УДК 004.05, 004.42

О.О. ЛЕТИЧЕВСЬКИЙ, В.С. ПЕСЧАНЕНКО, В.С. ХАРЧЕНКО, В.А. ВОЛКОВ, О.М. ОДАРУЩЕНКО

МОДЕЛЬНИЙ СПОСІБ РОЗРОБЛЕННЯ АЛГОРИТМІВ ЦИФРОВИХ СИСТЕМ НА ПРОГРАМОВАНИХ ЛОГІЧНИХ ІНТЕГРАЛЬНИХ СХЕМАХ

Анотація. Розглянуто сучасні тенденції в галузі автоматизованого розроблення апаратного забезпечення, зокрема, розроблення цифрових систем з використанням програмованих логічних інтегральних схем на прикладі програмованих користувачем вентильних матриць. Запропоновано модельний метод розроблення, в якому використано алгебраїчну модель специфікацій дизайну, вимог та бінарного коду для застосування формальних методів верифікації, модельного тестування та методів алгебраїчного зіставлення. Специфікаціями алгебраїчної моделі апаратного забезпечення слугує алгебра поведінок, визначена на множині дій та поведінок.

Ключові слова: програмовані користувачем вентильні матриці, символьне моделювання, алгебраїчне зіставлення, алгебра поведінок.

ВСТУП

Модельний спосіб розроблення передбачає використання моделей різного рівня абстракції на кожному етапі розроблення апаратних або програмних систем. У разі застосування цього підходу суттєво зростає можливість автоматизації проєктування, що сприяє значному підвищенню надійності системи та надає змогу здійснити оптимізацію тестування системи, її формальну верифікацію та перевірку відповідності вимогам.

У цій статті розглянуто процес розроблення цифрових систем з використанням програмованої логічної інтегральної схеми (ПЛІС), зокрема, її різновиду — програмованої користувачем вентильної матриці (ПКВМ), відомої як FPGA (Field-Programmable Gate Array) [1].

Для проєктування цифрових систем на ПКВМ реалізують чітко визначений процес розроблення системи та її компонент. Використовують різні моделі життєвого циклу розроблення згідно з відповідною послідовністю та вимогами стандартів. Зазвичай процес розроблення на ПКВМ для критичних систем ґрунтується на так званій V-моделі [2], яка складається з етапів, зазначених на рис. 1.

На етапі розроблення вимог здійснюється збір вимог до майбутнього пристрою, що має бути реалізований на ПКВМ. Вимоги можуть бути зібрані у вигляді текстових або табличних даних, якими скористаються під час розроблення інженери та тестувальники. Спеціальної мови представлення вимог до алгоритмів немає, але підтримку автоматизації деяких етапів високорівневого розроблення та відповідної генерації коду з рівня вимог реалізовано у Mathworks, Intel (HLS Compiler), Xilinx (HDL coder). Етап дизайну полягає в конструюванні архітектури майбутнього продукту та

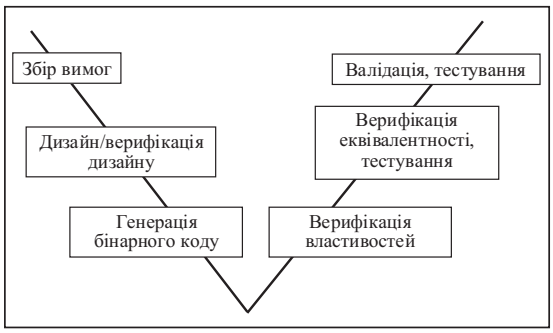

Рис. 1. Основні етапи розроблення апаратного забезпечення

його компонент за допомогою мов HDL (Hardware Design Level). До таких мов відносять VHDL [3], System Verilog [4].

Основні компанії, що очолюють напрям автоматизації дизайну електроніки (EDA — Electronic Automation Design), такі як Cadence [5], Xilinx [6] та Synopsis [7], застосовують власні інструменти симуляції та розроблення. Особливо популярними є комплексні рішення та платформи, наприклад середовище Potium [8] фірми Cadence. Здебільшого для потреб дизайну ПКВМ використовують інтегровані середовища розробників, що містять засоби візуалізації, налаштування та трансляції таких мов розроблення, як VHDL та System Verilog.

