Testable design of control digital automatic machines

Maryna Miroshnyk
dept. Specialized Computer Systems
Ukrainian State University of Railway Transport
Kharkiv, Ukraine
marinagmiro@gmail.com
0000-0002-2231-2529

Dariia Rakhlis
dept. Computer Engineering Design
Kharkiv National University of Radio Electronics
Kharkiv, Ukraine
dariia.rakhlis@nure.ua
0000-0002-6652-1840

Elvira Kulak
dept. Computer Engineering Design
Kharkiv National University of Radio Electronics
Kharkiv, Ukraine
elvira.kulak@nure.ua
0000-0002-8441-5187

Olexander Shkil
dept. Computer Engineering Design
Kharkiv National University of Radio Electronics
Kharkiv, Ukraine
oleksandr.shkil@nure.ua
0000-0003-1071-3445

Inna Filippenko
dept. Computer Engineering Design
Kharkiv National University of Radio Electronics
Kharkiv, Ukraine
inna.filippenko@nure.ua
0000-0002-3584-2107

***Abstract*—The aim of the work is to analyze hardware costs of ensuring the testability of finite state machines with various options for organizing an additional transition between FSM's states depending on the presence of an unconditional transition, a conditional transition, and the absence of transitions between states of analyzed FSM. The conclusion on additional hardware costs is made on the basis of a comparison of synthesis results of testable HDL-models by means of CAD FPGA. The paper solved the problem of computer-aided design of testable control FSM based on the application of methods for setting FSM in a given state. The best way to organize additional transitions during setting of control FSM in an arbitrary state is the transition for which the total hardware cost estimate for the excitation functions is minimal, taking into account the coding of FSM's states.**

***Keywords*—hardware costs, design for testability, control FSM, transition-output table, hardware description language, automated synthesis.**

I. INTRODUCTION

In the modern cycle of computer-aided design of digital devices (DD), the specification for the designed device is usually set in the form of a functioning algorithm, presented in one of the hardware description languages (HDL). For such way of DD's representing, the application of structural diagnostic methods oriented to the identification of constant faults becomes less effective. On the other hand, it is rather difficult to apply functional methods due to the dimension of modern digital devices.

One of the common ways for the initial description of specialized digital devices for data processing and control is a finite state machine (FSM), and the form of its presentation is the state table (ST) and the state diagram constructed on its basis. Due to the complexity of diagnostic experiments (DE) with FSM, different methods have been proposed to ensure testability, namely, modifications of automata-based models of digital devices, which introduce hardware redundancy both at the structural level by introducing additional inputs and outputs to ensure the simplicity of conducting DE and at the functional level by making additions and changes in the functional description of the FSM, specifically, in its ST. This approach allows us to make a testable FSM more easy-tested, that is, one for which the task of providing test diagnostics is solved as simply as possible within the established design costs. You can make an object easy-tested by reducing the cost of one or more of the main factors that determine the complexity of diagnosis or the cost of additional hardware costs.

Methods to increase the testability of the DD by introducing hardware redundancy into the circuit implementation are quite developed and are widely used in design. In the classic work on the development and classification of structural methods of testable design [1], the concepts of "controllability" and "observability" are introduced, which are the basis for assessing structural testability. The principles of the organization of shift registers in the storage part of the DD and the construction of a scanned path on their basis are described. In addition, modifications of well-known structural methods for constructing tests using scanning methods are proposed. Practical recommendations on the use of structural methods to increase the testability of the DD and construct diagnostic algorithms based on them are described in detail in [2].

In the fundamental work [3], methods of built-in self-testing are considered as the basis for ensuring the testability of the designed devices. The application of BIST (built-in self-test) technology in the design of digital systems on FPGA is shown. In [4], the ideas of test-private design based on Boundary Scan technology using the JTAG port were further

developed. IEEE 1149.x, IEEE 1500, and IEEE boundary scanning standards are reviewed, as well as the IEEE 1687 test design standard.

The functional methods of the design for testability are discussed in details in [5]. For digital FSM presented in the form of ST, the concepts of diagnosed and definitely-diagnosed classes of machines are introduced and methods for ST construction for these classes are proposed. The procedures for conducting DE with FSMs using installation, synchronizing, diagnostic and characteristic sequences are considered. In addition, the idea of increasing the testability of the FSM due to the introduction of hardware redundancy by expanding the input alphabet, the output alphabet and the alphabet of states is substantiated.

