UDC 681.326

DESIGN TIMED FSM WITH VHDL MOORE PATTERN

Miroshnyk M. A. – Dr. Sc., Professor, Professor of the Department of Specialized Computer Systems, Ukrainian State University of Railway Transport, Kharkiv, Ukraine.

Shkil A. S. – PhD, Associate Professor, Associate Professor of the Department of Design Automation Department, Kharkiv National University of Radioelectronics, Kharkiv, Ukraine.

Kulak E. N. – PhD, Associate Professor, Associate Professor of the Department of Design Automation Department, Kharkiv National University of Radioelectronics, Kharkiv, Ukraine.

Rakhlis D. Y. – PhD, Associate Professor, Associate Professor of the Department of Design Automation Department, Kharkiv National University of Radioelectronics, Kharkiv, Ukraine.

Miroshnyk A. M. – Post-graduate student of the Department of Design Automation Department, Kharkiv National University of Radioelectronics, Kharkiv, Ukraine.

Malahov N. V. – Graduate student of the Department of Design Automation Department, Kharkiv National University of Radioelectronics, Kharkiv, Ukraine.

ABSTRACT

Context. The relevance of the work consists in the development of computer-aided design methods for automatic real-time logic control devices by developing a single template in a synthesized subset of the hardware description language in the style of automata-based programming with implementation on the PLD hardware platform (FPGA, CPLD). Development of the description template for timed control finite state machines (FSM) in the hardware description language VHDL, automated synthesis and implementation of the model in PLD (FPGAs, CPLDs) using Xilinx ISE, subsequent analysis of the received circuit implementation for compliance with values of timing parameters of the circuit after implementation.

Objective. The aim of the work is to develop principles for constructing models of timed control FSM in the VHDL hardware description language. In this work, we solved the problem of constructing a pattern for describing models of timed control Moore FSM using VHDL, automated synthesis and implementation of the obtained VHDL model in PLDs (FPGA, CPLD) using Xilinx ISE and subsequent analysis of the resulting circuit implementation for compliance with values of timing parameters of the circuit after implementation.

Method. Realization of models' parameters of timed FSM in logical control systems using VHDL statements. Development of VHDL language constructions of timed FSM models for timing parameters implementation that provide the correct automated synthesis and implementation of these models in PLDs (FPGA, CPLD) using CAD tools Xilinx ISE.

Results. Synthesis and implementation of proposed templates of VHDL-models of timed control Moore FSM in logic control systems by XILINX ISE CAD tools confirmed the receipt of not redundant circuits in PLD (FPGA, CPLD), and simulation after implementation showed the efficiency of such models.

Conclusions. The work solves the problem of computer-aided design of timed control FSM in real-time logic control systems. To solve this problem, VHDL-models of timed control Moore FSM were developed, which made it possible to implement control FSM with time constraints, timeouts and output delays. Automated synthesis and simulation of VHDL models based on the developed templates confirmed the efficiency and correctness of the proposed models.

The scientific novelty of the work consists in the further development of methods for constructing templates of HDL models of timed control Moore FSM, which made it possible to implement control FSM with time constraints, timeouts and output delays, as well as perform their correct automated synthesis and simulation.

The practical value of results is in the development of procedures for constructing VHDL models of timed Moore control FSM in real-time logic control systems, which made it possible to automate the synthesis of control FSM taking into account the possibility of processing external events and implementing arbitrary delays for output signals and to increase the flexibility and speed of designed systems. The developed procedures can be useful for designers of timed control FSM in Xilinx ISE.

KEYWORDS: timed finite state machine, state diagram, hardware description language, automata-based pattern, simulation, synthesis, implementation, CAD, Xilinx ISE.

ABBREVIATIONS

ASM is an algorithmic state machine;

C is a programming language;

CAD is a computer-aided design system;

CLK is a synchronization signal;

CPLD is a complex programmable logic device;

DD is a digital device;

FPGA is a field-programmable gate array;

FSM is a finite state machine;

HDL is a hardware description language;

ModelSim is a software tool produced by Mentor Graphics;

PLD is a programmable logic device;

RTL is a register transfer level;

SD is a state diagram;

TB is a testbench;

TO is the timeout;

TFSM is a timed finite state machine;

VHDL is a very high speed integrated circuits HDL;

UUT is a unit under test;

Xilinx ISE (Integrated Synthesis Environment) is a software tool produced by Xilinx.

DOI 10.15588/1607-3274-2020-2-14

NOMENCLATURE

X set of input signals,

X_i is an i-th element of a set of $\{X_C, X_E\}$;

T_c set of timed variables for timing constraints on each arc of the state diagram;

t_{ci} is an ci-th element of a set of $\{t_{c1}, t_{c2}...t_{cp}\}$, $t_{ci} = \{1, k\}$, $k = \{1, \infty\}$;

T_{to} is set of timed variables for timeouts of each state of the FSM;

t_{toi} is an toi-th element of a set of $\{t_{to1}, t_{to2}...t_{ton}\}$, $t_{toi} = \{1, n\}$;

T_d set of delays for implementing the corresponding output signal;

t_{di} is an tdi-th element of a set of $\{t_{d1}, t_{d2}...t_{dm}\}$, $t_{dm} = \{1, l\}$;

$Z(t+1)$ is the set of internal variables which encoding determines FSM states,

$Y(t)$ is the set of output signals;

t is the FSM' time which is determined in the FSM clock cycles;

g is outputs function of the structural FSM;

f is transitions function of the structural FSM;

T is the set of timing parameters of the FSM;

t_c is timing constraints;

t_o is input timeouts;

t_d is output delays;

X_C set of input signals from the control object;

X_E set of external events;

Y_c set of reactions (control signals),

Y_F iset of activities (initial functions);

g is output function;

z_0 is code of the initial state of the FSM;

n is number of arcs in the state diagram;

p is maximum number of constraints on transitions to the i-th node of the state diagram in the event processing mode;

k is corresponds exclusively to the transition event function;

t_{toi} is timeout for each state;

n is the number of states of the FSM;

m is number of output variables;

l is maximum number of clock cycles for implementing output signals in the indicated state of the FSM;

$t_{to}(a_i)$ is a timeout;

$t_{dj}(a_i)$ is a output delays;

c_1 is the lower limit of timing constraints in FSM clocks;

c_2 is the upper limit of indicated constraints;

t_0 is beginning of timing constraints' "window";

t_1 is ending of timing constraints' "window" ;

X iset of Onn, St, Btn;

Y set of R1, YRG, YGR, G1, R2, G2;

R1, YRG, YGR, G1, R2, G2 are elements of a set of Y;

Onn, St, Btn are elements of a set of X;

On is a signal to turn on the traffic light and start the night cycle;

St is a signal to start the daytime traffic cycle;

Btn is signal to turn on the green light on a pedestrian crossing;

R1 is a signal to turn on red light on a main road;

G1 is a signal to turn on green light on a main road;

YRG is a signal to turn on yellow light on a main road (on transition R – G);

YGR is a signal to turn on yellow on the main road (on transition G – R);

R2 is a signal to turn on red light on a pedestrian crossing;

G2 is a signal to turn on green light on a pedestrian crossing.

INTRODUCTION

Among the whole set of control systems, a significant part is logic control systems, in which control signals take logic zero or one depending on the boundary values of physical quantities that determine these parameters. For the technical implementation of these systems, the most suitable model is a structural FSM, and the state diagram is a visual representation of the functioning algorithm. A distinctive feature of finite state machines in logic control systems is the presence among input values not only signals of the control object, but also external, to the controlled system, events of the external world, which ensure the interaction of the logic control system with the external environment.

Control finite state machine operate in machine time, which is determined by the clock cycles of the FSM. But most real logic control systems interact with the outside world in metric time, i.e. they are real-time systems in which the resulting action (activity) depends not only on the logic values of the control signals, but also on the time during which these actions are performed. For their implementation, it is customary to use the model of timed FSM, which, through the implementation of timing parameters in machine time, allows taking into account the influence of the metric time on transitions between the technical states of the controlled system.