Відповідні транслятори в бінарний код, розроблені, наприклад, компанією Intel (Quartus [9]), використовують для створення коду, що завантажується безпосередньо у ПКВМ. На етапі тестування також застосовують верифікаційні інструменти аналізу часових властивостей та живлення. Для розробників апаратного забезпечення є важливим здійснення перевірки еквівалентності специфікацій на різних рівнях абстракції, яка може бути порушена під час трансляції або оптимізації.

Ми розглядаємо доповнення EDA модельним методом розроблення, що підвищує ступінь надійності та безпеки апаратного забезпечення. Цей метод передбачає використання формальних алгебраїчних моделей та генерацію каркасу специфікацій на кожному етапі розроблення.

Під час розгляду формальної моделі вимог створюють модель найвищого рівня абстракції, яка використовується в системному тестуванні. Для моделі можна застосовувати відповідні формальні мови, наприклад BPMN (Business Process Model and Notation) [10], UCM (Use Case Maps) [11], UML (Unified Modeling Language) [12]. Маючи формальну модель, можна вже на початковому етапі забезпечити верифікацію вимог та генерацію тестів для використання їх у системному тестуванні та тестуванні моделі дизайну. Генерація каркаса дизайну з формальних вимог забезпечує створення коду у формальній мові дизайну для подальшої деталізації розробником. Деталізований код можна перевірити за допомогою тестів, згенерованих на відповідному етапі, та верифікувати відповідно до необхідних властивостей. З моделі дизайну автоматично генерується бінарний код, який перевіряють за допомогою тестів рівня вимог, верифікують відповідно до певних властивостей та досліджують на вразливості.

Розглянемо процес розроблення апаратного забезпечення, в якому мову алгебри поведінок використано як мову моделей, а її методи застосовано для верифікації, тестування та аналізу безпеки та вразливостей.

АЛГЕБРА ПОВЕДІНОК ЯК ФОРМАЛЬНА МОВА МОДЕЛЕЙ

Розглянемо моделі в контексті інсерційного моделювання [13], де модельовані сутності визначаються як агенти, що взаємодіють у деякому середовищі. Середовище є атрибутним, тобто визначається станами агентів та власне середовища за допомогою формул над типізованими атрибутами.

Алгебра поведінок [14] є алгеброю, елементи якої — це дії та поведінки агентів. Операціями в алгебрі поведінок є префіксінг («.»), недетермінований вибір («+»), послідовна («;») та паралельна («||») композиція.

Розглянемо приклад фрагмента проєкту FPGA, що реалізує деякий алгоритм, у термінах алгебри поведінок. На рис. 2 наведено схему, яка відображає взаємодію програмованих елементів. Програмовані елементи є агентами, що взаємодіють один з одним за допомогою сигналів, які змінюють стани середовища. З використанням операцій алгебри поведінок можна описати загальну поведінку фрагмента алгоритму.

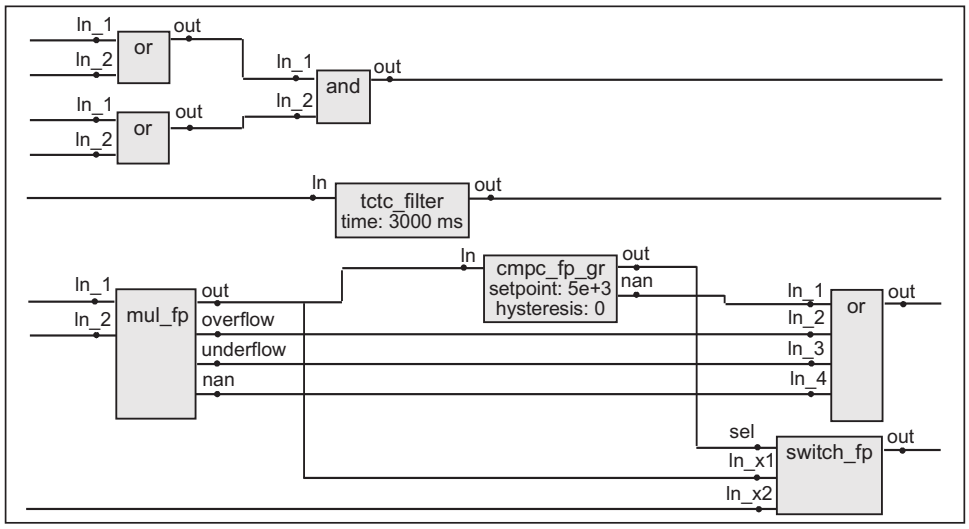

Рис. 2. Фрагмент алгоритму FPGA

Поведінка визначається таким рівнянням алгебри поведінок B0:

$$B0 = P1 \parallel P2 \parallel P3.$$

Це рівняння є паралельною композицією трьох інших поведінок P1, P2 та P3. Кожна поведінка являє собою відповідне рівняння, що містить композиції інших поведінок та дій агентів. Розглянемо поведінку P1:

P1 = (Aor2(or2_1, ENV, ENV, and) || Aor2(or2_2, ENV, ENV, and));
Aand (and, or2_1, or2_2, ENV).

Поведінки та дії можуть бути параметризовані іменами агентів, які беруть участь у дії або поведінці. Ім'я може належати ключовому агенту, тобто такому, що виконує дію, та іншим агентам, з якими ключовий агент взаємодіє. У P1 ми розглядаємо два види дії Aor2 та дію Aand. Дві дії Aor2 виконуються паралельно, а далі послідовно виконується Aand. Імена агентів, що взаємодіють у цьому фрагменті, є такими: or2_1, or2_2, and, tctc_filter, cmpc_fp_gr, switch_mp, mul_fp, or4. Також ми маємо агент ENV, який втілює все, що є поза межами фрагменту. Тоді, наприклад, параметризована дія Aor2(or2_1, ENV, ENV, and) означає, що в параметрах є ключовий агент or2_1, що виконує дію, входом дії є два сигнали із зовнішнього середовища ENV, а вихідний сигнал приймає агент and. Аналогічно розпишемо інші поведінки P2 і P3:

P2 = Atctc_filtered(tctc_filter, ENV, ENV, 3000) +
Atctc_not_filtered(tctc_filter, ENV, ENV, 3000),
P3 = Amul_fp(mul_fp, ENV, ENV, cmpc_fp_gr, switch_fp, or4, or4, or4);
Acmpc_fp_gr (cmpc_fp_gr, mul_fp, switch_fp, or4, 5000, 0);
(Aor4(or4, cmpc_fp_gr, mul_fp, mul_fp, mul_fp, ENV) ||
Aswitch_fp(switch_fp, cmpc_fp_gr, mul_fp, ENV, ENV))

Усі поведінки P1, P2, P3 представлено як композицію дій відповідних агентів, хоча вони можуть містити композицію дій з іншими поведінками, які також можна параметризувати. Семантика дій має бути також представлена у формальній мові у вигляді такої трійки: передумова, процесна компонента та післяумова. Цю семантику наведено у табл. 1.

Таблиця 1

Дія з параметрами	Передумова	Процесна компонента	Післяумова
Acmpc_fp_gr(s, x1, x2, y, setpoint : real, hysteresis : real) Локальні параметри дії: (in : real, out : bool, nan : bool)	1	receive(x1 : signal(in)), send(x2 : signal (out), signal (y : nan))	out = (in > setpoint – hysteresis)
Aor2(s, x1, x2, x3) Локальні параметри дії: (in_1, in_2, out : bool)	1	receive(x1 : signal (in_1)), receive(x2 : signal (in_2)), send(y : signal(out))	out = (in_1 || in_2)
Aor4(s, x1, x2, x3, x4, out) Локальні параметри дії: (in_1, in_2, in_3, in_4, out : bool)	1	receive(x1 : signal (in_1)), receive(x2 : signal (in_2)), receive(x3 : signal (in_3)), receive(x4 : signal (in_4)), send(y : signal(out))	out = (in_1 || in_2 || in_3 || in_4)
Aand(s, x1, x2, y) Локальні параметри дії: (in_1, in_2, out : bool)	1	receive(x1 : signal (in_1)), receive(x2 : signal (in_2)), send(y : signal(out))	out = (in_1 && in_2)
Amul_fp(s, x1, x2, y0, y1, y2, y3, y4) Локальні параметри дії: (in_1 : real, in_2 : real, out : real, overflow : bool, underflow : bool, nan : bool)	1	receive(x1 : signal(in_1)), receive(x2 : signal(in2)), send(y0 : signal(out)), send(y1 : signal(out)), send(y2 : signal(overflow)), send(y3 : signal(underflow)), send(y4 : signal(nan))	out = (in_1 * in_2); overflow = (out > maxReal32); underflow = (out < minReal32)
Atctc_filtered(s, x, y, n) Локальні параметри дії: (in, out : bool)	(TIMER == 0)	receive(x : signal (in)), send(y : signal (out))	in == out
Atctc_filtered(s, x, y, n) Локальні параметри дії: (in, out : bool)	(TIMER == 1)	receive(x : signal (in))	1
Aswitch_fp(s, x1, x2, x3, y) Локальні параметри дії: (sel : bool, in_x1 : real, in_x2 : real, out : real)	1	receive(x1 : signal(in_x1)), receive(x2 : signal(in_x2)), receive(x3 : signal(sel)), send(y : signal(out))	(sel == 0) && (out == in_x1) || (sel == 1) && (out == in_x2)

