Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Кваліфікаційна наукова праця
на правах рукопису

ПШЕНИЧНИЙ КИРИЛО ЮРІЙОВИЧ

УДК 658:512.011: 681.326: 519.713

ДИСЕРТАЦІЯ
«МОДЕЛІ ТА МЕТОДИ ВЕРИФІКАЦІЇ ТЕМПОРАЛЬНИХ МОДЕЛЕЙ КІНЦЕВИХ АВТОМАТІВ НА МОВАХ ОПИСУ АПАРАТУРИ»

123 – Комп’ютерна інженерія
Технічні науки
Подається на здобуття наукового ступеня доктора філософії

Дисертація містить результати власних досліджень.

Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело _____ К.Ю. Пшеничний

Науковий керівник: Шкіль Олександр Сергійович, кандидат технічних наук, доцент

Харків – 2024

АНОТАЦІЯ

Пшеничний К.Ю. Моделі та методи верифікації темпоральних моделей кінцевих автоматів на мовах опису апаратури. – Кваліфікаційна наукова праця на правах рукопису.

Дисертація на здобуття наукового ступеня доктора філософії за спеціальністю 123 «Комп'ютерна інженерія». – Харківський національний університет радіоелектроніки, Міністерство освіти і науки України, Харків, 2024.

У роботі розв'язано науково-практичні задачі проєктування, верифікації та діагностики систем реального часу на базі моделей кінцевих автоматів через впровадження шаблонів на мовах опису апаратури (Hardware Descrtiption Languages, HDL), які використовуються для синтезу цифрових пристроїв реального часу.

Завдання дослідження.

1. Розробка HDL шаблонів на базі дискретних автоматів для розв'язання завдання проєктування цифрових пристроїв реального часу.

2. Розробка методів тестування темпоральних параметрів кінцевих часових автоматів на базі апарату асерцій.

3. Розробка методів діагностики кінцевих часових автоматів за рахунок введення апаратурної надлишковсті на етапі проєктування.

4. Розробка програмного комплексу автоматизації запропонованих методів верифікації та проєктування.

Об'єктом дослідження є процеси проєктування та діагностування керуючих автоматів в пристроях логічного управління реального часу.

Предмет дослідження – моделі та методи проєктування і верифікації HDL- моделей темпоральних керуючих автоматів. Математичний апарат: теорія графів, теорія автоматів, технічна діагностика.

У процесі дослідження використовуються наступні методи: побудова шаблонів на мовах опису апаратури Verilog, System Verilog та VHDL для

проєктування цифрових пристроїв реального часу; впровадження додаткових асерційних конструкцій в HDL-код цифрового пристрою для зменшення часу верифікації та підвищення якості тесту; впровадження апаратурної надлишкості для підвищення діагностованості кінцевого пристрою; удосконалення темпорального графу переходів; аналіз та класифікації систем реального часу залежно від способів обробки зовнішніх подій та видачі вихідних сигналів. Вищенаведені пункти спрямовані на скорочення часу проєктування та верифікації, а відповідно на зменшення часу виходу готового цифрового виробу на ринок (time-to-market, TTM), що є світовим трендом у сфері автоматизації проєктування обчислювальної техніки.

Наукова новизна дисертаційного дослідження полягає у такому.

1. Розроблено моделі верифікації темпоральних параметрів часових автоматів за допомогою апарату асерцій та формальних методів верифікації, що суттєво скоротило час верифікації через скорочення пошуку помилкової HDL конструкції.

2. Набули подальший розвиток моделі та методи моделювання цифрових систем логічного управління з обробкою зовнішніх подій з невизначеною тривалістю, що дозволило значно розширити клас подієвих пристроїв логічного керування, які можуть бути представлені часовими автоматними моделями.

3. Набули подальший розвиток моделі побудови легкотестованого часового автомату на основі введення апаратурної надлишковості у HDL опис, що суттєво скоротило довжину тестової послідовності шляхом підвищення керованості графової моделі часового автомату.

4. Удосконалено трьохкомпоненту HDL-модель кінцевого часового автомату з використання синтезуючої підмножини мов Verilog та VHDL, що на відміну від наявних моделей, розширило клас систем логічного управління реального часу, які описуються за допомогою часових автоматів.

Практична цінність результатів дослідження полягає у таких аспектах.

1. Скорочення часу проєктування цифрового пристрою завдяки використанню готових HDL-шаблонів на базі часових автоматів.

2. Скорочення часу верифікації та відладки HDL-моделі через використання асерційних точок, які суттєво скорочують час пошуку помилкової HDL консрукції.

3. Спрощення процесу проведення діагностичного експерименту через введення апаратурної надлишковості, що дає змогу встановлювати пристрій реального часу в будь-який стан за детерміновану кількість тактів.

Запропоновані підходи та методи було апробовано на низці цифрових проєктів із використання ПЛІС CPLD та FPGA у середовищі автоматизації проєктування Xilinx, Vivado, Aldec Reviera Pro із використанням мов VHDL та Verilog. Розглянуті методи було представлено на двох міжнародних конференціях у напрямі EDA.

Результати дисертації в складі проєктування пристроїв реального часу впроваджені в навчальний процес Харківського національного університету радіоелектроніки (акт про впровадження від 25.01.2024).

На основі результатів дослідження на підприємстві ТОВ «ТІМДЕВ» було створено програмний модуль із використанням фреймворка Molybden на замовлення клієнта з галузі EDA для розроблення моделей логічного керування реального часу. Програмний модуль використовується для створення та візуалізації темпоральних графів переходів та подальшої генерації HDL-коду моделі пристрою, що розробляється.

Ключові слова: модель, метод, аналіз, технології автоматизації проєктування, моделювання, класифікація, діагностичний експеримент, цифрові пристрої, FPGA, тестові, послідовності, системи реального часу, мови опису апаратури, події, керуючий автомат, граф переходів.

СПИСОК ОПУБЛИКОВАНИХ РОБІТ ЗА ТЕМОЮ ДИСЕРТАЦІЇ

Список публікацій здобувача, в яких опубліковані основні наукові результати дисертації:

1. Мірошник М.А, Шкіль О.С., Рахліс Д.Ю., Пшеничний К.Ю. Обробка подій у цифрових пристроях реального часу // Збірник наукових праць «Вісник ЧДТУ». 2023. №2. – С. 50-57. (Фахове видання категорії Б). DOI: https://doi.org/10.24025/2306-4412.2.2023

2. Miroshnyk M.A., Shmatkov S. I., Shkil O. S., Miroshnyk A. M., Pshenychnyi, K. Y. Temporal events processing models in finite state machines // Radio Electronics, Computer Science, Control. (Фахове видання категорії А, Web of Science). 2023. №4. P. 49 - 57. DOI: https://doi.org/10.15588/1607-3274-2023-4-5

3. Хаханова Г. В., Пшеничний К.Ю. Методи верифікації темпоральних властивостей цифрових автоматів // Радіоелектроніка та інформатика. 2019. №3. С. 39-41. (Фахове видання категорії Б). DOI: https://doi.org/10.30837/1563-0064.3(86).2019.214975

4. Мірошник М.А., Пшеничний К.Ю., Шафранський А.В., Шкіль О.С. Підвищення тестопридатності часових автоматів Мура. // Вісник Харківського національного університету імені В. Н. Каразіна, сер. «Математичне моделювання. Інформаційні технології. Автоматизовані системи управління». 2023. Вип. 58. С. 37-46. (Фахове видання категорії Б). DOI: https://doi.org/10.26565/2304-6201-2023-58-04

5. Мірошник М.А., Пахомов Ю. В., Пшеничний К.Ю., Шафранський А.В. Асерацийна верифікація моделей пристроїв реального часу з недетрмінованими зовнішніми подіями // Інформаційно-керуючі системи на залізничному транспорті. 2024. Інформаційно-керуючі системи на залізничному транспорті. Том 29, № 1. С. 37-44. (Фахове видання категорії Б). DOI: https://doi.org/10.18664/ikszt.v29i1.300988

Результати, які засвідчують апробацію матеріалів дисертації:

6. A. Shkil, A. Miroshnyk, G. Kulak, K. Pshenychnyi Assertion Based Design of Timed Finite State Machine // Proceedings of IEEE East-West Design & Test Symposium (EWDTS-2021), Batumi, Georgia. 2021, P. 1-4. DOI: 10.1109/EWDTS52692.2021.9581046.

7. A. Shkil, K. Pshenychnyi Testable Design Of Moore Timed Digital Finite State Machines // Proceedings of the 12-th International Scientific and Technical Conference, Kharkiv, Ukraine. November 28, 2023 – December 01, 2023, P. 1 – 7.

8. Пшеничний К.Ю. Функціональна верифікація переходів кінцевих автоматів за допомогою мови SystemVerilog // Матеріали 23 Міжнародного молодіжного форуму «РАДІОЕЛЕКТРОНІКА ТА МОЛОДЬ У XXI СТОЛІТТІ» 16–18 квітня 2019 р. Харків, 2019. С. 5-6.

ABSTRACT

Pshenychnyi K.Y. Temporal finite state machines models and verification methods in hardware description languages. – Qualifying scientific work on the rights of the manuscript.

The dissertation for the competition PhD scientific degree on a specialty 123 "Computer engineering". – Kharkiv National University of Radio Electronics, Ministry
of Education and Science of Ukraine, Kharkiv, 2024.

The work solves the scientific and practical problems of real-time systems based on finite state machines models design, verification and diagnosis by introducing hardware description language (HDL) templates used for the synthesis of real-time devices.

The objectives of the study are:

1. HDL pattern development based on finite state machine (FSM) approach for solving the real-time digital devices design task.

2. Development of methods for testing the temporal parameters of timed FSM based on the assertion apparatus.

3. Development of methods for diagnosing timed FSM via the introduction of hardware redundancy at the design stage.

4. Development of a software complex for the automation of the proposed verification and design methods.

The object of research is the design and diagnosis processes of control automata in real-time logical control devices.

The subject of research is design and verification models and methods of temporal control automata HDL-models. Mathematical apparatus: graph theory, automata theory, technical diagnostics.

In the course of the research, the following methods are used: construction of templates using Verilog, System Verilog and VHDL hardware description languages for real-time digital devices design; introduction of additional assertion

structures in the HDL code of the digital device in order to reduce the verification time and improve the quality of tests; introduction of hardware redundancy in order to increase the diagnostic ability of the target device; improvement of the temporal state diagram; analysis and classification of real-time systems based on the external events processing and issuing output signals. The above methods are aimed at reducing the design and verification time and, accordingly, at reducing the time-to-market of the target digital product, which is the global trend in the field of computer engineering design automation.

The scientific novelty of the dissertation research is:

1. For the first time, the timed FSM temporal parameters verification methods based the apparatus of assertions and formal verification were proposed, which significantly reduced the length and time of the diagnostic experiment by increasing the observability of each state of the finite state automaton, and also made it possible to increase the quality of the test.

2. Digital logic control systems with processing of external events of unknown duration models received further development, which made it possible to significantly expand the class of event logic control devices that can be represented by time automatic models.

3. Model of easy-tested timed FSM based on the introduction of hardware redundancy in the HDL description received futher development, which significantly reduced the length of the test sequence by increasing the controllability of the graph model of the time machine.

4. The three-component timed FSM HDL-model was improved using a synthesizing subset of the Verilog and VHDL languages, which, unlike the existing ones, expanded the class of real-time logic control systems described by models of time-based automata.

The practical value of the research results arises in the following aspects.

1. Reduction of digital device design time using ready-made HDL templates based on timed FSM.

2. Reducing the time of verification and debugging of the HDL model due to the use of assertion points, that significantly reduces the time of searching for an erroneous HDL operator.

3. Diagnostic experiment simplification through the introduction of hardware redundancy, which makes it possible to install a real-time device in any state in a specified number of cycles.

The proposed approaches and methods were tested on a number of digital projects using CPLD and FPGA in the Xilinx, Vivado, Aldec Reviera Pro design automation environments using VHDL and Verilog languages. The considered methods were presented at two international conferences in the field of EDA.

Based on the research results, a software module using the Molybden framework was created at the company "TOB «TIMDEV»" on the order of a client from the EDA industry for the development of real-time logic control models. The software module is used to create and visualize temporal state diagrams and generate the HDL code of the developed digital device.

Keywords: model, method, analysis, design automation technology, simulation, classification, diagnostics experiment, digital devices, FPGA, test sequences, real time systems, hardware description languages, events, control FSM, state diagram.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАК, ОДИНИЦЬ І ТЕРМІНІВ

ПЛІС – Програмована Логічна Інтегральна Схема

ДЕ – Діагностичний Експеримент

КА – Керуючий автомат

САПР – Система Автоматизованого Проєктування

HDL – Hardware Description Language

FSM – Finite State Machine

SVA – System Verilog Assertions

TFSM – Timed Finite State Machine

ASIC – Application-specific integrated circuit

FPGA – Field-Programmable Gate Array

RTL – Register Transfer Level

SoC – System-on-a-Chip

VHDL – Very High-Speed Integrated Circuits Hardware Description Language

BEL – Basic Elements of Logic

LUT – Look-up Table

ВСТУП

Логічні системи управління є базовим вузлом будь-якої системи автоматизованого управління процесами. Особливе місце посідають системи реального часу (real time devices). Для таких систем швидкодія оброки вхідних та вихідних сигналів є критичною. Найпоширенішою моделлю представлення цифрових пристроїв є модель кінцевих автоматів, у якій вхідні сигнали представляють події ззовні, дискретний алфавіт – внутрішні стани системи, а вихідні сигнали – керуючі сигнали. Питання проєктування та верифікації моделей пристроїв реального часу на базі кінцевих автоматів є актуальним для сучасного стану галузі автоматизації проєктування обчислювальної техніки.

Також варто зазначити, що сучасні цифрові пристрої – це складні системи, які складаються з мільйонів і мільйонів логічних елементів. Мови опису апаратури, як Verilog і VHDL, забезпечують вищий рівень абстракції, що дозволяє інженеру працювати із цифровим проєктом як кодом, який описує його функціональність. Отже, проєктувальник використовує синтаксис мови для реалізації цільового пристрою незалежно від його складності та розміру. Водночас питання діагностики та тестування є важливим аспектом проєктування.

Варто зазначити, що структурні методи діагностики, які використовуються для виявлення константних несправностей, не є ефективними. Крім того, ці методи практично неможливо застосувати на практиці для реальних пристроїв через їхні габарити та розміри.

Відомо, що верифікація цифрових проєктів, розроблених на базі ASIC, FPGA та SoC займає до 70% від загального часу проєктування. Як результат, до 80% кодової бази проєкту займає код тестування (testbench). Зменшення цих двох параметрів зменшує час випуску пристрою на ринку (time-to-market) і є одним з головних завдань галузі автоматизації проєктування обчислювальної техніки.

Отже, окрім завдання проєктування пристроїв реального часу, актуальним є завдання тестування HDL-моделей таких пристроїв, а також їхня діагностика та підвищення тестопридатності.

Чималий внесок у розв'язання проблем тестопридатного проєктування та діагностування цифрових систем зробили провідні вітчизняні вчені: А. П. Горяшко, Є. С. Согомонян, В. Г. Тоценко, С.І. Баранов, Д. В. Сперанський, Л. В. Дербунович, В. С. Харченко, М. Л. Малиновський, Г. Ф. Кривуля, В. І. Хаханов, Р-Й. Убар та закордонні: E. G. Hennie, P. G. Bennetts, M. A. Breuer, M. Abramovici, J. Roth, V. Pedroni, R. Alur, D. Hanna.

Об'єктом дослідження є процеси проєктування та діагностування керуючих автоматів у пристроях логічного управління реального часу.

Предмет дослідження – моделі та методи проєктування і верифікації HDL- моделей темпоральних керуючих автоматів. Математичний апарат: теорія графів, теорія автоматів, технічна діагностика.

Мета дослідження – проєктування, верифікації та діагностики систем реального часу на базі моделей кінцевих автоматів з допомогою впровадження шаблонів на мовах опису апаратури, які використовуються для синтезу пристроїв реального часу.

Завдання дослідження:

1. Розроблення HDL шаблонів на базі дискретних автоматів для розв'язання завдання проєктування цифрових пристроїв реального часу.

2. Розроблення методів верифікації темпоральних параметрів кінцевих часових Розроблення на базі апарату асерцій.

3. Розроблення методів тестопридатного проєктування кінцевих часових автоматів за рахунок введення апаратурної надлишковсті на етапі проєктування.

4. Розроблення програмного комплексу автоматизації запропонованих методів верифікації та проєктування.

Для вирішення вищезазначених завдань використовуються такі методи: побудова шаблонів на мовах опису апаратури Verilog, System Verilog та VHDL

для проєктування цифрових пристроїв реального часу; впровадження додаткових асерційних конструкцій в HDL-код цифрового пристрою для зменшення часу верифікації та підвищення якості тесту; впровадження апаратурної надлишкості для підвищення тестопридатності кінцевого пристрою; удосконалення темпорального графу переходів; аналіз та класифікації систем реального часу залежно від способів обробки зовнішніх подій та видання вихідних сигналів; формальні методи верифікації цифрових систем. Вищенаведені пункти спрямовані на скорочення часу проєктування та верифікації, а відповідно на зменшення часу виходу готового цифрового виробу на ринок (time-to-market), що є світовим трендом у сфері автоматизації проєктування обчислювальної техніки.

Наукова новизна результатів досліджень:

1. Розроблено моделі верифікації темпоральних параметрів часових автоматів за допомогою апарату асерцій та формальних методів верифікації, що суттєво скоротило час верифікації через скорочення пошуку помилкової HDL конструкції.

2. Набули подальший розвиток моделі та методи моделювання цифрових систем логічного управління з обробкою зовнішніх подій з невизначеною тривалістю, що дозволило значно розширити клас подієвих пристроїв логічного керування, які можуть бути представлені часовими автоматними моделями.

3. Набули подальший розвиток моделі побудови легкотестованого часового автомату на основі введення апаратурної надлишковості у HDL опис, що суттєво скоротило довжину тестової послідовності шляхом підвищення керованості графової моделі часового автомату.

4. Удосконалено трьохкомпоненту HDL-модель кінцевого часового автомату з використання синтезуючої підмножини мов Verilog та VHDL, що на відміну від наявних моделей, розширило клас систем логічного управління реального часу, які описуються за допомогою часових автоматів.

Практична цінність результатів дослідження полягає у таких аспектах.

1. Скорочення часу проєктування цифрового пристрою завдяки використанню готових HDL-шаблонів на базі часових автоматів.

2. Скорочення часу верифікації та відладки HDL-моделі через використання асерційних точок, які суттєво скорочують час пошуку помилкової HDL консрукції.

3. Спрощення процесу проведення діагностичного експерименту через введення апаратурної надлишковості, що дає змогу встановлювати пристрій реального часу в будь-який стан за детерміновану кількість тактів.

Розглянуті методи було представлено на двох міжнародних конференціях у галузі EDA.

Обґрунтованість наукових положень. Отримані в процесі виконання досліджень наукові висновки і практичні результати з моделювання пристроїв реального часу, а також методів тестування та діагностики є достовірними, що підтверджується достатньою кількістю проведених експериментів, точністю розрахунків, апробацією результатів на міжнародних науково-практичних конференціях, впровадженням результатів в освітній процес.

Результати дисертації в складі проєктування пристроїв реального часу впроваджені в навчальний процес Харківського національного університету радіоелектроніки (акт про впровадження від 25.01.2024).

На основі результатів дослідження на підприємстві ТОВ «ТІМДЕВ» було створено програмний модуль із використанням фреймворка Molybden на замовлення клієнта з галузі EDA для розроблення моделей логічного керування реального часу. Програмний модуль використовується для створення та візуалізації темпоральних графів переходів та подальшої генерації HDL-коду моделі пристрою, що розробляється.

Особистий внесок здобувача. Усі наукові і практичні результати отримані автором особисто. У роботах, опублікованих зі співавторами, здобувачеві належать (Додатки А, Б):

[1] – розглянуто питання моделювання цифрових систем логічного управління з обробкою зовнішніх подій з невідомою тривалістю;

[2] – запропоновано трьохкомпонентний HDL шаблон часового автомата з подіями з невідомою тривалістю;

[3] – метод верифікації HDL-моделей пристроїв реального часу пропонується застосовувати механізм властивостей та асерцій мови System Verlog;

[4] – метод підвищення діагностованості цифрового пристрою реального часу завдяки введенню апаратурної надлишковості;

[5] – метод верифікації моделей пристроїв реального часу з обробкою зовнішніх подій із недермінованою тривалістю, що описані з використанням мов опис апаратури завдяки використанню апарату асерцій для опису темпоральної природи вищезазначених моделей.

Апробація результатів дисертації. Результати роботи були представлені та обговорені на таких конференціях: XXIII Міжнародний молодіжний форум «Радіоелектроніка і Молодь у XXI столітті»; IEEE East-West Design and Test Symposium 2021; 12th International Science and Technical Conference «Information Systems and Technologies» IST-2023.

Публікації. Основні положення та результати дисертаційної роботи досить повно відображені у 5 друкованих працях, серед яких, 5 – у наукових журналах, включених до «Переліку наукових фахових видань України», з них 1 – категорії А, що входять до наукометричної бази Scopus; 4 – категорії Б, а також 3 тез доповідей у матеріалах міжнародних наукових конференцій.

Структура дисертації: 109 сторінок (з них 77 представляють основний текст) і містить: 4 розділи, 63 рисунка, 2 таблиці, список джерел з 102 назв (на 9 с.), 1 додаток (на 2 с.), анотації на 8 с.

1 ПРОЄКТУВАННЯ ЦИФРОВИХ СИСТЕМ РЕАЛЬНОГО ЧАСУ

1.1 Проєктування систем реального часу, сучасний стан питання

У сучасному світі питання автоматизації управління промисловими та побутовими процесами відіграє важливу роль. Серед усього класу завдань управління є такі, для яких важливим є не тільки факт настання зовнішньої події, але і час, у який ця подія відбувається. Час реакції таких систем на зміну вхідних даних або внутрішнього стану має бути детермінованим. Дотримання часових обмежень є особливо актуальним для систем управління в медицині, важкій та хімічній промисловості, атомній енергетиці, галузі Internet of Things [1].

Міжнародні стандарти та державні стандарти України визначають режим реального часу цифрової системи як режим, що забезпечує взаємодію обчислювальної системи з зовнішніми щодо неї процесами зі швидкістю протікання цих процесів [2, 3]. Окрім того, на такого роду системи накладаються додаткові вимоги щодо надійності, як апаратурної, так і програмної компоненти [4].

Сучасний цифровий пристрій (ЦП) – складна багаторівнева система, що складається з мільйона, а подекуди десятка мільйонів різних елементів: логічні вентилі, мультиплексори, суматори, елементи пам’яті тощо. Очевидно, що підвищення складності цифрових систем призвело до пошуку нових способів спрощення проєктування та тестування. Важливим поштовхом у цьому напрямі стали мови опису апаратури, такі як Verilog і VHDL. Такі мови забезпечують високий рівень абстракції, що дозволяє інженеру працювати із цифровим пристроєм у вигляді коду, який описує його функціональність. Такий підхід отримав назву високорівневий синтез – High Level Synthesis.

Спрощений процес проєктування цифрової системи представлено на рисунку. 1.1. Як видно з цієї схеми проєктування – це ітеративний процес, яким на різних етапах, як правило, займаються різні команди інженерів [5].

Загальновідомо, що верифікація цифрових проектів, розроблених на базі ASIC, FPGA та SoC, займає до 70% від загального часу проєктування [6-8]. Як результат, до 80% кодової бази проєкту займає код тестування (testbench). Зменшення цих двох параметрів зменшує час випуску пристрою на ринку (time-to-market) і є однією з головних задач галузі автоматизації проєктування обчислювальної техніки.

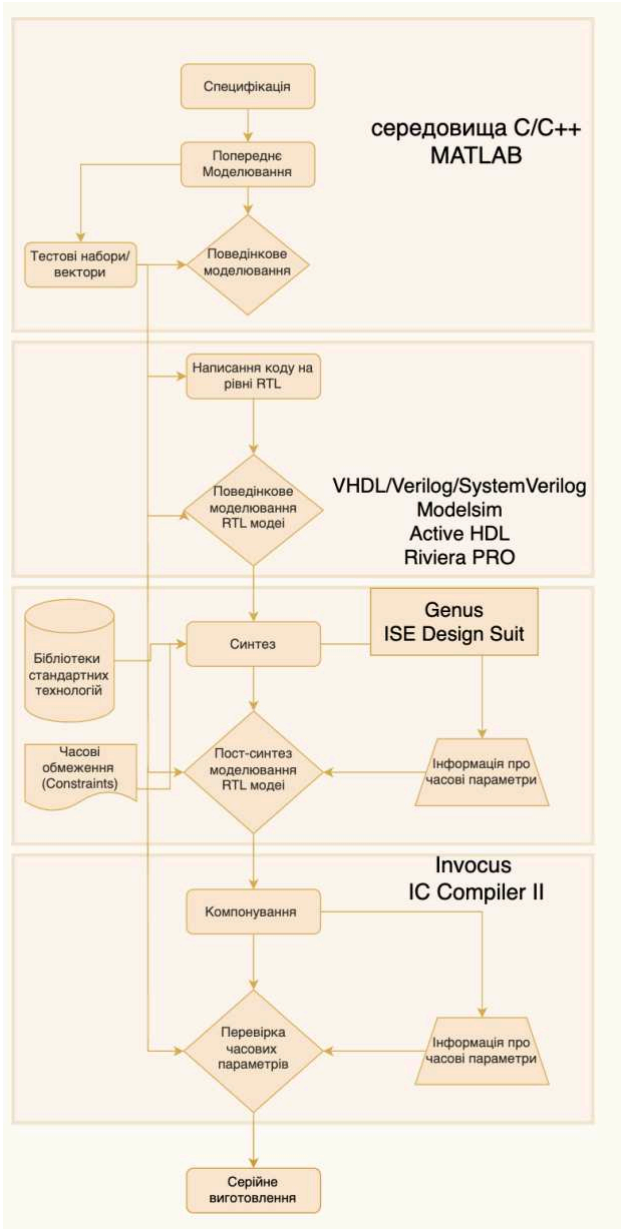


Рисунок 1.1 – Спрощена блок-схема процесу проєктування цифрового пристрою

Як видно з рис 1.1 до основними етапами проєктування є:

1. Концепція - початковий етап, на якому оцінюється актуальність ідеї проєкту з погляду потреб ринку й можливості реалізації.