In [6], structural models of finite state machines are considered that allow to use values of output variables as codes of internal states. This approach was used in the synthesis of Mealy finite state machines on FPGA, which allows to reduce the cost of implementation by 1.5-2 times. Also, the proposed method of encoding states can reduce the power consumption of circuit implementations of finite state machines [7].

In [8], with the aim of increasing the testability of control finite state machine, authors described the concept of introducing hardware redundancy into the abstract state machine model, which consists of adding the column Sh to the state table (additional arc Sh in the state diagram of the FSM), which makes it possible to set the machine to any state in no more than n-1 cycles. At Sh = 1, the machine operates in the installation mode to any given state, and at Sh = 0, the machine implements the specified algorithm. During the automated synthesis of these FSM using models in the hardware description languages (HDL models), the introduction of an additional column is realized by adding the signal Sh to the description of the transition function in a two-process automata-based template of the HDL model. In the synthesis, it was implemented by additional multiplexers, which allowed us to organize a scan of FSM states during the diagnostic experiment. It was noted that the additional hardware costs in this case do not exceed 25-30%, depending on the type of FSM and the method of states' coding, but the natural order of states bypassing of the FSM was used, which, of course, affected the additional hardware costs.

In the paper [9], a practical method for constructing de Bruijn sequence generators based on shift registers with nonlinear feedback function is proposed. That algorithm allows to implement the recursive procedure for the formation of Hamiltonian cycles in the graph of the n-bit shift register based on the homomorphic mapping of its states with the (n-1)-bit shift register state graph and the use of finite state machine models of spanning tree graphs of shift register states. The effectiveness of the proposed method is proved by practical implementation of universal generator with an option to choose nonlinear feedback function arbitrary for built-in NLFSR.

The article [10] is devoted to the development and improvement of methods of quality control of the communication channel based on an assessment of the statistical characteristics of error stream at the output of discrete communication channel. The proposed methods are based on the topological representation of error formation on the length of the code blocks. This method allows to optimize the error weighting function of n-sequences in the set (n, m) partitions at the output of real communication channel. A method for determining parameters of error-correcting codes, which numerical values are determined on the base of values' minimization of potential probability of decoding errors, was proposed.

Thus, the urgent task is to minimize additional hardware costs by organizing the optimal bypass of FSM states.

The purpose of the article is the analysis of hardware costs for ensuring testability of finite state machines with different options for organizing an additional transition between FSM states, depending on the presence of an unconditional transition, a conditional transition and the absence of transitions between FSM states that are analyzed. Hardware costs are analyzed by comparing the synthesis results of HDL-models testability using CAD FPGA.

II. Analysis of Hardware Costs in the Automated Design of Digital State Machines

As an abstract model DD with memory we will use a finite state machine, which is determined by the five $W =< X, A, Y, \delta, \lambda >$, where $X = \{x_1, x_2, ..., x_m\}$ – is the set of letters of the input alphabet; $A = \{a_1, a_2, ..., a_n\}$ – set of FSM states; $Y = \{y_1, y_2, ..., y_r\}$ – set of the output alphabet; $\delta(a_i, x_k) = a_j$ – transitions function of the FSM; $\lambda(a_i, x_k) = y_\alpha$ – output function of the FSM.

In the high-level design of digital control devices based on finite state machines, the form for presenting the specifications of the designed device is a state-table or a state diagram. One of the ways to describe models of control systems in the form of finite state machines using VHDL language is an automata-based template, that is, such a way to describe models of control finite state machines, the specification for which is specified in the form of ST or SD. This is a special structure of the VHDL model in which the transition and output functions are allocated to separate processes (process), and the new state is assigned in a special process that is associated with synchronization. When designing testable control FSM, hardware redundancy, which provides easy testability, is advisable to introduce at the initial design stage, i.e., during construction of HDL models of devices. A finite state machine, for which a diagnostic experiment of minimal length can be constructed by ensuring that the machine is set to any state in minimum number of clock cycles, is called easy-tested.

