Reduction of Fault Latency in Sequential Circuits by using Decomposition

Ilya Levin
i.levin@ieee.org

Benjamin Abramov
avramov@post.tau.ac.il

Vladimir Ostrovsky
vladio@post.tau.ac.il

Tel Aviv University, Israel

Abstract

The paper discusses a novel approach for reduction of fault detection latency in a self-checking sequential circuit. The Authors propose decomposing the finite state machine (FSM) which describes the sequential circuit of interest, thus obtaining a number of component FSMs respectively describing the number of component circuits. Being decomposed to the number of component circuits, the initial circuit becomes able to detect faults much faster since, at each specific moment of time, one of the component circuits (FSMs) is working and all the others are being tested. The paper deals with the following aspects: a) the decomposition procedure; b) evaluation of the proposed approach based on a fault injection simulation; c) estimation of trade-off between the reduction of latency and the required hardware overhead. Results of the study are tested on a number of standard benchmarks.

1. Introduction

Concurrent checking can be performed in circuits having ability to self-exam their operational "health" during normal functioning, and allows indicating the circuit's potential failures. While such an indication is highly desirable, designing of self-checking circuits is not trivial. Issues to be addressed in the design include the hardware cost, possible performance degradation, fault coverage, as well fault latency known as a time period between appearance of a fault and its manifestation.

In this paper we deal with sequential circuits that implement controllers. Specifically, we study the mentioned trade-offs in order to formulate guidelines for synthesis of controller circuits which should possess some required latency-overhead balance. The way we propose to study is a method of decomposing an FSM describing the controller into a network of interacting component FSMs in such way, that only one of the components is functioning at each specific moment of time while all the rest of the components are being tested by their self-testing means. Since the probability of fault detection in concurrently checking component circuits is higher than that in the large initial circuit, it seems obvious that the proposed network provides the desired reduction of the fault latency in comparison with the latency of the initial FSM.

The paper addresses the following questions:

1. How to decompose the initial FSM to the equivalent decomposition network?
2. How to provide the proper functioning of the decomposition network?
3. What is an estimated value of the latency reduction for the proposed decomposition?
4. What is the "price" for the above latency reduction it terms of the hardware overhead?

The paper is built as follows. After reviewing the related works in Section 2, Section 3 presents the proposed method of constructing the decomposition network and ensuring its functioning. Description of the proposed method of latency estimation for the decomposed FSM is provided in Section 4. Section 5 gives specific benchmarks results of the latency estimation. Conclusions are provided in Section 6.

1550-5774/07 $25.00 © 2007 IEEE
DOI 10.1109/DFT.2007.24

2. Related works

A number of known approaches to concurrent errors detection, on-line testing and self-checking may be classified by their position within the trade-off space between hardware overhead and fault detection latency [1]. The majority of the approaches represent one of the two ends of this space. The main difficulty at the low end of the space is the necessity to apply all possible input combinations before obtaining any indication of the fault manifestation, which leads to significant fault latency. At the high end of the trade-off space, there are methods that check the circuit at every clock to guarantee the zero error detection latency. However, such methods usually require the overhead cost greater than the duplication cost. To the best of our knowledge, while there are a number of approaches in between the two ends, methods for synthesis of sequential circuits that provide a desirable balance between the overhead and the fault latency were never investigated.

FSM decomposition techniques were studied as a means for optimization [2, 3], low-power design [4], and self-checking design [5]. In [3], a problem of minimizing interconnections was firstly studied. The author introduced a supervisor FSM into the decomposition network, which FSM coordinates functioning of the decomposition network and allows encoding of interconnecting signals separately for each of component FSMs (CFSMs) of the network. Partitioning of the FSM into a pair of interacting portions, for reducing the fault latency, was reported in [6]. In [7], an analytical method for calculation of the FSMs fault latency was presented. A design of concurrent error detection FSMs with bounded latency was proposed in [8]. A statistical analysis of the FSM decomposition network behavior was demonstrated in [9].

The present paper describes a decomposition method for reduction of fault latency in a sequential circuit. We decompose the circuit's FSM into three components. We organize interconnections between the components by introducing the mentioned supervisor FSM. In order to provide interaction between the CFSMs of the network, additional hardware should comprise the supervisor FSM and some recourse required for performing extra transitions within the components. Using this type of decomposition, we study trade-off between the two main parameters – the latency and the overhead for decomposed sequential circuits.