2. Специфікація, тобто розроблення докладних вимог до функціональності, продуктивності і інших характеристик проєкту. Цей етап включає в себе створення високорівневого програмного прототипу пристрою - моделі системного рівня, у якій можуть використовувати популярні мови системного рівня, такі як SystemC [9], TLA+ [10].

3. Проєктування RTL-моделі. На основі специфікації розробляється докладна архітектура системи, що проєктується: визначається структура й інтерфейси основних функційних блоків. Модель із системного рівня крок за кроком перетворюється на модель RTL-рівня шляхом використання мов опису апаратури - VHDL [11], Verilog [12].

4. Поведінкове моделювання. Розроблена HDL-модель перевіряється на наявність помилок у реалізації функційних вимог через використання testbench – HDL-коду, який імітує роботу пристрою у реальному середовищі [13].

5. Логічний синтез. За допомогою спеціальних програмних засобів (синтезаторів), що є частиною САПР, робоча RTL-модель перетвориться на модель рівня логічних примітивів – вентилів.

6. Пост-синтез моделювання – моделювання моделі, отриманої унаслідок синтезу. Цей етап дає змогу побачити чітку картину поведінки розробленої HDL-моделі.

7. Моделювання затримок, відоме також як часова верифікація, включає оцінку часових параметрів вентильної моделі. Під час цього етапу враховуються елементні (інерційні) і транспортні затримки, а також сигнальні перегони.

8. Розміщення. Обирається технологія фізичної реалізації системи, що проєктується, здійснюється реалізація елементів вентильної структури на реальні комірки чіпа, виконується оптимізація розміщення [14]. На цьому ж

етапі в пристрій вносяться елементи діагностичної інфраструктури [15].

9. Фізична верифікація та конструкторське проєктування включають аналіз енергоспоживання, виявлення температурних аномалій, паразитних ємностей та інших низькорівневих ефектів. На завершальній стадії процесу чіп укладається в ізольований корпус та розміщується на друкованій платі разом з іншими компонентами системи, яку проєктують.

Проблема тестування HDL-моделей є надзвичайно актуальною, про що свідчить велика кількість наукових робіт, що ведуться в цьому напрямі, а також інструментальних засобів, що розробляються для спрощення тестування таких моделей. Нині скорочення часу тестування, а також підвищення якості тестів досягається через спеціальні розширення HDL, методологій (UVM, OVM), створенням нових математичних моделей для моделювання несправностей [16-17].

У [18-19] показані особливості застосування та проєктування розподілених систем реального часу та їхнє застосування для різних прикладних галузей: медицина, робототехніка тощо. Особливості проєктування програмного забезпечення (ПЗ) для систем реального часу описано у [20]. У [21] розглянуті питання розроблення та тестування ПЗ для систем реального часу. Пропонуються шаблони тестування, описуються типові бібліотеки та фреймворки для діагностики.

1.2 Автоматні моделі в системах логічного управління реального часу

Шаблон кінцевого автомата (FSM) є канонічною моделлю під час проєктуванні цифрових систем управління та обробки інформації. Автомат є математичною абстракцією, що складається з дискретних множин внутрішніх станів, вихідного та вхідного алфавітів, та початкового стану [22]. У [23] показано, що будь-який цифровий пристрій можна розглядати як комбінацію двох основних компонентів: операційного та керуючого автоматів (КА).

Операційний пристрій або операційний автомат (ОА) працює на основі принципу мікропрограмного управління, що передбачає такі кроки.

1. Будь-яку операцію пристрою можна розглядати як послідовність елементарних логіко-арифметичних операцій на вхідних словах.

2. Логічні умови визначають послідовність виконання цих операцій.

3. Мікропрограма виражає алгоритм у термінах мікрооперацій та логічних умов.

4. Мікропрограма представляє функціональність пристрою, визначаючи його структуру та порядок функціонування в часі.

Операційний автомат забезпечує зберігання слів інформації, виконання мікрооперацій алгоритму, робить розрахунки значень логічних умов, необхідних для виконання алгоритму. Його характеризують такі дискретні множини: $Y = \{y_1, y_m\}$ - мікрооперації; $X = \{x_1, x_l\}$ - логічні умови, що виникають під час виконання алгоритму; $D = \{d_1, d_h\}$ - дані, які подаються на вхід; $R = \{r_1, r_q\}$ - результати операцій; $S = \{s_1, s_n\}$ - внутрішні слова, що містять інформацію в процесі виконання операцій. На рисунку 2.1 показано структуру ОА як комбінацію операційної та керуючої частини та зв’язків між ними.

Рисунок 2.1 – Представлення цифрового пристрою як комбінації операційної та керуючої частин

У [24-28] розглянуто процеси проєктування цифрових пристроїв із використанням мов опису апаратури Verilog, VHDL. Зокрема, окремі розділи присвячено проєктуванню на основі автоматів Мура та Мілі з використання двох та трьохпроцесорних шаблонів.

Системи логічного управління є різновидом системи логічного управління (СЛУ). Особливістю таких систем полягає в тому, що вхідні сигнали *X* і вихідні сигнали *Y* можуть приймати тільки два значення – 0 та 1 залежно від граничних значень конкретних фізичних величин, що визначають ці значення [29]. Об’єктами керування СЛУ є різноманітні датчики, пристрої регулювання дорожнього руху, тиску, температури тощо. Варто зазначити, що СЛУ впливають на об’єкти керування через виконавчі вузли – магнітні пускачі, електромеханічні перетворювачі, гідроромеханічні перетворювачі тощо. Отже, при проєктуванні системи логічного управління передбачається, що вона буде складником кінцевого об’єкта керування. Інформація про роботу та стан об’єкта надходить в СЛУ від сигналізаторів становища, стану й параметрів.

Під час використання автоматних моделей для опису та проєктування систем реального часу необхідно враховувати особливості характерні для цих систем. Цикли синхросигнала визначають машинний (автоматний) час, упродовж якого функціонує автомат. Однак системи реального часу працюють у метричному часі. Іншими словами, стан таких пристроїв залежить як від вхідних сигналів, так і від часу, упродовж якого ці сигнали обробляються. Для опису таких систем використовується модель часового автомата (TFSM). Концепцію часового автомата як способу опису систем реального часу запропоновано у [30, 31]. Граф переходів доповнюється кінцевою множиною таймерів, які приймають дійсні значення. Вершини графу називаються позиціями, а ребра – переходами. Кожен таймер скидається до нуля під час переходу та збільшує своє значення з кожним циклом FSM. Кожен перехід пов’язаний з обмеженням тактового сигналу, що

означає, що цей перехід можна здійснити, лише якщо поточні значення таймера задовольняють цьому обмеженню. Кожна позиція має обмеження за таймером, яке називається інваріантом; система може перебувати в цьому стані лише доти, поки задовольняється її інваріант.

У [32] досліджено формальні методи тестування, які враховують часові характеристики системи. Для опису поведінки системи була використана модель TFSM, де замість багатьох таймерів введена одна часова змінна. Кожен перехід між станами TFSM характеризується часом (затримкою), упродовж якого буде виконано перехід. Для верифікації TFSM введено поняття часової карти (time trace) як послідовності переходів між станами та часом, упродовж якого вони відбувалися. Запропоновано алгоритм генерації повного набору тестів, який дає змогу перевірити відповідність моделі заданим часовим параметрам. У [33] модель TFSM розширено шляхом введення тайм-ауту в стані та затримки під час переходу, що дозволяє більш точно враховувати параметри синхронізації під час тестування синхронізованого автомата через побудову часових карт, починаючи від початкового стану.

Під час побудови тестів для синхронізованого автомата в роботі [34] розглянуто модель TFSM, яка враховує тайм-аути в станах і затримки вихідних сигналів з урахування реалізації функції переходів. Водночас враховується, що якщо впродовж тайм-ауту не надходить вхідний сигнал, то автомат переходить у детермінований стан.

Модель часового автомата включає три типи часових параметрів: тайм-аути в станах, часові обмеження на приймання вхідних сигналів і час обробки вхідного сигналу, тобто затримка вихідного сигналу щодо вхідного. Крім того, можна розглядати часові автомати з меншою кількістю параметрів [35-37]. У цих роботах розглядаються задачі мінімізації часових автоматів, перевірки їхньої еквівалентності та створення тестів.

У [38] введено нову класифікацію моделей кінцевих автоматів залежно від реалізації переходів та вихідних сигналів. Усі автомати розділено на 3

класи. Автомати першої категорії (регулярні, regular) переходи залежать виключно від вхідних сигналів, а значення вихідних сигналів – тільки від станів. Це класичні автомати Мілі та Мура. В автоматів другої категорії (часові, timed) переходи залежать від вхідних сигналів та часу появи цих сигналів; вихідні сигнали – тільки від станів. Переходи автоматів третьої категорії (рекурсивні. recursive) залежать від вхідних сигналів та часу їх появи, а вихідні – від теперішнього стану та попереднього, тобто для вихідних сигналів у стані a_i реалізується функція $y_i = y_i + y_j$, де a_j - попередній стан автомата.

В [39] розглянута узагальнена модель часового автомата з таймаутами й часовими обмеженнями та вихідними затримками. Ця робота є фундаментальною погляду проєктування систем логічного управління реального часу, адже вперше представлено узагальнену 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 автоматних тактах, d - функція виходів та 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 гаймаути і t_d - вихідні затримки. Окрему увагу приділено способу обробки зовнішніх подій. Для кожного стану a_i встановлюються вхідні обмеження $t_c(a_i)$, тобто період часу, упродовж якого автомат у стані a_i може обробляти вхідні події. Часові обмеження визначаються в циклах такту синхросигнала й обчислюються *як* $t_c = (t_1 – t_0)$. Якщо зовнішня подія відбулася за межами «вікна» часових обмежень – автомат не реагує на неї. У цій роботі запропоновано автоматний шаблон на мові VHDL, який забезпечує реалізацію часових параметрів, що забезпечує коректний синтез та імплементацію запропонованої моделі у FGPA та CPLD.

У [40] запропоновано уточнену класифікацію системних подій. Ця класифікація використовується під час розроблення специфікації програмного та апаратного забезпечення. Є три класи зовнішніх подій:

– бізнес-подія – дія користувача-людини, яка стимулює діалог із програмним або апаратним забезпеченням, наприклад, коли користувач натискає кнопку;

– сигнальна подія – така подія реєструється, коли система отримує сигнал керування, зчитування даних або переривання від зовнішнього апаратного пристрою чи іншої програмної системи;

– часова подія – подія, що ініціюється в певні проміжки часу, наприклад, коли годинник комп'ютера досягає визначеного часу або коли попередньо встановлений проміжок часу минув після попередньої події (як у системі, яка записує показання температури датчика кожні 10 секунд).

У [41] розглядаються системи реального часу, у яких деякі часові параметри є невизначеними. Також розглянуто питання параметричних автоматів із верхньою межею (The Upper-bound Parametric Timed Automata) – одне з найпростіших розширень часових автоматів із параметрами, у якому параметри використовуються як верхня межа таймера.

У [42] розроблено абстрактні моделі та виділено класи безпечних автоматів. Запропоновано табличні та графічні методи завдання таких автоматів, а також методи синтезу безпечних автоматів із функційною деградацією, які ґрунтуються на формуванні множин відповідальних операцій і побудові, аналізі та перетворенні χ-автоматів.

1.3 Методи діагностики цифрових пристроїв реального часу

Важливим аспектом побудови систем автоматизованого управління реального часу є забезпечення надійності функціонування, що неможливо без використання систем автоматизації діагностування. Діагностування має на меті підвищення надійності та ресурсу технічних систем. Основним завдання

технічної діагностики є отримання достовірних даних про стан технічного об’єкта шляхом збору, аналізу та обробки різних параметрів обладнання, які можуть бути як якісними, так і кількісними. Цей процес також включає управління обладнанням відповідно до визначеного алгоритму діагностування.

У роботі [43], яка має вагомий вплив у теорії експериментів з автоматами, пропонується використання характеристичних послідовностей, які дають змогу ідентифікувати таблицю переходів-виходів (ТПВ) справного автомата. Ці процедури найкраще працюють для класу мінімальних сильнозв'язних автоматів із характеристичними послідовностями. У цій самій роботі визначений клас послідовностей, які є встановлюючими, синхронізуючими та характеристичними, і які допомагають у побудові діагностичних експериментів для автоматів, що не мають характеристичних послідовностей.

Важливим напрямом тестування автоматних моделей є знаходження унікальних вхід-вихідних послідовностей для ідентифікації внутрішніх станів автоматів. Проблема розроблення формальних методів генерації тестових послідовностей для автоматів була активною дослідницькою галуззю в останні десятиліття. Наприклад, у [44-47] представлено огляди тестування на основі автоматних моделей. У [49-56] запропоновано деякі типові методи виведення тестових послідовностей.

У [57, 58] запропоновано спосіб побудови діагностичного експерименту над графовою моделлю автомата через обхід всіх дуг графу, починаючи з початкової вершини. Водночас перевіряються всі поодинокі несправності переходів, а також справність функцій переходів автомата.

Питання тестування часових автоматів є окремою сферою досліджень. У статті [59] представлено метод отримання повного набору тестів для детермінованого автомата з тайм-аутами, коли відомі лише верхня межа кількості станів і найбільший тайм-аут стану. Показано, що набір тестів, отриманий для відповідного класичного автомата, набагато довший, ніж

отриманий безпосередньо з автомата з тайм-аутами. Проблеми написання тестів для часових автоматів, ідентифікації внутрішніх станів, black-box верифікації присвячено роботи [60-64].

У методиці, описаній у [65], запропоновано отримання тестів з гарантованим покриттям несправностей на основі моделі синхронізованого часового автомата з одним тактовим сигналом. Процес отримання тесту базується на визначенні області несправностей, що дозволяє створювати набори тестів із прийнятною довжиною. Ця область включає всі можливі помилкові реалізації автомата з відомими найбільшими межами часових обмежень та мінімальною тривалістю часових обмежень.

Використання САПР дає змогу проєктувати цифрові системи будь-якої складності. Це, зокрема, стало можливим завдяки мовам опису апаратури. Очевидно, що з одного боку HDL-код є набором синтаксичних конструкцій, а з іншої – моделлю пристрою, що проєктується. Таким чином, тестування HDL-моделей є важливим аспектом у процесі проєктування. Удосконалення існуючих та впровадження нових методів верифікації HDL-моделей є окремим напрямом досліджень.

Відомо, що найпростішою системою тестування HDL-коду є testbench. Методології написання testbench з використання мови SystemVerilog присвячено працю [66], у якій розглянуто різні методології написання тестів для верифікації різних особливостей цифрового проєкту, а також питання асерцій, функціонального покриття та багаторівневої архітектури тестового середовища.

У [67-69] розглянуто різні методології та технології верифікації Application-Specific Integrated Circuit (ASIC) та System-on-a-Chip (SoC) проєктів. Актуальність проблеми тестування, популярність підходів та рішень призвело до виникнення спеціальних HDL бібліотек та методологій, таких як OVM [70] та UVM [71]. Це дало змогу скороти час написання тестової інфраструктури за рахунок використання шаблонів із готових бібліотек.

У [72] було представлено модульну конструкцію SoC, яка слідкує за активацією структур об'єктів інтелектуальної власності (IP-Cores) у тестових, налагоджувальних та функційних режимах. Покращена інфраструктура SoC Design-for-Test (DfT) включає заходи безпеки для запобігання зовнішнім атакам на IP-Core у режимах тестування. Спочатку захищаються активи IP-Cores та SoC від атак у тестових та налагоджувальних режимах, а потім інфраструктура DfT використовується для виявлення атак у функційному режимі.

У [73], пропонується комплексний огляд різних аспектів безпеки SoC в режимі тестування, включно з відомими моделями загроз, класифікацію атакуючих та існуючих методів. Також представлена концепція для побудови безпечної інфраструктури тестування SoC з фокусом на тестування вбудованих ядер.

Зі зростанням технологічної складності сучасних конструкцій SoC продовжують збільшуватися в розмірах та залучати дедалі більше IP-Cores. У зв'язку із цим стає набагато складнішим завершити тестування великих SoC в межах заданих часових витрат і вартості. Автоматизований ієрархічний тест зазвичай допомагає ефективно вирішити цю проблему, але для таких систем підготовка вхідних даних, особливо інформації на рівні IP-Cores та опису тестових послідовностей, зазвичай займає багато часу. У [74], було представлено ефективне рішення для підготовки вихідних даних для ієрархічної системи тестування IP-Cores у складі SoC.

1.4 Методи тестопридатного та асерційного проєктування цифрових пристроїв

Зростання складності систем збільшило наявний розрив між можливостями проєктування та методами тестування та верифікації. Це призвело до виникнення нових методологій верифікації сучасних проєктів на системному рівні, а саме методів формальної верифікації. Під формальною

верифікацією цифрового проєкту будемо розуміти процес доведення або спростування правильності системи відповідно до певного набору формальних вимог із використанням математичних методів [75].

Основним завданнями при використанні формальних методів є виявлення властивостей цифрового проєкту, їхній опис та подальша верифікація. У роботах [76-78] описано теоретичні засади формальних методів верифікації при проектуванні апаратури. Математичний апарат формальних методів детально розглянуто в працях [80-86].

Іншою фундаментальною роботою в цій галузі є дослідження [79], у якому розглянуто питання практичного застосування формальних методів на реальних проєктах індустрії, а саме на мікропроцесорах, обладнанні для обчислень із плаваючою комою, протоколах, підсистемах пам'яті та комунікаційному обладнанні. Також було дано докладні характеристики різним механізмам опису формальних властивостей, як-от темпоральна логіка, логіка предикатів, абстрагування та уточнення. Розглянуті методи верифікації включають перевірку моделі, автоматизовані теоретичні методи, автоматизоване доведення теорем та підходи, які об’єднують вищезазначені методи.

У [87] запропоновано модель верифікації SoC проєктів. Розглянута модель базується на розширення HDL-коду за допомогою програмних конструкцій у формі асерцій, які дають змогу істотно зменшити час створення прототипу проєкту. Викладені теоретичні положення і практичні приклади, які доводять ефективність механізму асерцій для вирішення завдань перевірки проєктів на системному рівні.

Формальні методи перевірки проєктування апаратного забезпечення, включно з секвенційними та паралельними системами, часовою логікою, алгоритмами перевірки моделі, символьною перевіркою моделі та інструментами формальної перевірки детально розглянуто у [88-92].

У [93] запропоновано модель верифікації цифрових систем на чіпі (SoC). Модель базується на розширення дизайну з допомогою програмного

коду у формі асерційних конструкцій, які дають змогу істотно зменшити час створення прототипу дизайну. Викладено теоретичні положення і практичні приклади, які доводять ефективність запропонованих підходів для вирішення завдань проєктування перевірки на системному рівні.

Окремим напрямом практичного застосування формальних методів є вираз властивостей спеціальними мовами. Основними мовами опису властивостей є Property Specification Language (PSL) [94] (використовується у VHDL, Verilog, System Verilog) та SystemVerilog Assertions (SVA) [11] (підмножина мови System Verilog).

Питанню використання PSL присвячено праці [95-96]. У цих працях розглянуто основні оператори, аспекти опису темпоральних властивостей, рівні верифікації та проєкти з багатьма синхросигналами.

Синтаксичні особливості та використання мови SVA є темою праць [97-99]. У запропонованих роботах розглянуто синтаксис мови, опис різного типу властивостей, стратегії використання асерцій для підвищення якості процесу верифікації та скорочення його часу.

1.5 Мета та постановка задач дослідження

Проаналізувавши список публікацій з тематики проєктування цифрових пристроїв реального часу з використанням автоматних шаблонів, можна зробити висновок, що це питання має низку недосліджених аспектів. Це насамперед стосується питання обробки зовнішніх подій у моделях таких пристроїв, верифікації та діагностики, а також побудови легкотестованих HDL-моделей.

Відомо, що основним способом тестування HDL-моделей автоматів є написання testbench, у якому реалізується певна стратегія обходу усіх станів та дуг із відповідним аналізом вихідних сигналів. Цей спосіб є ефективним за заздалегідь відомій моделі автомата, але не підходить під час black-box тестування, коли тестувальник не знає внутрішні деталі імплементації. Окрім

того, системи логічного управління реального часу потребують верифікації часових параметрів проєктованої моделі. Це не завжди можливо зробити з використанням класичних підходів.

Сутність дослідження полягає у вдосконаленні наявних HDL-шаблонів моделей пристроїв логічного управління реального часу, а саме аспекту обробки зовнішніх подій, що дозволить розширити клас пристроїв логічного управління реального часу; скороченні часу проєктування через використання асерційних конструкцій для опису темпоральних особливостей HDL-моделей; скороченні часу проведення діагностичного експерименту через введення апаратурної надлишковості на етапі проєктування HDL-моделі.

Ринкова привабливість дослідження полягає у впровадженні готових HDL-шаблонів пристроїв логічного управління реального часу на основі керуючих автоматів, що дасть змогу скоротити час написання початкової HDL-моделі; впровадженні моделей та методів асерційного проєктування, що має на меті зменшення часу верифікації, а відповідно TTM.

Об’єктом дослідження є процеси проєктування та тестування керуючих автоматів у пристроях логічного управління реального часу.

Предмет дослідження – моделі та методи проєктування і верифікації HDL-моделей темпоральних керуючих автоматів. Математичний апарат: теорія графів, теорія автоматів, технічна діагностика.

Мета дослідження – проєктування, верифікації та діагностики систем реального часу на базі моделей кінцевих автоматів з допомогою впровадження шаблонів на мовах опису апаратури, які використовуються для синтезу пристроїв реального часу.

Під час дослідження використовуються такі методи: побудова шаблонів на мовах опису апаратури Verilog, System Verilog та VHDL для проєктування цифрових пристроїв реального часу; впровадження додаткових асерційних конструкцій в HDL код цифрового пристрою для зменшення часу верифікації та підвищення якості тесту; впровадження апаратурної надлишкості для підвищення тестопридатності кінцевого пристрою; удосконалення

темпорального графу переходів; аналіз та класифікації систем реального часу залежно від способів обробки зовнішніх подій та видачі вихідних сигналів. Вищенаведені пункти спрямовані на скорочення часу проєктування та верифікації, а відповідно на зменшення часу виходу готового цифрового виробу на ринок (time-to-market), що є світовим трендом у сфері автоматизації проєктування обчислювальної техніки.

Завдання дослідження.

1. Розроблення методи верифікації темпоральних моделей часових автоматів за допомогою апарату формальної верифікації для зменшення часу етапу верифікації цифрового проекту.

2. Удосконалення наявних та розроблення нових моделей та методів моделювання цифрових систем реального часу з обробкою зовнішніх подій.

3. Розвиток способів побудови легкотестованого часового автомату на основі введення апаратурної надлишковості у HDL опис, що суттєво скоротить довжину тестової послідовності через підвищення керованості графової моделі часового автомату.

4. Розроблення програмного комплексу автоматизації запропонованих методів верифікації та проєктування.

2 HDL-МОДЕЛІ ОБРОБКИ ПОДІЙ У СИСТЕМАХ РЕАЛЬНОГО ЧАСУ

Другий розділ роботи присвячено розробленню моделей та методів обробки зовнішніх подій у системах логічного управління реального часу, які описані через використання автоматних шаблонів.

У процесі роботи необхідно вирішити наступні питання:

1) класифікувати пристрої реального часу залежно від методів обробки зовнішніх події;

2) на базі запропонованої класифікації розробити HDL-шаблони обробки зовнішніх подій на базі автоматних шаблонів реального часу;

3) провести моделювання та синтез запропонованих HDL-шаблонів.

2.1 Семантика часового автомата

Однією з канонічних форм представлення операційного та керуючих блоків є кінцевий автомат (Finite State Machine). Така форма представлення пристрою дає змогу синтезувати дискретні схеми без урахування часових параметрів та конкретних фізичних елементів, з яких ці схеми побудовані.

Концепція автомата реального часу – це спосіб опису систем реального часу, введений у [30]. Графова модель автомата доповнюється скінченою множиною таймерів, які приймають натуральні величини. Візуальним описом моделі часового автомата є темпоральний граф переходів. Такий граф розширюється таймером, який використовується для затримок у станах. Таймер використовується для перебування в стані впродовж певної кількості тактів синхросигнала.

Є два типи темпоральних переходів – часові та умовно-часові. На рисунку 2.1 наведено фрагмент автомата з часовими та умовно-часовими переходами. У нижній частині представлено розгорнуті графи з використанням явного значення таймера в описі умов. Перехід між станами A та B – це звичайний часовий перехід, за якого автомат залишається в стані A упродовж T циклів синхросигнала. Умова написана у формі T - 1, оскільки

внутрішній таймер прийматиме значення від 0 до t = T - 1. Перехід BA залежить від стану допоміжного таймера і вхідного сигналу X. Отже, автомат перебуватиме в стані B принаймні впродовж T - 1 тактів. Такий тип переходів називається умовно-часовими.