Thus, a functional model of an abstract FSM is given in the form of ST or SD, and on its basis a VHDL model is constructed in the form of an automata-based template. It is necessary to consider various ways of introducing hardware redundancy into the VHDL model to ensure testability, and choose the optimal method from the point of view of additional hardware costs. Hardware redundancy in VHDL-models is provided by adding conditional operators to the HDL-code, which ensure the construction of the scanned path in the

storage part of the FSM, which is confirmed by the results of automated synthesis. The optimal way from the point of view of introducing additional hardware costs, we will consider the method that provides the minimum additional hardware costs according to assessment of Quine of the gate equivalent of a circuit and synthesis in a chip of programmable logic integrated circuits (FPGA) in automatic mode by computer-aided design tools (CAD).

Let's consider the possible options for adding a transition with signal Sh between nodes of the state diagram (FSM states) a_i and a_j.

The first option is to add Sh to the unconditional transition $(a_0 \to a_1)$ in the Moore FSM. Note that to use the above fragments of the state diagram in the process of automated synthesis, all transition options should be taken into account and, also, a back arc (which ensures the absence of nodes that "hang", Fig. 1, a). The ST of FSM fragments for the initial diagram and the diagram with Sh are shown in Fig. 1, b. As a result, two arcs turn up with signals Sh and $\overline{Sh}$ (Fig. 1c).

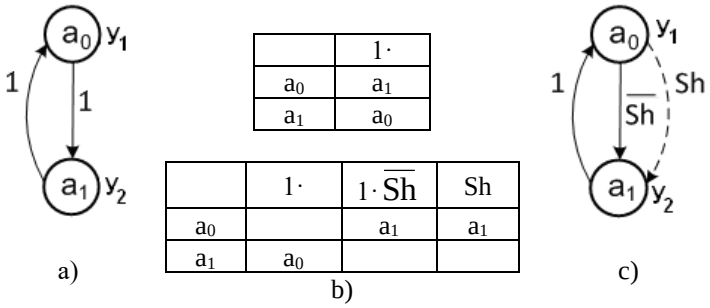


Figure 1 - – Adding Sh to the unconditional transition

In this case, the transition function to state a1 will have the form: $a_1 = a_0 Sh \vee a_0 \overline{Sh} = a_0$.

Applying the gluing law, we obtain a0, which means that when implementing the minimized expression there are no additional hardware costs. To encode two states of the FSM, one D-trigger is used, a0 is encoded 0, a1 is encoded 1. Thus, the expression a1 = a0 is actually implemented by the inverter.

Let's consider the VHDL model of the given fragment of the FSM (Fig. 2) and perform its automated synthesis (Fig. 3).

Synthesis results of this VHDL-model using a CAD package: XILINX ISE 10.1, Spartan 3E board, FPGA XC3S500E chip, Package FG 320 are shown below (fig. 3).

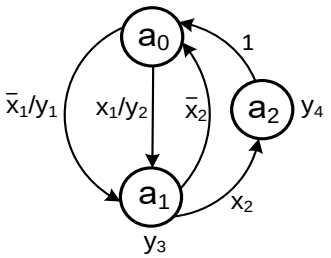
The second option is to add Sh to the transitions between ai and aj in the absence of transitions to other states other than aj. This situation is typical for Mealy FAM, but for the analysis of this situation we will use a mixed three-state FSM, for which the states a1 and a2 are reflected by the Moore FSM model, and the transitions $(a_0 \to a_1)$ by the Mealy FSM model. The state diagram of the proposed FSM is shown on Fig. 4 (a), and its ST – on Fig. 4 (b). For simplicity, in the ST the signal $\overline{Sh}$ is not indicated. The state diagram of the FSM with signals s Sh and $\overline{Sh}$ is shown on Fig. 4 (c).

```
library IEEE;
use IEEE.std_logic_1164.all;
entity FSM is
      port ( Sh, Reset,  Clk: in
STD_LOGIC;
      y1, y2: out STD_LOGIC);
end;
architecture Moore of FSM is
      type State_type is (a0, a1);
      signal State, NextState:
State_type;
begin
Sreg0_CurrentState: process (Clk, reset)
      begin
        if Reset='1' then State <= a0;
        elsif Clk'event and Clk = '0 then
State <= NextState;
        end if;
      end process;
Sreg0_NextState: process (State, Sh)
      begin
        case State is
          when  a0=> if sh='1' then
NextState <= a1;
            else NextState <= a1;
            end if;
          when  a1=>  NextState <= a0;
          when others => NextState <= a0;
        end case;
      end process;
      y1 <= '1' when State=a0 else '0';
      y2 <= '1' when State=a1 else '0';
end;
```

