Форма № 24

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій
(повне найменування інституту, факультету)

Кафедра комп'ютерних систем та мереж
(повне найменування кафедри)

Пояснювальна записка

до дипломного проекту (роботи)
бакалавра
(ступінь вищої освіти)

на тему РОЗРОБКА ГЕНЕРАТОРА ЧАСОВИХ МІТОК ДЛЯ РЕЄСТРАЦІЇ ПОДІЙ НА БАЗІ ПЛІС

Виконав: студент(ка) 4 курсу, групи КНТ-522

Спеціальності 123 «Комп'ютерна інженерія»
(код і найменування спеціальності)

Освітня програма (спеціалізація)
_____ «Комп'ютерна інженерія» _____

_____ПОМПА Є.С._____
(прізвище та ініціали)

Керівник __ГРУШКО С.С._____
(прізвище та ініціали)

Рецензент_____ЩЕРБАКОВ В.І_____

2026

Форма № 25

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет «Запорізька політехніка»

(повне найменування вищого навчального закладу)

Факультет Комп'ютерних наук і технологій

Кафедра «Комп'ютерні системи та мережі»

Ступінь вищої освіти бакалаврський

Спеціальність 123 Комп'ютерна інженерія

(код і найменування)

Освітня програма (спеціалізація) Комп'ютерна інженерія

(назва освітньої програми (спеціалізації)

ЗАТВЕРДЖУЮ

Завідувач кафедри КУДЕРМЕТОВ Р.К.

"____" _____ 2026 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА

ПОМПИ Євгенія Сергійовича

(ПРІЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Розробка генератора часових міток для реєстрації подій на базі ПЛІС

керівник проєкту (роботи) к. т. н., доцент, ГРУШКО Світлана Сергіївна

(науковий ступінь, вчене звання, ПРІЗВИЩЕ, ім'я, по батькові)

затверджені наказом вищого навчального закладу від "14" квітня 2026 року № 160

2. Строк подання студентом проєкту (роботи) 30 травня 2026 року

3. Вихідні дані до проєкту (роботи) тактова частота системи 100 МГц; роздільна здатність часових міток 10 нс; розрядність лічильника часових міток 32 біти (діапазон вимірювання до 42,95 с); триступеневий синхронізатор вхідного сигналу для придушення метастабільності; детектування подій за переднім або заднім фронтом з програмним вибором типу фронту.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1) Аналіз методів та засобів генерації часових міток;
2) Проєктування генератора часових міток;
3) Програмна реалізація генератора часових міток;
4) Тестування та експериментальне дослідження.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРІЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4	ГРУШКО С. С., к.т.н., доцент		
нормоконтроль	ГОЛУБ Т.В., к.т.н., доцент		

7. Дата видачі завдання 14.04.2026 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Аналіз технічного завдання. Огляд сервоприводів, методів керування та порівняльний аналіз реалізацій контролерів на MCU та FPGA	14.04.26 – 20.04.26	
2	Огляд наявних комерційних та академічних рішень. Обґрунтування вибору апаратної платформи та середовища розробки	20.04.26 - 25.04.26	
3	Проєктування модульної архітектури контролера: структурна схема, модулі ШІМ, енкодера, ПІД-регулятора, UART	25.04.26 - 10.05.26	
4	Реалізація модулів контролера мовою VHDL	10.05.26 – 20.05.26	
5	Функціональна верифікація та моделювання роботи системи в Active-HDL	20.05.26 - 22.05.26	
6	Оформлення пояснювальної записки	22.05.26 - 25.05.26	
7	Оформлення графічного та допоміжного матеріалу	25.05.26 – 30.05.26	

Студент (ка) _____ **Євгеній ПОМПА**
(підпис) (Ім'я ПРІЗВИЩЕ)

Керівник проєкту (роботи) _____ **Світлана ГРУШКО**
(підпис) (Ім'я ПРІЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної роботи бакалавра: 75 с., 12 табл., 6 рис., 11 лістингів, 53 джерела.

