Test Generation for Asynchronous Sequential Digital Circuits

Roland Dobai*
Institute of Informatics
Slovak Academy of Sciences
Dúbravská cesta 9, 845 07 Bratislava, Slovakia
roland.dobai@savba.sk

Abstract

The dissertation thesis is aimed at test generation for asynchronous sequential digital circuits, contributes to their time- and cost-effective testing, and indirectly supports their wider application, which improves the performance, the power consumption and the electromagnetic emission of future digital circuits. The main scientific contribution is design of the new test generator (optimized for test length and area overhead) for wide spectrum of asynchronous sequential digital circuits. The contributions are identification of unacceptable signal transitions before test generation, reduced number of generated test patterns for combinational representation, effective state justification on the gate level, fast (optimized for test length) sequential fault propagation to outputs and effective fault simulation. Experimental results confirmed the generation of optimal tests with good fault coverage and without application of any method for increasing the testability. The developed methods can be used with wider spectrum of circuits than other recently developed test generators, and at the same time their effectiveness ensures fast test generation.

Categories and Subject Descriptors

B.7.3 [**Reliability and Testing**]: Test generation

Keywords

test pattern generation, asynchronous circuit, stuck-at-zero fault, stuck-at-one fault, state justification, sequential fault propagation, fault simulation

1. Introduction

Today's digital circuits are mainly synchronous because they are well supported in commercial computer-aided design tools. However requirements against digital circuits are growing rapidly which causes many difficulties, e. g. high power consumption and electromagnetic interference. Asynchronous sequential digital circuits (ASDCs) are able to overcome the limitations of synchronous circuits but the problem of testing emerges when it comes to their practical use. The methods for testing synchronous circuits is a well explored area, but the testing of ASDCs still faces many challenges [1, 31].

ASDCs perform self-synchronization and hence an external clock signal is not required. Another major difference is in use of other memory elements, first of all C-elements [19]. C-element holds the output value while different logic values are on inputs, or sets it to the equal input value.

These fundamental differences may result in oscillations (signal values on feedbacks never stabilize), races (signal values on feedbacks change simultaneously) or hazards. Hazards are unwanted temporary signal value changes; and require attention during design and test also. Basically, they are results of varying delays along the paths in circuit. The temporary change of the assumed stable value is called static hazard. The dynamic hazard is defined as an unwanted multiple signal transition replacing the single one. Hazards can arise also in synchronous circuits but they disappear before the clock period ends, therefore they do not cause any problem [14].

Test pattern generation (TPG) is the process of generating test patterns to test the circuit, usually performed by an automatic test pattern generator (ATPG). The test patterns are generated to detect a desired set of faults. ATPGs for synchronous circuits target various fault models, e. g. stuck-at faults (SAFs), stuck-open faults, bridging faults, delay faults. ATPGs for ASDCs were published mainly for SAFs [27, 10]. Delay faults of ASDCs [28, 12] are delay constraint failures and should not be confused with delay faults of synchronous circuits.

The TPG process is more simple if design-for-testability (DfT) methods [32] were previously applied to the sequential circuit. These methods are intended to ensure faster fault activation and test response evaluation by increasing controllabilities and observabilities with additional circuit elements; therefore a negative side effect is the significant area overhead. Time-frame expansion methods [3] are used when DfT methods are not acceptable (e. g. the ASDC could become nonfunctional due to timing constraint violation). The area overhead is eliminated in this

*Recommended by thesis supervisor:
Assoc. Prof. Elena Gramatová
Defended at Institute of Informatics, Slovak Academy of Sciences on January 19, 2011.

Dobai, R. Test Generation for Asynchronous Sequential Digital Circuits. Information Sciences and Technologies Bulletin of the ACM Slovakia, Vol. 3, No. 1 (2011) 73-83

case, but the test is longer because the fault is activated and propagated to an observable point by a sequence of test patterns.

Test patterns generated by an ATPG are evaluated by fault simulation with the goal to determine the fault coverage (ratio of detected and undetected faults). The fault simulation of combinational and synchronous sequential circuits is a well explored area. The basic methods are the serial, the parallel, the deductive and the concurrent methods. The serial method is based on evaluating the circuit's response individually for every fault. The parallel method simulates simultaneously more than one fault. Deductive and concurrent methods are propagating the lists of detectable faults in the circuit. Every fault is considered with only one simulation overpass, hence deductive and concurrent methods are the fastest among these fault simulation methods [21].

The aim of the dissertation thesis was to improve the TPG for ASDCs and to contribute to the time- and cost-effective testing of ASDCs. The developed methods support indirectly the wider application of ASDCs, which improves the performance, the power consumption and the electromagnetic emission of future digital circuits. The main scientific contribution is development of the new method for TPG (optimized for test length and area overhead) for wide spectrum of ASDCs. The contributions are identification of unacceptable signal transitions before TPG, reduced number of generated test patterns for the combinational representation (CR) of ASDC, effective state justification on the gate level, fast (optimized for test length) sequential fault propagation to outputs and effective fault simulation.

The rest of the paper is organized as follows. Section 2 contains closely related research results to TPG and fault simulation of ASDCs. Goals of the dissertation thesis are outlined in Section 3. Section 4 is dedicated to the new TPG method. The new fault simulation method is described in Section 5. Section 6 analyzes the achieved results and Section 7 concludes the paper.

2. Related work

Many multi-valued logics have been developed and published for hazard detection starting with 3-valued till even 27-valued ones [2]. These logics are intended to represent every possible kind of hazards besides the stable logic values and transitions. Algorithms were also developed to determine if a combinational circuit is hazard-free even without examining every gate, therefore providing efficient hazard detection [15, 17]. These multi-valued logics and hazard detection algorithms are able to determine if the existence of hazards is possible under the given circumstances, but are unable to determine all possible hazardous conditions. Usually they provide many information about hazards but were developed for combinational circuits only, hence they do not consider ASDCs.

TPG for combinational circuits is a well explored area. Usually 5-valued logic {`0`, `1`, `X`, `D`, $\overline{\texttt{D}}$} is used for considering SAF in the circuit (where `0` is logic false; `1` is logic true; `X` is undefined/do-not-care value; `D` is logic true in fault-free and logic false in faulty circuit; and $\overline{\texttt{D}}$ is logic false in fault-free and logic true in faulty circuit). Considerable advance in TPG performance was achieved in the last decades starting with D algorithm [23], PODEM [13], FAN [11] and followed by others [3]. These algorithms are based on search for a test pattern which ensures the detection of the desired fault (the output of the circuit will be different in the faulty and the fault-free state). The faulty signal $\texttt{D}/\overline{\texttt{D}}$ is assigned to the fault site and the fault effect is propagated to primary outputs (POs). The propagation is based on creation of sensitive paths along which the faulty signal can be propagated. The sensitive path is created by assigning logic values to the gates along the path and by propagating the D-frontier. The D-frontier is the set of logic gates with a faulty signal on an input and an undefined value on the output. Initial objects containing the required value assignment are created and propagated in the circuit by a process called backtrace. The final objects after this propagation are used to assign a value to some place in the circuit. The value assignment cannot be determined always uniquely and can cause conflict of logic values. This conflict is called inconsistency and is tried to be resolved by backtracks. The backtrack is the process of assigning the opposite value at some place in the circuit where previously a decision of value assignment was made. A process called implication is used after every value assignment for determination of every other value uniquely implied by this assignment. If every sensitive path from the fault site to POs goes through the same logic gate then it is preferable to assign immediately the values supporting the fault propagation at that logic gate. This process is called the unique sensitization and is intended to speed up TPG.