Any local digital DD that implements an information processing or control algorithm can be implemented in two ways: hardware or software-hardware. During hardware implementation, a given algorithm is described in HDL and synthesized by tools of CAD systems in PLD. The advantage of this approach is the hardware flexibility (the ability to implement any algorithm) and sufficiently large speed. When describing the functionality of DD for logic control in CAD, one of constructing styles of the HDL code structure is automata-based programming style, and it is customary to use the so-called FSM template to correctly represent HDL models of control FSM from the point of circuit synthesis. Features of the con-

© Miroshnyk M. A., Shkil O. S., 2020
DOI 10.15588/1607-3274-2020-2-14

struction of these templates affect synthesis results of automatic logic control devices using CAD for PLD.

Thus, **the task of developing** single template in the hardware description language for describing automatic real-time logic control devices in the style of automata-based programming with their implementation on PLD hardware platform (FPGA, CPLD) becomes urgent.

The research object in this work is the process of computer-aided design of control FSMs in logical control systems. The research subject is the design method of VHDL models for timed control Moore FSM, their synthesis and implementation in FPGA.

The aim of this work is to develop patterns for description of timed control FSM in the hardware description language VHDL, automated synthesis and implementation of the obtained model in PLDs (FPGA, CPLD) using Xilinx ISE and subsequent analysis of the resulting circuit implementation for compliance with values of timing parameters of the circuit after implementation.

1 PROBLEM STATEMENT

Let Moore timed FSM model was represented as $Y(t) = g(\ Z(t), T)$, $Z(t+1) = f(X(t), Z(t), T)$. The visual formalism of the model specification for the designed timed FSM is a temporal state diagram and a timing diagram (waveform). The temporal state diagram defines a list of input variables, output variables and FSM states, as well as delays and timing constraints, i.e. is a complete mathematical model of a timed FSM. Waveform reflects the functioning law of the controlled device in FSM time.

For the Moore FSM model it is necessary to develop a pattern of the VHDL model using style of automata-based programming, implement it using the example of building a VHDL model of the control FSM in a traffic light control system, and confirm the correctness of the proposed model by simulation, synthesis and implementation in PLD by CAD tools XILINX ISE.

The quality criterion for the obtained VHDL model is the coincidence of simulation results of the control FSM post-implementation model with the timing diagram which is specified in the specification.

2 REVIEW OF THE LITERATURE

Methods of using and implementing timed FSM in real-time systems are quite developed and are widely used in the design of logic control systems.

In [1], the role and place of logic control systems in the class of reactive systems that operate on the basis of reactions to external events is defined. To describe the control part of logic control systems, finite state machines, usually Moore models are used. To describe FSM in logic control systems state diagram are used, which are not the only visual representation of FSM' functioning algorithm, but also it's a full mathematical model. The designing method of automata-based logic control systems which takes into account real time and processing of external events is given.

In the classic work [2], to describe the behavior of reactive systems, it is proposed to use state charts that extend standard state diagrams. To describe the time, during which system stays in a certain state, the upper and lower time boundaries of being in the state and the concept of timeout, defined as delay in the implementation of transition to new state, were introduced. To describe the output control signals, the concept of action is introduced, the lifetime of which is ideally zero, and activity, which is performed for a certain period of time, determined by special events start and stop.

The concept of timed FSM, is a way of describing real-time systems, as introduced in [3, 4]. The state diagram of the FSM is supplemented by finite set of timers that take real values. Nodes of the diagram are called positions, and the edges are called transitions. Each timer is reset to zero at the time of transition and increases its value with each FSM' cycle. Each transition is associated with clock constraint, which means that this transition can only be carried out if the current timer values satisfy this restriction. Each position has a timer' constraint called an invariant; the system can be in this position only as long as its invariant is satisfied.

In [5], formal testing methods are studied that take into account timing characteristics of the system. To describe the behavior of the system, the model of TFSM was used, where instead of many timers, one timed variable is introduced. Each transition between TFSM' states is characterized by the time (delay) during which will be completed. To test TFSM, the concept of timed trace, as a sequence of transitions between states and the time during which they were performed, is introduced. The algorithm for generating a complete set of tests that allows checking whether the model matches the specified timing parameters is proposed. In [6], the TFSM model is extended due to the introduction of timeout in the state and delay – during the transition, which makes it possible to take into account more accurately timing parameters during testing timed FSM by constructing timed traces, starting from the initial state.

When constructing tests for timed FSM, the TFSM model, which takes into account timeouts in states and delays of output signals with respect to the implementation of state transition, is considered in [7]. At the same time, it is taken into account that if no input signal is received during the timeout, then the FSM switches to the next state.

In general, the timed FSM model includes three types of timing parameters: timeout in states, time limit on receiving input signals, and an input signal processing time, i.e. delay of the output signal relative to the input. Moreover, timed FSM with a smaller number of parameters can be considered [8, 9, 10]. In these works, minimization problems of timed FSM, checking their equivalence, and creation of tests are considered.

In [11], different design methods for digital controllers that implement real-time systems on different technological platforms, including programmable logic and microcontrollers, were proposed. To implement control algorithms, it was proposed to use different mathematical approaches, including UML diagrams, hardware description languages for finite state machine models, and Petri nets for describing microcontroller devices.

During creation of software models of logic control devices, the automata-based programming style is widely used

© Miroshnyk M. A., Shkil O. S., 2020
DOI 10.15588/1607-3274-2020-2-14

[12]. The essence of automata-based programming is to separate the description of device's behavior logic from the description of its output signals. Automata-based programs are strictly structured and they have three types of functions: transition functions, output functions, and functions for assigning a new state, which take into account time delays. Automata-based programs are also strictly stereotyped using multi-position selection operators (switch, case) and conditional operators (if, elsif – else) in the C programming language.

The method for implementing models of timed FSM in the VHDL language was proposed in [13]. To implement delays in the states of the Moore FSM, it is proposed to use loops in states and a special variable count, which decreases by 1 in each FSM cycle, which corresponds to a sync pulse. In terms of implementing the structure of the HDL model, it is proposed to use a single-process template. Results of behavioral simulation of traffic lights were presented.

Three categories of state machines were introduced in [14]: regular (simple) state machines, timed state machines, and recursive state machines. For these categories of machines, different templates of HDL models in the VHDL and Verilog languages are presented, as well as the results of their simulation in the ModelSim system (from Mentor Graphics) and synthesis using ISE (from Xilinx).

When analyzing timing parameters of the designed digital devices based on FPGA, the developer has the opportunity to conduct timing simulation at the following stages: after creating the initial project description (Simulate Behavioral VHDL Model); after performing synthesis and translation (Simulate Post-Translate VHDL Model); after the phase of mapping the logic description of the project to the physical resources of the crystal (Simulate Post-Map VHDL Model); after completion of the placement and tracing procedures (Simulate Post-Place & Route VHDL Model) [15]. Such organization allows to detect possible errors at the earlier stage of design, and thereby to avoid significant time losses.

3 MATERIALS AND METHODS

During description of real-time control systems' behavior, it is necessary to take into account time aspects of their behavior. For this, the state machine model is expanded by introducing a timed variable, and the concept of timed FSM is introduced. Logic control devices, built on the basis of timed FSM, operate in machine time, that are measured in FSM cycles, that is, discrete periods of time during which the machine passes from one state to another. The duration of the FSM cycle in real devices, as a rule, is determined by the clock Clk. Based on this timed variables are also measured in FSM cycles.

As a rule, three parameters are used to describe time aspects in timed FSM model: timing constraints t_c, (incoming) timeouts t_{to}, and output delays td, which are sometimes called output timeouts [10]. The input timeout determines the maximum waiting time for the input event for each state of the FSM. If input signal didn't came during timeout, then FSM start a survey of the input variables and go to another state. Timing constraints are intervals at transitions that limit the time during which a transition can be completed. Output delays (output timeouts) reflect the time taken by the FSM to complete the transition, i.e, the output signal will appear on the output after a time interval during which the transition is completed, which is determined by the output delay.