Кожна дія є параметризованою відносно агента, який її виконує, та вхідних і вихідних даних. Передумова визначає, чи можна застосувати певну дію за певних значень параметрів. Якщо параметри є конкретними, то передумова дії має бути істинною, а якщо — символьними або довільними, то передумова має бути виконуваною для деяких значень параметрів передумови. Післяумова визначає, як зміниться середовище, зокрема вихідні дані. Процесна компонента дії ілюструє дію отриманням або посиланням сигналу до інших агентів.

АЛГЕБРАЇЧНИЙ ПІДХІД У МОДЕЛЬНОМУ МЕТОДІ РОЗРОБЛЕННЯ

Наявність алгебраїчної моделі на кожному з етапів надає змогу використовувати методи алгебри поведінки для верифікації та аналізу властивостей моделей. На рис. 3 наведено життєвий цикл розроблення з використанням алгебраїчної моделі.

На етапі вимог алгебраїчну модель можна отримати у два способи — або створити вручну у вигляді рівняння алгебри поведінок із діями, або транслювати з мов верхнього рівня, якими записують вимоги. Отже, на цьому етапі можна використовувати методи алгебри поведінки для верифікації властивостей вимог, при цьому модель може слугувати тестовою на завершальному етапі розроблен-

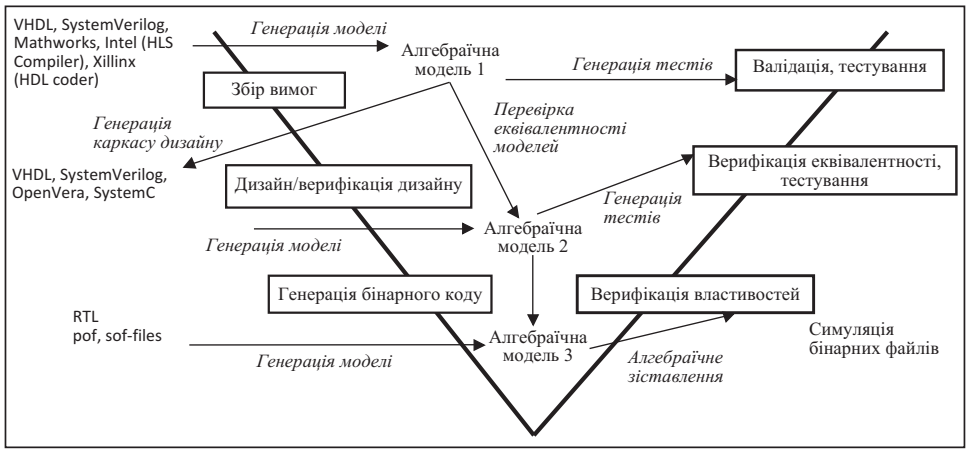

Рис. 3. Спосіб розроблення з використанням алгебраїчних моделей

ня, яким є тестування та валідація (перевірка системної функціональності). Маючи модель, можна генерувати каркас мовою дизайну для подальшої деталізації, якщо мови дизайну та вимог не збігаються.