Usually the ASDCs are represented in a form acceptable as input to TPG algorithms for combinational circuits. This form is called CR of ASDC and is intended to represent the ASDC in the environment of combinational ATPGs. The CR is constructed in two steps: (1) Every memory element is replaced by a set of simple gates. This set represents the function of the memory element without altering the function of the ASDC (the set and the original memory element have the same delay and the structure of the set does not create new hazards). (2) The feedbacks are cut into pairs of pseudo-primary inputs (PPIs) and pseudo-primary outputs (PPOs) denoted as (PPI_0, PPO_0), (PPI_1, PPO_1), ..., (PPI_{n-1}, PPO_{n-1}), where n is the number of feedbacks. These PPIs and PPOs are recognized by the combinational ATPG just like the regular primary inputs (PIs) and POs, and are specially handled only by the embracing sequential ATPG.

Many TPG algorithms for ASDCs were developed and published [9, 30, 22, 16, 27, 10]. SPIN-TEST [27] is a complete ATPG for SAFs of ASDCs. Hazard detection is performed using 13-valued logic during fault simulation [29]. SPIN-TEST transforms the ASDC into CR and uses ATALANTA [18] (derivate of FAN) to generate every possible test pattern for every SAF of this CR. ATALANTA performs this step by returning to every possible place where value assignment decision was made previously (through exhaustive number of backtracks). This principle is in contradiction with the requirement of fast TPG (backtrack number reduction), so ATALANTA is slower in this operational mode by many order of magnitude. SPIN-TEST performs forward state justification and sequential fault propagation. It uses the A* search algorithm [24] to generate the sequence of test patterns for a selected SAF. The goal for the state justification is to reach one of the states specified by test patterns (obtained from ATALANTA) for the targeted SAF from the initial

state of ASDC. Every state is considered at the same time because it is not possible to know in advance which state is the easiest to justify. Every possible assignment to the PIs (represents a test pattern) is considered for every state, and for every state a cost is computed. This cost predicts the number of patterns required to reach the goal after the acceptation of the given test pattern, and is based mainly on the number of different bit positions between the next state and the goal states, and on the range of corresponding different PPOs from the PIs (this affects how easily these values can be changed). The test pattern with the lowest predicted cost is selected, and the examination continues with the next state. The sequential fault propagation is performed similarly; the goal is to find a state when the faulty signal is on a PO. The cost is computed based on the range of the PPOs with a faulty signal from the POs. SPIN-TEST uses the A* search algorithm which results in a shorter test sequence in comparison to depth-first search [24] (where always the furthest state from the initial state is explored first and a closest one is considered only if the search is unsuccessful in the previous direction), but can not guarantee the shortest test length.

IB-TPG [10] is the most general method ever proposed for the ASDCs and is based on PODEM [13]. In the contrast to SPIN-TEST, it does not transform ASDC into CR. The feedbacks are removed using a DfT method [4]. C-elements remain in the circuit and their hold/set functions are determined. The possible previous states in the justification process are identified based on these functions. The possibility of hazards is examined during this identification. The generated test sequence is hypothetically shorter or equal than that of SPIN-TEST (because backward state justification is used together with breadth-first search [24] — all of the states reachable with s test patterns are examined before the states reachable with $s + 1$ test patterns). The sequential fault propagation is missing in IB-TPG; the fault is considered as undetected if the fault is propagated to a PPO and not to a PO. IB-TPG can be used only for ASDCs with C-elements and together with DfT methods.

The algorithm proposed in [9] is based on D algorithm and improves the test quality by considering faults even inside C-elements. Buffers are inserted to feedbacks and a new time frame is assumed when D algorithm reaches these buffers. Going back one time frame is executed when an inconsistency occurs. The method in [30] deals with timing constraint violations. It is not a general method, therefore it is applicable just to a smaller class of ASDCs. The ATPG in [22] represents the ASDC as synchronous finite-state machine and uses methods of synchronous circuits to generate test for the ASDC. The method in [16] is based on D algorithm and is limited to a specific gate library and DfT environment.

SPIN-SIM is a serial fault simulator for the speed-independent (SI) ASDCs [29, 26]. SPIN-SIM adopts the 13-valued logic [2] to improve the hazard detection accuracy and maintains the relative order of causal signal transitions using time stamps. The time stamp includes only a signal group ID and a time. The group ID is used to indicate causal transitions; signal transitions with a causal relation are assigned to the same group ID. The relative order of the causal transitions is recorded in the time field, which is incremented when the transition propagates.

3. Goals of the dissertation thesis

Many TPG methods were developed for ASDCs however these methods have many disadvantages: (1) The most significant problem of current methods is insufficient support of different types of ASDCs. (2) Generated tests are optimized for length only with application of DfT methods. This results in increased area overhead which may negatively influence the timing constraints of ASDCs. (3) Hazards are detected ineffectively just in the final phase of TPG. (4) Current ATPGs generate unnecessarily every test pattern for the CR of ASDC and (5) perform fault simulation the least effective way.

The main scientific goal of the dissertation thesis was to contribute to time- and cost-effective testing of ASDCs with development of the new ATPG which (1) can be used for different types of ASDCs, (2) generates shorter tests, (3) does not require DfT area overhead.

The dissertation goals were: (1) to identify the hazards before TPG, (2) to decrease the number of generated test patterns for CR of ASDC, (3) to justify the state in a universal way, (4) to propagate the fault sequentially using the shortest test, (5) to speed up the fault simulation.

4. The new test pattern generation method

The new developed test pattern generator is designated as TACTLESS (aTpg for Asynchronous CircuiTs incLuding brEadth-firSt Search) [7]. TACTLESS generates the test sequence for a given ASDC according to Figure 1. The ASDC is transformed into CR, the fault list is prepared, and hazardous signal transitions are identified. A fault in the CR is selected and a test pattern is generated for this fault by the new developed method. Another fault is selected if the fault is untestable (undetectable). The sequential TPG begins after the successful TPG for CR. The state defined by the test pattern is justified (sequence of test patterns is generated which initialize the ASDC to this state from the undefined state). The sequential fault propagation takes place after the successful state justification (sequence of test patterns is generated which propagates the fault effect from PPO to a PO if it is necessary). The TPG for CR is repeated if the state justification or the sequential fault propagation is unsuccessful. The new deductive fault simulator is used after the test sequence has been generated; every detected fault is removed from the fault list and TPG continues with the next fault. The logic value of port PPO_0 will be denoted as $\langle PPO_0 \rangle$ throughout the paper.

