UDC 629.7.014-519.05.083:519.713 **doi: 10.32620/reks.2023.3.14**

Valery SALAUYOU

Bialystok University of Technology, Bialystok, Poland

STRUCTURAL MODELS OF MEALY FINITE STATE MACHINES DETECTING FAULTS IN CONTROL SYSTEMS

*The **subject matter** of this article is a control system for unmanned aerial vehicles (UAVs) whose mathematical model is a finite state machine (FSM). The **goal** is to develop FSM structural models that enable (1) detection of multiple faults of FSM elements caused by an electromagnetic pulse or laser beam, and (2) prevent negative impacts on the controlled object. The **tasks** to be solved are as follows: to develop FSM structural models to detect invalid input vector X for the whole FSM and in each state, to detect invalid output vector Y for the whole FSM, at each transition and in each state, invalid code of the present (current) state, invalid code of the next state, and invalid transition between states; to determine the possible causes of the faults, which can be the failure in the logic Φ of forming the code of the next state, the invalid input vector X, the failure in the feedback circuit, the failure in the logic Ψ of forming the output vector, the failure in the state register R, the failure in the wire between the FSM input and the input of the logic Ψ; development of a combined structural model for the detection of all listed faults with a minimum number of additional combinational circuits, as well as a structural model that combines all additional combinational circuits. The **methods** used are: the theory of finite state machines, structural models of FSMs, state encoding methods of FSMs, representation methods of FSMs, and Verilog hardware description language. The following **results** were obtained: (1) the Mealy FSM structural models were developed to detect all the above mentioned faults, (2) the combined FSM structural models were developed, and (3) the possible causes of faults detected by each FSM structural model were identified. Experimental studies have shown that for the presented FSM structural models, the area overhead averages 3-23%, for one-hot encoding of FSM states, and 2-8%, for binary encoding of FSM states. **Conclusions**. The scientific novelty of the obtained results consists in the following for the first time FSM faults that are not caused by radiation and cosmic rays but by an electromagnetic pulse affecting the control device are considered; the number of faults is not limited for the state codes as well as for the input and output vectors; the faults can be detected not only in the state register R but also in the input vector X, in the logic Φ of generating the next state code, in the logic Ψ of generating the output signals, and in the feedback circuit; the invalid transitions of FSMs and the transitions to invalid states are also detected; the proposed structural models not only detect FSM failures but also prevent their negative impact on the controlled object; combined structural models allow simultaneous detection of faults in all elements of the FSM. Future research will focus on developing structural models for correcting FSM failures.*

Keywords: *structural model; fault detection; finite state machine (FSM); area overhead; control system; unmanned aerial vehicles (UAVs); field programmable gate array (FPGA).*

Introduction

In recent years, unmanned aerial vehicles (UAV) have an increased popularity in all fields of applications, including military conflicts [1, 2]. One method of neutralizing UAV is to use a powerful electromagnetic pulse (EMP) [3]. An effective way to protect the UAV from EMP impacts is to implement the control device of the UAV as a fault-tolerant finite state machine (FSM).

Fig. 1 shows the traditional structural model of FSMs, where X is the input signal (the input vector), Y is the output signal (the output vector), R is the state register in which the code of the present (current) state is stored; Φ is the combinational circuit (logic) determining the code of the next state; Ψ is the combinational circuit determining the values of the output signals. Note that the output of register R is connected by feedback to the input of combinational circuit Φ. For Moore FSMs, the values of the output signals are formed on the basis of the present state code; however, for Mealy FSMs, the values of the output signals are formed on the basis of the present state code and the values of the input signals; therefore, for Mealy FSMs, the inputs X are connected to the inputs of the combinational circuit Ψ. The clock signal clk for register R is generated using the generator Oscillator.

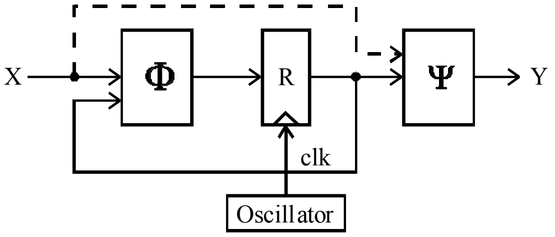

Fig. 1. Traditional structural model of FSMs

The following faults may occur in the FSM because of the EPM impact:

– at the inputs: an invalid input vector X;

– in the R register: an invalid present state code;

– in the feedback circuit: an invalid present state code;

– in the logic Φ: an invalid next state code;

– in logic Ψ: an invalid output vector;

– in the clock circuit: no clock signal;

– in the generator Oscillator: no clock signal.

In addition, there may be erroneous behavior of the FSM: transition of the FSM to invalid states, as well as invalid transitions to valid states (Fig. 2).

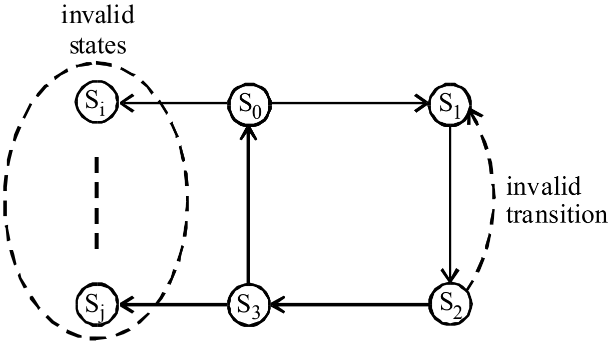

Fig. 2. Errors in the behavior of the FSM

A lot of studies has been conducted to build fault-tolerant FSMs. Most methods provide protection from single event upsets (SEUs), which are caused by radiation and cosmic rays. The SEUs change the contents of flip-flops or memory cells, while the primary inputs are always considered correct. However, the EPM is characterized by the following features when it impacts on FSMs:

– significant duration of the exposure time compared with a space particle;

– impacts simultaneously on all elements of the FSM;

– generates not short-lived SEUs but long-lasting multiple faults;

– affects mainly the wires (inputs, outputs, a feedback circuit);

– rarely changes the contents of registers or memory cells.

Note that modern development techniques for digital devices [4] differ significantly from the approaches used a few decades ago. As a result, the synthesis of the device is reduced to a correct description of the device behavior in hardware description language (HDL) [5]. At the same time, the traditional stages of the FSM synthesis are eliminated: encoding of states; forming of logical (Boolean) equations for the combinational circuit Φ and Ψ; minimization, factorization, and decomposition of the logic. All these stages are performed automatically using the synthesis tools. However, redundant logic aimed at building a fault-tolerant FSM can be removed from the design because of automatic optimization performed by synthesis tools. Therefore, new approaches to the design of fault-tolerant FSM are needed. However, before correcting the FSM failure using any method, the failure must be detected.

Since the most general model of an FSM is the Mealy FSM, in this paper, new structural models are proposed for detecting faults of Mealy FSMs. The proposed structural models allow the detection of multiple faults in various FSM elements and prevent their negative impact on the controlled object.

The novelty of the proposed approach is as follows:

– for the first time, the FSM faults that are not caused by radiation and cosmic rays, but by an electromagnetic pulse affecting the control device are considered;

– the number of faults is not limited to the state codes and the input and output vectors;

– the faults can be detected not only in the state register R but also in the input vector X, in the logic Φ of generating the next state code, in the logic Ψ of generating the output signals, and in the feedback circuit;

– the invalid transitions of FSMs and the transitions to invalid states are also detected;

– the proposed structural models not only detect FSM failures but also prevent their negative impact on the controlled object;

– structural models allow simultaneous detection of faults in all elements of the FSM;

– the proposed approach can be used to detect faults in both field programmable gate arrays (FPGAs) and application-specific integrated circuits (ASICs).

The goal of this work is to solve the structural problem of detecting multiple faults, which can be caused by an electromagnetic pulse, for various components of Mealy FSMs.