3. Decomposition network

In this section, we introduce some basic notations and describe the general idea of the proposed decomposition. We use the FSM notation taken from [2] for description both the initial sequential circuit and the decomposition network of component sequential circuits.

Let an FSM describing a certain sequential circuit is defined by its transition table shown in Tables 1 and 2. In the tables: a_m - a present state, a_s - a next state, $X(a_m, a_s)$ – a transition function, i.e. a Boolean function, which is equal to 1 when FSM moves from state a_m to state a_s, $Y(a_m, a_s)$ – a list of output functions, which are equal to 1 on the transition of the FSM from a_m to a_s, h – a serial number of the FSM transition.

Let $\pi = \{a_1a_2a_3; a_4a_5a_6; a_7a_8a_9\}$ be a predetermined partition on the set of the FSM states. We put the desirable decomposition network into a correspondence with the initial FSM and the partition π. Each of the future component FSMs of the decomposition network corresponds to a specific block of the partition $\tilde{\pi}$. The decomposition network consists of U component FSMs S_u $(u=1,...,U)$ and one specific supervisor FSM (SFSM). Each of the component FSMs produces an output vector in response to an input vector, and receives/produces connecting signals, providing by this interaction between component FSMs of the decomposition network. Actually, the component FSM (CFSM) transforms output connection signals of the component FSMs into input connecting signals of the component FSMs.

Table 1. Transition table of the initial FSM

a_m	a_s	$X(a_m,a_s)$	$Y(a_m,a_s)$	h
a_1	a_1	$x_3x_4x_5$	–	1
	a_5	$x_3x_4\overline{x_5}$	y_1,y_2	2
	a_7	$x_3\overline{x_4}$	y_1,y_3	3
	a_9	$\overline{x_3}$	y_2	4
a_2	a_1	$x_5x_6x_7$	y_2,y_3,y_4	5
	a_7	$x_5x_6\overline{x_7}$	y_3	6
	a_6	$x_5\overline{x_6}$	y_2,y_3	7
	a_6	$\overline{x_5}$	y_2	8
a_3	a_6	x_5x_6	y_2,y_3	9
	a_7	$x_5\overline{x_6}$	y_3,y_4	10
	a_8	$\overline{x_5}\,\overline{x_6}$	y_2	11
	a_2	$\overline{x_5}x_6$	y_3,y_4	12
a_4	a_3	x_6x_7	y_2	13
	a_5	$x_6\overline{x_7}$	y_2,y_3	14
	a_7	$\overline{x_6}x_7$	y_3	15
	a_8	$\overline{x_6}\,\overline{x_7}$	y_3,y_4	16

Table 2. Transition table of the initial FSM (contd)

a_m	a_s	$X(a_m,a_s)$	$Y(a_m,a_s)$	h
a_5	a_2	x_1x_2	y_3,y_4	17
	a_3	$x_1\overline{x_2}$	y_4,y_5	18
	a_4	$\overline{x_1}$	y_3,y_5	19
a_6	a_2	x_4x_5	y_1,y_3	20
	a_5	$x_4\overline{x_5}$	y_2	21
	a_7	x_4	y_1,y_3	22
a_7	a_4	x_3x_2	y_2	23
	a_5	$x_3\overline{x_2}$	y_1,y_3	24
	a_6	$\overline{x_3}$	y_3,y_4	25
a_8	a_5	x_3x_4	y_2	26
	a_6	$x_3\overline{x_4}$	y_1,y_3	27
	a_7	$\overline{x_3}x_5$	y_1,y_3	28
	a_8	$\overline{x_3}\,\overline{x_5}$	–	29
a_9	a_3	x_1x_3	y_3	30
	a_4	$x_1\overline{x_3}$	y_3,y_5	31
	a_5	$\overline{x_1}$	y_3,y_4	32

The decomposition network works as follows. At any clock, one and only one component FSM is functioning in a *work mode*. We call this component FSM as *working FSM*. All the rest of components FSMs of the network are functioning in the *test mode*. We call all such FSMs as *tested FSMs*. While applying a current input vector to the working FSM leads to performing corresponding transition and output functions, applying the input vectors to the tested FSMs performs their testing.

A schematic diagram of the decomposition network is presented in Figure 1.

