УДК 681.31 **DOI:** 10.18664/ikszt.v29i1.300988

МІРОШНИК М. А., д.т.н., професор, професор кафедри теоретичної та прикладної системотехніки, Харківський Національний Університет імені В. Н. Каразіна,
ПАХОМОВ Ю. В., к.т.н., доцент кафедри комп'ютерних наук та інформаційних технологій, Харківський національний університет міського господарства імені О. М. Бекетова,
ПШЕНИЧНИЙ К. Ю., аспірант кафедри автоматизації проектування обчислювальної техніки, Харківський національний університет радіоелектроніки,
ШАФРАНСЬКИЙ А. В., аспірант кафедри теоретичної та прикладної системотехніки, Харківський Національний Університет імені В. Н. Каразіна

Асерційна верифікація моделей пристроїв реального часу з недетермінованими зовнішніми подіями

Представив д.т.н., професор Приходько С.І.

Запропоновано метод верифікації моделей пристроїв реального часу з обробкою зовнішніх подій із недермінованою тривалістю, що описані з використанням мов опису апаратури (Hardware Description Language, HDL). У методології використано апарат асерцій для опису темпоральної природи вищезазначених моделей.

Ключові слова: автоматизація проєктування*, моделювання,* верифікація, системи реального часу, події, темпоральний граф переходів.

Вступ

Логічні системи керування є важливим елементом будь-якої цифрової системи. Такі системи використовують двійковий алфавіт для визначення поведінки блока керування. Відомо, що шаблон кінцевого автомата (Finite State Machine, FSM) є популярною моделлю для таких систем. Слід зазначити, що кінцевий автомат є математичною абстракцією, яку можна подати різними способами, такими як таблиця переходів станів, діаграма станів (граф переходів), граф-схема алгоритму тощо. При використанні шаблону автомата важливо розуміти, що поведінка цільової системи залежить від подій, які відбуваються в зовнішньому середовищі. У кінцевому автоматі події використовуються для моделювання певних зовнішніх дій, які викликають переходи з одного стану в інший. До подій належать сигнали, виклики, певний проміжок часу або зміна стану. Події можуть бути синхронними чи асинхронними; їх вираження у кінцевій моделі – одна з ключових частин процесу проєктування. Окрім того, існує клас пристроїв, логіка яких залежить не тільки від часу, у який надходять ці події, але і часових характеристик цих подій. Прикладом такої події є затискання кнопки, що призводить до вимикання пристрою.

Відомо, що верифікація цифрових дизайнів, розроблених на базі Application-Specific Integrated Circuit (ASIC), Field Programmable Gate Array (FPGA), System on Chip (SoC) займає до 70 % загального часу проєктування [1]. Як результат, до 80 % кодової бази цифрового дизайну займає код тестування (testbench). Зменшення цих двох параметрів зменшує час випуску пристрою на ринку (time-to-market) і є одним із головних завдань галузі автоматизації проєктування обчислювальної техніки.

Аналіз наявних рішень

У роботі [4] розглянуто узагальнену модель часового автомата з тайм-аутами і часовими обмеженнями та вихідними затримками. Узагальнена модель структурного темпорального керуючого автомата $Y(t) = g(X(t), Z(t), T)$, $Z(t+1) = f(X(t), Z(t), T)$, де X – множина вхідних сигналів, Z – множина

внутрішніх змінних, яка визначає стан автомата, Y – множина вихідних сигналів, t – машинний час, що визначається в автоматичних тактах, d – функція виходів, f – функція переходів структурного автомата. $T = \{T_c, T_o, T_d\}$ є множиною часових параметрів автомата, де t_c – часові обмеження, t_o – вхідні тайм-аути, t_d – вихідні затримки.

Модель автомата реального часу є способом опису систем реального часу, який був введений у роботі [2]. Графова модель автомата доповнюється дискретною множиною таймерів, що набувають натуральних значень. Як і в класичного графа, вершини називаються «станами», а дуги «переходами». Кожен таймер обнуляється в момент переходу і збільшується з кожним тактом автомата. З кожним переходом пов'язано часове обмеження (clock constraint), що вказує, що перехід можливий лише в тому випадку, якщо поточні значення таймера відповідають заданому обмеженню. Кожна позиція також пов'язана з обмеженнями таймерів – інваріантами; система може перебувати в цій позиції до тих пір, поки виконується її інваріант.