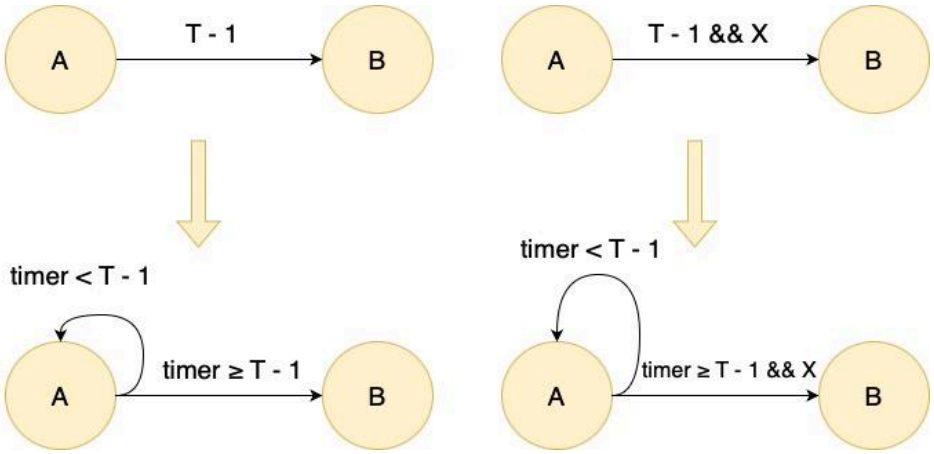


Рисунок 2.1 – Приклад темпоральних переходів

Питання опису HDL-шаблонів ґрунтовно розглянуто в роботах [38-39]. Зокрема, у цих роботах розглянуто обробку однотактових подій. Варто зауважити, що є клас пристроїв, функціональність яких залежить не тільки від зовнішньої поді, але і від її часових характеристик. Наприклад, затискання клавіші вимикає пристрій; тривалість затискання кнопки на пульті керування світлофором переводить його в різні режими роботи. Цей клас пристроїв вимагає відображення в наявних моделях на базі кінцевих автоматів.

2.2 Класифікація зовнішніх подій та методи їхньої обробки

Уточнення класифікації зовнішніх подій запропоновано у [40]. Усі події діляться на 3 класи: бізнес-події, сигнали та часові події. Ця класифікація є важливою під час створення специфікації цифрового пристрою. Іншим важливим аспектом є питання обробки подій у HDL-коді, а саме питання синхронізації, оскільки події можуть бути синхронними та асинхронними.

Синхронними подіями будемо називати події, обробка яких залежить від фронту синхросигнала; асинхронними – ті, які не залежать від фронту синхросигнала. Яскравим прикладом такого роду події є сигнал скидання (reset), який може бути як асинхронним, так і синхронним. Вибір між синхронним або асинхронним скиданням залежить від характеру логіки, що проєктується, та вимог проєкту. На рисунку 2.2 наведено приклад VHDL-коду із синхронним (ліва частина) та асинхронним сигналом скидання *rst* (права частина).

```
    process(clk)
    begin
        if rst = '1' then
            state <= a1;
            count <= "0000";
        elsif rising_edge(clk) then
            state <= next_state;
            count <= next_count;
        end if;
    end process;
```

```
    process(clk, rst)
    begin
        if rst = '1' then
            state <= a1;
            count <= "0000";
        elsif rising_edge(clk) then
            state <= next_state;
            count <= next_count;
        end if;
    end process;
```

Рисунок 2.2 – Приклад асинхронного та синхронного скидання

З погляду автоматних шаблонів обробка зовнішніх подій відбувається в процесі, який описує комбінаційну логіку переходів і не залежить від синхросигнала. У часових автоматах обробку зовнішніх подій визначає параметр t_c (timing constaint, часове обмеження) – період часу, упродовж якого FSM у стані a_i може обробляти вхідні події. Цей параметр визначається в циклах синхросигналу й обчислюється як $t_c = (t_1–t_0)$. При $t_1 = \infty$ *і* $t_0 = 0$ автомат не має часових обмежень на обробку вхідних сигналів. Якщо зовнішня подія сталася за межами «вікна» часових обмежень – автомат не реагує на неї.

2.3 HDL шаблони обробки зовнішніх подій у пристроях реального часу

2.3.1 Обробка зовнішніх подій з мінімальною тривалістю

Визначимо, що часові обмеження події – це значення $t_c(X(i)) = [c_1]$, що позначає мінімальний час тривалості певного вхідного сигналу. У прикладі з кнопкою блокування мобільного телефону $t_c = 3$ (для простоти викладення це значення наведено в секундах). Це обмеження покажемо на темпоральному графі як перехід. В умові цього переходу необхідно враховувати значення таймера зі значенням $t_c(X(i))$.

На рисунку 2.3 наведено приклад такого переходу. Як і 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 a_2/a_3 необхідно, аби сигнал події *evnt* був активним (приймав значення високого рівня, тобто ‘1’) та таймер був більшим за значення $t_c(evnt)$. Введемо два логічні підстани стану a_2: *a2/reset ma a2/inc*. У разі, якщо сигнал *evnt* приймає значення високого рівня, але значення таймера менше за $t_c(evnt)$, то автомат переходить у підстан *a2/inc*. У цьому стані таймер інкрементує внутрішній стан. Якщо сигнал evnt приймає значення низького рівня (‘0’) та значення таймера менше за $t_c(evnt)$ – автомат опиняється у підстані *a2/reset*, у якому таймер скидається у 0. Важливо зауважити, що необхідно мати стан, перехід у який не буде залежати від таймера. Це є важливим для уникнення циклів. У даному прикладі таким станом є a_1, у який автомат переходить при високому рівні сигналу *rst*.

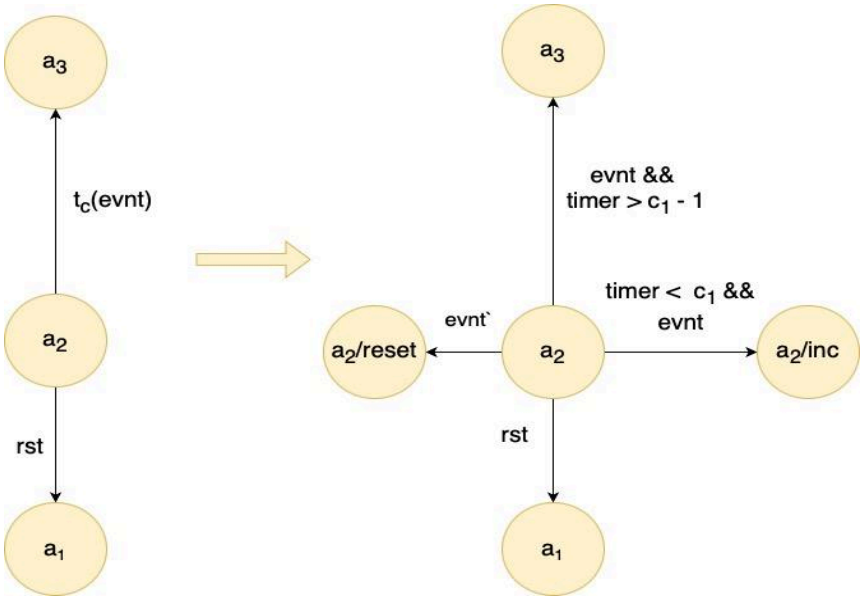


Рисунок 2.3 – Приклад темпорального переходу з подією

Залежно від характеру зовнішньої події, можна виділити чотири можливі типи поведінки, що відповідають або порушують часову вимогу. Для подальшого розгляду припустимо, що сигнал *evnt* мусить залишатися на високому рівні упродовж не менше ніж 4 тактів сигналу *clk*. Перший сценарій передбачає, що зовнішня подія триває рівно необхідний час. У цьому разі автомат переходить у наступний стан, і внутрішній лічильник скидається. На часовій діаграмі на рисунку 2.4 показано, як сигнал *evnt* залишається на високому рівні упродовж 4 тактів синхросигналу *clk* (мінімально необхідна тривалість для зміни станів), після чого автомат переходить у новий стан a_3. Червоними пунктирними лініями позначено початок та завершення обчислення події внутрішнім таймером.

Другий сценарій відповідає ситуації, коли подія триває довше, ніж t_c відповідного стану. Важливо зазначити, що для класу пристроїв, згаданих у вступі, більша тривалість події не є критичною, оскільки виконується умова мінімальної тривалості події. Так само, як і в попередньому випадку, автомат перейде в наступний стан зі скиданням лічильника.

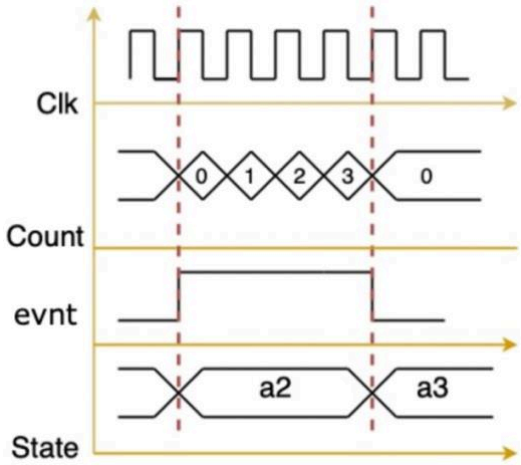


Рисунок 2.4 – Часова діаграма темпорального переходу з подією, яка триває рівно необхідну кількість тактів

На рисунку 2.5 показано часову діаграму для такого випадку. Згідно з діаграмою, сигнал *evnt* залишається на високому рівні упродовж 5 тактів, що на 1 такт довше від необхідного значення. Як і в попередньому випадку, автомат перейде в новий стан і розпочне знову лічильник. Відрізок, на якому сигнал *evnt* має позитивне значення, позначений зеленим пунктиром, а момент зміни стану автомата – червоним.

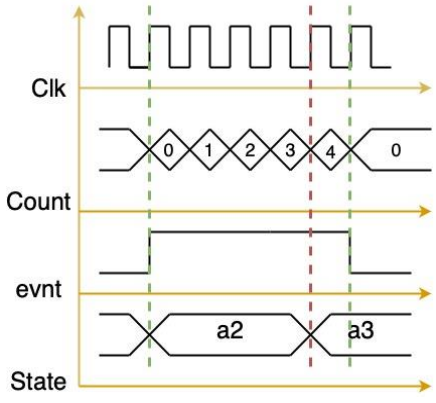


Рисунок 2.5 – Часова діаграма темпорального переходу з подією, тривалістю більше ніж необхідно

Третій сценарій відображає ситуацію, коли подія триває менше вказаного часу. Тут автомат залишається в тому самому стані зі скиданням лічильника, як показано на рисунку. 2.6. Скидання лічильника є важливою операцією, оскільки таймер повинен бути готовий до обробки наступних потенційних подій.

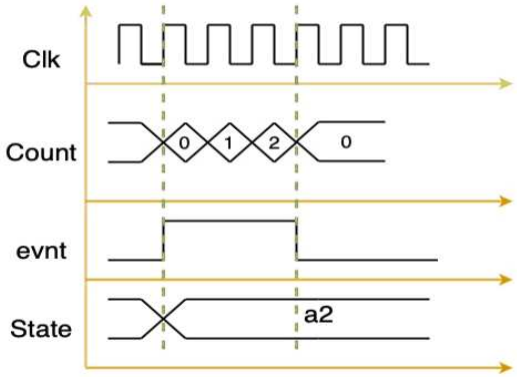


Рисунок 2.6 – Часова діаграма темпорального переходу з подією, тривалістю менше ніж необхідно

Важливо розглянути можливість тупика, до якого може потрапити автомат, якщо зовнішня подія не триває необхідний час або взагалі не відбувається. Для цього автомат має сигнал скидання, що переведе пристрій у попередньо заданий стан. Зазвичай для цього в інтерфейсі цифрового пристрою є сигнал скидання. Варто зазначити, що цей сигнал має вищий пріоритет, ніж інші вхідні сигнали. Це особливо важливо під час написання апаратурної реалізації пристрою з використанням мов опису апаратури. На рисунку 2.7 наведена часова діаграма, що описує цю ситуацію - сигнал *evnt* залишається на високому рівні одночасно із сигналом *rst*, але автомат переходить у стан a_1, оскільки сигнал *rst* має більший пріоритет.

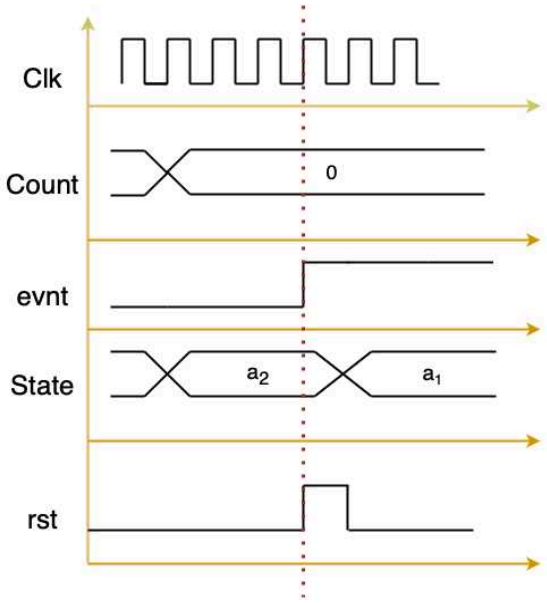


Рисунок 2.7 – Часова діаграма темпорального переходу при асинхронному скиданні

2.3.2 HDL-шаблон обробки подій з часовими вимогами

Проілюструємо теоретичні викладки з обробки зовнішніх подій із мінімальною тривалістю на прикладі моделі модуля енергозбереження. Цей модуль зазвичай використовується в системах із критичним енергоспоживанням для зменшення споживання енергії, коли система не працює впродовж певного періоду.

Модуль працює у двох режимах: байпас і енергозбереження. Вхідний алфавіт складається з таких двійкових сигналів $X = \{onn, evnt\}$, де *onn* – сигнал для включення 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. Інтерфейс модуля та його зв'язок з іншими вузлами системи показані на рисунку. 2.8.

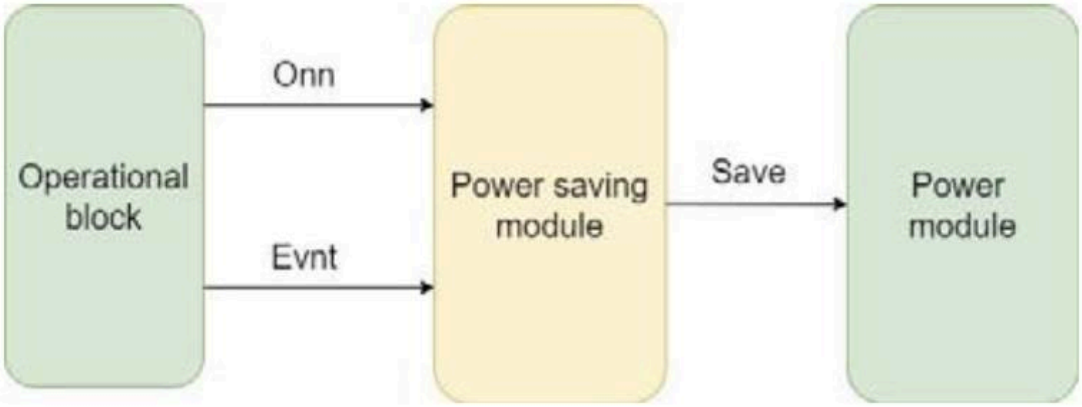


Рисунок 2.8 – Блок-схема модуля енергозбереження та його зв’язку з іншими вузлами

Наступним кроком є визначення станів автомата та часових обмежень. Набір станів такий:

1) стан a_0 – модуль працює в режимі байпас, тобто зовнішні події не відстежуються;

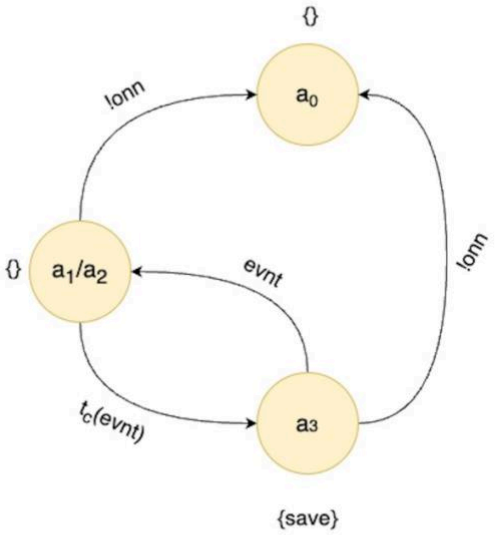
2) стани a_1/a_2 – розрахунок режиму енергозбереження – якщо впродовж фіксованого періоду часу не відбувається жодних подій, автомат переходить у стан a_3;

3) стан a_3 – режим енергозбереження – сигнал save приймає високий логічний рівень;

4) $t_c(evnt)$ – мінімальний час (у циклах синхросигнала) упродовж якого сигнал *evnt* має бути у низькому логічному значенні (0).

Алгоритм роботи модуля збереження енергії досить простий. Щоразу, коли цей блок вмикається сигналом *onn*, він починає моніторинг вхідного сигналу *evnt*. Якщо сигнал *evnt* не приймає високий логічний рівень упродовж фіксованого часу – система має перейти в режим енергозбереження (сигнал *save* приймає значення 1). Система виходить із режиму збереження в разі встановлення сигналу *evnt* або вимкнення модуля через вхідний сигнал *onn* з подальшим зняттям сигналу збереження. Відповідний темпоральний

граф переходів представлено та Verilog опис логіки переходів представлено на рисунку 2.9.




```
always @(*) begin
  next_count = 4'd0;
  case (state)
    a0: if (onn)
      next_state = a1;
    else
       next_state = a0;
    a1:
      if (!onn)
        next_state = a0;
      else if (!evnt)
        next_state = a2;
      else
        next_state = a1;
    a2:
      if (!onn)
        next_state = a0;
      else if (evnt)
        next_state = a1;
      else if (count >= C2 - 1)
        next_state = a3;
      else begin
        next_state = a2;
        next_count = count + 1'b1;
      end
    a3:
      if (evnt || !onn)
        next_state = a0;
      else
        next_state = a3;
    default: begin
      next_state = a0;
    end
  endcase
```

Рисунок 2.9 – Темпоральний граф переходів модуля енергозбереження та фрагмент Verilog опису

Відповідну часову діаграму показано на рисунку 2.10 – тут $t_c(evnt)$ дорівнює 5 тактам. Період обчислення події позначено червоними лініями; сині лінії позначають зміну сигналів (*evnt* і *onn*), що переводить автомат у режим моніторингу або байпас режим.

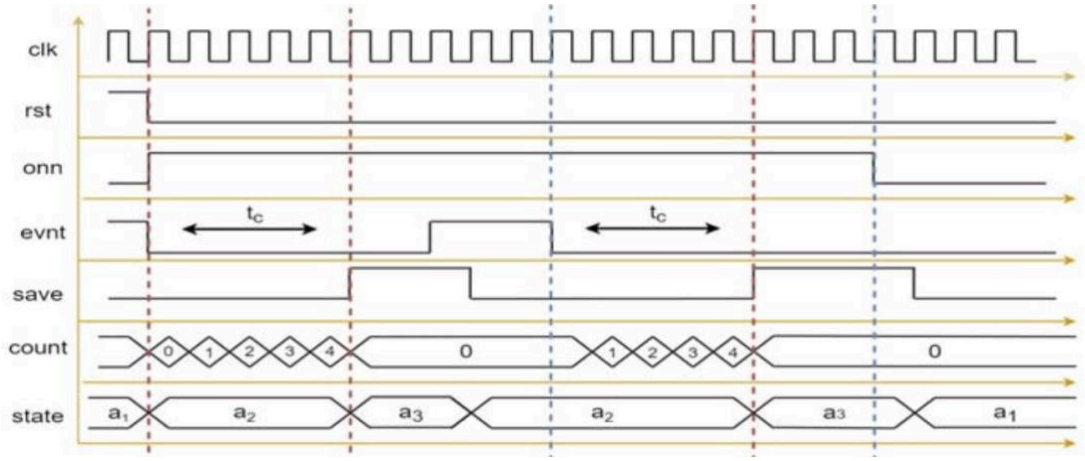


Рисунок 2.10 – Часова діаграма роботи модуля енергозбереження

Для верифікації та синтезу проєкту використовувалася САПР Xilinx ISE 14.7. Поведінкове моделювання та пост-синтез моделювання для початкового опису було виконано на CPLD XC9572XL-10-TQ100 (Post-Fit Simulation) і на FPGA XC3S100E-5vq100 (Post-Place & Route Simulation).

Затримка між переходами автомата 0 нс. Поодиноких короткочасних імпульсів не зафіксовано. Отже, під час синтезу пристрою на FPGA і CPLD його робота відповідає оригінальному опису (специфікації). Очікувана мінімальна кількість тригерів становить 5: 2 тригери для кодування 4 станів, 3 тригерів для лічильника (оскільки максимальний тайм-аут становить 3 такти). Схематичний звіт RTL для обох мікросхем підтвердив це.

У звіті відсутні латч-тригери. Для реалізації функцій переходів і виходів синтезовано комбінаційні схеми. 36 комбінаційних елемента синтезовано на XC9572-7PC44, а 24 комбінаційних елементи – на XC3S100E-5vq100.

Для FPGA: мінімальний тактовий період: 2,945 нс (максимальна частота: 339,570 МГц). Для CPLD: мінімальний тактовий період: 8,000 нс (максимальна частота: 125,000 МГц).

2.3.3 Обробка зовнішніх подій з вікном прийому

Окремим класом цифрових проєктів є пристрої, функціональність яких залежить від співвідношення вікна прийому події (t_c) та, власне, тривалості події. Такого роду моделі мають свої особливості під час побудови HDL-моделі. На рисунку 2.11 зображено граф такого роду моделі та можливих переходів за різному співвідношенні тривалості зовнішньої події та вікна прийому. Тут R – сигнал скидання, e – зовнішня подія, t_e – тривалість зовнішньої події, t_c – вікно прийому.

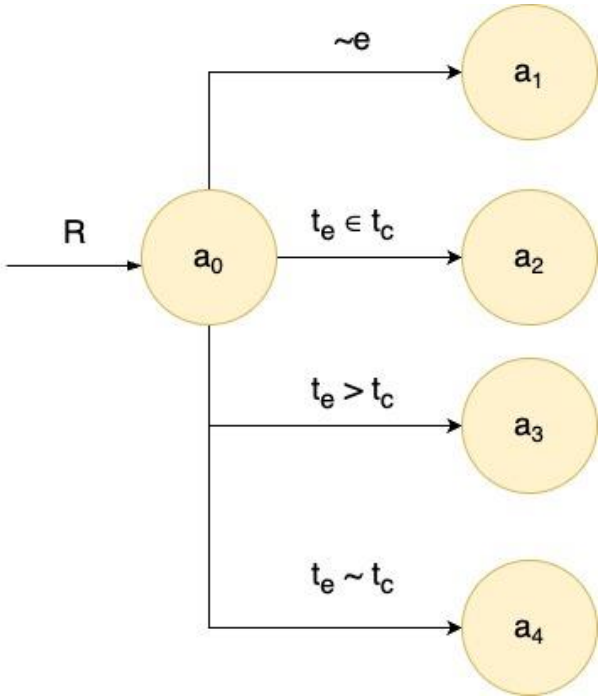


Рисунок 2.11 – Граф переходів, що залежить від тривалості зовнішньої події та її співвідношення з вікном прийому $t_c(a_0)$

Для такого типу моделей можливі чотири варіанти переходу:

– a_0 – a_1 за відсутності події в період $t_c(a_0)$;

– a_0 – a_2 за умови, що подія повністю у вікні прийому - $t_e \in t_c(a_0)$;

– a_0 – a_3 за умови, що подія частково у вікні прийому - $t_e \sim t_c(a_0)$;

– a_0 – a_4 за умови, що тривалість події більша за вікно прийому - $t_e > t_c(a_0)$;

Вищеописані сценарії поведінки представлено на рисунку 2.12. Тут жовтим кольором позначено вікно прийому.

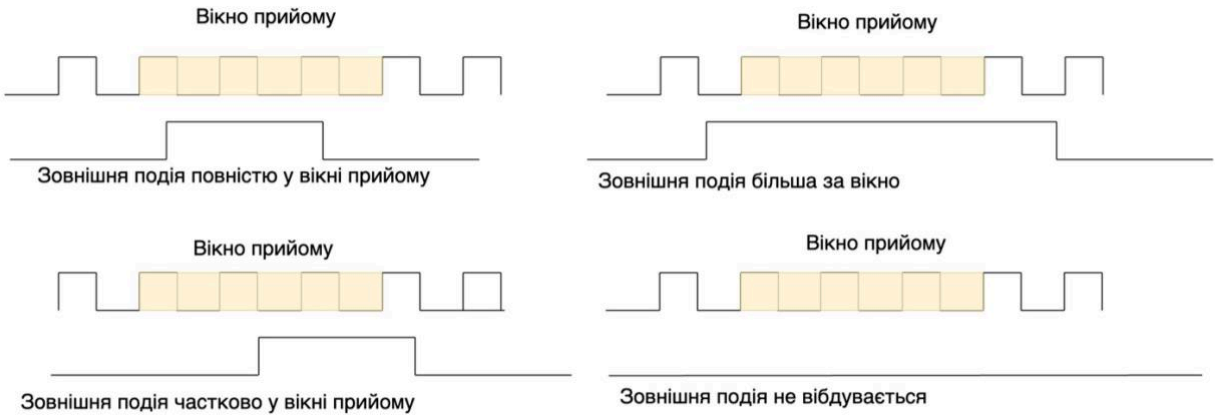


Рисунок 2.12 – Співвідношення вікна прийому та тривалості події

Для простоти побудови HDL опису (зменшення конструкцій if-else) розширимо граф 2.11 додатковими станами.

a_0 – початковий стан, у якому автомат перебуває до початку вікна прийому. Під час настання вікна прийому відбувається перехід у стан a_1 при відсутності події або в стан a_4 (початок події відбувся до вікна прийому).

a_1 – стан виявлення початку події (передній фронт) у вікні прийому. Перехід у стан a_2 за виявлення переднього фронту; у стан a_7 – за відсутності події.

a_2 – виявлення кінця події. Перехід у стан a_3, якщо кінець події відбувся у вікні прийому, інакше – у стан a_6.

a_3 – стан, який показує, що подія відбулася у вікні прийому.

a_4 – виявлення кінця події при умові, що початок припав за вікном прийому. Перехід у стан a_5, якщо кінець події відбувся у вікні прийому, інакше – у стан a_6.

a_5 – стан, який показує, що подія частково відбулася у вікні прийому.

a_6 – стан, який показує, що тривалість події поза вікном прийому.

a_7 – стан, який показує відсутність події під час вікна прийому.

