Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет комп’ютерної інженерії та управління
(повна назва)

Кафедра електронних обчислювальних машин
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

Рівень вищої освіти другий (магістерський)

Методи розробки спеціалізованих інтегральних схем для систем управління

(тема)

Виконав:

студент II курсу, групи СПм-22-5

Лубан А.С.
(прізвище, ініціали)

Спеціальність

123 «Комп’ютерна інженерія»
(код і повна назва спеціальності)

Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма

Системне програмування
(повна назва освітньої програми)

Керівник: доц. Федорченко В.М.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри ЕОМ _____ Коваленко А.А.
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет комп’ютерної інженерії та управління

Кафедра електронних обчислювальних машин

Рівень вищої освіти другий (магістерський)

Спеціальність 123 «Комп’ютерна інженерія»
(код і повна назва)

Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма Системне програмування
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
“____” _____ 20___ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студенту Лубану Артему Сергійовичу
(прізвище, ім’я, по батькові)

1. Тема роботи Методи розробки спеціалізованих інтегральних схем для систем управління

затверджена наказом по університету від “ 01 ” квітня 2024 р. № 257 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 15 червня 2024 р.

3. Вхідні дані до роботи
інтегральна схема
специфікація
обчислювальний комплекс
система-на-кристалі

4. Перелік питань, що потрібно опрацювати у роботі
Специфіка проектування систем-на-кристалі та спеціалізованих інтегральних схем
Особливості проектуваннясистем-на-кристалі та мови опису апаратних засобів
Метод створення архітектурних специфікацій

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) 15 слайдів

6. Консультанти розділів роботи (заповнюється за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Отримання завдання та аналіз літератури	01.04.2024 – 06.04.2024	
2	Огляд існуючих рішень та методів	07.04.2024 – 12.04.2024	
3	Розробка методу	13.04.2024 – 18.04.2024	
4	Вибір програмних засобів	19.04.2024 – 25.04.2024	
5	Програмна реалізація	26.04.2024 – 02.05.2024	
6	Аналіз отриманих результатів	03.05.2024 – 16.05.2024	
7	Оформлення записки	17.05.2024 – 14.06.2024	
8	Представлення роботи в ЕК	15.06.2024	

Дата видачі завдання 01 квітня 2024 р.

Студент _____
(підпис)

Керівник роботи _____ доц. Федорченко В.М.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 58 с., 14 рис., 2 дод., 20 джерел.

СИСТЕМА НА КРИСТАЛІ, ОБЧИСЛЮВАЛЬНА СИСТЕМА, СПЕЦИФІКАЦІЯ, VHDL, АРХІТЕКТУРА, МЕТОД, ІНТЕГРАЛЬНА СХЕМА.

Метою роботи є аналіз методів розробки спеціалізованих інтегральних схем для систем управління з використанням архітектурних специфікацій систем-на-кристалі архітектурних специфікацій.

У ході виконання кваліфікаційної проведено аналіз методів розробки спеціалізованих інтегральних схем для систем управління з використанням архітектурних специфікацій систем-на-кристалі архітектурних специфікацій. Для спрощення створення виконуваних моделей SoC у набір методів включений метод автоматизації переведення специфікацій у виконувані моделі на формальних мовах; а для раннього спільного аналізу різних компонентів системи та протоколів її функціонування розроблений підхід передбачає застосування різних методів верифікації даних на стадії створення специфікації. Крім того, розроблений метод також дозволяє підвищити якість взаємодії спеціалізованої інтегральної плати та системи на кристалі.

ABSTRACT

Master’s thesis: 58 pages, 14 figures, 2 appendices, 20 sources.

SYSTEM ON CRYSTAL, COMPUTING SYSTEM, SPECIFICATION, VHDL, ARCHITECTURE, METHOD, INTEGRATED CIRCUIT.

The major goal is to analyze the methods of developing specialized integrated circuits for control systems using the architectural specifications of systems-on-a-crystal architectural specifications.

In order to analysis analysis of the methods of developing specialized integrated circuits for control systems using the architectural specifications of systems-on-a-crystal architectural specifications was carried out. To simplify the creation of executable SoC models, the method of automating the translation of specifications into executable models in formal languages is included in the set of methods; and for the early joint analysis of various components of the system and the protocols of its operation, the developed approach involves the use of various data verification methods at the stage of creating the specification. In addition, the developed method also allows to improve the quality of the interaction between the specialized integrated circuit board and the system on the crystal.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

RTL – рівень регістрових передач

UML – уніфікована мова моделювання

НВІС – надто велика інтегральна схема

ОС – обчислювальна система

СНК – система на кристалі (SOC)

ВСТУП

Першим кроком розробки методів створення специфікацій має бути вибір форматів даних, які використовуватимуться у межах цих методів. Критерії вибору таких форматів серед безлічі використовуються на ранніх етапах проектування системи на кристалі (СНК) мають бути виведені з цілей та завдань, поставлених у роботі. Завдання підвищення якості специфікацій передбачає, що обрані формати повинні надавати можливість найповнішого, несуперечливого та недвозначного викладу даних. При цьому несуперечність (і частково повнота) може бути досягнута за допомогою механізму ранньої верифікації специфікацій системи.

Важливим доповненням до першого завдання, поставленого у роботі, є вимога відсутності суттєвих додаткових зусиль архітекторів розробки специфікації. З нього випливають відразу дві основні вимоги до форматів даних:

- з використанням цих форматів має бути зручно створювати документацію. Незважаючи на суб'єктивність оцінки цього критерію, він є одним із головних у процесі майбутнього аналізу. Не повинно бути значного бар'єру для початку роботи з вибраними форматами. Це означає щонайменше, що формати мають бути широко застосовні у процесі проектування архітектури обчислювальних систем та знайомі архітекторам;

- легкість розуміння специфікацій усіма споживачами – також передбачає відсутність бар'єрів як необхідності для проектувальників вивчати нові мови, що робить ще більш важливим застосовність обраних форматів у процесі розробки НВІС;

- наявність автоматичної можливості трансляції на формальні мови слідує аналогічний критерій вибору форматів: для кожного з них у літературі повинні бути описані засоби для такої трансляції. Варто зазначити, що часто подібні кошти не є повністю автоматичними: наприклад, для успішної

трансляції користувач повинен крім специфікації надати додаткові дані: словники, таблиці відповідності структур даних, списки правил тощо далі. Якщо для якогось формату відомі лише такі транслятори, то будемо вважати, що він частково задовольняє критерію можливості трансляції. Перелічені критерії випливають із цілей, поставлених у роботі. Однак можна назвати ще один критерій вибору форматів, вкрай важливий, але не наступний з цілей роботи, а мається на увазі неявно. Так як розглядаються всі формати, що використовуються при проектуванні апаратно-програмних комплексів, а набір методів розробляється спеціалізовано для систем на кристалі, необхідно також, щоб ці формати добре підходили для опису як мінімум кількох аспектів архітектури СНК.

Метою роботи є аналіз методів створення архітектурних специфікацій обчислювальних комплексів на базі SoC.

Об’єктом дослідження є методи забезпечення функціонування обчислювальних комплексів на SoC.

Завдання:

- проведення аналізу існуючих методів створення архітектурних специфікацій обчислювальних комплексів на базі SoC;

- проведення порівняння різних форматів архітектурних специфікацій SoC;

- розробка методу створення та використання архітектурних специфікацій обчислювальних комплексів на базі SoC.

1 СПЕЦИФІКА ПРОЕКТУВАННЯ СИСТЕМ-НА-КРИСТАЛІ ТА СПЕЦІАЛІЗОВАНИХ ІНТЕГРАЛЬНИХ СХЕМ

1.1 Загальні відомості

Складність сучасних систем на кристалі призводить до безлічі проблем в процесі їх проектування. Незважаючи на те, що традиційний процес розробки НВІС в застосуванні до СНК в останні десятиліття зазнав істотних змін, компанії-виробники программноаппаратних комплексів досі стикаються з серйозними труднощами при розробці таких систем. Особливої уваги заслуговують проблеми перших етапів проектування РНК, пов'язані з розробкою специфікації, проектуванням архітектурних моделей і ранньої верифікацією систем. Метою даної частини роботи є виявлення таких проблем на ранніх етапах проектування СНК.

СНК – обчислювальна система, на єдиному кристалі якої інтегровані всі основні необхідні елементи, список яких зазвичай включає в себе [1]:

- обчислювальні елементи, які можуть являти собою як мікропроцесори загального призначення, так і вузькоспеціалізовані обчислювальні елементи;
- модулі оперативної пам'яті;
- шини: центральна (для високошвидкісного обміну інформацією між процесором, оперативною пам'яттю і іншими блоками, які працюють на високих частотах) і периферійна (для обміну з іншими пристроями);
- контролер переривань;
- контролери введення і виведення інформації.

Схема типової системи на кристалі для мобільного сегменту, WonderMedia PRIZM WM8950 [2], представлена на рисунку 1.1.