In logic control systems, the concept of «input values» is divided into input actions and events. Events along with input variables provide the interaction of the control FSM with the external environment. Input actions are implemented by FSM by interrogation of input variables in accordance with the algorithm of its operation in the control cycle, and events are realized instantly and lead to a change in the state of the FSM.

Depending on the purpose and features of using models of timed control FSM, there are many modifications of such models that take into account both the method of processing events and the method of processing delays in the implementation of transitions and output signals of the FSM.

Based on features of the functioning of logic control systems, a model of a structural timed FSM was proposed in [16]. It can be represented by nine W is a set of $X, Y, Z, f, g, z_0, T_c, T_{to}, T_d$.

In general, a timed FSM can contain all three timing parameters, but for a specific task, timed machines with one or two of these parameters can be used.

Timed FSM model, which consists of three timing parameters $< t_c, t_{to}, t_d >$ cannot be directly attributed to the traditional Moore model. Its output function is similar to the Moore FSM, but the output signal is formed taking into account the delay, but not during FSM' transition to a new state. The time of appearance (change) of output signals is tied to the working edge of the clock signal. In the proposed model of a timed FSM, the logic of its operation is as follows.

When the FSM goes into the current state a_i, the main timing parameter $t_{to}(a_i)$ is determined for it, i.e., the time during which the FSM must be in the current state, if an external event ahead of time transfers the machine to another state. t_{to} is detected in FSM cycles. After the time t_{to} has elapsed, the FSM responds to the input signals (interrogates them) and goes to the next state.

Output signals of the FSM in the current state a_i appear at outputs at the moment determined by $t_{dj}(a_i)$, that is, by output delays for signals y_j in the state a_i. For each of the output signals y_j, its delay is determined in FSM cycles and can be different. At $t_{dj}(a_i) = 0$, the model of timed FSM approaches the classical Moore model.

Processing external events is as follows. For each state a_i, input constraints $t_c(a_i)$ are set, that is, the period of time during which the FSM, in state a_i, can process incoming events. Timing constraints are determined in FSM cycles and calculated as $t_c = (t_1–t_0)$. For $t_1 = \infty$ and t_0=0, timed FSM without input timing constraints is considered. If an external event occurred outside of the timing constraints' "window", the FSM does not respond to it.

© Miroshnyk M. A., Shkil O. S., 2020
DOI 10.15588/1607-3274-2020-2-14

To describe a timed FSM, a temporal state diagram is used, which is represented in the form of FSM template in the hardware description language. To implement the Moore FSM model with a single timed variable, which is represented in the hardware description language, it is proposed to use an additional counter count, which is used to count the number of FSM cycles during which the FSM implements certain timing parameters. When the timed FSM makes a transition to a new state, the value of this counter is reset to 0. In a two-process VHDL template, the assignment of the new state and the new value of the counter occur in one process which is associated with the formation of the synchronous sequential part of the FSM. The VHDL code of the synchronization process, the assignment of the new state and the new counter value are shown in Listing 1.

Listing 1. VHDL code of the formation process of the FSM sequential part

```
process (Clk, Reset)
  begin
    if Rst = '1' then   state <= a1;
count <= ( others => '0' );
    elsif rising_edge(Clk) then
state <= next_state;
count <= next_count;
    end if;
  end process;
```

It should be noted that in the second process, which describes the combinational part of the FSM, it is necessary to add next_count <= (others => '0'); after the standard when others => next_state <= a1;. It is important.

Let's consider implementation approach of the three aforementioned timing parameters in the VHDL hardware description language, namely, incoming timeouts, timing constraints on events processing, output delays.

The TO_i timeout is realized by a multiple transition from state to the same state, wherein the number of transitions determined by the number of FSM timeout cycle. The value of the counter is compared with TO_i-1, since during transition to a_i state, the FSM will stay there during one cycle until the count will be checked and that the timeout would be equal exactly to TO_i cycles, then TO_i-1 cycles must be repeated. That is, in the temporal state diagram, the state timeout is implemented using loops, conditions for which are checks of counter signal value. Fig. 1 shows a graphical representation of the Moore FSM state with timeout TO_i and timing diagram of this timeout implementation.

A fragment of the VHDL-code of timeout implementation for state a1 is shown in Listing 2.

Listing 2. VHDL-code timeout implementation

```
process (state, count, … )
   begin
next_count <= (others => '0');
case state is
  when a1 =>
if count < TO1 - 1 then
    next_state <= a1;
      next_count <= count + 1;
   else  next_state <= a2;
   end if;
```

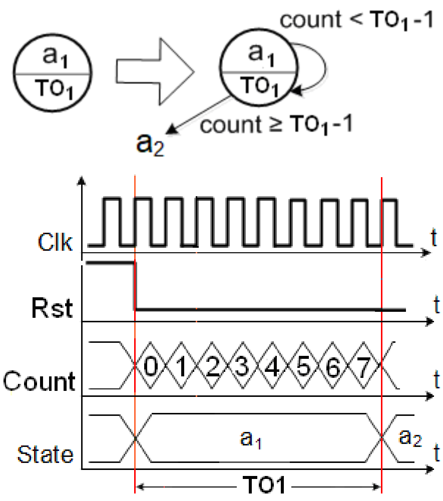

Figure 1 – Implementation of the timeout in the state of the Moore FSM

Timing constraints t_c, during which the processing of external events in the indicated state of the FSM is allowed, are defined as $t_c = [c_l, c_2]$. To implement the time limit, it is necessary to compare the value of the counter signal with the lower and upper bounds of the corresponding time limit c_l. Let's consider the lower time limit c_l. The counter value is compared with c_l-1, since during transition to state a_i, FSM stays there during one clock cycle until the count will be checked and so that the lower boundary corresponds exactly to c_l clocks, it is necessary more c_l-1 clock cycles during which FSM will stay in this state. Similarly, in the last cycle, in which event processing is enabled, the counter value is compared with c_2-1.

Fig. 2 presents conditions of the transition by an external event (Btn), taking into account timing constraints, and timing diagram of the process for external event processing under the condition $t_c = [2, 5]$. The event triggers on the third FSM clock. A fragment of the VHDL-code of event processing in the state a_l is shown in listing 3.

Listing 3. VHDL code for timing constraints and event processing

```
process (state, Btn, count, …  )
begin
next_count <= (others => '0');
case state is
when a1 =>
 if Btn = '1' and count >= c1-1 and count < c2 then
        next_state <= a3;
 elsif count < TO1 - 1 then
        next_state <=a1;
        next_count <= count + 1;
 else  next_state <= a2;
 end if;
```

© Miroshnyk M. A., Shkil O. S., 2020
DOI 10.15588/1607-3274-2020-2-14

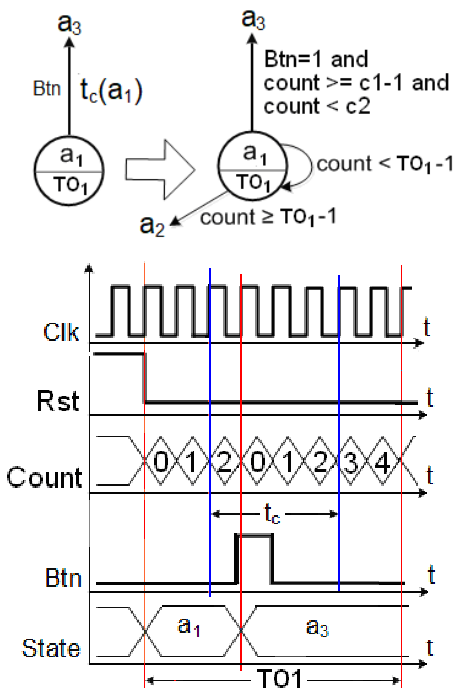

Figure 2 – Implementation of timing constraints and events' processing

For the hardware implementation of output delays, it is necessary that values of the output signals of the Moore FSM depend not only on the current state, but also on the current value of the counter, which determines the delay of the output signal from the moment when FSM moves to the corresponding state. Fig. 3 shows the implementation of the initial delay during d_1 FSM clock cycles (d_1=2) for signal y_1 in the form of state diagram and timing diagram.