Related works

The problem of fault-tolerant computing is as old as the first computers. However, much greater requirements for the FSM protection from cosmic rays were posed by the space program in the early 1960s. The traditional solution to this problem is multiple duplication of the FSM architecture, with the triple modular redundancy method (TRM) being the most common method for protection against a single event upset (SEU) [6]. In general, the fault tolerance of a digital system can be provided by architecture redundancy, runtime increase, and data redundancy [7].

A lot of research has been conducted to improve the TMR method. In [8], a single error correction (SEC) code is used to implement the FSM, which allows the SEU in the logic of the next state and in the state register. The work in [9] compares architectures of fault-tolerant FSMs (TRM, SEU-ITRM, duplex, EEC, modified EEC, and IEC), which admit single errors when states are switched.

The use of FPGAs to design FSMs offers several advantages over ASICs: small size, low power consumption and low cost, short time to market, and possibility of reprogramming. However, FPGAs are more susceptible to SEUs caused by space particles than ASICs. Therefore, Xilinx has released special FPGAs of the Virtex family, which support the TMR method at the hardware level [10]. The work in [11] proposes that in systems on a chip (SoC), when a fault is detected in the FPGA, the FPGA generates an interrupt for the microcontroller, which triggers the procedure of partial reprogramming of the FPGA. A method for improving the TMR approach, which combines duplication with comparison (DWC) and concurrent error detection (CED) based on runtime redundancy, is proposed in [12].

Separate works use methods of encoding states in the synthesis of fault-tolerant FSMs. The work in [13] considers four methods of state encoding for fault-tolerant FSMs: binary, one-hot, Hamming with distance 2 (H2), and Hamming with distance 3 (H3); it compares the fault tolerance and resource utilization. The methods of state encoding (binary, one-hot, and H3) to eliminate SEU in the state register are investigated in [14]; it is recommended to manually set the logic for recovery from an invalid state.

Many methods based on state encoding have been developed to improve the TMR approach. The dual modular redundancy (DMR) method and the use of a parity bit in the FSM implementation in embedded memory blocks of FPGAs were presented in [15]. The work in [16] evaluates two methods of fault-tolerant FSM synthesis: duplication with self-check and TMR. Here, the following state coding methods are used: binary, one-hot and Gray. A method to improve the TMR is proposed in [17]; it uses Hamming code to implement the FSM in embedded FPGA memory blocks. The work in [18] improves the TMR method from Xilinx [10]. To do this, the system is represented as a set of FSMs, and the control points are introduced to detect the faulty domain. When a fault is detected, only the faulty domain is restored and the rest of the system continues to work. As a result, the system recovery time after the failure is reduced.

Some methods propose the introduction of additional states into the FSM. In [19], redundant equivalent states are added to the FSM to protect states with a high probability of failure. A synthesis method for fault-tolerant FSMs based on single error correction and double error detection (SEC-DED) code is proposed in [20]; it involves returning the FSM to the known safe state or to the reset state.

Recently, interest in the design of fault-tolerant FSMs has not weakened. In [21], three synthesis methods for fault-tolerant FSMs are investigated: TMR, H3, and safe synthesis. The work in [22] improves the fault tolerance of FSMs by selectively applying the TMR method according to the importance of the state. In [23], the quasi-delay-sensitive architecture of an FSM is compared with that of a TMR.

This analysis shows that almost all methods of fault-tolerant FSM synthesis are aimed at improving the TMR method to correct the SEU in the state register. Most of the methods assume that the primary inputs, the logic of generating the next state code, the logic of forming outputs, and the additional logic for detecting and correcting errors do not have failures.

On the other hand, the structural models of FSMs are very effective for improving performance and reducing area and power consumption when implementing FSMs on FPGAs [24]. Structural models for the fault detection of Moore FSMs are reviewed in [25, 26].

This paper presents structural models of the Mealy FSM for detecting multiple faults in various elements of the FSM and preventing their negative impact on the controlled object.

The demonstration example

As an example, consider the Mealy FSM, the state transition graph (STG) of which is shown in Fig. 3. Our FSM has 4 states, 3 inputs and 3 outputs. The vertices of the STG correspond to states S_0, ..., S_3, and the arcs of the STG correspond to the transitions of the FSM. The input vector that initiates this transition is written near each the STG arc, and the output vector that is formed at this transition is written through a slash ("/"). Here, the hyphen ("-") can take any bit value: 0 or 1.

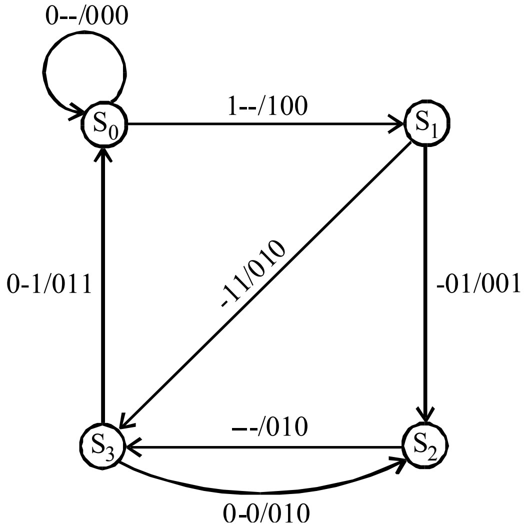

Fig. 3. The STG of the Mealy FSM

The valid transitions between states and the valid output vectors at each transition can be determined directly from the STG (Table 1). Valid input vectors in each state are determined by the developer based on the behavior of the control device. Let the valid input vectors for our FSM be determined using Table 2.

Table 1
Valid transitions and valid output vectors at each FSM transition

The present state	The next state	The output vector
S_0	S_0	000
	S_1	100
S_1	S_2	001
	S_3	010
S_2	S_3	010
S_3	S_0	011
	S_2	010

Table 2
Valid input vectors of the FSM in each state

The present state	The output vectors
S_0	---
S_1	001 011 101 111
S_2	---
S_3	000 001 010 011

Valid input and output vectors for the entire FSM are also defined by the developer based on the behavior of the control device. For our FSM, let the valid input and output vectors be determined using Table 3.

Table 3
Valid input and output vectors for the entire FSM

The input vectors	The output vectors
000	000
001	001
010	010
011	011
100	100

Structural models of Mealy FSMs for fault detection

The structural models of Mealy FSMs for fault detection and prevention of the negative impact of the faults on the controlled object are shown in Fig. 4 and Fig. 5.

The structures in Fig. 4 and Fig. 5 are formed by introducing the following combinational circuits into the traditional structure shown in Fig. 1: TVI, VI, TVO, VTO, VO, VS, VNS, and VT. Each additional combinational circuit generates a diagnostic signal of the same name on the output, the value of 1 indicating the absence of a fault.

To prevent a negative impact of failure on the operation of the FSM, the input CE (clock enable) of the state register R is controlled by the diagnostic signals tvi, vi, tvo, vto, vo vs, vns, and vt. As a result, if a failure occurs, the FSM does not switch to the next state and remains in the current state until the fault is resolved. In addition, to prevent the negative impact of failure on the controlled object, the output register R_O is added to the considered structures. The input CE of the register R_O is also controlled by the diagnostic signals tvi, vi, tvo, vto, vo vs, vns, and vt. In case of a failure, the register R_O will not be switched, and the outputs of the FSM will remain the last correct value of the output vector Y. Note that the structure VNS has no output register R_O, since an invalid code of the next state does not change the output vector.

To determine the corresponding fault, the necessary values are supplied to the inputs of each additional combination circuit.

The structure TVI (total valid inputs) defines valid input vectors for the entire FSM. For this purpose, the input vector X arrives at the input of the combinational circuit TVI. For our example, the valid input vectors are shown in Table 3.

Structure VI (valid inputs) defines the valid input vectors in each state. Therefore, the input of the combinational circuit VI receives the code of this state and the input vector X. For our example, the functioning of combinational circuit VI is defined in Table 2.

