МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

Факультет комп'ютерних наук,
управління та адміністрування

Кафедра інформаційних технологій

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему: Моделі оцінювання та технології верифікації систем діагностування помилок для HDL-проєктів

Виконав: студент 2 курсу групи МІС-20
спеціальності 122 Комп'ютерні науки

Перетятко Дмитро Олександрович

Керівник: професор каф. АСМНСІ, д. т. н.,
доцент Великодний Станіслав Сергійович

Консультант _____

Рецензент: д. ф.-м. н., доцент
Буяджи Василь Володимирович

Одеса 2021

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

Факультет Комп'ютерних наук, управління та адміністрування
Кафедра Інформаційних технологій
Рівень вищої освіти магістр
спеціальність 122 Комп'ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедри _____
_____
"28" жовтня 2021 р.

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Перетятку Дмитру Олександровичу
(прізвище, ім'я, по батькові)

1. Тема роботи Моделі оцінювання та технології верифікації систем діагностування помилок для HDL-проєктів
_____
керівник роботи Великодний Станіслав Сергійович, д. т. н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом закладу вищої освіти від «18» жовтня 2021 р. № 216 «С»

2. Строк подання студентом роботи 09 грудня 2021 р.

3. Вихідні дані до роботи: HDL-модель; стандарт тестування IEEE 1149 та IEEE 1500; високорівневі технології проектування – Electronic System Level Design та моделінга – Transaction Level Modeling; алгебро-логічний метод; граф-схема алгоритму управління; системи убудованого сервісного обслуговування – Infrastructure Intellectual Property; моделі функціональностей – Functional Intellectual Property

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити):
вступ; аналіз технічного завдання; вимоги до програмного забезпечення систем сервісного обслуговування SoC-пам'яті; тестопридатність програмно-апаратних модулів; формування системи сервісної ідентифікації SoC-пам'яті; складові оцінки програмування SoC-пам'яті; інфраструктура процесу верифікації проекту; модель процесу верифікації; аналіз тестопридатності програмних HDL-моделей; верифікація програмних модулів ГАС у пакеті кристалів (SoC); тестопридатність графа керування; верифікація DCT IP-core, Xilinx; висновки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень):
графічний матеріал виконаний у вигляді презентації
_____

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання “28” жовтня 2021 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Термін виконання етапів роботи	Оцінка виконання етапу	
			у %	за 4-х бальною шкалою
1	Отримання завдання на роботу	26.10.2021		
2	Аналіз вихідних даних	30.10.2021		
3	Виконання аналітичного розділу	08.11.2021		
4	Вимоги до програмного забезпечення систем сервісного обслуговування	14.11.2021		
5	Формування системи сервісної ідентифікації SoC-пам'яті	18.11.2021		
6	Рубіжна атестація	22–26.11.2021		
7	Інструкції сервісного обслуговування SoC-пам'яті	28.11.2021		
8	Розробка складових оцінки програмування SoC-пам'яті	02.12.2021		
9	Розробка інфраструктури процесу верифікації проєкту	04.12.2021		
10	Тестування програмних HDL-моделей	05.12.2021		
11	Формування висновків до роботи	06.12.2021		
12	Подання роботи на кафедру	09.12.2021		
13	Перевірка роботи на плагіат	10.12.2021		
14	Складання графічної частини	12.12.2021		
15	Підготовка доповіді на захист	15.12.2021		
16	Проходження рецензування	16.12.2021		
	Інтегральна оцінка виконання етапів календарного плану (як середня по етапам)			

Студент _____ Перетятко Д._О._
(підпис) (прізвище та ініціали)

Керівник роботи _____ Великодний С. С.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

«Моделі оцінювання та технології верифікації систем діагностування помилок для HDL-проєктів». Перетятко Дмитро Олександрович.

Обчислювальна і апаратна складність сучасних гнучких автоматизованих систем, в основу організації яких закладені цифрові системи на кристалах, вимагають створення та впровадження нових високорівневих технологій проєктування, моделінгу і вбудованого сервісного обслуговування. Це означає, що пошук швидкодіючих методів і засобів призводить всіх дослідників до необхідності підвищення рівня абстракції моделей створюваних функціональностей, які вбудовані у кристал.

Мета роботи – моделювання оцінки технологій верифікації цифрових систем для діагностування і виправлення помилок HDL-моделей, шляхом спільного використання механізму асерцій і технологій тестопридатного проєктування. Об'єкт роботи – вбудований у гнучку автоматизовану систему програмний модуль верифікації.

Розглядається технологія верифікації в програмних модулях гнучких автоматизованих систем, орієнтована на істотне підвищення якості спроєктованих компонентів цифрових систем на кристалах і зменшення часу розробки шляхом використання середовища моделювання, тестопридатного аналізу логічної структури HDL-програми для квазиоптимального розміщення механізму асерций. Представлена технологія дозволяє здійснювати пошук помилок із заданою глибиною у програмному HDL-коді за прийнятний для розробника час, шляхом введення в критичні точки програмної моделі асерційної надлишковості, обумовленої за допомогою синтезованих логічних функцій тестопридатності.

Магістерська кваліфікаційна робота містить: 73 с.; 19 рис., 24 використаних літературних джерела.

КЛЮЧОВІ СЛОВА: пакет кристалів, програмний модуль, верифікація, математична модель, інфраструктура процесу, маршрут, граф, інструкція, ідентифікація, асерция, масив.

SUMMARY

«Evaluation models and verification technologies of error diagnosis systems for HDL projects». Peretyatko Dmitry.

Computational and hardware complexity of modern flexible automated systems, based on the organization of which are based on digital systems on crystals, require the creation and implementation of new high-level technologies of design, modeling and embedded service. This means that the search for fast-acting methods and tools leads all researchers to the need to increase the level of abstraction of models of the created functionalities, which are built into the crystal.

The purpose of the work is to model the evaluation of digital systems verification technologies for diagnosing and correcting errors of HDL-models, by sharing the mechanism of assertions and technologies of test-appropriate design. The object of work is a software verification module built into a flexible automated system.

Verification technology in software modules of flexible automated systems is considered, focused on significant improvement of quality of designed components of digital systems on crystals and reduction of development time by using modeling environment, test-appropriate analysis of logical structure of HDL-program for quasi-optimal placement of assertion mechanism.

The presented technology allows to search for errors with a given depth in the software HDL-code in a time acceptable to the developer, by introducing to the critical points of the software model assertion redundancy due to the synthesized logical functions of dough.

Master's qualification work contains: 73 pages; 19 fig., 24 used literature sources.

KEY WORDS: crystal package, software module, verification, mathematical model, process infrastructure, route, graph, instruction, identification, assertion, array.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАК

АЛМ – алгебро-логічний метод.

Асерція – є вислів системного рівня, що визначає коректність перетворень в процесі проєктування щодо вхідного опису поточного етапу або вимог специфікації.

Валідація – є процес визначення працездатності системи і її компонентів, шляхом перевірки відповідності основним вимогам специфікації після виконання кожної стадії проєктування.

ВЕП – вектор експериментальної перевірки.

Верифікація – є процес аналізу системи або компонентів для визначення коректності перетворень вхідного опису на черговий стадії проєктування.

ГАС – гнучка автоматизована система.

ГСАУ – граф-схема алгоритму управління.

ДНФ – діз'юнктівная нормальна форма.

ЗГУ – змістовний граф управління.

ІС – інтегральна схема.

КНФ – кон’юнктива нормальна форма.

ПЗП – постійний запам’ятовуючий пристрій.

Сертифікація – є письмова гарантія того, що система або її компоненти повністю задовольняють вимогам специфікації і прийнятні для використання за призначенням.

ТН – таблиця несправностей.

ADTG – тестовий генератор, призначений для перевірки суматорних схем.

ANSI – American National Standards Institute.

ASIC – Application Specific Integrated Circuits.

ATPG – Automated Test Pattern Generator.

BIRA – Built-In Repair Analysis.

BISR – Built-In Self Repair.

BIST – Built-In Self Test.

BSTG – тестовий генератор для шинних структур прийому-передачі даних.

COM – communication port.

CTL – Core Test Language.

DFM – Design-for-Manufacturing.

DFTG – синтезатор тестів для автоматів, що задані в вигляді графа-схем алгоритмів.

DRAM – Dynamic Random-Access Memory.

EEPROM – Electrically Erasable Programmable Read-Only Memory.

EPROM – Erasable Programmable Read-Only Memory.

ESL – Electronic System Level.

FDT – Fault Detection Table.

F-IP – Functional Intellectual Property.

GUI – Graphic User Interface.

HDL – Hardware Description Language (мова опису апаратури інтегральних схем).

IEEE – Institute of Electrical and Electronics Engineers.

I-IP – Infrastructure Intellectual Property.

ISO – International Organization for Standardization.

KGD – Known-Good-Die.

METG – генератор тестів, який орієнтований на перевірку матричної пам’яті.

NVRAM – Non-Volatile Random-Access Memory.

PROM – Programmable Read-Only Memory.

PRTG – псевдовипадковий генератор вхідних стимулів з рівномірним законом розподілу нульових і одиничних сигналів по вхідним змінним..

RCTG – тестовий генератор для послідовних лічильно-реєстрових структур і тригерних схем.

RFID – Radio-Frequency Identification.

RPROM – Re-programmable Read-Only Memory.

SATG – тестовий генератор шістнадцятирічних кодів на основі сигнатурного аналізу.

SDL – Simple DirectMedia Layer.

SECT – Standard for Embedded Core Test.

SiP – System in Package.

SLI – System Level Integration.

SoC – System-on-Chip.

SPTG – алгоритмічний генератор тестових векторів, які активізують одномірні логічні шляхи, орієнтовані на перевірку заданих одиночних несправностей.

TAM – Test Access Mechanism.

TG – Transaction Graph.

TLM – Transaction Level Modeling.

UML – Unified Modeling Language.

Wi-Fi – Wireless Network Protocol.

WLBI – Wafer-Level Burn-In.

yield – вихід готової продукції.

ВСТУП

У своєму безперервному розвитку ринок мікроелектроніки постійно висуває все нові і більш жорсткі вимоги до виробів, що з’являються. Споживач хоче отримувати швидкодіючу, надійну і, в той же час, малогабаритну продукцію, що споживає малу потужність. Дві ці суперечливі вимоги посилюються тим, що мікроелектронні покоління дуже швидко старіють, час морального зносу обчислюється іноді місяцями, тому особлива увага приділяється постійному скороченню часу виходу на ринок нових виробів. Терміни, що відводяться на розробку, проєктування, верифікацію і випуск в серію нових інтегральних схем (ІС), прагнуть скорочувати усіма силами, не забуваючи при цьому пред'являти підвищені вимоги до якості самих ІС і їх надійності.

Одним із способів вирішення даного протиріччя стало створення замовних ІС з великим числом елементів і зі складною внутрішньою структурою, від яких були потрібні можливість гнучкої спеціалізації "під задачу" і найкоротший час виходу на ринок. Такі замовні мікросхеми класу Application Specific Integrated Circuits (ASIC) набули широкого поширення у всьому світі, оскільки це було єдиним прийнятним рішенням при реалізації складних виробів мікроелектроніки для портативної і переносної апаратури.

Основною перевагою замовних ІС є низька вартість кінцевого масового продукту, тому, з постійним вдосконаленням технологічного циклу виробництва мікросхем знижуються і вимоги до мінімальних замовлень ASIC. Стає вигідно замовляти "свої" мікросхеми навіть для середніх обсягів виробництва, отримуючи основний прибуток після реалізації кінцевої продукції. При цьому замовник є власником як кінцевого продукту, так і закладеної в нього ідеї, і, отже, несе на собі всю тяжкість і відповідальність прийняття рішення.

На жаль, проєкти на ASIC мають свої недоліки: високий рівень початкових неповернутих витрат, тривалий час розробки і верифікації, а також значні кількості для мінімального замовлення партії готових мікросхем. Як

результат, замовні ІС доступні тільки для кінцевих виробів за умови їх тиражу і тривалого терміну їх активного використання. Вимоги щодо мінімального обсягу замовлення мікросхем ASIC часто перевищують $500 тис. В розрахунку на проєкт і на рік. Проєкти ж з коротким часом їх "життя" (до морального старіння), малих або середніх обсягів тиражности, які вимагають якнайшвидшого виходу на ринок або частого поновлення реалізованих стандартів або алгоритмів швидше за все не можуть собі дозволити бути реалізованими у вигляді ASIC. Причому навіть у разі, коли критерій "обсяг / ціна" є для даної розробки прийнятним, будь-яка зміна для виправлення допущеної помилки або для її вдосконалення залишить замовника з великими складськими запасами, можливо, нікому не потрібних мікросхем і запустить заново весь тривалий цикл (не менше 4 – 6 місяців) створення нової версії ASIC.

Дана проблема особливо актуальна для швидко еволюціонують сегментів промисловості, таких як гнучкі автоматизовані системи (ГАС) ринку. Очевидно, що тут більш кращі програмовані, що конфігуруються рішення в реалізації ІС пам'яті, які можуть бути змінені як на стадії розробки, так і в стадії сервісного обслуговування.

Обчислювальна і апаратна складність сучасних ГАС, в основу організації яких закладені цифрові системи на кристалах (System-on-Chip – SoC), що характеризуються мільйонами еквівалентних вентилів і вимагають створення та впровадження нових високорівневих технологій проєктування – Electronic System Level (ESL) Design, моделинга – Transaction Level Modeling (TLM) і вбудованого сервісного обслуговування – Infrastructure Intellectual Property (I-IP). Це означає, що пошук швидкодіючих методів і засобів призводить всіх дослідників до необхідності підвищення рівня абстракції моделей створюваних функциональностей – Functional Intellectual Property (F-IP), вбудованих в кристал.

Ринок програмних продуктів EDA вже пропонує інструменти для автоматизації процесів моделинга і верифікації пристроїв системного рівня, починаючи з компіляторів HDL-мов (C++, SystemC, SystemVerilog, UML, SDL)

і закінчуючи графічними оболонками (Simulink, LabView, Xilinx EDK). Дані засоби дозволяють створювати проєкти з існуючих бібліотечних компонентів шляхом використання ESL-меппінга і створення TLM-інтерфейсів.

Ринкова привабливість імплементації цифрової системи в кристал FPGA визначається застосуванням порівняно дешевих чипів замість універсальних процесорів, малою споживаною потужністю, габаритними розмірами, якісним і надійним виконанням основних функцій, завдяки вбудованій I-IP-інфраструктурі, що є актуальним в століття мобільних обчислювальних пристроїв.

Таким чином, метою магістерської кваліфікаційної роботи – є моделювання оцінки технологій верифікації цифрових систем для діагностування і виправлення помилок HDL-моделей, шляхом спільного використання механізму асерцій і технологій тестопридатного проєктування.

Об'єктом роботи – є вбудований в гнучку автоматизовану систему програмний модуль верифікації.

Завдання дослідження:

- класифікація технологій тестопридатного проєктування HDL-моделей для створення цифрових систем на кристалах;
- розробка моделей верифікації і тестування системної HDL-моделі на основі використання асерцій;
- розробка метрики оцінювання тестопридатності HDL-моделей на основі нової логічної функції тестопридатності;
- застосування технологічної моделі асерцій для верифікації IP-core фільтра на основі дискретного косинусного перетворення.

1 АНАЛІЗ ТЕХНІЧНОГО ЗАВДАННЯ ТА ВИМОГ ДО СЕРВІСНОГО ОБСЛУГОВУВАННЯ

1.1 Сучасні тенденції розвитку систем на кристалі

Вираз «система на кристалі» не є, суворо кажучи, терміном. Це поняття відображає загальну тенденцію до підвищення рівня інтеграції за рахунок інтеграції функцій.

Під приладами класу «система на кристалі», в загальному випадку, розуміються прилади на єдиному кристалі яких інтегровані процесор (процесори, у т. ч. спеціалізовані), деякий обсяг пам'яті, низка периферійних пристроїв та інтерфейсів, тобто максимум того, що необхідно для вирішення завдань, поставлених перед системою.

Продуктивність приладів класу «система на кристалі», в значній мірі, залежить від ефективності взаємодії всіх встановлених компонентів і від ефективності їх взаємодії із зовнішнім, щодо приладу, світом. В першу чергу це пов'язано з різницею у швидкодії вбудованих компонентів, особливо організації інтерфейсів.

На даний час, значна частина подібних проєктів розробляється у вигляді друкованої плати як комбінації мікросхем програмованої і жорсткої логіки, аналогових блоків, мікроконтролерів, масивів пам'яті і фіксованих периферійних вузлів (інтерфейс T1, ATM, 10/100 PHY, відео / аудіо кодеки тощо). Незважаючи на те, що такі комбіновані рішення дозволяють досить швидко створювати різноманітні швидко мінливі проєкти, вони не можуть реально конкурувати з точки зору продуктивності, енергоспоживання, надійності і масогабаритних характеристик з монолітним рішенням – інтегральної мікросхемою системного рівня інтеграції.

Таким чином, з'явилися всі передумови до реального створення комерційних версій ІС нового покоління, що поєднують в собі переваги традиційних замовлених виробів класу ASIC, мікросхем програмованої логіки і інтег-

руючих широкий діапазон системних ресурсів для більшої функціональності. Нові мікросхеми було віднесено до групи виробів системного рівня інтеграції SLI (System Level Integration), і до теперішнього часу рівень SLI був реалізований лише в замовних мікросхемах з фіксованою архітектурою тому, що це було єдиним прийнятним технологічним рішенням.

Під інтеграцією різних системних ресурсів тут не слід розуміти механістичне об'єднання окремих систем, якими можуть бути (нехай і як завгодно складні, але в той же час типові, стандартні) мікропроцесори, блоки пам'яті і периферійні вузли. Можливість поєднання різних типів електронних осередків на площі одного кремнієвого кристала вивільняє нові споживчі якості щодо мікросхем, дозволяє цілеспрямовано орієнтувати нову продукцію на необхідні сегменти ринку, забезпечуючи виробникам сучасної електронної апаратури технічну і економічну вигоду.

Реалізована на практиці ідея SLI виявилася настільки багатою, що відразу ж знайшла широкий відгук у багатьох світових лідерів із виробництва мікроелектронних виробів. Інтеграція всіх основних системних вузлів на одній системно-орієнтованої мікросхемі забезпечує підвищення продуктивності, зниження енергоспоживання, зменшення ціни кінцевого виробу в цілому і дозволяє випускати малогабаритну продукцію. Всі ці переваги особливо важливі в області ГАС, з точки зору телекомунікаційних додатків портативній апаратурі їх сервісного обслуговування, а також в мережевих додатках. Вироби нового покоління, що виконуються по ідеології SLI, стали називати «система на кристалі» – System on a Chip або SoC. І основною перешкодою на шляху активного впровадження мікросхем SoC у масове виробництво аж до початку 2000-х років, були лише технологічні обмеження напівпровідникової промисловості.