Definition 1. *The ordered n-tuple* $S = (s_0, s_1, \ldots, s_{n-1})$ *is the state of ASDC with n feedbacks if* $\forall q \in \{0, 1, \ldots, n-1\} : s_q = \langle PPO_q \rangle$.

Definition 2. S_a *is the previous state of* S_b *and* S_b *is the next state of* S_a *if at least one pattern exists for the PIs of ASDC which changes the state from* S_a *to* S_b.

Definition 3. *State* $S_u = (su_0, su_1, \ldots, su_{n-1})$ *is the undefined state of ASDC with n feedbacks only if* $\forall q \in \{0, 1, \ldots, n-1\} : su_q = \mathtt{X}$.

Definition 4. *The process of finding a sequence of test patterns for PIs of ASDC which will initialize the ASDC*

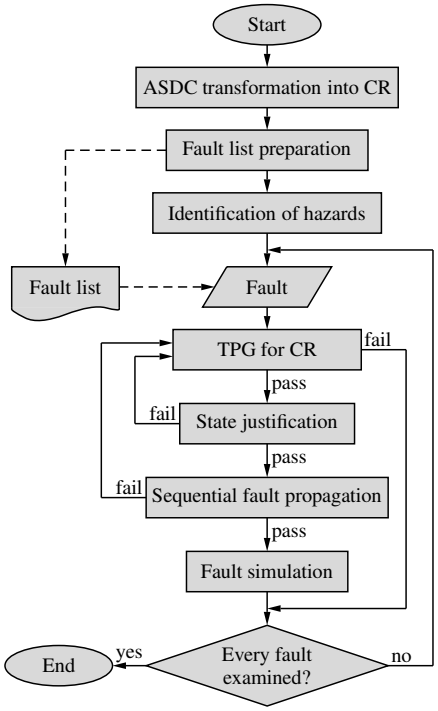

Figure 1: Flowchart of TACTLESS

to state S from the undefined state S_u is called state justification of S.

Definition 5. *The process of finding a sequence of test patterns for PIs of ASDC which will propagate the faulty signal from the PPO to a PO is called sequential fault propagation.*

Definition 6. *The state justification of S is considered forward if the space of possibilities is searched in the direction from the undefined state S_u to S, and is called backward if the search is performed in the opposite direction.*

Definition 7. *The sequential fault propagation is considered forward if the space of possibilities is searched in the direction from PPOs to POs, and is called backward if the search is performed in the opposite direction.*

4.1 Identification of hazards

Current ATPGs are able to detect hazards only in the final phase of TPG (by fault simulation). The detection results in the repetition of the entire TPG process. This is very ineffective because not only TPG for CR is executed again but state justification and sequential fault propagation too. The contribution of TACTLESS in this area is identification of hazardous signal transitions before the TPG process. The hazardous signal transitions are not accepted during the sequential TPG, therefore the number of insufficient test sequences is reduced. However, not every hazard can be identified in advance; e. g. hazards can be invoked by faults too. These remaining hazards are detected by fault simulation.

A new 6-valued logic A_6 was developed for hazard identification. Every value of this logic is intended to represent

Table 1: Values of the new 6-valued logic

Symbol	Meaning
`0`	logic false
`1`	logic true
`X`	do-not-care value
`T`	`0` $\rightarrow$ `1` or `1` $\rightarrow$ `0`
$\overline{\texttt{T}}$	inverse to `T`
`H`	all hazard types

two subsequent logic values on the given circuit line. Table 1 enumerates the values of the proposed A_6 logic. This new logic does not recognize the direction of transitions because it is not necessary to know it during hazard identification. Another unique characteristic of this 6-valued logic is that every type of hazard is denoted as `H`. Again, it is sufficient to know if a hazard arises somewhere inside the circuit and propagates to an output. The logic operations on the proposed 6-valued logic are defined in [7]. Hazard vectors (Definition 8) are used in TACTLESS to identify possible hazards.

Definition 8. *Let k be the number of PIs and n the number of PPIs for a given CR of ASDC, then the vector $(a_0, a_1, \ldots, a_{k-1}, b_0, b_1, \ldots, b_{n-1})$ where $a_r \in A_6 \wedge a_r = \left(\langle PI_r\rangle^{(t)} \rightarrow \langle PI_r\rangle^{(t+1)}\right)$ for all $r \in \{0, 1, \ldots, k-1\}$ and $b_q \in A_6 \wedge b_q = \left(\langle PPI_q\rangle^{(t)} \rightarrow \langle PPI_q\rangle^{(t+1)}\right)$ for all $q \in \{0, 1, \ldots, n-1\}$, is the hazard vector representing transition of logic values on PIs and PPIs in two subsequent steps t and $t+1$ if it is causing a hazard on at least one PO and/or PPO.*

A new algorithm was developed for hazard identification with the A_6 logic. The flowchart of the proposed algorithm was published in [7]. Hazard vectors are identified in CR for every PO and PPO. Hazard value `H` is assigned to them and backward propagation (in direction from outputs to inputs) is executed. Values are assigned to the gate inputs based on logic operations. Usually more than one possibility exists for justification of the given value. The first possibility is assigned to the gate inputs and the others are inserted into the stack for later evaluation. The assignments continue in the direction to CR inputs. Another assignment possibility is examined from the stack if a contradictory (inconsistent) assignment is detected on a fanout. The hazard vector (values on CR inputs) is saved after the assignments are finished and the inputs of CR are reached. The search for hazard vectors continues with another assignment possibility from the stack. All of the hazard vectors are found for the current examined CR output with assigning every possibility from the stack. This process is repeated for every CR output. Hazards are considered on the PPOs too, therefore hazardous signal transitions are not allowed neither on the feedbacks of ASDC. Similarly, the hazard vectors contain signal transitions on PPIs also, which will eliminate hazards caused by state changes.

The space search would become unacceptably extensive if every meaningless possibility is considered for gate output justification. Heuristic 1 and 2 were developed to reduce effectively this space under the assumption of single input changes (SICs). If during TPG only SICs are considered then these heuristics does not interfere with identification

of every hazard. Firstly, if only one transition can occur on the CR inputs then on the inputs of a given gate more than one transition can occur only if those gate inputs are reachable from a fanout; and these transitions can be of opposite direction only if this fanout reconvergence is of inverse polarity. This implication is used to limit the consideration of hazardous gate outputs by Heuristic 1. Secondly, hazards can be propagated just from hazard origins (Definition 9). This is used to reduce the number of hazardous gate inputs by Heuristic 2.