Structure TVO (total void outputs) defines valid output vectors for the entire automaton. For this purpose, the input of the combinational circuit TVO receives the output values of the logic Ψ. For our example, the valid output vectors for the entire automaton are presented in Table 3.

Structure VTO (void transition outputs) defines valid output vectors at each transition of the FSM. The input of the combinational circuit VTO receives the code of the present state and the code of the next state, as well as the values of the output signals generated by the logic Ψ. For our example, the functioning of the combinational circuit VTO is presented in Table 1.

Structure VO (void outputs) defines the valid output vectors in each state. The input of the combinational circuit VO receives the code of the present state and the values of the outputs generated by the logic Ψ. In our example, the functioning of the combinational circuit VO is defined in Table 1 (first and third columns).

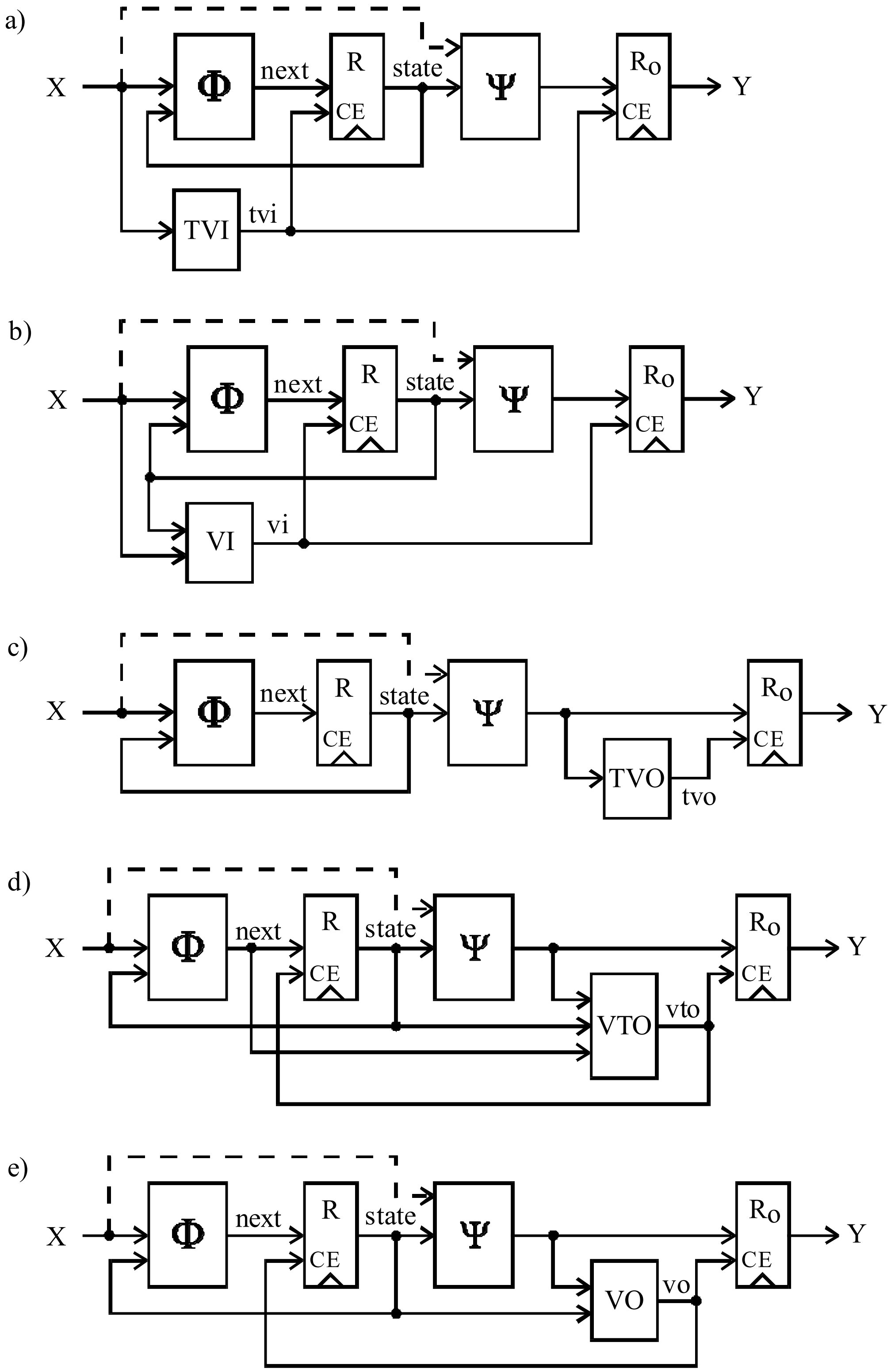

Fig. 4. The structural models of Mealy FSMs for detecting:
a – valid input vectors for the whole FSM (the structure TVI);
b – valid input vectors in each state (the structure VI);
c – valid output vectors for the whole FSM (the structure TVO);
d – valid output vectors at each transition (the structure VTO);
e – valid output vectors in each state (the structure VO)

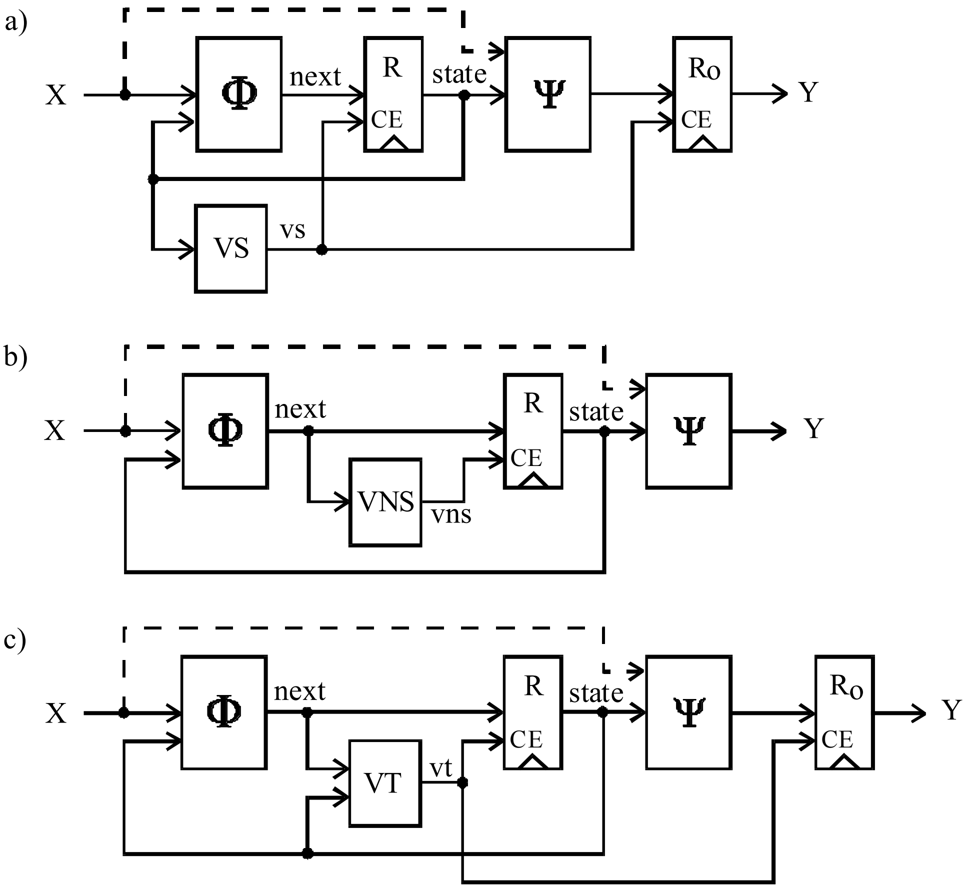

Fig. 5. The structural models of Mealy FSMs for detecting:
a – a valid code of this state (the structure VS); b – a valid code of the next state (the structure VNS);
c – valid transitions between states (the structure VT)

The structure VS (valid states) defines the valid present state code. The input of the combinational circuit VS receives the present state code. Similarly, the structure VNS (valid next states) defines the valid code of the next state. The next states code generated by logic Φ arrives to the input of the combinational circuit VNS. Note that the state codes may be defined either by the user or automatically by the design tool. For our example, the valid state codes are the codes of states S_0, ..., S_4. Note that it is possible to describe combinational circuits VS and VNS in HDL without specifying particular state codes. The structure VT (valid transitions) defines valid transitions between states.