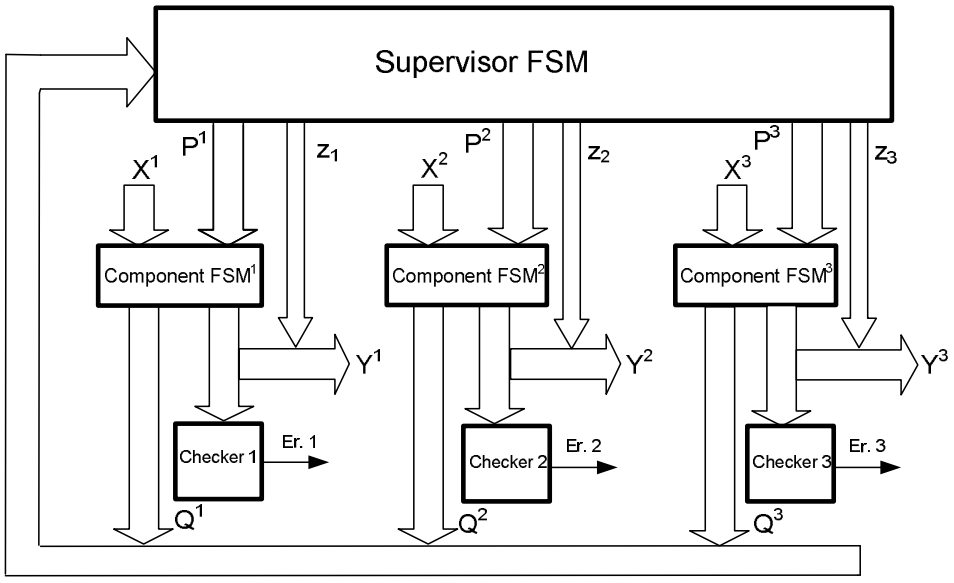

Figure 1. Schematic diagram of the FSM Decomposition network

In this figure, each of the component FSMs is connected with SFSM by Q^u output and P^u input lines. Outputs Y^u of each of the component FSMs are checked by a self-checking checker and "enabled" by signals z_u sent by the SFSM.

4. Decomposition model

Let $\pi=\{A^1,...,A^U\}$ be a partition on the set of states of the initial FSM, U is a number of partition blocks.

Each partition block corresponds to a certain CFSM of the network, defined as follows:

1) the set of states of the CFSM is the set of states of a corresponding partition block;

2) if the transition occurs between states of the same partition block, the transition and the output functions of the CFSM respectively correspond to the transition and the output functions of the initial FSM; if the initial state and the final state of the transition belong to different partition blocks, then the CFSM containing the initial state moves to a predefined state (one of the states of the CFSM), and the CFSM containing the final state of the transition is caused, by a specific connection signal, to move from the special predefined state into the final state of the transition;

3) each input vector of a CFSM comprises an external input vector and a connecting input vector;

4) each output vector of a CFSM comprises an external output vector and a connecting output vector.

The proposed decomposition allows independent encoding of input connecting signals. The independent encoding minimizes the number of external lines required for transmitting variables of the connecting vectors. It becomes possible by introducing into the FSM network a SFSM, which converts output connecting signals to input connecting signals of the CFSMs.

The SFSM is constructed as follows:

1) states of the SFSM correspond to partition blocks of the initial FSM;

2) transitions of the SFSM correspond to transitions of the initial FSM between states of different partition blocks;

3) transitions between states of the same partition block does not change the current state of the SFSM;

4) the number of different input vectors of the input alphabet of the SFSM is equal to that of the largest alphabet of output connecting signals of component FSMs;

5) each output set of SFSM comprises input connection signals of the corresponding CFSMs.

CFSMs of the decomposition network interact as follows. At each clock, a certain CFSM is functioning (i.e. is in the work mode). Its output signals are "enabled". All remaining CFSMs are in their test mode. Some current input vector initiates the testing process. Output signals of the CFSMs being tested are "disabled". Nevertheless, these signals are checked by the corresponding checkers. After moving into a first state from which it's testing mode starts, a specific CFSM sends a connecting vector signal Q^u to inputs of the SFSM. At the same clock, the SFSM produces an output vector P^u signal that initiates functioning (work mode) of another one of the component FSMs of the decomposition network. The SFSM then moves to its state corresponding to the presently functioning component FSM.

Let us define the decomposition network formally.