Революційні зміни в технології виробництва мікроелектронних виробів дали можливість комбінувати на одному кремнієвому кристалі кілька різнорідних типів електронних осередків: CMOS + Flash, CMOS + EEPROM, SiGe / BiCMOS. Були випущені перші інтегральні замовні мікросхеми ASIC, що

реалізують як цифрову, так і аналогову обробку даних, в тому числі і для радіочастотного діапазону.

Удосконалення технологічного процесу дозволило постійно збільшувати кількість інтегрованих транзисторів у межах однієї і тієї ж площі кремнієвого кристала, проте виявилося неможливим повноцінно використовувати всі переваги цього збільшення без значного подовження часового циклу розробки проєктів, особливо у зв'язку із постійно зростаючою складністю останніх.

1.2 Тестопридатність програмно-апаратних модулів

Ядром, що ініціювало появу нових технологій тестування та верифікації у програмної та комп'ютерної інженерії слід вважати силіконовий кристал, який є основою для створення обчислювальних і / або комунікаційних пристроїв. Кристал розглядається як випробувальний полігон для апробації нових засобів і методів трасування, розміщення, синтезу та аналізу компонентів. Технологічні рішення, які витримали випробування часом в мікроелектроніці, далі захоплюються і адаптуються у макроелектроніці, яка представлена комп'ютерними системами і мережами. Ось деякі історичні факти, пов'язані з наступністю розвитку технологічних інновацій.

1. Стандарти граничного сканування [1] [1)] на рівні плати і кристала привели до появи механізму асерцій для тестування і верифікації програмних продуктів.

2. Засоби аналізу тестопридатності [2], [3] [2)] (спостереження та управління) цифрових структур можна адаптувати для оцінки програмного

[1)] [1] Хаханов В. И., Хаханова А. В., Литвинова Е. И. Алгебро-логический метод ремонта встроенной памяти SoC. *Відмовостійкі системи*. 2018. С. 99–109.

[2)] [2] Проєктирование и диагностика компьютерных систем и сетей. Бондаренко М. Ф., Кривуля Г. Ф., Рябцев В. Г. Киев: НМЦ ВО, 2010. 306 с.

[3] Хаханов В. И., Хаханова И. В. VHDL + Verilog = Синтез за минуты. Харьков: СМИТ, 2017. 264 с.

коду з метою визначення критичних місць і подальшого поліпшення програмного продукту щодо критерію тестованості.

3. Технології аналізу якості покриття [4] [1] наперед заданих несправностей тестовими блоками слід використовувати для створення таблиці покриття дефектів програмних продуктів з метою оцінки достовірності тестів і встановлення діагнозу.

4. Графові моделі регістрових передач Тетта-Абрахама [5] [2] та С. Г. Шаршунова [6], [7] [3] використовуються при тестуванні програмних продуктів, які шляхом структурно-логічного аналізу наводяться до більш технологічною формі. Тут інтерес представляє удосконалення графа до транзакционной моделі HDL-коду, що записується у вигляді алгебраїчної форми подання графа для підрахунку тестопридатності з метою визначення критичних компонентів (точок) програми для установки асерцій.

5. Поділ автомата на керуючу, операційну структурну та поведінкову частини [8], [9] [4] також застосовується для спрощення процесу верифікації програмного коду на основі попереднього синтезу графів управління і передачі даних.

---

[1] [4] Парфентий А. Н., Хаханов В. И., Литвинова Е. И. Модели инфраструктуры сервисного обслуживания цифровых систем на кристаллах. *АСУ и приборы автоматики*. 2017. Вып. 138. С. 83–99.

[2] [5] Золотухо Р. System Designer – пакет для разработки устройств на основе FPSLIC. *Chip News*. 2011. №2. С. 8–14.

[3] [6] Золотухо Р., Кривченко И. Конфигурируемая система на кристалле E5 – первое знакомство. *Компоненты и технологии.* 2011. №1. С. 26–29.

[7] Программируемые приборы класса "система-на-кристалле" для встраиваемых применений. *Компоненты и технологии.* 2001. №2. С. 13.

[4] [8] Великодний С. С., Бурлаченко Ж. В., Зайцева-Великодна С. С. Реінжиніринг графічних баз даних у середовищі відкритої системи автоматизованого проєктування BRL-CAD. Моделювання структурної частини. *Вісник Кременчуцького національного університету ім. Михайла Остроградського.* 2019. Вип. 3 (116). С. 130–139. (кат. «Б») DOI: 10.30929/1995-0519.2019.3.130-139.

[9] Великодний С. С., Бурлаченко Ж. В., Зайцева-Великодна С. С. Реінжиніринг графічних баз даних у середовищі відкритої системи автоматизованого проєктування BRL-CAD. Моделювання поведінкової частини. *Вісник Кременчуцького національного університету ім. Михайла Остроградського.* 2019. Вип. 2 (115). С. 117–126. (кат. «Б») DOI: 10.30929/1995-0519.2019.2.117-126.

6. Крива життєвого циклу апаратного виробу [10] [1)] відображає також тимчасові стадії зміни «yield» при створенні, тиражуванні і супроводі програмного продукту.

7. Платформно-орієнтовний синтез [11] [2)] HDL-коду (platform-based electronic system-level design) із використанням існуючих наборів компонентів (chip set) під керуванням GUI – Graphic User Interface, ізоморфний технології об'єктно-орієнтованого програмування на основі використання напрацьованих бібліотек. Застосування технології Electronic System Level (ESL) в програмуванні дає можливість використовувати програмні компоненти готових функціональностей з базових бібліотек, розроблених для створення нових програмних продуктів [12] [3)]. У цьому випадку основна процедура проєктування полягає у виконанні меппінга, який орієнтований на покриття функцій специфікації існуючими компонентами, де новий код становить не більше 10% проєкту [13] [4)].

8. Поняття «testbench» [14] [5)], що використовується для тестування і верифікації апаратних проєктів за допомогою HDL-компіляторів, з'являється і в програмних продуктах, реалізованих на рівні мов C++ і вище.

9. Платформно-орієнтовний орієнтований синтез “testbench” (platform-based testbench synthesis) з використанням існуючих бібліотек тестів (ALINT) для компонентів – стандартизованих функциональностей F-IP SoC під управ-

---

[1)] [10] Великодный С. С. Методологические основы реинжиниринга систем автоматизированного проєктирования. *Управляющие системы и машины.* 2014. № 2. С. 39–43. Видання національної академії наук України.

[2)] [11] Хаханов В. И. Инфраструктура сервисного обслуживания SoC. *Вестник томского государственного университета.* №4. 2008. С. 74–101.

[3)] [12] Великодный С. С. Проблема реинжиниринга видов обеспечения систем автоматизированного проєктирования. *Управляющие системы и машины.* 2014. № 1. С. 57–61, 76. Видання національної академії наук України.

[4)] [13] Великодний С. С. Реінжиніринг систем моніторингу та дистанційного управління судновими енергетичними установками. Матер. XXII міжн. конф. з автом. управл. «Автоматика 2015», 10–11 вер. 2015, Одеса, 2015. С. 133–134.

[5)] [14] Hahanov V., Kteaman H., Ghribi W., Fomina E. HEDEFS – Hardware embedded deductive fault simulation. Proc. volume from the 3-rd IFAC Workshop, Rydzyna, Poland, 2016. P. 25–29.

лінням інтерфейсу GUI слід використовувати для генерування тестів програмних модулів на основі напрацьованих бібліотек провідних компаній.

10. Стандартні рішення сервісного обслуговування F-IP в рамках I-IP можна використовувати для вбудованого тестування компонентів програмної системи, в тому числі і для відновлення працездатності дефектного програмного модуля.

11. Забезпечення планарності (двовимірності) в структурі взаємопов'язаних функціональних компонентів (IP-cores) створюваного програмного продукту орієнтоване на використання мультиядерних архітектур [14] [1)] для технологічного розпаралелювання обчислювальних процесів, що в умовах науково-технічної революції, запропонованої компанією Intel, є вельми актуальним.

12. Створення адресного простору для функциональностей SoC, реалізованих як в апаратному, так і в програмному виконанні [15] – [17] [2)], надає цифровий системі чудову властивість самовідновлення працездатності програмних і апаратних компонентів засобами I-IP. Прикладом тому може служити мультипроцесорні виконання апаратних продуктів, яке є стійким до виникаючих дефектів. При цьому адресний компонент, що відмовився, може бути замінений іншим, справним, в процесі виконання функціональності [18], [19] [3)]. Властивість адресованості слід використовувати при створенні критичного програмного продукту, в якому наявність адресованих діверсних (му-

---

[1)] [14] Hahanov V., Kteaman H., Ghribi W., Fomina E. HEDEFS – Hardware embedded deductive fault simulation. Proc. volume from the 3-rd IFAC Workshop, Rydzyna, Poland, 2006. P. 25–29.

[2)] [15] Федотов Я., Щука А. Система на кристалле. *Электронные компоненты*. 2011. №2. С. 3–5.

[16] Mixed-Signal ASIC Solutions. AMI Press, 2010. 86 p.

[17] System-on-Chip. micron AG Press, 2010. 104 p.

[3)] [18] Analog and Mixed-Signal ASICs. PREMA Semiconductor Press, 2010. 208 p.

[19] ASIC: Парад технологий. Переклад. О. Александрова. *Chip News.* 2012. №9. С. 8–11.

льтіверсних) компонентів надає програмної системі здатність заміняти один одного при виникненні несправностей [20], [21] [1)].

13. Дуже цікавою і ринково привабливою представляється технологічна проблема автономного внутришньо кристального самотестування, самодіагностування і самовідновлення із застосуванням зовнішніх засобів (і без них) [9-11], якої сьогодні займаються всі провідні компанії. Для вирішення даної проблеми залучаються модні сучасні бездротові Internet-технології дистанційного сервісного обслуговування.

Зворотний бік медалі полягає в несанкціонованому доступі до вмісту кристала на відстані, що може привести до небажаних деструктивних наслідків і виведення з ладу цифрового виробу. Проте, специфіка цифрових систем на кристалах полягає в дивовижній можливості виправляти помилки на відстані, завдяки наявності зв'язку кристала із зовнішнім світом за допомогою Internet або бездротових технологій (Wi-Fi, Wi-Max, bluetooth), присутніх в кремнії. Дистанційна корекція програмних помилок стає можливою завдяки використанню пам'яті (що займає до 94% кремнію) SoC для зберігання програм, куди можна, в разі виявлення некоректності, записати новий код, який не має помилок. Дистанційна корекція апаратних помилок стала можливою завдяки використанню програмованих логічних інтегральних схем, куди можна, в разі виявлення несправності, записати новий bit stream, що не має помилок, який фактично створює нову апаратуру шляхом повторного програмування кристала.

Зближення і взаємопроникнення технологій призводить до ізоморфних методів проєктування, тестування і верифікації по відношенню до програмних і апаратних комплексів, що по суті є закономірним процесом асиміляції прогресивних концепцій. Тому сприяє факт, що найбільш важливі параметри

[1)] [20] Кривченко И. Системная интеграция в микроэлектронике FPSLIC. *Chip News*. 2012. №3. С. 4 –10.

[21] Кривченко И. Системная интеграция в микроэлектронике FPSLIC. Часть 2: FPSLIC – вопросы и ответы. *Chip News*. 2012. №4. С. 62–64.

життєвого циклу виробу, такі як «time-to-market» і «yield» стають порівнянними за часом і виходу придатної продукції. Крива життєвого циклу апаратного виробу (HDL-проєкту), представлена на рис. 1, з точністю до ізоморфізму відображає етапи програмного продукту, який проходить аналогічні стадії: проєктування, збільшення обсягу випуску і виробництво з доопрацюванням і супроводом виробу.

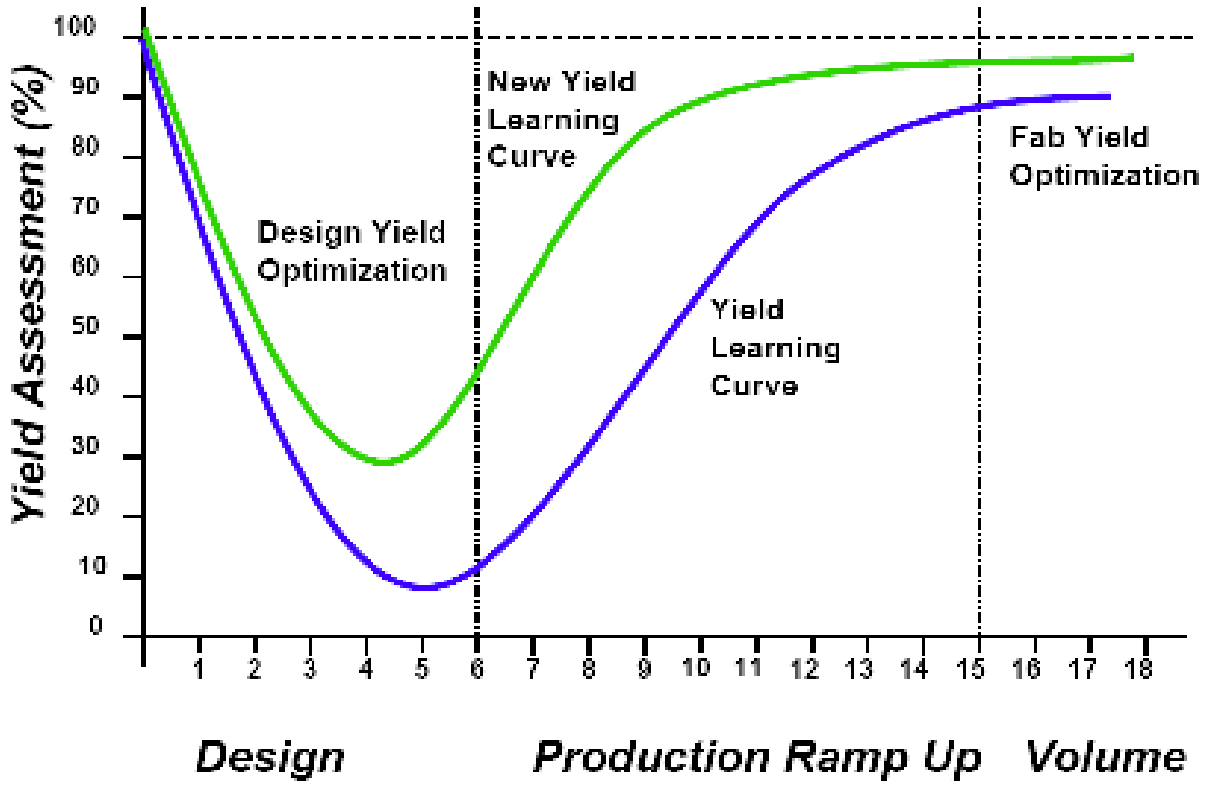


Рисунок 1 – Графік життєвого циклу HDL-проєкту

В контексті життєвого циклу існують дві актуальні проблеми, які пов'язані з підняттям графіка (кривої) вгору по осі ординат, а також з компресією згаданої кривої з тимчасової осі, що означає зменшення параметра «time-to-market». Тут підвищення «yield» відбувається на всіх стадіях:

– design – за рахунок усунення помилок розробки;
– production ramp up – за рахунок виправлення коду, імплементувати в пам'ять системи на кристалі;

– volume – за рахунок випуску service pack, що коригує помилки шляхом поширення через Internet або супутники (satellites).

Згідно [22] [1)] вартість верифікації програмно-апаратних продуктів на основі ASIC, IP-core, SoC становить 70% від загальних витрат проєктування. Аналогічна оцінка, близько 80%, визначає розмірність testbench-коду від загальної довжини формального опису проєкту. Завдання, які вирішуються в процесі верифікації, пов'язані з усуненням помилок проєктування якомога на більш ранній його стадії з метою приведення коду прототипу у відповідність з його специфікацією. Пропуск помилки збільшує її вартість на порядок при переході від рівня проєктування блоку до рівня кристала і далі до системи.

Уведення в проєкт програмної надмірності – механізму асерцій [23] [2)] – дозволяє виконувати аналіз основних специфікованих умов в процесі моделювання проєкту і діагностувати помилки в разі їх виявлення на ранніх стадіях проєктування програми (C++) або апаратури (HDL).

1.3 Вимоги до програмного забезпечення систем сервісного обслуговування SoC-пам’яті

Зважаючи на велику кількість фірм-виробників програмного забезпечення для SoC-мікросхем, в даному розділі магістерської кваліфікаційної роботи, дамо загальну характеристику вимогам, задовольняти яким повинні програмні продукти для систем сервісного обслуговування вбудованої пам'яті.

По-перше, програмне середовище повинно включати повний цикл розробки, починаючи від введення алгоритму, включаючи процес налагодження та закінчуючи програмуванням кристала. Розробка програми може бути як на

[1)] [22] Королев Н. ATMEL FPSLIC – элементная база XXI века. *Chip News*. 2011. №1. С. 16–19.

[2)] [23] Ajay Khoche. System-in-Package is Coming to Consumer Products: Is Test Ready? Proceedings of the International Test Conference 2017 (ITC’17). 2017. pp. 1166.

рівні асемблера, так і на макрорівні з маніпуляцією багатобайтовими величинами зі знаком.

По-друге, на відміну від програм на класичному асемблері, програма повинна вводитися у вигляді алгоритму з деревоподібними розгалуженнями та відображатися на площині, в двох вимірах. Мережа умовних і безумовних переходів повинна відображатися графічно, в зручній векторній формі. Це до того ж звільняє програму від незліченних імен міток, які в класичному асемблері є неминучим баластом. Вся логічна структура програми повинна бути наочною.

По-третє, повинні бути враховані можливості графічних технологій, які розкривають нові можливості для програмістів. Візуальність логічної структури зменшує ймовірність помилок і скорочує терміни розробки. З'являється таке поняття, як дизайн алгоритму, що припускає певний художній смак програміста, в рамках поняття технічної естетики.

По-четверте, з огляду на наростаючий дефіцит часу, як атрибут технологічного розвитку, в порівнянні з класичним асемблером, час на розробку програмного забезпечення має бути скорочено в 3 – 5 разів.

По-п'яте, повинна підтримуватися автоматична перекодування рядків ANSI-кодів Windows в коди русифікованого літерно-цифрового РКІ.