The input of the VT combinational circuit receives the code of this state and the code of the next state. In our example, the functioning of the combinational circuit VT is defined in Table 1 (first and second columns).

Table 4 summarizes the possible causes of the faults detected by the structural models in Fig. 4 and Fig. 5, where Φ is a failure in the logic Φ; X is the invalid input vector; *feedback* is a failure in the feedback circuit; Ψ is a failure in the logic Ψ; R is a failure in the state register R; X→ Ψ is a failure in the circuit between the input of the FSM and the input of the combinational circuit Ψ.

Table 4
Possible causes of faults detected by FSM structural models

Causes of a failure	TVI	VI	TVO	VTO	VO	VS	VNS	VT
Φ				*			*	*
X	*	*	*	*	*		*	*
feedback		*	*	*	*	*	*	*
Ψ			*	*	*			
R		*	*	*	*	*	*	*
X→ Ψ			*	*	*			

Only the structure TVI can clearly identify the specific fault: invalid input vectors for the entire FSM, which may not exist at all, since most FSMs have no input vector constraints. The structure VS allows us to find a fault in the state register R or in the feedback circuit. Structure VI finds a fault in input vector X, state register R, or feedback circuit. The structures TVO and VO determine the same number of faults (X, feedback, Ψ, R, and X→ Ψ). The structures VNS and VT also determine the same number of faults (Φ, X, feedback, and R). The structure VTO allows us to find all the considered FSM faults. In addition, the structure VT allows us to find transitions of the FSM into illegal states, and invalid transitions into legal states (see Fig. 2).

Note that all the FSM structural models considered are not targeted for implementation on a particular electronic component: each structural model can be implemented on both ASIC and FPGA.

For more effective detection of the faulty element of the FSM, the structure shown in Fig. 4 and Fig. 5 can be combined into one structure. Note that structure TVI is covered by structure VI, i.e., if a fault is detected using structure TVI, the fault will certainly be detected by structure VI. Similarly, structure TVO is covered by structure VO, which in turn is covered by structure VT. In addition, the structures VS and VNS are covered by the structure VT. Therefore, when all the considered structures are combined, the structures TVI, TVO, VO, VS, and NVS can be omitted. Fig. 6 shows a combined structure VITTO that combines the VI, VT, and VTO structures.

For the most accurate detection of the faulty element of the FSM, all structures in Fig. 4 and Fig. 5 can be combined into one structure (Fig. 7).

Diagnostic signals tvi, vi, tvo, vto, vo, vs, vns, and vt generated by the FSM structural models in Fig. 4 and Fig. 5, can be output to the external outputs of the FSM to detect the location of the fault in the ASIC. If necessary, the diagnostic signals can be combined to indicate an error (Fig. 8), for example, to reconfigure the FPGA.

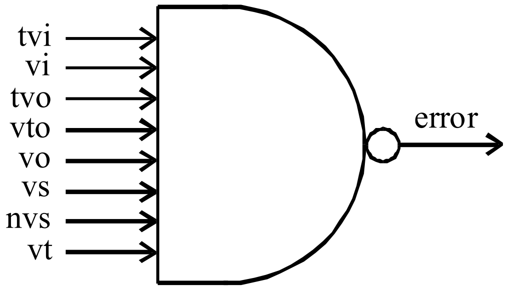

Fig. 8. Circuit for error detection of Mealy FSMs

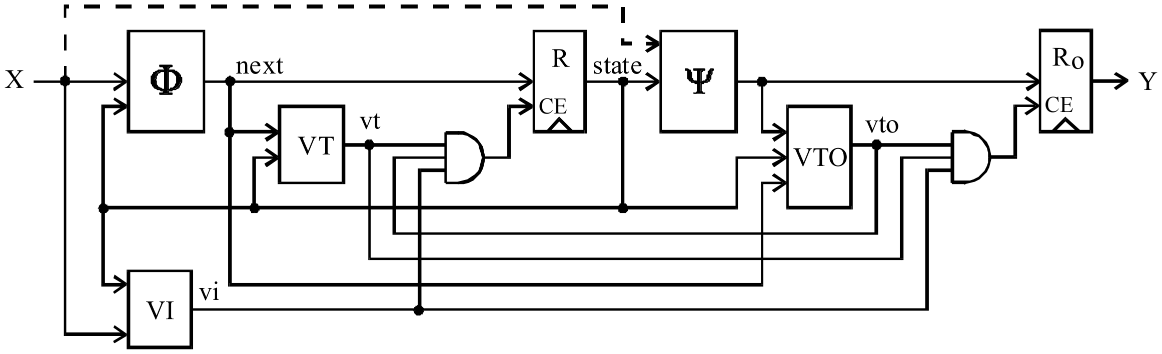

Fig. 6. The combined structure VITTO for fault detection of the Mealy FSM

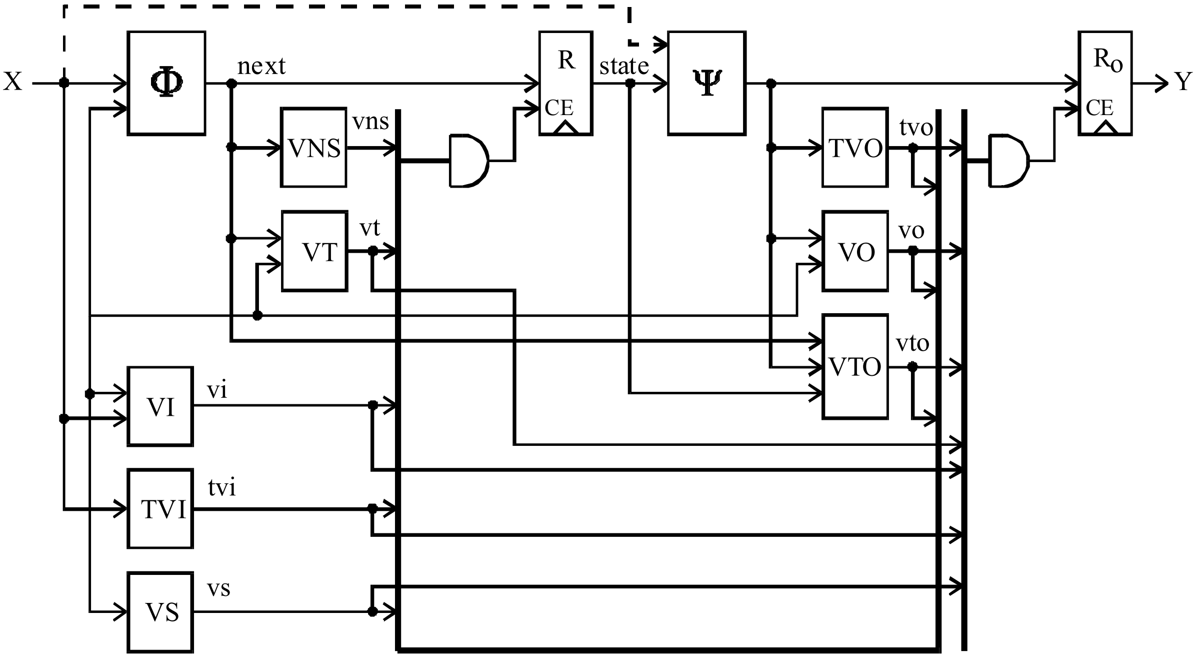

Fig. 7. The combined structural model of the Mealy FSM for the most accurate detection of faulty elements

