuits and Systems*, Newcastle (UK), April 7-11, 2008, pp.161-170.

[16] K.Weinberger, S.Bulach, R.Bosch. "Application of workflow petri nets to modeling of formal verification processes in design flow of digital integrated circuits". *Proceedings of Design, Automation and Test in Europe*, Munich (Germany), March 10-14, 2008, pp.937-938.

[17] S.Miremadi, B.Lennartson, K.Akesson. "A BDD-based approach for modeling plant and

supervisor by extended finite automata". *IEEE Trans. on Control Systems Technology*, vol.20, no.6, 2012, pp.1421-1435.

[18] O.Keren, I.Levin, R.S.Stankovic. "Determining the number of paths in decision diagrams by using autocorrelation coefficients". *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol.30, no.1, 2011, pp.31-44.

[19] J.A.Carrasco, V.Sune. "An ROBDD-based combinatorial method for the evaluation of yield of defect-tolerant systems-on-chip". *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, vol.17, no.2, 2009, pp.207-220.