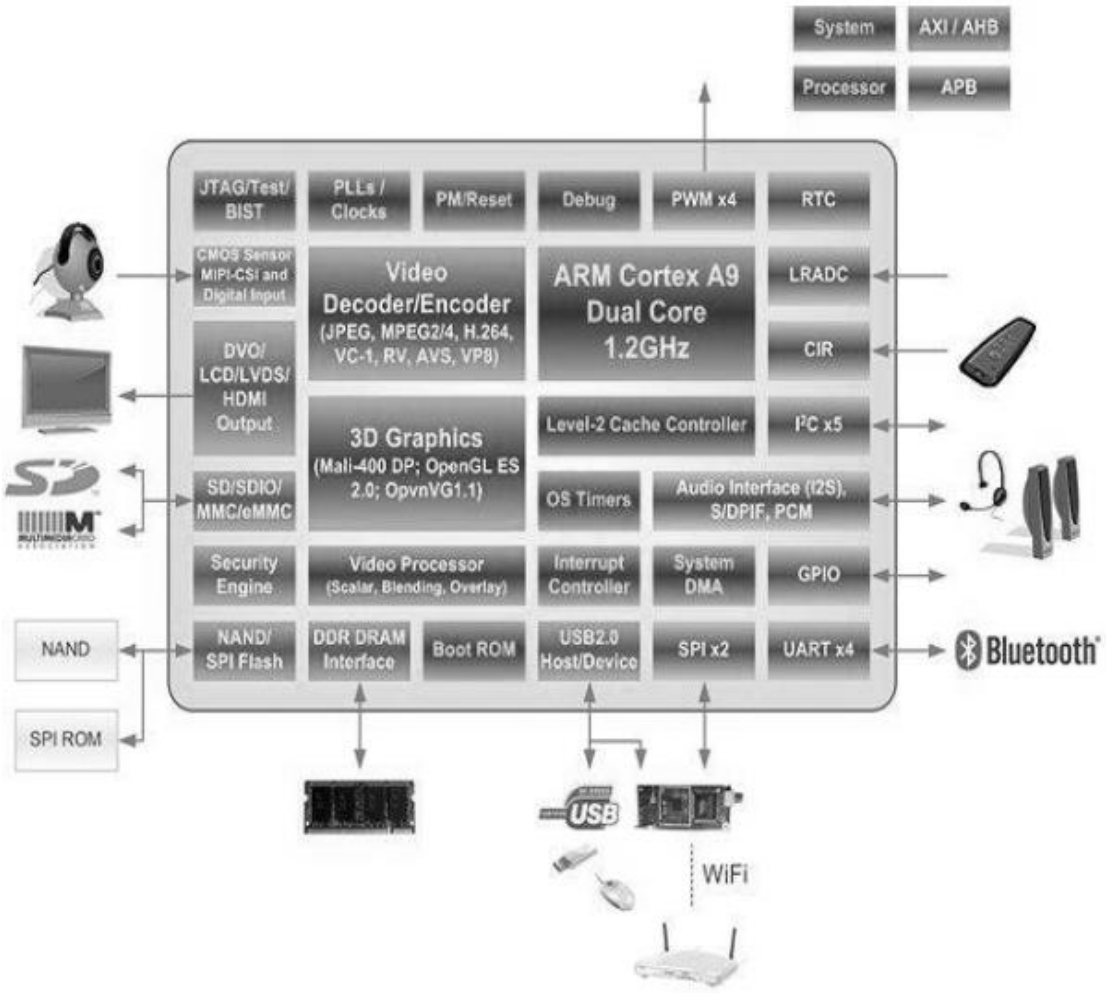

Рисунок 1.1 – Система на кристалі для мобільного сегменту

Сучасні системи на кристалі включають в себе десятки так званих IP-блоків, готових складних функціональних компонентів для проектування мікросхем. Перевагою сучасного підходу до проектування СНК є можливість перевикористати різні IP-блоки в рамках різних платформ. При визначенні поняття «система на кристалі» потрібно відзначити ще одну важливу особливість СНК: принципову роль програмно-апаратних засобів в їх конструкції. Це пояснюється двома причинами. По-перше, функціональність сучасних систем занадто широка, і покрити всі функції системи апаратної реалізацією досить складно. По-друге, у зв'язку з мінливістю вимог до системи при проектуванні часто буває простіше використовувати

програмовані апаратні блоки, які в разі змін в проекті перепрограмувати набагато простіше, ніж створити нову апаратну частину. Розбиття функціональності системи на апаратну і програмну проводиться зазвичай з міркувань продуктивності: чим вище вимоги до продуктивності певних частин СНК, тим краще реалізувати їх у вигляді апаратних, а не програмних блоків. Розглянемо тепер традиційний процес проектування інтегральних схем і його зміни, що відбулися в зв'язку з суттєвим ускладненням проектів.

1.2 Класичний підхід до розробки СНК

Раніше протягом десятиліть проектування HBIC велося по класичній каскадніій моделі [1]. Класичний процес проектування обчислювальної системи може бути формально розділений на кілька стадій: планування, розробка специфікації архітектури, розробка мікроархітектури, опис системи на рівні регістрових передач (RTL), а також логічний і фізичний синтез. Стадія планування відноситься більшою мірою до загальної стратегії компанії, ніж до конкретного проекту. На цій стадії розробляються і визначаються основні технології, які будуть використовуватися при проектуванні обчислювальних систем, і вирішується, які кінцеві продукти будуть розроблятися.

На стадії розробки специфікації архітектури визначаються основні вимоги до системи, її майбутні характеристики і основні функціональні блоки. На стадії проектування мікроархітектури розробляються основні елементи кожного функціонального блоку з урахуванням вимог до площі кристала, енергоспоживанню і інших.

На наступному етапі створюється RTL-код апаратних блоків системи на мовах опису апаратури, таких як Verilog або VHDL. RTL-код детально описує потоки сигналів між апаратними регістрами і логічні операції над даними. Після розробки RTL-коду в класичному процесі починається функціональна верифікація системи.

Подальші етапи включають в себе планування топології кристала, логічний синтез, результатом якого є опис системи на вентильному рівні у вигляді списку ланцюгів, проектування систем харчування і синхронізації, тимчасову верифікацію системи, а також фізичний синтез - розміщення елементів на схемі і трасування зв'язків між ними. Після фізичного синтезу і наступної за ним фізичної верифікації слід створення фізичного прототипу НВІС і його тестування.

Загальна схема традиційного каскадного процесу розробки СНК приведена на рисунку 1.2. З ускладненням розроблюваних НВІС виникла проблема: традиційний каскадний процес не може забезпечити достатній рівень покриття верифікації системи. Це пояснюється тим, що в рамках традиційного процесу проектування більшість дій з валідації систем проводилися вже після створення RTL-коду.

Однією з головних причин цього була відсутність повних і завершених моделей навіть окремих аспектів структури і поведінки системи [3]. Однак опис на рівні регістрових передач є занадто низькорівневим для того, щоб в короткі терміни провести повну валідацію системи, що містить сотні мільйонів елементів. Тому з ускладненням проектованих обчислювальних систем стали прикладатися серйозні зусилля для того, щоб перейти на більш високий рівень абстракції для опису і верифікації систем.

Потрібно було створити нові способи формального опису системи, причому як апаратної її частини, так і програмного забезпечення, і розробити нову методологію створення і верифікації такого опису на етапі створення специфікацій архітектури та мікроархітектури системи. Потрібно було забезпечити можливість верифікації архітектури системи на ранніх етапах її розробки, адже, як відомо, помилки, допущені на ранніх етапах проектування, «стоять» компаніїразработчіку найдорожче [4-7].

Розробники НВІС зіткнулися з проблемою вибору нових мов опису системного рівня і методології проектування і верифікації з використанням цих мов. Будь-подібна мова системного рівня повинен був володіти двома

основними якостями: мати можливість об'єднувати безліч різнорідних моделей і працювати на високому рівні абстракції. В останні двадцять років багато можливостей були запропоновані для вирішення даного завдання [3]:

- використання мов опису апаратури з деякими доповненнями, як, наприклад, розширення мови Verilog до SystemVerilog;
- адаптація мов програмування високого рівня (C/C++, Java) для опису архітектурних специфікацій систем на кристалі;
- створення принципово нових мов системного рівня спеціально для проектування систем на кристалі. Даний підхід може привести до найкращих результатів, але робота в цьому напрямку вимагає величезних зусиль;
- розширення можливостей існуючих мов високорівневого моделювання та застосування їх разом з іншими мовами нижчого рівня для опису архітектурних специфікацій систем.

У сучасній практиці створення високорівневих моделей найбільш часто застосовуються мови, розроблені з використанням перших двох підходів, такі як SystemC і SystemVerilog.

Другою проблемою традиційного підходу є те, що з ускладненням проектованих ОС стало практично неможливо з першого разу отримати якісну і безпомилкову модель системи на кожному етапі процесу проектування [8], що призвело до необхідності використовувати ітерації на кожному такому етапі.

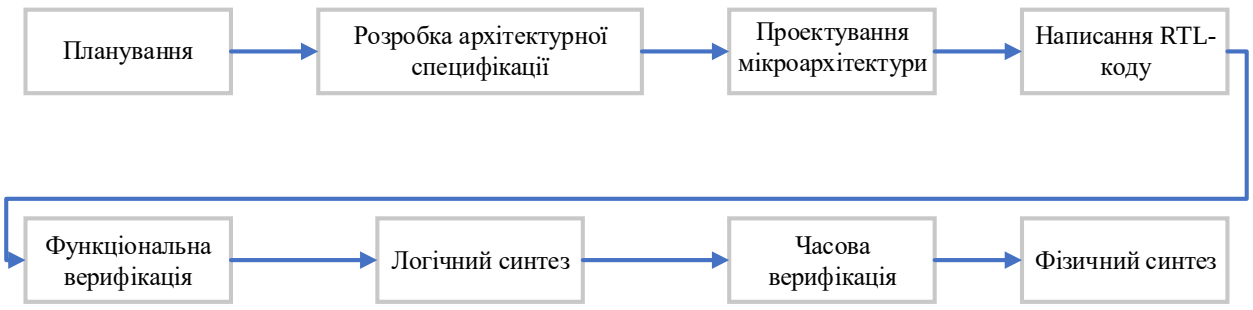

Рисунок 1.2 – Традиційний процес розробки ЗВІС

1.3 Актуальний стан проблеми

На перших кроках, як і в класичному процесі розробки, виявляються вимоги до майбутньої системи і створюється специфікація проекту – неформальна модель, що включає в себе опис структури системи, вимог до неї та алгоритмів її функціонування. Другий крок у різних варіаціях сучасного процесу проектування представлений по-різному. Передбачається, що на цьому етапі необхідно створити високорівневу модель системи на формальних мовах, провести верифікацію цієї моделі і за допомогою аналізу продуктивності визначити, які компоненти будуть реалізовані на апаратному рівні, а які – на програмному. Однак практика показує, що повністю реалізувати цей крок за розумний час можливо тільки для невеликих СНК, наприклад, у вбудованих системах. Для складних СНК загального призначення на другому кроці визначається тільки розбиття функцій між апаратним та програмним забезпеченням, а подальше створення моделей і їх верифікація проводиться паралельно з подальшими кроками розробки.

На цьому етапі розробники системи, застосовуючи ітеративний підхід, оцінюють продуктивність системи в різних ситуаціях і знаходять оптимальний розподіл функцій апаратного і програмного забезпечення. Модель СНК, що отримується на цьому етапі, є по суті абстрактної платформою і складається з абстрактних моделей апаратного і програмного забезпечення. Наприклад, модель програмного забезпечення може точно імітувати найбільш часто використовувану функціональність, виконання операцій введення / виводу і інші найважливіші операції.

Абстрактна модель апаратної частини платформи може бути описана на поведінковому рівні або на рівні транзакцій. Результатом цього етапу проектування є золота модель архітектури, в подальшому використовується в ролі абсолютного стандарту, на відповідність яким перевіряється кожен рівень опису системи. Ця золота модель може бути як успадкованою від існуючої СНК, так і створена з нуля. Після того, як в золотий моделі

архітектури визначено розбиття задач між апаратної і програмної частинами системи, розробка апаратних і програмних блоків може йти паралельно.

Крім цього, паралельно з розробкою IP-блоків йде процес фізичного проектування майбутньої СНК. Це пов'язано з тим, що при використанні сучасних технічних процесів вже на стадії кодування RTL необхідно мати деякі відомості щодо використовуваної технології, бібліотек, а також про попередньої плануванні кристала. Наступні кроки процесу проектування можна умовно розбити на три паралельні процеси: розробка апаратних блоків СНК, розробка програмно-апаратних засобів та фізичне проектування [11].

Незважаючи на паралельне виконання цих процесів, вони взаємопов'язані, і результати певних етапів одних є вхідними даними для деяких етапів інших. Розробка апаратних блоків СНК починається з ієрархічного тимчасового планування, після чого, маючи остаточні дані по ієрархії апаратних блоків і вимогам до них, команда розробки приступає до створення RTL-коду цих блоків.