По-шосте, середовище має об'єднати в собі графічний редактор, компілятор алгоритму, симулятор мікроконтролера, внутрісхемний програматор. При використанні внутрисхемного програматора, мікроконтролер може підключатися до COM-порту комп'ютера через нескладний адаптер, наприклад: три діода і кілька резисторів. Програматор веде підрахунок числа перепрограмування кристала, зберігаючи лічильник безпосередньо у кристалі.

По-сьоме, програмне середовище повинно забезпечувати моніторне налагодження на кристалі (On Chip debug), яка дозволяє спостерігати вміст реального кристала в заданій точці зупинки. При цьому для зв'язку мікроконтролера з комп'ютером, може використовуватися тільки один вивід, причому

за вибором користувача. Мониторне налагодження може бути застосовано до будь-якого типу кристала, що має SRAM.

І нарешті, програмне забезпечення повинно підтримувати найбільш широку лінійку типів кристалів і призначатися для роботи у всіх операційних системах.

2 КОМПОНЕНТИ ІНФРАСТРУКТУРИ СЕРВІСНОГО ОБСЛУГОВУВАННЯ SOC-ПАМ'ЯТІ ГАС

2.1 Структура сервісів SoC-мікросхем

На рис. 2 представлена усічена структура, орієнтована на виконання наступних завдань:

– тестування функціональності на основі вхідних послідовностей, що генеруються (Automated Test Pattern Generator – ATPG) і аналізу вихідних реакцій;

– моделювання (Fault Simulator) несправностей з метою забезпечення діагностування та ремонту на основі таблиці несправностей (Fault Detection Table – FDT);

– діагностування дефектів із заданою глибиною, шляхом використання мультізонда стандарту IEEE 1500;

– вбудований ремонт матричної пам'яті, на основі використання запасних компонентів (spare).

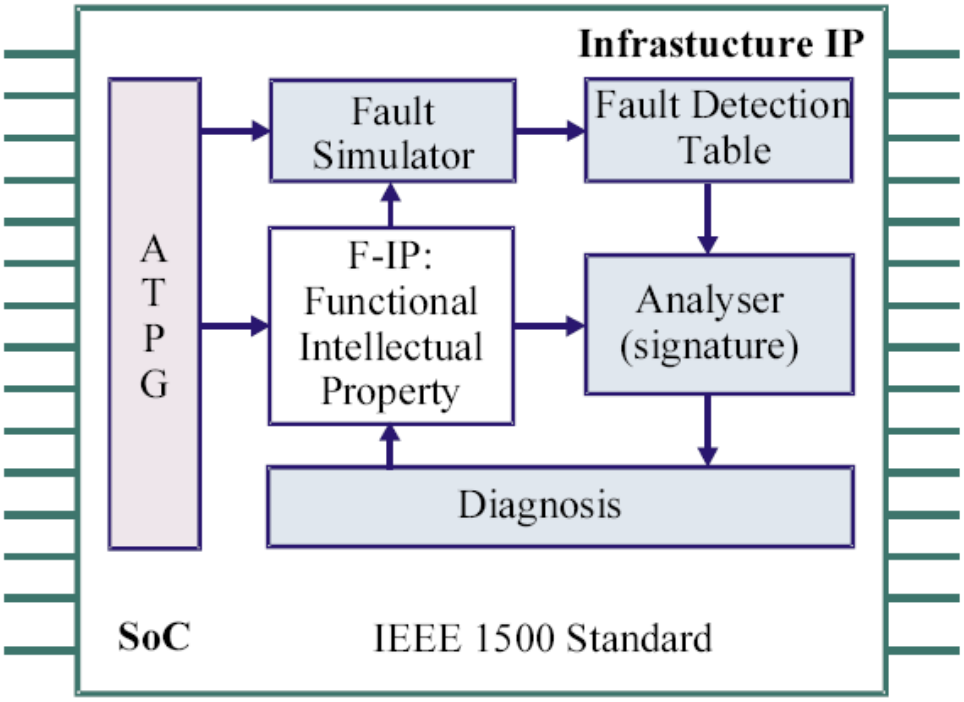


Рисунок 2 – Інфраструктура сервісів SoC DSP

2.1.1 Модуль синтезу тестів

Модуль синтезу тестів, призначений для перевірки функціональності і одиночних несправностей. У його склад входить набір генераторів вхідних послідовностей, які забезпечують створення наступних тестів:

- PRTG – псевдовипадковий генератор вхідних стимулів з рівномірним законом розподілу нульових і одиничних сигналів по вхідним змінним;
- SATG – тестовий генератор шістнадцятирічних кодів на основі сигнатурного аналізу;
- SPTG – алгоритмічний генератор тестових векторів, які активізують одномірні логічні шляхи, орієнтовані на перевірку заданих одиночних несправностей;
- ADTG – тестовий генератор, призначений для перевірки суматорних схем АЛУ;
- BSTG – тестовий генератор для шинних структур прийому і передачі даних;
- METG – генератор тестів, орієнтований на перевірку матричної пам'яті;
- DFTG – синтезатор тестів для автоматів, заданих у вигляді граф-схем алгоритмів;
- RCTG – тестовий генератор для послідовних лічильно-реєстрових структур і тригерних схем.

Модуль-генератор аналізує структурно-функціональну модель блоку, що підлягає тестуванню, і призначає підмножина таких синтезаторів, які забезпечують задану якість покриття дефектів (F_c) і функціональних режимів (P_c):

$$F^C\left(\bigcup_{i=1}^{n_{min}} T_i\right) \geq F^C_{min};\ \ P^C\left(\bigcup_{i=1}^{n_{min}} T_i\right) \geq P^C_{min},$$

$$T = \{T_1^{PR}, T_2^{SA}, T_3^{SP}, T_4^{AD}, T_5^{BS}, T_6^{ME}, T_7^{DF}, T_8^{RC}\}. \quad (1)$$

Узагальнена структура синтезу Testbench, представлена на рис. 3, включає також генератор HDL-коду, який призначений для тестування і верифікації функціональностей на стадії розробки проєкту.

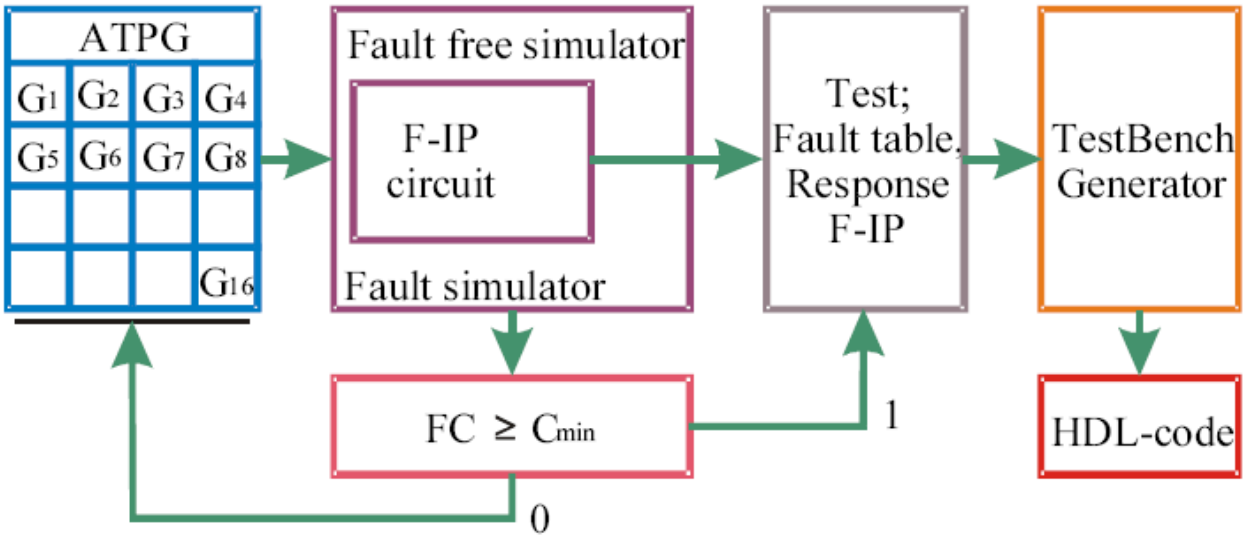


Рисунок 3 – Структура процесу синтезу Testbench для F-IP

Кількість тестових генераторів на стадії проєктування SoC може бути істотно більшою, ніж та підмножина, яка далі вбудовується в кристал. Тому на стадії моделювання і верифікації виконується аналіз властивостей, що покривають, кожного тест-генератора з метою пошуку їх мінімальної сукупної конфігурації, яка задовольняє виразу (1).

Важливо відзначити, що в найближчі 5 років ідеологія синтезу тестів для цифрових систем на кристалах буде запозичувати кращі традиції, що йдуть від ESL- і TLM-проєктування:

– використання бібліотек тестів (Testbench) провідних компаній для тестування і верифікації стандартизованих функціональностей, позначених як F-IP;
– застосування стандартних рішень сервісного обслуговування I-IP для вбудованого тестування компонентів системи на кристалі;
– створення власних бібліотек тестів для тих функціональностей, що знову розроблюються;
– впровадження нової технології синтезу тестів для цифрової системи, заснованої на дискретному меппінгу покриття функціональностей і дефектів вихідної специфікації за допомогою мінімальної сукупності Testbench, з бібліотеки тестів (рис. 4);
– застосування вбудованих засобів тестопридатності, таких як IEEE boundary scan – засоби граничного сканування, і шести компонентів I-IP для підвищення технологічності процедур синтезу тестів.

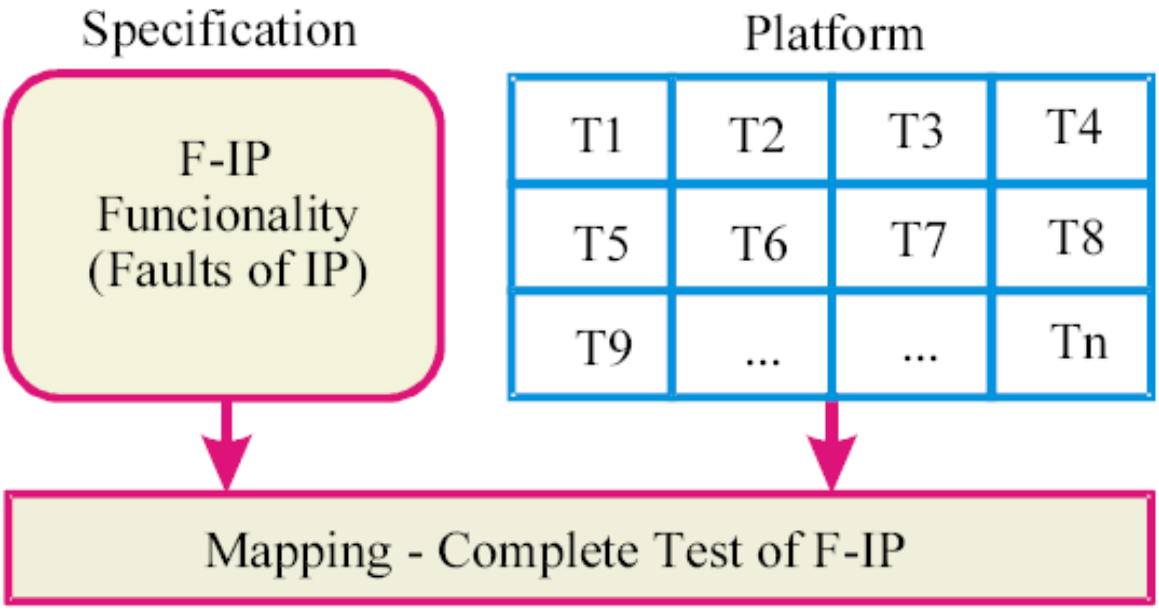


Рисунок 4 – Mapping моделі синтезу тестів для F-IP

2.1.2 Модуль аналізу несправностей

Модуль аналізу несправностей використовує дедуктивний алгоритм, орієнтований на перевірку одиночних дефектів, генерованих на основі аналітичного або табличного описів функціональностей SoC. Це означає, що дедуктивне моделювання може обробляти проєкти, що представлені як на вентильному, так і на будь-якому іншому, більш високому рівні абстракції (регістровому, системному).

Основна ідея методу полягає у створенні дедуктивної моделі функціональності на основі використання відомого виразу:

$$F = f\left[(X_1 \oplus T_1), (X_2 \oplus T_2), \dots, \left(X_j \oplus T_j\right), \dots, \left(X_{n_i} \oplus T_{n_i}\right)\right] \oplus T_i,$$

де дедуктивна функція F на тест-векторі T є модифікований опис справної поведінки, що дозволяє обчислювати списки вхідних несправностей, які транспортуються на вихід схеми під впливом вхідних сигналів. На прикладі функції *Xor* демонструється синтез дедуктивної функції по карті Карно:

$F =$

$(xy)\backslash(ab)$	00	01	11	10
00	0	1	0	1
01	0	1	0	1
11	0	1	0	1
10	0	1	0	1

Змінні *xy* – булеві, а сигнали *ab* – реєстрові для запису списків дефектів:

$$L = f(x, y, a, b) = \bar{a}b \vee a\bar{b}. \quad (2)$$

Апаратна реалізація дедуктивної функції, що представлена формулою (2), зображена на рис. 5.

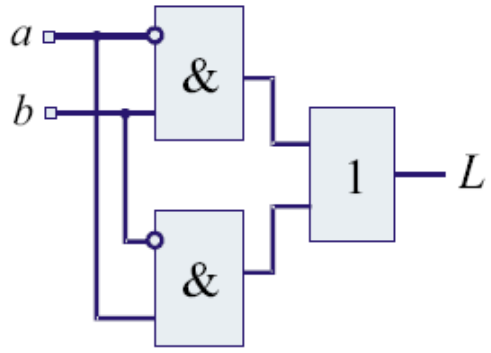


Рисунок 5 – Дедуктивний примітив функції Xor

Схемний примітив є універсальним по відношенню до різних тестових послідовностей. Стратегія, розглянута в поданій магістерській кваліфікаційній роботі щодо синтезу моделей, ґрунтується на створенні бібліотеки дедуктивних елементів, що покривають всі стандартизовані конструктиви функціональностей, якими оперує розробник, створюючи в автоматизованому режимі проєкт у вигляді цифрової системи на кристалі. В даному випадку мова йде про синтез дедуктивної структури на основі меппінгу, суть якого представлена на рис. 6.

Запропонований підхід до дедуктивного аналізу, передбачає створення на кристалі ще однієї вбудованої моделі, яка повинна забезпечувати практично всі шість сервісів, передбачених стандартом інфраструктури I-IP.

Платою за якість діагностичного і тестового обслуговування є досить висока вартість додаткових витрат апаратури, які перевищують штатну функціональність у 10 – 15 разів. При цьому виграш у швидкодії, в порівнянні з зовнішньою програмної реалізацією дедуктивного аналізу, становить 2 – 3 порядки, що практично забезпечує сервісне обслуговування в реальному масштабі часу.

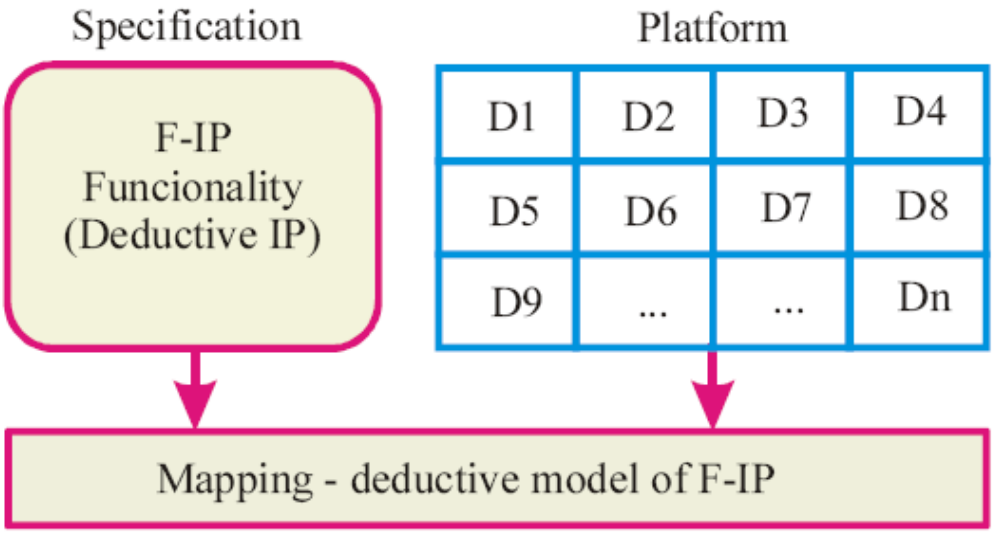


Рисунок 6 – Mapping дедуктивної моделі для F-IP

Інше, більш економічне рішення проблеми, що пов'язане з інтерактивною модифікацією схемної структури дедуктивної моделі для кожного тест-вектора. Для цього використовується внутрішня пам'ять кристала, де формується модель за правилами. Меппінг на рис. 6 дає дедуктивну функцію, де апаратні витрати дорівнюють вартості функціональності F-IP.

2.2 Формування системи сервісної ідентифікації SoC-пам'яті

На завершальній стадії розробки ГАС, для заміни маскової ROM, як правило, використовується пам'ять типу OTP і EPROM з ультрафіолетовим стиранням, яка зручна тим, що вона досить легко перепрограмовується.

Мікросхеми, що випускаються, мають ємність від 256 кбіт до 64 Мбіт при живленні 5 і 3 В, достатню швидкодіє, різними корпусами, в тому числі і для поверхневого монтажу. Організація пристроїв пам'яті може бути типу x8, x16 і x8 / x16. Розшифровка ідентифікації мікросхем пам'яті виду OTP і UV EPROM приведена на рис. 7.