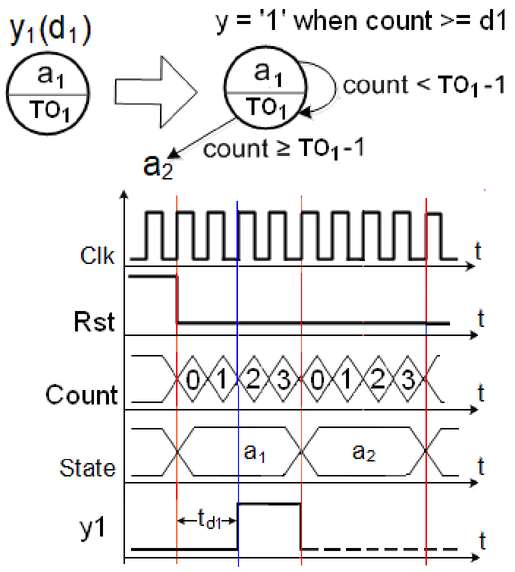

Figure 3 – Implementation of output delays for signal y_1

Note that the output signal in the VHDL model of the Moore FSM is realized using a conditional signal assignment statement outside process.

```
y1 <= '1'  when ( state = a1 and count >= d1 ).
```

To implement the VHDL-model of the timed FSM, it is proposed to use a two-process template: one process to describe sequential part of the FSM and second process to describe transition function and counting function of FSM clock cycles.

4 EXPERIMENTS

To test the proposed method of VHDL-models formation of timed FSM, let's consider a traffic light control system on a pedestrian crossing.

Let's consider the functioning algorithm of the traffic light, which operates in two modes: daytime and night.

The set of input signals X a set of Onn, St, Btn, Onn.

Thus, Onn and St are actions, and Btn is an event.

The set of output signals for traffic lights Y is a set of R1, YRG, YGR, G1, R2, G2. Note that $\forall R \rightarrow \overline{G}$.

The interface of the traffic light control system is shown in Fig. 4.

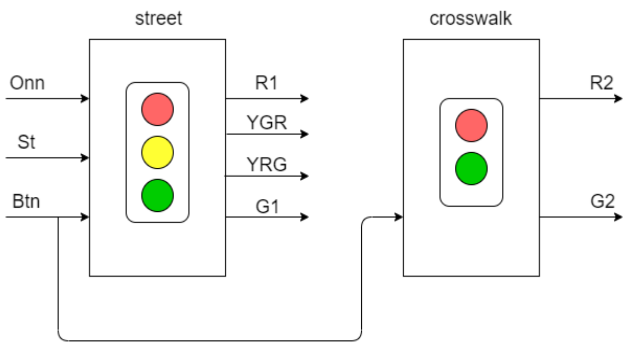

Figure 4 – Traffic light control system interface

Let's define states of the control FSM:

– state a_1 – turning on the FSM, there are no output signals, delay to_1;

– state a_2 – yellow when changing (G–R), outputs {YGR, R1, R2}, delay to_2;

– state a_3 – red on a road, green on a pedestrian crossing, outputs {R1, G2}, delay to_3;

– state a_4 – yellow when changing (R–G), outputs {YRG, R1, R2}, delay to_2;

– state a_5 – green on a road, red on a pedestrian crossing, outputs {G1, R2}, delay to_3;

– state a_6 – switching to green on a pedestrian crossing and red on a road by button Btn, delay to_6, outputs {R1, G2} with delay of the appearance t_d, delay in the state to_6;

– state a_7 – only yellow is on, delay is to_1.

The traffic light algorithm is as follows. When the control device is turned on (Onn = 1), the night cycle of the traffic light starts, yellow light flashes on the main road (a_1 – a_7), the traffic light on a pedestrian crossing does not work. After starting the daytime cycle (St = 1)

DOI 10.15588/1607-3274-2020-2-14

the transition' system ($a_2 - a_3 - a_4 - a_5 - a_2$) is implemented. At state a_5, when on a main road green light is on, reception "window" t_c for external event Btn is defined (pressing the jump button). When this event is processed, the control FSM go to state a_6. At the same time, on a main road red light is turned on, but red light is held up on a pedestrian crossing, and green is turned on with a delay t_d (time to prepare for crossing). During the period t_c, only one signal (external event) Btn can be received.

Thus, for traffic control system temporal state diagram of timed Moore FSM can be built (Fig. 5).

Fig. 6 shows the timing diagram of Moore FSM for traffic light control system in the daytime cycle of the work, which is essentially is specification of the designed device.

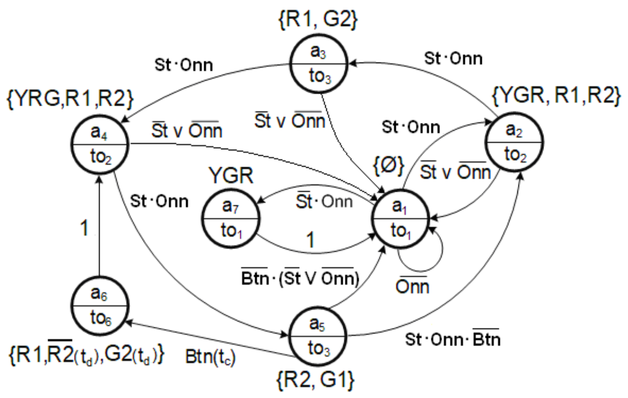

Figure 5 – State diagram of the Moore timed FSM for traffic light control system

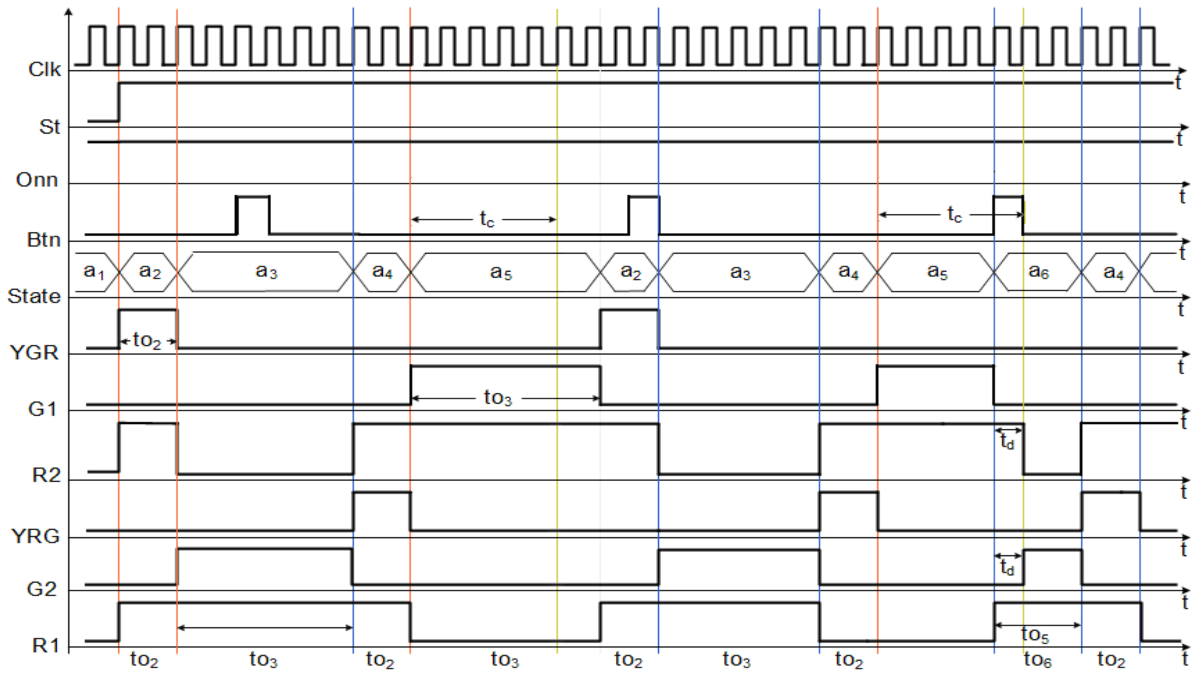

Figure 6 – Timing diagram of Moore FSM for traffic light control system

5 RESULTS

To confirm the reliability of obtained results, a two-process VHDL-model of the timed Moore control FSM was developed for the traffic light control system without putting the counter into a separate process. A fragment of the two-process VHDL-model is shown in listing 4.