Потім, після готовності RTL коду, як і в традиційному процесі проектування, проводиться його верифікація. Для процесу, в якому до RTL була створена золота модель системи, найважливішою частиною верифікації RTL-коду є його перевірка на відповідність цієї моделі, в зворотному випадку золотий моделлю стає сам RTL-код. Після верифікації RTL проводиться логічний синтез, статичний часовий аналіз, фізичний синтез і так далі. Розробка програмно-апаратних засобів РНК зазвичай складається з розробки прототипу, його тестування на відсутність помилок і на відповідність золотий архітектурної моделі, розробки самого програмного забезпечення та його подальшого тестування. Після закінчення процесів проектування апаратних і програмних блоків настає стадія системної інтеграції, в ході якої ці блоки за допомогою описаних на фазі проектування архітектури інтерфейсів з'єднують з системою повідомлення блоків на

кристалі, після чого відбувається перевірка коректності спільної роботи розроблених блоків.

Нарешті, процес фізичного проектування починається з вибору технології виробництва плати і вибору або створення бібліотеки елементів. Наступним етапом фізичного проектування є початкове планування кристала і планування ланцюгів харчування і дерева синхронізації. Після цього, коли готовий RTL-код компонентів системи, виконується уточнення плану кристала з урахуванням нових даних. Нарешті, останнім етапом фізичного проектування, що виконується після фізичного синтезу, є фінальне розміщення елементів на кристалі і детальна трасування. Загальна схема сучасного процесу розробки СНК представлена на рисунку 1.3.

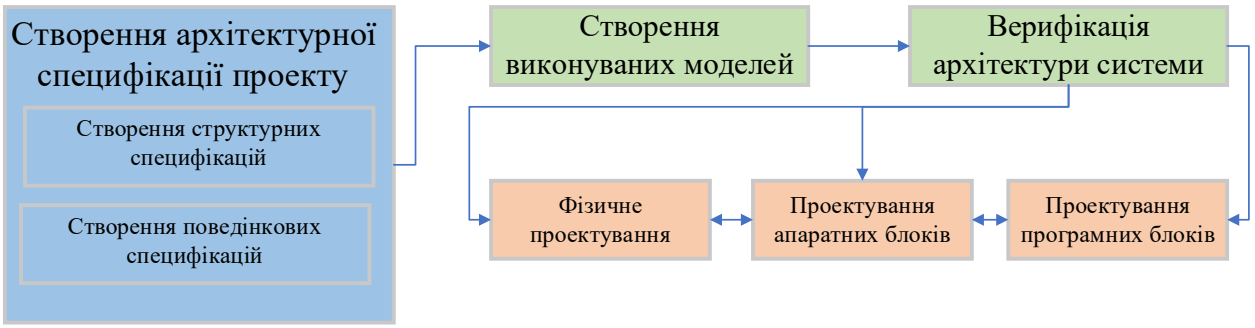

Рисунок 1.3 – Сучасний процес розробки ЗВІС

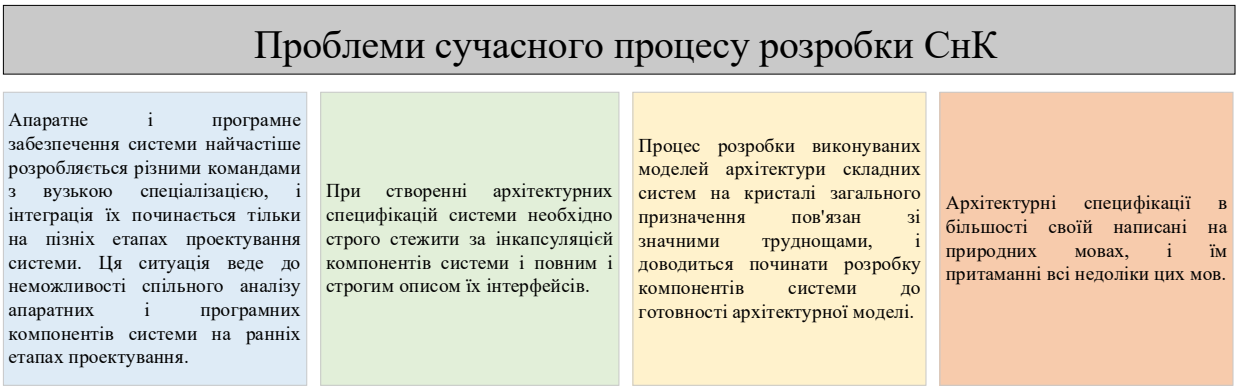

Рисунок 1.4 – Проблеми сучасного процесу розробки SoC

Важливо відзначити, що описаний процес називається спіральним через ітеративності кожного етапу проектування.

При складності, властивої сучасним проектам СНК, надзвичайно складно домогтися безпомилкового виконання навіть одного етапу проектування, тому кожен етап або кілька етапів проводять в декілька ітерацій для досягнення потрібного результату. Розмір таких циклів може бути різним.

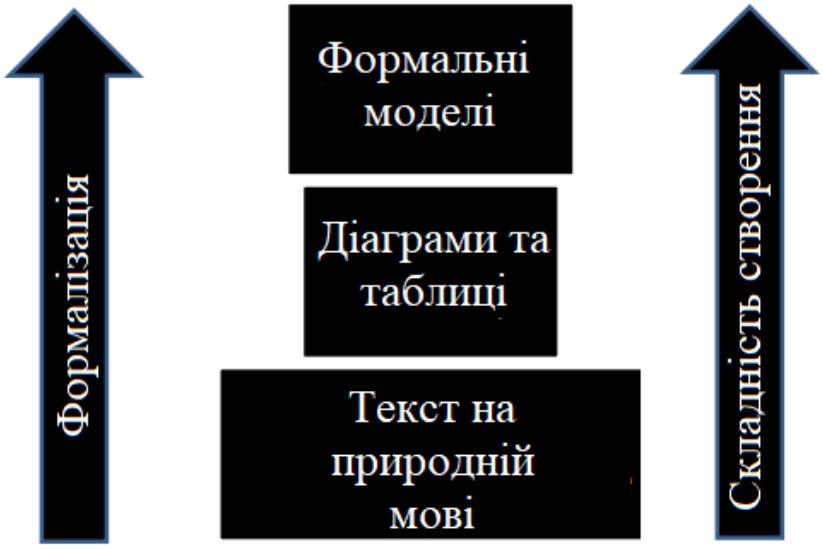

Рисунок 1.5 – Форми представлення специфікацій

Наприклад, зустрічаються ситуації, коли після розробки RTL-коду і подальшого планування розміщення компонентів на кристалі виникає необхідність змінити спроектовані набагато раніше ланцюга харчування і синхронізації, або після спроби інтеграції декількох блоків потрібно переписати RTL-код одного з них і заново провести для нього логічний і фізичний синтез.

Однак такі масштаби кожної ітерації значно збільшують загальний час розробки проекту, і в більшості випадків намагаються уникати помилок, здатних привести до появи таких циклів в процесі розробки [1].

1.4 Проблеми сучасного процесу розробки СнК

Процес розробки виконуваних моделей архітектури складних систем на кристалі загального призначення пов'язане зі значними труднощами, і доводиться починати розробку компонентів системи до готовності архітектурної моделі.

Архітектурні специфікації в більшості своїй написані на природних мовах, і їм притаманні всі недоліки цих мов.

Апаратне і програмне забезпечення системи найчастіше розробляється різними командами з вузькою спеціалізацією, і інтеграція їх починається тільки на пізніх етапах проектування системи. Ця ситуація веде до неможливості спільного аналізу апаратних і програмних компонентів системи на ранніх етапах проектування.

При створенні архітектурних специфікацій системи необхідно строго стежити за інкапсуляцією компонентів системи і повним і строгим описом їх інтерфейсів.

2 ОСОБЛИВОСТІ ПРОЕКТУВАННЯ SOC ТА МОВИ ОПИСУ АПАРАТНИХ ЗАСОБІВ

Процес розробки сучасних цифрових та змішаних систем неможливо уявити без широкого застосування інструментальних засобів автоматизованого проектування (САПР). Мови опису апаратури, які використовуються в САПР електроніки, можна розділити на дві групи: універсальні стандартизовані мови (VHDL, Verilog, SystemC) і предметно-орієнтовані (як правило, специфічні для кожної САПР - Altera HDL і т.д.).

Зростання вентильной і архітектурної складності розроблюваних систем, повсюдне використання технології проектування «систем-на-кристалі» і істотний рівень початкових вкладень при розробці ASIC-проектів вимагає проведення всебічного і великого етапу функціональної і тимчасової верифікації. При цьому розробка тестового забезпечення, як правило, досить серйозно обмежена в часі, що накладає певні вимоги до ефективності процесу моделювання. У будь-якому випадку, необхідно виконувати остаточну верифікацію саме RTL-коду Verilog або VHDL-описів, при цьому необхідно забезпечити ефективне моделювання тривалих тимчасових інтервалів (від декількох модельних секунд до хвилин).

Одним з варіантів підвищення швидкодії процесу моделювання HDL-описів є паралельне виконання окремих імітаційних моделей на робочих станціях під керуванням єдиного сервера, який здійснює синхронізацію приватних моделей і поширення інформації про системні події.

2.1 Формальні апарати уявлення динамічних систем

Динамічна система загального вигляду в класичному розумінні [1] являє собою систему типу «вхід-вихід», що володіє всього двома властивостями:

- значення вихідної величини в будь-який момент часу залежить від стану системи, під яким мається на увазі деяка частина інформації про теперішній стан і минуле системи;

- система повинна бути причинно-зв'язаною в тому сенсі, що безліч моментів часу ti впорядковано та що минуле впливає на майбутнє, але не навпаки (знання стану $x(t_1)$ і вхідного впливу в інтервалі часу $(t_1, t_2]$ є необхідною і достатньою умовою, що дозволяє визначити стан $x(t_2)$ якщо $t_1 < t_2$).

Динамічну систему за класифікацією Р.Калмана можна визначити наступним чином (формула 1.1):

$$\Sigma = \{T, X, U, \Omega, Y, \Gamma, \varphi, \eta\}, \tag{2.1}$$

де T – впорядкована множина моментів часу, $T \subseteq R$;

X – множина станів (станом динамічної системи називається вектор значень змінних системи);

U – безліч значень вхідних впливів;

$\Omega = \{\omega: T \rightarrow U\}$ – набір функцій вхідних впливів,

Y – безліч вихідних величин;

$\Gamma = \{\gamma: T \rightarrow Y\}$ – набір функцій породження вихідних величин;

$\varphi: T \times T \times X \times \Omega \rightarrow X$ – перехідна функція стану;

$x(t) = \varphi(t; x_0, t_0, \omega)$, де x – стан системи в момент t, якщо в початковий момент t0 вона перебувала в початковому стані x_0 і була схильна до вхідного впливу ω;

$\eta: T \times X \rightarrow Y$ – вихідна відображення, що визначає потік вихідних величин.