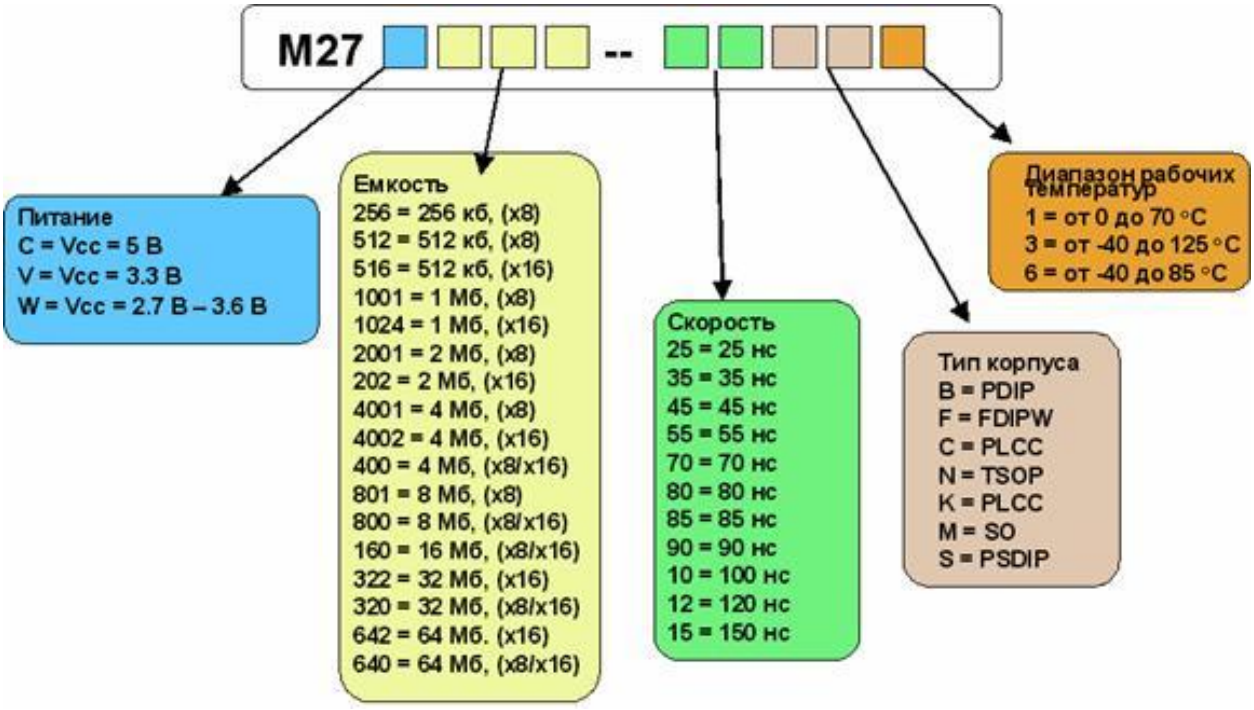


Рисунок 7 – Система ідентифікації SoC-пам'яті типу OTP і UV EPROM

Набір продукції включає стандартні мікросхеми з живленням 5 В і 3,3 В, вдосконалені мікросхеми сімейства Tiger Range з живленням 3 В (2,7 ÷ 3,6 В) і мікросхеми нового сімейства FlexibleROM ™.

Мікросхеми цих типів пам'яті доступні в FDIP керамічних корпусах з віконцем і PDIP пластикових дворядних корпусах, а також в корпусах PLCC і TSOP для поверхневого монтажу.

Для низьковольтної серії Tiger Range використана новітня технологія OTP і UV EPROM. Структурні удосконалення, що пов'язані із товщиною основних верств, дозволили значно поліпшити електричні характеристики. Зменшення на 25% товщини оксидного шару затвора дозволило знизити порогову напруга осередку та збільшити швидкість вибірки при живленні від 2,7 В.

Для поліпшення електричних характеристик, при першому сервісному обслуговуванні, рекомендується замінювати "V" серію з живленням 3 ÷ 3,6 В

на серію "W" – Tiger Range, яка має кращі характеристики при живленні 2,7 ÷ 3,6 В.

Часові параметри для серії Tiger Range гарантуються подвійним тестуванням мікросхем при напрузі 2,7 В і 3 В. Час доступу при живленні 2,7 В маркується на мікросхемі і більш швидкий час доступу специфікується в описі. Часи доступу для напруги живлення вище 2,7 В є робочими.

Сімейство UV і OTP EPROM Tiger Range характеризується надмалим споживанням, високою швидкістю роботи і, одночасно, швидким доступом з коротким часом програмування. Час програмування мікросхем однаково як для послівного, так і побайтного режимів програмування. Для самих останніх мікросхем з параметрами 4 Мб і 8 Мб швидкість програмування доведена до 50 мкс на слово або байт.

Мікросхеми низьковольтної серії Tiger Range повністю сумісні зі стандартною серією 5 В UV і OTP EPROM. Це гарантує їх повну відповідність для додатків, в яких мікропроцесорне живлення замінюється з 5 В на 3 В.

2.3 Інструкції сервісного обслуговування SoC-пам'яті

Технологія щодо EPROM безперервно удосконалюється. Нові перспективи відкриваються з впровадженням нової архітектури мікросхем пам'яті, заснованої на використанні технології багаторозрядної комірки пам'яті для отримання високої щільності запису, починаючи з ємності в 64 Мбіт. Крім того, кожна нова розробка містить кілька фотолітографічних нововведень, що поліпшують електричні характеристики мікросхем.

На даному етапі відкрилися нові можливості поставок мікросхем пам'яті типу PROM (programmable ROM) / RPROM (re-programmable ROM). Ці мікросхеми випускаються у трьох робочих температурних діапазонах: комерційному (від 0 до +70° С), індустріальному (від мінус 40 до +85° С) і військовому (від мінус 55 до +125° С). Крім того, деякі компоненти виготовляються за стандартом для військового призначення (SMD), в тому числі і EPROM.

Самою останньою розробкою в області електрично програмованих постійних запам’ятовуючих пристроїв (ПЗП) є сімейство FlexibleROM ™, яке може використовуватися як проста заміна для будь-якого ПЗП. Це одноразове програмоване сімейство, яке виготовляється за так званою 0.15 мкм технологією, доступно споживачеві з початковою ємністю пам'яті у 16 Мбіт. Нове сімейство мікросхем пам'яті FlexibleROM відноситься до типу енергонезалежної пам'яті і призначене для зберігання програмного коду. FlexibleROM – ідеально підходить для використання замість масочного ПЗП (MaskROM) і переходу від Flash-пам'яті на ПЗП, після налагодження програми, якщо в подальшому не планується зміни програмного коду.

Завдяки технології, заснованої на Flash, час програмування також істотно зменшено. Мікросхеми FlexibleROM забезпечені типовою здатністю багатошаровій програми з великим потоком даних, що дозволяє програмувати пристрій з ємністю 64 Мбіт всього за дев'ять секунд.

Ще однією перевагою, у порівнянні з іншими одноразово-програмованими ПЗУ, є висока продуктивність програмування, оскільки 100% функціональних можливостей масиву пам'яті перевіряються в ході тестування.

Мікросхеми сімейства пам'яті FlexibleROM використовують напругу живлення від 2,7 В до 3,6 В для операцій читання і від 11,4 В до 12,6 В для програмування. Пристрої мають 16-розрядну організацію, за замовчуванням при включенні живлення встановлюється режим пам'яті «Зчитування» так, що вони можуть читатися як ПЗП (ROM) або ЕПЗУ (EPROM).

2.4 Система збереження параметрів сервісного обслуговування

Послідовна незалежна пам'ять – найбільш гнучкий тип довготривалої енергонезалежній пам'яті, яка забезпечує можливість запису аж до байтового рівня, без необхідності стирання даних перед записом нового значення. Це робить їх ідеальними для зберігання параметрів.

Сімейства послідовної Flash-пам'яті мають можливість "секторного стирання / сторінкової прошивки" і "сторінкового стирання / сторінкової прошивки". Це стало можливо завдяки більш тонкої дрібнокоміркової пам'яті у порівнянні зі стандартною Flash-пам'яттю, характеристика зернистості якої не відповідає характеристиці байтового рівня послідовного ЕПЗП.

Електроніка управління виконавчими пристроями ГАС, а також ринку компонентів комп'ютерів і периферії – основні споживачі мікросхем довготривалої пам'яті.

Для EEPROM компанією використовується 0.35 мкм технологія виробництва, що дозволило довести ємність пам'яті до 1 Мбіт у відповідність до потреб ринку. У той же час технологія виготовлення послідовної Flash-пам'яті досягла рівня 0.18 мкм і з'явилася можливість виробництва і цього виду пам'яті повністю відповідно до ринкових запитів.

Асортимент мікросхем послідовної незалежної пам'яті включає набір схем ємністю від 256 біт до 16 Мбіт. Всі мікросхеми пам'яті забезпечені описами, прикладами із застосування і модельними файлами, що робить їх зручними у використанні. По напрузі живлення мікросхеми послідовної незалежній пам'яті ST доступні у п'яти діапазонах:

- від 4,5 В до 5,5 В;
- від 2,5 В до 5,5 В;
- від 2,7 В до 3,6 В;
- від 1,8 В до 5,5 В;
- від 1,8 В до 3,6 В.

Проєктна зносостійкість EEPROM – понад мільйон циклів перезапису з збереженням даних протягом більш ніж 40 років. Мікросхеми виробляються в різних корпусах, включаючи традиційні PSDIP, TSSOP, SO, а також сучасного типу LGA і SBGA (тонкоплівкові). Крім того, є можливість поставки мікросхем в упаковках.

Розроблено широкий діапазон високоякісної послідовної пам'яті EEPROM, з щільністю від 1 кб до 1 Мб, з трьома промисловими стандартами послідовних шин (400 кГц, 2-дротова шина з щільністю до 1 Мбіт, швидка 1 МГц шина типу MICROWIRE з щільністю від 1 кбіт до 16 кбіт і надшвидка 10 МГц шина типу SPI з щільністю до 256 кбіт), із живленням 5 В; 2,5 В і 1,8 В. Система запису позначень послідовної EEPROM для типових корпусів показана на рис. 8. Слід зауважити, що для пластин і мікросхем в барабанах позначення можуть дещо відрізнятися.

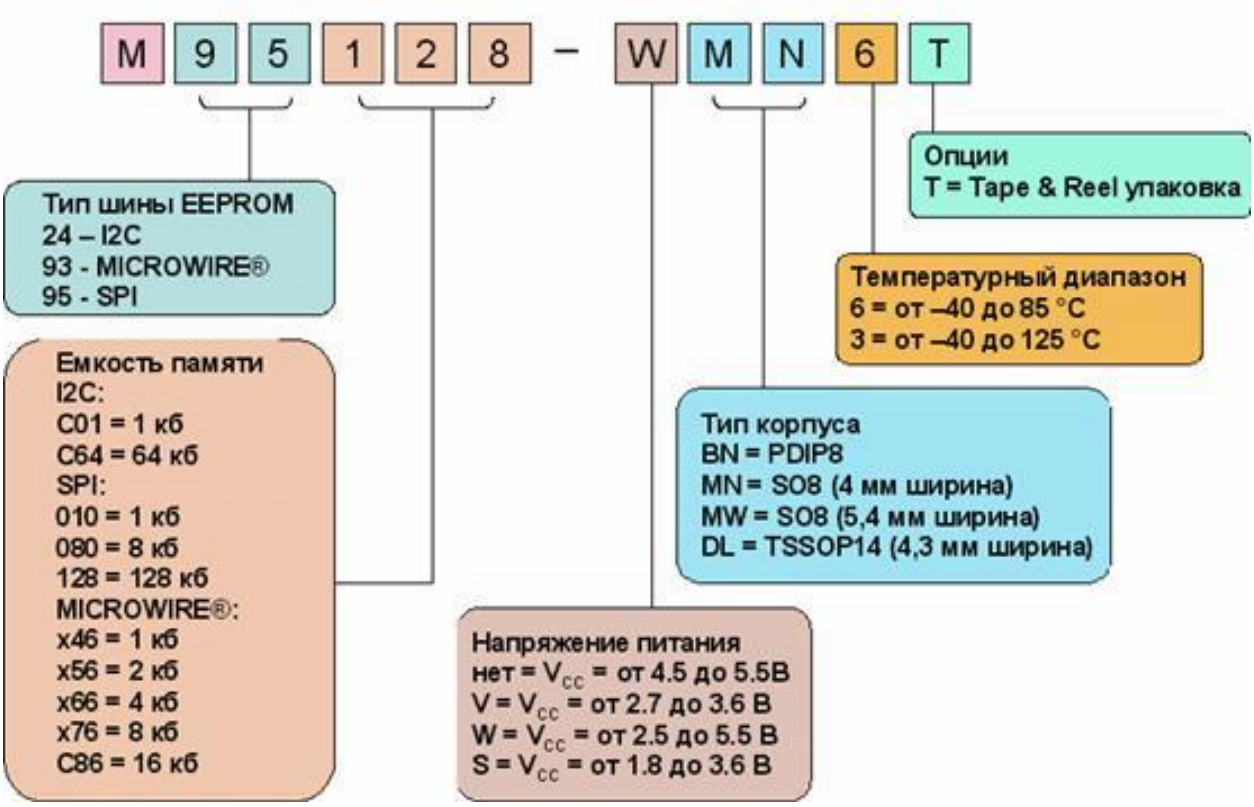


Рисунок 8 – Система запису позначень мікросхем пам’яті ST типу EEPROM

Мікросхеми послідовної EEPROM, рекомендуються для використання у додатках, які не вимагають високої шинної швидкості для накопичення і зберігання даних, але бажаючих мати можливість побайтового і сторінкового читання / запису. Шина працює з тактовою частотою 400 кГц при напрузі живлення до 1,8 В. Послідовна EEPROM випускається у різних корпусах:

пластикових DIP із дворядним розташуванням виводів, SO, MSOP, TSSOP для поверхневого монтажу і SBGA з матрицею кулястих виводів.

Мікросхеми пам'яті EEPROM з шиною SPI кращі для додатків з високошвидкісною передачею інформації по шині. З появою мікросхем зі швидкістю від 5 МГЦ до 10 МГц і ємністю від 512 кбіт до 1 Мбіт, ця шина швидко завойовує популярність на ринку мікросхем пам'яті. EEPROM з шиною SPI мають вхід HOLD ("Захоплення"), який дозволяє зберігати синхронізацію при паузах у процесі передачі послідовностей даних по шині. Крім того, є спеціальний керуючий вхід «W» для захисту матриці пам'яті від запису.

Мікросхеми пам'яті EEPROM з шиною MICROWIRE доступні з ємністю від 256 біт до 16 кбіт. В даний час шина MICROWIRE широко застосовується в багатьох сучасних пристроях, для яких потрібно досить висока швидкість передачі даних без використання зовнішніх шин адреси / даних.

Сімейство мікросхем високошвидкісної низковольтової послідовної Flash-пам'яті має чотирьохпровідний SPI-сумісний інтерфейс, що дозволяє використовувати Flash-пам'ять замість послідовної EEPROM. Виготовляються по високозносостійкої КМОП Flash-технології, дані мікросхеми забезпечують, принаймні, 10 тис. циклів перепрограмування на сектор з збереженням даних понад 20 років.

В даний час доступні дві підродини послідовної Flash-пам'яті з можливістю стирання сектора або сторінки:

- послідовна Flash-пам'ять із секторним стиранням і сторінковим програмуванням: серія M 25 Pxx;
- послідовна Flash-пам'ять із сторінковим стиранням і програмуванням: серія M 45 PExx.

Якщо розглянути різні види мікросхем послідовної довгострокової пам'яті з високою щільністю, то M 25 Pxx з тактовою частотою 25 МГц виявляються значно швидше, ніж багато інших типів схем Flash-пам'яті з послідовною вибіркою.

Сімейство послідовної Flash-пам'яті ST дозволяє завантажувати в оперативну пам'ять 1 Мб за 43 мс при мінімальному числі команд, що робить їх зручними у використанні. Технічні і програмні засоби захисту оберігають збережену інформацію від перезапису.

Для зниження споживаної потужності ці мікросхеми працюють від одного джерела живлення від 2,7 В до 3,6 В і мають режим зниженого енергоспоживання, при якому струм менше 1 мкА. Крім того, чотирьохпровідний інтерфейс значно зменшує число виводів пристрою, що використовуються для управління передачею даних по шині, що забезпечує високу інтеграцію і меншу вартість у порівнянні з іншими подібними схемами. Мікросхеми пам'яті серії M 25 Pxx випускаються в широкому і вузькому корпусах, а саме:

– S08;
– LGA;
– MLP.

2.5 Складові оцінки програмування SoC-пам'яті

У M25Pxx є зручний програматор / зчитувач. Цей програматор підключається безпосередньо до комп'ютера і забезпечує користувачеві прямий доступ і управління послідовною Flash-пам'яттю M25xxx в будь-якої конфігурації.

M45PExx – серія мікросхем незалежної пам'яті високої продуктивності, що володіє більш високою зернистістю, ніж раніше. Будь-яка сторінка у 256 Байт може бути окремо стерта і запрограмована, а команда «Write» передбачає можливість модифікування даних на байтовому рівні. Крім того, архітектура M45PExx оптимізована по мінімуму необхідного прикладного програмного забезпечення. Для модифікування однієї сторінки у 256 байт потрібен час 12 мс для запису, 2 мс для програмування або 10 мс для стирання. Це робить високопродуктивні мікросхеми послідовної незалежної пам'яті

M45PExx зручними для використання у додатках, що вимагають зберігання великої кількості даних, що часто змінюються.

Спеціалізовані мікросхеми пам'яті мають індивідуальні характеристики для конкретних додатків або розробляються відповідно до вимог, що ставляться. Вони засновані на стандартних матрицях пам'яті зі специфічною електричною схемою введення-виведення і спеціалізованої внутрішньою логікою. Ці вироби засновані на послідовній EEPROM і включають логіку для додатків типу комп'ютерного монітора "Plug and Play" зі стандартом VESA, комп'ютерні модулі DRAM тощо.

Серед даних мікросхем можна відзначити M24164 – 16 Кб, що каскадується EEPROM зі спеціальною адресацією, можливістю використання 8 пристроїв каскадом на одній шині і спеціальної адресацією, що використовується при конфліктах на шині I2C.

Іншою спеціалізованою мікросхемою, яка може знайти широке застосування на нашому ринку є M34C00 – електронний дескриптор плати, призначений для зберігання невеликих електронних заміток про плату. У M34C00 можна зберігати реєстраційний номер, заводські установки (за замовчуванням), призначені для користувача установки, дані про події протягом терміну служби плати, відомості про відмови і сервісному обслуговуванні будь-якої плати тощо.

Дана мікросхема має:

- 3 банки по 128 біт (один, що не стирається OTP-типу);
- один стандартний банк EEPROM;
- один стандартний банк EEPROM з можливістю постійного захисту від запису;
- дводротовий I2C шинний послідовний інтерфейс;
- живлення від 2,5 В до 5,5 В;
- корпус SO8 або TSSOP 8;
- робочий діапазон температур від мінус 40 до +85° С.