Питання моделювання та верифікації цифрових дискретних подієвих систем розглядається в роботі [3], де введено зв'язок між методологією Event-B і автоматизованим моделюванням. Цей документ важливий для верифікації моделей на основі автоматів.

У роботі [4] подано спосіб підвищення якості верифікації часових автоматів шляхом вираження часових переходів за допомогою розширення автоматного шаблону додатковими асерційними конструкціями мовою System Verilog Assertions (SVA). Показано, як такий підхід скорочує час пошуку помилки оператора в SystemVerilog коді.

У роботі [5] розглянуто питання логічних систем з обробкою зовнішніх подій із мінімальною тривалістю, запропоновано новий вид переходу, який враховує не тільки зовнішню подію, але і її тривалість. Технічно це реалізується за допомогою додаткового таймера. Розглянуто різні варіанти поведінки автомата залежно від тривалості зовнішньої події.

У роботі [6] розглянуто принципи побудови систем верифікації та вбудованої самодіагностики систем на кристалі (SoC), що містять систему обробки PS (processing system) і програмовану логіку PL (programmable logic). Реалізація моделі пристрою керування світлофором виконана мовою програмування C з використанням стека інструментальних засобів САПР Vivado/Vitis/Vitis HLS. Неруйнівний діагностичний експеримент також реалізується за допомогою мови C шляхом повного обходу всіх можливих дуг графа, який описує систему логічного керування.

Мета дослідження

Метою дослідження є зменшення часу верифікації за рахунок введення надлишковості у HDL-модель пристрою реального часу з обробкою недетермінованих зовнішніх подій – механізму асерцій [7-10], який дає змогу явно виражати характеристики дизайну та аналізувати помилки HDL-коду під час моделювання.

Постановка задачі

Розробити модель вираження темпоральних властивостей моделей пристроїв реального часу з недетермінованими зовнішніми подіями шляхом розширення HDL-опису пристрою асерційними конструкціями.

Основний матеріал

Візуальним описом моделі часового автомата і його повною математичною моделлю є темпоральний граф переходів. Такий граф розширюється таймером, який реалізує затримки у станах. Таймер використовується для перебування у стані впродовж певної кількості тактів синхросигналу. На рис. 1 наведено приклад автомата з часовими переходами.

Як уже було зазначено, існує категорія пристроїв, які відрізняються реакцією на події, що тривають у часі. Для цієї групи пристроїв важливі часові параметри зовнішньої події. Часові обмеження події – це значення $t_c(X(i)) = [c1]$, що вказує на мінімальний час тривалості певного вхідного сигналу. У прикладі з кнопкою блокування мобільного телефону $t_c = 3$ (для зручності значення подано в секундах). Це обмеження виражено на темпоральному графі у формі переходу. При визначенні цього переходу необхідно враховувати значення таймера зі значенням $t_c(X(i))$.

На рис. 2 наведено приклад такого переходу. Як і у випадку темпорального переходу, в умові береться до уваги стан таймера. Головною відмінністю є те, що таймер тепер має нижню границю, тобто для успішної зміни стану a2/a3 необхідно, аби сигнал події evnt був активним (набував значення високого рівня, тобто ‘1’) і таймер більшим за значення tc(evnt).