Оновлений граф переходів зображено на рисунку 2.13.

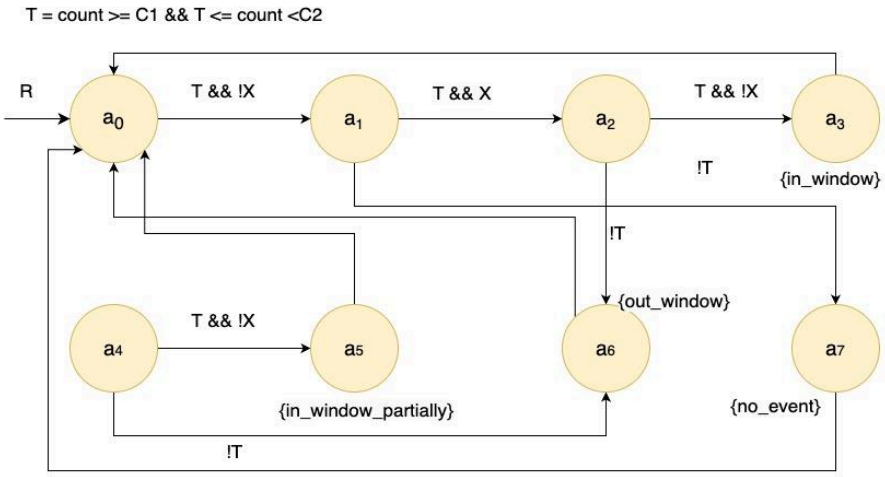


Рисунок 2.13 – Оновлений граф переходів, що залежить від тривалості зовнішньої події та її співвідношення з вікном прийому $t_c(a_0)$

2.3.4 HDL-шаблон обробки зовнішніх подій з вікном прийому

Для ілюстрації запропонованих теоретичних викладок побудуємо HDL шаблон вищезазначеної моделі з подальшим моделюванням та синтезом. На рисунку 2.14 показано Verilog шаблон обробки зовнішніх подій із вікном прийому.

```
always @(*) begin
  next_count = 3'd0;
  case (state)
    a0:
      if (count >= C1 && count <= C2) begin // вікно
        if (~btn) begin
          next_state = a1;
          next_count = count + 1'b1;
        end else
        begin
          next_state = a4;
          next_count = count + 1'b1;
        end
      end
      else begin
        next_state = a0;
        next_count = count + 1'b1;
      end

    a1:
     if (count <= C2) begin // вікно
        if (btn) begin // posedge detected
          next_state = a2;
          next_count = count + 1'b1;
        end else begin
          next_state = a1;
          next_count = count + 1'b1;
        end

      end else
        next_state = a7;
```

```
    a2:
      if (count <= C2) begin // вікно
        if (~btn) begin // negedge detected
          next_state = a3;
        end
        else begin
          next_state = a2;
          next_count = count + 1'b1;
        end
      end
      else
          next_state = a6;

    a3: next_state = a0; // event in window
    a4:
     if (count <= C2) begin // вікно
        if (~btn) begin // кінець події
          next_state = a5;
        end else begin
          next_state = a4;
          next_count = count + 1'b1;
        end
      end else begin
          next_state = a6;
      end

    a5: next_state = a0;
    a6: next_state = a0;
    a7: next_state = a0;

    default: next_state = a0;

  endcase
end

assign in_window = state == a3;
assign in_wnd_part = state == a5;
assign out_window = state == a6;
assign no_evnt = state == a7;
```

Рисунок 2.14 – Verilog шаблон обробки подій з вікном прийому

Розглянемо більш ґрунтовно можливі сценарії поведінки та відповідні часові діаграми. Для подальших викладок приймемо, що вікно прийому зовнішньої події t_c = [1; 5] тактів синхросигнала. На рисунку 2.15 зображено перший випадок поведінки, коли зовнішня події повністю у вікні прийому. Червоними стрілками показано початок та кінець зовнішньої події *btn*. На цій діаграмі видно, що подія перебуває повністю у вікні прийому (початок події припадає на значення таймера 2, а кінець – на 5). Одразу після кінця події автомат переходить у стан a_3 та сигнал *in_window* приймає високий логічний рівень (подія повністю у вікні прийому).

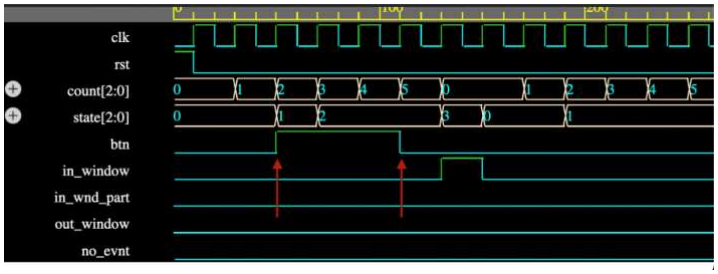


Рисунок 2.15 – Фрагмент часової діаграми, коли подія повністю у вікні прийому

Ситуацію, коли подія частково у вікні прийому зображено на рисунку 2.16. Видно, що початок події перебуває за вікном прийому (початок події припадає на значення таймера 0), а її кінець – на 5. Одразу після кінця події автомат переходить у стан a_5 та сигнал *in_wnd_part* приймає високий логічний рівень (подія повністю у вікні прийому).

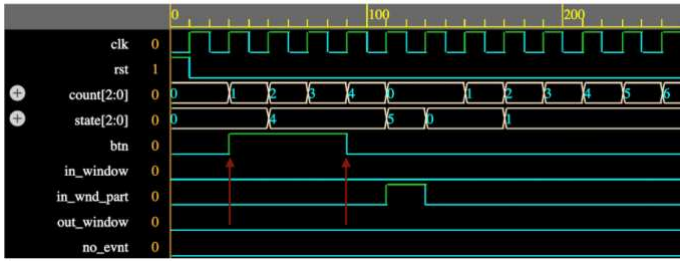


Рисунок 2.16 – Фрагмент часової діаграми, коли подія частково у вікні прийому

Часова діаграма на рисунку 2.17 описує ситуацію, коли подія виходить за вікно прийому. Видно, що початок події і кінець події знаходяться за вікном прийому (початок події припадає на значення таймера 0), а її кінець – на 5. Одразу після того, як внутрішній лічильник (*count*) приймає значення 6 (*count* > *C2*) автомат переходить у стан a_6 та сигнал *out_wnd* приймає високий логічний рівень (подія за вікном прийому).

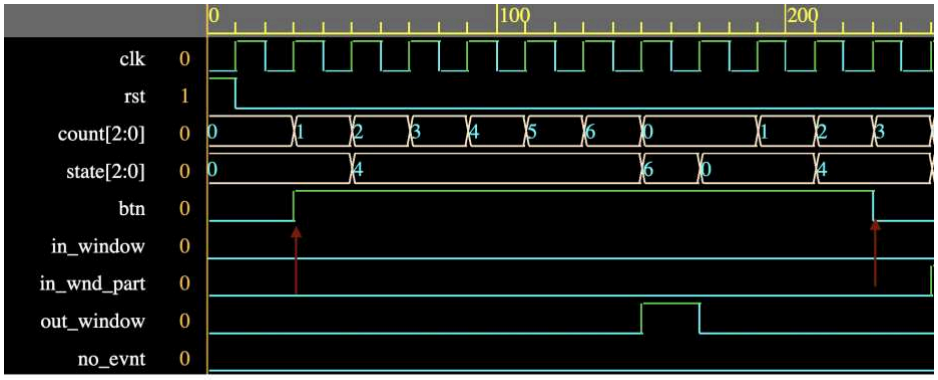


Рисунок 2.17 – Фрагмент часової діаграми, коли подія за вікном прийому

Останній сценарій – подія не відбувається впродовж вікна прийому. Як видно з діаграми на рисунку 2.18, у такому разі автомат переходить у стан a_7 зі збудженням сигналу *no_evnt*.

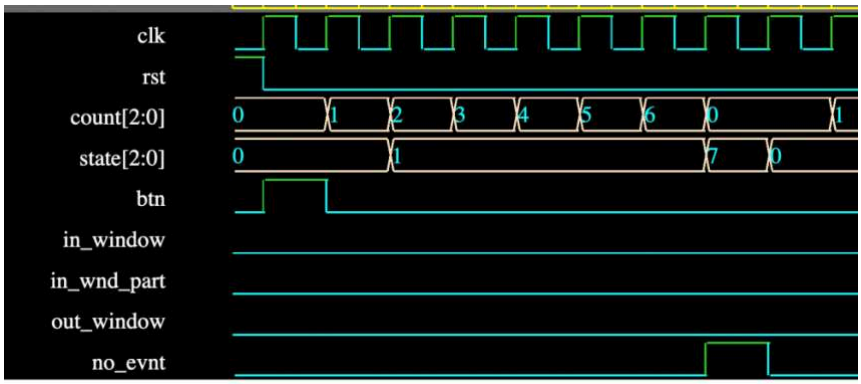


Рисунок 2.18 – Фрагмент часової діаграми, коли подія відсутня у вікні прийому

Синтез запропонованої моделі було проведено в середовищі Vivado з використанням SoC сімейства Zynq-7000. На рисунку 2.19 представлено результати фрагмент результатів синтезу рівня регістрів (RTL). Під час синтезу було використано 11 FF тригерів (8 для кодування станів унітарним кодом там 3 для внутрішнього таймера) та 10 LUT для реалізації комбінаційної частини логіки.

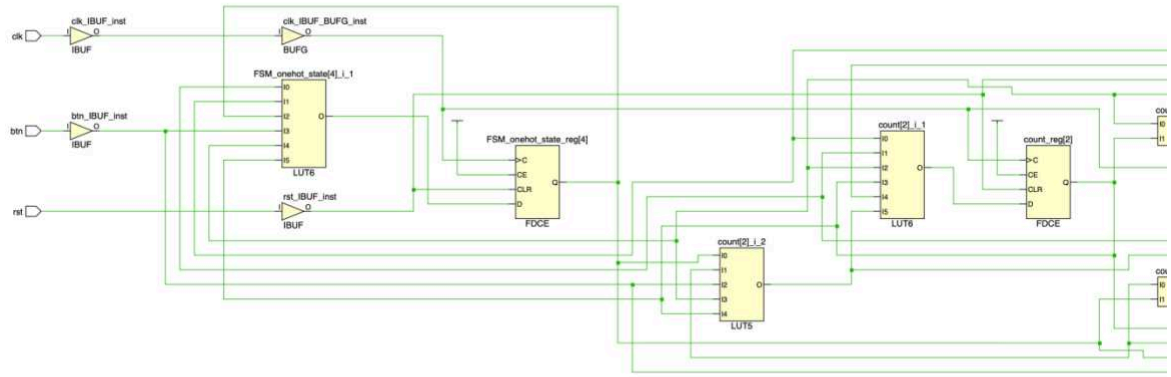


Рисунок 2.19 – Фрагмент результатів синтезу для SoC Zynq-7000

2.4 Висновки до розділу 2

1. Розширено клас пристроїв реального часу – пристрої, які функціонують на базі подій із певною тривалістю в часі та пристрої, функціональність яких залежить від співвідношення тривалості події та вікна прийому.

2. Розроблено HDL-моделі обробки зовнішніх подій із тривалістю. Моделювання проводилося з використанням системи моделювання EDA Playground.

3. Верифікацію запропонованих HDL моделі було проведено у двох САПР: Vivado з використанням SoC сімейства Zynq 7000 та Xilinx з використанням FPGA XC3S500E-5fg320 та CPLD XC9572XL-10-TQ100. Автоматизований синтез та пост-синтез моделювання підтвердило працездатність запропонованих шаблонів.

3 АСЕРЦІЙНЕ ПРОЄКТУВАННЯ ТА ДІАГНОСТУВАННЯ ЧАСОВИХ АВТОМАТІВ

Третій розділ роботи присвячено проєктуванню систем реального часу з допомогою асерцій та методів формальної верифікації.

У процесі дослідження необхідно вирішити такі питання:

1) визначити місце формальних методів верифікації в сучасному процесі проєктування;

2) розробити методика розстановки асерцій у HDL-моделях пристроїв реального часу;

3) розробити шаблони асерційних конструкцій для верифікації часових параметрів пристроїв реального часу.

3.1 Узагальнена модель верифікації цифрових пристроїв

У зв'язку з інтенсивним розвитком цифрових пристроїв і ростом їхньої складності перевірка цифрових пристроїв стає все більш актуальною темою. Високі витрати, зумовлені трудоємкістю перевірки функціонально і структурно складних схем, можуть досягати 70% від загального часу розробляння проєкту. Використання мов опису апаратури дозволило підвищити рівень абстракції у процесі проєктуванні. Водночас найнижчим рівнем деталізації цифрових пристроїв з погляду автоматизації проєктування є рівень реєстрових передач (RTL). Модель для тестування цифрової системи HDL-коду представлена таким xor-відношенням між параметри специфікацією F, реалізацією T та несправностями F (помилками проєктування):

$$L \oplus T \oplus \mathrm{F} = 0, \tag{3.1}$$

Верифікація – це процес перевірки того, що проектована система (тобто реалізація) поводиться відповідно до заданого набору вимог (специфікацію) й описується як T⊕L=F.

Тестування – процес знаходження помилок проєктування для заданого класу несправностей із використання вхідного вектору та аналізом вихідного вектору F=T⊕L.

Асерційна точка – спеціальна конструкція для перевірки відповідності поведінки цифрового пристрою заданій специфікації.

Зокрема, у [100] описано технологію діагностики SoC HDL-моделі, що ґрунтується на графіку транзакцій коду (Code-Flow Transaction Graph). Запропоновано метод діагностики, спрямований на зменшення часу виявлення несправності та пам'яті для зберігання діагностичної матриці, через формування трикомпонентних зв'язків між тестом, монітором (асерціями) та функціональними компонентами. Запропоновано модель розстановки асерції на основі аналізу ABC-графу, який представляє HDL-код.

3.2 Формальна верифікація та її місце у проектуванні

Є два способи верифікації моделей цифрових пристроїв. Більш традиційним способом є моделювання (simulations). Такий спосіб передбачає подання певних вхідних послідовностей (testbench) та аналіз відповідних вихідних векторів (зазвичай на часових діаграмах). Цей спосіб є все ще найбільш використовуваним та призвів до виникнення методологій, як-от Universal Verification Methodology, та мов для проєктування комплексних середовищ верифікації, які включають у себе автоматичну генерацію тестових векторів, валідацію вихідних послідовностей та аналіз кодового покриття. Основною вадою такого виду верифікації є збільшення складності процесу верифікації, а відповідно й часу проведення, зі збільшенням розміру проєкту.

Альтернативним підходом є формальна методологія, яка використовує специфікації/обмеження (записані як властивості) і використовує математичний апарат для доведення (або спростування), що проєктована система відповідає цим вимогам. Якщо описана властивість виявляється неправдивою, інструмент надасть контрприклад з усіма вхідними даними, необхідними для моделювання несправності.

Власне формальна верифікація складається з трьох основних кроків: 1) компіляція формальної моделі проєктованого пристрою; 2) написання точної та несуперечливої специфікації; 3) застосування автоматизованого алгоритму перевірки моделі.

На першому етапі формального процесу перевірки властивостей (property checking) проєктувальники створюють формальну модель дизайну через компіляцію HDL опису (наприклад, модель Verilog RTL) у форму прийнятну для алгоритму перевірки. Для простоти подальших викладок приймемо, що цифрові системи – це паралельні системи з дискретним числом станів. Наприклад, значення поточного стану системи може бути визначено в певний момент часу через аналіз стану всіх елементів системи. Наступний стан системи може бути обчислений як функція, що залежить від поточного стану системи та вхідних значень. Пара поточний-наступний стани описує одне конкретне відношення переходу системи. Наприклад, (s_i, s_{i+1}) - це відношення переходу, де представляє поточний стан системи s_i та безпосередньо досяжний стан s_{i+1}. Зазвичай сукупність усіх можливих переходів між станами описується через певні структури даних, наприклад, бінарного дерева.

Шлях у стані *s* – це нескінченна послідовність станів $\pi=s_0s_1s_2s_3\ldots$, яка представляє прогрес у часі та послідовність станів. Набір шляхів представляє поведінку системи. Отже формальна модель може бути створена через компіляцію синтезованої моделі цифрової системи у граф переходів, відомою як модель Кріпке.

Модель Кріпке – це кортеж із 4 елементів *(S, S_0, R, L),* де *S* – це кінцева множина станів; S_0 – множина початкових станів; $R \subset S \times S$ – відношення переходів; $L - S \rightarrow 2^{AP}$ – функція міток, яка помічає кожний стан множиною тверджень, які виконуються в цьому стані. Така моделює систему через граф, де вершина позначає стан, а ребро – перехід між станами [102].

На наступному етапі формальної перевірки властивостей необхідно визначити властивості як твердження дизайну, які необхідно перевірити. Властивість - сукупність логічних і часових зв'язків між підпорядкованими булевими виразами, послідовностями та іншими властивостями, які в сукупності представляють набір поведінки (шлях) [94]. Є два класи властивостей – безпечні (safety) та недетерміновані властивості (liveness).

Безпечна властивість – властивість, яка визначає інваріант над станами в дизайні. Інваріант не обов'язково обмежується одним циклом, але він обмежений у часі. Іншими словами, така властивість стверджує, що система не може увійти в невалідний стан. Наприклад, властивість «сигнали *wr_en* і *rd_en* є взаємовиключні" та "щоразу, коли сигнал *req* приймає високий логічний рівень, сигнал *ack* приймає високий логічний рівень упродовж 3 циклів» є безпечними властивостями.

Недетермінована властивість – властивість, яка описує можливу подію недетерміновану в часі. Іншими словами, така властивість описує, що «щось хороше» зрештою станеться. Прикладом такого типу властивостей є твердження "щоразу, коли сигнал *req* приймає високий логічний рівень, сигнал *ack* приймає високий логічний рівень у майбутньому».

Здебільшого мови опису властивостей використовують формалізм відомий як пропозиційно-темпоральна логіка, який дає змогу аналізувати послідовності переходів між станами. Розгалужена у часі темпоральна логіка (branching-time temporal logic, CTL) та лінійно-часова логіка (linear-time logic, LTL) – класи формалізмів для опису послідовностей. CTL – приклад логіки розгалуженої в часі. Темпоральні оператори цього формалізму дозволяють

аналізувати про всі шляхи-наступники даного стану. Темпоральні оператори LTL дають змогу аналізувати події в одному шляху обчислення.

Після створення формальної модель, що представляє цифровий пристрій, і формальної специфікації, що точно описує властивості, які необхідно перевірити, наступним кроком є застосування автоматичного алгоритму доказу. Наприклад, маючи формальну модель цифрового пристрою як моделі Кріпке $M = (S, S_0, R, L)$ та формулу темпоральної логіки, яка описує певну бажану властивість пристрою, проблему доказу правильності включає знаходження множини всіх станів з S, які б задовольнили f:

$$\{s \in S \mid M, s \models f\}, \tag{3.2}$$

де $s \models f$ означає, що властивість, представлена темпоральною формулою f, виконується у стані s.

Варто зазначити, що формальна модель задовольняє специфікацію тоді й тільки тоді, коли всі початкові стани містяться в множині всіх станів, які задовольняють f. Рисунок 3.1 ілюструє (на високому рівні) один алгоритм перевірки, який використовується для знаходження набору всіх станів у S, які задовольняють f.

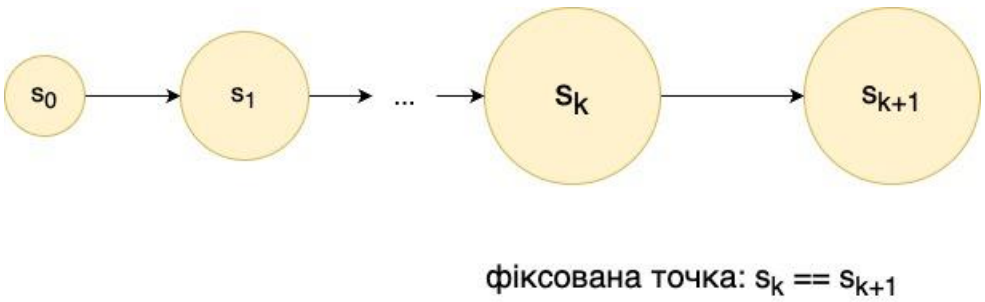


Рисунок 3.1 - Досяжні стани з фіксованою точкою

Вищенаведений алгоритм доказу відомий як аналіз досяжності через обчислення образів. Алгоритм починається зі множини початкових станів S_0,

як показано на малюнку 3.1. Усі досяжні стани з S_0 обчислюються використовуючи відношення переходу R за один крок (цикл синхросигнала). Процес обчислення називається обчисленням образів. У цьому прикладі S_1 - новий набір досяжних станів. Процес повторюється ітеративно, створюючи новий набір досяжних станів на кожному кроці, доки не буде виявлено нових досяжних станів (тобто фіксована точка виникає, коли $S_k = S_{k+1}$).

Важливо сказати, що для безпечних властивостей, описаних темпоральною формулою f, можливо підтвердити, що f виконується для кожного нового стану, обчисленого упродовж ітерації процесу обчислення образів.

Для алгоритму перевірки з фіксованою точкою можливо три різних результати.

1. Процес досягає фіксованої точки, та f виконується на всіх досяжних станах.

2. Процес не досяг фіксованої точки, та f не виконується для певного стану s_i,.

Отже, можна знайти контрприклад (тобто шлях $\pi=s_0s_1s_2s_3\ldots s_i$), який використовується для усунення проблеми.

3. Процес переривається до досягнення фіксованої точки через умову, відому як вибух станів (state explosion), тобто є забагато станів, щоб механізм перевірки міг представити в пам’яті.

Важливим аспектом процесу формальної верифікації є власне розстановка асерційних точок. Раніше було визначено, що асерційна точка є певною конструкцією для перевірки відповідності дизайну специфікації. Процес розстановки асерцій передбачає виявлення властивостей цифрового пристрою (design property) та їхній опис, використовуючи відповідні мови.

Є ряд формалізованих підходів визначення HDL-конструкцій та блоків, які необхідно покрити асерціями. Наприклад, у [101] пропонується стратегія розміщення асерційних точок для керуючих та операційних автоматів на основі аналізу графу переходів та досяжності вершин. Ця методика є дієвою

для класичних автоматів, але не підходить для часових автоматів, де переходи мають темпоральну природу. Окрім того, коли ідеться про системи реального часу важливим аспектом є питання часу реакції системи на вхідну подію. Зазвичай, часові параметри чітко описані в специфікації, яка є вхідною точкою проєктування та основним джерелом властивостей системи.

У розділі 2 було показано, що повною математичною моделлю часового автомата є темпоральний граф переходів. На такому графі явно описано затримки в станах у тактах синхросигнала, які надалі використовуються у HDL-описі. Пропонується розширити такий граф мітками, які представляють асерційні точки. Такий універсальний підхід дає змогу розподілити процес написання HDL-коду та коду верифікації. На рисунку 3.2 представлено темпоральний граф функціонування світлофора, запропонований у [32], доповнений асерційними точками.