2.5.1 Особливості безконтактних мікросхем пам'яті

Безконтактні мікросхеми пам'яті є специфічним продуктом. За класифікацією їх, з одного боку, можна віднести до спеціалізованих EEPROM, а, з іншого боку, їх можна виділити як самостійний вид пам'яті, який отримує останнім часом дуже широке застосування у різних сферах.

Новий стандарт ISO для безконтактної комунікаційної пам'яті – ISO 14443 тип B (реалізований у мікроконтролерних пристроях ГАС, транспорту тощо), а також ISO 15693 та ISO 18000.

В даний час запропонована нова серія мікросхем безконтактної пам'яті і безконтактних мікросхем зв'язку з радіочастотним інтерфейсом для додатків типу міток, радіочастотної ідентифікації (RFID) і безконтактних систем доступу з використанням спеціалізованих мікросхем пам'яті. Відзначимо особливості деяких мікросхем даного типу.

Мікросхема SRIX 4K має 4096 призначених для користувача біт EEPROM з OTP, двійковий лічильник і захист записи. Відповідає стандарту ISO 14443 – 2/3 тип B. Має запатентовану компанією France Telecom функцію антиклонування. Працює на частоті 13,56 МГц із частотою, що несе 847 кГц, частота зі швидкістю передачі даних 106 кбіт / с. Використовується амплітудна модуляція (ASK) даних при передачі зі зчитувача на карту і двійкова фазова модуляція (BPSK) для передачі з карти на зчитувач.

Мікросхема LRI 512 має 512 біт з блокуванням на рівні блоку даних. Вона повністю відповідає стандарту ISO 15693 (до 1 метра) і вимогам E. A. S. Працює на частоті 13,56 М Гц з 1/4 і 1/256 імпульсним кодуванням на високій і низькій швидкостях передачі даних на одній або двох несучих частотах. Проводиться амплітудна модуляція даних при передачі зі зчитувача на карту і «манчестерське» кодування при передачі з карти на зчитувач.

У мікросхемі CRX 14 є вбудований у чіп механізм радіозв'язку з протоколом і модуляцією за стандартом ISO 14443 типу B (Radio Service). Володіє запатентованою France Telecom функцією антиклонування. Забезпечує послі-

довний доступ до бази на частоті 400 кГц за дводротовою послідовною шиною I2C із можливістю з'єднання по одній шині з вісьмома CRX 14. Має буфер 32 байта для вхідного і вихідного пакету і вбудований обчислювач циклічного надлишкового коду (CRC-calculator). Випускається у корпусі S016 Narrow (стиснений).

Мікросхеми енергонезалежних ПЗП (NVRAM) дозволяють забезпечити збереження даних ПЗП при збоях і втраті зовнішнього живлення (батареї), що розташовується безпосередньо вгорі мікросхеми або на системній платі. Виходячи із завдань, що вирішуються, із використанням ПЗП, виробляється чотири типи мікросхем NVRAM:

- супервізори;
- ZEROPOWER NVRAM;
- послідовні годинники реального часу (Serial RTC);
- TIMEKEEPER NVRAM.

2.5.2 Особливості супервізорів

Виділяють наступні класи супервизоров:

- супервізори мікропроцесорів (Microprocessor supervisor);
- супервізори енергонезалежних ПЗУ (NVRAM supervisor);
- також можлива комбінація обох класів.

Основними функціями супервізора мікропроцесора (μP) є моніторинг напруги і функція сторожового таймера (Watchdog). Більшість супервізорів мікропроцесорів включають ці функції. У комбінованих мікросхемах можлива інтеграція інших функцій. Основними функціями супервізора NVRAM є моніторинг напруги з перемиканням батареї і захист запису.

3 ІНФРАСТРУКТУРА ПРОЦЕСУ ВЕРИФІКАЦІЇ ПРОЄКТУ

3.1 Модель процесу верифікації

Модель процесу верифікації проєкту на системному рівні можна представити у вигляді узагальненого рівняння виявлення помилок $T \oplus S = L$ або більш детально у компонентах:

$$(T, A) \oplus (P, S, F) = (L_V).$$

де T, A – тестові та асерційні впливи з очікуваними реакціями;

P, S, F – специфікація, HDL-модель функціонального покриття оцінки повноти тесту для верифікації проєктованого виробу.

При тестуванні апаратної реалізації вступає в силу аналітичний вираз:

$$(T) \oplus (S, B) = (L_T)$$

де B – регістр граничного сканування від стандарту IEEE 1500, який використовується у якості доповнення до моделі для забезпечення необхідної глибини діагностування. При цьому L_V, L_T – списки помилок і несправностей, що отримані на стадіях верифікації проєкту і тестування готового виробу.

Для більш точного розуміння співвідношень між ключовими поняттями: верифікація, валідація, сертифікація та асерція уводяться такі визначення [20], [22] [1)]:

[1)] [20] Кривченко И. Системная интеграция в микроэлектронике FPSLIC. *Chip News*. 2012. №3. С. 4 –10.

[22] Королев Н. ATMEL FPSLIC – элементная база XXI века. *Chip News*. 2011. №1. С. 16–19.

– верифікація – є процес аналізу системи або компонентів для визначення коректності перетворень вхідного опису на черговий стадії проєктування;
– валідація – є процес визначення працездатності системи і її компонентів, шляхом перевірки відповідності основним вимогам специфікації після виконання кожної стадії проєктування;
– сертифікація – є письмова гарантія того, що система або її компоненти повністю задовольняють вимогам специфікації і прийнятні для використання за призначенням;
– асерція – є вислів системного рівня, що визначає коректність перетворень в процесі проєктування щодо вхідного опису поточного етапу або вимог специфікації.

Слідуючи останнім визначенням, асерцію можна записати у вигляді предикату:

$$A_i = f(A_{i1}, A_{i2}, ..., A_{ij}, ..., A_{i_n}) = Y = \{0,1\},$$

який на множині змінних i_n приймає значення: {true, false}.

Відповідно до згаданого визначенням, асерції можна диференціювати на три типи:

$$A = \{A_f, A_p, A_s\},$$

а саме перевірки:

– несправностей:
– функціональних шляхів;
– можливих станів виробу.

Подібна класифікація дозволяє перевірити у програмному коді як механізм обчислень, так і управління обчислювальними процесами. Механізм асерцій – є повною системою висловлювань $A = (A_1, A_2, ..., A_i, ..., A_n)$ та засо-

бами їх аналізу, що призначені для перевірки та затвердження процесу проєктування, а також діагностування помилок.

Стратегії верифікації та тестування мають різні моделі, програми та технології, що орієнтовані на згадані процеси. Для верифікації характерною ознакою є робота з HDL-моделлю, отриманою за специфікацією проєкту, яка проходить стадії перетворень, пов'язаних з його проєктуванням (рис. 9) (синтез та імплементація) у силіконовий кристал.

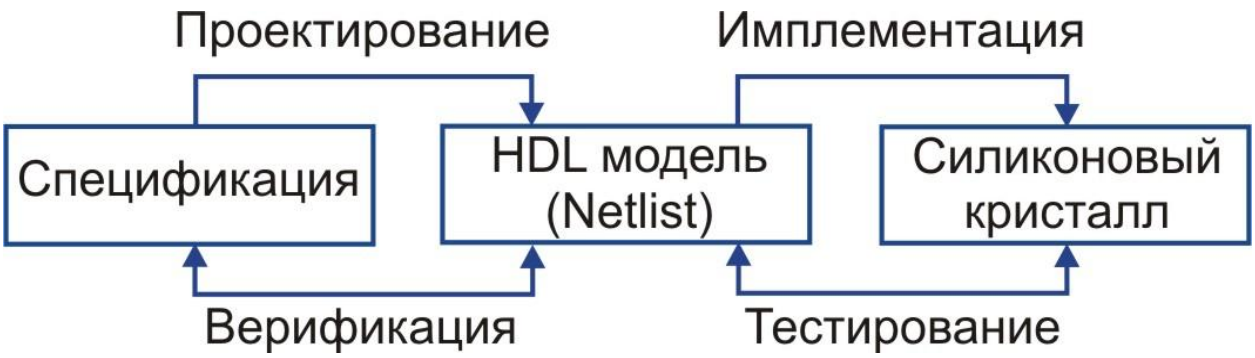


Рисунок 9 – Стратегії розробки проєкту

Процес тестування тут відображає взаємодію апаратної та HDL-моделі на несистемному (регістровому або вентильному) рівні проєктування при імплементації ідеї у кристал програмованої логіки. Для кристалів ASIC така технологія недоцільна, оскільки перепрограмування помилки тут буде коштувати до мільйона доларів.

З урахуванням наведених визначень і пояснень модель середовища або макропроцесу верифікації програмної стадії проєкту орієнтована на зменшення часу створення виробу та підвищення рівня виходу придатної продукції за рахунок використання надмірності коду у вигляді механізму асерцій і використання Testbench спільно з метрикою визначення якості тесту або функціональної повноти.

Технологія тестування і верифікації HDL-моделі представлена на рис. 10, де специфікація проєкту, описана формальною мовою високого рівня, є вихідною інформацією для створення метрики оцінювання:

– якості тесту;
– моделі проєкту;
– тесту з еталонними реакціями – testbench;
– асерційної структури, яка є доповненням до основної моделі, що необхідно для прискорення тестування і налагодження проєкту.

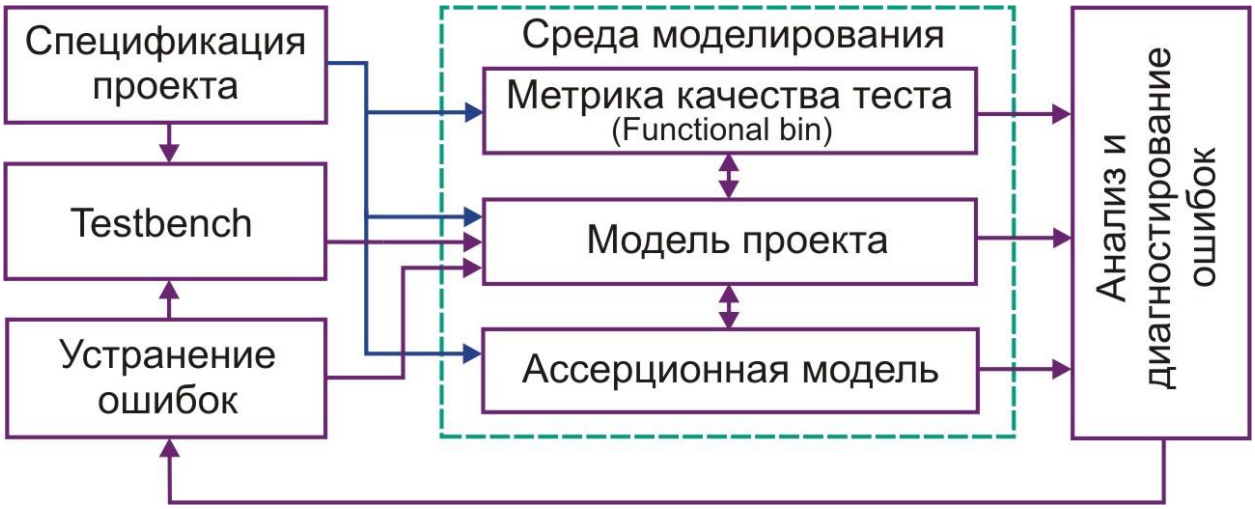


Рисунок 10 – Середа верифікації проєкту

Середа верифікації представлена системою моделювання (наприклад, Questa, Mentor Graphics), тестування (Testbench), механізмом асерцій (Assertion Engine) і власне системним кодом моделі на мовах VHDL, Verilog, System Verilog.

Модуль Testbench задає вхідні стимули і еталонні реакції на них, записані на HDL-мовах, орієнтовані на перевірку функциональностей (змінні, функції, послідовності), параметри яких визначаються у функціональному кошику. Механізм асерцій – модельна надмірність, яка доповнює testbench, в частині перевірки внутрішніх за часом станів проєкту, представлена вхід-

вихідними висловлюваннями і призначена для прискорення тестування, верифікації, діагностування та виправлення помилок проєктування в системному коді. Асерції можна згенерувати не по специфікації, але і по HDL-моделі, прибираючи непотрібні конструкції, а решту необхідно модифікувати до форми асерцій.

3.2 Аналіз тестопридатності програмних HDL-моделей

Для ідентифікації узагальненого стану HDL-моделі формуються еталонні сигнатури критичних точок у часі і в просторі. Потім виконується діагностування HDL-моделі по сигнатурам асерційної матриці, яка описує стан проєкту в часі і в просторі. Доцільно формувати асерційну матрицю незалежно від HDL-моделі проєкту. Асерційна і функціональна моделі обробляються паралельно і незалежно один від одного середовищем моделювання.

Асерційна модель орієнтована на тестування внутрішніх в часі і в просторі точок проєкту, позначених його просторово-часової матрицею. Асерційна модель визначає поведінку проєкту в істотних точках просторово-часової еталонної матриці. Формат асерції повинен відповідати формату матриці HDL-моделі проєкту. Асерції орієнтовані на додаток повноти Testbench в частині тестування внутрішніх точок матриці HDL-моделі.

Аналітична форма подання інфраструктури верифікації представлена наступними виразами:

$$M = \{P, S, A, T, F, D, C\},$$

$$1)\ S = f_1(P) = \left| S_{ij} \right|;$$

$$2)\ F = f_2(P, S) = \{F_1, F_2, ..., F_i, ..., F_n\};$$

$$3)\ T = f_3(P,S,F) = \{T_1, T_2, ..., T_i, ..., T_n\};$$

$$4)\ A = f_4(P,S,F,T) = \left|A_{ij}\right|;$$

$$5)\ D = f_5(P,S,F,T,A) = \left|L_{ij}\right| \in [L^V \vee L^T];$$

$$6)\ C = [\bigcup_{i=1}^{n} F_i \in F = P] \wedge [\bigcup_{i=1}^{n} T_i \in T = F];$$

$$7)\ L^V = (T \oplus P) \vee (A = T^A \oplus P^A);$$

$$8)\ L^T = (T \oplus S),$$

де P – специфікація проєкту, S – soft-модель, A – асерційна модель, T – Testbench, F – кошик завдання функціональностей для визначення їх повноти покриття, D – модуль діагностування помилок і C – умови діагностування помилок.

Тут вираз 6 визначає умови повноти функціональних кошиків щодо специфікації та повноти тесту щодо функціональностей проєкту. Рядок 7 задає функцію обчислення помилок проєктування при переході від системного до регистрового рівня, використовуючи всі атрибути верифікаційної інфраструктури. Функція 8 регламентує перебування несправностей на стадії експлуатації цифрової системи на кристалі. Слід зазначити, що асерційна надмірність є функцією від критичних точок функціональної моделі, граничне число яких може бути дорівнює числу, визначених специфікацією, тимчасових фреймів функціональних компонентів, що ілюструється наступними взаємодіючими матрицями:

$$T_i^B, T_i^A, S_i^t, A_i^t, R_i^B, R_i^A,$$

де відповідно: testbench, асерційні вхідні стимули, функціональний компонент у момент *t*, асерційний компонент у момент *t*, реакція функції, реакція асерції.

T_1^B T_2^B T_i^B T_n^B	$Sim \rightarrow$	S_1^1 S_1^2 S_1^t S_1^m S_2^1 S_2^2 S_2^t S_2^m S_1^1 S_i^2 S_i^t S_i^m S_n^1 S_n^2 S_n^t S_n^m	$Sim \rightarrow$	R_1^B R_2^B R_i^B R_n^B
		$\updownarrow$		
T_1^A T_2^A T_i^A T_n^A	$Sim \rightarrow$	A_1^1 A_1^2 A_1^t A_1^m A_2^1 A_2^2 A_2^t A_2^m A_1^1 A_i^2 A_i^t A_i^m A_n^1 A_n^2 A_n^t A_n^m	$Sim \rightarrow$	R_1^A R_2^A R_i^A R_n^A

Матриця асерцій повинна створюватися незалежно від матриці функціональностей і бажано іншим розробником. Така умова забезпечить диверсифікацію згаданих моделей щодо єдиної специфікації, а також виявлення та виправлення помилок у обох матрицях.

Інтерес представляє і послідовність дій зі створення середовища верифікації, де єдиним аргументом є специфікація проєкту, все інше – похідні від неї, представлені матрицею контрдосяжності:

$$M = \begin{array}{|c|c|c|c|c|c|} \hline P & S & & & & \\ \hline P & S & F & & & \\ \hline P & S & F & T & & \\ \hline P & S & F & T & A & \\ \hline P & S & F & T & A & D \\ \hline \end{array}$$

Відповідно до моделі *M* на рис. 11 зображена структура взаємозв'язків процесу проєктування та діагностування з подальшим виправленням помилок в HDL-коді програми.

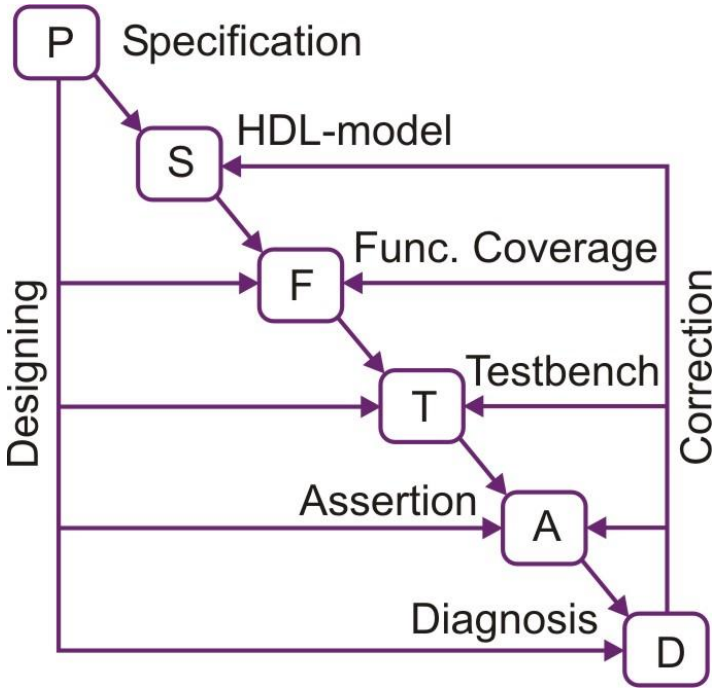


Рисунок 11 – Маршрут верифікації проєкту

Досить істотна надмірність HDL-моделі передбачає визначення ефективності її використання в цілях підвищення тестопридатності структури розробленого або написаного коду. Використовуючи стандарти тестопридатного проєктування апаратури, можна перенести і адаптувати основні поняття стосовно HDL-коду системних і реєстрових програмних моделей.