HEURISTIC 1. *Assign transitions of opposite directions* $\mathtt{T}$, $\overline{\mathtt{T}}$ *at the gate with inputs* $I_0, I_1, \ldots, I_{j-1}$ *only to* I_o *and* $I_p : o, p \in \{0, 1, \ldots, j-1\} \wedge o \neq p$ *and if* I_o *and* I_p *are reachable with different polarity from a fanout (reconvergence with inverse polarity).*

DEFINITION 9. *The j-input gate with inputs* $I_0, I_1, \ldots, I_{j-1}$ *is called hazard origin if* $\exists$ *at least one* I_o *and* $I_p : o, p \in \{0, 1, \ldots, j-1\} \wedge o \neq p$, *such* I_o *and* I_p *are reachable with different polarity from a fanout.*

HEURISTIC 2. *Assign hazard value* $\mathtt{H}$ *at the gate with inputs* $I_0, I_1, \ldots, I_{j-1}$ *only to* $I_o : o \in \{0, 1, \ldots, j-1\}$ *and if* I_o *is reachable from at least one hazard origin.*

4.2 Test pattern generation for CR of ASDC

New TPG method was developed for CR of ASDCs [8]. This method is based on FAN which was developed for combinational circuits. TPG methods for combinational circuits cannot be directly used for ASDCs. The new method differently handles PIs, PPIs, POs and PPOs; considers complex gates and uncontrollable/unobservable lines of ASDCs. The proposed method extends the existing strategies of FAN and contains new algorithms for successful TPG for CR of ASDCs. The contributions in this area are: (1) development of the new method based on FAN, (2) generation of test patterns one-by-one according to the requirements from state justification and sequential fault propagation (another test pattern is generated only if the previous one was insufficient), (3) generation of better test patterns for CR of ASDC (the test pattern is better if by a shorter test is possible to justify the state and propagate the fault effect to a PO).

A new definition supporting complex gates is proposed for the lines in ASDC. This definition eliminates inconsistencies at complex gates in the fanout-free part of the circuit. The line of the circuit can be free or bound. A bound line is a line which is reachable from a fanout point while a free line is not reachable from any. A free line is a head line if it is adjacent to some bound line.

DEFINITION 10. *Let the Boolean function F of variables* $x_1, x_2, \ldots, x_k$ *be in disjunctive normal form (DNF) with s conjunctions representing the k-input complex gate G, then mark the input* x_i *of G for every* $i \in \{1, \ldots, k\}$ *as a head line and the output F and all of the subsequent lines in the direction of signal propagation as bound line if* x_i *is free line and* $x_i \in X_g \wedge x_i \in X_h$, $g \neq h$, *where* $g \in \{1, \ldots, s\}, h \in \{1, \ldots, s\}$, X_g *is the set of variables of conjunction* m_g, X_h *is the set of variables of conjunction* m_h.

4.2.1 New strategies for test pattern generation

Existing strategies of FAN were extended with the following new ones to ensure the effective and successful TPG for ASDCs.

The new strategy for the early identification of undetectable faults is not necessary for the correct execution of the TPG but could make it more effective. The necessary condition for the fault detection is to assign the fault-free value to the fault site. This strategy helps to identify if this condition is not met.

STRATEGY 1. *Consider the stuck-at-one (stuck-at-zero) fault as undetected if it is on a* $\mathtt{0}$*-uncontrollable (*$\mathtt{1}$*-uncontrollable) line.*

New strategies eliminate some problems during unique sensitization of ASDCs by suggesting execution of backtrack when inconsistency occurs with a previously implied logic value or at a complex gate with an inner fanout. It is advised to execute multiple backtrace if none of logic gates can be uniquely sensitized. This will allow to assign both values if required (by backtrack).

STRATEGY 2. *If the required fault propagation is not possible during unique sensitization then backtrack should be executed.*

STRATEGY 3. *If unique sensitization is not possible for any logic gate then multiple backtrace should be executed.*

New strategies for determination of final objectives advise to execute backtrack in two more cases: (1) When the set of initial objects is empty before the multiple backtrace (this could happen when the set of unjustified lines is empty and the D-frontier cannot be propagated because of uncontrollable lines). (2) When the set of head lines is empty after the multiple backtrace (this could happen when none of the unjustified lines can be justified).

STRATEGY 4. *If the set of initial objects is empty before the multiple backtrace then execute backtrack.*

STRATEGY 5. *If the set of head lines is empty after the multiple backtrace then execute backtrack.*

The new strategy for multiple backtrace allows to stop the multiple backtrace at free lines and feedbacks. Stopping at free lines is important because the multiple backtrace can miss the head lines in the presence of complex gates with inner feedbacks. Stopping at feedbacks is necessary because the test pattern should be generated for CR of ASDC.

STRATEGY 6. *Add the current object to the set of head lines during the multiple backtrace if the line is head, free or is connected to a PPI.*

4.2.2 New algorithms for test pattern generation

Algorithm FAN executes many propagations of different types. These propagations are ensured by many specific algorithms, e. g. multiple backtrace. These algorithms were replaced in the proposed new method by new ones to support ASDCs.

Testability measurements (controllabilities and observabilities) were previously used in ATPGs for combinational and synchronous sequential circuits. These measurements for complex gates of ASDCs are computed in the proposed ATPG as follows. Firstly, controllabilities are determined for every logic conjunction in DNF of the given complex gate. This computation is based on application of the existing rule for the logic gate *AND*. Consequently the controllability is computed for the output of the complex gate by application of the rule for the logic gate *OR* to previous results. Secondly, observabilities are transferred from the output of the complex gate to the conjunctions in DNF of this gate (rules of the logic gate *OR* are used). Consequently the observabilities are determined for every gate input with application of rules of the logic gate *AND* to previous results.

The backward implication is the process of assigning uniquely implied logic values in the direction to POs and PPOs. The justification process of free lines is very similar with only one difference. Assignments during the justification of free lines are not necessarily unique. When more than one possibility exists then the assignment is executed randomly. The logic value at the output of the complex gate is justified/implicated as follows:

- `X` — The undefined value is not justified or implicated.
- `D` or `1` — The logic (fault-free) value `1` at the gate output is justified by setting one gate input to value `1` while this assignment together with the values on other inputs for the same conjunction in DNF implicate the value `1` at this conjunction. When this assignment is not possible then an inconsistency occurred. When there is more than one possibility then the assignment is not unique. The assignment is executed by the multiple backtrace in this case during the backward implication, or a random one is selected during the justification of free lines.
- $\overline{\texttt{D}}$ or `0` — The logic (fault-free) value `0` at the gate output should be implicated by setting one gate input while this assignment should be supported by other inputs for the same conjunction. When there is a logic `1` implied at least on one conjunction then an inconsistency occurred. When this assignment is not unique than multiple backtrace follows during the backward implication, or a random one is selected during the justification of free lines.