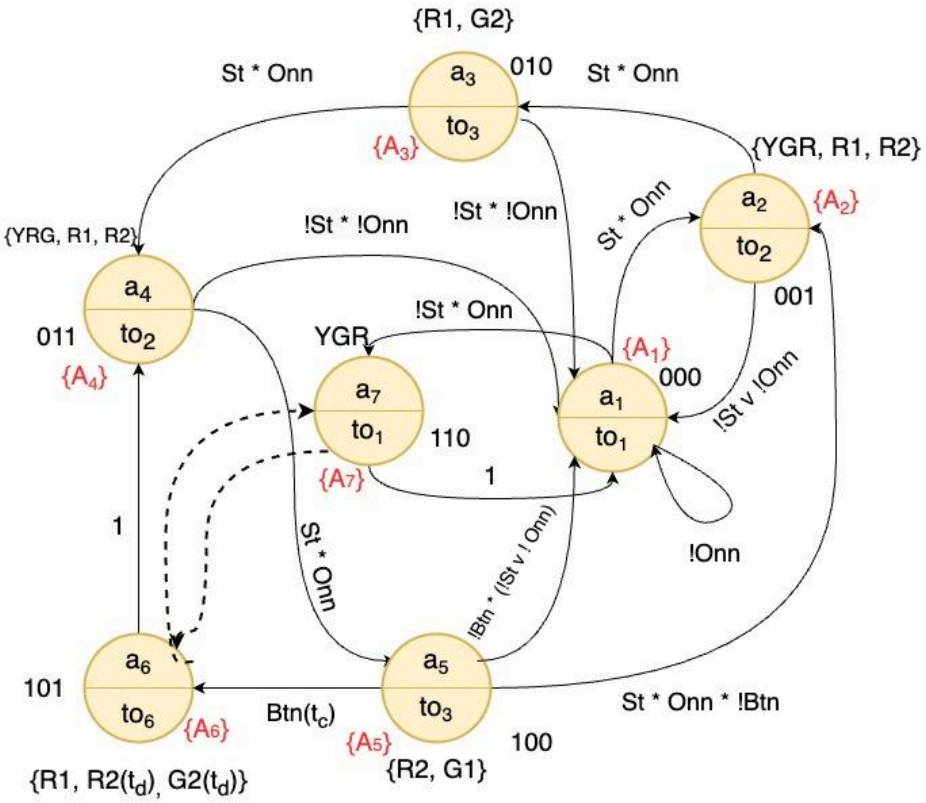


Рисунок 3.2 – Темпоральний граф, доповнений асерційними точками

3.3 Асерційна верифікація темпоральних параметрів автоматів

3.3.1 Асерційні конструкції верифікації темпоральних переходів

Правильність реалізації HDL-опису часового автомата визначається правильним описом темпоральних переходів. Темпоральні переходи, також передбачають що автомат залишається в певному стані впродовж детермінованої кількості циклів. Наведене вище твердження є прикладом властивості цифрового дизайну, тобто умови дизайну, необхідної для його належного функціонування. Цей підрозділ присвячено опису темпоральних властивостей мовою SystemVerilog Assertions (SVA) та їх подальшому використанню під час асерційної перевірки.

Для опису темпоральних властивостей використовуються такі конструкції SVA: 1) послідовність – представляє послідовність подій, охоплених у часі, виражених у тактах синхросигнала; 2) властивість – конструкція, яка об’єднує різні послідовності операторами SVA; 3) властивість assert – представляє точку перевірки, яка використовується для перевірки властивості під час моделювання або в процесі формальної верифікації.

Для ілюстрації подальших викладок будемо використовувати автомат, який описується темпоральним графом на рисунку 3.3.

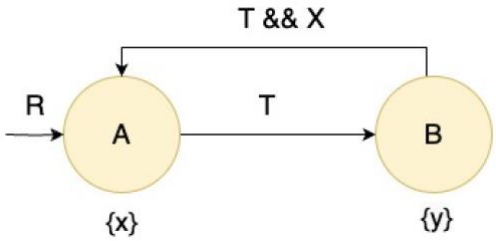


Рисунок 3.3 – Темпоральний граф часового автомата

Як було зазначено в розділі 2, є два основних видів темпоральних переходів: темпоральні та умовно-темпоральні. Властивість часового

переходу складається з трьох компонентів: 1) передумова – логічний вираз, що описує стан автомата перед часовим переходом; 2) затримка в стані – конструкція SVA для перевірки того, що автомат залишається в певному стані упродовж N тактових циклів; 3) перехід – останній компонент, який перевіряє, що FSM переходить у попередньо визначений стан, указаний у специфікації. На рисунку 3.4 показано темпоральний перехід (ліва частина) та відповідні SVA компоненти.

```
sequence AB_precondition;
  ( (state != next_state) && (next_state == A));
endsequence


property trans_prop;
  disable iff(!rst_n)
  @(posedge clk) AB_precondition |=> state == A[*2] ##1 state == B;
endproperty
```

Рисунок 3.4 – SVA опис темпорального переходу

Результати моделювання для автомата з рисунку 3.3 показані на рисунку 3.5. Тут зелені стрілки позначають початкову точку обчислення асерції, червоні – кінцеву точку, де вона досягає успіху. Діаграма показує, що автоматичний автомат залишається в стані *A* упродовж періоду від 30 нс до 70 нс, що відповідає 2 тактам із періодом такту, рівним 20 нс. Ці моменти є важливим для подальшої відладки.

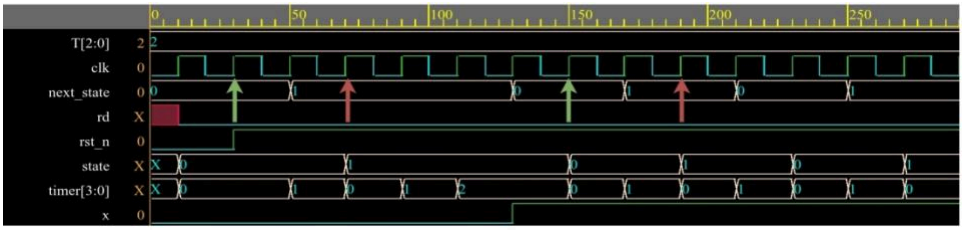


Рисунок 3.5 – Моделювання темпорального переходу автомата з відповідним позначенням точок обрахування асерції

Схожим чином описується темпорально-умовний перехід. Як і звичайний часовий перехід, властивість такого переходу матиме ті самі частини (передумова, зациклювання та перехід). Однак частина переходу має бути описана як окрема властивість через нечіткі часові параметри, тобто перехід відбувається після того, як уся умова виконується, а не після N тактових циклів. На рисунку 3.6 *trans_property_3* використовує специфікацію необмеженого часу в послідовності, оскільки перехід у стан *A* може відбутися в будь-який час у майбутньому. Ця послідовність використовується як аргумент у функції *strong*, яка робить цю властивість сильною. Сильна послідовність виконується тоді й тільки тоді, коли основна послідовність має збіг [99].

```
property trans_prop_2;
  disable iff(!rst_n)
  @(posedge clk) BA_precondition |=> (state == B)[*2];
endproperty

property trans_prop_3;
  disable iff(!rst_n)
  @(posedge clk) BA_precondition |-> strong(##[1:$](state == A));
endproperty
```

Рисунок 3.6 – SVA опис умовно-темпорального переходу

Відповідні результати моделювання переходу *AB* представлені на рисунку 3.7. Як і в попередньому прикладі, стрілки позначають початок/кінець оцінки асерції: зелена – початкова точка, червона – успішне перебування в стані (*trans_prop_2*), жовта – перехід у новий стан. Діаграма показує, що автомат залишається в стані *B* упродовж 4 тактів (від 70 нс до 150 нс). Він переходить у стан *A* на наступному позитивному фронті синхросигнала після того, як сигнал *x* приймає високе логічне значення. Наступного разу автоматичний автомат залишається в *B* лише впродовж 2 циклів (від 190 нс до 230 нс) і негайно переходить в *A*, оскільки сигнал *x* має високий рівень.

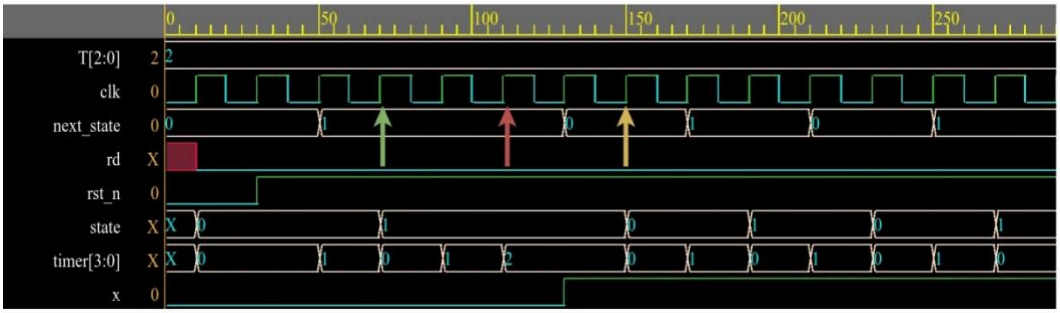


Рисунок 3.7 – Моделювання умовно-темпорального переходу автомата з відповідним позначенням точок обрахування асерції

Останнім і найважливішим кроком є специфікація рівня перевірки, яка визначає поведінку середовища САПР у разі збою властивості. Здебільшого команда тестувальників зацікавлена в докладних повідомленнях про помилки. Таким чином, необхідно використовувати властивість з оператором *assert*. У разі порушення властивості під час моделювання цей оператор видасть повідомлення. Для прикладу розглянемо помилку у HDL описі з рисунку 3.8, коли замість зациклювання у стані *A* автомат переходить у стан *B*. Наприклад, САПР Aldec Riviera Pro 2020.04 реєструє таке повідомлення в разі порушення асерції: ASSERT: *Assertion FAILED at time: 50ns (3 clk), scope: testbench.inst, start-time: 30ns (2 clk)*. Інженери можуть використовувати ці повідомлення для аналізу значень внутрішніх регістрів і сигналів у конкретний момент. На цьому етапі процес усунення помилок HDL стає подібним до процесу відладки програмного забезпечення, написаного на мові програмування C++ або Java.

```
case (state)
  A: if (timer >= T - 1)
       next_state = B;
  else
       next_state = B;
```

Рисунок 3.8 – Приклад помилкового опису переходів

3.4.2 Асерційні конструкції верифікації обробки зовнішніх подій

У розділі було введено HDL моделі обробки зовнішніх подій, а саме обробка подій із вікном прийому та подій із часовими вимогами. Розглянемо перший різновид модель та відповідні асерційні конструкції. Для ілюстрації запропонованих моделей будемо використовувати модель модуля енергозбереження, представлений у розділі 2.

Першим кроком необхідно описати початковий стан моделі. На рисунку 3.9 представлено конструкцію *property*, яка описує передумову обчислення режиму енергозбереження – модуль включений, зовнішня подія не відбувається та сигнал *save* має низький логічний рівень.

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

Рисунок 3.9 – SVA опис передумови обчислення режиму енергозбереження

На рисунку 3.10 представлено SVA опис поведінки, коли зовнішня подія триває необхідну кількість тактів - у цьому випадку 5 тактів. На наступному такті сигнал *save* має мати високий логічний рівень. Часова діаграма, яка описує таку ситуацію, наведено на рисунку 3.11. Жовтою стрілкою позначено момент початку обчислення властивості – виконання умови *onn && !evnt && !save*, зеленою – момент успішного обчислення властивості – сигнал *save* приймає значення логічної одиниці. Варто зазначити, що цей опис покриває ситуацію, коли зовнішня подія триває більше необхідної кількості тактів.

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

Рисунок 3.10 – SVA опис поведінки, коли зовнішня подія триває необхідну кількість тактів

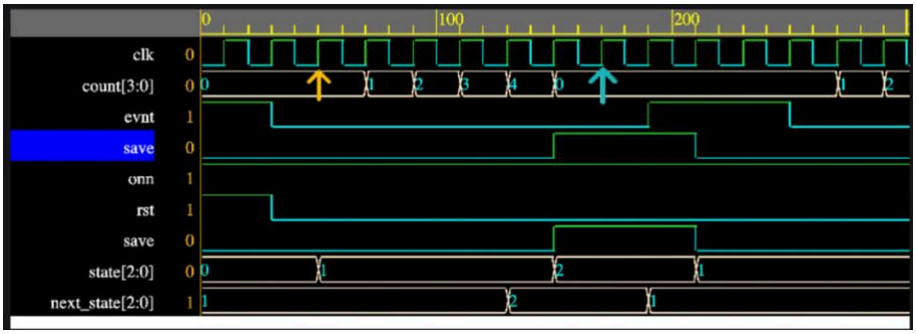


Рисунок 3.11 – Часова діаграма обробки зовнішньої події, яка триває необхідну кількість тактів

Аналогічним чином описується ситуація, коли зовнішня подія не задовольняє часовим вимогам. На рисунку 3.12 представлено SVA код опису такого сценарію.

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

Рисунок 3.12 – SVA опис поведінки, коли зовнішня подія триває менше необхідної кількості тактів

Відповідна часова діаграма представлена на рисунку 3.13. Червоною стрілкою позначено момент початку обчислення властивості – виконання умови *onn && !evnt && !save*, жовтою – момент, коли сигнал *evnt* приймає значення логічної одиниці, зеленою – момент успішного обчислення властивості – сигнал *save* приймає значення логічного нуля.

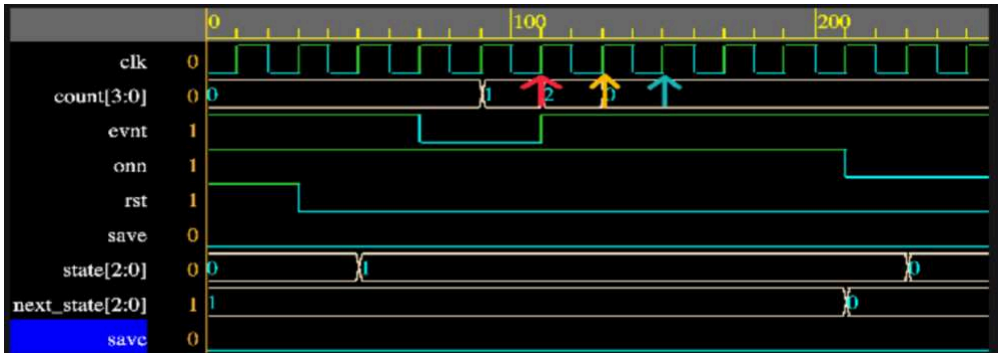


Рисунок 3.13 – Часова діаграма обробки зовнішньої події, яка триває менше необхідної кількості тактів

Останній варіант поведінки – модуль включений (сигнал *onn* має рівень логічної одиниці), та сигнал *evnt* має високий логічний рівень весь час (зовнішня подія не відбувається взагалі). SVA опис такої поведінки представлено на рисунку 3.14.

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

Рисунок 3.14 – SVA-опис поведінки, коли зовнішня подія не задовольняє часові вимоги

Відповідна часова діаграма представлена на рисунку 3.15. Червоними стрілками позначено момент початку обчислення властивості – виконання умови *onn && evnt*, жовтими – моменти успішного обчислення властивості – сигнал *save* приймає значення логічного нуля.

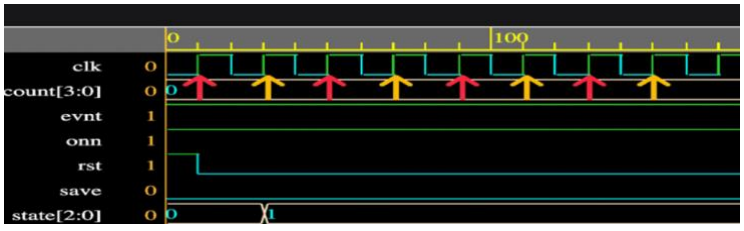


Рисунок 3.15 – Часова діаграма роботи автомата, коли зовнішня подія не відбувається

Подібно необхідно описати сценарії поведінки HDL-моделі, коли функціональність автомата залежить від довжини події та вікна прийому. У розділі 2 було показано, що для такого класу пристроїв є 4 можливі поведінки: 1) подія повністю у вікні прийому; 2) подія частково у вікні прийому; 3) подія більша за вікно прийому; 4) подія не відбувається під час вікна. Для подальшої ілюстрації асерційних конструкцій будемо використовувати модель запропоновану в пункті 2.3.4.

На рисунку 3.16 представлено SVA опис поведінки, коли подія повністю у вікні прийому. При описі конструкції property використовується імплікативна структура (оператор |=>): у лівій частині виразу описується поведінка, коли подія починається і закінчується у вікні прийому. Такий сценарій розбито на окремі послідовності (*sequence*) для простоти: 1) *wnd* – описує вікно прийому порівнюючи значення внутрішнього лічильника; 2) *begin_evnt* – початок події (позитивний фронт сигналу *btn*). Далі послідовності компонуються операторами для утворення передумови. У правій частині виразу описується результат, а саме активація сигналу *in_window* та відсутність активації інших сигналів.

```
sequence wnd;
  count >= C1 && count <= C2;
endsequence

sequence begin_evnt;
  @(posedge clk) !btn ##1 btn;
endsequence

property exact_match;
  disable iff(rst)
  @(posedge clk) wnd and begin_evnt ##[1:4] (wnd and !btn) |=> in_window and !in_wnd_part
  and !out_window and !no_evnt;
endproperty
```

Рисунок 3.16 – SVA опис поведінки, коли зовнішня подія повністю у вікні прийому

На рисунку 3.17 представлено часову діаграму моделювання такої поведінки та з явним визначенням точок обрахування асерції. Червоною стрілкою показано виконання умови початку події у вікні прийому (*wnd and*

begin_evnt), жовтою – її кінець (*wnd and !btn*), синьою – виконання результату (*in_window and !in_wnd_part and !out_window and !no_evnt*)

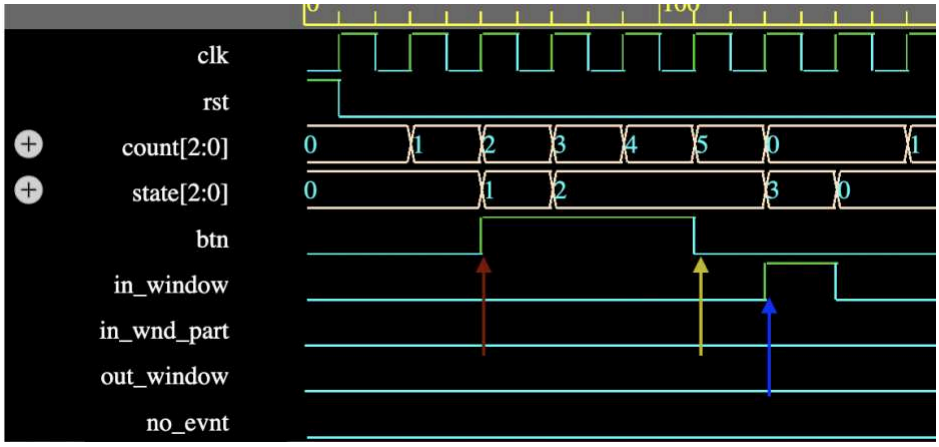


Рисунок 3.17 – Часова діаграма обчислення асерції, коли зовнішня подія повністю у вікні прийому

Аналогічно описується ситуація, коли подія частково у вікні прийому. SVA такого сценарію представлено на рисунку 3.18. Фактично дана конструкція перевіряє, що кінець події припадає на вікно прийому.

```
sequence end_evnt;
  @(posedge clk) btn ##1 !btn;
endsequence

property part_match;
  disable iff(rst)
  @(posedge clk) wnd and end_evnt |=> in_wnd_part && !in_window && out_window && !no_evnt;
endproperty
```

Рисунок 3.18 – SVA-опис поведінки, коли зовнішня подія частково у вікні прийому

На рисунку 3.19 представлено часову діаграму моделювання цієї поведінки та з явним визначенням точок обрахування асерції. Червоною стрілкою показано виконання умови початку події у вікні прийому (*wnd and*

begin_evnt), жовтою – її кінець (*wnd and !btn*), синьою – виконання результату (*in_window and !in_wnd_part and !out_window and !no_evnt*)

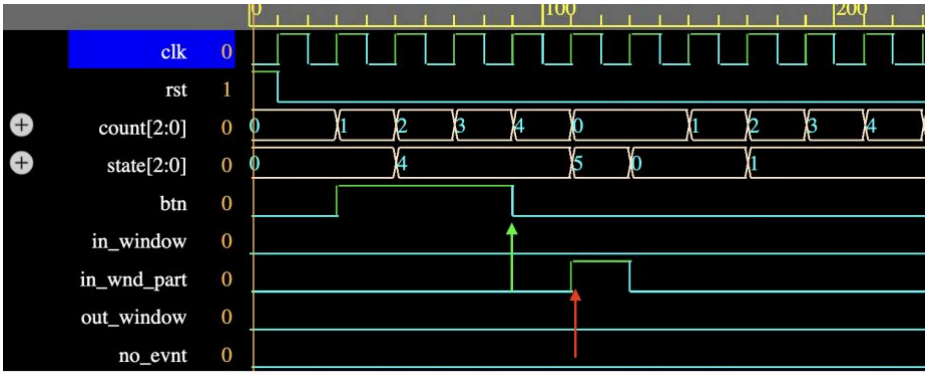


Рисунок 3.19 – Моделювання асерції, коли зовнішня подія частково у вікні прийому

Наступною поведінкою є ситуація, коли подія більша за вікно прийому. Відповідна SVA конструкція представлена на рисунку 3.20. Для опису цієї ситуації використовується оператор *first_match*, який вибирає перший із можливих кількох збігів заданої послідовності – оскільки умова «подія триває у вікні прийому» (*wnd and btn*) може виконуватися декілька разів. Це дає змогу виключити всі наступні збіги з розгляду. Другою частиною передумови є вираз *count > C2 and btn* – подія триває поза вікном.

```
property bigger_event_match;
  disable iff(rst)
  @(posedge clk) first_match(wnd and btn) ##[1:4] (count > C2 and btn) |=> out_window &&
  !in_wnd_part && !in_window && !no_evnt;
endproperty
```

Рисунок 3.20 – SVA-опис поведінки, коли зовнішня подія виходить за вікно прийому

Часову діаграму моделювання цієї поведінки та з явним визначенням точок обрахування асерції представлено на рисунку 3.21. Зеленими стрілками

показано виконання умов *wnd and btn* та *count > C2 and btn,* синьою – виконання результату (*out_window && !in_wnd_part && !in_window && !no_evnt*).

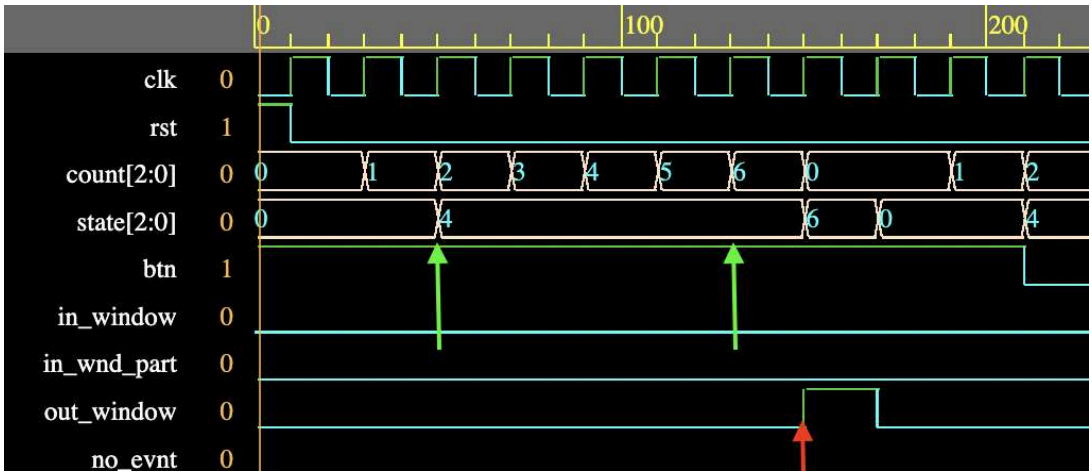


Рисунок 3.21 – Моделювання асерції, коли зовнішня подія виходить за вікно прийому

Останнім сценарієм є поведінка, коли зовнішня подія на відбувається під час вікна прийому. У цьому випадку необхідно перевірити, що сигнал *btn* (зовнішня подія) зберігає низький логічний рівень упродовж всього вікна прийому (рисунок 3.22).

```
property no_event_match;
  disable iff(rst)
  @(posedge clk) count == C1 and !btn[*5] |=> !out_window &&
  !in_wnd_part && !in_window && no_evnt;
endproperty
```

Рисунок 3.22 – SVA-опис поведінки, коли зовнішня подія не потрапляє у вікно прийому

Відповідна часова діаграма представлена на рисунку 3.23. Червоними стрілками показано вікно прийому, зеленою виконання *умови !out_window && !in_wnd_part && !in_window && no_evnt.*

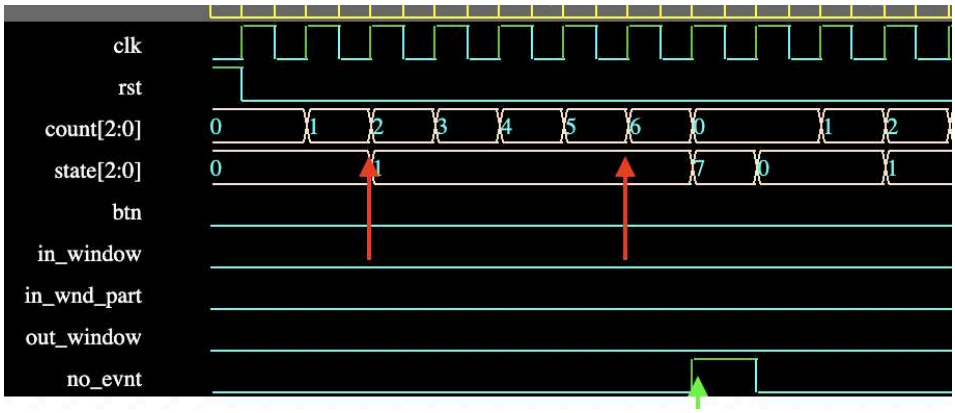


Рисунок 3.23 – Моделювання асерції, коли зовнішня подія не відбувається за вікно прийому

3.4 Висновки до розділу 3

1. Визначено місце використання асерцій та методів формальної верифікації у проєктуванні пристроїв реального часу.

2. Запропоновано методику розстановки асерційних точок у HDL-моделі часових автоматів.

3. Розроблено SVA моделі верифікації темпоральних характеристик часових автоматів залежно від характеру темпоральних переходів.