Define a component FSM $S^u\left(B^u,X^u,Y^u,\delta^u,\lambda^u,a_1\right)$ as follows.

1. A set of states of CFSM S^u : $B^u=A^u$.

2. A set of input variables $X^u=\bigcup_{a_m\in A^u}X(a_m)\bigcup P^u$, where: $X\left(a_m\right)$ is a set of input variables of the initial FSM sampled at transitions from a_m; P^u is a set of additional connecting input variables arriving to the input of S^u from the output of the SFSM. In our example: $X^1=\{x_1,x_2,x_3,x_4,x_5,x_6,x_7,p_1,p_2,p_3\}$; $X^2=\{x_2,x_3,x_4,x_5,p_4,p_5,p_6\}$;

$X^3=\{x_1,x_3,x_5,x_6,x_7,p_7,p_8,p_9\}$.

3. A set of output variables $Y^u = \bigcup_{a_m \in A^u} Y(a_m) \bigcup Q^u$, where $Y(a_m)$ – is a set of output variables of the initial FSM, generated at transitions from a_m; Q^u – is a set of additional connecting output variables arriving from the output of S^u to the input of the SFSM. In our example:

$Y^1=\{y_1, y_2, y_3, y_4, y_5, q_1, q_2, q_3, q_4\}; Y^2=\{y_1, y_2, y_3, y_4, q_1, q_2, q_3, q_4\}; Y^3=\{y_2, y_3, y_4, y_5, q_1, q_2, q_3, q_4\}.$

4. The transition function and the output functions of the component FSM are defined as follows:

a) $\left(\delta(a_m, X_h)=a_j\right)$ & $\left(\lambda(a_m, X_h)=Y_t\right)$ & $\left(a_m, a_j \in A^u\right) \Rightarrow$

$\Rightarrow \left(\delta^u(a_m, X_h)=\delta(a_m, X_h)=a_j\right)$ & $\left(\lambda^u(a_m, X_h)=\lambda(a_m, X_h)=Y_t\right);$

b) $\left(\delta(a_m, X_h)=a_j\right)$ & $\left(\lambda(a_m, X_h)=Y_t\right)$ & $\left(a_m \in A^u, a_j \in A^k\right)$ & $(u \neq k) \Rightarrow$

$\Rightarrow \left(\delta^u(a_m, X_h \,\&\, P_0)=a_{(l)}\right)$ & $\left(\lambda^u(a_m, X_h)= Y_t \cup Q_j, Q_j \in Q^u\right)$ &

& $\left(\delta^k(a_l, P_j)=\delta(a_m, X_h)=a_j\right)$ & $\left(\lambda^k(a_l, P_j)=Y_0\right)$

Where: $a_{(l)}$ is a predefined initial testing state of the component FSM, where the FSM begins the testing mode. This state is one of the states of a component FSM. Notice that not every state may be chosen as $a_{(l)}$. The necessary condition is to save the reachability property for $a_{(l)}$, which means that $a_{(l)}$ has to be reachable from other states of the component FSM. a_l is a certain state of the component FSM, which state the FSM leaves when moves to the required transition state a_j, $P_j = p_1^{\alpha_1} \,\&\, \ldots \,\&\, p_T^{\alpha_T}$, where $\alpha_1 \ldots \alpha_T$, is a binary representation of j-th vector of connecting input variables of S^u.

The initial state of the component FSM is defined as follows:

$$\left(a_1 \in A^u\right) \Rightarrow \left(a_1^u = a_1\right); \quad \left(a_1 \notin A^u\right) \Rightarrow \left(a_1^u = a_{(l)}\right).$$

Transition tables of component FSMs S^1, S^2 and S^3 are presented in Tables 3, 4 and 5, respectively. The "-" symbol in the last column corresponds to the empty vector Q_0.

The supervisor FSM (SFSM) $\left(C, P, Q, \delta^C, \lambda^C, C_1\right)$ is defined as follows.

1. The set C of states of the SFSM corresponds to the partition π. In the example: $\pi = \{a_1a_2a_3; a_4a_5a_6; a_7a_8a_9\}$, $C=\{C_1, C_2, C_3\}$.

2. Input variables of the SFSM are additional output variables of the component FSMs $P = \bigcup_{u=1,\ldots,U} P^u$. In the example: $P=\{p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8, p_9\}$.