ГЕНЕРАТОР ЧАСОВИХ МІТОК, ПРОГРАМОВАНА ЛОГІЧНА ІНТЕГРАЛЬНА СХЕМА, ПЛІС, VHDL, ПЕРЕТВОРЮВАЧ ЧАС-КОД, РЕЄСТРАЦІЯ ПОДІЙ, БУФЕР FIFO, СИНХРОНІЗАЦІЯ

Об'єкт розробки – генератор часових міток для реєстрації подій на базі програмованої логічної інтегральної схеми (ПЛІС).

Мета роботи – розробка автономного параметризованого генератора часових міток, що забезпечує прецизійну фіксацію моментів надходження зовнішніх подій із роздільною здатністю 10 нс та збереження зареєстрованих даних у внутрішньому буфері.

У дипломній роботі досліджено та проаналізовано сучасні методи вимірювання часових інтервалів і реєстрації подій, зокрема метод лічильника, часового розтягування, перетворення час-амплітуда (TAC), Верньє та ліній затримки з відводами (TDL). Розглянуто переваги й недоліки реалізації генераторів на мікроконтролерах, спеціалізованих інтегральних схемах (ASIC) та ПЛІС, обґрунтовано вибір ПЛІС як платформи, що поєднує необхідну роздільну здатність, багатоканальність і гнучкість конфігурування.

Розроблено модульну архітектуру генератора, що включає чотири основні функціональні блоки: триступеневий синхронізатор вхідного сигналу для придушення метастабільності, детектор фронтів із програмним вибором переднього або заднього фронту, 32-бітний лічильник часових міток та буфер FIFO глибиною 16 записів. Систему параметризовано за допомогою механізму generic, що дозволяє змінювати розрядність лічильника та глибину буфера без зміни вихідного коду.

ЗМІСТ

ВСТУП

Сучасний розвиток науки та техніки характеризується постійним зростанням вимог до точності вимірювання часових параметрів фізичних процесів. Генератори часових міток, які забезпечують прецизійну фіксацію моментів настання подій, є ключовими компонентами в широкому спектрі застосувань — від фізики високих енергій та позитронно-емісійної томографії до систем лазерної локації та телекомунікаційних мереж [1, 2]. Стрімкий прогрес у галузі програмованих логічних інтегральних схем (ПЛІС) відкрив нові можливості для реалізації високоточних систем реєстрації подій, що поєднують гнучкість конфігурування з продуктивністю, характерною для спеціалізованих апаратних рішень [3].

Проблема точного вимірювання часових інтервалів має глибоке історичне коріння, що сягає середини XX століття, коли розвиток ядерної фізики та космічних досліджень створив потребу у високоточних засобах реєстрації подій. Перші електронні системи вимірювання часу базувалися на аналогових методах — перетворенні часового інтервалу в амплітуду напруги з наступним аналого-цифровим перетворенням. Такі системи забезпечували роздільну здатність до десятків пікосекунд, проте характеризувалися значними габаритами, високим енергоспоживанням та чутливістю до зовнішніх факторів [4].

Революційні зміни у галузі вимірювання часових інтервалів відбулися з появою інтегральних схем та, особливо, програмованих логічних пристроїв. У 1997 році група польських дослідників на чолі з J. Kalisz вперше продемонструвала можливість реалізації перетворювача час-код (TDC — Time-to-Digital Converter) на базі ПЛІС з роздільною здатністю 200 пс [3]. Ця робота заклала фундамент для подальшого розвитку ПЛІС-орієнтованих методів вимірювання часу та стимулювала численні дослідження в цьому напрямку.

Актуальність теми дослідження зумовлена зростаючими потребами експериментальної фізики, медичної діагностики та промислових вимірювальних систем у засобах високоточної реєстрації подій з часовою роздільною здатністю на

рівні пікосекунд. Сучасні детектори частинок у фізиці високих енергій налічують мільйони каналів реєстрації, кожен з яких потребує індивідуального вимірювача часу з роздільною здатністю краще 100 пс. Позитронно-емісійна томографія нового покоління (time-of-flight PET) досягає субміліметрової просторової роздільної здатності завдяки використанню TDC з роздільною здатністю 10–20 пс [5, 6].