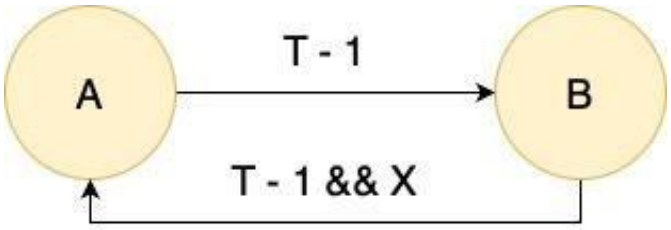

Рис. 1. Приклад темпорального графа переходів

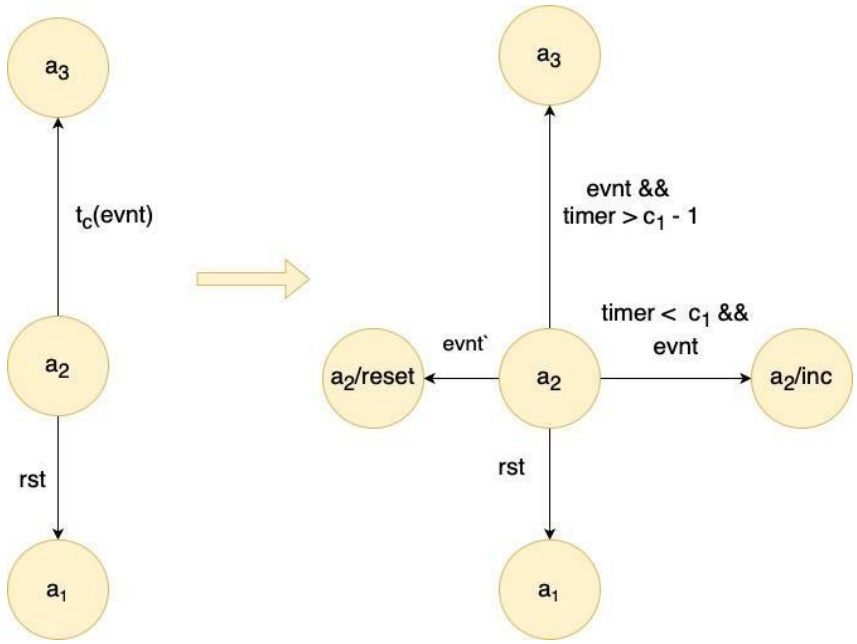

Рис. 2. Приклад темпорального графа переходів із зовнішньою подією з мінімальною тривалістю

У роботі [5] розглянуто різні випадки тривалості події та відповідні реакції автомата на них: 1) зовнішня подія триває рівно стільки часу, скільки вказано у специфікації; 2) зовнішня подія триває менше необхідного часу; 3) зовнішня подія триває більше необхідного часу; 4) зовнішня подія не відбувається. Очевидно, що HDL-опис має враховувати всі можливі сценарії. Окрім того, ці поведінки мають бути покритими тестовими сценаріями testbench у процесі верифікації.

Для ілюстрації запропонованих методів будемо використовувати модель модуля збереження енергії. Модуль працює у двох режимах: стандартному і енергозбереження. Вхідний алфавіт складається з двійкових сигналів X = {*onn*, *evnt*}, де *onn* – сигнал для вмикання алгоритму енергозбереження, *evnt* – сигнал сповіщення про те, що сталася зовнішня подія, і система має вийти з режиму енергозбереження. Вихідні двійкові сигнали налаштовані так: Y = {*save*}, де *save* означає вхід у режим енергозбереження.

Виділимо стани автомата, що описує функціональність модуля:

– стан a_1 – модуль працює в режимі обходу, тобто зовнішні події не відстежуються;

– стан a_2 – розрахунок режиму енергозбереження: якщо протягом фіксованого часу не відбувається жодних подій, автомат переходить у стан a_3;

– стан a_3 – режим енергозбереження.

Алгоритм роботи модуля збереження енергії досить простий. Щоразу, коли цей блок вмикається сигналом onn, він починає моніторинг вхідного сигналу *evnt*. Якщо сигнал *evnt* не з'являється протягом фіксованого часу, система має перейти в режим енергозбереження (збереження вихідного сигналу встановлено на високий рівень). Система виходить із режиму збереження в разі встановлення сигналу *evnt* або вимкнення модуля через вхідний сигнал *onn* із подальшим зняттям сигналу збереження. На рис. 3 зображено граф переходів автомата Мура такого пристрою.

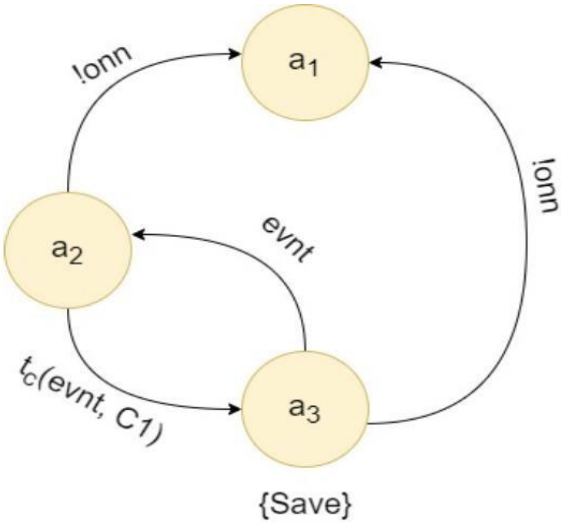

Рис. 3. Граф переходів кінцевого автомата Мура для модуля енергозбереження

На рис. 4 зображено процес переходів такого автомата мовою Verilog.