Алгебраїчна модель, отримана на етапі дизайну методом трансляції з мови дизайну, може слугувати як об'єктом верифікації, так і тестовою моделлю для тестування бінарного коду. Отриманий бінарний код теж можна транслювати в алгебраїчну модель для аналізу безпеки, вразливостей та властивостей, які перевіряють на найнижчому рівні, тобто напруги, температури, електромагнітних властивостей тощо.

Усі три моделі різних рівнів абстракції можна перевірити за допомогою формальних методів на еквівалентність.

ВЕРИФІКАЦІЯ МОДЕЛЕЙ

Верифікація моделі полягає в дослідженні поведінкових властивостей безпеки або життєздатності.

Формальні методи алгебри поведінок, які використовуються для доведення досяжності властивостей, що перевіряються, можуть бути статичними, динамічними, комбінованими та частково динамічними. Входом у процедуру верифікації є модель у вигляді алгебри поведінок та формула властивості, що перевіряється, а виходом є вердикт, який підтверджує її досяжність, та сценарій, реалізація якого призводить до відповідного стану.

До статичних методів відносять такі, для яких достатньо доведення властивості виконуваності, що задана формулою в базовій мові, тобто існують такі значення атрибутів формули, для яких формула може бути істинною. Доведення властивості виконуваності здійснюється за допомогою таких машин розв'язування, як Microsoft Z3 [15] та cvc4 [16].

Динамічні методи застосовують до поведінкових властивостей, тобто властивостей, де формула представлена за допомогою виразів алгебри поведінки, та семантики дій, де формула представлена як вираз в базовій мові. Такі методи потребують символьного моделювання та розв'язання системи поведінкових рівнянь. Усі зазначені методи алгебри поведінки наведено в роботах [17, 18].

Комбінованими методами є ті, для яких доведення виконуваності формул не є достатнім і потрібно довести досяжність формули символьним моделюванням. Частково в динамічних методах використовують техніку генерації інваріантів [19] в алгебрі поведінок.

Основними методами верифікації, що застосовуються для апаратного забезпечення в межах алгебри поведінок, є такі:

— статична перевірка несумісності (недетермінізму) в алгоритмах схем шляхом перевірки виконуваності формули $\bigwedge_i a(i) = 0$, де $a(i)$ — передумова дії;

— перевірки неповноти (тупики) в алгоритмах схем шляхом перевірки виконуваності формули $\bigvee_i a(i) = 0$;

— перевірка сумісності часових властивостей та проблем синхронізації. Ці властивості перевіряють шляхом символьного моделювання поведінкових рівнянь із деякого початкового стану, заданого формулою над атрибутним середовищем. Під час моделювання передумову разом із поточним станом середовища перевіряють на виконуваність. Якщо вона є, то стан середовища змінюється відповідно до післяумови;

— перевірка досяжності критичних станів, заданих формулою над атрибутним середовищем, що виражають стан «голодування», коли всі агенти чекають отримання сигналів, стан порушення обмежень або інші властивості безпеки, що можуть бути задані користувачем. Такі властивості можуть бути як локальними (анотації в коді дизайну, англ. assertion), так і глобальними;

— перевірка перегонів сигналів, коли різна послідовність отримання сигналів створює різні стани середовища, за допомогою символьного моделювання або генерації інваріантів;

— перевірка властивостей життєдіяльності, що виражають досяжність бажаного стану, за допомогою символьного моделювання.

МОДЕЛЬНЕ ТЕСТУВАННЯ

Алгебраїчна модель може слугувати моделлю для генерації трас, які використовуються для створення тестового набору у відповідному середовищі тестування. Перевірку відповідності артефактів моделі можна здійснювати на кожній фазі розроблення. Тестувати можна як специфікації дизайну, так і бінарний код. Основними методами генерації тестів є пряме та обернене символьне моделювання. Вимоги до генерації тестів визначають необхідне покриття тестами рядків коду.

Покриття також визначається моделлю, яка може передбачати вичерпне тестування, тобто генерацію всіх можливих станів поведінки моделі, покриття дій у моделі або покриття всіх переходів між діями. Для апаратного забезпечення тестом вважають послідовність прийомів та відправлень сигналів між агентами. Під час виконання тестів розрізняють інстанцію, що тестує, та інстанцію, що тестується. У цьому процесі здійснюють відправлення сигналів на інстанцію, що тестується, та порівнюють результат з очікуваним.