3. Output variables of the SFSM are additional input variables of the component FSMs. In the example: $Q=\{q_1, q_2, q_3\}$.

4. Functions of the SFSM are defined as follows:

a) $\left(\delta(a_i, X_h)=a_j\right)$&$\left(\lambda(a_i, X_h)=Y_t\right)$ & $\left(a_i, a_j \in A^u\right) \Rightarrow \left(\delta^C(C_u, Q_0)=C_u\right)$ & $\left(\lambda^C(C_u, Q_0)=P_0 \cup Z_u\right).$

Where Z_u is 1-hot code having 1 value in the u-th position.

b) $\left(\delta(a_1, X_h) = a_j\right) \& \left(\lambda(a_1, X_h) = Y_t\right) \& \left(a_i \in A^u, a_j \in A^k, u \neq k\right) \Rightarrow$
$\Rightarrow \left(\delta^C\left(C_u, Q_j\right) = C_k\right) \& \left(\lambda^C\left(C_u, Q\right) = P_0 \cup Z_k\right).$

Where $Q_j = q_1^{\alpha_1} \& \ldots \& q_T^{\alpha_T}$, where $\alpha_1 \ldots \alpha_T$, is a binary representation of j-th vector of output connecting variables of S^u.

5. The initial state of the SFSM is defined as follows: $\left(a_i \in A^u\right) \quad C_1^u = C_u$. In our example: $C_1^u = C_1$.

The transition table of the SFSM for our example is shown in Table 6.

Table 3. Component FSM S^1

a_m	a_s	$X(a_m,a_s)$	$P(a_m,a_s)$	$Y(a_m,a_s)$	$Q(a_m,a_s)$	h
a_1	a_1	$x_3x_4x_5$	P_0	–	–	1
	a_5	$x_3x_4\overline{x_5}$	P_0	y_1y_2	–	2
	$a_{(5)}$	$x_3\overline{x_4}$	P_0	y_1y_3	Q_7	3
	$a_{(5)}$	$\overline{x_3}$	P_0	y_2	Q_9	4
	a_2	1	P_2	–	–	5
	a_5	1	P_5	–	–	6
a_2	a_1	$x_5x_6x_7$	P_0	$y_2y_3y_4$	–	7
	$a_{(5)}$	$x_5x_6\overline{x_7}$	P_0	y_3	Q_7	8
	$a_{(5)}$	$x_5\overline{x_6}$	P_0	y_2y_3	Q_6	9
	$a_{(5)}$	$\overline{x_5}$	P_0	y_2	Q_3	10
	a_2	1	P_2	–	–	11
	a_5	1	P_5	–	–	12
a_5	a_2	x_1x_2	P_0	y_3y_4	–	13
	$a_{(5)}$	$x_1\overline{x_2}$	P_0	y_4y_5	Q_3	14
	$a_{(5)}$	$\overline{x_1}$	P_0	y_3y_5	Q_4	15
	a_2	1	P_2	–	–	16
	a_5	1	P_5	–	–	17

Table 4. Component FSM S^2.

a_m	a_s	$X(a_m,a_s)$	$P(a_m,a_n)$	$Y(a_m,a_n)$	$Q(a_m,a_n)$	h
a_6	$a_{(8)}$	x_4x_5	P_0	y_1y_3	Q_2	1
	$a_{(8)}$	$x_4\overline{x_5}$	P_0	y_2	Q_5	2
	a_7	$\overline{x_4}$	P_0	y_1y_3	–	3
	a_6	1	P_6	–	–	4
	a_7	1	P_7	–	–	5
	a_8	1	P_8	–	–	6
a_7	$a_{(8)}$	x_3x_2	P_0	y_2	Q_4	7
	$a_{(8)}$	$x_3\overline{x_2}$	P_0	y_1y_3	Q_5	8
	a_6	$\overline{x_3}$	P_0	y_3y_4	–	9
	a_6	1	P_6	–	–	10
	a_7	1	P_7	–	–	11
	a_8	1	P_8	–	–	12
a_8	$a_{(8)}$	x_3x_4	P_0	y_2	Q_5	13
	a_6	$x_3\overline{x_4}$	P_0	y_1y_3	–	14
	a_7	$\overline{x_3}x_5$	P_0	y_1y_3	–	15
	a_8	$\overline{x_3}\,\overline{x_5}$	P_0	–	–	16
	a_6	1	P_6	–	–	17
	a_7	1	P_7	–	–	18
	a_8	1	P_8	–	–	19

5. Experimental evaluation of the FSM decomposition

This section describes a fault injection simulation tool developed for experimental evaluation of the proposed decomposition approach. The section further provides experimental results and also comprises analysis of the results.