System Verilog Assertions (SVA) є підмножиною мови Verilog, що дає змогу виражати особливості дизайну за допомогою спеціальних конструкцій. Слід зазначити, що всі конструкції SVA ігнорують під час синтезу, а отже, ніяк не впливають на кінцеву схему.

Темпоральні властивості будь-якої HDL-моделі можуть бути виражені за допомогою трьох конструкцій SVA:

– *sequence* – використовується для подання послідовності подій моделювання в часі, який виражається циклами синхросигналу;

– *property* – конструкція, що об'єднує різні sequence з використанням операторів SVA;

– *assert* – подає асерційну точку, яка використовується для перевірки *property* під час моделювання або формальної верифікації.

```
always @(state, evnt, onn, count)
  case (state)
    a1: if (onn) begin
          next_state = a2;
        end
       else begin
          next_state = a1;
       end

    a2: if (!onn) begin
          next_state = a1;
        end
        else if (count >= C1 - 1 && !evnt) begin
          next_state = a3;
          next_count = 3'd0;
        end
        else if (!evnt) begin
          next_state = a2;
          next_count = count + 1'b1;
        end
        else begin
          next_state = a2;
          next_count = 3'd0;
        end

    a3: if (!onn) begin
          next_state = a1;
        end
        else if (evnt) begin
          next_state = a2;
        end else begin
          next_state = a3;
        end

    default: begin
      next_state = a1;
      next_count = 3'd0;
    end
  endcase
```

Рис. 4. Verilog опис процесу переходів модуля енергозбереження

Опишемо можливі сценарії поведінки дизайну залежно від тривалості зовнішньої події за допомогою вищезазначених конструкцій. На рис. 5 наведено конструкцію *property*, яка описує передумову обчислення режиму енергозбереження – модуль ввімкнений, зовнішня подія відсутня, сигнал *save* має низький логічний рівень.

```
sequence no_event;
  (onn && !evnt && !save);
endsequence
```

Рис. 5. SVA-опис передумови обчислення режиму енергозбереження

Наступним кроком необхідно описати можливі ситуації поведінки дизайну. На рис. 6 наведено опис поведінки, коли зовнішня подія триває необхідну кількість тактів – у цьому випадку 5 тактів.

```
property exact_match;
  disable iff(rst)
  @(posedge clk) no_event |=> !evnt [*5] |=> save;
endproperty
```

Рис. 6. SVA-опис поведінки, коли зовнішня подія триває необхідну кількість тактів

На наступному такті сигнал *save* повинен мати високий логічний рівень. Часову діаграму, яка описує таку ситуацію, наведено на рис. 7. Жовтою стрілкою позначено момент початку обчислення властивості – виконання умови *onn && !evnt && !save*, зеленою – момент успішного обчислення властивості – сигнал *save* набуває значення логічної одиниці. Слід зазначити, що це опис ситуації, коли зовнішня подія триває більше необхідної кількості тактів.

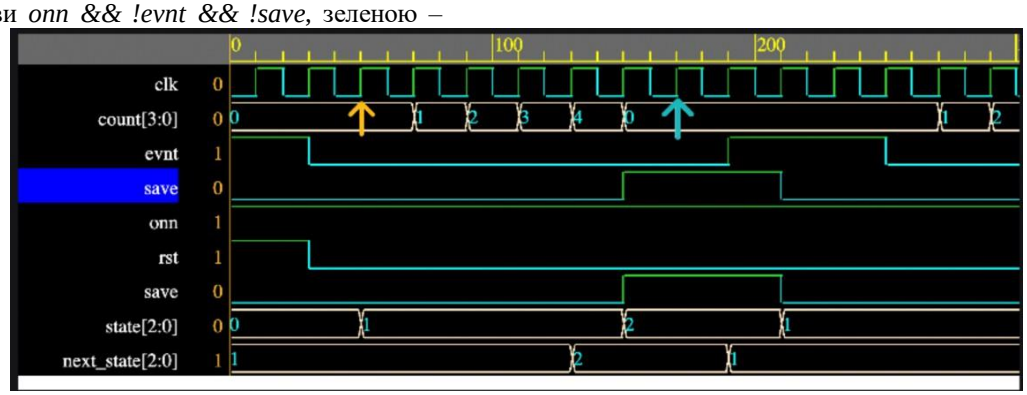

Рис. 7. Часова діаграма обробки зовнішньої події, яка триває необхідну кількість тактів