Experimental results and discussion

The experiments were conducted to evaluate the area and performance of the presented structural models. To this end considered structural models for our example of the Mealy FSM were described in Verilog HDL in the style with three processes [5]. As a result, the following FSM designs were created for our example:

– based_FSM – the traditional (basic) structural model of the Mealy FSM (see Fig. 1);

– TVI_FSM – FSM with check of the valid input vectors for the whole FSM (see Fig. 4, a);

– VI_ FSM – FSM with a check of the valid input vectors in each state (see Fig. 4, b);

– TVO_FSM – FSM with check of the valid output vectors for the whole FSM (see Fig. 4, c);

– VTO_FSM – FSM with check of the valid output vectors at each transition (see Fig. 4, d);

– VO_FSM – FSM with check of the valid output vectors in each state (see Fig. 4, e);

– VS_FSM – FSM with check of the valid code of this state (see Fig. 5, a);

– VNS_FSM – FSM with check of the valid code of the next state (see Fig. 5, b)

– VT_FSM – FSM with check of the valid transitions between states (see Fig. 5, c);

– VITTO_FSM – FSM with the union of structures VI, VT, and VTO (see Fig. 6);

– V_ALL_FSM – FSM with the union of all considered structures (see Fig. 7).

Appendix A list the descriptions in the Verilog HDL of the proposed structural models of FSMs for our example.

To estimate the area and performance, the FSM designs were synthesized using Quartus Prime tool (version 22.4) from Intel on the Cyclone 10 LP FPGA for the case of one-hot coding of states. The experimental results are shown in Table 5, where L is the number of used LUTs (look-up table) or area; F is the maximum operating frequency (in megahertz) or speed; L_b and F_b are similar parameters for the basic structural model (see Fig. 1); L/L_b and F/F_b are the relations of the corresponding parameters.

Table 5 shows that for our example of the Mealy FSM the structures TVI, VI, TVO, and VNS do not increase the area, and the structure VS even reduces the FSM area compared to the basic structure. The structures VO and VT increase the area of the FSM by 20%, and the structure VTO increases the area by 30%. The structures VITTO and V_ALL have the highest area overhead (40% and 60%, respectively).

Table 5 also shows that the use of the proposed FSM structures increases the performance of the basic structure from 1% (the structure VTO) to 24% (the structure VO). The exceptions are structures VNS (the performance is the same as the base structure), VITTO (the performance decreases by 6%), and V_ALL (the performance decreases by 7%). The performance increase of the proposed structures is explained by the addition of the output vector R_O to the basic structure, but this leads to a delay of the output signals by one clock cycle.

Table 5

Results of experimental studies of structural models for our example of the Mealy FSM in the case of the one-hot coding of states

Design	L	L/L_b	F	F/F_b
based_FSM	10	1	310	1
TVI_FSM[1]	10	1	363	1.17
VI_FSM[1]	10	1	330	1.06
TVO_FSM[1]	10	1	371	1.20
VTO_FSM[1]	13	1.30	312	1.01
VO_FSM[1]	12	1.20	383	1.24
VS_FSM[1]	9	0.90	358	1.15
VNS_FSM	10	1	310	1
VT_FSM[1]	12	1.20	371	1.20
VITTO_FSM[1]	14	1.40	290	0.94
V_ALL_FSM[1]	16	1.60	287	0.93

[1] the delay per one clock cycle

The consideration of one example of the Mealy FSM does not allow us to fully evaluate the proposed structural models; therefore, the structural models have been investigated using FSM benchmarks from MCNC [27]. The results of these studies are shown in Tables 6 - 8.

Note that for the proposed structural models, the area overhead depends on the state encoding. Table 6 shows the results of comparing the area of the proposed structural models with the basic FSM structure for one-hot coding and Table 7 for binary coding. Table 8 shows similar results of the comparison for combined structures VITTO and V_ALL, where *i* is the number of FSM inputs; *o* is the number of FSM outputs; *s* is the number of FSM states; C_b is the area (number of LUT) of the basic FSM structure; C is the FSM area when using one of the proposed structural models; C_b/C is the ratio of corresponding parameters; and *mid* is the arithmetic mean value.

Tables 6 and 7 show that the area overhead for the proposed structural models average ranges from 3% (the structure TVO) to 23% (the structure VTO) in the case of one-hot coding and from 2% (the structures VS and VNS) to 8% (the structure VTO) in the case of binary coding. The combined structural models are the costliest in terms of area (Table 8). For the structure VITTO, the average area increase is 57% when using one-hot code and 20% when using binary code. Similar values for the structure V_ALL are 97% and 38%, respectively.

Table 6

Area overhead comparison of the proposed structural models for one-hot coding

FSM	C_b	TVI		VI		TVO		VTO		VO		VS, VNS		VT	
		C	C/C_b	C	C/C_b	C	C/C_b	C	C/C_b	C	C/C_b	C	C/C_b	C	C/C_b
bbase	41	43	1.05	49	1.20	43	1.05	54	1.32	49	1.20	46	1.12	52	1.27
Cse	83	85	1.02	91	1.10	85	1.02	96	1.16	91	1.10	88	1.06	94	1.13
Ex1	106	109	1.03	115	1.08	112	1.06	124	1.17	118	1.11	112	1.06	118	1.11
Ex2	41	42	1.02	48	1.17	42	1.02	54	1.32	48	1.17	47	1.15	54	1.32
Ex3	26	27	1.04	30	1.15	27	1.04	33	1.27	30	1.15	29	1.12	33	1.27
Ex5	22	23	1.05	26	1.18	23	1.05	29	1.32	26	1.18	25	1.14	28	1.27
Keyb	69	71	1.03	78	1.13	70	1.01	82	1.19	76	1.10	75	1.09	82	1.19
Planet	134	136	1.01	152	1.13	140	1.04	171	1.28	156	1.16	150	1.12	165	1.23
Pma	115	118	1.03	126	1.10	118	1.03	134	1.17	126	1.10	123	1.07	131	1.14
S208	67	71	1.06	77	1.15	68	1.01	80	1.19	74	1.10	73	1.09	79	1.18
S298	647	648	1.00	720	1.11	649	1.00	793	1.23	727	1.12	719	1.11	791	1.22
S386	46	48	1.04	53	1.15	48	1.04	57	1.24	53	1.15	50	1.09	55	1.20
S420	33	39	1.18	45	1.36	34	1.03	46	1.39	40	1.12	39	1.18	45	1.36
S820	88	94	1.07	102	1.16	94	1.07	111	1.26	103	1.17	96	1.09	105	1.19
S1488	202	205	1.01	221	1.09	208	1.03	239	1.18	224	1.11	218	1.08	233	1.15
Sand	178	182	1.02	192	1.08	181	1.02	202	1.13	192	1.08	189	1.06	199	1.12
styr	158	161	1.02	171	1.08	161	1.02	181	1.15	171	1.08	168	1.06	178	1.13
mid			1.04		1.14		1.03		1.23		1.13		1.10		1.20