Відрізок вхідного впливу $\omega(t_0,t_1]$ є історією вхідних сигналів в період $(t_0, t_1]$, $t_0, t_1 \in T$. Подією системи Σ називається пара <t, x>. Траєкторією називається функція φ в просторі подій $T \times X$. Управлінням називається вхідний вплив ω.

Сучасне уявлення [4] про динамічну систему виявляється ще більш загальним і полягає в тому, що динамічну систему утворює деяка модель часу і будь-який набір траєкторій (без уточнення множин вхідних, вихідних і керуючих змінних).

Класифікація, запропонована Р.Калманом в математичної теорії систем [1], виділяє класи стаціонарних, скінченновимірних, дискретних, лінійних і гладких динамічних систем в залежності від форми і виду множин T, X, Ω, Y, Г і класів функцій φ і η.

Динамічну систему вважають стаціонарної при виконанні наступних умов:

- T є адитивною групою;
- Ω замкнуто щодо оператора зсуву $z\tau$: z^{τ}: $\omega \rightarrow \omega'$, що визначається співвідношенням $\omega'(t) = \omega(t + \tau)$, $\forall\, t, \tau \in T$;
- існує такий момент часу s, що $\varphi(t; \tau, x, \omega) = \varphi(t + s; \tau + s, x, zs, \omega)$;
- відображення η не залежить від t.

Динамічна система є системою з безперервним часом (з дискретним часом) якщо $T = R$ ($T = Z$).

2.2 Формалізація моделей

Під безперервно-дискретними системами прийнято розуміти паралельні та розподілені динамічні системи, які складаються з великого числа елементів різної природи, зокрема, з елементів, поведінка яких описується безперервними процесами, що мають кінцеву тривалість, і елементів, поведінка яких описується дискретними процесами, час реакції на події в яких несуттєво для аналізу системи.

Хоча безперервно-дискретна система має багато спільного з дискретними паралельними і розподіленими системами, тим не менш, вона не може бути зведена тільки до дискретних моделей, так як динаміка її безперервних компонентів є досить складною. З іншого боку, в такій системі

можуть виникати події, в результаті яких змінюється глобальне поведінка або структура системи, що істотно ускладнює опис її в рамках класичної теорії динамічних систем. При цьому така подія може породжувати нові, а сам дискретний процес, в результаті якого може відбуватися вибір нового поведінки, описується складним дискретним алгоритмом, який в загальному випадку можна уявити графом миттєвих переходів.

Неможливість уявлення безперервно-дискретної системи або тільки дискретними, або тільки динамічними моделями дозволяє виділити ці системи в окремий клас, який може описуватися у вигляді нескінченної і впорядкованої у часі послідовності безперервних або миттєво-дискретних поводжень.

Поняття агрегату разом з розробленими алгоритмами моделювання використовувалося для створення систем імітаційного моделювання складних систем управління.

Агрегатом називається наступна математична модель (формула 2.2):

$$A = \{T , Z , X , \Gamma , Y , H , G\}, \qquad (2.2)$$

де $T = [0, T_f] \subset R$ – інтервал моделювання (зазвичай кінцевий);

Z – множина станів (фазовий простір), що складається з набору фазових координат z = (z1, z2,..., Zl), де z1,. . . zl - фазові координати;

{Z (t)} – фазові траєкторії;

{Z (0)} – множина початкових станів;

X – множина вхідних сигналів ($x = [x^{(1)}, . . . , x^{(l)}]$), де l – число вхідних контактів, кожен з яких приймає сигнали різних типів. Моменти приходу вхідних сигналів визначаються через тимчасову послідовність $\{t_j\}$, $t_j \in T$. Послідовність подій $< t_j , x_j >$, $x_j \in X$ називається послідовністю прийому вхідних сигналів;

Г – множина керуючих сигналів (g = [$g^{(1)}$, . . . , $g^{(r)}$], де r – кількість вхідних сигналів). Моменти приходу керуючих сигналів визначаються через тимчасову послідовність $\{\tau_i\}$, $\tau_i \in T$. Послідовність подій $< \tau_i, g_i >$, $g_i \in Г$ називається послідовністю прийому сигналів;

Y – безліч вихідних сигналів. Моменти видачі вихідних сигналів визначаються через тимчасову послідовність $\{\xi_k\}$, $\xi_k \in T$, де елемент ξ_k відповідає часу перетину фазового траєкторією z (t) меж деякого безлічі з системи множин $\{Z_y\}$, визначеної на просторі станів Z завданням набору предикатів над Z. Послідовність подій $< \xi_k, y_k >$, $y_i \in Y$ називається послідовністю видачі вихідних сигналів.

H – оператор переходів, який визначає поточний стан по передісторії;

G – оператор виходів, що дозволяє обчислити значення вихідних ліній у вигляді y (t) = G [z (0), t].

У загальному випадку всі послідовності подій в агрегаті є реалізаціями випадкових послідовностей із заданими законами розподілу, оператор H також є випадковим оператором, тобто будь-якого z (0) відповідає безліч значень z (t) з деяким законом розподілу.

Стан агрегату в момент зміни вхідних і керуючих сигналів або видачі вихідного сигналу прийнято називати особливим, так як в цей момент агрегат може стрибкоподібно змінювати свій стан.

Концепція агрегату може бути використана або для моделювання всієї безперервно-дискретної системи або для опису її окремих елементів. У другому випадку моделируемая система це множина агрегатів з детермінованими і незмінними каналами передачі даних (зв'язками).

В.М.Глушков визначає математичну модель безперервно-дискретної системи на базі дискретного подієвого підходу. Особливістю його моделі є можливість зміни структури системи шляхом породження, видалення, активізації і пассивизации одних елементів системи іншими в процесі їх паралельного виконання. Моделювання поведінки системи здійснюється

послідовною вибіркою з календаря запланованих подій з можливістю використання безперервних чисельних методів як підпрограм нижчого рівня.

На відміну від агрегативного підходу, запропонованого Н.П.Бусленко, алгоритм В.М.Глушкова базується на дискретно-подієвому підході до моделювання складних систем.

Безперервно-дискретної системою з точки зору В.М.Глушкова називається наступна математична модель (формула 2.3):

$$S = \{T , P , e , E , K , F\}, \quad (2.3)$$

де $T = \{ti\}$, $ti \in R$ – дискретна модель часу;

P – множина класів процесів;

e – множина класів подій (причин зміни поведінки системи);

E – множина алгоритмів класів подій (підготовчих дискретних дій при переході до нової поведінки системи).

До елементарних діям відносяться запуск і зупинка породження і знищення процесів, зміна значень змінних процесу і запис в календар планування подій позначки про майбутню подію.

$K = \{< t_i, e_i >, L\}$ – календар планування подій, в який записуються позначки про події окремими об'єктами і за допомогою якого описується динаміка системи. Планування події на увазі явне завдання моменту його настання або завдання умови L його настання через предикат (планування події за умовою).

F – список рівнянь, що характеризують локальні поведінки процесів у тимчасових інтервалах між подіями.

Структура процесу і його поведінка описується наступною математичною моделлю (формула 2.4):

$$P = \{X , Y , V_s , V_d , B\}, \quad (2.4)$$

де X, Y – канали входу і виходу відповідно;

Vs – множина статичних змінних процесу, які задаються алгебраїчними виразами і можуть змінюватися тільки при виконанні алгоритмів подій;

V_d – множина «змінних-функцій» – динамічних змінних, які задаються диференціальними рівняннями з множини F.

Під моделюванням поведінки безперервно-дискретної системи з точки зору В.М.Глушкова розуміється побудова набору послідовностей подій, які приходять до зміни її поведінки, зараховуючи до події і стан початкової ініціалізації системи. Поведінка системи в цілому моделюється за допомогою спеціального процесу (монітора), який змінює системне модельне час, вибираючи наступний момент часу з календаря запланованих подій або відповідно до аналізу часу настання події, яке планується за умовою. Процес моделювання завершується за умови спустошення календаря запланованих подій.

Автори А.Пнуелі, Д.Харел, які є засновниками так званого «сучасного підходу» до моделювання безперервно-дискретних систем, як математична модель пропонують використання гібридного автомата. При цьому гібридний автомат визначається як набір станів (відповідних якійсь підмножені станів системи, що моделюється) і подіями, що призводять до зміни поведінки (яким відповідають дуги системи переходів).

Гібридна система описується наступною моделлю (формула 2.4):

$$H = \{S , X , E , F , \varphi , \psi , \lambda\}, \qquad (2.4)$$

де S – кінцева множина локацій;

X – кінцева множина змінних;

E – кінцева множина дуг, де дугою є кортеж виду $e = < s, a, \varphi_e, \lambda(s_0), s_0> \in E$, s, s0 ∈ S – вихідна і цільова локації для дуги e, a ∈ Σ, где Σ – алфавіт міток переходів (алфавіт подій);

φ – безліч предикатів над X, що описують умови переходу по дугам E;

λ – безліч початкових значень змінних безлічі X для кожної локації;

F – оператор, що задає тип поведінки всередині кожної локації;

Ψ – безліч предикатів над X, що описують область значень змінних X в локаціях s ∈ S (інваріант в s).

При цьому під станом гібридної системи розуміється вектор значень змінних x ∈ X, до якого додається локація s ∈ S, до якої належить цей вектор (тобто для якої $\varphi_s(x)$ є істиною). Таким чином, стан гібридної системи визначається парою < s, x >, где s ∈ S, x ∈ X.

Математичною моделлю, яка задає функціонування гібридної системи, є наведена система переходів (формула 1.6):

$$\Sigma = \{T , Q , Q_0 , Q_F , TR\}, \qquad (2.6)$$

де $T = \{t_i\}$ – упорядочена тимчасова послідовність;

Q – простір станів, що включає Q0, QF – множину початкових і кінцевих станів, що визначаються спеціальними предикатами над X;

TR – множина правил переходів між станами, в якій можна виділити переходи двох типів: перехід (зміна стану) при фіксованому часу (відповідає активізації процесів моделі В.М.Глушкова) та тимчасової перехід (зміна модельного часу всередині локації з відповідною зміною залежних безперервних змінних).

Як показано в роботі [3], всі наведені вище моделі безперервно-дискретних систем можуть бути приведені один до одного. В силу громіздкості дані докази в тексті даної роботи не наводяться.

Проведений аналіз методів моделювання, які використовуються в сучасних програмних комплексах (TESS, Model Vision for Windows, Statemate MAGNUM, i-Logic Rhapsody, HyTech), показує, що на сьогодні існує два основних підходи до дослідження безперервно-дискретних систем:

- уявлення поведінки системи послідовністю класичних динамічних систем;

- спрощення безперервної частини (або заміна її параметрическими оцінками) і використання методів моделювання та аналізу дискретних процесів.

Таким чином, на сьогодні не існує підходу до моделювання та аналізу безперервно-дискретних систем, в якому б рівноправно співіснували методи дослідження дискретної і безперервної компоненти. Існуючі системи моделювання реалізують лише обмежене розширення базових безперервних або дискретних моделей і методів.