Тут можна використовувати граф реєстрових або транзакційних передач С. Г. Шаршунова, який надає користувачеві інформацію про взаємозв'язки булевих і реєстрових змінних, пам'яті і інтерфейсних шинах. Такі дані досить легко можна отримати автоматично, аналізуючи синтаксис рядків HDL-коду. Згенерований граф повинен покривати згадані вище компоненти програмної моделі і позначати всі існуючі зв'язки для прийому, передачі і перет-

ворення інформації між вершинами графа транзакцій (*TG* – Transaction Graph).

Керованість і спостережливість є метрика оцінювання тестопридатності не сполучних ліній, а компонентів HDL-коду, таких як:

– регістр;

– лічильник;

– пам'ять або масиви;

– вхідні-вихідні шини;

– вектори;

– логічні або арифметичні змінні.

Зазначені вище характеристики мають функціональну залежність від структурної глибини знаходження компонента (в просторі або в часі), а також від процентного відношення числа команд, що мають вхідний (вихідний) доступ до вершини при аналізі даної програми:

$$Q = \frac{1}{Z}(U \times N) = \frac{Z(S)}{Z(S)+Z(F)+Z(T)+Z(A)} \times$$

$$\times (\frac{1}{n}\sum_{i=1}^{n} U_i) \times (\frac{1}{n}\sum_{i=1}^{n} N_i);$$

$$U_i = \frac{1}{T}\sum_{j=1}^{x_i} T_j^i \times \frac{1}{d_i^x \vee t_i^x};$$

$$N_i = \frac{1}{T}\sum_{j=1}^{y_i} T_j^i \times \frac{1}{d_i^y \vee t_i^y}.$$

де Q – тестопридатність, залежить від керованості (U), спостережливості (N), а також від модельної надмірності, представленої компонентами: метрика функціонального покриття (F), testbench (T), механізм асерцій (A).

При цьому керованість є функція від числа операторів, що входять до вершини (що виходять з вершини) транзакційного графа, а також від структурної глибини розглянутого елемента – відстані від вхідної (вихідної) шини

або від кількості часових тактів, необхідних для управління (спостереження) компонента в заданому стані на тимчасової осі. Наведений критерій тестопридатності може бути також використаний і для оцінки якості граф-схеми управління обчислювальним процесом. Тут розглядаються тільки операторні вершини, навантажені вхідними умовами, а також позиція вершини по відношенню до початку або закінчення схеми управління.

Позиція операторної вершини корелюється з тимчасовим тактом управління обчислювальним процесом. Кількість умов виконання сукупності операцій у кожній вершині, об'єднане операцією *Or*, підвищує тестопридатність графа в частині керованості. Аналогічно обчислюється спостережливость, на яку впливає не тільки структурна глибина, але і потужність умов, створювана функціональними операціями *And, Or*.

Структурна глибина розглянутої вершини, а значить і тестопридатності може бути опосередкована тільки логічною функцією, заданою у вигляді кон'юнктивної нормальної форми (КНФ). При цьому тестопридатність (керованість і спостережливість) буде визначатися оцінкою по Квайну обчислювальної складності КНФ. У загальному випадку логічні функції спостереження та управління поточної вершини транзакційного графа представлені кон'юнкцією діз'юнктивних термів:

$$U_i = \frac{1}{T}(\bigwedge_{j=1}^{x_1} T_{1j}^{x})...(\bigwedge_{j=1}^{x_{i-1}} T_{i-1j}^{x})(\bigvee_{i=1}^{x_i} T_{ij}^{x})$$

$$N_i = \frac{1}{T}\bigwedge_{j=1}^{x_i} T_j^i(\bigvee_{i=1}^{x_i} T_j^i d_i^x \vee t_i^x)$$

Тут потужність діз'юнктивних термів відповідає кількості дуг, входять до вершини, а число кон'юнкцій є структурна глибина розташування розглянутого компонента.

Цікавим видається рішення, коли керованість і спостережливість поточної вершини транзакційного графа обчислюється на підставі побудованих

логічних функцій керованості та спостереження (U_i, N_i) при використанні апарату – алгебраїчної форми подання графа. Формули підрахунку згаданих критеріїв мають такий вигляд:

$$U_i = \frac{1}{t^x_{\max} \times n^x_t} \times \sum_{i=1}^{n^x_t} \sum_{j=1}^{k^x_i} (t^x_{\max} - \left|t^x_{ij}\right| + 1);$$

$$N_i = \frac{1}{t^y_{\max} \times n^y_t} \times \sum_{i=1}^{n^y_t} \sum_{j=1}^{k^y_i} (t^y_{\max} - \left|t^y_{ij}\right| + 1),$$

де $t^x_{\max}, n^x_t, k^x_i, \left|t^x_{ij}\right|$ – для критерію керованості кон'юнктивний терм максимальної довжини; кількість термів в логічній функції керованості; кількість транзакцій (літер) у поточному термі функції; потужність даної транзакції в термі. Аналогічні позначення використовуються і для критерію спостережливості – $t^y_{\max}, n^y_t, k^y_i, \left|t^y_{ij}\right|$.

Для фрагмента графа, представленого на рис. 12, перетворення кон'юнктівної форми в диз'юнктивну структуру для вершини V_3, формує логічну функцію керованості:

$$V_3 = (T_1 \vee T_2 \vee T_3)T_5T_8 \vee (T_4 \vee T_6)T_8 \vee (T_7 \vee T_9) =$$
$$T_1T_5T_8 \vee T_2T_5T_8 \vee T_3T_5T_8 \vee T_4T_8 \vee T_6T_8 \vee T_7 \vee T_9.$$

$$V2 = (T_1 \vee T_2 \vee T_3)T_5 \vee T_4 \vee T_6 =$$
$$T_1T_5 \vee T_2T_5 \vee T_3T_5 \vee T_4 \vee T_6.$$

$$V_1 = T_1 \vee T_2 \vee T_3.$$

Дотримуючись правил, здійснюється визначення керованості компонента V_3:

$$U_i = \frac{1}{t^x_{\max} \times n^x_t} \times \sum_{i=1}^{n^x_t} \sum_{j=1}^{k^x_i} (t^x_{\max} - \left| t^x_{ij} \right| + 1) =$$

$$= \frac{1}{3 \times 7} \times (1+1+1+2+2+3+3) = 0{,}61$$

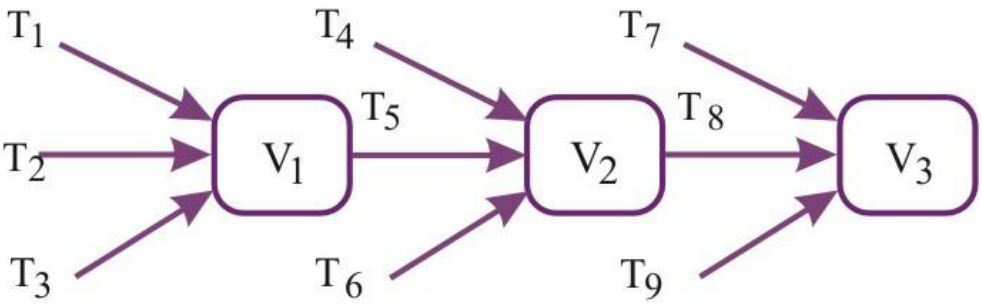


Рисунок 12 – Фрагмент графу для підрахунку керованості

Для іншого фрагмента графа, представленого на рис. 13, перетворення кон'юнктивної форми в диз'юнктивну структуру дозволяє визначити логічну функцію спостережливості вершини V_1:

$$V_1 = T_6 \vee T_7(T_3 \vee T_5 \vee T_4(T_1 \vee T_2) =$$
$$= T_6 \vee T_7T_3 \vee T_7T_5 \vee T_7T_4T_1 \vee T_7T_4T_2$$
$$V_2 = T_3 \vee T_5 \vee T_4(T_1 \vee T_2) = T_3 \vee T_5 \vee T_4T_1 \vee T_4T_2$$
$$V_3 = T_1 \vee T_2.$$

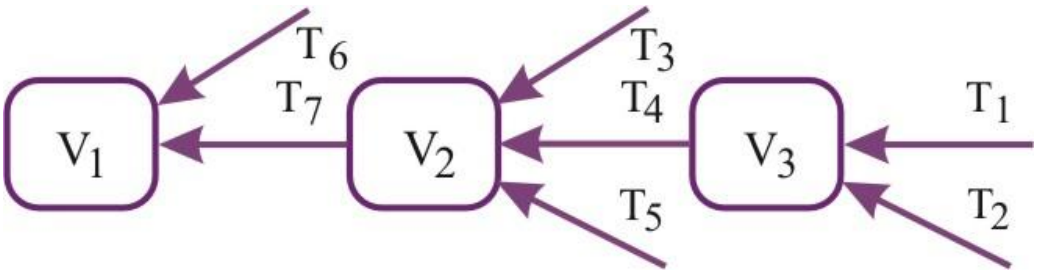


Рисунок 13 – Фрагмент графу для підрахунку спостережливості

Дотримуючись правил, визначимо:

$$N_i = \frac{1}{t^y_{max} \times n^y_t} \times \sum_{i=1}^{n^y_t} \sum_{j=1}^{k^y_i} (t^y_{max} - \left| t^y_{ij} \right| + 1) =$$

$$= \frac{1}{3 \times 5} \times (3 + 2 + 2 + 1 + 1) = \frac{9}{15} = 0{,}6$$

Для випадку, коли дуги в транзакційному графі мають вагові коефіцієнти, що показують число операторів, задіяних в передачі інформації між вершинами (компонентами), формули підрахунку тестопридатності мають такий вигляд:

$$U_i = \frac{1}{t^x_{\max} \times (n^x_t = \sum_{i=1}^{n^x_t} \prod_{j=1}^{k^x_i} b^x_{ij})} \times \sum_{i=1}^{n^x_t} \prod_{j=1}^{k^x_i} b^x_{ij} (t^x_{\max} - \left| t^x_{ij} \right| + 1);$$

$$N_i = \frac{1}{t^y_{\max} \times n^y_t = (\sum_{i=1}^{n^y_t} \prod_{j=1}^{k^y_i} b^y_{ij})} \times \sum_{i=1}^{n^y_t} \prod_{j=1}^{k^y_i} b^y_{ij} (t^y_{\max} - \left| t^y_{ij} \right| + 1).$$

4 ВЕРИФІКАЦІЯ ПРОГРАМНИХ МОДУЛІВ ГАС У ПАКЕТІ КРИСТАЛІВ

4.1 Тестопридатності графа управління

З огляду на, що автоматна модель програмного продукту представлена взаємодією операційного і керуючого автомата [1] [1)], рис. 14, то поряд з моделюванням транзакційного графа, необхідно мати можливість аналізувати тестопридатності граф-схеми алгоритму управління (ГСАУ).

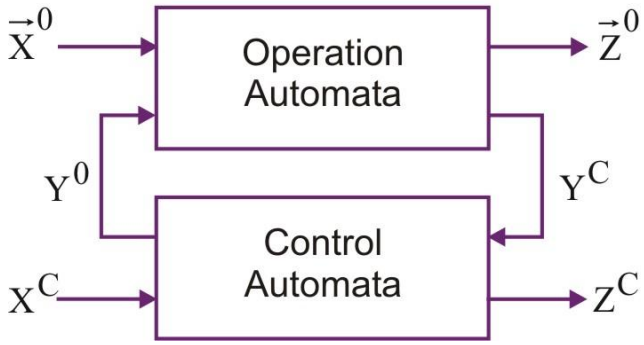


Рисунок 14 – Автоматна модель HDL-програми

Пропонується ГСА представити у вигляді змістовного графа управління (ЗГУ), який є подібним до транзакційного графу. Тут вершини є операції програмного коду, а дуги представляють умови переходу з однієї вершини в іншу для виконання команди, позначеної вершиною-стоком. Отже, для ЗГУ можна використовувати процедури, раніше розроблені для підрахунку критеріїв тестопридатності транзакційного графа в частині спостереження та управління. Прикладом змістовного графа може служити рис. 15, що має 6 вершин і 9 дуг.

[1)] [1] Хаханов В. И., Хаханова А. В., Литвинова Е. И. Алгебро-логический метод ремонта встроенной памяти SoC. *Відмовостійкі системи*. 2018. С. 99–109.

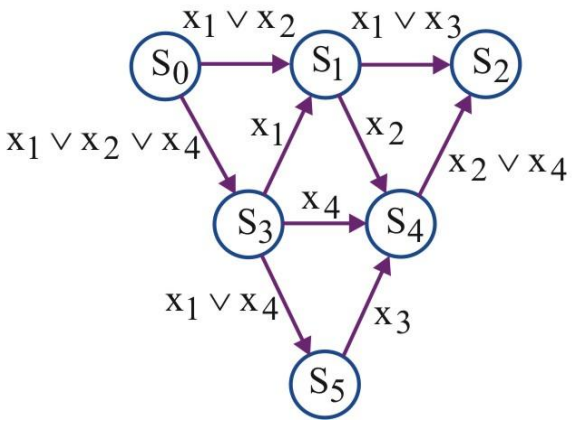


Рисунок 15 – Змістовний граф HDL-програми

Підрахунок керованостей графа, представленого на рис. 15, має такий вигляд:

$$S_1 = T_3^3 T_4^1 \vee T_1^2;\ \ S_3 = T_3^3;$$

$$S_4 = T_3^3 T_4^1 T_6^1 \vee T_3^3 T_5^1 \vee T_3^3 T_8^2 T_9^1.$$

$$S_2 = T_3^3 T_4^1 T_6^1 T_7^2 \vee T_1^2 T_2^2 \vee T_3^3 T_5^1 T_7^2 \vee T_3^3 T_4^1 T_7^2 \vee T_3^3 T_8^2 T_9^1 T_7^2.$$

$$S_5 = T_3^3 T_8^2.$$

Підрахунок спостережливості графу, представленого на рис. 15, містить такі вирази:

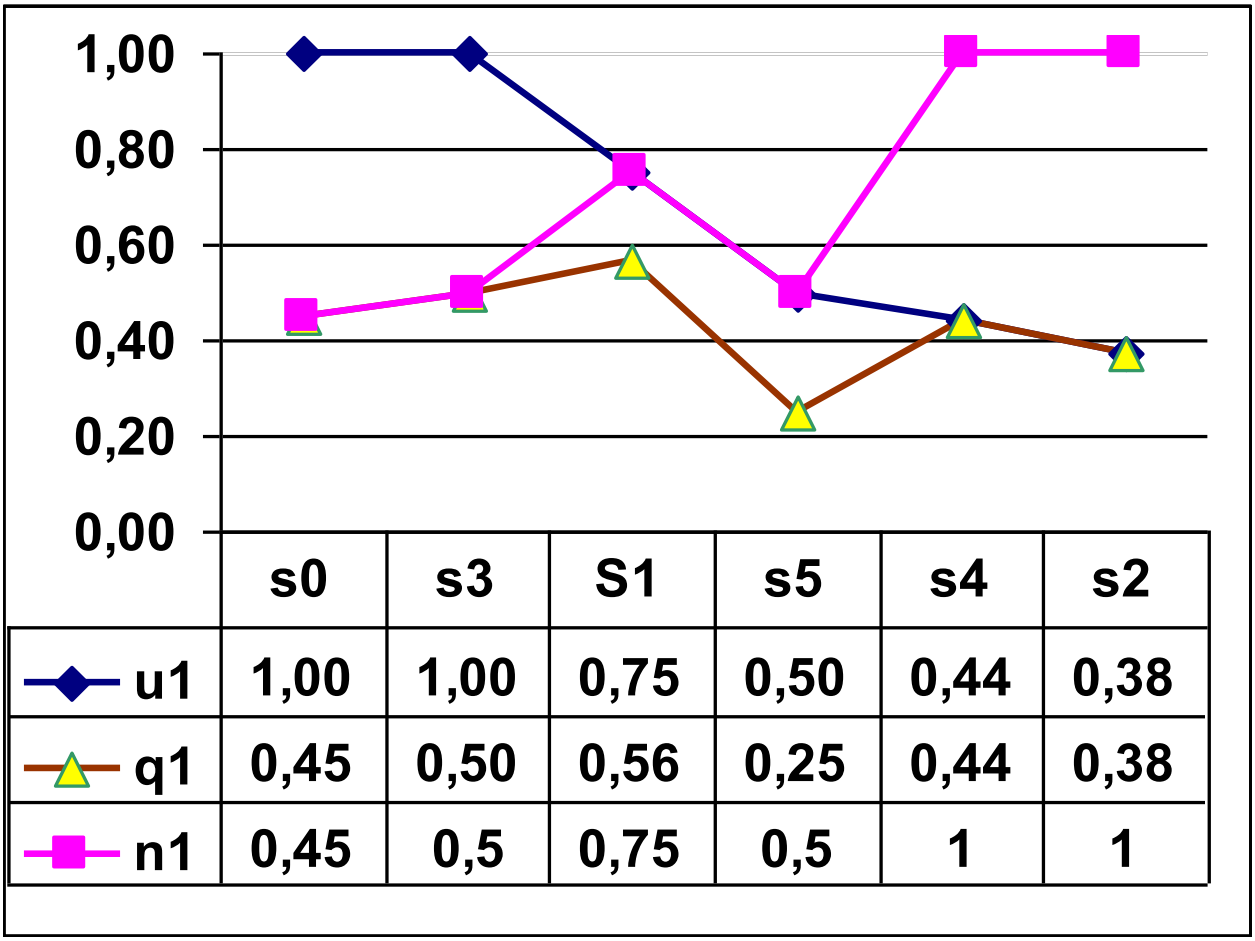
$$S_0 = T_2^2 T_1^2 \vee T_7^2 T_6^1 T_4^1 T_3^3 \vee T_7^2 T_5^1 T_3^3 \vee T_7^2 T_5^1 T_3^3 \vee T_7^2 T_9^1 T_8^2 T_3^3 \vee T_2^2 T_4^1 T_3^3.$$

$$S_1 = T_2^2 \vee T_7^2 T_6^1;$$

$$S_3 = T_7^2 T_5^1 \vee T_7^2 T_9^1 T_8^2 \vee T_7^2 T_6^1 T_4^1 \vee T_2^2 T_4^1.$$

$$S_4 = T_7^2;\ S_5 = T_7^2 T_9^1.$$

Для використання тестопридатності виконується побудова спостереження та управління всіх компонентів HDL-моделі (рис. 16).



	s0	s3	S1	s5	s4	s2
u1	1,00	1,00	0,75	0,50	0,44	0,38
q1	0,45	0,50	0,56	0,25	0,44	0,38
n1	0,45	0,5	0,75	0,5	1	1

Рисунок 16 – Графіки тестопридатності для графа управління

Потім обчислюється узагальнена характеристика – тестопридатності кожного компонента як добуток спостереження та управління:

$$Q_i = U_i \times N_i .$$

Далі інтерес представляє створення таблиці тестопридатності, спостереження та управління, а також відповідний їм графік для візуального контролю «зіпсованих» компонентів. Фіксація певної планки тестопридатності, нижче якої значення будуть вважатися неприйнятними, дозволить розробнику створювати асерції і інші додаткові засоби підвищення тестопридатності для проблемних функціональних блоків.