Table 7

Area overhead comparison of the proposed structural models for binary coding

FSM	C_b	TVI		VI		TVO		VTO		VO		VS, VNS		VT	
		C	C/C_b	C	C/C_b	C	C/C_b	C	C/C_b	C	C/C_b	C	C/C_b	C	C/C_b
bbase	41	43	1.05	45	1.10	43	1.05	46	1.12	45	1.10	42	1.02	44	1.07
Cse	83	85	1.02	87	1.05	85	1.02	88	1.06	87	1.05	84	1.01	86	1.04
Ex1	106	109	1.03	111	1.05	112	1.06	116	1.09	114	1.08	108	1.02	109	1.03
Ex2	41	42	1.02	43	1.05	42	1.02	45	1.10	43	1.05	43	1.05	44	1.07
Ex3	26	27	1.04	28	1.08	27	1.04	29	1.12	28	1.08	26	1.00	29	1.12
Ex5	22	23	1.05	24	1.09	23	1.05	25	1.14	24	1.09	23	1.05	25	1.14
Keyb	69	71	1.03	73	1.06	70	1.01	73	1.06	71	1.03	71	1.03	72	1.04
Planet	134	136	1.01	138	1.03	140	1.04	144	1.07	142	1.06	136	1.01	138	1.03
Pma	115	118	1.03	119	1.03	118	1.03	121	1.05	119	1.03	117	1.02	118	1.03
S208	67	71	1.06	72	1.07	68	1.01	71	1.06	69	1.03	69	1.03	70	1.04
S298	647	648	1.00	651	1.01	649	1.00	654	1.01	652	1.01	650	1.00	652	1.01
S386	46	48	1.04	50	1.09	48	1.04	51	1.11	50	1.09	47	1.02	49	1.07
S420	33	39	1.18	41	1.24	34	1.03	37	1.12	35	1.06	35	1.06	36	1.09
S820	88	94	1.07	96	1.09	94	1.07	98	1.11	96	1.10	90	1.02	91	1.03
S1488	202	205	1.01	207	1.02	208	1.03	212	1.05	210	1.04	204	1.01	206	1.02
Sand	178	182	1.02	183	1.03	181	1.02	184	1.03	183	1.03	180	1.01	181	1.02
styr	158	161	1.02	163	1.03	161	1.02	165	1.04	163	1.03	160	1.01	161	1.02
mid			1.04		1.07		1.03		1.08		1.06		1.02		1.05

Table 8

Area overhead comparison of combined structural models

FSM	i	o	s	C_b	One-hot				Binary			
					VITTO		V_ALL		VITTO		V_ALL	
					C	C/C_b	C	C/C_b	C	C/C_b	C	C/C_b
bbase	7	7	16	41	72	1.76	95	2.32	53	1.29	63	1.54
Cse	7	7	16	83	114	1.37	137	1.65	95	1.14	105	1.27
Ex1	9	19	18	106	145	1.37	178	1.68	124	1.17	144	1.36
Ex2	2	2	19	41	73	1.78	94	2.29	51	1.24	58	1.41
Ex3	2	2	10	26	43	1.65	55	2.12	34	1.31	40	1.54
Ex5	2	2	9	22	38	1.73	49	2.23	30	1.36	36	1.64
Keyb	7	2	19	69	103	1.49	126	1.83	80	1.16	89	1.29
Planet	7	19	48	134	222	1.66	284	2.12	153	1.14	182	1.36
Pma	8	8	24	115	160	1.39	192	1.67	129	1.12	142	1.23
S208	11	2	18	67	101	1.51	124	1.85	80	1.19	90	1.34
S298	3	6	218	647	1013	1.57	1236	1.91	663	1.02	676	1.04
S386	7	7	13	46	72	1.57	92	2.00	58	1.26	68	1.48
S420	19	2	18	33	69	2.09	94	2.85	48	1.45	61	1.85
S820	18	19	25	88	141	1.60	185	2.10	109	1.24	132	1.50
S1488	8	19	48	202	290	1.44	354	1.75	221	1.09	242	1.20
Sand	11	9	32	178	237	1.33	279	1.57	193	1.08	208	1.17
styr	9	10	30	158	214	1.35	253	1.60	173	1.09	188	1.19
mid						1.57		1.97		1.20		1.38

Conclusions

This paper presents new structural models of Mealy FSMs that detect the following faults: invalid input vectors for the entire FSM and in each state; invalid code for the present and next state of the FSM; invalid transitions between the FSM states; and invalid output vectors for the entire FSM, in each state, and at each transition of the FSM. In addition, the combined structural models allow simultaneous detection of all of the faults listed above.

The considered structural models allow detection of multiple failures in the logic of the next state code formation, in the logic of the output signal value formation, in the state register, and in the feedback circuit. These structural models of FSMs not only detect faults, but also prevent their negative impact on the controlled object. For the presented structural models of FSMs, the area overhead averages 3-23% when using one-hot coding, and 2-8% when using binary coding.

Future research will focus on developing structural models for correcting FSM failures.

Acknowledgements

This study was supported by a grant WZ/WI-III/5/2023 from Bialystok University of Technology and founded from the resources for research by Ministry of Science and Higher Education.

Appendix A

```
// description of the based_FSM design
module based_FSM(
    input clk, reset,
    input [2:0] x,
    output reg [2:0] y);// description of ports

    // declaration of the state variables
    reg [1:0] state, next;

    // declaration of the state
    localparam [1:0] s0=0, s1=1, s2=2, s3=3;

    // description of the state register
    always @(posedge clk, negedge reset)
        if (~reset)         state <= s0;
        else                state <= next;

        always @(*)       // description of transitions
                case (state)
                        s0: casex (x)
                                3'b0??:  next = s0;
                                3'b1??:  next = s1;
                                default: next = s0;
                            endcase
                        s1: casex (x)
                                3'b?01:  next = s2;
                                3'b?11:  next = s3;
                default:  next = s1;
              endcase
            s2:               next = s3;
            s3: casex (x)
                3'b0?0:   next = s2;
                3'b0?1:   next = s0;
                default:  next = s3;
              endcase
            default:          next = s0;
        endcase

    always @(*)     // description of outputs
        case (state)
            s0: casex (x)
                3'b0??:   y = 3'b000;
                3'b1??:   y = 3'b100;
                default:  y = 3'b000;
              endcase
            s1: casex (x)
                3'b?01:   y = 3'b001;
                3'b?11:   y = 3'b010;
                default:  y = 3'b000;
              endcase
            s2:               y = 3'b010;
            s3: casex (x)
                3'b0?0:   y = 3'b010;
                3'b0?1:   y = 3'b011;
                default:  y = 3'b000;
              endcase
            default:          y = 3'b000;
        endcase
endmodule
```

// **description of the TVI_FSM design**

```
module TVI_FSM(       // description of the ports
    input clk, reset,
    input [2:0] x,
    output reg [2:0] y_out);

    …// declaration of the state variables
    …// declaration of the state

    reg [2:0] y;      // output of the logic Ψ
    reg tvi;          // the signal tvi

    // status register description
    always @(posedge clk, negedge reset)
        if (~reset)       state <= s0;
        else if (tvi)     state <= next;
        else              state <= state;

    …// description of transitions
    …// description of outputs

    always @(*)       // description of the TVI logic
        case (x)
            3'b101:  tvi = 1'b0;
            3'b110:  tvi = 1'b0;
            3'b111:  tvi = 1'b0;
            default: tvi = 1'b1;
        endcase

    // description of the register Ro
    always @(posedge clk, negedge reset)
        if (~reset)        y_out <= 0;
        else if(tvi)       y_out <= y;
        else               y_out <= y_out;
endmodule
```

// **description of the VI_FSM design**

```
module VI_FSM(…);      // description of the ports
    …
    always @(*)        // description of the VI logic
        case (state)
            s0: vi = 1'b1;
            s1: case(x)
                3'b001,
                3'b011,
                3'b101,
                3'b111: vi = 1'b1;
                default: vi = 1'b0;
              endcase
            s2:                 vi = 1'b1;
            s3: case(x)
                3'b000,
                3'b001,
                3'b010,
                3'b011: vi = 1'b1;
                default: vi = 1'b0;
              endcase
            default:   vi = 1'b0;
        endcase
    …
endmodule
```