4. Розроблені моделі апробовано та перевірено на моделях часових автоматів поведінковим моделюванням у середовищі Aldec Riviera Pro 2022.

4 АВТОМАТИЗАЦІЯ ПРОЄКТУВАННЯ ТЕСТОПРИДАТНИХ ЧАСОВИХ АВТОМАТІВ

У четвертому розділі розглянуто питання методи тестопридатного проєктування часових керуючих автоматів через введення додаткових апаратних витрат на етапі проєктування HDL-моделі, а саме – розширенням вхідного алфавіту автомата. Цей спосіб дає змогу встановити автомат у довільний стан у детермінований час, суттєво скорочує довжину та час діагностичного експерименту.

Завдання, які необхідно вирішити в процесі дослідження.

1. Розробити методи підвищення тестопридатності моделей кінцевих часових автоматів через розширення вхідного алфавіту шляхом введення додаткового стовпця в таблицю переходів-виходів, що дасть змогу встановлювати керуючий автомат у довільний стан за (n-1) тактів, де n – число станів автомата;

2. Провести автоматизований синтез запропонованих діагностичних моделей та проаналізувати додаткові апаратурні витрати, які виникають при розширенні вхідного алфавіту.

3. Розробити методи проведення неруйнівного діагностичного експерименту в графових моделях керуючих автоматів реального часу за рахунок організації гамільтонових циклів у темпоральному графі переходів.

4.1 Тестопридатне проєктування часових автоматів

Цифровий пристрій вважається таким, що необіхдно перевірити, якщо задовольняються такі параметри щодо фінансових і часових витрат у визначених межах: 1) генерація тестових наборів; 2) оцінка ефективності набору тестів; 3) проведення тестової діагностики.

Типовий діагностичний експеримент (ДЕ) над автоматом складається з двох етапів: подання вхідних послідовностей та аналіз відповідних вихідних реакцій. Експерименти поділяються на три категорії: ідентифікація станів

автомата, ідентифікація вхідної послідовності автомата та ідентифікація автомата з *n* станами, які відрізняються від усіх інших автоматів із такою ж кількістю станів. Діагностичний експеримент, який обходить усі вузли графа переходів, використовується для виявлення проблем проєктування в моделях керуючих автоматів, що представлені таблицею переходів або графом переходів.

Легкотестований автомат визначається як автомат, який можна встановити в будь-який стан упродовж менш ніж *n* циклів без синхронізуючих послідовностей, де *n* – загальна кількість станів [52].

Відомо, що для класичних автоматів підвищення тестопридатності можливе лише через розширення вхідного алфавіту *X*, внутрішньої множини станів *Z* або вихідного алфавіту *Y* [51]. З одного боку, це збільшує кількість елементів, які використовуються для цільової схеми. З іншого боку, це підвищує тестопридатність цифрового пристрою. Введення апаратурної надлишковість у HDL- опис автомату через додавання фрагментів коду, які дають змогу встановити FSM у довільний стан без синхронізуючих послідовностей, є найпоширенішим способом підвищення тестопридатності. Зазвичай вхідний алфавіт розширюється новим символом для увімкнення режиму діагностики, який дає змогу обходити всі стани автомата, реалізуючи цикл Гамільтона.

З аналізу методів видно, що побудова повних діагностичних експериментів з автоматами стає простішою, коли використовується мінімальний, сильнозв’язний автомат із відмінною послідовністю мінімальної довжини, а також із синхронізуючими і встановлюючими послідовностями.

Якщо пам'ять автомата реалізована на тригерних елементах, і прийняте обмеження на клас виявлених несправностей, яке не збільшує число станів автомата, то ймовірність появи таких несправностей невелика, коли число станів точно дорівнює ступеню 2, тобто $n=2^k$, де k – число використовуваних

тригерів. Але якщо $n \neq 2^k$, то такі несправності дуже ймовірні. З огляду на це, бажано, щоб вихідний автомат мав усі $2^k = n$ станів.

Побудові легкотестованих автоматних моделей присвячена [46]. У цій роботі запропоновано й математично обґрунтовано розширення вхідного алфавіту керуючого автомата через введення додаткового стовпця у таблицю переходів-виходів, що дає змогу встановлювати автомат в довільній стан за (n-1) тактів, де n – кількість станів автомата. Зокрема, пропонується методологія розстановки додаткових дуг на графі переходів, що забезпечує мінімальні додаткові апаратурні витрати при подальшому синтезі. Запропоновану методологію було використано для побудови відповідних HDL-моделей із подальшою верифікацією та аналізом апаратних витрат.

Реалізація будь-яких методів побудови легкотестованих часових автоматів має насамперед брати до уваги темпоральні властивості такого роду моделей. Необхідно враховувати затримку в стані t_{to} та затримку обробки зовнішньої події t_c. Визначимо легкотестований автомат як такий, для якого можна побудувати ДЕ мінімальної довжини через забезпечення встановлення КА у довільний стан за (n-1), де n – число станів автомата. Опишемо даний інваріант як:

$$\exists\lambda(z_i, x_o)=z_{i+1},\ T(z_i, z_{i+1})<n\text{-}1,\ \forall i=\overline{1,n}, \tag{4.1}$$

де T – функція обчислення часу переходу між двома станами,

Очевидно, що виконання цього інваріанту потребує додаткових змін у структурі часового автомата, для якого є справедливим таке твердження: нехай в автоматі A = (X, Y, Z, δ, λ, T_c, T_{to}, T_d) зі множиною станів Z= $\{z_1\}$, i= $\overline{1, n}$, виконується принаймні одна умова:

$$t_{to}(a_i) \notin \emptyset,\ \forall i = \overline{1,n}, \tag{4.2}$$

$$t_c(a_i) \notin \emptyset,\ \forall i = \overline{1,n}, \tag{4.3}$$

Отже, необхідно удосконалити модель розширення вхідного алфавіту з урахування затримок, а саме лічильникову функцію переходів (ЛФП). Дана функція визначається через вхідний символ x_0 алфавіту X, додаток якого породжує функцію переходів:

$$\delta(z_i, x_o) = z_{i+1} mod n, \forall i=\overline{1,n}, \tag{4.4}$$

Модифікуємо, множину затримок автомата A з урахуванням, що x = {0,1}:

$$t_{to}(a_i) = \begin{cases} \emptyset, x_0=1 \\ t_{to}(a_i), x_0=0 \end{cases} \forall i=\overline{1,n}, \tag{4.5}$$

$$t_c(a_i) = \begin{cases} \emptyset, x_0=1 \\ t_c(a_i), x_0=0 \end{cases} \forall i=\overline{1,n}, \tag{4.6}$$

Отже, (4.4) та (4.4) дозволяють перетворити часовий автомат у класичний автомат Мура і через додавання додаткового стовпця у ТПВ перетворювати його в сильнозв'язний автомат, для якого можлива побудова ДЕ мінімальної послідовності. Така модифікація забезпечує виконання інваріанту (4.1) в незалежності від типу темпоральних переходів, що присутні в автоматній моделі.

4.2 Побудова легкотестованих часових автоматів шляхом модифікації темпорального графу переходів

Відомо, що вхідною точкою проєктування цифрового пристрою є специфікація, що задається як алгоритм функціонування та реалізується мовами опису апаратури. Для такого способу подання ЦП структурні методи діагностування, спрямовані на виявлення константних несправностей, стають менш ефективними. З іншого боку, застосування функціональних методів стає досить складним через великі розмірності сучасних пристроїв.

Одним із поширених способів опису моделей реального часу є часовий автомат, способом його представлення є темпоральний граф, який також є його повною математичною моделлю. На високорівневому рівні темпоральний граф переходів використовується для побудови опису пристрою через автоматний шаблону на мовах опису апаратури (VHDL, Verilog, SystemVerilog). Автоматним шаблон називають спеціальне структурування HDL-моделі, у якій функції переходів і виходів виділені в окремі процеси (процес), а призначення нового стану здійснюється в спеціальному процесі, пов'язаному із синхросигналом та сигналом скидання (reset). Під час проєктування в опис керуючих автоматів вводять апаратурну надлишковість, що забезпечує легкотестованість. Таку надлишковість доцільно вносити ще на початковому етапі проєктування, тобто при побудові HDL-моделей пристроїв, які проєктуються. З огляду на вищезазначене, легкотестованим будемо називати кінцевий автомат, для якого можна побудувати діагностичний експеримент мінімальної довжини через забезпечення встановлення автомата в будь-який стан за мінімальне число тактів.

У [46] пропонується методологія розстановки додаткових дуг на графі переходів, що забезпечує мінімальні додаткові апаратурні витрати при подальшому синтезі. Запропоновану методологію було використано для побудови відповідних HDL моделей із подальшою верифікацією та аналізом апаратних витрат.

З вищезазначеного випливають задачі розроблення процедур процесу автоматизації проєктування легкотестованих кінцевих часових автоматів, представлених мовами опису апаратури, що включає в себе порівняння додаткових апаратурних витрат при використанні структурних і функціональних методів для підвищення тестопридатності через порівняння результатів синтезу тестопридатних HDL-моделей у САПР ПЛІС.

Схема вважається тестопридатною, якщо процес генерації тестових наборів, їхня ефективність та реалізація тестового діагностування можуть

бути здійснені в межах встановлених фінансових та часових обмежень, з урахуванням показників, що характеризують здатність схеми до виявлення несправностей, пошуку місця несправностей та реалізації тестового діагностування.

Експериментом над автоматом назвемо процес введення вхідних послідовностей, спостереження відповідних вихідних послідовностей та роботу з отриманими даними. Залежно від мети експерименту розрізняються експерименти з ідентифікації станів автомата, знаходження вхідних послідовностей автомата та виявлення автомата з n станами, який відрізняється від усіх інших автоматів з такою ж кількістю станів. Для виявлення помилок у проєктуванні моделей керуючих автоматів, які подаються у формі графа переходів або графа станів, застосовується діагностичний експеримент, пов'язаний із проходженням усіх вершин графа. Із цього погляду автомат вважається легкотестованим, якщо він може переходити в будь-який стан не більше, ніж за n тактів, де n - кількість станів автомата без використання синхронізуючих послідовностей.

Визначимо вимоги до HDL-моделей часових автоматів:

– модель має містити спеціальний механізм переводу автомата у режим діагностики і, навпаки, на будь-якому такті роботи автомата;

– в режимі тестування автомат може бути встановлений в будь-який стан за (n-1) тактів, де n - число станів автомата, і може бути організований Гамільтонів цикл для будь-якого зі станів автомата;

– легкотестований автомат має будуватися в автоматизованому режимі засобами САПР;

4.2.1 Легкотестована структура з розширенням вхідного алфавіту

При функціональному підході до проєктування (аналізу) цифрових пристроїв, заданих моделлю кінцевого автомата підвищення тестопридатності автомата можливо тільки розширенням вхідного алфавіту *X*, алфавіту станів *A* або вихідного алфавіту *Y* [68]. З одного боку, це

призводить до внесення в схему пристрою апаратурної надлишковостіі, з іншого – забезпечує збереження алгоритму функціонування автомата, заданого, зазвичай, ТПВ або його графом переходів.

Надалі будемо розглядати модифікацію темпорального графу переходів додатковими дугами як основний спосіб модифікації автомата для перетворення його в легкотестований.

На підставі теоретичних засад, викладених у 4.1 розглянемо автомат, який реалізує модуль управління світлофором на пішохідному переході, запропоновану в [32].

Визначимо множину станів:

1) a_1 – включення автомата, вихідні сигнали відсутні, затримка до одиниці;
2) a_2 – жовтий у випадку зміни (G–R) , виходи {YGR, R1, R2}, затримка to_2;
3) a_3 - червоний на дорозі, зелений на переході, виходи {R1, G2}, затримка to_3;
4) a_4 - жовтий у випадку зміни (R–G) , виходи {YRG, R1, R2}, затримка to_2;
5) a_5 - зелений на дорозі, червоний на переході, виходи {G1, R2}, затримка to_3;
6) a_6 - зелений на дорозі, червоний на переході, виходи {G1, R2}, затримка to_3;
7) a_7 - горить тільки жовтий, затримка to_1.

На рисунку 4.1 представлено темпоральний граф переходів автомата Мура пристрою.

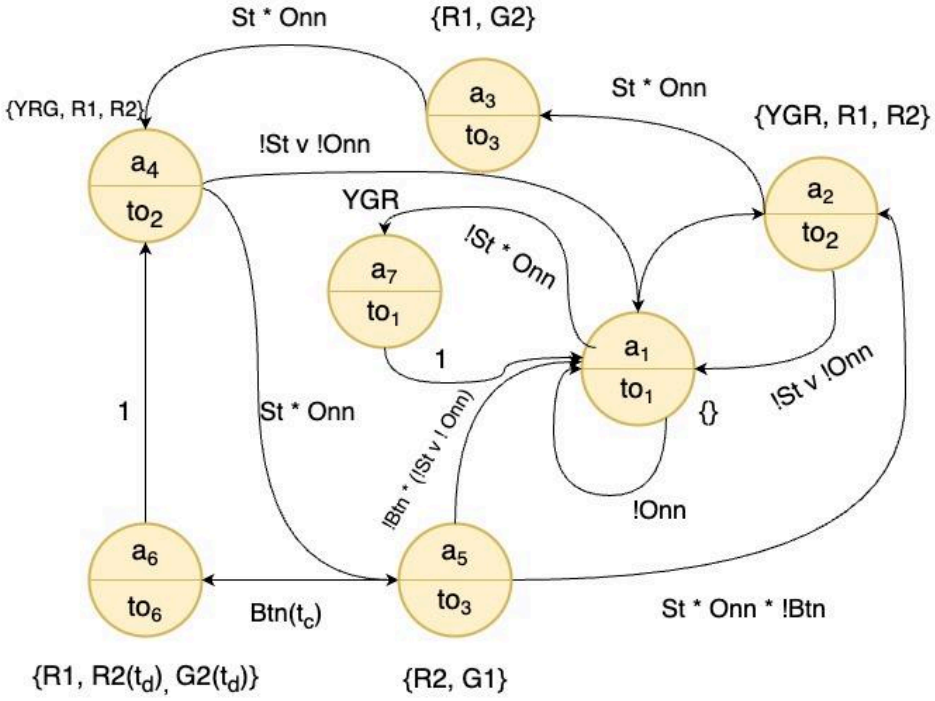


Рисунок 4.1 – Темпоральний граф переходів часового автомата Мура для управління світлофором

Алгоритм функціонування світлофора описується так: коли пристрій керування ввімкнено (Onn=1), спочатку запускається нічний цикл роботи світлофора, під час якого жовте світло на головній дорозі мигає (a_1 – a_7), а світлофор на пішохідному переході не активний. Після початку денного циклу (St=1) система переходів запускається (a_2 – a_3 – a_4 – a_5 – a_2). У стані a5, коли на головній дорозі горить зелене світло, встановлюється вікно прийому t_c для зовнішньої події Btn (натискання кнопки переходу). При обробці цієї події керуючий автомат переходить у стан a_6. На цей момент на головній дорозі вмикається червоний світлодіод, на пішохідному переході червоний світлодіод затримується, а зелений світлодіод вмикається зі затримкою t_d (час на підготовку до переходу). Упродовж періоду t_c може бути прийнятий лише один сигнал (зовнішня подія) Btn.

Для розширення вхідного набору сигналу додамо Bps (Bypass). Коли цей сигнал дорівнює 1, автомат працює в режимі діагностики, коли всі стани

автомата послідовно перевіряються. При Bps = 0 автомат виконує заданий алгоритм. Оновлений граф переходів показаний на рисункуунку 4.2.

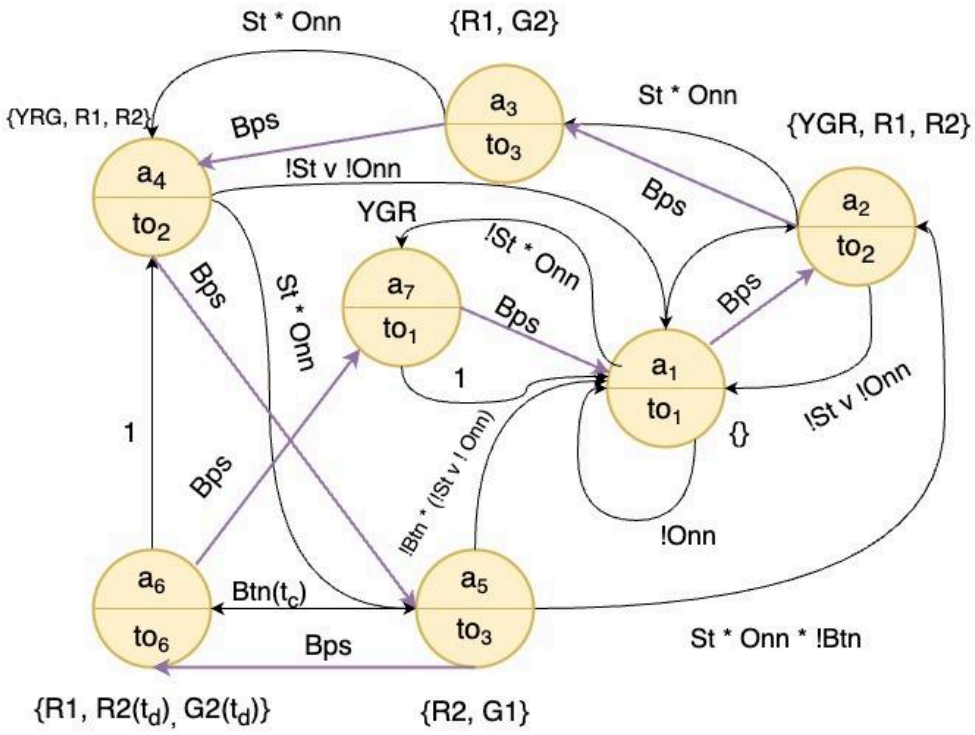


Рисунок 4.2 – Темпоральний граф переходів часового автомата Мура для управління світлофором з діагностичним входом Bps

Необхідно зробити відповідні зміни в автоматному шаблоні, а саме в процесі, що відповідає за визначення наступного стану. Перевірка сигналу *bps* має найвищий пріоритет у операторі if-else. Відповідний фрагмент коду Verilog представлено на рисунку 4.3. Тут сигнал *bps* перевіряється перед іншими сигналами та внутрішнім лічильником. Важливо зазначити, що сигнал count1 повинен бути призначений в операторі *if*, щоб уникнути синтезу латч-тригера для цього сигналу.

```
a1: begin
  if (bps)
    nextState = a2;
    count1 - 3'b000;
  else
    if(count < T1 - 1) begin
      nextState = a1;
      count1 = count + 1'b1;
    end
  else if(st && onn) begin
    nextState = a2;
    count1 = 3'b000;
  end
  else if (!st && onn) begin
    nextState = a7;
    count1 = 3'b000;
  end
  else
    begin
      nextState = a1;
      count1 = 3'b000;
    end
end
```

Рисунок 4.3 - Пріоритизація сигналу bps в описі процесу переходів

На рисунку 4.4 показано результати моделювання модифікованого автомата. Якщо сигнал *bps* дорівнює 0, то автомат працює в стандартному режимі (від 0 до 330 пс). Якщо *bps* дорівнює 1 – усі стани обходяться циклічно (цикл Гамільтона).

Результати синтезу пристрою на ПЛІС FPGA XC3S500E-5fg320 та CPLD XC9572XL-10-TQ100 показують, що автомат має 7 станів. Після модифікації HDL опису та введення додаткового діагностичного входу bps результати не показали збільшення кількості використаних тригерів (flips-flops), що кодують стани, і відсутність використання латч-тригерів (latches). Проте введені модифікації призвели до збільшення використання основних елементів логіки (BELs) для обох ПЛІС. Для FPGA використання субблоків (Slices) і таблиць перетворення (LUTs) зросло з 16/30 до 18/35 відповідно. Для CPLD кількість використаних макрокомірок (macrocells) зросла з 10 до 12. Середні додаткові апаратні витрати становлять 20% (рисунок 4.4). Основні апаратні витрати стосуються комбінаційної логіки, яка відповідає за встановлення окремих бітів регістру стану (додаткові конструкції *if*).

Таблиця 4.1 –Результати початкового синтезу автомата для ПЛІС FGPA та CPLD

ПЛІС	Тригери	BELs	Slices/LUTs	Макрокомірки
FPGA	10	30	16/30	
CPLD	6	120		10

Таблиця 4.2 – Результати синтезу автомата для ПЛІС FGPA та CPLD з введенням додаткового діагностичного входу bps

ПЛІС	Тригери	BELs	Slices/LUTs	Макрокомірки
FPGA	10	36	18/35	
CPLD	6	143		12

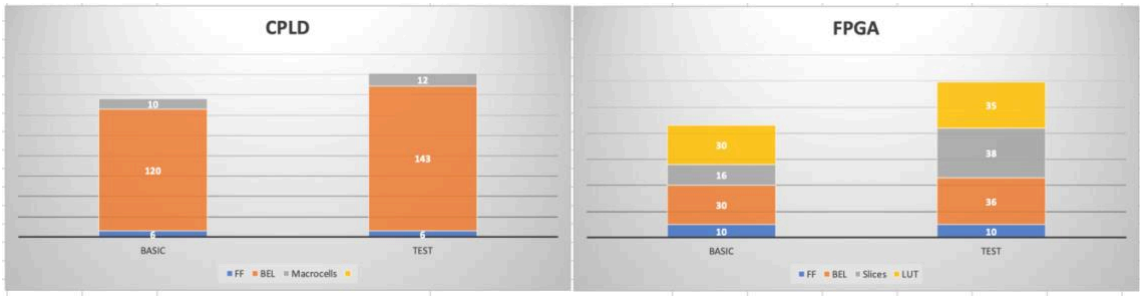


Рисунок 4.4 – Порівняння апаратних витрат початкової моделі та моделі з додатковим входом

Щодо аналізу швидкодії, то для CPLD не виявлено змін у мінімальному тактовому періоді та максимальній частоті. Для FPGA мінімальний тактовий період збільшився з 3.900 нс до 4.112 нс, а максимальна частота зменшилася з 256.430 МГц до 243.188 МГц. Отже, зниження швидкодії становить менше 8% (рисунок 4.5).

Таблиця 4.3 - Порівняльні результати швидкодії автомата для FGPA та CPLD з додатковим діагностичним входом

	FPGA	FPGA з додатковим входом	CPLD	CPLD з додатковим входом
Мінімальний період тактового сигнала, нс	3.900	4.112	6.300	6.300
Максимиальна частота, МГц	256.430	243.188	158.730	158.730

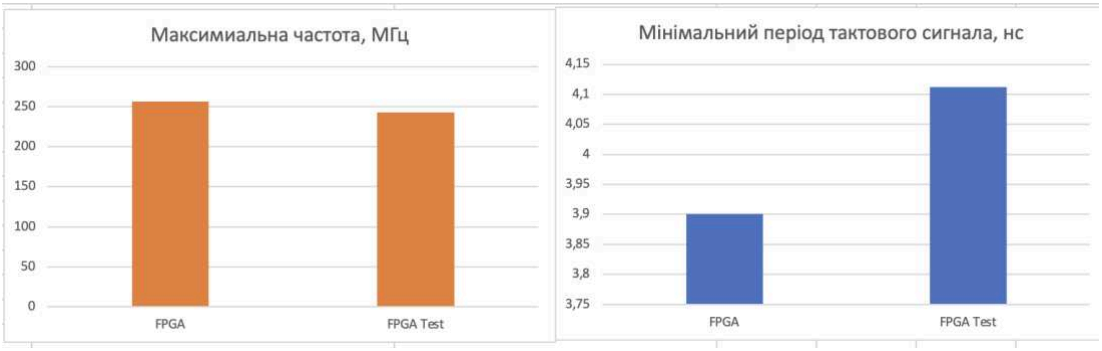


Рисунок 4.5 – Порівняння швидкодії початкової моделі та моделі з додатковим входом для FPGA

4.3 Аналіз апаратурних витрат при побудові легкотестованих автоматів

4.3.1 Аналіз тестопридатності функцій переходів

У підрозділі 4.2 було запропоновано та обґрунтовано введення апаратурної надлишковості на етапі проєктування керуючого автомата введенням додаткових дуг на темпоральному графі переходу та розширенням вхідного алфавіту додатковим сигналом Bps з відповідною модифікацією HDL-опису автомата. Такі перетворення дають змогу встановити автомат у довільний стан за детерміновану кількість автоматних тактів. При синтезі додаткові конструкції if-else Verilog/VHDL опису реалізуються

мультиплексори. Зазначено, що додаткові апаратурні витрати не перевищують 25-30%.

Окремим аспектом запропонованої методології є мінімізація апаратних витрат при модифікації темпорального графу та оптимізація процесу обходу всіх станів. Нижче буде розглянуто різні варіанти додавання додаткової дуги залежно від типу переходів.

Перший варіант полягає в додаванні додаткового переходу зі стану із затримкою ($t_{to}(a_i) \neq 0$). На рисунку 4.4 (зліва) позначено початковий темпоральний граф автомат – перехід зі стану a_0 в a_1 відбувається через t_0 циклів синхросигнала. Для більш докладного пояснення в умовах переходів на дугах графу використано інваріант часу – T (значення лічильника більше за $t_{to}(a_0)$ - 1).

Застосовуючи формулу (4.5) отримаємо граф, представлений на рисунку 4.6 (праворуч).