Initial objects supporting the fault propagation through complex gates are created as follows. Conjunctions in DNF of the given complex gate are analyzed. If the analyzed conjunction does have an input with a faulty signal then for every other inputs of this conjunction objects requesting logic value `1` are constructed. If the conjunction does not have a faulty input then an object requesting logic value `0` is created on that input where `0` is the easiest to control.

The algorithm for the multiple backtrace was extended by the new process for the propagation of objects through complex gates. The input to this process is the object for the gate output. Outputs of the proposed process are the objects for gate inputs. The object is propagated to conjunctions in DNF of the given complex gate. The number of requested values `1` is preserved only for that conjunction where value `1` is the easiest to control. The created new objects for conjunctions are propagated further to gate inputs. During this propagation the number of requested values `0` is preserved only for the easiest `0`-controllable input of the given conjunction.

The new algorithm developed for unique sensitization supports also complex gates. The sensitization is not always possible uniquely. The sensitization should not be executed for the complex gate where sensitive paths go through more than one conjunction in DNF of the given gate. This algorithm handles differently conjunctions with and without a sensitive path. Logic value `1` is assigned for every fault-free input of conjunction with a sensitive path. Supporting the propagation through the complex gate by the conjunction without a sensitive path is ensured by changing one undefined value `X` to value `0`. When there is more than one input with value `X` in the given conjunction then the sensitization is not possible uniquely.

4.3 State justification

Backward state justification is executed by TACTLESS after the test pattern for CR has been generated. The state required by this test pattern is justified performing breadth-first search [24]. Backward state justification together with breadth-first search guarantee shortest test lengths. The contribution of TACTLESS in this area is state justification on the gate level (TACTLESS can be used with any ASDC and is not limited to C-elements; simple gates are used hence it is just the matter of ASDC transformation into CR) and stepwise composition of state graph (it is possible to find interconnection with the existing part of the graph, which can make the later state justifications more effective).

Definition 11. *Let* $V = \{(s_0, s_1, \ldots, s_{n-1}) \mid (s_q \in A_N \wedge s_q = \langle PPO_q \rangle)$ *for all* $q \in \{0, 1, \ldots, n-1\}\}$ *denote the state set based on N-valued logic* A_N *and* $\Gamma = \{(a_0, a_1, \ldots, a_{k-1}) \mid (a_r \in \{\texttt{0}, \texttt{1}, \texttt{X}\} \wedge a_r = \langle PI_r \rangle)$ *for all* $r \in \{0, 1, \ldots, k-1\}\}$ *denote the test pattern set of ASDC with n PPOs and k PIs, then the directed graph* $G = (V, E, \Psi)$ *with* V *as the finite set of vertexes,* $E \subseteq \{(u, v) \mid u, v \in V \wedge u \neq v\}$ *as the finite set of edges and* $\Psi : E \to \Gamma$ *as the edge labeling function is the state graph of ASDC with state logic* A_N *if* $\forall (u, v) \in E$ *: test pattern* $\Psi((u, v))$ *changes the ASDC from state u to state v.*

Definition 12. *The state graph G of ASDC with state logic* $A_3 = \{\texttt{0}, \texttt{1}, \texttt{X}\}$ *is the justification state graph.*

Definition 13. *Let* $G = (V, E, \Psi)$ *be the justification state graph of the ASDC and* u_t *is a state, then for every* $(u_0, u_1), (u_1, u_2), \ldots, (u_{t-2}, u_{t-1}), (u_{t-1}, u_t) \in E$ *the sequence* $\Psi((u_0, u_1)), \Psi((u_1, u_2)), \ldots, \Psi((u_{t-2}, u_{t-1})), \Psi((u_{t-1}, u_t))$ *is the sequence of test patterns for state justification of state* u_t *if* $\forall o, p \in \{0, 1, \ldots, t\}$ $(u_o, u_p \in V \wedge o \neq p) : u_o \neq u_p$ *and state* u_0 *is the undefined state.*

The flowchart of the proposed state justification was published in [7]. Path in the justification state graph is attempted to be found from the undefined state to the required state and if this search is successful then the sequence of test patterns for state justification is reported; and the state justification ends. If the path does not exist yet then the previous states (together with the corresponding test patterns) are determined and put into list L (the algorithm for identification of previous states is described later). Every item from L is examined and is added as an edge to the state graph if the transition is not hazardous (two subsequent states and test patterns are combined into one vector of 6-valued logic; this vector is evaluated based on the hazard vector list; the list was previously generated as it was described in the previous subsection). The state part of this item is placed in the list D for later breadth-first evaluation. Every previous state will become evaluated while list L is becoming empty. A new state from list D becomes the current (the state selection is executed according to breadth-first principle) and its previous states are examined. This process is repeated until every state from D is not considered and the undefined state is not found. The path in the state graph is attempted to be identified again. The justification sequence is reported if this search is successful. Otherwise, generation of another test pattern for CR is suggested to the embracing sequential TPG. It is possible to find a connection to the existing part of the graph where a path was previously identified. The TPG time is reduced in this case because it is unnecessary to continue the search.

An algorithm very similar to that one proposed for hazard identification is used for identification of previous states. The differences are the following: (1) Standard 3-valued logic {0, 1, X} is used instead of A_6 logic. (2) The values for PPOs (the state) are assigned all at once instead of individual processing of the outputs. (3) The current SAF is considered inside the CR. (4) The vector defined by PPIs is considered as previous state, and the vector defined by PIs as the pattern required to change the previous state to the current state.

4.4 Sequential fault propagation

TACTLESS uses forward sequential fault propagation if the fault is propagated to PPOs instead of a PO by the test pattern for CR. The test sequence assembled before is supplemented by the sequential fault propagation sequence which ensures the propagation of faulty signal from the PPOs to a PO. The forward sequential fault propagation performed together with breadth-first search ensures shortest test sequences [24]. The recently published ATPGs for ASDCs either do not deal with sequential fault propagation or can not guarantee shortest test sequences.

The algorithm for sequential fault propagation is very similar to that of state justification. The differences are the following: (1) Different state graph is used (Definition 14) which considers faulty signals in the state because the search is conducted to propagate further these faulty signals. Therefore, the search is interrupted in the direction of states without any faulty signal. (2) The sequential fault propagation sequence is interpreted according to Definition 15. (3) Not previous states but next ones are searched for.