Крім того, засоби підвищення тестопридатності повинні забезпечувати глибину діагностування до функціонального компонента і прив'язаних до нього операцій, з метою швидкого відновлення працездатності програмної HDL-моделі. З метою побудови алгоритмів пошуку помилок в програмному коді можна використовувати таблицю несправностей, за аналогією з технологією тестування hardware.

Цікаве рішення в процесі перевірки функціональних блоків пов'язано з сигнатурним аналізом, де узагальнена сигнатура ототожнюється зі справною поведінкою всього коду, а також з кожним компонентом. Будь-яка розбіжність еталонної сигнатури з фактичною призводить до виконання процедури діагностування та відновлення працездатності HDL-моделі, шляхом виправлення семантики коду.

Запропонована модель верифікації HDL-проєкту використовує:

- testbench;
- функціональне покриття;
- механізм асерції;
- метрику оцінки тестопридатності;
- інструкцію з усунення помилок;
- вектор експериментальної перевірки (ВЕП), що формується за заданими контрольним точкам шляхом порівняння сигнатур.

Функціональне обмеження testbench пов'язано з нерозрізненістю компонентів програмного коду, в яких можуть бути помилки. Його основне призначення – перевірка справності HDL-моделі. Тому в якості доповнення до процедури перевірки надається механізм асерції, основна мета якого із заданою глибиною – до програмного компонента – визначити місце і вид помилки на стадії виконання діагностування, після того, як testbench зафіксував неправильне функціонування програмного проєкту.

Дві стадії верифікації: тестування і діагностування – представлені нижче у вигляді наступних двох векторно-матричних операцій:

T_1^B				$\oplus$	S_1^1	S_1^2	S_1^t	S_1^m	$=$	L_1^B									
T_2^B					S_2^1	S_2^2	S_2^t	S_2^m		L_2^B									
T_i^B					S_1^1	S_i^2	S_i^t	S_i^m		L_i^B									
T_n^B					S_n^1	S_n^2	S_n^t	S_n^m		L_n^B									
							$\cup$												
A_1^1	A_1^2	A_1^t	A_1^m	$\oplus$	S_1^1	S_1^2	S_1^t	S_1^m	$=$	L_1^1	L_1^2	L_1^t	L_1^m						
A_2^1	A_2^2	A_2^t	A_2^m		S_2^1	S_2^2	S_2^t	S_2^m		L_2^1	L_2^2	L_2^t	L_2^m						
A_1^1	A_i^2	A_i^t	A_i^m		S_1^1	S_i^2	S_i^t	S_i^m		L_1^1	L_i^2	L_i^t	L_i^m						
A_n^1	A_n^2	A_n^t	A_n^m		S_n^1	S_n^2	S_n^t	S_n^m		L_n^1	L_n^2	L_n^t	L_n^m						

Для першої стадії використовується двійковий вектор експериментальної перевірки Z^B, що формується на основі процедури тестування. На другій стадії використовується вже матриця Z^A експериментальної перевірки, яка з наперед заданою глибиною визначає діагноз проєкту на основі порівняння технічних станів HDL-моделі та механізму асерцій:

$$Z^B = (Z_1, Z_2, .., Z_i, ..., Z_n);$$

$$Z^A = \begin{array}{|c|c|c|c|} \hline Z_1^1 & Z_1^2 & Z_1^t & Z_1^m \\ \hline Z_2^1 & Z_2^2 & Z_2^t & Z_2^m \\ \hline Z_1^1 & Z_i^2 & Z_i^t & Z_i^m \\ \hline Z_n^1 & Z_n^2 & Z_n^t & Z_n^m \\ \hline \end{array}$$

В процесі виконання процедури верифікації, виконується порівняння фактичного і еталонного (специфікованого) технічного стану компонента шляхом застосування операції *Xor*:

$$\{ Z_i, Z_i^t \} = S_i^t \oplus S_i^t(r) = \{0,1\}.$$

Практично, якщо виконані умови тестопридатності і правильно розставлені асерції в критичних точках програмного коду для діагностування всіх компонентів, то ВЕП може однозначно ідентифікувати адресу (місце) і тип помилки на основі побудованої раніше таблиці несправностей – механізму асерцій.

4.2 Верифікація проєкту Xilinx IP-core

Представлені моделі верифікації програмного HDL-коду перевірені на реальному проєкті Xilinx IP-core з метою визначення наявності в ньому помилок. При цьому вдалося отримати позитивний результат щодо невірної семантики роботи програми для подальшого виправлення коду. Фрагмент модуля дискретного косинусного перетворення представлений лістингом 1 (Xilinx.com). Вся HDL-модель налічує 900 рядків коду System Verilog.

Лістинг 1

```
module Xilinx
`timescale 1ns/10ps
module dct ( CLK, RST, xin,dct_2d,rdy_out);
output [11:0] dct_2d;
input CLK, RST;
input[7:0] xin; /* input */
output rdy_out;
wire[11:0] dct_2d;
…….

/* The first 1D-DCT output becomes valid after 14 +64 clk cycles. For the first 2D-DCT output to be valid it takes 78 + 1clk to write into the ram + 1clk to write out of the ram + 8 clks to shift in the 1D-DCT values + 1clk to register the 1D-DCT values + 1clk to add/sub + 1clk to take compliment + 1 clk for multiplying +  2clks to add product. So the 2D-DCT output will be valid at the 94th clk. rdy_out goes high at 93rd clk so that the first data is valid for the next block*/
Endmodule
```

Відповідно до правил тестопридатного аналізу, наведеними вище, спроєктований транзакційний граф як розвиток графа реєстрових передач,

представлений на рис. 17, який для module Xilinx має 28 вершин-компонентів:

– вхідні шини;
– вихідні шини;
– логічні змінні;
– реєстрові змінні;
– вектори;
– пам'ять.

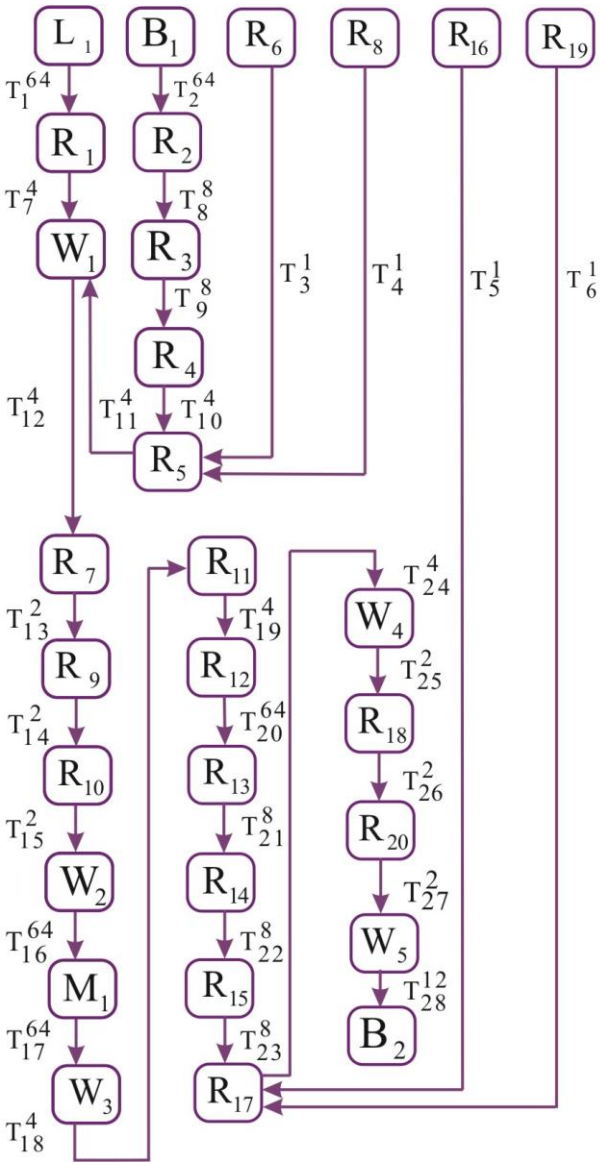


Рисунок 17 – Транзакційний граф Xilinx моделі

Ідентифікатор дуги має верхній індекс, що позначає число транзакцій в програмі між вихідною та вхідною вершинами. Для кожної вершини будуються логічні функції спостереження та управління. Приклад логічної функції керованості для вершини має наступний вигляд:

$$
\begin{aligned}
B_2 = {} & T_{28}^{12} T_{27}^{2} T_{22}^{2} T_{25}^{2} T_{24}^{4} (T_5^1 \vee T_6^1 \vee \\
\vee {} & T_{12}^{4} T_{13}^{2} T_{14}^{2} T_{15}^{2} T_{15}^{64} T_{17}^{64} T_{18}^{4} T_{19}^{4} T_{20}^{64} T_{21}^{8} T_{22}^{8} T_{23}^{8} (T_1^{64} T_7^4 \vee \\
\vee {} & T_{11}^{4} T_2^{64} T_8^8 T_9^8 T_{10}^4 \vee T_{11}^4 T_3^1 \vee T_{11}^4 T_4^1)) = \\
= {} & T_{28}^{12} T_{27}^{2} T_{22}^{2} T_{25}^{2} T_{24}^{4} T_5^1 \vee T_6^1 T_{28}^{12} T_{27}^{2} T_{22}^{2} T_{25}^{2} T_{24}^{4} \vee \\
\vee {} & T_{28}^{12} T_{27}^{2} T_{22}^{2} T_{25}^{2} T_{24}^{4} T_{12}^{4} T_{13}^{2} T_{14}^{2} T_{15}^{2} T_{15}^{64} T_{17}^{64} T_{18}^{4} \wedge \\
\wedge {} & T_{19}^{4} T_{20}^{64} T_{21}^{8} T_{22}^{8} T_{23}^{8} T_1^{64} T_7^4 \vee \\
\vee {} & T_{28}^{12} T_{27}^{2} T_{22}^{2} T_{25}^{2} T_{24}^{4} T_{12}^{4} T_{13}^{2} T_{14}^{2} T_{15}^{2} T_{15}^{64} T_{17}^{64} T_{18}^{4} \wedge \\
\wedge {} & T_{19}^{4} T_{20}^{64} T_{21}^{8} T_{22}^{8} T_{23}^{8} T_{11}^{4} T_2^{64} T_8^8 T_9^8 T_{10}^4 \vee \\
\vee {} & T_{11}^{4} T_3^1 T_{28}^{12} T_{27}^{2} T_{22}^{2} T_{25}^{2} T_{24}^{4} T_{12}^{4} T_{13}^{2} T_{14}^{2} T_{15}^{2} T_{15}^{64} \wedge \\
\wedge {} & T_{17}^{64} T_{18}^{4} T_{19}^{4} T_{20}^{64} T_{21}^{8} T_{22}^{8} T_{23}^{8} \vee \\
\vee {} & T_{11}^{4} T_4^1 T_{28}^{12} T_{27}^{2} T_{22}^{2} T_{25}^{2} T_{24}^{4} T_{12}^{4} T_{13}^{2} T_{14}^{2} T_{15}^{2} T_{15}^{64} \wedge \\
\wedge {} & T_{17}^{64} T_{18}^{4} T_{19}^{4} T_{20}^{64} T_{21}^{8} T_{22}^{8} T_{23}^{8}.
\end{aligned}
$$

Для інших вершин аналогічно виконується обчислення ДНФ функцій керованості.

Приклади обчислення функцій спостереженості для окремих вершин мають такий вигляд:

$$
\begin{aligned}
R_2 = {} & T_{28}^{12} T_{27}^{2} T_{22}^{2} T_{25}^{2} T_{24}^{4} T_{23}^{8} T_{22}^{8} T_{21}^{8} T_{20}^{64} T_{19}^{4} T_{18}^{4} \wedge \\
\wedge {} & T_{17}^{64} T_{16}^{64} T_{15}^{2} T_{14}^{2} T_{13}^{2} T_{12}^{4} T_{11}^{4} T_{10}^{4} T_9^8 T_8^8.
\end{aligned}
$$

$$
\begin{aligned}
B_1 = {} & T_{28}^{12} T_{27}^{2} T_{22}^{2} T_{25}^{2} T_{24}^{4} T_{23}^{8} T_{22}^{8} T_{21}^{8} T_{20}^{64} T_{19}^{4} T_{18}^{4} \wedge \\
\wedge {} & T_{17}^{64} T_{16}^{64} T_{15}^{2} T_{14}^{2} T_{13}^{2} T_{12}^{4} T_{11}^{4} T_{10}^{4} T_9^8 T_8^8 T_2^{64}.
\end{aligned}
$$

$$L_1 = T_{28}^{12} T_{27}^{2} T_{22}^{2} T_{25}^{2} T_{24}^{4} T_{23}^{8} T_{22}^{8} T_{21}^{8} T_{20}^{64} \wedge$$
$$\wedge T_{19}^{4} T_{18}^{4} T_{17}^{64} T_{16}^{64} T_{15}^{2} T_{14}^{2} T_{7}^{4} T_{1}^{64}.$$

Синтезовані логічні функції задають всі можливі шляхи управління, як в часі, так і в просторі, що можна вважати новою аналітичної формою опису тестопридатності проєкту. За ДНФ, слідуючи виразами для підрахунку тестопридатності, можна визначити критерії керованості (спостережливості) для всіх компонентів HDL-моделі.

Тут слід розглянути для варіанта (сценарію) обрахунку програмної моделі:

- враховується тільки графова структура, де вага кожної дуги дорівнює 1, незалежно від числа транзакцій в програмному коді;
- усі дуги графа відзначаються реальною кількістю транзакцій, що мають місце бути між двома розглянутими вершинами-компонентами транзакційного графа.

Оцінки тестопридатності описаних процедур можуть істотно відрізнятися один від одного. Користувач повинен визначитися, що важливіше: якщо тільки структура програмного коду – застосувати перший сценарій, або мати більш складну і точну модель транзакцій, розподілених у часі, на безлічі графових компонентів. Як приклад нижче наводиться процедура обчислення керованості для вершини B_2:

$$U(B_2) = \frac{1}{22 \times 6} \times (6 + 6 + 19 + 22 + 19 + 19) = 0{,}54$$

Застосування аналогічних обчислень керованості (спостережливості) для інших вершин графа дає результат у вигляді графіка, представленого на рис. 18, які дозволяють визначити критичні точки для установки необхідних асерцій.

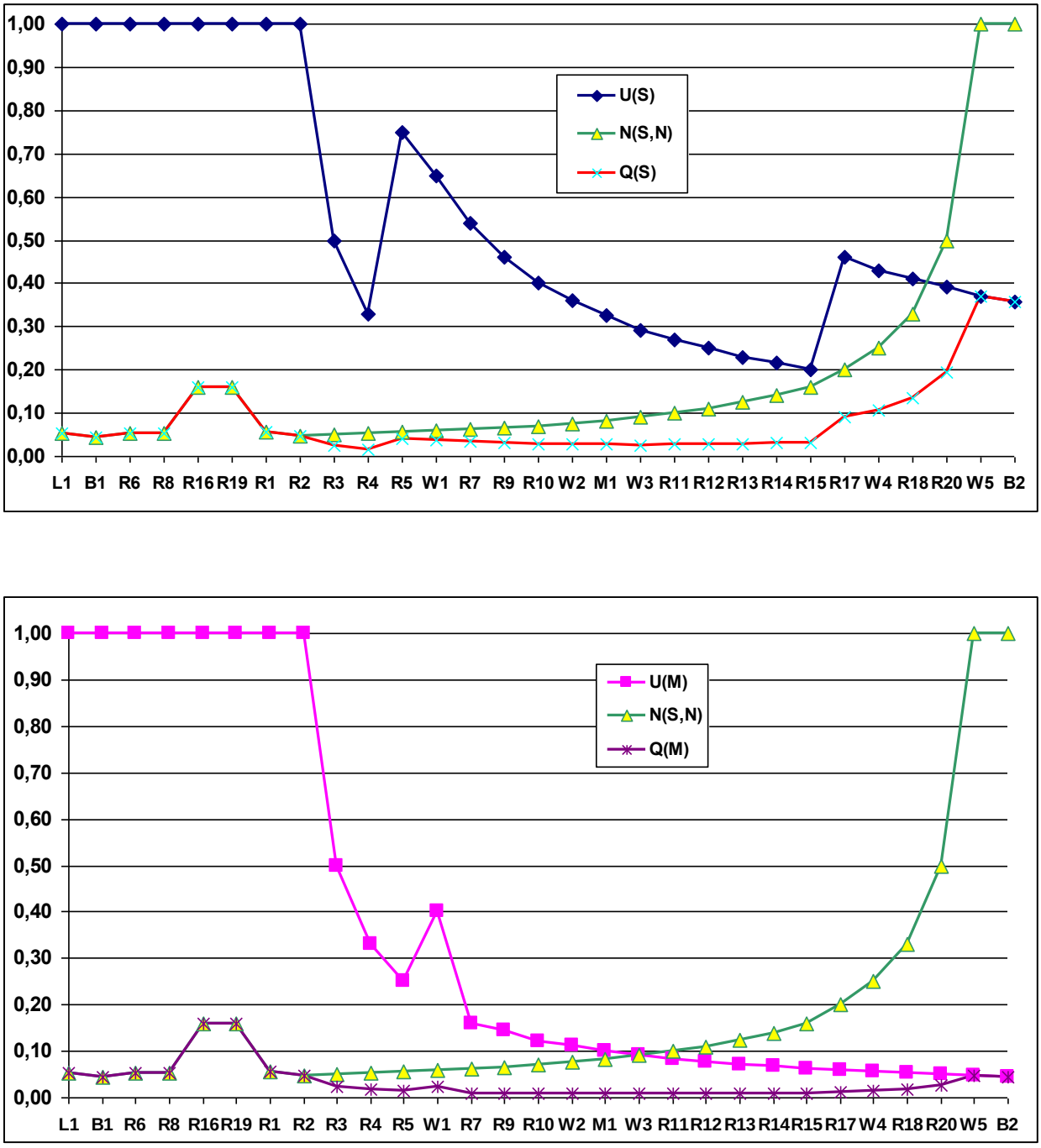


Рисунок 18 – Графіки *M*-тестопридатності Xilinx моделі

Такою вершиною може бути компонент R_{15}, якщо транзакційний граф представлений поодинокими дугами. Для випадку, коли дуги відзначені реальною кількістю транзакцій, критичні вершини належать компонентам, які перебувають ближче до вихідної шини. Тут істотним представляється не структура графа, а вага вхідної дуги, який більшою мірою впливає, якщо

структурна глибина розглянутого компонента досить висока. Використовується формула обчислення тестопридатності з мультиплікативними членами $U_i \times N_i$, що дає оцінку нижче, ніж будь-який із співмножників (керованість, спостережливість).