У разі використання алгебри поведінок розглядають такі види тестування:

— тестування методом «чорної скриньки», тобто за допомогою набору тестів, де код інстанції, що тестується, є невідомим;

— тестування методом «білої скриньки», де процес визначається символьним виконанням паралельної композиції моделі та коду. Під час тестування порівнюють на еквівалентність стани середовища.

КІБЕРБЕЗПЕКА: ПОШУК ВРАЗЛИВИХ ПОВЕДІНОК

За допомогою моделей алгебри поведінок можна оцінювати здатність алгоритму витримувати атаки зловмисників та відповідно знаходити вразливі поведінки в моделях. До таких уразливостей відносять витік даних або можливість перезапису інформації у ПКВМ. Під час оцінювання опірності до атак, що призводять до помилок, аналізують здатність системи зберігати конфіденційну інформацію.

Уразливу поведінку або поведінку зловмисника моделюють за допомогою шаблонів. Шаблоном є вираз в алгебрі поведінок, що містить дії, які представляють уразливості, та невідомі поведінки, що призводять до них. Наприклад, якщо маємо дії читання інформації A та отримання сигналу, що містить прочитану інформацію, то шаблон має вид такої поведінки:

$$S = read(A).X; send(A),$$

де X — довільна поведінка.

Задача полягає в пошуку поведінки X у моделі, та тих шляхів, що ведуть до цих дій. Таку задачу розв'язують за два етапи. Перший етап — це розв'язання рівняння алгебри поведінок, що визначає модель. Наприклад, нехай $B0$ — система рівнянь алгебри поведінок:

$$B0 = R(a1, a2, \ldots, B1, B2, \ldots,),$$

$$B1 = R(a11, a12, \ldots, B11, B12, \ldots), \ldots,$$

де $B1, B2, \ldots B11, B12, \ldots$ — поведінки, $a1, a2, \ldots$ — дії.

Потрібно знайти таку поведінку Y, що $B0 = Y;S$ і S — шаблон уразливості. Результатом може бути послідовність дій, яка веде до вразливості, що складає деяку трасу або множину трас. Визначивши трасу, маємо ще довести, що такий сценарій поведінки є досяжним, тобто є відповідні значення атрибутів середовища. Доведення досяжності є другим етапом задачі та реалізується з використанням методу символьного моделювання. Маючи трасу з відповідними формулами станів середовища, можна згенерувати конкретні значення та отримати експлойт (exploit), тобто дані, для яких здійснення атаки є можливим.

Отже, алгебра поведінок є важливою технологією в модельному способі розроблення і порівняно із симуляцією, яку найчастіше використовують для верифікації ПКВМ, забезпечує більше можливостей для аналізу надійності апаратного забезпечення. Використання алгебраїчних методів сприяє значному зменшенню відсоткової частки помилково позитивних артефактів і підвищенню точності виявлення вразливостей та артефактів верифікації. Алгебраїчні моделі в розробленні систем на ПКВМ можна перевіряти на еквівалентність за допомогою формальних методів, що гарантує правильний результат на відміну від симуляції.

Наступним логічним кроком досліджень є поєднання модельних методів з методами верифікації, які ґрунтуються на ін'єктуванні дефектів і/або вразливостей у проєкти ПКВМ, а також визначенні відповідних метрик для оцінювання повноти і достовірності перевірки [20–22]. До того ж, цікавим з теоретичного і практичного погляду є використання формальних методів для створення диверсних проєктів та перевірки наявних проєктів на диверсність [23, 24].

ВИСНОВКИ

Апробацію методу здійснено у межах виконання промислових проєктів НВП «Радій», м. Кропивницький. Під час розв'язання задач формальної верифікації отримано такі вагомі результати:

- побудовано алгебраїчний опис обраного фрагмента специфікації вимог для електронних компонент проєкту, які реалізовано мовою VHDL. Для фрагмента специфікації вимог виконано аналіз наявності порушень характеристик несуперечливості, повноти та ін., що є особливо важливим для процесів верифікації цифрових пристроїв. Одержано тестові набори для здійснення функціональної (поведінкової) симуляції роботи обраного бібліотечного компонента. Цей результат надає змогу знизити суб'єктивні ризики виконання поведінкового тестування, тобто ризики якості формулювання вимог розробником та їхнього розуміння тестувальником, а також якості та кількості тестових наборів, які розробляються та виконуються з цифровим пристроєм.