Аналогічно опишемо ситуацію, коли зовнішня подія не задовольняє часові вимоги. На рис. 8 наведено SVA-код опису такого сценарію. Відповідна часова діаграма зображена на рис. 9. Червоною стрілкою позначено момент початку обчислення властивості – виконання умови *onn && !evnt && !save*, жовтою – момент, коли сигнал *evnt* набуває значення логічної одиниці, зеленою – момент успішного обчислення властивості: сигнал *save* набуває значення логічного нуля.

```
property smaller_match;
  disable iff(rst)
  @(posedge clk) no_event |=> evnt |=> !save;
endproperty
```

Рис. 8. SVA-опис поведінки, коли зовнішня подія триває менше необхідної кількості тактів

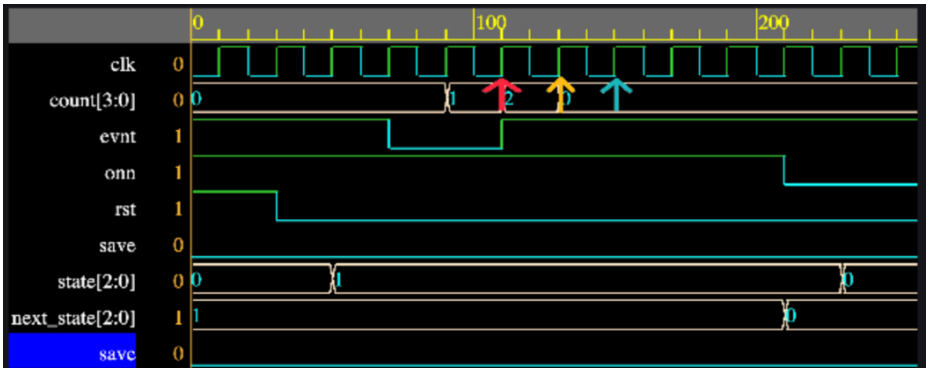

Рис. 9. Часова діаграма обробки зовнішньої події, яка триває менше необхідної кількості тактів

Останній варіант поведінки – модуль ввімкнений (сигнал *onn* має рівень логічної одиниці), сигнал *evnt* має високий логічний рівень увесь час

(зовнішня подія не відбувається взагалі). SVA-опис такої поведінки подано на рис. 10.

```
property event_all_time;
  disable iff(rst)
  @(posedge clk) onn and evnt |=> !save;
endproperty
```

Рис. 10. SVA-опис поведінки, коли зовнішня подія не задовольняє часові вимоги

Відповідна часова діаграма зображена на рис. 11. Червоними стрілками позначено момент початку обчислення властивості – виконання умови *onn && evnt*, жовтими – моменти успішного обчислення властивості: сигнал *save* набуває значення логічного нуля.

Рис. 11. Часова діаграма роботи автомата, коли зовнішня подія не відбувається

Важливим аспектом написання верифікаційного коду з використанням асерцій є визначення поведінки середовища моделювання в разі порушення визначених властивостей. Окрім того, враховуючи імплікаційну природу властивостей, необхідно знати, чи активує тестовий код (testbench) такі сценарії. Для вирішення цих завдань у SVA використовують конструкції *assert* та *cover* відповідно. На рис. 12 наведено фрагмент використання цих конструкцій для раніше розглянутих властивостей.

```
assert property (exact_match); // асерційна точка
cover property (exact_match); // точка функціонального покриття
```

Рис. 12. Використання операторів *assert property* та *cover property* для визначення точок асерції та функціонального покриття