DEFINITION 14. *The state graph G of ASDC with state logic $A_5 = \{0, 1, X, D, \overline{D}\}$ is the state graph for sequential fault propagation.*

DEFINITION 15. *Let $G = (V, E, \Psi)$ be the state graph for sequential fault propagation of the ASDC, then for every (u_0, u_1), (u_1, u_2), $\ldots$, (u_{t-2}, u_{t-1}), $(u_{t-1}, u_t) \in E$ the sequence $\Psi\big((u_0, u_1)\big)$, $\Psi\big((u_1, u_2)\big)$, $\ldots$, $\Psi\big((u_{t-2}, u_{t-1})\big)$, $\Psi\big((u_{t-1}, u_t)\big)$ is the sequential fault propagation sequence for state u_0 if $\forall o, p \in \{0, 1, \ldots, t\}(u_o, u_p \in V \wedge o \neq p)$: $u_o \neq u_p$ and a faulty signal (D or $\overline{D}$) is in state u_t on at least one PO.*

The next states are determined by exhaustive simulation of every test pattern for the given state. This would results in a huge space search without another feature of the inner ATPG for CR, i. e. the faulty signal propagation is performed in the direction closest to a PO which means POs have higher priorities than PPOs, and if the propagation to a PO is impossible then the signal is propagated to that PPO from which the further propagation to a PO will be the easiest. This results in a relatively short sequential fault propagation, and represents another contribution in comparison to other ATPGs for ASDCs because their external ATPGs for CR does not consider ASDCs.

5. The new fault simulation method

The new fault simulator was developed for ASDCs [6]. This fault simulator is used in TACTLESS for determination of detectable faults. The proposed fault simulator is able to detect hazards and oscillations too.

The new method was developed for faster fault simulation of ASDCs because the existing methods are based on the slowest method (on the serial method). The proposed method is based on the deductive method which was previously used for combinational circuits only. The fault simulator propagates the list of detectable faults through feedbacks (when the fault lists differ on the input and output side of the feedback and when the maximum number of sequential propagations is not reached).

Existing rules for the propagation of fault lists were extended by the developed new algorithm supporting complex gates of ASDCs. This algorithm is universal and can be used for any complex gate represented with Boolean function in DNF. The list of detectable faults is determined based on fault lists on inputs of the given gate.

5.1 Fault list propagation through complex gates

The deductive fault simulator is based on the propagation of detectable faults in the circuit. The list of detectable faults is necessary to propagate through logic gates. This propagation is executed according to existing rules [21].

Existing rules for logic gates are insufficient for ASDCs which can contain complex gates too. The new method developed for complex gates is able to determine detectable faults at the output of the complex gate if the fault lists are known for the inputs. The first conjunction in DNF of the given complex gate is analyzed. Logic values and fault lists of this conjunction are processed according to rules for the logic gate *AND*. The result (faults) is placed to the set A or B based on the evaluated logic value

Table 2: Fault coverage comparison for random test

Circuit name	SAFs	SAF coverage			
		[25]	[26]	serial method	deductive method
alloc_outbound	58	92%	100.0%	100.00%	100.00%
chu133	60	97%	96.9%	98.33%	98.33%
chu150	40	82%	97.1%	95.00%	95.00%
converta	56	46%	91.9%	96.43%	96.43%
dff	34	79%	85.7%	100.00%	100.00%
ebergen	46		95.7%	100.00%	100.00%
half	34		100.0%	94.12%	94.12%
hazard	40	86%	97.0%	100.00%	100.00%
master_read	132	46%	97.7%	95.45%	95.45%
mp_forward_pkt	66	95%	100.0%	100.00%	100.00%
nak_pa	76	91%	100.0%	100.00%	100.00%
nowick	50	98%	100.0%	100.00%	100.00%
ram_read_sbuf	84	89%	100.0%	100.00%	100.00%
rcv_setup	36	93%	100.0%	100.00%	100.00%
rpdft	26	92%	100.0%	100.00%	100.00%
sbuf_ram_write	82	78%	100.0%	100.00%	100.00%
sbuf_send_ctl	66	49%	94.9%	98.48%	98.48%

of the current conjunction. The set A is used when this value is `1`. In the other case the faults are placed into the set B. This process is repeated for every conjunction but the faults are placed with set operations into these sets. Faults of the conjunction with value `1` are put into the set A using operation intersection, i. e. $A = A^p \cap Z$ where A^p is the previous content of the set A and Z is the fault list of the given conjunction. Fault of the conjunction with value `0` are put into the set B using operation union, i. e. $B = B^p \cup Z$ where B^p is the previous content of the set B and Z is the fault list of the given conjunction. Sets A and B after the analysis of every conjunction will contain faults which will be used to determine the fault list for the output of the given complex gate. The rules for the logic gate *OR* are applied to these sets, i. e. the final list will be the content of the set B if there is not any conjunction with logic value `1`, or otherwise it will be $A - B$.

5.2 Fault list propagation in ASDCs

The new fault list propagation algorithm developed for ASDCs is based on the algorithm for synchronous sequential circuits. The main difference is that more than one simulation overpasses are executed. These overpasses are repeated until logic values and fault lists on corresponding PPIs and PPOs are not the same, or until the maximum number of overpasses is not reached (for oscillation detection). This number is specified by the user. The next difference is the use of the 13-valued logic for hazard detection.

6. Results

TACTLESS was implemented in C++. The evaluation was executed with a set of SI control circuits [20] synthesized by Petrify [5] and with a set of quasi-delay-insensitive (QDI) datapath circuits [33]. The results were compared to the published results of SPIN-TEST [27] and IB-TPG [10], the two newest available ATPGs for ASDCs, and to the two most recent fault simulation methods [26, 25] for ASDCs.

6.1 Fault simulation of random test

The developed and implemented fault simulator was tested also with a random test pattern generator. Table 2 shows achieved fault coverages with 200 random test patterns (published results for the other fault simulators were achieved with 10000 test patterns). This number was determined as the lower limit without a fault coverage loss. Fault coverages are approximately the same than for other methods.

Table 3 compares the serial and the deductive method based on the execution time. The new deductive method is faster by 60%–80% than the previously used serial method. The serial method was faster in some cases (*chu133*, *converta*, *half*) because of the used fault dropping. When a fault coverage near to 100% is reached then the serial fault simulator will be working with a smaller fault list than the deductive method (which works always with the full fault list). In this case the serial fault simulator will be faster. However, this imperfection is eliminated by use of deterministic tests.

Table 4 contains the memory requirement comparison for the random test. The results are not accurate for some circuits (*chu150*, *nak_pa*) because of low execution times. In these cases lower memory requirements are reported for the deductive simulator. Another reason can be the use of relatively small circuits. The memory requirement of the deductive simulator (in the worst case) is just by 14% higher than for the serial one.