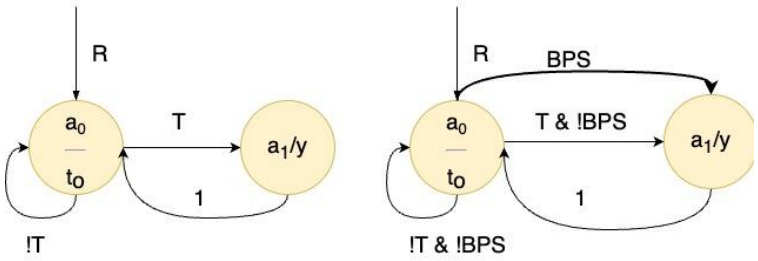


Рисунок 4.6 – Автомат з часовим переходом

Таким чином отримуємо автомат з безумовними переходом з функцією переходу виду: $a_1=a_0Bps \vee a_0T\overline{Bps}=a_0(Bps \vee T\overline{Bps}) = a_0Bps \vee a_0T$. Застосувавши закон Блейка-Порецького отримали $a_0Bps \vee a_0T$. Це означає що додаткові апаратні витрати є витрати на терм a_0Bps - це вхід сигнала bps в тригери лічильника та LUT, що визначає стан автомат, а також вхідний буфер, який генерується для усіх входів. Це твердження підтвердив автоматизований синтез у САПР Vivado з використання чипу сімейства Zynq 7000.

Синтезовану схему наведено на рисунку 4.7. Модифікований процес переходів на мові Verilog наведено на рисунку 4.8.

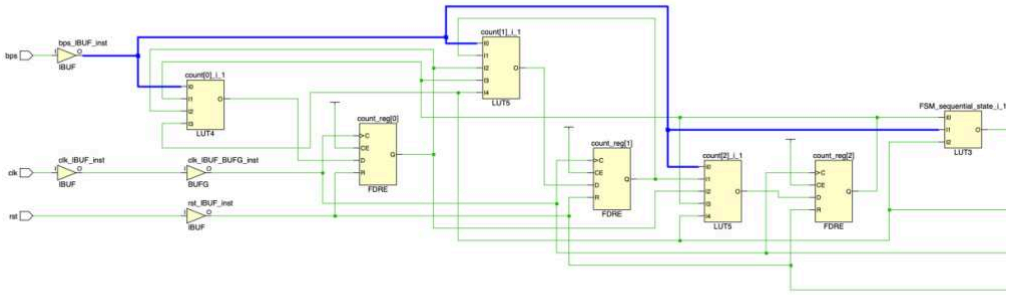


Рисунок 4.7 – Результати синтезу Verilog-моделі автомата з входом bps для обходу стану з часовим переходом

```
always @(*) begin
  next_count = 4'd0;
  case (state)
    a0:
      if (bps)
        next_state = a1;
      else if (count < C2 - 1) begin
        next_state = a0;
        next_count = count + 1'b1;
      end
      else
        next_state = a1;
    a1: next_state = a0;
    default: begin
      next_state = a0;
    end
  endcase
end
```

Рисунок 4.8 – Процес переходів автомата з часовим переходом

Другий варіант полягає в обході стану з умовно-часовим переходом. Такого роду автомат переходить у стан a_1 під час першого фронту синхросигнала після перебування у стані a_0 впродовж як мінімум $t_{to}(a_0)$ циклів синхросигнала. Приклад такого автомата показано на рисункуунку 4.9. У правій частині рисунка представлено модель після застосування формули (4.5).

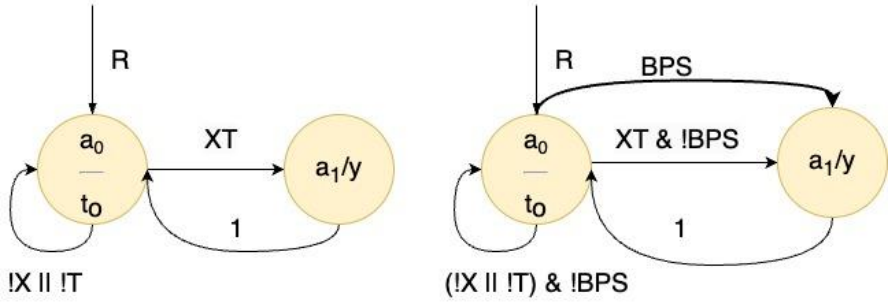


Рисунок 4.9 – Приклад автомата з умовно-часовим переходом

Функція переходів у цьому разі матиме вигляд із застосуванням закону Блейка-Порецького: $a_1 = a_0 Bps \vee a_0 XT\overline{Bps} = a_0(Bps \vee XT\overline{Bps}) = a_0 Bps \vee a_0 XT$. Це означає, що додаткові апаратурні витрати при реалізації мінімізованого виразу для функції переходу в стан a_j – вентиль, який реалізує терм $a_0 Bps$. Оскільки сигнал bps використовується для обходу таймера, то як і в попередньому випадку додаткові витрати – це вхід сигнала bps в тригери лічильника та LUT, що визначає стан автомат, а також вхідний буфер, який генерується для усіх входів. Фрагмент синтезованої схеми наведено на рисунку 4.10, модифікований процес переходів – на рисунку 4.11.

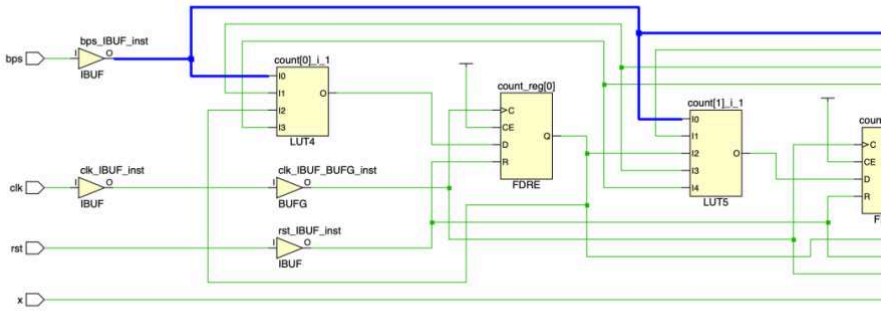


Рисунок 4.10 – Фрагмент синтезу Verilog-моделі автомата з входом bps для обходу стану з умовно-часовим переходом

```
always @(*) begin
  next_count = 4'd0;
  case (state)
    a0:
      if (bps)
        next_state = a0;
      else if (count < C2 - 1) begin
          next_state = a0;
          next_count = count + 1'b1;
      end else if (x)
         next_state = a1;
      else
        next_state = a0;
    a1: next_state = a0;
    default: begin
      next_state = a0;
    end
  endcase
end
```

Рисунок 4.11 – Процес переходів автомата з умовно-часовим переходом

Іншим можливим варіантом такого типу автоматів є випадок, коли умовно часовий перехід має більше ніж одну вихідну дугу у стани відмінні від a_j як на рисунку 4.12.

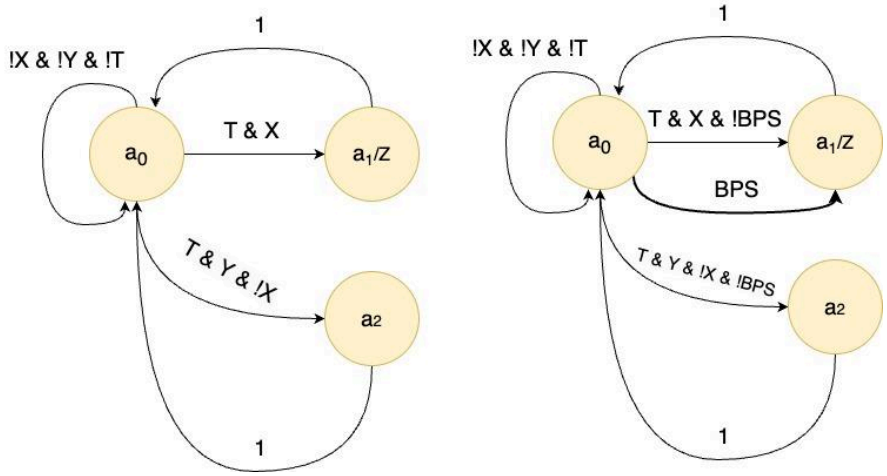


Рисунок 4.12 – Приклад автомата з умовно-часовим переходом у декілька станів

Функція переходів у такому випадку матиме вигляд із застосуванням закону Блейка-Порецького: $a_1 = a_0 Bps \vee a_0 XT\overline{Bps} = a_0(Bps \vee XT\overline{Bps}) = a_0 Bps \vee a_0 XT$. Цей випадок є ідентичним попередньому, оскільки перехід у стан a_1 чи a_2 можливий лише при наявності вхідного сигналу при виконанні інваріанту часу, оскільки саме такою є семантика часових автоматів.

Підсумуємо вищенаведені викладки. Перетворення часової моделі автомата в легкотестований несе за собою введення додаткових дуг зі станів з інваріантами часу та подій. Додаткові дуги необхідні для виконання умови (4.1) і дають змогу оминати значення часу та очікування зовнішніх подій при організації діагностичного експерименту. Такі маніпуляції дозволяють перетворити темпоральний автомат в класичний автомат Мура або Мілі для оптимальної побудови цикла Гамільтона.

4.3.2 Розрахунок тестопридатності темпоральних моделей автоматів

Як було показано в підрозділі 4.3 (п. 4.3.1) додаткові апаратурні витрати, які забезпечують тестопридатність схемної реалізації часового автомата введенням додаткових дуг на темпоральному графі, які дають змогу встановити автомат у довільний стан, залежить насамперед від типу автоматного переходу та способу кодування станів автомата. Аналіз показав, що є прямо-пропорційна залежність між типом темпорального переходу та надлишковими апаратурними витратами на реалізацію додаткової дуги *bps* у темпоральному графі переходів КА. З цього випливає, що є спосіб оптимізації (зменшення) апаратурних витрати на забезпечення тестопридатності через визначення оптимального порядку встановлення додаткових дуг *bps* в графі переходів керуючого автомата.

Для оцінювання апаратурних витрат на реалізацію функцій збудження, що забезпечують перехід керуючого автомата в довільний стан a_i, використовуватимемо показник досяжності. У технічних системах управління під досяжністю розуміється можливість (з урахуванням часу та виконаних дій) встановлення системи в певний технічний стан.

Щодо керуючих автоматів у системах автоматного управління, під досяжністю ми розуміємо ваговий коефіцієнт, який призначений кожній вершині графа переходів керуючого автомата. Цей коефіцієнт відображає в умовних одиницях складність переходу автомата у вказаний стан, починаючи з початкового стану.

Потрібно враховувати, що складність функцій виходів має прямий вплив на спостережуваність стану автомата й непрямий вплив на складність проведення діагностичного експерименту. Однак для задачі встановлення автомата в довільний стан можна не враховувати складність функцій виходів. Додавання додаткової дуги *bps* впливає на складність апаратурної реалізації функцій збудження, а функції виходів залишаються незмінними.

Розрахунок досяжності керуючого автомата будемо робити через аналіз темпорального графа переходів, де стани автомата умовно кодуються.

Припустимо, що в графі переходів автомата є вершина a_n із затримкою $t_o(a_n)$ або вікном прийому $t_c(a_n)$, яка має k попередників та, відповідно, j вихідних дуг (4.13).

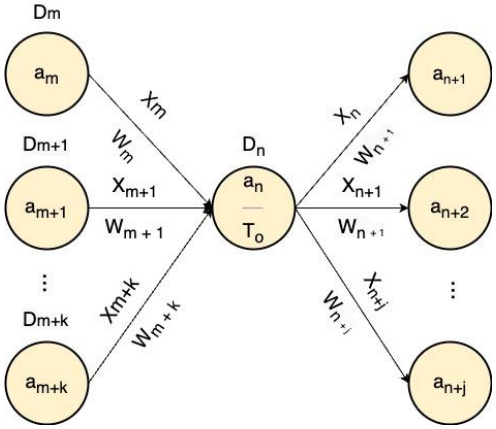


Рисунок 4.13 –Коефіцієнти досяжності в темпоральному графі переходів

Для кожної k-ї дуги темпорального графа $(a_{m+k} \to a_n)$ обчислюється її коефіцієнт досяжності W_{m+k}, який враховує сумарну вагу умов переходу,

співвідношення двійкових кодів станів автомата a_{m+k} та a_n та затримки в стані.

$$W_{k+m}=q_{k+m}\sum_{i=1}^{R}p_r+t_r, \tag{4.7}$$

де R - кількість умов переходу на k-й дузі, p_r - вага відповідної умови переходу, t_r – коефіцієнт інваріанту часу, який має задовольнити під час переходу, q_{m+k} - кодова відстань між двійковими кодами станів автомата a_{m+k} та a_n. Варто зауважити, що при унітарному кодуванні $q_{k+m}=1$, $\forall i=\overline{1,R}$. Коефіцієнт інваріанту часу обраховується як кількість умов переходу, що залежать від таймера та його значення. Пропонується такий спосіб визначення:

$$t_r=\sum_{i=1}^{n}0.5*C_n, \tag{4.8}$$

де n - кількість значень таймера на дузі, C_n - бажане значення таймера для здійснення переходу у тактах синхросигнала.

Зазначимо, що для КА в системах логічного управління p_r залежить від складності пристрою, що перетворює аналогові сигнали датчиків у логічні значення $x_r=\{0, 1\}$ залежно від меж допустимих значень вхідних сигналів.

Досяжність вершини графа a_n є мінімальною сумою досяжностей вершин-попередників та коефіцієнтів досяжності W_{m+k} для дуг $(a_{m+k}\to a_n)$:

$$D_n=\min_{k=1}^{K}(D_{k+m}+W_{k+m}), \tag{4.9}$$

Зауважимо, що для початкової вершини a_0 $D_0 = 1$, петлі при розрахунку

досяжності не враховуються.

Пропонується такий метод обчислення досяжностей у темпоральному графі КА.

1. Помічаємо початкову вершину графу a_0 і присвоюємо початкову досяжність $D_0=1$,.

2. Обчислюємо коефіцієнти досяжності всіх вихідних дуг W_j для кожної поміченої вершини, де $j=\overline{1,J}$ - кількість вихідних дуг, за формулою (4.8) та ці дуги помічаються.

3. Обчислюємо коефіцієнт досяжності кожної поміченої вершини D_n за формулою (4.8).

4. Пункти 2 та 3 повторюються доти, поки всі вершини графа не будуть помічені.

При виборі пари станів автомата, між якими встановлюється додатковий перехід (дуга *bps* у темпоральному графі), обирається той наступний стан, для якого загальне оцінювання апаратурних витрат для функцій збудження є мінімальною з урахуванням кодування станів автомата. При цьому відсутність переходів між станами рівносильна переходу з однією умовою.

При розміщенні додаткової дуги *Bps* у переході $a_n \rightarrow a_j$ при наявності двох та більше вихідних дуг зі стану a_n застосовується функція переваги:

$$F = \min_{j=1}^{J} |D_n - D_j|, , \tag{4.10}$$

де D_n - досяжність вершини, з якої виходить додаткова дуга *bps*, де D_j - досяжність вершини-наступника, j = 1, n - кількість вихідних дуг із вершини. Отже, додаткову дугу необхідно ставити у стан із мінімальною досяжністю.

Вищезазначена методика розстановки додаткових дуг є евристичною з двох причин: 1) обчислення досяжності залежить від способу кодування, яке автоматизується САПР при проєктуванні, і, відповідно, проєктувальник не завжди може впливати на цей процес; 2) вибір лише однієї дуги на кожному

кроці алгоритму, яка визначає досяжність наступної вершини, фактично розглядає тільки один із можливих шляхів у графі, за яким автомат може бути встановлений у зазначений стан і не враховує можливість наявності циклів у графі переходів автомата, що в кінцевому підсумку може вплинути на обчислення досяжностей.

Одним із результатів використання коефіцієнтів досяжності є обчислення тестопридатності з забезпечення діагностичного експерименту з обходу графа переходів за визначеним *p*-м шляхом. Показник складності р-го шляху S_p обчислюється за формулою:

$$S_p = \sum_{i=1}^{H} W_{m,n}, \tag{4.11}$$

де *H* - кількість дуг для *p*-го шляху. Перевага при обранні шляху обходу графа переходів віддається тому, у якого S_p мінімальна.

4.3 Методика розстановки дуг

У попередніх викладках було обумовлено, що неруйнівний ДЕ можливо побудувати обходом всіх вершин із поверненням у початковий стан, тобто через гамільтонів цикл.

Нагадаємо, що за визначенням граф є гамільтоновим, якщо є замкнутий маршрут, який включає кожну вершину виключно один раз. Такий цикл не обов’язково має містити всі дуги. Очевидно, що довільний орієнтований граф не є гамільтоновим. Водночас гамільтонів граф може містити один або декілька гамільтонових циклів. Визначимо, що для реалізації неруйнівного ДЕ граф КА має бути перетворений у гамільтонів через введення додаткових дуг Bps.

Відомо, що достатніми умовами існування гамільтонова циклу є умови Оре та Дірака.

Для графа з n вершинами, якщо найменший ступінь вершини не менше $\frac{n}{2}$, то такий граф є графом Дірака і він є гамільтоновим. Крім того, для графа з більш ніж двома вершинами, якщо сума ступенів усіх несуміжних вершин не менше $\frac{p}{2}$, то цей граф також є гамільтоновим. Ця умова відома як умова Оре.

Теорема Бонді-Хватала надає узагальнення умов Дірака та Оре. Згідно із цією теоремою, граф є гамільтоновим лише тоді, коли його замикання також є гамільтоновим графом. Для графа G = <U, V> з n вершинами замикання будується через додавання ребер (u,v) для кожної пари несуміжних вершин u і v, сума ступенів яких не менше n.

Проаналізуємо темпоральний граф, що описує роботу світлофора, наведений на рисунку. 4.2. Аналіз показує, що для цього графу не виконується умова Дірака, оскільки ступінь вершин a_6 та a_7 менше 3 ($\lfloor \frac{7}{2} \rfloor$). Крім того, не виконується умова Оре для вершин a_2/a_7, a_2/a_6, a_2/a_6, a_6/a_7. Візуальний аналіз показує, що дуга між станами a_6 a_7 дозволить виконати вимогу Оре. Оновлений граф представлено на рисунку. 4.14.

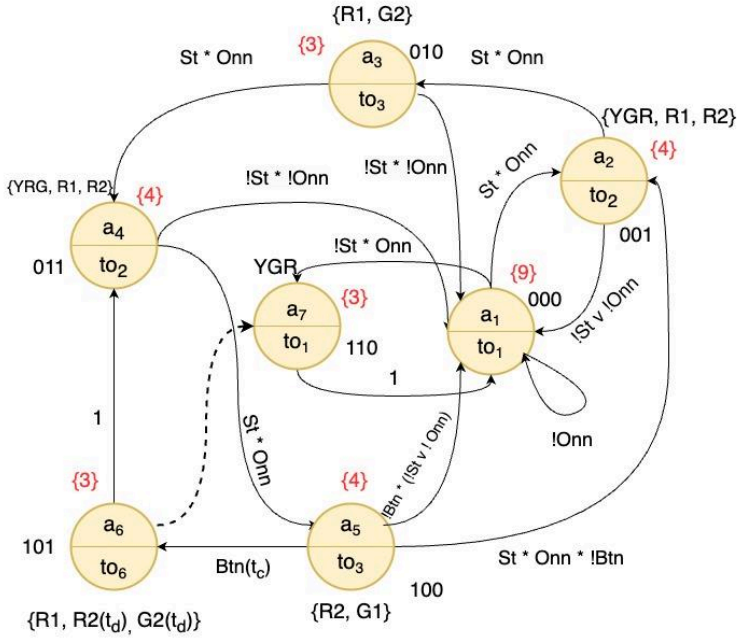


Рисунок 4.14 – Оновлений граф переходів з виконанням умов Оре та Дірака

Введення додаткових дуг Bps між вершинами, які не мають прямих переходів у початковому графі переходів, спрямоване на мінімізацію кількості гамільтонових циклів у графі. Це здійснюється застосуванням умов теореми Бонді-Хватала. З погляду мінімізації апаратурних витрат перевага віддається такому шляху обходу графа КА, для якого функція переваги (вага р-го шляху) буде $S_p = \sum_{i=1}^{H} W_{m,n}$. Водночас враховувати, що вага додаткових дуг у графі для неіснуючих переходів дорівнює a_iBps при переході a_i/a_j.

Для графа переходів КА на рисунку.4.15 обчислюємо коефіцієнти досяжності W_{m+k} та сформуємо матрицю суміжності з розрахунком досяжностей вершин у останньому рядку матриці, яка наведена на рисунку. 4.13. Нагадаймо, що $to_1 = 4$, $to_2 = 6$, $to_6 = 8$, $to_3 = [2, 4]$;

	a_1	a_2	a_3	a_4	a_5	a_6	a_7
a_1		5					7
a_2			8				
a_3	5			5			
a_4					10		
a_5	7	10				5	
a_6				8			5
a_7	5					5	
D	1	6	18	23	33	38	

Рисунок 4.15 – Матриця суміжності для темпорального графа з розрахунком досяжностей

Сума коефіцієнтів досяжності у циклі a_1–a_2–a_3–a_4–a_5–a_6–a_7–a_1 буде S_1 = 5+8+8+5+10+5+5+5=51, а для шляху a_1–a_7–a_6–a_4–a_5–a_2–a_3–a_1 буде S_2 = 7+5+8+10+10+8+5 =. Тобто перший шлях має меншу вагу та можна припустити, що додаткові дуги Bps за цим через обхід графа дадуть менші надлишкові апаратурні витрати.

Варто зазначити, що задача визначення оптимального циклу є аналогічною задачі комівояжера, а отже – NP-повною задачею. Є маса алгоритмів вирішення цієї задачі: повний перебір, динамічне програмування, метод гілок та меж тощо. У розділі 4.5 наведено інформацію про програмний модуль автоматизації проеткування часових автоматів, який включає в себе реалізацію пошуку оптимального цикла Гамільтона на мові JavaScript з урахування мінімізації апаратних витрат.

4.4 Програмний модуль побудови тестопридатних та асерційних HDL-моделей

Для автоматизації запропонованих методів побудови легкотестованих та модель, з використанням асерцій було побудовано програмний модуль на базі фреймворку Molybden. Molybden використовує браузер із відкритим кодом Chromium для відображення графічного інтерфейса застосунків, який будується з допомогою web-технологій (JavaScript, HTML, CSS).

Робота з модулем починається з вибору цільової мови опису апаратури, на якій буде згенеровано кінцеву модель. Наступним кроком необхідно задати переходи. Програма підтримує декілька типів переходів: прості, часові та умовно-часові. На рисунку 4.14 показано приклад вводу вхідних даних.

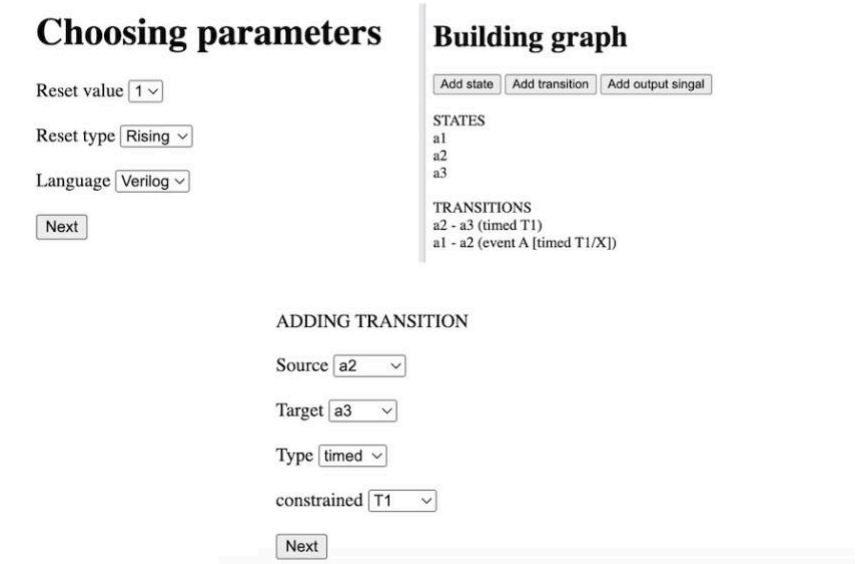


Рисунок 4.16 – Приклад вводу вхідних даних моделі

Наступним кроком модуль генерує темпоральний граф переходів та відповідний Verilog опис. Приклад цієї функціональності наведено на рисунку 4.15.

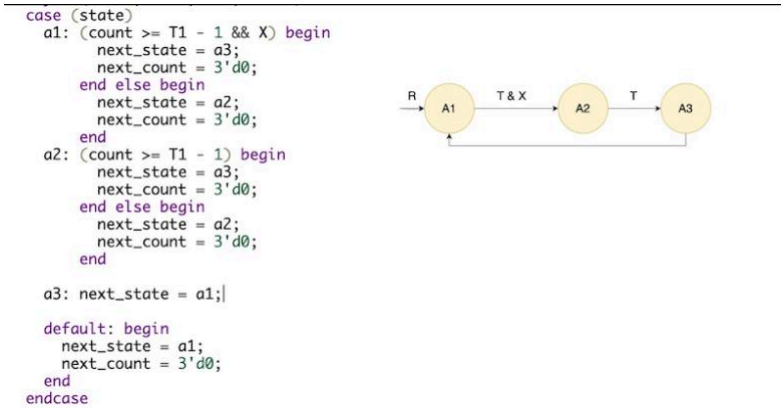


Рисунок 4.17 – Приклад згенерованого темпорального графу та Verilog опису

Також модуль дає змогу генерувати асерційні точки для темпоральних переходів на основі введених даних. Асерційні точки генеруються на основі шаблонів, представлених у розділі 3. Наразі асерційні конструкції генеруються тільки для Verilog моделей з використанням мови SVA (рисунок 4.15).