2.3 Вибір мови опису апаратних засобів

Мова VHDL, розроблена за ініціативою міністерства оборони США і стандартизований Інститут інженерів з електротехніки та електроніки (IEEE) в 1987 році [10], спочатку містив засоби опису та моделювання дискретно-подієвих систем. При цьому в редакціях 1987 і 1993 року можна відзначити суттєві зміни щодо вдосконалення та приведення до єдиного стилю всіх синтаксичних елементів, введення цілої низки нових концепцій, розширення доступного переліку визначених атрибутів типів даних, сигналів і інших структурних об'єктів. Також необхідно відзначити, що мова VHDL володіє як вбудованим типом даних real, призначеного для відображення безперервного безлічі дійсних значень на дискретне безліч чисел, які представлені в форматі плаваючою точки, так і певними операторами для роботи з цим типом даних. У той же час операції з речовими даними при синтезі не підтримуються відповідно до вимог стандарту на IEEE 1076.6.

Всі оператори мови VHDL поділяються на дві групи: паралельні і послідовні.

Порядок виконання паралельних операторів не залежить від порядку розташування їх у тексті програми і визначається моментами часу зміни внутрішніх чи зовнішніх сигналів (портів). До паралельних операторів відносяться:

- оператор призначення сигналу з можливостями опису часових параметрів призначення;
- оператор процесу (використовується для високорівневих поведінково-алгоритмічних описів);
- оператор конкретизації компонента (використовується для побудови ієрархічних описів, що включають готові модулі як окремі складові більш складного проекту);
- оператор блоку (використовується для виділення окремих фукнціонального-структурних частин проекту з можливостями оголошення локальних об'єктів);
- оператори опису регулярних повторюваних структур (for generate / if generate);
- оператори виклику підпрограм (функцій або процедур);
- оператори виведення налагоджувальних повідомлень assert / report.

Послідовні оператори повторюють множину алгоритмічних конструкцій більшості мов програмування і використовуються всередині процесів і підпрограм. До послідовним операторам можна віднести:

- оператор присвоювання змінної;
- оператор призначення сигналу;
- оператори розгалужень і циклів;
- оператори виклику підпрограм (функцій або процедур);
- оператори виведення налагоджувальних повідомлень assert / report;
- оператор пассивизации процесу wait.

Більшість систем моделювання та автоматизованого синтезу підтримують тільки стандарти 1987 і 1993 років.

У 1999 році стандарт мови VHDL був істотно розширений за рахунок прийняття підмножини IEEE 1076.1 – VHDL AMS (Analog and Mixed Systyems), призначеного для моделювання аналогових і змішаних цифро-аналогових систем.

Для досягнення заданих завдань введений ряд доповнень, що дозволяють використовувати VHDL-AMS для багатоаспектного моделювання безперервних процесів. Концепція багатоаспектного моделювання заснована на аналогіях компонентних і топологічних рівнянь, що складають математичні моделі об'єктів в багатьох предметних областях на макрорівні. При цьому під макрорівнем моделювання розуміється сукупність моделей з зосередженими параметрами (тобто описуються системами звичайних диференціальних рівнянь) і методів їх вирішення. Залежними змінними в цих моделях є фазові змінні, які можуть бути двох типів – змінними типу потенціалу або змінними типу потоку.

У VHDL-AMS перші з них називаються змінними across quantity, другі – through quantity. Компонентні рівняння виражають зв'язки між across і through змінними в окремих елементах модельованих об'єктів, а топологічні рівняння – між однотипними фазовими змінними різних елементів. Так, в разі електричних систем прикладами компонентних рівнянь можуть бути рівняння закону Ома, а топологічних – рівняння законів Кірхгофа.

3 МЕТОД СТВОРЕННЯ АРХІТЕКТУРНИХ СПЕЦИФІКАЦІЙ

На сьогоднішній день специфікація проекту зазвичай складається з безлічі документів, які специфікують окремі аспекти роботи проектованої системи. Ці документи створюються з використанням різних мов і форматів опису поведінки і характеристик системи, таких як текстові описи природньою мовою, що підрозділяються на неструктуровані і структуровані, різні діаграми послідовності операцій, таблиці, рисунки, а також формальні мови.

3.1 Вимоги до формату файлів

Розглянемо всю безліч форматів даних, що використовуються при проектуванні архітектури обчислювальних систем. На сьогоднішній день специфікація проекту зазвичай складається з безлічі документів, специфікує окремі аспекти роботи проектованої системи. Ці документи створюються з використанням різних мов і форматів опису поведінки і характеристик системи, таких як текстові описи природною мовою, що підрозділяються на неструктуровані і структуровані, різні діаграми послідовності операцій, таблиці, малюнки, а також формальні мови. При цьому в процесі створення специфікації опису деяких характеристик системи проходять певну послідовність трансформацій [19]. Наприклад, початкові вимоги були оформлені у вигляді тексту на природній мові, потім на основі цього тексту були складені таблиці, за допомогою яких на останніх етапах створення специфікації був написаний код на одній з мов програмування. Для аналізу виберемо наступний набір форматів:

- тексти на природній мові. Тексти на природній мові є прикладом найменш формалізованих описів системи та, відповідно, мають всі недоліки, пов'язані з неточністю, двозначністю і можливістю різних трактувань.

Незважаючи на всі ці недоліки, тексти на природних мовах на ранніх етапах процесу проектування зазвичай складають основну масу даних про системі. Є кілька причин такого широкого поширення текстів на природною мовою в високорівневих специфікаціях: простота створення, відсутність семантичних обмежень, що дозволяють описувати з їх допомогою лише окремі аспекти системи, а також відсутність необхідності серйозної підготовки фахівців, що створюють документацію, пов'язану з освоєнням необхідних інструментів і мов.

Істотним недоліком текстів на природній мові є складність, а в багатьох випадках і неможливість автоматичної трансформації їх в інші, більш формалізовані форми специфікації, а також неможливість перевірки коректності таких текстів. системі. Є кілька причин такого широкого поширення текстів на природною мовою в високорівневих специфікаціях: простота створення, відсутність семантичних обмежень, що дозволяють описувати з їх допомогою лише окремі аспекти системи, а також відсутність необхідності серйозної підготовки фахівців, що створюють документацію, пов'язану з освоєнням необхідних інструментів і мов. Істотним недоліком текстів на природній мові є складність, а в багатьох випадках і неможливість автоматичної трансформації їх в інші, більш формалізовані форми специфікації, а також неможливість перевірки коректності таких текстів.

- моделі на формальних мовах. Формальні мови надають можливість найбільш точного, позбавленого проблем двозначності опису системи. Зазвичай опису на формальній мові є строго описані приватні моделі, що відображають певні аспекти поведінки майбутньої системи. Такі моделі зручні тим, що їх верифікація може бути проведена автоматично, що є серйозною перевагою в разі, коли складність системи досить велика. Незважаючи на переваги специфікації поведінки і характеристик системи з використанням формальних мов, як правило, лише невелика частина специфікацій описана з їх допомогою. Це пояснюється високою складністю освоєння цих мов і створення з їх допомогою моделей системи, їх поганий

читабельністю, що призводить до неможливості швидкого обміну даними про майбутню систему між зацікавленими сторонами, а також семантичними обмеженнями, значно звужують сферу застосування формальних мов.

- діаграми є однією з найпоширеніших і зручних форм створення специфікацій ОС. З їх допомогою можна описати і структуру майбутньої системи, і її поведінка. Найбільш поширеним візуальною мовою створення діаграм в сфері розробки специфікацій є УМЛ (Unified Modelling Language) [17], що став практично стандартом для написання специфікацій програмних систем.

З метою створення специфікацій ОС використовується кілька типів діаграм: структурні діаграми для опису будови системи, а також різні діаграми діяльності та взаємодії для опису її поведінки Найбільш поширеним мовою створення структурних діаграм є UML. Такі діаграми показують на високому рівні абстракції складу системи і зв'язку між вхідними в неї блоками. На рисунку 3.1 представлений приклад UML-діаграми компонентів, яка описує структуру простий програмної системи електронної комерції [20].

Діаграми діяльності служать для візуального відображення структури алгоритмів поведінки модельованої системи. Вони являють собою покроковий опис дій (змін стану системи) з підтримкою умовних переходів, ітерацій і паралельного виконання.

На рисунку 3.3 представлений приклад UML-діаграми діяльності, описує алгоритм проходження сигналу до наступного блоку, що передбачає отримання відповіді від N з опитаних M блоків [21].

Існує інший схожий на UML-діаграми діяльності візуальна мова опису алгоритмів - BPMN (Бусінесс Process Model анд Notation). Основним завданням нотацій BPMN є графічний опис бізнес-процесів, легкий в створенні і розумінні різними зацікавленими сторонами. У процесі аналізу ми об'єднаємо діаграми діяльності UML і нотації BPMN в одну групу.

Діаграми взаємодії. Діаграми взаємодії показують для набору об'єктів їх життєвий цикл на осі часу (створення, використання, знищення), а також їх взаємодію з використанням повідомлень. Основними складовими елементами цих діаграм є об'єкти, їх лінії життя, елементи, що відображають функції, що їх об'єктами, а також сполучення між об'єктами.

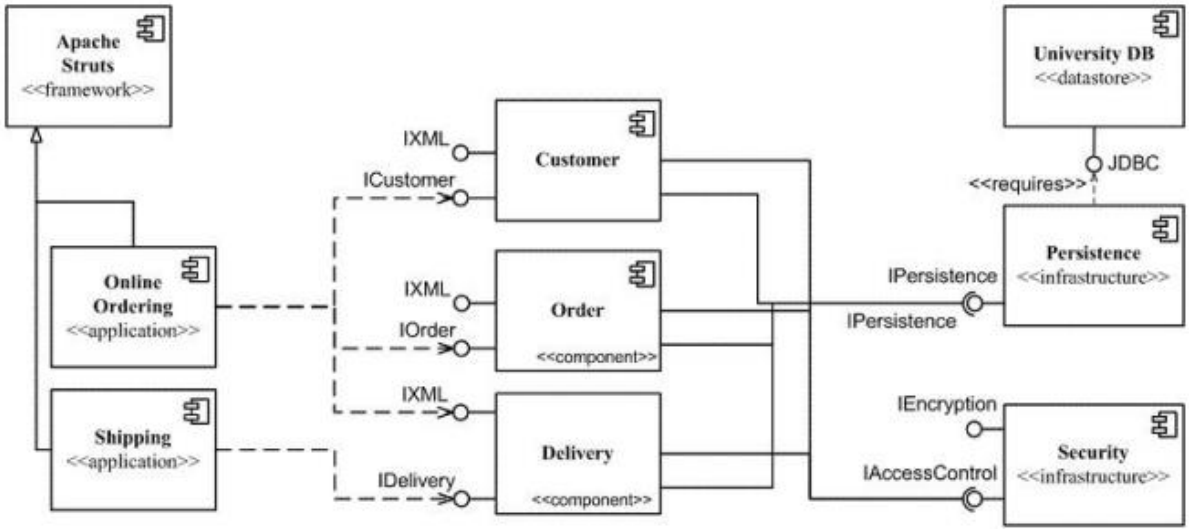

Рисунок 3.1 – Приклад UML-діаграми компонентів

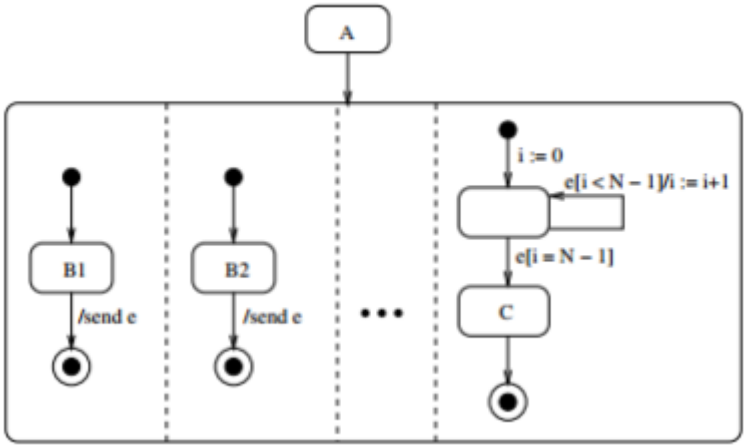

Рисунок 3.2 – Приклад UML-діаграми компонентів діяльності

Крім UML, одним відомим візуальною мовою, так само описує взаємодію об'єктів в системі, є діаграми послідовності повідомлень (MSC,

Message Sequence Charts). На рисунку 3.3 представлений простий приклад діаграми послідовності повідомлень [23]. У процесі аналізу ми об'єднаємо діаграми взаємодії UML і діаграми послідовності повідомлень в одну групу.