6.2 Deterministic test pattern generation

Table 5 compares the SAF coverages of TACTLESS with SPIN-TEST and IB-TPG for SI and QDI circuits. SPIN-TEST outperformed the others, but TACTLESS and IB-TPG follow different goals (shortest test length, wider spectrum of ASDCs). The SAF coverages of SPIN-TEST are in average just slightly better. TACTLESS computes them based on the total number of SAFs, therefore full SAF coverage cannot be achieved like with SPIN-TEST (SPIN-TEST works with redundancy-checked SAF list). TACTLESS performed in some cases better and in other

Table 3: Execution time comparison for random test

Circuit name	CPU time [s]		Time reduction
	serial method	deductive method	
alloc_outbound	0.19	0.07	63%
chu133	0.23	0.29	-26%
chu150	0.09	0.07	22%
converta	0.18	0.26	-44%
dff	0.05	0.00	100%
ebergen	0.07	0.01	86%
half	0.08	0.13	-63%
hazard	0.05	0.01	80%
master_read	0.70	0.39	44%
mp_forward_pkt	0.24	0.04	83%
nak_pa	0.32	0.08	75%
nowick	0.14	0.05	64%
ram_read_sbuf	0.25	0.06	76%
rcv_setup	0.07	0.02	71%
rpdft	0.04	0.00	100%
sbuf_ram_write	0.15	0.01	93%
sbuf_send_ctl	0.19	0.10	47%

Table 4: Memory requirement comparison for random test

Circuit name	Memory [kB]		Memory increase
	serial method	deductive method	
alloc_outbound	1668	1700	2%
chu133	1680	1780	6%
chu150	1660	1552	-7%
converta	1560	1744	12%
dff	1548	1672	8%
ebergen	1552	1700	10%
half	1548	1680	9%
hazard	1552	1692	9%
master_read	1732	1964	13%
mp_forward_pkt	1584	1608	2%
nak_pa	1684	1612	-4%
nowick	1664	1704	2%
ram_read_sbuf	1688	1756	4%
rcv_setup	1660	1664	0%
rpdft	1536	1640	14%
sbuf_ram_write	1688	1748	2%
sbuf_send_ctl	1684	1728	3%

cases worse than IB-TPG, but TACTLESS does not require support by any DfT method, therefore the area overhead is avoided, but sometimes the fault coverage is reduced too. The reason for lower fault coverages of TACTLESS is caused also by TPG for CR [8]. There are many uncontrollable/unobservable lines in CR, and the faults on those lines will be undetectable if undefined initial values are assumed. Possible solutions for increasing the fault coverage are: (1) modification of the TPG for CR to work with the initial values of ASDCs, and (2) use of DfT methods.

Table 6 compares TACTLESS to IB-TPG regarding test lengths. The test lengths were not published for SPIN-TEST. Again, TACTLESS performed in some cases better and in other cases worse than IB-TPG, but it is important to realize that TACTLESS executes sequential fault propagation too and does not require support by any DfT method (which would reduce the test lengths). Hence, it is possible to reduce the test lengths of TACTLESS with application of DfT methods. TACTLESS generates test patterns randomly before the deterministic phase. The random test pattern generation phase [6] ends when the latest test pattern does not detect at least 3 new SAFs. The increase of this number according to our experiments will not result in higher number of random test patterns nor in better SAF coverage because only the most easily detectable SAFs could be covered with this condition and the others always require deterministic TPG.

7. Conclusion

The aim of the dissertation thesis was to improve TPG of ASDCs and to contribute to their time- and cost-effective testing. Developed methods support the wider application of ASDCs, which improves the performance, the power consumption and the electromagnetic emission of future digital circuits.

The main scientific contribution is development of the new method for TPG (optimized for test length and area overhead) for wide spectrum of ASDCs. The contributions are (1) identification of unacceptable signal transitions before TPG, (2) reduced number of generated test patterns for CR of ASDC, (3) effective state justification on the gate level, (4) fast (optimized for test length) sequential fault propagation to outputs, (5) effective fault simulation.

It is possible to increase the fault coverages by improving the inner ATPG for CR. This improvement should be aimed at modification of FAN algorithm to work with the initial values of ASDCs. The use of DfT methods together with TACTLESS offers a way to further improve the fault coverages and to reduce the test lengths. Another interesting improvement would be an extension of TACTLESS to handle other fault models too.

Acknowledgments. This work was supported by Slovak national project VEGA 2/0135/08 Reliable Architectures and Digital Systems Testability.

References

[1] International technology roadmap for semiconductors, 2009 edition. `www.itrs.net/Links/2009ITRS/Home2009.htm`, [Accessed: 8.10.2010], 2009.

[2] J. A. Brzozowski, Z. Ésik, and Y. Iland. Algebras for hazard detection. In *Proc. 31st IEEE Int. Symp. Multiple-Valued Logic*, pages 3–12. 2001.

[3] M. L. Bushnell and V. D. Agrawal. *Essentials of Electronic Testing for Digital, Memory and Mixed-Signal VLSI Circuits*. Kluwer Academic Publishers, Massachusetts, USA, 2000.

[4] K.-T. Cheng and V. D. Agrawal. A partial scan method for sequential circuits with feedback. *IEEE Trans. Comput.*, 39(4):544–548, 1990.

[5] J. Cortadella, M. Kishinevsky, A. Kondratyev, L. Lavagno, and A. Yakovlev. Petrify: a tool for manipulating concurrent specifications and synthesis of asynchronous controllers. *IEICE Trans. Information and Systems*, 80(3):315–325, 1997.

[6] R. Dobai and E. Gramatová. Deductive fault simulation for asynchronous sequential circuits. In *12th Euromicro Conf. Digital System Design, Architectures, Methods and Tools*, pages 459–464. 2009.

[7] R. Dobai and E. Gramatová. A novel automatic test pattern

Table 5: Fault coverage comparison for deterministic test

Circuit name	SAFs	Undetected SAFs	SAF coverage		
			TACTLESS	SPIN-TEST	IB-TPG
alloc_outbound	58	2	96.6%	100.0%	100.0%
chu133	60	9	85.0%	96.9%	100.0%
chu150	40	2	95.0%	97.1%	92.3%
converta	54	14	74.1%	91.9%	91.2%
dff	34	0	100.0%	85.7%	55.6%
ebergen	46	6	87.0%	95.7%	38.5%
half	34	0	100.0%	100.0%	100.0%
hazard	40	6	85.0%	97.0%	44.4%
mp_forward_pkt	66	5	92.4%	100.0%	97.5%
nak_pa	76	0	100.0%	100.0%	100.0%
nowick	50	5	90.0%	100.0%	97.1%
ram_read_sbuf	84	3	96.4%	100.0%	100.0%
rcv_setup	36	1	97.2%	100.0%	95.0%
rpdft	26	0	100.0%	100.0%	100.0%
sbuf_ram_write	82	6	92.7%	100.0%	100.0%
sbuf_send_ctl	66	15	77.3%	94.9%	95.8%
qdi27cd	96	7	92.7%		89.0%
drAdder	102	3	97.1%		100.0%
dimsAdder	132	27	79.5%		98.9%
14Adder	278	57	79.5%		99.2%

Table 6: Test length comparison for deterministic test

Circuit name	Random test patterns	Deterministic test patterns	Total test patterns	
			TACTLESS	IB-TPG
alloc_outbound	3	20	23	25
chu133	3	40	43	29
chu150	4	16	20	8
converta	3	2	5	22
dff	4	2	6	5
ebergen	4	0	4	5
half	4	0	4	15
hazard	4	0	4	5
mp_forward_pkt	4	13	17	14
nak_pa	3	46	49	20
nowick	3	4	7	16
ram_read_sbuf	3	103	106	35
rcv_setup	3	8	11	13
rpdft	3	18	21	9
sbuf_ram_write	2	49	51	42
sbuf_send_ctl	3	13	16	37
qdi27cd	2	87	89	52
drAdder	3	70	73	118
dimsAdder	3	56	59	181
14Adder	3	198	201	315

generator for asynchronous sequential digital circuits. *Microelectron. J.*, 2010. In press. Doi:10.1016/j.mejo.2010.10.013.