На рис. 13 показано помилку оператора в HDL-коді автомата і відповідну реакцію середовища моделювання. Після запуску тестів з'являються помилкові повідомлення, які свідчать про порушення раніше описаних властивостей – у цьому випадку *exact_match* з зазначенням початкової та кінцевої точки обчислення властивості.

```
assign save = state == a2; // Помилка в HDL коді
```

```
Error: ASRT_0005 design.sv(87): Assertion FAILED at time: 150ns, scope: testbench.inst, start-time: 30ns
Error: ASRT_0005 design.sv(93): Assertion FAILED at time: 170ns, scope: testbench.inst, start-time: 150ns
Error: ASRT_0005 design.sv(93): Assertion FAILED at time: 190ns, scope: testbench.inst, start-time: 170ns
Error: ASRT_0005 design.sv(93): Assertion FAILED at time: 210ns, scope: testbench.inst, start-time: 190ns
Error: ASRT_0005 design.sv(93): Assertion FAILED at time: 230ns, scope: testbench.inst, start-time: 210ns
```

Рис. 13. Помилка HDL-оператора та реакція середовища моделювання

Отже, значно зменшується час пошуку помилкових HDL-операторів і конструкцій. Окрім того, за рахунок оператора *cover property* проєктувальник має змогу бачити, які сценарії поведінки покрито тестами.

Обговорення результатів

У статті розглянуто питання скорочення часу верифікації цифрових приладів шляхом введення додаткових конструкцій в HDL-код опису пристрою. На практиці показано використання мови SVA (підмножини мови Verilog) для опису асерційних конструкцій. Слід зазначити, що запропоновані підходи також можна реалізовувати при діагностиці та верифікації автоматних моделей, написаних із використанням мов програмування C/C++, які також мають підтримку асерційних конструкцій. На рис. 14 показано приклад використання оператора *assert* при написанні тестового коду пристрою логічного керування світлофором мовою С. Особливу актуальність цей підхід має при реалізації пристроїв логічного керування в класичних SoC, наприклад сімейства ZYNQ-7000 фірми Xilinx Inc [6]. Реалізація моделі пристрою керування мовою програмування С з використанням асерційних конструкцій на базі стека інструментальних засобів САПР Vivado/Vitis/Vitis HLS дає змогу виконати наскрізне проєктування та верифікацію як апаратної частини програмованої логіки (PL), так і процесорного програмного забезпечення в частині PS.

```
// Simulate transition to A7
assert(fsm.get_current_state() == fsm_states::A1);
for (int i = 0; i < T1; ++i) {
    street_fsm_controller( input_signals: OnSignal,  &: control_output,  extClock: false);
    street_fsm_controller( input_signals: OnSignal,  &: control_output,  extClock: true);
}
assert(fsm.get_current_state() == fsm_states::A7);
assert(fsm.get_control_signals() == YGR);
```

Рис. 14. Використання оператора *assert* у тестовому коді мовою С

Висновки

Запропоновано модель верифікації часових автоматів із зовнішніми подіями з недетермінованою тривалістю шляхом введення додаткових асерційних конструкцій, які описують можливі сценарії поведінки автомата залежно від тривалості зовнішньої події. Асерційні точки є потужним інструментом, які спрощують пошук помилкового HDL-оператора під час верифікації проєкту. Окрім того, показано, що асерції можна використовувати для підвищення якості тестів (functional coverage), зменшуючи час верифікації і, як наслідок, час випуску готового продукту на ринок (time-to-market).

Запропоновані теоретичні викладки проілюстровано на прикладі розширення Verilog опису автомата модуля енергозбереження з використанням мови SVA.

Список використаних джерел

1. Bergeron Janick. Writing testbenches: functional verification of HDL models. Boston: Kluwer Academic Publishers. 2001. 354 c.

2. Design timed FSM with VHDL Moore pattern / M. A. Miroschnyk, A. S. Shkil, E. N. Kulak, D. Y. Rakhlis, A. M. Miroshnyk, N. V. Malahov. *Radio*

3. *Electronics, Computer Science, Control.* 2020. № 2(53). P. 137-148.

4. Iabah Al-Fedaghi. Modeling Physical. *Digital Systems: Formal Event-B vs. Diagrammatic Thinging Machine. International Journal of Computer Science and Network Security*. 2020. No. 20 (4). P. 208–220. hal-02614504, version 1 (20-05-2020).

5. Miroshnyk A. M., Kulak G., Pshenychnyi K. Y. Assertion Based Design of Timed Finite State Machine. *2021 IEEE East-West Design & Test Symposium (EWDTS)*. Batumi, Georgia. 2021. P. 1-4.