• виконано алгебраїчний опис набору сценаріїв логік користувача User Application Logic (UAL), які виконуються програмованим логічним контролером (ПЛК), при цьому визначено ймовірні критичні шляхи виконання цих логік.

Загалом отримані результати є важливими та корисними для підвищення надійності та функціональної безпеки [25] цифрової інформаційно-керувальної платформи RadICS (виробництво НВП «Радій», м. Кропивницький) під час реалізації процесів її розроблення, тестування, сертифікації та ліцензування на відповідність нормам міжнародних стандартів.

СПИСОК ЛІТЕРАТУРИ

1. FPGA. URL: www.intel.com/content/www/us/en/products/programmable/fpga/new-to-fpgas/resource-center/overview.html.
2. V-model. URL: https://www.testingexcellence.com/v-model-in-software-testing/.
3. VHDL. URL: www.intel.com/content/www/us/en/programmable/support/support-resources/design-examples/design-software/vhdl.html.
4. SystemVerilog. URL: www.asic-world.com/systemverilog/tutorial.html.
5. Cadence. URL: www.cadence.com/.
6. Xilinx. URL: www.xilinx.com.
7. Synopsis. URL: www.synopsys.com.
8. Potium. URL: www.cadence.com/en_US/home/tools/system-design-and-verification/fpga-basedprototyping/protium-s1-fpga-based-prototyping-platform.html.
9. Quartus. URL: www.intel.com/content/www/us/en/programmable/downloads/download-center.html
10. BPMN. URL: www.bpmn.org/.
11. ITU-T Recommendation, Z.151, User Requirements Notation (URN) — Language definition.
12. UML. URL: www.uml.org/.
13. Letichevsky A., Letychevskyi O., Peschanenko V. Insertion modeling and its applications. *Computer Science Journal of Moldova.* 2016. Vol. 24, Iss. 3. P. 357–370.
14. Letichevsky A., Gilbert D. A model for interaction of agents and environments. In: *Recent Trends in Algebraic Development Technique. LNCS.* Bert D., Choppy C. (Eds.). Berlin; Heidelberg: Springer-Verlag, 2000. Vol. 1827. P. 311–328.
15. Z3 decision procedure. URL: https://github.com/Z3Prover/z3.
16. CVC4 decision procedure. URL: http://cvc4.cs.stanford.edu.
17. Letichevsky A. Algebra of behavior transformations and its applications. In: *Structural Theory of Automata, Semigroups, and Universal Algebra, NATO Science Series II: Mathematics, Physics and Chemistry*. Kudryavtsev V.B., Rosenberg I.G. (Eds.). Dordrecht: Springer, 2005. Vol. 207. P. 241–272.
18. Letychevskyi O., Letichevsky A. Predicate transformers and system verification. *Proc. Third International Workshop on Symbolic Computation in Software Science (SCSS 2010)*. (29–30 July 2010, Hagenberg, Austria). Hagenberg, 2010. P. 148-149.
19. Letychevskyi O., Letichevsky A., Godlevsky A., Guba A., Peschanenko V., Kolchin A. Invariants in symbolic modeling and verification of requirements. *Proc. 9th Conference Computer Science and Information Technologies (CSIT 2013)*. (23–27 September 2013, Yerevan, Armenia). Yerevan, 2013. P. 23–27.
19. Reva L., Kulanov L., Kharchenko V. Design fault injection-based technique and tool for FPGA projects verification. *Proc. 9th East-West Design & Test Symposium (EWDTS 2011*). (9–12 September 2011, Sevastopol, Ukraine). Sevastopol, 2011. P. 1–6.
21. Kharchenko V., Odarushchenko O., Sklyar V., Ivasyuk A. Fault insertion testing of FPGA-based NPP I&C systems: SIL certification issues. *Proc. International Conference on Nuclear Engineering (ICONE 22).* (11–14 July 2014, Prague, Czech Republic). Prague, 2014. Vol. 6. P. 1–8. https://doi.org/10.1115/ICONE22-31163.
22. Yasko A., Babeshko E., Kharchenko V. Verification of FPGA based NPP I&C systems considering multiple faults: Technique and automation tool. *Proc. International Conference on Nuclear Engineering (ICONE 25)*. (2–6 July 2017, Shanghai, China). Shanghai, 2017. Vol. 9. P. 1–9. https://doi.org/10.1115/ICONE25-67065.