The fault injection is a known technique of digital systems validation that is defined in the following way [10]:

Fault injection is the validation technique of the Dependability of Fault Tolerant Systems which consists in the accomplishment of controlled experiments where the observation of the system's behavior in presence of faults is induced explicitly by the writing introduction (injection) of faults in the system.

We use a simulated fault injection technique, i.e., the technique where the system under test is simulated in another computer system [11, 12, and 13]. The faults are manifested by alerting some logical values during the simulation.

Table 5. Component FSM S^3

a_m	a_s	$X(a_m,a_s)$	$P(a_m,a_s)$	$Y(a_m,a_s)$	$Q(a_m,a_s)$	h
a_3	$a_{(9)}$	x_5x_6	P_0	y_2y_3	Q_6	1
	$a_{(9)}$	$x_5\overline{x_6}$	P_0	y_3y_4	Q_7	2
	$a_{(9)}$	$\overline{\overline{x_5x_6}}$	P_0	y_2	Q_8	3
	$a_{(9)}$	$\overline{x_5}x_6$	P_0	y_3y_4	Q_2	4
	a_9	1	P_9	1	–	5
	a_3	1	P_3	1	–	6
	a_4	1	P_4	1	–	7
a_4	a_3	x_6x_7	P_0	y_2	–	8
	$a_{(9)}$	$x_6\overline{x_7}$	P_0	y_2y_3	Q_5	9
	$a_{(9)}$	$\overline{x_6}x_7$	P_0	y_3	Q_7	10
	$a_{(9)}$	$\overline{\overline{x_6x_7}}$	P_0	y_3y_4	Q_8	11
	a_9	1	P_9	–	–	12
	a_3	1	P_3	–	–	13
	a_4	1	P_4	–	–	14
a_9	a_3	x_1x_3	P_0	y_3	–	15
	a_4	$x_1\overline{x_3}$	P_0	y_3y_5	–	16
	$a_{(9)}$	$\overline{x_1}$	P_0	y_3y_4	Q_5	17
	a_9	1	P_9	–	–	18
	a_3	1	P_3	–	–	19
	a_4	1	P_4	–	–	20

Table 6. Supervisor FSM (SFSM)

C_m	C_s	$Q(C_m,C_s)$	$P(C_m,C_s)$	z_1,z_2,z_3	Z
C_1	C_2	Q_6	P_6	010	Z_2
	C_2	Q_7	P_7	010	Z_2
	C_3	Q_3	P_3	100	Z_3
	C_3	Q_4	P_4	100	Z_3
	C_3	Q_9	P_9	100	Z_3
	C_1	Q_0	P_0	001	Z_1
C_2	C_1	Q_2	P_2	001	Z_1
	C_1	Q_5	P_5	001	Z_1
	C_3	Q_4	P_4	100	Z_3
	C_2	Q_0	P_0	010	Z_2
C_3	C_1	Q_2	P_2	001	Z_1
	C_1	Q_5	P_5	001	Z_1
	C_2	Q_6	P_6	010	Z_2
	C_2	Q_7	P_7	010	Z_2
	C_2	Q_8	P_8	010	Z_2
	C_3	Q_0	P_0	100	Z_3

5.1. Fault injection environment

A specific VHDL based fault injection environment was developed for evaluation of the proposed decomposition technique and its efficiency from the point of view of the fault latency reduction.

A three-input fault injection circuit (see Fig. 2) is proposed for injecting faults between any two connected gates *A*, *B* of a combinational portion of the sequential circuit under test. The fault injection circuit is able to control two signals K_0 and K_1, and to assign a desired logical value between the two gates *A* and *B*. The fault injection circuit and one point of its embedding into a combinational circuit are shown in Figure 2.