Figure 2 - VHDL-model of FSM fragment with an unconditional transition

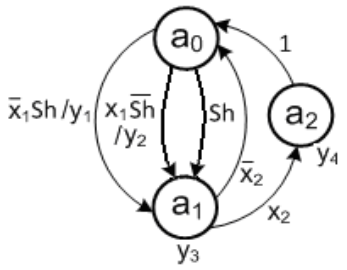
```
Synthesizing Unit <FSM>.
Related source file is
"C:/Temp/shkil2018/shkil_2states_2018/fsm
.vhd".
WARNING:Xst:647 - Input <Sh> is never
used. This port will be preserved and
left unconnected if it belongs to a top-
level block or it belongs to a sub-block
and the hierarchy of this sub-block is
preserved.
Found 1-bit register for signal
<State<0>>.
```

Figure 3 – Synthesis results of the FSM VHD- model with an unconditional transition



a)

b)	1·	x_1	$\overline{x_1}$	x_2	$\overline{x_2}$	Sh
a_0		a_1/ y_2	a_1/ y_1		a_1	a_1
a_1/y_3				a_2	a_0	
a_2/y_4	a_0					



c)

Figure 4 - Adding Sh in the presence of transitions only between a_0 and a_1

The transition function to state a1 for two output transitions and additional transition Sh will have the form: $a_1 = a_0 Sh \vee a_0 x_1 \overline{Sh} \vee a_0 \overline{x_1}\,\overline{Sh} = a_0 Sh \vee a_0 \overline{Sh} = a_0$.

Applying the gluing law step by step, we obtain a0, which means that there is no additional hardware costs when implementing the minimized expression. For any number of arcs between ai and aj, the result will be the same. When the initial expression is implemented, the additional hardware costs are an additional input $\overline{Sh}$ to all gates, which implements the terms of the initial transition function (Fig. 5, a) and a gate that implements the aiSh term (Fig. 5, b).

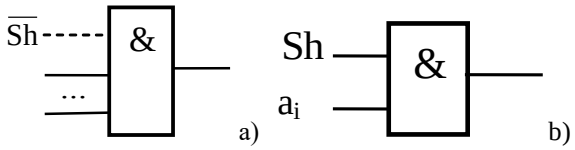


Figure 5 - Additional hardware costs in the presence of transitions only between a_0 and a_1

Let's consider the fragment of the FSM VHDL-model (Fig. 6) and perform its automated synthesis (Fig. 7). To reduce the volume of VHDL code in the future, it is advisable to list only fragments of the architecture of VHDL models, namely, only transition functions.

```
  Sreg0_NextState: process (State, Sh,
x1, x2)
  begin
    case State is
      when  a0=>if sh='1' then NextState
<= a1;
            elsif x1 ='1' then NextState
<= a1; y2 <= '1';
      else NextState <= a1; y1 <= '1';
            end if;
      when  a1=>if x2='1' then NextState
<= a2;
            else NextState <= a0;
            end if;
      when  a2=> NextState <= a0;
      when others => NextState <= a0;
    end case;
  end process;
  y3 <= '1' when State=a0 else '0';
  y4 <= '1' when State=a1 else '0';
                    end;
```

Figure 6 – A fragment of the FSM VHDL model with transitions between a_0 and a_1

```
Found finite state machine <FSM_0> for
signal <State>.
Analyzing FSM <FSM_0> for best encoding.
Optimizing FSM <State/FSM> on signal
<State[1:2]> with gray encoding.
State | Encoding   a0    | 00   a1    |
01   a2    | 11
```

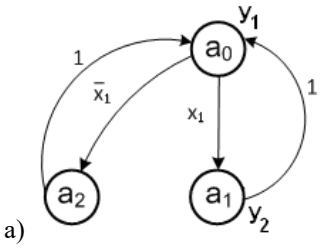
Figure 7– Synthesis results of the FSM VHDL model with transitions only between a_0 and a_1

The obtained circuit confirms that the signal Sh (In0) through the AND gate is added to the excitation function of each of triggers.