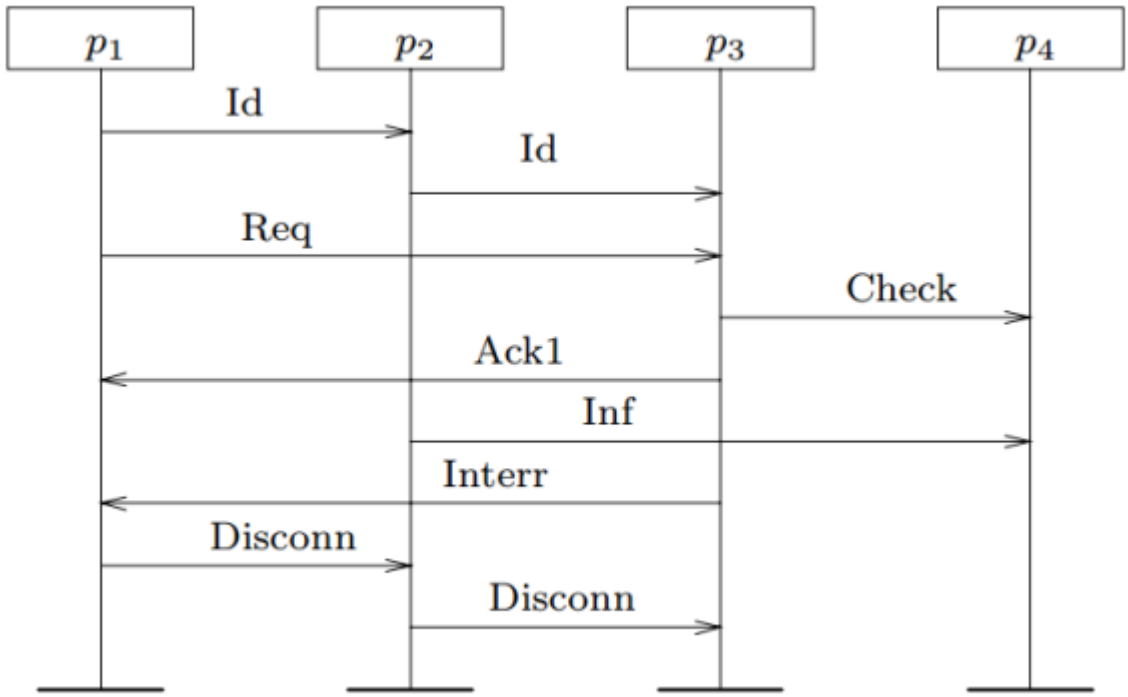

Рисунок 3.3 – Приклад діаграми ослідовності повідомлень

- таблиці. Нарешті, останньою з розглянутих нами груп мов розробки специфікацій є таблиці. Вони зазвичай служать для детального опису певних даних про структуру системи. Табличні дані формалізуються і можуть бути піддані аналізу та валідації, якщо задана структура кожної таблиці і її зв'язок з іншими даними, що містяться в специфікації. Отже, вибір відповідних форматів буде проводитися з таких:

- текст на природній мові;
- формальні мови;
- структурні діаграми;
- діаграми діяльності (UML-діаграми діяльності і нотації BPMN);

діаграми взаємодії (UML-діаграми послідовності, комунікації,

тимчасові діаграми і оглядові діаграми взаємодії, а також діаграми послідовності повідомлень);

3.2 Результати аналізу форматів

Більшість розглянутих форматів широко використовуються при створенні специфікацій обчислювальних систем, за винятком формальних мов. Однак, існують підходи до створення специфікацій і з використанням формальних мов.

Складність створення специфікацій прямо пропорційна ступеню формалізації використовуваної мови. З цієї причини необхідно шукати компроміс між якістю специфікації, безпосередньо залежать від її формалізації, і складністю її створення.

Можливості верифікації відсутні тільки у специфікацій, написаних на природних мовах, що і є однією з проблем, що вирішуються в даній роботі. Всі інші формати даних підтримують можливість перевірки коректності даних. У разі таблиць, однак, повноцінна верифікація вимагає створення таблиць по заздалегідь заданим шаблоном.

Можливості трансляції на формальні мови також відсутні тільки у специфікацій на природних мовах. У разі таблиць для такої трансляції також необхідно заздалегідь знати структуру таблиці.

Для опису структурних аспектів архітектури СнК підходять природні мови, формальні мови та в меншій мірі структурні діаграми.

3.3 Метод створення та використання архітектурних специфікацій

Розробка методів створення специфікацій систем на кристалі і роботи з ними базується вимогах до цих методів, що включають: - підвищення якості створюваних специфікацій, що не вимагає істотних додаткових зусиль архітекторів системи:

- прискорення процесу перевикористання специфікацій;
- автоматизація процесу трансляції специфікацій в формальні моделі.

Беручи до уваги недоліки, часто притаманні архітектурним специфікаціям (неповнота, неточність, двозначність, а також фрагментованість), можна відзначити наступний набір методів, необхідний для виконання першої поставленої задачі:

- вона повинна включати в себе автоматичну перевірку коректності створених специфікацій, вирішальну проблеми неповноти, неточності і двозначності;

- вона повинна дозволяти створювати зв'язку між фрагментами специфікації, що вирішує проблему фрагментированности специфікації. Перевикористання існуючих специфікацій значно прискорює розробку архітектури системи, а також дозволяє уникнути помилок при створенні специфікації.

Так як часто для нових проектів, особливо для СНК, значна частина специфікації створюється шляхом перевикористання специфікацій колишніх проектів, важливою частиною нового набору методів для роботи з архітектурними специфікаціями є трансляція специфікацій з найбільш поширених форматів даних в обрані формати. Нарешті автоматизація процесу трансляції специфікацій в формальні моделі повинна бути реалізована на останньому кроці роботи зі специфікаціями, коли вони вже мають закінчену форму і досить деталізовані. Спираючись на процес проектування систем на кристалі в цілому, неважко уявити, як буде виглядати процес створення специфікацій і роботи з ними.

Після підготовчих кроків, на яких визначається структура специфікації, створюються різні її фрагменти на основі розроблених моделей даних або з нуля, або з використанням даних з існуючих специфікацій з минулих проектів. В останньому випадку застосовується функціонал по трансляції даних з існуючих специфікацій в розроблені моделі даних. Потім застосовується механізм створення зв'язків між фрагментами створеної

специфікації і проводиться базова перевірка корекції отриманої специфікації. Після цього специфікація набуває вигляду, готовий для подальшого використання. Зокрема, один із шляхів використання специфікації, який буде детально розглянуто в даній роботі – трансляція даних з неї в різні формальні моделі з метою подальшого використання цих моделей для верифікації архітектури і на наступних стадіях проектування системи на кристалі.

Вибір способу отримання даних з діаграм залежить від їх типу. У разі структурних діаграм найкращим вибором буде використання для вирішення цього завдання алгоритмів роботи з графами. Наприклад, для отримання ієрархічної структури даних з структурної діаграми може використовуватися простий пошук в ширину, початківець роботу з елементів діаграми верхнього рівня. У разі ж діаграм діяльності або діаграм комунікації можуть бути використані алгоритми приведення їх до виду кінцевого автомата, після чого відкриваються широкі можливості для застосування апарату теорії кінцевих автоматів.

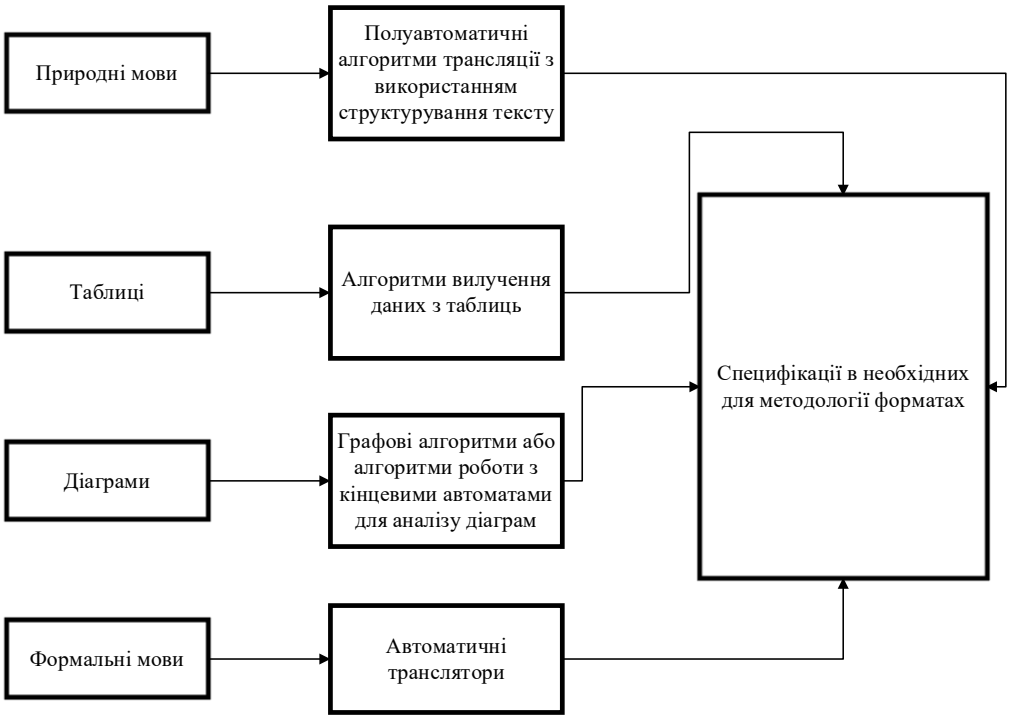

Рисунок 3.4 – Метод створення та використання архітектурних специфікацій СнК

Для трансляції даних з різних формальних мов необхідно розробляти алгоритми для кожного такого мови. Прикладом такого алгоритму може бути синтаксично керований переклад. Схема трансляції даних з існуючих специфікацій представлена на рисунку 3.4.