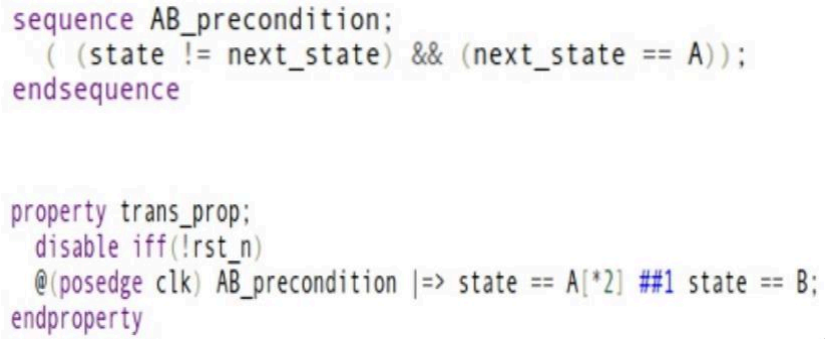


Рисунок 4.18 – Результат роботи програми з генерації асерцій

Програма має функціональність знаходження оптимального циклу Гамільтона. Для цього модуль використовує дві матриці – матрицю суміжностей та матрицю досяжностей, яка містить вагу переходів. Алгоритм перевіряє виконання умов Оре та Дірака та, у разі, якщо ці умови виконуються, знаходить оптимальний цикл Гамільтона методом повного перебору. Наприклад, для графу з рисунку 4.12 вивід буде як на рисунку 4.16.

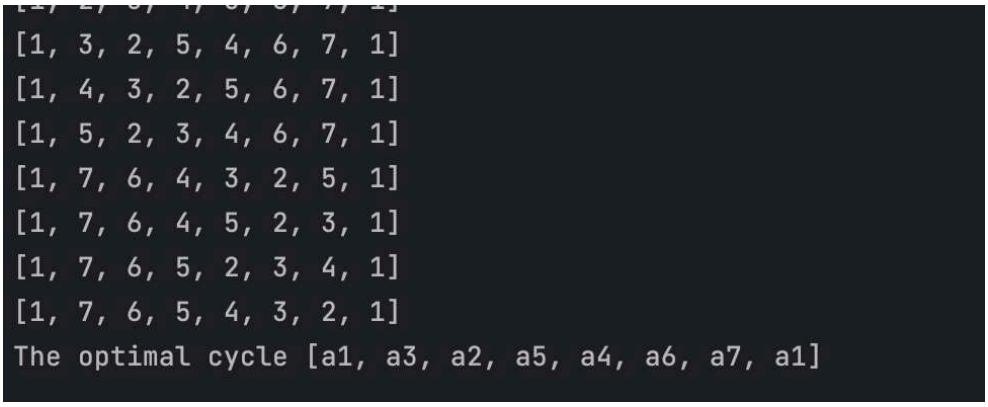


Рисунок 4.19 – Результат роботи програми з пошуку оптимального циклу Гамільтона

4.5 Висновки до розділу 4

1. Запропоновано й математично обґрунтовано розширення вхідного алфавіту моделі керуючого автомата реального часу через введення додаткових дуг у темпоральний граф переходів, що дозволяє встановлювати автомат у довільній стан за (n-1) тактів, де n - кількість станів автомата.

2. Розроблені та верифіковані HDL-моделі легкотестованих автоматів Мура через введення у HDL-код додаткового зовнішнього входу *bps*, що відповідає модифікації темпорального графу переходів. Моделювання проводилося з використанням системи моделювання EDA Playground. Аналіз апаратурних витрат показав, що оптимальним за апаратурними затратами є

введення додаткового режиму підвищення керованості станів автомату. Апаратурні затрати не перевищують 20-25% у залежно від типу автомату.

3. Були проаналізовані апаратурні витрати, пов'язані з введенням додаткового входу *bps* у HDL-моделі керуючих автоматів для різних варіантів часових переходів між станами автомата. Виявлено, що додаткові апаратурні витрати прямо залежать від складності функцій переходів, з урахуванням способу кодування станів автомата. Оцінювання апаратурних витрат проводилася через аналіз результатів автоматизованого синтезу HDL-моделей у САПР VIVADO для чіпу сімейства ZYNQ-7000.

4. Запропоновано математичний апарат для оптимізації обходу вершин темпорального графу з урахуванням складності переходів між вершина графу, який може використовуватися в процесі побудови оптимального циклу обходу темпорального графу при побудові неруйнівного ДЕ.

ВИСНОВКИ

У цьому дослідженні було розглянуто питання проєктування та верифікації цифрових пристроїв логічного керування з використанням HDL - шаблонів часових автоматів. Було вирішено низку практично-наукових задач, серед котрих – зменшення часу проєктування цифрових пристроїв реального часу через використання готових HDL-шаблонів, а також використання асерційних підходів, що зменшує час верифікації, а, отже, й час виходу пристрою на ринок.

Під час дослідження було досягнуто таких результатів:

1. Розроблено HDL-шаблони дискретних автоматів для проєктування пристроїв з обробкою зовнішніх подій. Це дало змогу моделювати цифрові пристрої, функціональність яких залежить від тривалості зовнішньої події та внутрішнього вікна прийому, та системи з обробки подій з певною тривалістю. Запропоновані HDL-шаблони було перевірено на тестових проєктах із подальшим синтезом у САПР для чіпів CPLD та FPGA. Успішний синтез підтвердив працездатність запропонованих шаблонів.

2. Розроблено моделі верифікації темпоральних параметрів часових автоматів через використання асерцій. Розроблено шаблони асерційних конструкцій, які покривають можливі сценарії поведінки часових автоматів, а саме: часові та умовно-часові переходи, обробка зовнішніх подій при наявності вікна прийому та обробка подій із мінімальною тривалістю. Було показано ефективність такого підходу, що дає змогу зменшити час верифікації та суттєво спрощує пошук помилки у HDL-коді. Перевірку моделей було зроблено за допомогою поведінкового моделювання в системі Aldec Reviera Pro з детальним аналізом часових діграм.

3. Удосконалено модель тестопридатних часових автоматів з використанням розширеного вхідного алфавіту. Наявну модель було удосконалено для моделей часових автоматів. Додаткові переходи дозволяють встановлювати керуючий автомат у будь-який стан упродовж (n - 1) тактів, де

n - кількість станів керуючого автомата. Це значно зменшує час, тривалість ДЕ і підвищує точність методів контролю та діагностики апаратурної складової автоматичних систем логічного управління. Проаналізовано додаткові апаратні витрати для різних типів темпоральних переходів при введені додаткових дуг на темпоральному графі.

4. Удосконалено модель побудови неруйнівного ДЕ за допомогою обходу всіх вершин темпорального графу переходів (цикл Гамільтона). Запропоновано математичний апарат оцінювання тестопридатності темпоральних моделей автоматів на основі досяжності станів.

5. Розроблено програмний модуль з інтерфейсом для створення темпорального графу переходів автомата з подальшою генерацією HDL-моделі у формі автоматного шаблону в синтезованій підмножині мов VHDL/Verilog. Також модуль надає можливість автоматичної генерації асерційних конструкцій SVA для заданої моделі, а також пошуку оптимального циклу Гамільтона для побудови неруйнівного ДЕ. Програмний модуль розроблений із використанням фреймворку Electron, а отже є кросс-платформеним рішенням.

Ринкова привабливість дослідження полягає у зменшені часу проєктування через використання запропонованих шаблонів часових автоматів та часу верифікації використанням асерційних конструкцій.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Buttazzo G. Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications. New York, New York: Springer, 2011. 524 p.

2. ДСТУ 2226-93. Автоматизовані системи. Терміни та визначення. – Чинний від 1994-07-01. –К.: Держспоживстандарт України, 2001. 45 с.

3. IEEE Std. 802.1AS-2020. IEEE Standard for Local and Metropolitan Area Networks--Timing and Synchronization for Time-Sensitive Applications, 2020.

4. IEC 615081:2010 (EQV). Functional safety of electrical/electronic/ programmable electronic safety-related systems. Part 1: General requirements. NSAI Standards, 2010. 17 p.

5. ДСТУ 3433-95. Надійність техніки. Моделі відмов. Терміни та визначення. – Чинний від 1996-01-01. – К.: Держспоживстандарт України, 2001. 45 с.

6. Ashok B. M. ASIC/SoC Functional Design Verification: A Comprehensive Guide to Technologies and Methodologies. Springer Cham, 2018. 328 p.

7. Faisal H., Jonathan M., Khizar K. The Art of Verification with SystemVerilog Assertions. Verification Central, 2006. 664 p.

8. Leena S., Leonard D. Advanced Verification Techniques: A SystemVerilog-Based Approach for Successful Tapeout. Springer, 2010. 394 p.

9. Grotker T., Liao S., Martin G., Swan S. System Design with SystemC. Kluwer Academic Publishers, 2002. 236 p.

10. Lamport L. Specifying Systems. Addison-Wesley Professional, 2002. 384 p.

11. IEEE Std. 1800–2005 IEEE Standard for SystemVerilog – Unified Hardware Design, Specification and Verification Language, 2005. 648 p.

12. IEEE Std. 1076-2008 IEEE Standard VHDL Language Reference

Manual, 2008. 640 p.

13. Zwolinski M. Digital System Design with VHDL, 2nd. ed. New York: Pearson, 2004. 384 p.

14. Chu P. P. FPGA Prototyping by VHDL Examples: Xilinx Spartan-3 Version. Cleveland: Wiley-Interscience, 2011. 468 p.

15. Хаханов В.И, Литвинова Е. И., Гузь О.А. Проектирование и тестирование цифровых систем на кристаллах. Харьков: ХНУРЭ, 2009. 484 с.

16. Hahanov V.I., Hyduke S.M., Gharibi W., Litvinova E.I., Chumachenko S.V., Hahanova I.V. Quantum Models and Method for Analysis and Testing Computing Systems. *2014 11th International Conference on Information Technology: New Generations*. Las Vegas, NV, 2014. P. 430-434.

17. Armstrong D. B. A Deductive Method for Simulating Faults in Logic Circuits. IEEE Transactions on Computers. May, 1972. Vol. C-21, No. 5. P. 464-471.

18. Hermann K. Real-Time Systems Design Principles for Distributed Embedded Applications. Springer Science+Business Media, 2011. 378 p.

19. Williams R. Real-Time Systems Development. Butterworth-Heinemann, 2005. 320 p.

20. Hassan G. Real-Time Software Design for Embedded Systems. Cambridge University Press, 2016. 602 p.

21. Broekman B., Notenboom E. Testing Embedded Software. Addison-Wesley, 2002. 368 p.

22. Arbib M.A. Theories of Abstract Automata. 1st Edition. Prentice-Hall, Inc. 1969. 412 p.

23. Baranov S. Logic and System Design of Digital Systems. Tallinn: TUT Press, 2008. 267 p.

24. Wakerly J.F. Digital Design: Principles and Practices. Pearson, 2005. 895 p.

25. Harris D., Harris S. Digital Design and Computer Architecture, Morgan Kaufmann, 2012. 720 p.

26. Vahid F. Digital Design with RTL Design, VHDL, and Verilog 2nd Edition. Wiley, 2010. 594 p.

27. Palnitkar S. Verilog HDL: A Guide to Digital Design and Synthesis 2nd Edition. Prentice Hall, 2005. 450 p.

28. Vahid F., Lusecky R. VHDL for Digital Design. Wiley, 2007. 192 p.

29. Kuo B.C. Digital Control Systems. 2nd Edition. Oxford University Press, 1995. 784 p.

30. Alur R., Dill D.L. A theory of timed automata. // *Theoretical Computer Science*, 1994. Vol .126, No. 2. P. 183–235.

31. Alur. R. Timed automata // Proceedings of the 11th International Conference on Computer Aided Verification, Trento, Italy, July 6–10, 1999. Springer, 1999. P. 8–22.

32. Merayo M. G., Núñez M., Rodríguez I. Formal testing from timed finite state machines. // *Computer Networks*, 2008. Vol.52, No 2. P. 432-460.

33. Merayo M. G., Núñez M., Rodríguez I. Extending FSM to specify and test timed systems with action durations and time-outs // *IEEE Transactions on Computers*. 2008. Vol. 57, No. 6. P. 835–844.

34. Zhigulin M., Yevtushenko N., Maag S. FSM-Based Test Derivation Strategies for Systems with Time-Outs // Proceedings of the 11th International Conference on Quality Software (QSIC 2011), Madrid, 2011. P. 141–149.

35. 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, P. 343–355.

36. Bresolin D., El-Fakih K., Villa T., Yevtushenko N. Deterministic Timed Finite State Machines: Equivalence Checking and Expressive Power // Proceedings of Intern Conf. GANDALF, 2014. P. 203–216.

37. Minimizing Deterministic Timed Finite State Machines / Bresolin D. et al. // In 14th IFAC Workshop on Discrete Event Systems WODES 2018. IFAC-PapersOnLine, 2018, Vol. 51, No. 7. P. 486-492.

38. Pedroni V. A. Finite state machines in hardware: theory and design

(with VHDL and SystemVerilog). Cambridge, MA: MIT Press., 2013. 338 p.

39. Design timed FSM with VHDL Moore pattern / Shkil A.S. et al. *Radio Electronics, Computer Science, Control*. 2020. Vol. 53, No 2. P. 137-148.

40. Wiegers K. Beatty Joy Software Requirements (Developer Best Practices) 3rd Edition. Developer Best Practices, 2013. 672 p.

41. Ramparison M., André E., Didier L. TCTL Model Checking Lower/Upper- Bound Parametric Timed Automata Without Invariants // Proceeding of International Conference Formal Modeling and Analysis of Timed Systems FORMATS 2018, September 4– 6, Biejing, China, 2018. P. 37–52.

42. Малиновский М. Л. Синтез безопасных автоматов с функциональной деградацией. *Управляющие системы и машины*, 2010. № 1. С. 84-91.

43. Hennie E.G. Fault detection experiments for sequential circuits // Proceeding of Fifth Symposium on Switching Circuit Theory and Logical Design, 1964. P. 95 – 110.

44. Zhang X., Yang M., Zhang J., Shi H., Zhang W. A study on the extended unique input/output sequence. *Information Sciences*, 2012. Vol. 203. P. 44-58.

45. Pakhomov Y. V., Miroshnyk M.A. Model of automated hardware diagnostics of remote energy systems management points. *Світлотехніка та електроенергетика: міжнар. наук.-техн. журнал*. Харків: ХНУМГ ім. О. М. Бекетова, 2017. № 3. С. 3-9.

46. Design automation of easy-tested digital finite state machines / Miroschnyk M. et al. *Radio Electronics, Computer Science, Control*, 2018. № 2. P. 117-124.

47. Bernholtz O., Vardi M. Y., Wolper P. An automata-theoretic approach to branching-time model checking // Proceedings of the Sixth International Conference on Computer-Aided Verification (CAV'94, June). Springer-Verlag, New York, 1994. 142-155 p.

48. Omer N. T., Petrenko A., Ramesh S. Multiple Mutation Testing from

Finite State Machines with Symbolic Inputs // Proceedings of the 29th IFIP WG 6.1 International Conference, ICTSS 2017, St. Petersburg, Russia, October 9-11, 2017. Springer Cham, 2017. P. 108-125.

49. Шкиль А.С., Серокурова А.С., Фастовец Г.П. Автоматизация поиска ошибок проектирования в HDL-моделях конечных автоматов. *АСУ и приборы автоматики*, 2014. Вып. 168. С. 43-52.

50. Шкиль А.С., Серокурова А.С., Кулак Э.Н. Диагностирование HDL-моделей микропрограммных автоматов. *АСУ и приборы автоматики*, 2015. Вып. 172. С. 22-31.

51. Simulation and Testing of Deterministic Finite Automata Machine / Vayadande K. et al. *International Journal of Computer Sciences and Engineering*. 2022. Vol. 10, No 1. P. 13-17.

52. Design automation of testable finite state machines / Miroschnyk M. et al. Proceedings of the 2017 IEEE East-West Design & Test Symposium (EWDTS), September 29 – October 2, 2017, Novi Sad, Serbia. IEEE, 2017. P. 1-6.

53. Accelerating Finite State Machine-Based Testing using Reinforcement Learning / Turker U. et al. *IEEE Transactions on Software Engineering*. 2024. P. 1-22.

54. Salauyou V. Structural Models for Fault Detection of Moore Finite State Machines // Proceedings of the Eighteenth International Conference on Dependability of Computer Systems DepCoS-RELCOMEX, July 3–7, 2023, Brunów, Poland. Springer Cham, 2023. P. 231–241.

55. Salauyou V. Fault Detection of Moore Finite State Machines by Structural Models // Proceedings of Computer Information Systems and Industrial Management

22nd International Conference, CISIM 2023, Tokyo, Japan, September 22–24, 2023. Springer Cham, 2023. P. 394–409.

56. Sovov'ev V.V., Klimovisz A. Transformation of a Mealy Finite-State Machine into a Moore Finite-State Machine by Splitting Internal States. *Journal of Computer and Systems Sciences International*. 2010, No. 49. P. 900 – 908.

57. Timo O. N., Prestat D., Rollet A. Multiple Mutation Testing for Timed Finite State Machine with Timed Guards and Timeouts // Proceedings of the 31st IFIP WG 6.1 International Conference, ICTSS 2019, Paris, France, October 15–17, 2019. Springer Cham, 2019. P. 104-120.

58. Хаханов В.И., Ковалев В.Е., Ханько В.В., Масуд М.Д. Система генерации тестов для проектирования цифровых автоматов в среде ACTIVE-HDL. *АСУ и приборы автоматики*, 2000. Вып. 111. С. 15-22.

59. Шкиль А.С., Сыревич Е.Е., Альмадхоун. С. Поиск ошибок проектирования в HDL-моделях цифровых автоматов. *Вестник Херсонского НТУ*. 2013. №2 (46). С. 377-383.

60. Zhigulin M., Yevtushenko N., Maag S, Cavalli A. FSM-Based Test Derivation Strategies for Systems with Time-Outs // Proceeding of 11th International Conference on Quality Software. Madrid, 2011. P. 141-149.

61. Melnichenko I., Kamkin I., Smolov S. An Extended Finite State Machine-Based Approach to Code Coverage-Directed Test Generation for Hardware Designs // Proceedings of the Institute for System Programming, 2015, Vol. 27, No. 3. P. 161-182.

62. Model-Based Testing of Reactive Systems : Advanced Lectures / Manfred B. et al. Heidelberg : Springer Berlin, 2005. 664 p.

63. Gnessi S., Margaria T. Formal Methods for Industrial Critical Systems: A Survey of Applications. Wiley-IEEE Press, 2013. 292 p.

64. Merayo M.G., Nunez M., Rodriguez I. Formal Testing from Timed Finite State. *Computer Networks,* 2008, Vol. 52, No. 2. P. 432-460.

65. Шкиль А.С., Кучеренко Д.Е., Кулак Э. Н., Мирошник М.А. Обнаружение ошибок проектирования в HDL-моделях конечных автоматов с использованием синхронизирующих последовательностей. *Радиоэлектроника и информатика*, 2016 No. 3(74). С. 39-46.

66. El-Fakih K., Yevtushenko N., Simao A. A practical approach for testing timed deterministic finite state machines with single clock. *Science of Computer Programming*, 2014, Vol. 80. P. 343–355.

67. Bergeron J. Writing Testbenches using SystemVerilog. New York : Springer NY, 2006. 412 p.

68. Sprear C., Tumbush G., SystemVerilog for Verification. New York : Springer NY, 2012. 464 p.

69. A Single-Stage RISC-V Processor to Mitigate the Von Neumann Bottleneck / Kanamoto T. *et al*. *2019 IEEE 62nd International Midwest Symposium on Circuits and Systems (MWSCAS)*. Dallas, TX, USA, 2019. P. 1085-1088.

70. Rosenberg S. A Practical Guide to Adopting the Universal Verification Methodology. Cadence Design Systems, 2010. 296 p.

71. Salemi R. The UVM Primer: A Step-by-Step Introduction to the Universal Verification Methodology. Boston Light Press, 2013. 194 p.

72. A secure design-for-test infrastructure for lifetime security of SoCs / Backer J. et al. // Proceedings IEEE International Symposium on Circuits and Systems (ISCAS'2015), 2015. P. 37–40.

73. Zorian Y., Tshagharyan G., Harutyunyan G., Shoukourian S. Securing. Test Infrastructure of System-on-Chips // Proceedings of IEEE East-West Design & Test Symposium (EWDTS'2016), 2016. P. 29–32.

74. Zorian Y., Sargsyan D., Harutyunyan G., Shoukourian S. Securing. Automated Flow for Test Pattern Creation for IPs in SoC // Proceedings of IEEE East-West Design & Test Symposium (EWDTS'2017), 2017. P. 21–24.

75. Sanghavi A. What is formal verification? *EE Times Asia*. 2010. P 109-118.

76. Gupta A. Formal hardware verification methods: A survey. *Formal Methods in System Design*. 1992, Vol. 1. P. 151-238.

77. McFarland M.C. Formal verification of sequential hardware: a tutorial. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*. 1993, Vol 12, No. 5. P. 633-654.

78. Udaya Shankar A. An introduction to assertional reasoning for concurrent systems. *ACM Computing Surveys*. 1993, Vol. 25, No. 3. P. 225-262.

79. Greenstreet M.R., Kern C. Formal Verification In Hardware Design: A Survey. *ACM Transactions on Design Automation of Electronic System.* 1999, Vol. 4, No. 2. P. 123-193.

80. Bryant R.E. Symbolic Boolean manipulation with ordered binary-decision diagrams. *ACM Computing Surveys*. 1992, Vol. 24, No. 3. P. 293-318.

81. Bryant R. E. Graph-based algorithms for Boolean function manipulation. *ACM Computing Surveys*. 1986, Vol. C-35, No. 8. P. 677 - 691.

82. Bernholtz O. Vardi M. Y., Wolper P. An automata-theoretic approach to branching-time model checking // Proceeding of International Conference on Computer Aided. Verification. Stanford, CA, USA, 1994. P. 142-155.

83. Hu A.J., Dill D.L. Reducing BDD Size by Exploiting Functional Dependencies // Proceeding of 30th ACM/IEEE Design Automation Conference, 14-18 June 1993. Dallas, TX, USA, 1993. P. 266-271.

84. Jain J., Abraham J. A., Bitner J., Fussell D. S. Probabilistic verification of boolean functions. *Formal Methods in System Design*. 1996, Vol. 1, No. 1. P. 63-115.

85. Thomas W. Automata on infinite objects: Handbook of Theoretical Computer Science (vol. B): Formal Models and Semantics / J. van Leeuwen. Ed. MIT Press, Cambridge, MA, 1990. P. 133-191.

86. Kumar R., Schneider K., Kropf T. Structuring and automating hardware proofs in a higher-order theorem-proving environment. *Formal Methods in System Design*. 1993, Vol. 2, No. 2. P. 165-223.

87. Navabi Z. Assertion-Based Verification for System-Level Designs // Proceedings of 15th Int'l Symposium on Quality Electronic Design (ISQED), 3-5 March 2014. Santa Clara, CA, 2014. P. 582-588.

88. Seligman E., Schubert T., Achutha Kiran Kumar M. V. Formal Verification: An Essential Toolkit for Modern VLSI Design. Morgan Kaufmann, 2015. P. 408.

89. Braidbant T., Chipala A. Formal Verification of Hardware Synthesis. // Proceedings of 25th International Conference, CAV 2013, Saint Petersburg,

Russia, July 13-19, 2013. Springer, Berlin, Heidelberg, 2013. P. 213-228.

90. Gnesi S., Margaria T. Formal Methods for Industrial Critical Systems: A Survey of Applications. Wiley-IEEE Press, 2013. 292 p.

91. Baier C., Katoen J.P. Principles of Model Checking. MIT Press, 2008. 975 p.

92. Handbook of Model Checking / ed. Edmund M. Clarke et al. Springer Cham, 2018. 1212 p.

93. Assertion-based mechanism for the functional verification of the digital design / Hahanov V. et al. // Proceedings of IEEE East-West Design & Test Workshop (EWDTW'05), Odesa, Ukraine, September 15-19, 2005. P. 261-265.

94. IEEE Std. 1850–2010 IEEE Standard for Property Specification Language (PSL), 2010. 182 p.

95. Eisner C., Fisman D. A Practical Introduction to PSL. Springer, 2006. 240 p.

96. Cohen B. Using PSL/Sugar with Verilog and VHDL, Guide to Property Specification Language for ABV. VhdlCohen Publishing, 2003. 186 p.

97. Ramanathan M., Vijayaraghavan S. A Practical Guide for SystemVerilog Assertions. Springer New York, NY, 2005. 334 p.

98. Cerny E., Dudani S., John H., Korchemny D. SVA: The Power of Assertions in SystemVerilog. Springer Cham, 2014. 590 p.

99. Mehta A. B. SystemVerilog Assertions and Functional Coverage. Springer New York, NY, 2013. 356 p.

100. Hahanov V., Zaychenko S., Varchenko V. Method for Diagnosing SoC HDL-code // Proceedings Of 12th IEEE East-West Design & Test Symposium (EWDTS 2014), Kyiv, Ukraine, September 26-29, 2014. P. 97-102.

101. Хаханов В.И., Каминская М.А., Зайченко С.А. Верификация цифровых устройств на основе использования анализа тестопригодности и ассерционных библиотек. *Автоматизированные системы управления и приборы автоматики.* 2007, No. 140. С. 75-83.

102. Fitting M. C. Intuitionistic logic, model theory and forcing. Amsterdam : North-Holland Pub. Co, 1969. 191 p.