All delays described in a separate package, which fragment is represented in the listing 5.

For verification of the presented model, as well as for synthesis and implementation, the Xilinx ISE 14.7 system was used. Behavioral and post-implementation simulations for initial description were performed on the CPLD XC9572XL-10-TQ100 (Post-Fit Simulation) and on the FPGA XC3S500E-5fg320 (Post-Place & Route Simulation). The clk period for simulation is 100 ns. Synthesis report of the latch-triggers absence was analyzed. Timing diagram of behavioral simulation, which illustrates a fragment associated with reaction to event B_{tn}, is shown in the Fig. 7.

The fragment of timing diagram after implementation on the XC3S500E-5fg320, corresponding to the reaction to the event Btn is shown in fig. 8. Delays (lags, control signals overlapping) are fractions of a percent of the clk period. For example, the intersection of green with red R2-> G2 is 0.59 ns (0.59%). Also, during the simulation, short-term pulses arise: signal R2 has a zero pulse with duration 2.06 ns, G2 has a one pulse with duration 1.31 ns, R2 has a one pulse with duration 1.14 ns, and G2 has a zero pulse with duration 1.26 ns. They constitute approximately 1.3-2% of the clk period. All delays and short-term pulses have a short period and absolutely not critical in terms of their influence on the operation of the FSM.

Listing 4. A fragment of two-process VHDL model of control FSM

```
-- tfsm_two_proc_next_count.vhd
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use work.tfsm_pack.all;
entity tfsm_two_proc_next_count is
        port ( clk, reset : in std_logic; Onn, St : in std_logic; Btn
: in std_logic;
G1, YGR, YRG, R1 : out std_logic; G2, R2 : out std_logic);
end;
architecture beh of tfsm_two_proc_next_count is
        type state_type is ( a1, a2, a3, a4, a5, a6, a7 );
        signal state, next_state : state_type;
        signal  count,  next_count  :  std_logic_vector  (
count_length - 1 downto 0 );
begin
        -- state synchronization
        process (Clk, Reset)
  begin
    if Reset = '1' then
      state <= a1;
        count <= ( others => '0' );
    elsif rising_edge(Clk) then
      state <= next_state;
        count <= next_count;
    end if;   end process;
  process (state, Onn, St, Btn, count )
   begin
        next_count <= (others => '0');
        case state is
                when a1 =>
                when a5 =>
                        if Btn = '1' and count >= con-
straint_a5_L - 1 and count < constraint_a5_H then
                                next_state <= a6;
                        elsif count < T3 - 1 then
                                next_state <= state;
                                next_count <= count + 1;
                        elsif Onn = '0' then
                                next_state <= a1;
                        elsif St = '0' then
                                next_state <= a1;
                        else    next_state <= a2;
                        end if;
                when a7 =>
                        if count < T1 - 1 then
                                next_state <= state;
                                next_count <= count + 1;
                        else    next_state <= a1;
                        end if;
                when others =>
                        next_state <= a1;
                        next_count <= ( others => '0' );
        end case;   end process;
        G1 <= '1' when  ( state = a5 ) else '0';
YGR <= '1' when ( ( state = a2 ) or ( state = a7 ) ) else '0';
        YRG <= '1' when  ( state = a4 )  else '0';
        R1 <= '1' when  ( ( state = a2 ) or ( state = a3 ) or ( state
= a4 ) or ( state = a6 ) )  else '0';
        G2 <= '1' when  ( state = a3 ) or ( state = a6 and count
>= output_delay_G2 )  else '0';
R2 <= '1' when      ( ( state = a2 ) or ( state = a4 ) or ( state = a5
) ) or ( state = a6 and count < output_delay_G2 ) else '0';
end architecture beh;
```

Listing 5. A fragment of the package work.tfsm_pack.all

```
        library ieee;
        use IEEE.std_logic_1164.all;
        package tfsm_pack is
                constant count_length : integer := 6;
        constant  T1  :  std_logic_vector  (  count_length  -  1
downto 0 ) := "000001";
                -- T1=1
        constant   constraint_a5_H   :   std_logic_vector   (
count_length - 1 downto 0 ) := "101000"; -- a5_H =40
        end tfsm_pack;
```

The fragment of simulation timing diagram after implementation on the XC9572XL-10-TQ100, which corresponds to the reaction to the event Btn, is represented in fig. 9.

Switching delay is 0 ns. Single short-term pulses do not occur. Thus, when implementing the device on FPGA and CPLD, its operation must comply with the original description (specification).

As a result of the synthesis of the device on the FPGA XC3S500E-5fg320 and CPLD XC9572XL-10-TQ100, CAD tool identified an FSM with 7 states, which codes are the same for both chips.

Synthesis results: FSM <state/FSM> on signal <state[1:3]> with user encoding**.** State codes: a1 (000), a2 (001), a3 (010), a4 (011), a5 (100), a6 (101), a7 (110).

The expected minimum number of triggers is 9: 3 triggers for encoding 7 states, 6 triggers for the counter (since the maximum timeout is to_3 = 45). RTL schematic report for both chips confirmed this. However, in the synthesis report for FPGA, 11 triggers appear, 2 of which are used due to the peculiarities of the technological implementation of the RTL circuit on FPGA, which is confirmed by the report in the technology schematic format. Latch triggers are absent in the report. To implement functions of transitions and outputs, combinational circuits were synthesized. 218 combinational elements were synthesized on the XC9572XL-10-TQ100, and 69 combinational elements – on the XC3S500E-5fg320. Table 1 shows some totals from the synthesis report.

Table 1 – Synthesis results of the FSM based on FPGA and CPLD

PLD	Flip-Flops	Latches	BELS	Slices / LUTs	Macro-cells
FPGA XC3S500E-5fg320	11	0	69	33/62	–
CPLD XC9572XL-10-TQ100	9	0	218	–	15

For FPGA: minimum Clk period: 5.700 ns (Maximum Frequency: 175.434 MHz).

For CPLD: minimum Clk period: 6.600 ns (Maximum Frequency: 151.515 MHz).