6. Temporal events processing models in finite state machines / M. A. Miroshnyk, S. I. Shmatkov, O. S. Shkil, K. Y. Pshenychnyi (2023). *Radio Electronics, Computer Science, Control.* (4), 49.

7. Проєктування та самодіагностика кіберфізичних пристроїв керування на платформі SoC / О. С. Шкіль, Д. Ю. Рахліс, І. В. Філіпенко, В. Р. Корнієнко. *Сучасний стан наукових досліджень та технологій в промисловості*. 2023. No 4 (26). С. 122–134.

8. Components of a complete assertions-based verification solution. Cadence. 2005. URL: www.cadence.com/products/functional_ver/abv_dt.aspx.

9. IEEE Std 1076-2002, IEEE Standard VHDL Language Reference Manual. 124 p.

10. IEEE Std 1364-2001, IEEE Standard for Verilog Hardware Description Language. Electrical and Electronics Engineers, Inc.USA. 2003. P. 1-23.

11. Хаханов В. І., Каминська М. А., Зайченко С. О. Верифікація цифрових пристроїв на основі використання аналіза тестопридатності та асерційних бібліотек (російською мовою). *АСУ та прилади автоматики*. Харків, 2007. № 140. С. 75–83.

Miroshnyk M. A., Shkil O. S., Pshenychnyi K., Shafranskyi A. Assertion based verification of real-time device with non-deterministic external events. A real-time device models with processing of external events with indefinite duration described using hardware description languages (HDLs) verification method, is proposed. The methodology uses the apparatus of assertions to describe the temporal nature of the above models. It is shown that the introduction of additional HDL structures allows to increase the level of testability and reduce the verification time without additional hardware, time and energy costs of the digital product. The effectiveness of the proposed approaches was illustrated on the example of the energy saving module model. The HDL code of the device has been extended with additional constructs in the System Verilog Assertions language to describe the nature of the model depending on an external event. Different scenarios of the device were considered depending on the duration of the external event with corresponding illustrations on time diagrams. It was also shown how the proposed methodology can be used in automatic models of digital devices described in the C language.

Keywords: *design automation, simulation, verification, real time systems, events, temporal state diagram.*

Мірошник Марина Анатоліївна, д.т.н., професор, професор кафедри теоретичної та прикладної системотехніки, Харківський Національний Університет імені В. Н. Каразіна. E-mail: m.miroshnyk@karazin.ua https://orcid.org/[0000000222312529].

Пшеничний Кирило Юрійович, аспірант кафедри автоматизації проектування обчислювальної техніки, АПОТ, ХНУРЕ, м. Харків, Україна. E-mail: kyrylo.pshenychnyi@nure.ua, https://orcid.org/[0009-0007-0799-6604].

Шафранський Андрій Вікторович, аспірант кафедри теоретичної та прикладної системотехніки, Харківський Національний Університет імені В. Н. Каразіна. E-mail: shafranskyi.andrei@student.karazin.ua. https://orcid.org/[0009-0004-7725-3556].

Miroshnyk Maryna, Doctor of Technical Sciences, ProfessorProfessor of theoretical and applied systems engineering department, V. N. Karazin Kharkiv National University, Svobody Sq., 4, Kharkiv, Ukraine, 61022. E-mail: m.miroshnyk@karazin.ua https://orcid.org/[0000000222312529].

Pakhomov Yurii, Doctor of Philosophy, Associate professor of the Department of Computer science and information technologies, O. M. Beketov National University of Urban Economy in Kharkiv, Marshala Bazhanova Str., 17, Kharkiv, Ukraine, 61002. E-mail: Yuriy.Pahomov@kname.edu.ua , https://orcid.org/0000-0002-2267-8600.

Pshenychnyi Kyrylo, Ph.D., graduate student of the department Design Automation Department, DAD, KHNURE, Kharkiv, Ukraine. E-mail: kyrylo.pshenychnyi@nure.ua, https://orcid.org/[0009-0007-0799-6604].

Shafranskyi Andrei, graduate student of the Department of theoretical and applied systems engineering, Kharkiv National University named after V. N. Karazin

Kharkiv National University, Svobody Sq., 4, Kharkiv, Ukraine, 61022. E-mail: shafranskyi.andrei@student.karazin.ua https://orcid.org/[0009-0004-7725-3556].