// **description of the TVO_FSM design**

```
module TVO_FSM(…);   // description of the ports
    …
    always @(*)// description of the TVO logic
        case (y)
            3'b000:  tvo = 1'b1;
            3'b001:  tvo = 1'b1;
            3'b010:  tvo = 1'b1;
            3'b011:  tvo = 1'b1;
            3'b100:  tvo = 1'b1;
            default: tvo = 1'b0;
        endcase
    …
endmodule
```

// **description of the VTO_FSM design**

```
module VTO_FSM(…);   // description of the ports
  …
  always @(*)// description of the VTO logic
    case (state)
      s0:case(next)
        s0:        if (y == 3'b000)  vto = 1'b1;
```

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

```
                else              vto = 1'b0;
        s1:     if (y == 3'b100)  vto = 1'b1;
                else              vto = 1'b0;
                default:          vto = 1'b0;
        endcase
     s1:case(next)
        s2:     if (y == 3'b001)  vto = 1'b1;
                else              vto = 1'b0;
        s3:     if (y == 3'b010)  vto = 1'b1;
                else              vto = 1'b0;
                default:          vto = 1'b0;
        endcase
     s2:case(next)
        s3:     if (y == 3'b010)  vto = 1'b1;
                else              vto = 1'b0;
                default:          vto = 1'b0;
        endcase
     s3:case(next)
        s0:     if (y == 3'b011)  vto = 1'b1;
                else              vto = 1'b0;
        s2:     if (y == 3'b010)  vto = 1'b1;
                else              vto = 1'b0;
                default:   vto = 1'b0;
        endcase
      default: vto = 1'b0;
   endcase
        …
endmodule
```

```
// description of the VO_FSM design
module VO_FSM(…);       // description of the ports
        …
        always @(*)     // description of the VO logic
                case (state)
                        s0:case(y)
                                3'b000: vo = 1'b1;
                                3'b100: vo = 1'b1;
                                default: vo = 1'b0;
                        endcase
                        s1:case(y)
                                3'b001: vo = 1'b1;
                                3'b010: vo = 1'b1;
                                default: vo = 1'b0;
                        endcase
                        s2:case(y)
                                3'b010: vo = 1'b1;
                                default: vo = 1'b0;
                        endcase
                        s3:case(y)
                                3'b011: vo = 1'b1;
                                3'b010: vo = 1'b1;
                                default: vo = 1'b0;
                        endcase
                        default: vo = 1'b0;
                endcase
        …
endmodule
```

```
// description of the VS_FSM design
module VS_FSM(…);       // description of the ports
        …
        always @(*)     // description of the VS logic
                case (state)
                        s0:             vs = 1'b1;
                        s1:             vs = 1'b1;
                        s2:             vs = 1'b1;
                        s3:             vs = 1'b1;
                        default:  vs = 1'b0;
                endcase
        …
endmodule
```

```
// description of the VNS_FSM design
module VNS_FSM(…);      // description of the ports
        …
        always @(*)// description of the VNS logic
                case (next)
                        s0: vns = 1'b1;
                        s1: vns = 1'b1;
                        s2: vns = 1'b1;
                        s3: vns = 1'b1;
                        default:  vns = 1'b0;
                endcase
        …
endmodule
```

```
// description of the VT_FSM design
module VT_FSM(…);       // description of the ports
        …
        always @(*)     // description of the VT logic
                case (state)
                        s0:case(next)
                                s0:     vt = 1'b1;
                                s1:     vt = 1'b1;
                                default: vt = 1'b0;
                        endcase
                        s1:case(next)
                                s2:     vt = 1'b1;
                                s3:     vt = 1'b1;
                                default: vt = 1'b0;
                        endcase
                        s2:case(next)
                                s3:     vt = 1'b1;
                                default: vt = 1'b0;
                        endcase
                        s3:case(next)
                                s0:     vt = 1'b1;
                                s2:     vt = 1'b1;
                                default: vt = 1'b0;
                        endcase
                        default:  vt = 1'b0;
                endcase
        …
endmodule
```

References

1. Fesenko, H. V., & Kharchenko, V. S. Determination of an optimal route for flight over of specified points of a potentially dangerous object territory by UAV fleet. *Radioelectronic and Computer Systems*, 2019, no. 3, pp. 63-72. DOI: 10.32620/reks.2019.3.07.

2. Naso, D., Pohudina, O., Pohudin, A., Yashin, S., & Bartolo, R. Autonomous flight insurance method of unmanned aerial vehicles Parot Mambo using semantic segmentation data. *Radioelectronic and Computer Systems*, 2023, no. 1, pp. 147-154. DOI: 10.32620/reks.2023.1.12.

3. Min, S. H., Jung, H., Kwon, O., Sattorov, M., Kim, S., Park, S. H., Hong, D., Kim, S., Park, C., Hong, B. H., Cho, I., Ma, S., Kim, M., Yoo, Y. J., Park, S. Y., & Park, G. S. Analysis of electromagnetic pulse effects under high-power microwave sources. *IEEE Access*, 2021, no. 9, pp. 136775-136791. DOI: 10.1109/ACCESS.2021.3117395.

4. Solov’ev, V. V. ASMD–FSMD technique in designing signal processing devices on field programmable gate arrays. *Journal of Communications Technology and Electronics*, 2021, vol. 66, no. 12, pp. 1336-1345. DOI: 10.1134/S1064226921120184.

5. Salauyou, V., & Zabrocki, Ł. Coding Techniques in Verilog for Finite State Machine Designs in FPGA. *IFIP International Conference on Computer Information Systems and Industrial Management (CISIM)*, Belgrade, Serbia, 2019, pp. 493-505. DOI: 10.1007/978-3-030-28957-7_41.

6. Lyons, R. E., & Vanderkulk, W. The use of triple-modular redundancy to improve computer reliability. *IBM journal of research and development*, 1962, vol. 6, no. 2, pp. 200-209. DOI: 10.1147/rd.62.0200.

7. Aviziens, A. Fault-tolerant systems. *IEEE transactions on computers*, 1976, vol. 100, no. 12, pp. 1304-1312. DOI: 10.1109/TC.1976.1674598.

8. Rochet, R., Leveugle, R., & Saucier, G. Analysis and comparison of fault tolerant FSM architecture based on SEC codes. *IEEE International Workshop on Defect and Fault Tolerance in VLSI Systems*, Venice, Italy, 1993, pp. 9-16. DOI**:** 10.1109/DFTVS.1993.595604.

9. Niranjan, S., & Frenzel, J. F. A comparison of fault-tolerant state machine architectures for space-borne electronics. *IEEE Transactions on Reliability*, 1996, vol. 45, no. 1, pp. 109-113. DOI**:** 10.1109/24.488925.

10. Carmichael, C. Triple module redundancy design techniques for Virtex FPGAs. *Xilinx Application Note XAPP197, v.1.0.1,* 2006. 37 p. Available at: https://www.amd.com/en/search/site-search.html#q=XAPP197. (accessed 11 August 2023).

11. Pontarelli, S., Cardarilli, G. C., Malvoni, A., Ottavi, M., Re, M., & Salsano, A. System-on-chip oriented fault-tolerant sequential systems implementation methodology. *IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems*, Los Alamitos, USA, 2001, pp. 455-460. DOI: 10.1109/DFTVS.2001.966799.

12. Lima, F., Carro, L., & Reis, R. Designing fault tolerant systems into SRAM-based FPGAs. *40th annual Design Automation Conference (DAC)*, Anaheim, USA, 2003, pp. 650-655. DOI: 10.1145/775832.775997.

13. Burke, G. R., & Taft, S. *Fault tolerant state machines*, 2004. 10 p. Available at: https://dataverse.jpl.nasa.gov/api/access/datafile/9953?gbrecs=true. (accessed 11 August 2023).

14. Berg, M. A Simplified Approach to Fault Tolerant State Machine Design for Single Event Upsets. *Mentor Graphics Users’ Group User2User Conference*. 2004.