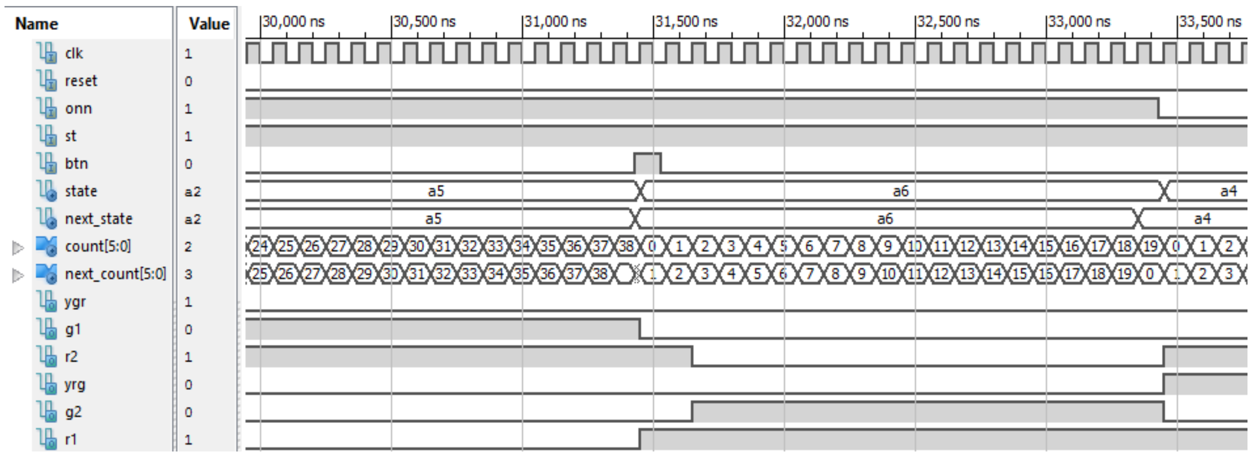

Figure 7 – Behavioral simulation of reaction to event Btn

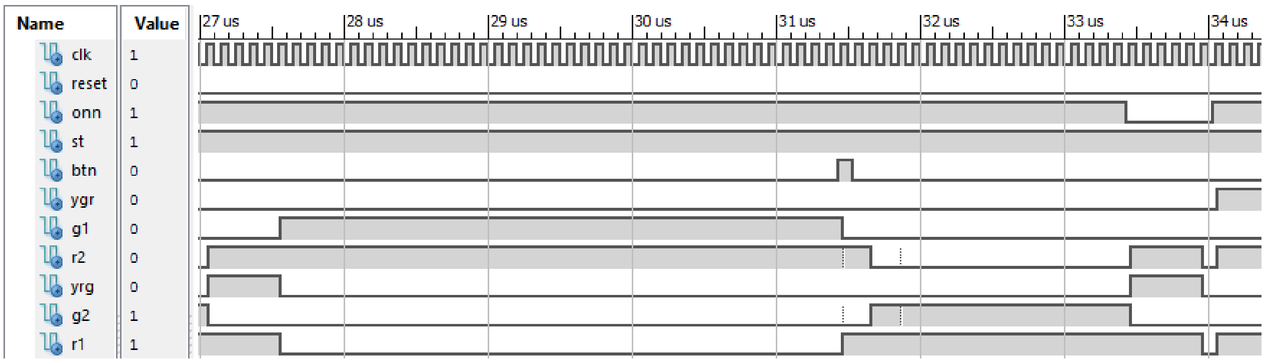

Figure 8 – Post-Place & Route Simulation of reaction to event Btn for XC3S500E-5fg320

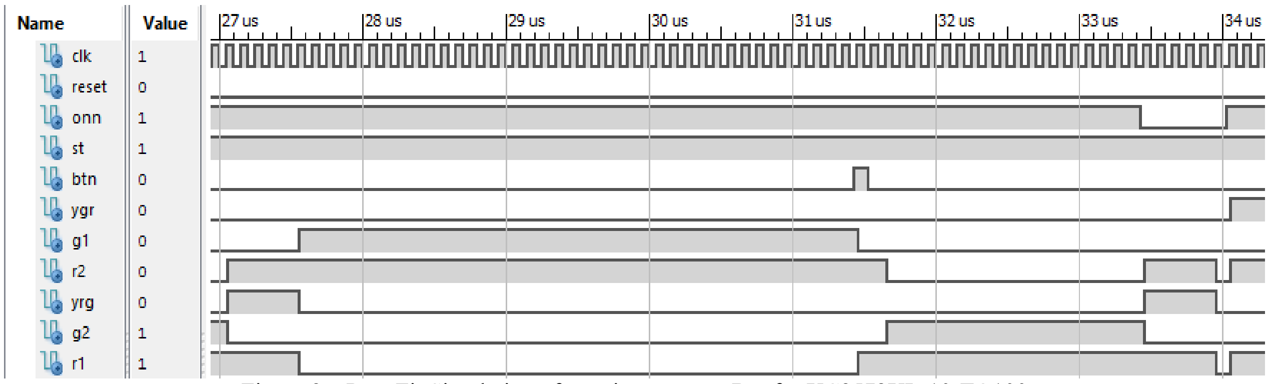

Figure 9 – Post-Fit Simulation of reaction to event Btn for XC9572XL-10-TQ100

6 DISCUSSION

The usage of Moore FSM models with timing delays is traditional for description logic control systems and has been worked out in sufficient detail for control programs in C language [1, 12]. Models of abstract timed FSMs with timeouts, timing constraints, and output delays are widely used in testing of timing parameters of microcontroller control systems and telecommunication protocols [5–10]. On the other hand, in [13–15], hardware implementations of timed FSM models in CAD tool using hardware description languages were proposed. But proposed FSM model' templates in hardware description languages did not take into account all parameters.

In this work, an attempt to define templates of language structures in VHDL for timed Moore control FSMs with the possibility of processing events, which are external to the control system, was made. Experiments' results of the synthesis of circuit implementations for timed control FSMs in PLDs confirmed that synthesizers in CAD tools (in particular, Xilinx ISE) interpret different language constructs of VHDL differently from the point of timing parameters implementation, although results of behavioral simulation for them coincide with specification. On the example of simulation (after implementation) for circuit implementation on PLD of control FSM in logic control system, elements of the HDL template are proposed to use that correctly implement timing parameters defined in the specification.

The direction of further research may be the construction of HDL templates for models of timed event-driven FSM in logic control systems and test verification of these models.

CONCLUSIONS

The method for constructing a HDL description for real-time systems, which can be synthesized into a programmable logic device, was developed in the work.

Models of real-time systems, such as a state diagram, a timed FSM with many timers, a timed FSM with one timer, an extended timed FSM, a timed FSM with timeouts and timing constraints were considered in this work. Based on these models, a complete structural model of timed Moore FSM was proposed.

Features of hardware and software-hardware approaches of real-time systems' realization were analyzed. A method for using an additional counter to store the value of a timed variable and implementation of timed parameters during usage of hardware approach based on the structural model of a timed Moore FSM, were proposed.

An algorithm of the traffic light operation was analyzed. Based on this algorithm, the temporal Moore state diagram was developed for the traffic light control system. Based on the temporal state diagram, a two-process VHDL-model of the timed Moore FSM (without moving the counter to a separate process), was proposed and developed.

Verification, synthesis and implementation of the developed VHDL model using Xilinx ISE environment were performed. Synthesis and simulation before and after implementation was performed for CPLD XC9572XL-10-TQ100 and FPGA XC3S500E-5fg320. Synthesis and simulation results of the circuit after implementation confirm the operability and correctness of the developed VHDL model.

The scientific novelty of the work lies in the definition of the method for constructing the HDL description of real-time system, a characteristic feature of which is the synthesis of the description into a programmable logic device, which allows to increase the speed of developed devices by increasing the development time of the input-output interface.

The practical value of the work lies in the possibility of hardware implementation of real-time systems based on the model of a timed FSM, which gives the possible to increase the flexibility and speed of the developed systems.

ACKNOWLEDGEMENTS

The work was carried out within the framework of the state budget research topic of the Kharkov National University of Radioelectronics "Cyber-physical system – Smart cloud Traffic Control" (state registration number 0115U-000712 from 04.03.2015, order of the Ministry of Education of Ukraine № 105 from 09.02.15) on the basis of the Design automation department of KNURE.

Authors are grateful to prof. Vladimir Hahanov for useful comments during preparation of this work.