Створення зв'язків між фрагментами специфікацій може відбуватися декількома шляхами. Перший і найпростіший з них передбачає безпосереднє створення кожного посилання між двома елементами специфікації за допомогою тих же інструментів, які використовуються для заповнення даних специфікацій. Цей шлях може бути реалізований двома способами. Перший з них передбачає, що посилання є частиною використовуваної моделі даних. Наприклад, при використанні табличного формату одне з полів таблиці може містити посилання на іншу таблицю, а при використанні діаграмного формату в список примітивів діаграм може бути доданий примітив логічного зв'язку між елементами.

Другий спосіб передбачає запис посилання у вигляді рядка даних в деяке спеціальне поле для посилань. Наприклад, при використанні діаграмного формату це поле може перебувати у властивостях елементу діаграми, а при використанні табличного формату - у властивостях таблиці. ругий шлях створення зв'язків між фрагментами специфікації передбачає створення класів посилань між ними за певною ознакою. Такі класи можуть записуватися в одному зі спеціальних форматів, наприклад, у вигляді коду на мові програмування або у вигляді ХМЛ документа.

Кожен клас таких посилань може описувати властивості, необхідні кожному з «кінців» посилання. При такому підході створення класів посилань і створення об'єктів-посилань повинно відбуватися в різний час: спочатку складаються класи посилань, потім, після готовності специфікації, алгоритм створення посилань по заданих класів автоматично генерує такі посилання. При цьому в процесі генерації посилань по описаним класів для кожного з них буде знайдено безліч об'єктів специфікації, відповідне кожному «кінця» посилання. У разі якщо необхідно виділити з цих множин

пари пов'язаних елементів, можна скористатися, наприклад, пошуком відповідності по ключу. Як приклад використання такого підходу можна навести випадок, коли початкові елементи всіх діаграм зв'язуються з першими записами таблиць з певною назвою.

Схема процесу створення посилань між фрагментами специфікацій представлена на рисунку 3.5.

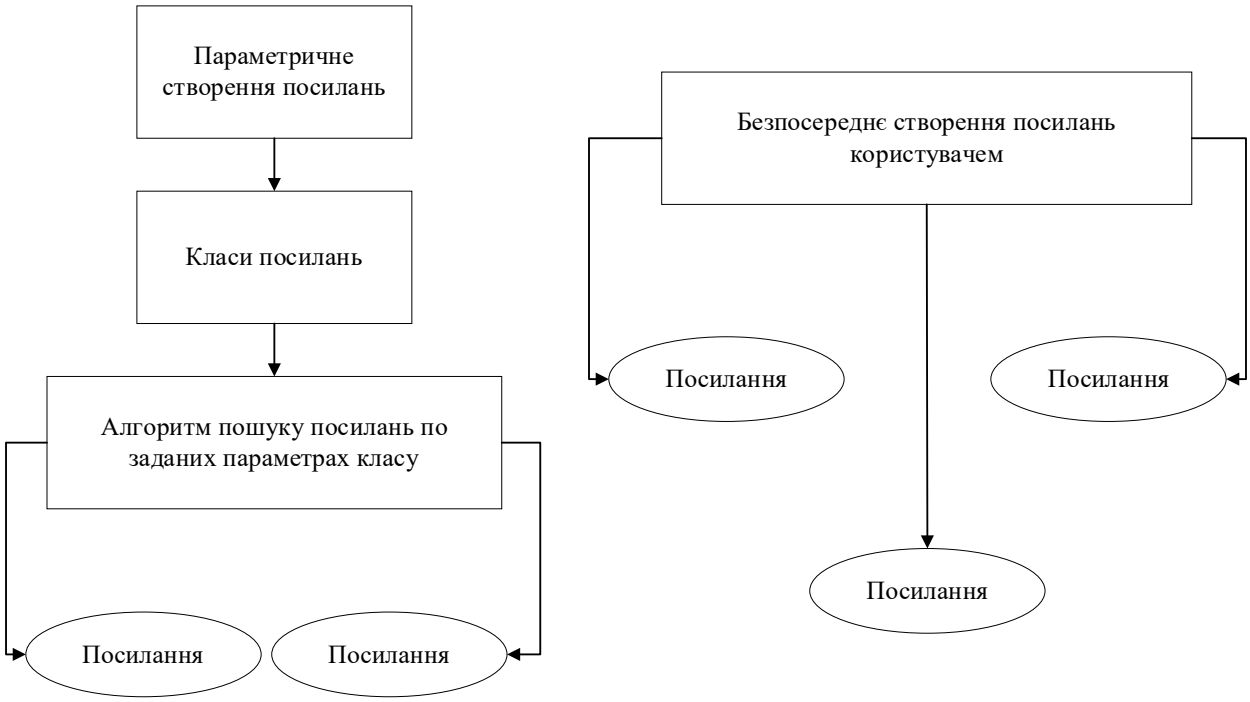

Рисунок 3.5 – Схема процесу створення посилань між фрагментами специфікацій

Процес перевірки коректності (рисунок 3.6) специфікацій може бути розділений на три етапи:

Перший етап – базова перевірка синтаксису специфікації. Другий етап – перевірка цілісності і несуперечності даних. Останній же етап перевірки коректності специфікації полягає в верифікації формальної моделі, побудованої на її основі.

Рисунок 3.6 – Процес перевірки коректності специфікацій

Метод створення специфікацій (рисунок 3.7) та роботи з ними виглядає наступним чином:

Визначення складу і структури майбутньої специфікації:

- створення або вибір шаблонів даних для кожного фрагмента специфікації на основі використовуваних моделей даних;

- створення або вибір алгоритмів трансляції даних з існуючих форм специфікацій в використовувані моделі даних. Цей крок необов'язковий, якщо специфікація створюється з нуля;

- опис класів зв'язків між елементами специфікації. Цей крок може виконуватися паралельно зі створенням специфікації;

- створення інструментарію для перевірки даних на цілісність і несуперечливість. Цей крок може виконуватися паралельно зі створенням специфікації;

- створення або вибір алгоритмів трансляції даних з використовуваних моделей на різні формальні мови. Цей крок може виконуватися паралельно зі створенням специфікації;
- трансляція даних з існуючих специфікацій в використовувані формати. Цей крок необов'язковий, якщо специфікація створюється з нуля;
- створення різних фрагментів специфікації;
- базова перевірка синтаксису специфікації;
- створення зв'язків між фрагментами специфікації;
- перевірка цілісності і несуперечності даних в специфікації;
- трансляція специфікації в різні формальні моделі;
- верифікація отриманих формальних моделей.

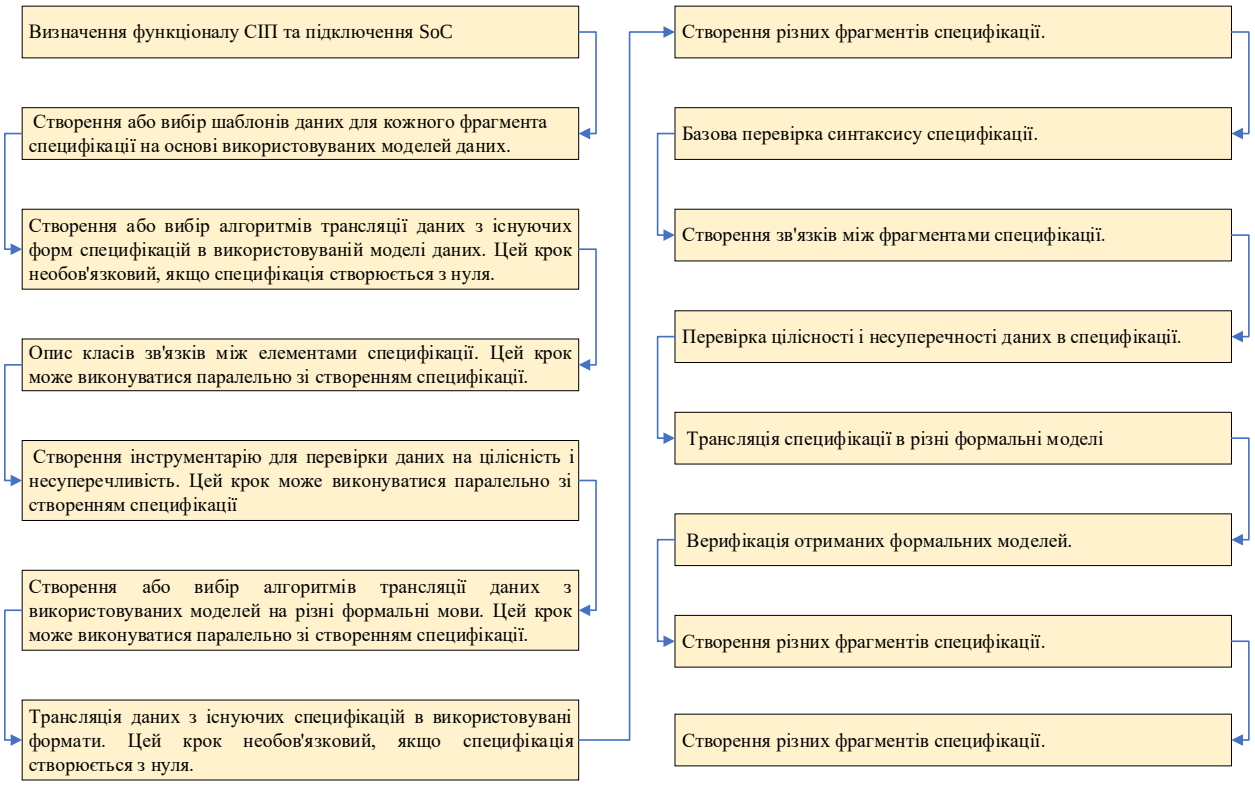

Рисунок 3.7 – Розроблений метод

Необхідно відразу зазначити обмеження на застосування запропонованого підходу. Можна виділити дві основні умови, при яких його застосування буде неефективним.

У разі, коли створення специфікацій ведеться відразу в вигляді формальних моделей і при цьому організаційними методами вирішені проблеми, властиві такому підходу (складність створення специфікацій і їх погана читабельність), застосування запропонованого підходу буде «кроком назад», що знижує ефективність процесу розробки архітектури системи .

У разі, коли специфікація нової системи на кристалі здебільшого складається з специфікацій готових функціональних блоків, написаних на формальних мовах, або повторює готові специфікації СНК, для яких також готові формальні моделі, застосування запропонованого підходу також може знижувати ефективність процесу розробки архітектури. Експериментальні результати показують, що запропонований підхід стає неефективною, коли як мінімум є як мінімум 75% готової специфікації, представленої у вигляді формальних моделей. В інших випадках застосування описаного підходу може значно підвищити якість специфікацій і прискорити процес роботи з ними.