15. Tiwari, A., & Tomko, K. A. Enhanced reliability of finite-state machines in FPGA through efficient fault detection and correction. *IEEE Transactions on Reliability*, 2005, vol. 54, no. 3, pp. 459-467. DOI: 10.1109/TR.2005.853438.

16. Cassel, M., & Lima, F. Evaluating one-hot encoding finite state machines for SEU reliability in SRAM-based FPGAs. *12th IEEE International On-Line Testing Symposium (IOLTS)*. Lake Como, Italy, 2006, pp. 1-6. DOI: 10.1109/IOLTS.2006.32.

17. Frigerio, L., & Salice, F. RAM-based fault tolerant state machines for FPGAs. *22nd IEEE International Symposium on Defect and Fault-Tolerance in VLSI Systems (DFT),* Rome, Italy, 2007, pp. 312-320. DOI: 10.1109/DFT.2007.33.

18. Azambuja, J. R., Sousa, F., Rosa, L., & Kastensmidt, F. L. Evaluating large grain TMR and selective partial reconfiguration for soft error mitigation in SRAM-based FPGAs. *15th IEEE International On-Line Testing Symposium*, Lisbon, Portugal, 2009, pp. 101-106. DOI: 10.1109/IOLTS.2009.5195990.

19. El-Maleh, A. H., & Al-Qahtani, A. S. A finite state machine based fault tolerance technique for sequential circuits. *Microelectronics Reliability*, 2014, vol. 54, no. 3, pp. 654-661. DOI: 10.1016/j.microrel.2013.10.022.

20. Sooraj, S., Manasy, M., & Bhakthavatchalu, R. Fault tolerant FSM on FPGA using SEC-DED code algorithm. *International Conference on Technological Advancements in Power and Energy (TAP Energy)*, Kollam, India, 2017, pp. 1-6. DOI: 10.1109/TAPENERGY.2017.8397309.

21. Nidhin, T. S., Bhattacharyya, A., Behera, R. P., Jayanthi, T., & Velusamy, K. Verification of fault tolerant techniques in finite state machines using simulation based fault injection targeted at FPGAs for SEU mitigation. *4th International Conference on Electronics and Communication Systems (ICECS)*, Coimbatore, India, 2017, pp. 153-157. DOI: 10.1109/ECS.2017.8067859.

22. Choi, S., Park, J., & Yoo, H. Area-Efficient Fault Tolerant Design for Finite State Machines. *International Conference on Electronics, Information, and Communication (ICEIC)*, Barcelona, Spain, 2020, pp. 1-2. DOI: 10.1109/ICEIC49074.2020.9051122.

23. Verducci, O., Oliveira, D. L., & Batista, G. Fault-Tolerant Finite State Machine Quasi Delay Insensitive in Commercial FPGA Devices. *13th Latin America Symposium on Circuits and System (LASCAS)*, Santiago, Chile, 2022, pp. 1-4. DOI: 10.1109/LASCAS53948.2022.9789092.

24. Klimowicz, A. S., & Solov’ev, V. V. Structural models of finite-state machines for their implementation on programmable logic devices and systems on chip. *Journal of Computer and Systems Sciences International*, 2015, vol. 54, no. 2, pp. 230-242. DOI: 10.1134/S1064230715010074.

25. Salauyou, V. Structural models for fault detection of Moore finite state machines. *International Conference on Dependability and Complex Systems (DepCoS)*, Springer, Cham, 2023, pp. 214-223. DOI: 10.1007/978-3-031-37719-8.

26. Salauyou, V. Fault Detection of Moore Finite State Machines by Structural Models. *International Conference on Computer Information Systems and Industrial Management (CISIM)*, Springer, Cham, 2023, pp. 1-16. DOI: 10.1007/978-3-031-42823-4_29.

27. Yang, S. *Logic synthesis and optimization benchmarks user guide: version 3.0.* Research Triangle Park, NC, USA: Microelectronics Center of North Carolina (MCNC), 1991. 45 p. Available at: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=4a86519e41bb8dbaa8d2c9ba434030f48de85ce7. (accessed 11 August 2023).

Received 11.08.2023, Accepted 20.09.2023

СТРУКТУРНІ МОДЕЛІ КІНЦЕВОГО АВТОМАТА МІЛІ ДЛЯ ВИЯВЛЕННЯ НЕСПРАВНОСТЕЙ У СИСТЕМАХ КЕРУВАННЯ

Валерій Соловйов

Об'єктом дослідження є система керування безпілотними літальними апаратами (БПЛА), математичною моделлю якої є скінченний автомат (FSM). Метою роботи є розробка структурних моделей автоматів, які дозволяють (1) виявляти множинні несправності елементів автоматів, спричинені електромагнітним імпульсом або лазерним променем, та (2) запобігати негативному впливу на об'єкт керування. Завдання, які необхідно вирішити, полягають у наступному розробка структурних моделей FSM для виявлення недопустимого вхідного вектора X для всієї FSM і в кожному стані, виявлення недопустимого вихідного вектора Y для всієї FSM, на кожному переході і в кожному стані, недопустимого коду поточного стану, недопустимого коду наступного стану, недопустимого переходу між станами; визначення можливих причин виникнення несправностей, якими можуть бути збій в логіці Φ формування коду наступного стану, невірний вхідний вектор X, збій в ланцюзі зворотного зв'язку, збій в логіці Ψ формування вихідного вектора, збій в регістрі станів R, збій в проводі між входом автомата і входом логіки Ψ; розробка комбінованої структурної моделі для виявлення всіх перелічених несправностей з мінімальною кількістю додаткових комбінаційних схем, а також структурної моделі, яка об'єднує всі додаткові комбінаційні схеми. Використані методи: теорія скінченних автоматів, структурні моделі автоматів, методи кодування станів автоматів, методи представлення автоматів, мова опису апаратури Verilog. Отримано такі результати: (1) розроблено структурні моделі Mealy FSM для виявлення всіх вищезгаданих несправностей, (2) розроблено комбіновані структурні моделі FSM, (3) визначено можливі причини несправностей, які виявляються кожною структурною моделлю FSM. Експериментальні дослідження показали, що для представлених структурних моделей FSM накладні витрати складають в середньому 3-23% при one-hot кодуванні станів FSM та 2-8% при бінарному кодуванні станів FSM. Висновки. Наукова новизна отриманих результатів полягає в тому, що вперше розглянуто несправності FSM, які викликані не радіацією та космічними променями, а електромагнітним імпульсом, що впливає на керуючий пристрій; кількість несправностей не обмежена як для кодів станів, так і для вхідних та вихідних векторів; несправності можуть бути виявлені не тільки в регістрі станів R, але й у вхідному векторі X, в логіці Φ формування чергового коду стану, в логіці Ψ формування вихідних сигналів, а також в ланцюзі зворотного зв'язку; виявляються також недопустимі переходи FSM і переходи в недопустимі стани; запропоновані структурні моделі дозволяють не тільки виявляти несправності FSM, але й запобігати їх негативному впливу на об'єкт керування; комбіновані структурні моделі дають змогу одночасно виявляти несправності у всіх елементах FSM. Подальші дослідження будуть зосереджені на розробці структурних моделей для виправлення відмов FSM.

Ключові слова: структурна модель; виявлення несправностей; скінченний автомат; повітряна лінія; система керування; безпілотний літальний апарат (БПЛА); програмована користувачем вентильна матриця (ПКВМ).

Соловйов Валерій Васильович – проф., д-р техн. наук, Факультет комп'ютерних наук Білостоцького політехнічного університету, Білосток, Польща.

Valery Salauyou – DSc, PhD, Eng, Prof. Faculty of Computer Science, Bialystok University of Technology, Bialystok, Poland,
e-mail: v.salauyou@pb.edu.pl, ORCID: 0000-0002-9174-8588, Scopus Author ID: 55699021000.