REFERENCES

1. Shalyto A. A. Software Automation Design: Algorithmization and Programming of Problems of Logical Control, *Journal of Computer and System Sciences International*, 2000, Vol. 39, No. 6, pp. 899–916.
2. Harel D. Statechars: a Visual Formalism for complex systems, *Science of Computer Programming*, 1987, Vol. 8, pp. 231–274.
3. Alur R., Dill D. L. A theory of timed automata, *Theoretical Computer Science*, 1994, Vol. 126, No. 2, pp. 183–235.
4. Alur R. Timed automata, *Proceedings of the 11th International Conference on Computer Aided Verification, CAV'99, Trento, Italy, July 6-10, 1999*. Springer, 1999, pp. 8–22.
5. Merayo M. G., Núñez M., Rodríguez I. Formal testing from timed finite state machines, *Computer Networks*. – 2008. – Vol.52(2). – P. 432-460. DOI 10.1007/978-3-642-05031-2_5.
6. Merayo M. G., Núñez M., Rodríguez I. Extending efsms to specify and test timed systems with action durations and timeouts, *IEEE Transactions on Computers,* 2008, Vol. 57 (6), pp. 835–844. DOI 0018-9340/08/25.00
7. Zhigulin M., Yevtushenko N., Maag S., Cavalli A. R. FSM-Based Test Derivation Strategies for Systems with Time-Outs, *Proceedings of the* 11th International Conference on Quality Software *(QSIC 2011),* Madrid, 2011, pp. 141–149. DOI:10.1109/QSIC.2011.30.
8. El-Fakih K., Yevtushenko N., Simão A. A practical approach for testing timed deterministic finite state machines with single clock, *Science of Computer Programming. Elsevier*, 2014, Vol. 80, pp. 343–355. DOI: 0.1016/j.scico.2013.09.008.
9. Bresolin D., El-Fakih K., Villa T., Yevtushenko N. Deterministic Timed Finite State Machines: Equivalence Checking and Expressive Power, *Intern Conf. GANDALF*, 2014, pp. 203–216. DOI: 10.4204/EPTCS.161.18.
10. Bresolin D., Tvardovskii A., Yevtushenko N., Villa T., Gromov M. Minimizing Deterministic Timed Finite State Machines, *In 14th IFAC Workshop on Discrete Event Systems WODES 2018. – IFAC-PapersOnLine*, 2018, Vol. 51, Issue 7, pp. 486-492.
11. Adamski M., Barkalov Al., and Wegrzyn M. (Eds) Design of Digital Systems and Devices. Berlin, Springer-Verlag, 2011, 366 p. ISBN 978-3-642-17545-9.
12. Shamgunov N., Korneev G., Shalyto A. State Machine Design Pattern, *Shot communication papers conference proceedings of 4-th International Conference .NET Technologies in Central Europe 2006, May 29 – June 1, 2006, Net Technologies University of West Bohemia*, pp. 51–57. DOI: 10.1155/2018/4094951
13. Haskell R., Hanna D. Digital Design Using Digilent FPGA Boards – VHDL / Active-HDL Edition. LBE Books Rochester Hills, MI, 2009, 381 p. ISBN 10: 0980133785 / ISBN 13: 9780980133783.
14. Pedroni, V. A. Finite state machines in hardware : theory and design (with VHDL and SystemVerilog). Cambridge, MA: MIT Press., 2013, 338 p. ISBN: 0884174985574 .
15. ISE In-Depth Tutorial. UG695 (v14.1) April 24, 2012 [Electronic resource] / www.xilinx.com. Access mode: https://www.xilinx.com/support/documentation/sw_manuals/ xilinx14_1/ise_tutorial_ug695.pdf.
16. Miroschnyk M. , Shkil A., Kulak E., Rakhlis D., Filippenko I., Hoha M., M. Malakhov, V. Serhiienko Design of real-time logic control system on FPGA, *Proceedings of 2019 IEEE East-West Design & Test Symposium (EWDTS'19), September 13–16, Batumi, Georgia, 2019*, P. 488–491. DOI: 10.1109/TCSII.2018.2875589.

Received 11.02.2020.
Accepted 10.03.2020.

DOI 10.15588/1607-3274-2020-2-14

УДК 681.326

ПРОЕКТУВАННЯ ЧАСОВИХ АВТОМАТІВ З ВИКОРИСТАННЯМ ШАБЛОНІВ АВТОМАТА МУРА

Мірошник М. А. – д-р техн. наук, професор, професор кафедри спеціалізованих комп’ютерних систем Українського державного університету залізничного транспорту, Харків, Україна.

Шкіль А. С. – канд. техн. наук, доцент кафедри автоматизації проектування обчислювальної техніки Харківського національного університету радіоелектроніки, Харків, Україна.

Кулак Е. М. – канд. техн. наук, доцент кафедри автоматизації проектування обчислювальної техніки Харківського національного університету радіоелектроніки, Харків, Україна.

Рахліс Д. Е. – канд. техн. наук, доцент кафедри автоматизації проектування обчислювальної техніки Харківського національного університету радіоелектроніки, Харків, Україна.

Мірошник А. М. – аспірант кафедри автоматизації проектування обчислювальної техніки Харківського національного університету радіоелектроніки, Харків, Україна.

Малахов Н. В. – магістр кафедри автоматизації проектування обчислювальної техніки Харківського національного університету радіоелектроніки, Харків, Україна.

АНОТАЦІЯ

Актуальність роботи полягає в розвитку методів автоматизованого проектування автоматних пристроїв логічного керування реального часу шляхом розробки єдиного шаблону в синтезованих підмножині мови опису апаратури в стилі автоматного програмування з реалізацією на апаратній платформі ПЛІС (FPGA, CPLD).

Метою роботи є розробка принципів побудови моделей тимчасових керуючих автоматів на мові опису апаратури VHDL. В роботі вирішена задача побудови шаблону опису моделей тимчасових керуючих автоматів Мура на VHDL, автоматизований синтез і імплементація отриманої VHDL-моделі в ПЛІС (FPGA, CPLD) з використанням Xilinx ISE і подальший аналіз отриманої схемної реалізації на предмет дотримання значень часових параметрів схеми після імплементації.

Метод. Реалізація в конструкціях мови VHDL параметрів моделей тимчасових автоматів в системах логічного управління. Розробка конструкцій мови VHDL для реалізації часових параметрів моделей тимчасових автоматів, які забезпечують коректний автоматизований синтез і імплементація цих моделей в ПЛІС (FPGA, CPLD) з використанням інструментальних засобів САПР Xilinx ISE.

Результати. Синтез і імплементація запропонованих шаблонів VHDL-моделей тимчасових керуючих автоматів Мура в системах логічного управління інструментальними засобами автоматизованого проектування XILINX ISE підтвердили отримання ненадлишкових схемних структур в ПЛІС (FPGA, CPLD), а моделювання після імплементації показало працездатність таких моделей.

Висновки. В роботі вирішена задача автоматизованого проектування тимчасових керуючих автоматів в системах логічного управління реального часу. Для вирішення даного завдання розроблені шаблони VHDL-моделей тимчасових керуючих автоматів Мура, що дало можливість реалізувати керуючі автомати з тимчасовими обмеженнями, таймаут і вихідними затримками. Автоматизований синтез і моделювання VHDL-моделей на основі розроблених шаблонів підтвердили працездатність і коректність запропонованих моделей.

Наукова новизна роботи полягає в подальшому розвитку методів побудови шаблонів HDL-моделей тимчасових керуючих автоматів Мура, що дало можливість реалізувати керуючі автомати з тимчасовими обмеженнями, таймаут і вихідними затримками, а також виконати їх коректний автоматизований синтез і моделювання.

Практична цінність отриманих результатів полягає в розробці процедур побудови VHDL-моделей тимчасових керуючих автоматів Мура в системах логічного управління реального часу, що дало можливість автоматизувати процес синтезу керуючих автоматів з урахуванням можливості обробки зовнішніх подій і реалізації довільних затримок для вихідних сигналів і збільшити гнучкість і швидкодія проектованих систем. Розроблені процедури можуть бути корисні проектувальникам часових керуючих автоматів в XILINX ISE.

КЛЮЧОВІ СЛОВА: кінцевий автомат з синхронізацією за часом, діаграма станів, мова опису апаратних засобів, шаблон автоматів, моделювання, синтез, реалізація, CAD, XILINX ISE.

УДК 681.326

ПРОЕКТИРОВАНИЕ ВРЕМЕННЫХ АВТОМАТОВ С ИСПОЛЬЗОВАНИЕМ ШАБЛОНА АВТОМАТА МУРА

Мирошник М. А. – д-р техн. наук, профессор, профессор кафедры специализированных компьютерных систем Украинского государственного университета железнодорожного транспорта, Харьков, Украина.

Шкиль А. С. – канд. техн. наук, доцент кафедры автоматизации проектирования вычислительной техники Харьковского национального университета радиоэлектроники, Харьков, Украина.