[8] R. Dobai and E. Gramatová. Test pattern generation for the combinational representation of asynchronous circuits. In *2010 IEEE 13th Int. Symp. Design and Diagnostics of Electronic Circuits and Systems*, pages 323–328. 2010.

[9] A. Efthymiou. Redundancy and test-pattern generation for asynchronous quasi-delay-insensitive combinational circuits. In *Proc. IEEE Design and Diagnostics of Electronic Circuits and Systems*, pages 1–6. 2007.

[10] A. Efthymiou. Initialization-based test pattern generation for asynchronous circuits. *IEEE Trans. VLSI Syst.*, 18(4):591–601, 2010.

[11] H. Fujiwara and T. Shimono. On the acceleration of test generation algorithms. *IEEE Trans. Comput.*, 32(12):1137–1144, 1983.

[12] G. Gill, A. Agiwal, M. Singh, F. Shi, and Y. Makris. Low-overhead testing of delay faults in high-speed asynchronous pipelines. In *Proc. 12th IEEE Int. Symp. Asynchronous Circuits and Systems*, pages 46–56. 2006.

[13] P. Goel. An implicit enumeration algorithm to generate tests for combinational logic circuits. *IEEE Trans. Comput.*, 30(3):215–222, 1981.

[14] S. Hauck. Asynchronous design methodologies: An overview. *Proc. IEEE*, 83(1):69–93, 1995.

[15] C. Jeong and S. Nowick. Fast hazard detection in combinational circuits. In *Proc. 41st Design Automation Conf.*, pages 592–595.

2004.

[16] A. Khoche and E. Brunvand. Critical hazard free test generation for asynchronous circuits. In *Proc. 15th IEEE VLSI Test Symp.*, pages 203–208. 1997.

[17] H. Krad. Multi level combinational circuits and hazard detection. In *Proc. IEEE Southeastcon*, volume 2, pages 573–575. 1992.

[18] H. K. Lee and D. S. Ha. On the generation of test patterns for combinational circuits. Technical Report 12_93, Dept. of Elect. Eng., Virginia Polytechnic Inst., 1993.

[19] D. E. Muller and W. S. Bartky. A theory of asynchronous circuits. In *Proc. Int. Symp. Theory of Switching*, pages 204–243. Harvard University Press, 1959.

[20] Myers Research Group. Atacs online demo. `www.async.ece.utah.edu/atacs-bin/demo`, [Accessed: 8.10.2010], 1999.

[21] O. Novák, E. Gramatová, R. Ubar, and col. *Handbook of Testing Electronic Systems*. Czech Technical University Publishing House, 2005.

[22] O. Roig, J. Cortadella, M. Peiia, and E. Pastor. Automatic generation of synchronous test patterns for asynchronous circuits. In *Proc. 34th Design Automation Conf.*, pages 620–625. 1997.

[23] J. P. Roth. Diagnosis of automata failures: a calculus and a method. *IBM J. Res. Dev.*, 10(4):278–291, 1966.

[24] S. J. Russel and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, New Jersey, USA, 1995.

[25] F. Shi and Y. Makris. Fault simulation and random test generation for speed-independent circuits. In *Proc. 2004 Great Lakes Symp. VLSI*, pages 127–130, 2004.

[26] F. Shi and Y. Makris. SPIN-SIM: Logic and fault simulation for speed-independent circuits. In *Proc. 2004 Int. Test Conf.*, pages 597–606, 2004.

[27] F. Shi and Y. Makris. Spin-test: automatic test pattern generation for speed-independent circuits. In *IEEE/ACM Int. Conf. Computer Aided Design*, pages 903–908. 2004.

[28] F. Shi and Y. Makris. Testing delay faults in asynchronous handshake circuits. In *Proc. 2006 IEEE/ACM Int. Conf. Computer Aided Design*, pages 193–197. 2006.

[29] F. Shi and Y. Makris. Enhancing simulation accuracy through advanced hazard detection in asynchronous circuits. *IEEE Trans. Comput.*, 58(3):394–408, 2009.

[30] F. Shi, Y. Makris, S. Nowick, and M. Singh. Test generation for ultra-high-speed asynchronous pipelines. In *Proc. IEEE Int. Test Conf.*, pages 1–10. 2005.

[31] J. Sparso and S. Furber. *Principles of asynchronous circuit design: A systems perspective*. Kluwer Academic Publishers, Dordrecht, The Neederlands, 2001.

[32] F. te Beest and A. Peeters. A multiplexer based test method for self-timed circuits. In *Proc. 11th IEEE Int. Symp. Asynchronous Circuits and Systems*, pages 166–175. 2005.

[33] W. Toms. *Synthesis of Quasi-Delay-Insensitive Datapath Circuits*. Ph.d. thesis, University Manchester, School of Computer Science, Manchester, United Kingdom, 2006. `ftp://ftp.cs.man.ac.uk/pub/amulet/theses/Toms06_phd.pdf`, [Accessed: 8.10.2010].

Selected Papers by the Author

M. Baláž, R. Dobai, E. Gramatová. Delay Faults Testing. In R. Ubar, J. Raik, and H. T. Vierhaus, editors, *Design and Test Technology for Dependable Systems-on-Chip*. IGI Global, Hershey, 2010. In press.

R. Dobai, E. Gramatová. A Novel Automatic Test Pattern Generator for Asynchronous Sequential Digital Circuits. *Microelectron. J.*, 2010. In press. Doi:10.1016/j.mejo.2010.10.013.

R. Dobai, E. Gramatová. Deductive Fault Simulation Technique for Asynchronous Circuits. *Comput. Inform.*, 29(6): 1025–1043, 2010.

R. Dobai, E. Gramatová. Deductive Fault Simulation for Asynchronous Sequential Circuits. In *2009 12th EUROMICRO Conf. Dig. System Design, Architectures, Methods and Tools: DSD 2009*, pages 459–464, Patras, Greece, 2009.

R. Dobai, E. Gramatová. Test Pattern Generation for the Combinational Representation of Asynchronous Circuits. In *13th IEEE Int. Symp. Design and Diagnostics of Electronic Circuits and Systems*, pages 323–328, Wien, Austria, 2010.

R. Dobai. Testing of Delay Faults in Asynchronous Circuits. In *14th IEEE European Test Symp.: ETS'09*, Sevilla, Spain, 2009. CD.