3.4 Програмна реалізація розробленого методу

У зв'язку з тим, що набір методів розділений на дві частини, одна з яких визначає методику роботи з поведінковими специфікаціями і використовує в якості моделі даних діаграми, а друга визначає методику роботи зі структурними специфікаціями і використовує в якості моделі даних таблиці, основними компонентами системи є модуль створення табличних специфікацій і модуль створення специфікацій у формі діаграм.

Модулі системи реалізовані у вигляді розширень для програмних продуктів Microsoft Office. Для розробки табличних специфікацій використовується розширення для Microsoft Excel, що використовує об'єктно-орієнтований підхід до створення таблиць, а для розробки діаграм послідовності операцій використовується розширення для Microsoft Visio. Крім того, важливою частиною програми для створення якісних

архітектурних специфікацій є його інфраструктурна складова, вирішальне завдання забезпечення цілісності і коректності даних, а також організації спільної роботи над специфікаціями.

Модуль управління діаграмними специфікаціями, який реалізує набір методів роботи з поведінковими специфікаціями, носить назву IFlow і використовує однойменні діаграми, описані при побудові моделі даних в розділі 3. Розроблена мова є простим і зручним у вивченні і використанні для архітекторів. Важливо також відзначити, що семантика діаграм IFlow передбачає їх використання для трансляції в формальні моделі для їх подальшої верифікації.

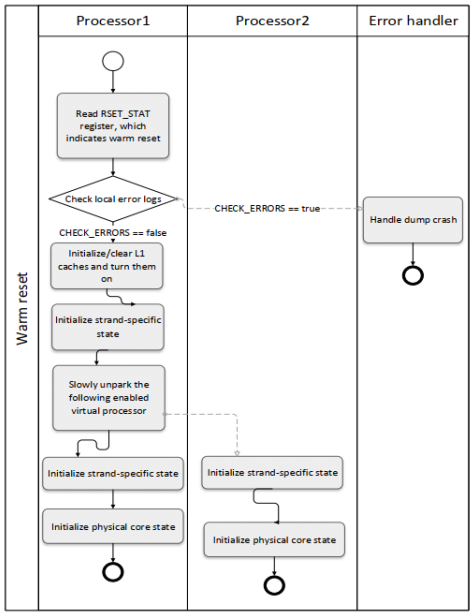

Рисунок 3.8 – Діаграма одного зі сценаріїв

Створюємо регістр для кожного унікального значення поля "RegType" в таблиці. Для кожного запису в таблиці створюємо бітове поле з ім'ям, відповідним значенням поля "FieldName" в регістрі з ім'ям, відповідним значенням поля "RegType". Задаємо характеристики кожного бітового поля: поле розміру заповнюємо значенням поля "BitWidth", поле характеристики

доступу – значенням поля "Access", поле відступу – значенням поля "BitOffset", поле значення після скидання – значенням поля "ResetVal" і так далі.

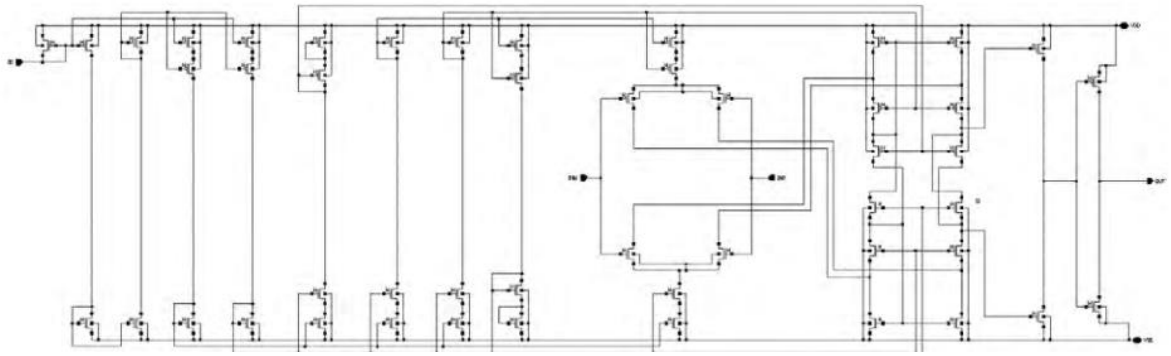

Рисунок 3.9 – Результати роботи

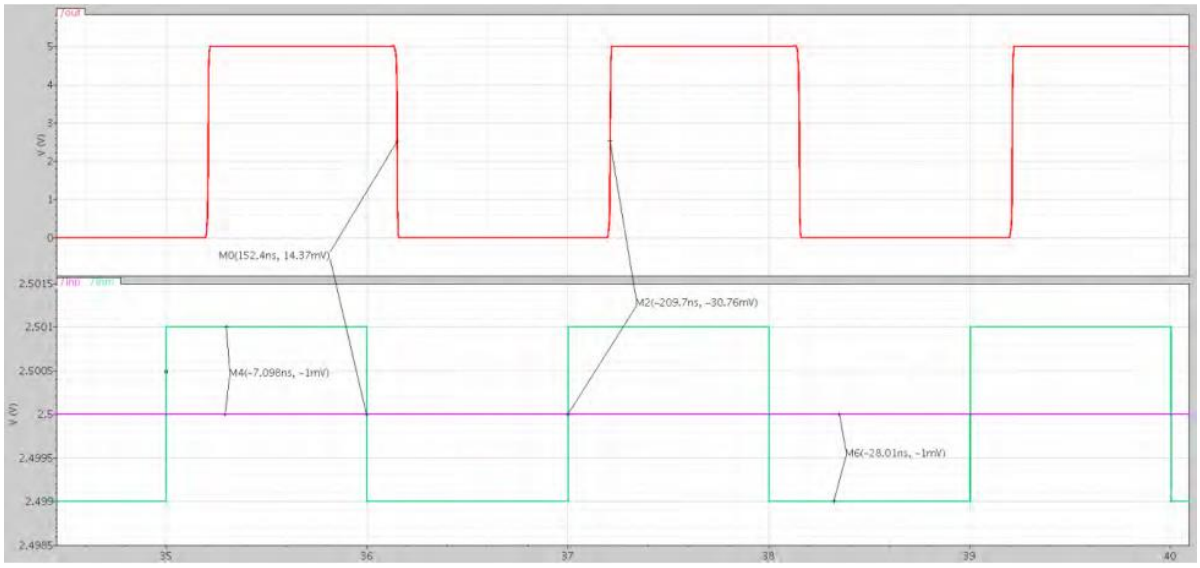

Рисунок 3.10 – Результати роботи

Після застосування описаного алгоритму з цієї таблиці получается наступний RDL-код

ВИСНОВКИ

У ході підготовки кваліфікаційної роботи проведено аналіз методів створення архітектурних специфікацій обчислювальних комплексів на базі SoC. Для спрощення створення виконуваних моделей SoC у набір методів включений метод автоматизації переведення специфікацій у виконувані моделі на формальних мовах; а для раннього спільного аналізу різних компонентів системи та протоколів її функціонування розроблений підхід передбачає застосування різних методів верифікації даних на стадії створення специфікації. Крім того, розроблений метод також дозволяє підвищити якість створюваних специфікацій без суттєвих додаткових зусиль архітекторів та автоматизувати перевикористання специфікацій обчислювальних систем

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Дяченко В.О., Зубенко С.П., Лубан А.С., Федорченко В.М. Методи та інструменти розробки спеціалізованих інтегральних схем для мобільних систем управління// Системи управління, навігації та зв'язку. 2024. № 4(78).

2. Евтушенко Н., Немудров В., Сырцов И. Методология проектирования систем на кристалле. Основные принципы, методы, программные средства // ЭЛЕКТРОНИКА: Наука, Технология, Бизнес. – 2003. – №. 6. – С. 7-11.

3. Спецификация системы на кристалле WonderMedia PRIZM WM8950. [Электронный ресурс] URL: http://www.wondermedia.com.tw/ en/products/platform/soc/wm8950

4. Riccobene E. et al. A SoC design methodology involving a UML 2.0 profile for SystemC // Proceedings of the conference on Design, Automation and Test in Europe-Volume 2. – IEEE Computer Society, 2005. – P. 704-709.

5. Lee Y. et al. Customer Requirements Elicitation based on Social Network Service // KSII Transactions on Internet & Information Systems. – 2011. – Т. 5. – №. 10 .

6. Keller T. Contextual requirements elicitation // Seminar in Requirements Engineering, Spring 2011, Department of Informatics.

7. Dhungana D., Seyff N., Graf F. Research preview: Supporting end-user requirements elicitation using product line variability models // International Working Conference on Requirements Engineering: Foundation for Software Quality. – Springer Berlin Heidelberg, 2011. – С. 66-71.

8. Kumari S. N., Pillai A. S. A survey on global requirements elicitation issues and proposed research framework // Software Engineering and Service Science (ICSESS), 2013 4th IEEE International Conference on. – IEEE, 2013. – P. 554-557.

9. Калман Р. Нариси з математичної теорії систем [Текст] / Р. Калман,

П. Фалбі, М. Арбиб. - М: Світ, 1971. - 294 с.

10. Бусленко М.П .: Моделювання складних систем [Текст] / Н.П. Бусленко. - М: Наука, 1978. - 488 с.

11. Глушков В.М. Програмне забезпечення моделювання безперервно-дискретних систем [Текст] / під заг. ред. В. М. Глушкова. - М: Наука ", 1975. - 188 с.

12. Колмогоров А. Теорія систем. Математичні методи і моделювання [Текст] / А. Колмогоров, С. Новіков; зб. Статей під заг. ред. А.Колмогорова. - М: Світ, 1989. - 314 с.

13. Парійскій Є.Ю. Порівняльний аналіз математичних моделей // Диференціальні рівняння та процеси керування. - 1997, №1. - с. 36-52.

14. Максимей І.В. Імітаційне моделювання на ЕОМ [Текст] / І.В. Максимей. - М .: Радио и связь, 1988. - 230 с.

15. Internal Intermediate Representation (IIR) Specification Version 4.6 Including Digital VHDL & VHDL-AMS support [Текст] / IEEE computer society, 2000. - 364 pp.

16. IEEE 1076-93 Standard VHDL Language Reference Manual [Текст] / IEEE Computer Society, 1993. - 146 pp.

17. IEEE 1076-2008 Standard VHDL Language Reference Manual [Текст] / IEEE Computer Society, 2008. - 254 pp.

18. Perry D.L. VHDL: Programming by Example [Текст] / Douglas L.Perry. - McGraw-Hill, 2004. - 476 pp.

19. Black D.C. SystemC from the Ground Up [Текст] / D.C.Black, J.Donovan. - Kluwer academic publishers, 2007. - 244pp.

20. Parr T. The definitive ANTLR reference [Текст] / T.Parr. - The Pragmatic Programmers, 2009. - 396 pp