Кулак Э. Н. – канд. техн. наук, доцент кафедры автоматизации проектирования вычислительной техники Харьковского национального университета радиоэлектроники, Харьков, Украина.

Рахлис Д. Е. – канд. техн. наук, доцент кафедры автоматизации проектирования вычислительной техники Харьковского национального университета радиоэлектроники, Харьков, Украина.

Мирошник А. М. – аспирант кафедры автоматизации проектирования вычислительной техники Харьковского национального университета радиоэлектроники, Харьков, Украина.

Малахов Н. В. – магистр кафедры автоматизации проектирования вычислительной техники Харьковского национального университета радиоэлектроники, Харьков, Украина.

АННОТАЦИЯ

Актуальность работы состоит в развитии методов автоматизированного проектирования автоматных устройств логического управления реального времени путем разработки единого шаблона в синтезируемом подмножестве языка описания аппаратуры в стиле автоматного программирования с реализацией на аппаратной платформе ПЛИС (FPGA, CPLD).

DOI 10.15588/1607-3274-2020-2-14

Целью работы является разработка принципов построения моделей временных управляющих автоматов на языке описания аппаратуры VHDL. В работе решена задача построения шаблона описания моделей временных управляющих автоматов Мура на VHDL, автоматизированный синтез и имплементация полученной VHDL-модели в ПЛИС (FPGA, CPLD) с использованием Xilinx ISE и последующий анализ полученной схемной реализации на предмет соблюдения значений временных параметров схемы после имплементации.

Метод. Реализация в конструкциях языка VHDL параметров моделей временных автоматов в системах логического управления. Разработка конструкций языка VHDL для реализации временных параметров моделей временных автоматов, которые обеспечивают корректный автоматизированный синтез и имплементация этих моделей в ПЛИС (FPGA, CPLD) с использованием инструментальных средств САПР Xilinx ISE.

Результаты. Синтез и имплементация предложенных шаблонов VHDL-моделей временных управляющих автоматов Мура в системах логического управления инструментальными средствами автоматизированного проектирования XILINX ISE подтвердили получение неизбыточных схемных структур в ПЛИС (FPGA, CPLD), а моделирование после имплементации показало работоспособность таких моделей.

Выводы. В работе решена задача автоматизированного проектирования временных управляющих автоматов в системах логического управления реального времени. Для решения данной задачи разработаны шаблоны VHDL-моделей временных управляющих автоматов Мура, что дало возможность реализовать управляющие автоматы с временными ограничениями, таймаутами и выходными задержками. Автоматизированный синтез и моделирование VHDL-моделей на основе разработанных шаблонов подтвердили работоспособность и корректность предложенных моделей.

Научная новизна работы состоит в дальнейшем развитии методов построения шаблонов HDL-моделей временных управляющих автоматов Мура, что дало возможность реализовать управляющие автоматы с временными ограничениями, таймаутами и выходными задержками, а также выполнить их корректный автоматизированный синтез и моделирование.

Практическая ценность полученных результатов заключается в разработке процедур построения VHDL-моделей временных управляющих автоматов Мура в системах логического управления реального времени, что дало возможность автоматизировать процесс синтеза управляющих автоматов с учетом возможности обработки внешних событий и реализации произвольных задержек для выходных сигналов и увеличить гибкость и быстродействие проектируемых систем. Разработанные процедуры могут быть полезны проектировщикам временных управляющих автоматов в XILINX ISE.

КЛЮЧЕВЫЕ СЛОВА: конечный автомат с синхронизацией по времени, диаграмма состояний, язык описания аппаратных средств, шаблон автоматов, моделирование, синтез, реализация, CAD, XILINX ISE.

ЛІТЕРАТУРА / ЛИТЕРАТУРА

1. Shalyto A.A. Software Automation Design: Algorithmization and Programming of Problems of Logical Control / A.A. Shalyto // Journal of Computer and System Sciences International. – 2000. – Vol. 39, No. 6. – P. 899–916.
2. Harel D. Statechars: a Visual Formalism for complex systems / D. Harel // Science of Computer Programming. – 1987. – Vol. 8. – P. 231–274.
3. Alur R. A theory of timed automata / R. Alur, D.L. Dill // Theoretical Computer Science. – 1994. – V.126. – N 2. – P. 183–235.
4. R Alur. Timed automata // Proceedings of the 11th International Conference on Computer Aided Verification, CAV’99, Trento, Italy, July 6–10, 1999. – Springer, 1999. – P. 8–22.
5. Merayo M. G., Núñez M., Rodríguez I. Formal testing from timed finite state machines. // Computer Networks. – 2008. – Vol. 52(2). – P. 432–460. DOI 10.1007/978-3-642-05031-2_5.
6. Merayo M. G., Núñez M., Rodríguez I. Extending efsms to specify and test timed systems with action durations and timeouts. // IEEE Transactions on Computers. – 2008. – Vol.57(6). – P. 835–844. DOI 0018-9340/08/25.00
7. Zhigulin M. FSM-Based Test Derivation Strategies for Systems with Time-Outs / M. Zhigulin, N. Yevtushenko, S. Maag, A.R. Cavalli // Proceedings of the 11th International Conference on Quality Software (QSIC 2011), Madrid, 2011. – P. 141–149. DOI:10.1109/QSIC.2011.30.
8. El-Fakih K. A practical approach for testing timed deterministic finite state machines with single clock / K. El-Fakih, N. Yevtushenko, A. Simão // Science of Computer Programming. Elsevier. – 2014. – Vol. 80. – P. 343–355. DOI: 0.1016/j.scico.2013.09.008.
9. Bresolin D. Deterministic Timed Finite State Machines: Equivalence Checking and Expressive Power / D. Bresolin, K. El-Fakih, T. Villa, N. Yevtushenko // Intern Conf. GANDALF. – 2014. – P. 203–216. DOI: 10.4204/EPTCS.161.18.
10. Bresolin D. Minimizing Deterministic Timed Finite State Machines / D. Bresolin, A. Tvardovskii, N. Yevtushenko, T. Villa, M. Gromov // In 14th IFAC Workshop on Discrete Event Systems WODES 2018. – IFAC-PapersOnLine, 2018. – Vol. 51, issue 7. – P. 486–492.
11. Adamski M. Design of Digital Systems and Devices / Marian Adamski, Alexander Barkalov, and Marek Wegrzyn (Eds). – Berlin: Springer-Verlag, 2011. – 366 p. ISBN 978-3-642-17545-9.
12. Shamgunov N. State Machine Design Pattern / N. Shamgunov, G. Korneev, A. Shalyto // Shot communication papers conference proceedings of 4-th International Conference .NET Technologies in Central Europe 2006, May 29 – June 1 2006. – Net Technologies University of West Bohemia. – P. 51–57. DOI: 10.1155/2018/4094951
13. Haskell R. Digital Design Using Digilent FPGA Boards – VHDL / Active-HDL Edition / Richard E. Haskell, Darrin M. Hanna. – LBE Books Rochester Hills, MI, 2009. – 381 p. ISBN 10: 0980133785 / ISBN 13: 9780980133783.
14. Pedroni, V. A. Finite state machines in hardware : theory and design (with VHDL and SystemVerilog) / Volnei A. Pedroni. – Cambridge, MA : MIT Press, 2013. – 338 p. ISBN: 0884174985574 .
15. ISE In-Depth Tutorial. UG695 (v14.1) April 24, 2012 [Electronic resource] / www.xilinx.com – Access mode: https://www.xilinx.com/support/documentation/sw_manuals / xilinx14_1/ise_tutorial_ug695.pdf.
16. Design of real-time logic control system on FPGA / [M. Miroschnyk, A. Shkil, E. Kulak et al] // Proceedings of 2019 IEEE East-West Design & Test Symposium (EWDTS’19), September 13–16, Batumi, Georgia, 2019. – P. 488–491. DOI: 10.1109/TCSII.2018.2875589.

© Miroshnyk M. A., Shkil O. S., 2020
DOI 10.15588/1607-3274-2020-2-14