The third option is to add Sh to the transitions between ai and aj in the presence of transitions and to other states other than aj (Fig. 8a).

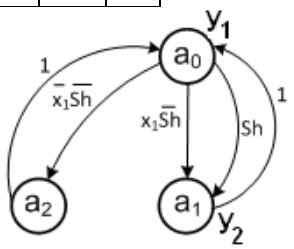
In this case, the transition function to the state a_i (a_1) for one output transition, the added transition Sh and the transition to the state (Fig. 8, c) will have the form: $a_j = a_i Sh \vee a_i x_1 \overline{Sh} = a_i (Sh \vee x_1 \overline{Sh}) = a_i Sh \vee a_i x_1$.

By applying the Blake-Poretsky law we got the expression $a_i Sh \vee a_i x_1$. This means that the additional hardware costs when implementing the minimized expression for the transition function to the state a_i is a gate that implements the term $a_i Sh$ (Fig. 5, b). When implementing the initial expression for the function transition a_j, additional hardware costs are a gate that implements the term $a_i Sh$ plus an additional input to the gate that implements the term of the original transition function (Figs. 5, a and 8, c).



a)

b)		1·	x_1	$\overline{x_1}$	Sh
a_0	y_1		a1	a_2	a_1
a_1	y_2	a_0			
a_2		a_0			



c).

Figure 8 – Adding Sh in the presence of transitions between a_0 and a_1 and other output transitions

If there are more than one transitions $a_i \rightarrow a_j$ in the state diagram, then an additional input is added to each gate for of these transitions. In addition, in this variant $\forall(a_i \rightarrow a_k)$, $a_k \neq a_j$ and transitions ai →ak also give additional hardware costs in the form of an additional input $\overline{Sh}$ to a gate that implements the term of the transitions function, for example, on Fig. 8, c $a_k = a_i \overline{x_1}\,\overline{Sh}$. Consider the VHDL-model of the reduced fragment of the automaton (Fig. 9) and perform its automated synthesis (Fig. 10).

```
  Sreg0_NextState: process (State, Sh, x)
  begin
    case State is
    when  a0=> if sh='1' then NextState
<= a1;
      elsif x ='1' then NextState <= a1;
            else NextState <= a2;
            end if;
      when  a1=>  NextState <= a0;
      when  a2=>  NextState <= a0;
      when others => NextState <= a0;
    end case;           end process;
  y1 <= '1' when State=a0 else '0';
  y2 <= '1' when State=a1 else '0';
  end;
```

Figure 9 – A fragment of the FSM VHDL-model in the presence of transitions $a_0 \rightarrow a_1$ and other transitions

```
Analyzing FSM <FSM_0> for best encoding.
Optimizing FSM <State/FSM> on signal
<State[1:2]> with speed1 encoding.
State | Encoding   a0 | 10    a1 | 01
a2 | 00
```

Figure 10 – Synthesis results of the FSM VHDL-model in the presence of transitions between a_0 and a_1 and other output transitions

Based on the foregoing, we can make following conclusions.

1) To select the best option for the function of additional transitions with Sh from the point of view of minimizing hardware costs, first option has higher priority, after that second, then 3 and last is 4.

2) When choosing the 1st or 2nd options during implementing not minimized expression of the transition function, it is better to choose the 1st one, since it is smaller in the number of output arcs, which means there will be fewer additional inputs to the gate. When implementing the minimized expression of the transition function, there is no difference between 1st and 2nd options, in both cases there are no additional hardware costs.

3) When choosing between 3rd and 4th options, the number of output arcs that satisfies the condition: $\forall(a_i \rightarrow a_k)$, $a_k \neq a_j$ will play determining role. In which case it is less, choose this one, since an additional gate to the transition ai →aj with Sh goes in both cases. This is important when choosing a method for implementing a minimized expression of the transition function.

To implement not minimized expression of the transition function, it is necessary in option 3 to calculate the number of arcs for comparison with option 4 to the condition $\forall(a_i \rightarrow a_k)$, $a_k \neq a_j$ add $\forall(a_i \rightarrow a_j)$ вesides $a_i \rightarrow a_j$ with Sh.