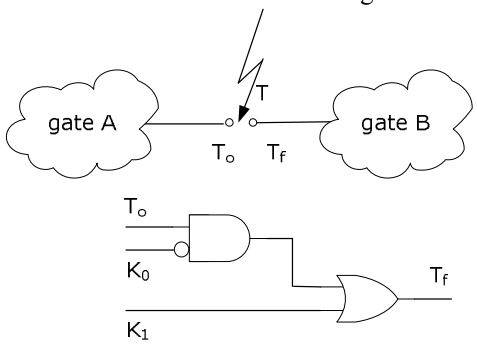

Figure 2. Three-input circuit for simulation of the fault injection

In this figure: point T - is the embedding point. T_0 and T_C correspond to points with the original and faulty values correspondingly. K_0 and K_1 inputs allow simulating the s-a-0 and s-a-1 faults. It is assumed that the default values of K_0 and K_1 are zeros and the initial circuit is functioning properly. Assigning 0 or 1 to K_0 or K_1 initiates s-a-0 and s-a-1 faults on the initial circuit poles. It simulates a fault from a specific predefined set.

5.2. Fault injection experiments

The aim of our experiments was to study the response of the VHDL model of a sequential circuit in presence of a permanent stack-at fault. The injection conditions we used were:

1. **Number of faults**: one fault at a time.
2. **Fault type:** The injected faults were transient and permanent s-a-1 and s-a-0 faults.
3. **Injection place**: Any inter-gate connection was provided with the fault injection circuit. The place of fault injection was selected randomly; the selection was repeated 1000 times for each sequential circuit under test.
4. **Sequential circuits** under test: each comprises from 200 to 400 inter-gate connections (all equipped with the fault injection circuits);
5. **Fault duration:** Three types of faults may be applied:
   a. Transient fault with a duration generated randomly;
   b. Transient fault with a fixed duration;
   c. Permanent faults.
6. **Analysis of results**: From the sample data, the following parameters were obtained:
   a. Latency of a fault both before and after the decomposition;
   b. Percentage of the latency reduction $\Omega_L = (L_B / L_A) \cdot 100\%$ as a result of the decomposition.

The developed tool provides random selection of a fault injection point. Detection of the fault manifestation on the output of the sequential circuit is performed on the simulated initially fault-free sequential circuit, upon injecting fault via the fault injection circuit. Outputs of the fault injection circuit and of the sequential circuit are compared with one another.

The experiments were performed for the initial sequential circuit and for the decomposition net of that circuit. The fault injection time is a controllable variable and may be changed. It allows simulating different kinds of faults (permanent, transient, intermitted). Fault injection points may be chosen both randomly and arbitrary.

Obviously, the latency reduction is achieved by the decomposition. However, the decomposition requires an additional overhead in turn, which is the "price" of the latency reduction. To study the trade-off between the latency and the overhead we insert the corresponding overhead data into Table 7, to summarize results of the experiments. The results were obtained for Altera Cyclone II FPGA.

The first column of Table 7 contains the benchmark's title; the next three columns indicate the hardware overhead (a number of logic elements (LE)) and the fault latency of an FSM implementation of the corresponding benchmark before the decomposition. The next three columns correspond to the same parameters after the proposed decomposition. The last three columns of Table 7 indicate the percentage of the overhead increase and the latency reduction.

Table 7. Benchmark results of the decomposition

Benchmark	Before decomposition			After decomposition			After/Before (%)		
	LE	Lat (s-a-1)	Lat (s-a-0)	LE	Lat (s-a-1)	Lat (s-a-0)	Ov	Lat (s-a-1)	Lat (s-a-0)
bbtas	9	2.02	4.1	19	1	1.83	111	50	45
ex6	95	2.25	5.21	237	1.21	2.02	149	54	39
bbsse	37	2.11	3.98	86	1	1.44	132	47	36
beecount	27	2.33	4.75	51	1.07	1.77	89	46	37
dk512	18	1.89	3.97	32	1	1.41	78	53	36
tav	9	1.98	4.97	15	1	1.98	67	51	40
s510	72	1.88	4.05	151	1	1.81	110	53	45
pma	122	2.07	3.89	217	1	1.51	78	48	39
dk14	48	2.12	5.34	104	1.03	2.21	117	49	41
sse	42	2.34	4.4	93	1	1.6	121	43	36
Average	**47.9**	**2.099**	**4.466**	**100.5**	**1.031**	**1.758**	**105**	**49**	**39**

The experiments show that the average latency reduction for the proposed method of decomposition (in the case of three CFSMs) is of about 50% for permanent s-a-1 faults, and of about 60% for permanent s-a-0 faults. At the same time, the overhead increase required for such a reduction is of about 100%. The certain value of the latency reduction and the certain value of the overhead increase will depend on the number of CFSMs in the decomposition network and, most probably, will grow with the number of components.