Якщо модифікувати формулу обчислення тестопридатності для компонентів до наступного вигляду:

$$Q_i = U_i + N_i,$$

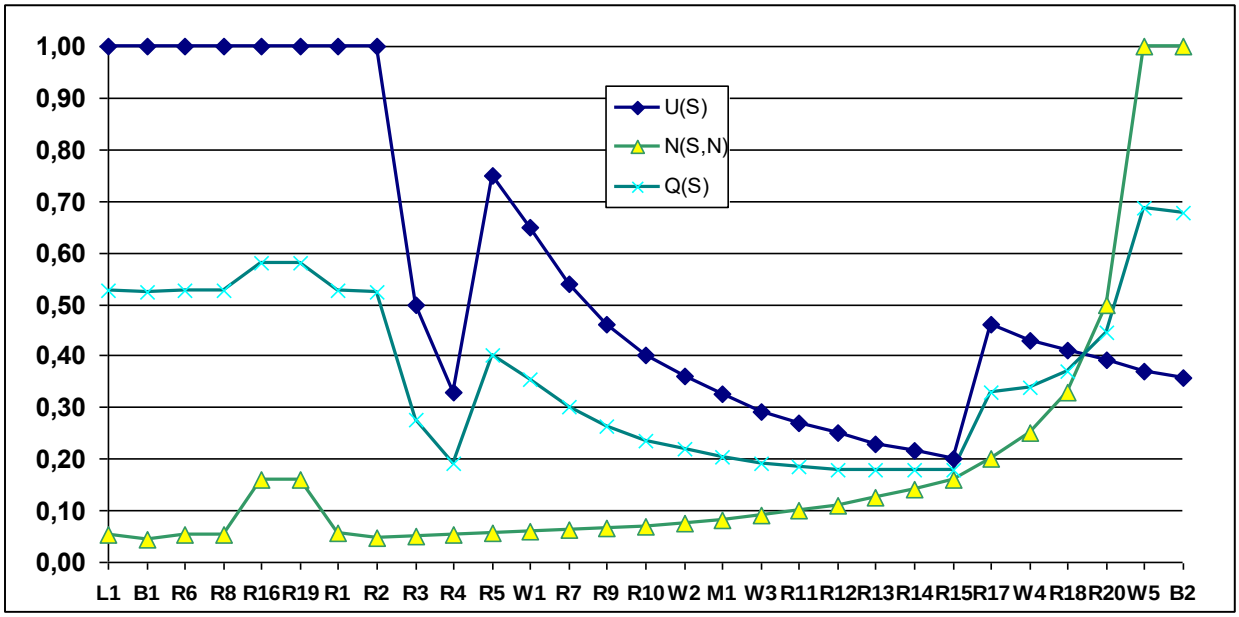
то крива тестопридатності істотно підніметься вгору по осі ординат, чим забезпечується менший розкид параметрів для кожної вершини. Дана обставина фіксує дещо відмінні графіки, представлені нижче (рис. 19).

Цікавим видається поведінка окремих вершин. Наприклад, керованість вершини R_{17} в мультипликативном транзакційному графі HDL-коду несподівано «впала» вниз в порівнянні з графом одиничних дуг. Це пов'язано з високою вагою транзакцій, що надходять на розглянуту вершину з боку вхідних компонентів L_1, B_1, які практично перетворюють на нуль значимість одиничних транзакцій від вершин R_{16}, R_{19}.

Після визначення спостереження та управління вершин транзакційного графа виконується підрахунок узагальненого критерію тестопридатності кожного компонента програмного коду. Потім визначається інтегральна оцінка тестопридатності проєкту за формулою:

$$Q = \frac{1}{n} \sum_{i=1}^{n} (U_i \times N_i),$$

яка визначає якість проєктного варіанту, що є досить суттєвим при порівнянні декількох альтернативних рішень.



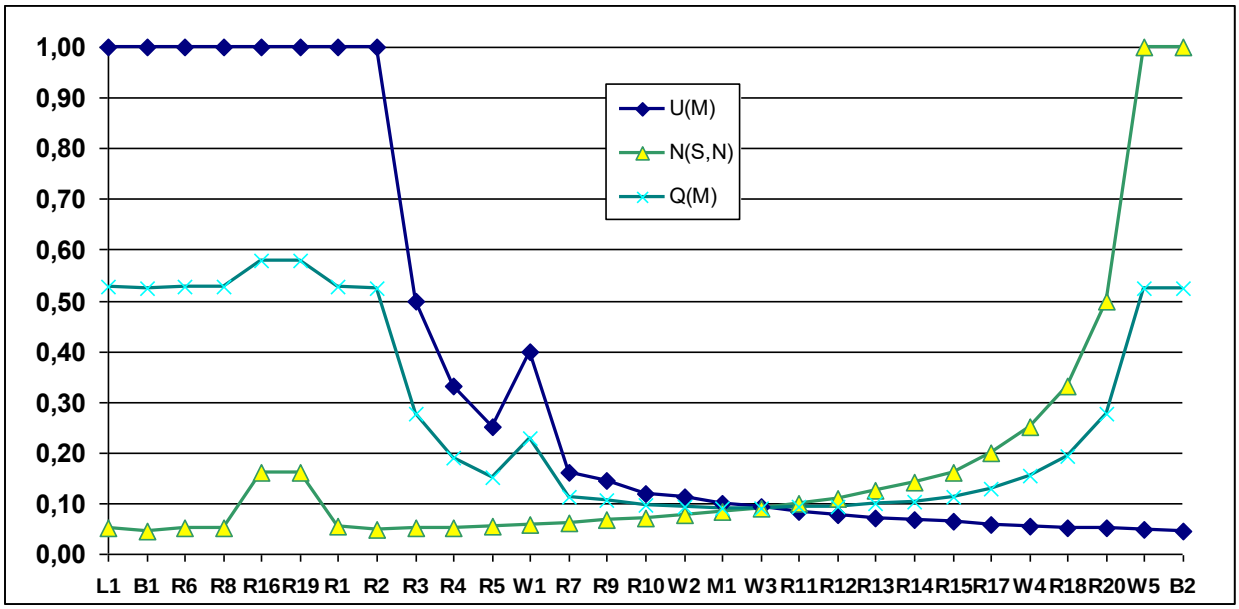


Рисунок 19 – Графіки *A*-тестопридатності Xilinx моделі

Як приклад позитивного використання розроблених моделей і методів пропонується аналіз тестопридатності програмного коду дискретного косинусного перетворення з Xilinx бібліотеки.

Було виконано:

– побудову транзакційної моделі;

– підрахунок характеристик тестопридатності ($Q = \{ 0{,}382; 0{,}287 \}$);

– визначення критичних точок.

Потім, відповідно до числа і типів компонентів, було розроблено функціональне покриття, фрагмент якого представлений лістингом 2.

Лістинг 2

```
c0: coverpoint xin
    {
      bins minus_big={[128:235]};
      bins minus_sm={[236:255]};
      bins plus_big={[21:127]};
      bins plus_sm={[1:20]};
      bins zero={0};
      }
  c1: coverpoint dct_2d
      {
      bins minus_big={[128:235]};
      bins minus_sm={[236:255]};
      bins plus_big={[21:127]};
      bins plus_sm={[1:20]};
      bins zero={0};
      bins zero2=(0=>0);
      }
endgroup
```

Для критичних точок, визначених у результаті аналізу тестопридатності транзакційного графа, розроблена асерційна модель перевірки основних характеристик дискретного косинусного перетворення. Істотний фрагмент коду механізму асерцій представлений лістингом 3.

Лістинг 3

```
sequence first( reg[7:0] a, reg[7:0]b);
     reg[7:0] d;
     (!RST,d=a)
      ##7 (b==d);
   endsequence
  property f(a,b);
     @(posedge CLK)
       // disable iff(RST||$isunknown(a)) first(a,b);
         !RST |=> first(a,b);
  endproperty
    odin:assert property (f(xin,xa7_in))
     // $display("Very good");
    else  $error("The  end,  xin  =%b,xa7_in=%b",  $past(xin,
7),xa7_in);
```

В результаті проведеної верифікації дискретного косинусного перетворення в середовищі Questa, Mentor Graphics були знайдені неточності у восьми рядках вихідного коду HDL-моделі:

```
//  add_sub1a <= xa7_reg + xa0_reg;//
```

Подальше виправлення помилок привело до появи виправленого фрагмента коду, який наведений у лістингу 4.

Лістинг 4.

```
add_sub1a <= ({xa7_reg[8],xa7_reg} + {xa0_reg[8],xa0_reg});
add_sub2a <= ({xa6_reg[8],xa6_reg} +{xa1_reg[8],xa1_reg});
add_sub3a <= ({xa5_reg[8],xa5_reg} +{xa2_reg[8],xa2_reg});
add_sub4a <= ({xa4_reg[8],xa4_reg} + {xa3_reg[8],xa3_reg});
end
else if (toggleA == 1'b0)
begin
add_sub1a <= ({xa7_reg[8],xa7_reg} - {xa0_reg[8],xa0_reg});
add_sub2a <= ({xa6_reg[8],xa6_reg} - {xa1_reg[8],xa1_reg});
add_sub3a <= ({xa5_reg[8],xa5_reg} - {xa2_reg[8],xa2_reg});
add_sub4a <= ({xa4_reg[8],xa4_reg} - {xa3_reg[8],xa3_reg}).
```

ВИСНОВКИ

У процесі виконання магістерської кваліфікаційної роботи, була досягнута мета роботи, а саме: змодельована оцінка технологій верифікації цифрових систем для діагностування і виправлення помилок HDL-моделей, шляхом спільного використання механізму асерцій і технологій тестопридатного проєктування.

Для досягнення мети роботи, були виконані наступні теоретичні етапи:

- дана загальна характеристика системи на кристалі;
- окреслено сучасні тенденції розвитку;
- приведена номенклатура продукції пам'яті на кристалі, що випускається;

При розробці компонентів інфраструктури верифікації та сервісного обслуговування SoC-пам'яті, відповідно до принципів створення сервісних систем, були отримані такі результати:

- сформована система сервісної ідентифікації SoC-пам'яті;
- складено інструкції при сервісному обслуговуванні SoC-пам'яті;
- розроблена система збереження параметрів сервісного обслуговування;
- систематизовані складові оцінки програмування SoC-пам'яті, що включають особливості безконтактних мікросхем пам'яті і особливості супервізорів;

Також, в роботі наведені вимоги до програмного забезпечення, що створюється стосовно завдань сервісного обслуговування систем SoC-пам'яті ГАС.

Проведені у поданій магістерській кваліфікаційній роботі дослідження, дозволяють зробити висновки щодо отриманих наукових результатів інноваційних технологій тестопридатного проєктування програмних і апаратних продуктів.

1. Показані основні напрямки використання технологій тестопридатного проєктування цифрових систем на кристалах в задачах тестування і верифікації програмних продуктів.

2. Представлена універсальна модель програмного компонента у вигляді транзакційного графа, на якому можна вирішувати завдання аналізу тестопридатності з метою досягнення необхідної глибини діагностування HDL-коду.

3. Запропоновано логічні функції тестопридатності HDL-моделей, на основі використання транзакційного графа з метою обчислення оцінок тестопридатності (керованість і спостережливість) програмних компонентів і всього HDL-проєкту в цілому.

4. Наведені приклади і графіки оцінювання тестопридатності (спостереження та управління) програмних моделей, представлених фрагментами транзакційних графів.

5. Запропоновано комплекс технологічних заходів і рекомендацій, орієнтованих на метрику тестопридатності і подальший синтез програмних продуктів, придатних для тестування і верифікації.

6. Показані приклади аналізу тестопридатності шляхом розрахунку спостереження та управління транзакційним та керуючим графами, з метою визначення критичних точок з подальшим вирішенням практичної проблеми виявлення і усунення більшості несправностей в реальному проєкті від компанії Xilinx.

7. Побудовано логічні функції керованості, спостережливості для двох реальних прикладів і графіків; відповідні їм, критичні точки, які дають можливість проаналізувати шляхи подальшого поліпшення коду проєкту шляхом установки асерцій.

8. Практична значимість запропонованих метрик і моделей полягає в ринковій привабливості і високій зацікавленості технологічних компаній в інноваційних рішеннях проблеми ефективного тестування та верифікації програмно-апаратних проєктів на системному рівні проєктування, з метою

зменшення параметру «time-to market» і підвищення виходу придатної продукції – «yield».

9. Подальші дослідження будуть спрямовані на розробку стандартних інтерфейсів, з метою подальшої інтеграції моделей, методів і програмних засобів в сформовані технологічні маршрути проєктування цифрових систем на кристалах.

З усього перерахованого вище можна зробити висновки, що стандартні вироби класу SoC забезпечують комбінацію гнучкості проєктування і швидкості сервісного обслуговування.

Очевидно, що системи на кристалі, що базуються на ASIC, є, в даний час, найбільш раціональним рішенням в якості вбудованої пам'яті ГАС. Будь-яка реалізована SoC-пам'ять, дешевше рішення, що програмується. Однак, цикл проєктування виробів цього класу складний і тривалий, вартість розробки та верифікації проєктів залишається високою.

Незважаючи на це, SoC-пам'ять є рішенням загальної проблеми отримання малогабаритного, функціонально насиченого, універсального класу мікросхем, які крім усього повинні реконфігуруватися та досить просто обслуговуватися.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Хаханов В. И., Хаханова А. В., Литвинова Е. И. Алгебро-логический метод ремонта встроенной памяти SoC. Відмовостійкі системи. 2018. С. 99–109.
2. Проєктирование и диагностика компьютерных систем и сетей. Бондаренко М. Ф., Кривуля Г. Ф., Рябцев В. Г. Киев: НМЦ ВО, 2010. 306 с.
3. Хаханов В. И., Хаханова И. В. VHDL + Verilog = Синтез за минуты. Харьков: СМИТ, 2017. 264 с.
4. Парфентий А. Н., Хаханов В. И., Литвинова Е. И. Модели инфраструктуры сервисного обслуживания цифровых систем на кристаллах. АСУ и приборы автоматики. 2017. Вып. 138. С. 83–99.
5. Золотухо Р. System Designer – пакет для разработки устройств на основе FPSLIC. Chip News. 2011. №2. С. 8–14.
6. Золотухо Р., Кривченко И. Конфигурируемая система на кристалле E5 – первое знакомство. Компоненты и технологии. 2011. №1. С. 26–29.
7. Программируемые приборы класса "система-на-кристалле" для встраиваемых применений. Компоненты и технологии. 2001. №2. С. 13.
8. Великодний С. С., Бурлаченко Ж. В., Зайцева-Великодна С. С. Реінжиніринг графічних баз даних у середовищі відкритої системи автоматизованого проєктування BRL-CAD. Моделювання структурної частини. Вісник Кременчуцького національного університету ім. Михайла Остроградського. 2019. Вип. 3 (116). С. 130–139. (кат. «Б») DOI: 10.30929/1995-0519.2019.3.130-139.
9. Великодний С. С., Бурлаченко Ж. В., Зайцева-Великодна С. С. Реінжиніринг графічних баз даних у середовищі відкритої системи автоматизованого проєктування BRL-CAD. Моделювання поведінкової частини. Вісник Кременчуцького національного університету ім. Михайла Остроградського. 2019. Вип. 2 (115). С. 117–126. (кат. «Б») DOI: 10.30929/1995-0519.2019.2.117-126.

10. Великодный С. С. Методологические основы реинжиниринга систем автоматизированного проєктирования. Управляющие системы и машины. 2014. № 2. С. 39–43. Видання національної академії наук України.
11. Хаханов В. И. Инфраструктура сервисного обслуживания SoC. Вестник томского государственного университета. №4. 2008. С. 74–101.
12. Великодный С. С. Проблема реинжиниринга видов обеспечения систем автоматизированного проєктирования. Управляющие системы и машины. 2014. № 1. С. 57–61, 76. Видання національної академії наук України.
13. Великодний С. С. Реінжиніринг систем моніторингу та дистанційного управління судновими енергетичними установками. Матер. XXII міжн. конф. з автом. управл. «Автоматика 2015», 10–11 вер. 2015, Одеса, 2015. С. 133–134.
14. Hahanov V., Kteaman H., Ghribi W., Fomina E. HEDEFS – Hardware embedded deductive fault simulation. Proc. volume from the 3-rd IFAC Workshop, Rydzyna, Poland, 2016. P. 25–29.
15. Hahanov V., Kteaman H., Ghribi W., Fomina E. HEDEFS – Hardware embedded deductive fault simulation. Proc. volume from the 3-rd IFAC Workshop, Rydzyna, Poland, 2006. P. 25–29.
16. Федотов Я., Щука А. Система на кристалле. Электронные компоненты. 2011. №2. С. 3–5.
17. Mixed-Signal ASIC Solutions. AMI Press, 2010. 86 p.
18. System-on-Chip. micron AG Press, 2010. 104 p.
19. Analog and Mixed-Signal ASICs. PREMA Semiconductor Press, 2010. 208 p.
20. ASIC: Парад технологий. Переклад. О. Александрова. Chip News. 2012. №9. С. 8–11.
21. Кривченко И. Системная интеграция в микроэлектронике FPSLIC. Chip News. 2012. №3. С. 4 –10.

22. Кривченко И. Системная интеграция в микроэлектронике FPSLIC. Часть 2: FPSLIC – вопросы и ответы. Chip News. 2012. №4. С. 62–64.
23. Королев Н. ATMEL FPSLIC – элементная база XXI века. Chip News. 2011. №1. С. 16–19.
24. Ajay Khoche. System-in-Package is Coming to Consumer Products: Is Test Ready? Proceedings of the International Test Conference 2017 (ITC'17). 2017. pp. 1166.