ACKNOWLEDGMENT

The paper presents a method for computer-aided design of testable control finite state machines by introducing hardware redundancy. The model of the FSM is presented in the VHDL language in the form of an automata-based template. A way to solve this problem is to introduce additional fragments of the VHDL-code, which ensure the automatic installation of the state machine into an arbitrary state. A method of state table expanding is proposed, which provides a traversal mode for all nodes of the FSM state diagram in the diagnostic mode. The hardware costs are analyzed for different options for organizing an additional transition between FSM states, depending on the presence of an unconditional transition, a conditional transition and the absence of transitions between FSM states. When choosing an additional transition (arc Sh in the FSM state diagram), that state-successor is selected for which the total estimate of the hardware costs for the excitation functions is minimal, taking into account the encoding of FSM states. This approach increases the controllability in determining the optimal bypass of FSM states, which significantly improves its

testability. Modeling FSM VHDL-models with Active-HDL tools confirmed the efficiency of this approach. The synthesis of these models using CAD tools XILINX ISE confirmed the testability receipt of the structures and confirmed the receipt of minimum hardware costs using the proposed method.

The scientific novelty of the work lies in the further development of the method of optimizing hardware costs while increasing the testability of finite state machines by expanding the input alphabet in HDL models in the form of an automata-based template, which made it possible to automate the design process of testable FSM with minimal hardware costs.

The practical value of the results lies in the development of procedures for introducing redundancy and expanding the input alphabet in HDL-models of finite-state machines in the form of an automata-based template by introducing additional conditional operators in the HDL-code, ensuring the installation of the FSM into an arbitrary state with minimal hardware costs. The developed procedures can be applied in the development of an additional CAD software module for digital devices, which will automatically generate an HDL code of a testable finite state machine.

REFERENCES

[1] Bennetts R.G. Proektirovanie testoprigodnyh logicheskih shem: per. s angl. / R. G. Bennetts. – M.: Radio i svjaz', 1990. – 176 p.

[2] Gorjashko A.P. Proektirovanie legko testiruemyh diskretnyh ustrojstv: idei, metody, realizacija / A.P. Gorjashko // Avtomatika i telemehanika. – 1984. – № 7. – P. 5-35.

[3] Stroud C.E. A designer's guide to built-in self-test / Charles E. Stroud. – Kluwer Academic Publishing, 2002 – 319 p.

[4] Gorodetsky A. Introduction to JTAG and DFT technology. Testing in edge scanning technologies and testable design /A. Gorodetsky // Palmarium Academic Publishing. – Germany, 2012.– 308 p.

[5] Tocenko V.G. Algoritmy tehnicheskogo diagnostirovanija cifrovyh ustrojstv / V.G. Tocenko. – M.: Radio i svjaz', 1985. – 240 p.

[6] Solov'ev V.V. Minimization of mealy finite-state machines by using the values of the output variables for state assignment / V.V. Solov'ev // Journal of Computer and Systems Sciences International. – January 2017.– Volume 56, Issue 1. – .P. 96–104.

[7] Solov'ev V.V. Minimization of Power Consumption of Finite State Machines by Splitting Their Internal States / T.N.Grzes, V.V. Solov'ev // Journal of Computer and Systems Sciences International.– 2015. – Vol. 54, No. 3. – P. 367–374.

[8] Shkil A. Design Automation of Testable Finite State Machines / M.Miroschnyk, Y. Pakhomov, E. German, A. Shkil, E. Kulak, D. Kucherenko // Proceedings of the International Sympos. EWDTS'2017, September 29-October 2, 2017 – Novi Sad, Serbia, 2017. – P.203–208.

[9] Miroshnyk M.A. Practical methods for de Bruijn sequences generation using non-linear feedback shift registers / Olexandr Demihev Maryna Miroshnyk, Dmitrij Karaman, Filippenko Inna, Krylova Viktoria, Tetyana // 14th International Conference on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering, TCSET 2018 - Proceedings, feabrary 2018 - Lviv-Slavske, Ukraine. – p.35/

[10] Miroshnyk, M.A. Communication channel statistical characteristics research methods. / Miroshnyk, M., Krylova, V. // Modern Problems of Radio Engineering, Telecommunications and Computer Science, Proceedings of the 13th International Conference on TCSET 2016.