6. Conclusions

We have proposed the idea to reduce fault latency in a sequential circuit described by an FSM by decomposing the FSM into a number of interacting FSMs (a decomposition network). While one of the CFSMs of the decomposition network is working all others are testing themselves. We have described a method for decomposition of an arbitrary FSM into a network comprising a number of CFSMs and a specific SFSM that organizes the proper interaction between the CFSMs.

The two main research questions "how the decomposition reduces the latency?" and "what is the price of the latency reduction?" were addressed. A specific VHDL based fault injection tool has been developed for studying the above questions. Stack-at faults were injected randomly in several points of the circuit.

Our experiments were conducted using a number of standard benchmarks and have answered the research questions. For partitioning the initial FSM into three CFSMs, the average fault latency reduction is of about 40-50%, which is slightly greater than the corresponding increase of the hardware overhead required for performing the decomposition.

We have shown results of our study of the FSM decomposition just for the case of permanent faults. As mentioned before, our tool allows injecting permanent/transient faults.

Based on the above, the following future research directions may be addressed:

- Study and development of a method of searching of a partition π on the set of FSM's states optimizing the latency reduction.
- Study of the influence of the number of components in the decomposition on its efficiency (latency reduction and required overhead).
- Study of behavior of the decomposed FSM upon a transient fault injection.
- Investigation of a recovering (self-healing) ability of the decomposed sequential circuits.

7. References

[1] Drineas, P., Makris, Y., "Non-intrusive design of concurrently self-testable FSMs", Proceedings of the 11th Asian Test Symposium, 2002, 33- 38.

[2] S. Baranov. Logic Synthesis for Control Automata. Kluwer Academic Publisher. Dordrecht/Boston. 1994.

[3] Levin I. "Decompositional Design of Automata Based on PLA with Memory". Automatic Control and Computer Sciences, Vol. 20, No. 2, 1986, 61-68.

[4] José C. Monteiro, Arlindo L. Oliveira, "Implicit FSM Decomposition Applied to Low-Power Design", IEEE Transaction on Very Large Scale Integration (VLSI) System, Vol. 10, No. 5, October 2002, 560-565.

[5] Busaba F. Y. and Lala P. K. "On Interacting Finite State Machine Design with Self-Checking Capability". Proceedings of SSST '93 24-th Southeastern Symposium on System Theory, 1993, 418-422.

[6] Karpovsky, M., Levin, I., Sinelnikov, V. "Decomposition Approach to Designing FPGA-Based Self-Checking Control Units", 6th IEEE International On-Line Testing Workshop, 2000.

[7] Goot, R., Levin, I., Ostanin, S., "Fault Latencies of Concurrent Checking FSMs", Euromicro Symposium on Digital System Design (DSD'02), 2002, p. 174.

[8] Almukhaizim, S. Drineas, P. Makris, Y. "On concurrent error detection with bounded latency in FSMs", Proceedings of Design, Automation and Test in Europe Conference, 2004, Vol.1, 596- 601.

[9] Goot, R., Levin, I., Ostanin, S. (2003). "Statistical Analysis of Decomposition Automata", Automatic Control and Computer Science. Vol. 37, No. 4, 6-13.

[10] Baraza, J. C., Gracia, J., Gil, D. and Gil P. J. "A prototype of a VHDL-based fault injection tool: description and application", Journal of Systems Architecture, Vol. 47, Issue 10, 2002, 847-867.

[11] Choi, G. S., Iyer, R. K., Carreno, V. A. "Simulated fault injection: a methodology to evaluate fault tolerant microprocessor architectures". IEEE Transactions on Reliability, Oct 1990, Volume: 39, (4) 486-491.

[12] Boue, J. Petillon, P. Crouzet, Y. "MEFISTO-L: a VHDL-based fault injection tool for the experimental assessment of fault tolerance". 28th Annual International Symposium on Fault-Tolerant Computing, 1998, Digest of Papers, 168-173.

[13] NA Kanawati, GA Kanawati, J Abraham. "Dependability evaluation using hybrid fault/error injection", IEEE International Computer Performance and Dependability Symposium (IPDS'95), 1995, p. 0224.