23. Kharchenko V., Illiashenko O. Diversity for security: case assessment for FPGA-based safety-critical systems. MATEC Web of Conferences. 2016. Vol. 76. P. 1–9. https://doi.org/10.1051/matecconf/20167602051.

24. Illiashenko O., Kharchenko V., Panarin A., Sklyar V. Hardware diversity for safety of critical instrumentation and control systems. *Proc. 9th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS 2017)*. (21–23 September 2017, Bucharest, Romania). Bucharest, 2017. P. 907–911.

25. IEC 61508:2010. Functional safety of electrical/electronic/programmable electronic safety-related systems. IEC Standards, 2010. 594 p. URL: https://www.iec.ch/functionalsafety/standards/page2.htm.

Надійшла до редакції 27.02.2020

А.А. Летичевский, В.С. Песчаненко, В.С. Харченко, В.А. Волков, О.Н. Одарущенко

МОДЕЛЬНЫЙ СПОСОБ РАЗРАБОТКИ АЛГОРИТМОВ ЦИФРОВЫХ СИСТЕМ НА ПРОГРАММИРУЕМЫХ ЛОГИЧЕСКИХ ИНТЕГРАЛЬНЫХ СХЕМАХ

Аннотация. Рассмотрены современные тенденции в области автоматизированной разработки аппаратного обеспечения, в частности разработки цифровых систем с использованием программируемых логических интегральных схем на примере вентильних матриц, программируемых пользователем. Предложен модельный метод разработки, в котором использована алгебраическая модель спецификаций дизайна, требований и бинарного кода для применения формальных методов верификации, модельного тестирования и методов алгебраического сопоставления. В качестве спецификаций алгебраической модели аппаратного обеспечения служит алгебра поведений, определенная на множестве действий и поведений.

Ключевые слова: программируемые пользователем вентильные матрицы, символьное моделирование, алгебраическое сопоставление, алгебра поведений.

O.O. Letychevskyi, V.S. Peschanenko, V.S. Kharchenko, V.A. Volkov, O.M. Odarushchenko

MODEL-DRIVEN DEVELOPMENT OF DIGITAL SYSTEM ALGORITHMS ON PROGRAMMABLE LOGIC INTEGRATED CIRCUITS

Abstract. The paper considers the current trends in the field of automated hardware development, in particular, the development of digital systems using programmable logic integrated circuits on the example of FPGA (Field-Programmable Gate Array). A model-driven development method is proposed that uses an algebraic model of design specifications, requirements, and binary code to apply formal verification methods, model testing, and algebraic matching methods. The specifications of an algebraic hardware model is a behavior algebra defined over set of actions and behaviors.

Keywords: Field-Programmable Gate Array, symbolic modeling, algebraic matching, behavior algebra.

Летичевський Олександр Олександрович,
доктор фіз.-мат. наук, завідувач відділу Інституту кібернетики ім. В.М. Глушкова НАН України, Київ, e-mail: oleksandr.letychevskyi@litsoft.com.ua.

Песчаненко Володимир Сергійович,
доктор фіз.-мат. наук, професор, завідувач кафедри Херсонського державного університету, e-mail: volodymyr.peschanenko@litsoft.com.ua.

Харченко В'ячеслав Сергійович,
доктор техн. наук, професор, завідувач кафедри Національного аерокосмічного університету ім. М.Є. Жуковського «Харківський авіаційний інститут», e-mail: v.kharchenko@csn.khai.edu.

Волков Владислав Анатолійович,
кандидат фіз.-мат. наук, старший науковий співробітник Інституту кібернетики ім. В.М. Глушкова НАН України, Київ, e-mail: vlad_volkov_98@yahoo.com.

Одарущенко Олег Миколайович,
кандидат техн. наук, доцент, провідний науковий співробітник Науково-виробничого підприємства «Радікс», Кропивницький, e-mail: odarushchenko@gmail.com.