Традиційні підходи, засновані на використанні мікроконтролерів або спеціалізованих інтегральних схем (ASIC), не завжди забезпечують оптимальне співвідношення між точністю вимірювань, вартістю розробки та можливістю адаптації до специфічних вимог конкретного застосування. Мікроконтролери обмежені послідовною природою обробки даних та недетермінованими затримками програмного забезпечення. ASIC забезпечують найкращі характеристики, проте їх розробка потребує значних фінансових та часових ресурсів, а неможливість модифікації після виготовлення є суттєвим обмеженням для науково-дослідних застосувань [7].

Реалізація генераторів часових міток на базі ПЛІС дозволяє подолати ці обмеження завдяки унікальному поєднанню властивостей: реконфігурованості, що забезпечує можливість швидкої адаптації до змінних вимог; паралельної обробки даних, що дозволяє реалізувати сотні незалежних каналів на одному кристалі; використання спеціалізованих апаратних ресурсів сучасних програмованих кристалів, зокрема ланцюгів прискореного перенесення (carry chains) з затримкою елемента порядку 10–30 пс для сучасних технологічних процесів [8, 9].

Особливої актуальності набуває розробка генераторів часових міток для застосувань, де критичними є одночасно висока роздільна здатність, багатоканальність та помірна вартість. До таких застосувань належать системи синхронізації розподілених вимірювальних комплексів, детектори космічного випромінювання, системи квантової криптографії та телекомунікаційні мережі з часовою синхронізацією на рівні пікосекунд [10].

1 АНАЛІЗ МЕТОДІВ ТА ЗАСОБІВ ГЕНЕРАЦІЇ ЧАСОВИХ МІТОК

1.1 Огляд сучасних систем реєстрації подій та вимоги до точності часових міток

Системи реєстрації подій з високою часовою роздільною здатністю знаходять широке застосування у сучасній науці та техніці. Необхідність точного вимірювання моментів настання подій виникає у численних галузях — від фундаментальних досліджень у фізиці елементарних частинок до прикладних задач промислової автоматизації та телекомунікацій. Характер вимог до систем реєстрації подій суттєво відрізняється залежно від конкретного застосування, що обумовлює різноманіття архітектурних рішень та технологій реалізації.

Фізика високих енергій є одним з найбільш вимогливих споживачів систем реєстрації подій. Сучасні детектори частинок на прискорювачах, таких як Великий адронний колайдер (LHC) у CERN, містять мільйони чутливих елементів, кожен з яких потребує індивідуального каналу вимірювання часу. Метод time-of-flight (TOF) використовується для ідентифікації частинок за їх швидкістю, яка визначається шляхом вимірювання часу прольоту між двома детекторами на відомій відстані [11, 12].

Для ефективної ідентифікації частинок у типовому діапазоні імпульсів необхідна часова роздільна здатність на рівні 50–100 пс. Детектори наступного покоління, що розробляються для модернізації LHC та нових прискорювачів, передбачають досягнення роздільної здатності 10–30 пс, що висуває безпрецедентні вимоги до електроніки зчитування [13]. Окрім високої роздільної здатності, детектори частинок потребують надзвичайно великої кількості каналів — від сотень тисяч до мільйонів, що вимагає мініатюризації та енергоефективності окремих каналів реєстрації.

Позитронно-емісійна томографія (ПЕТ) є важливим методом медичної діагностики, що базується на реєстрації пар анігіляційних гамма-квантів, які виникають при взаємодії позитронів (випромінюваних радіоактивним маркером) з

електронами тканин організму. Два гамма-кванти з енергією 511 кеВ розлітаються у протилежних напрямках і реєструються кільцем детекторів, що оточує пацієнта. Просторове положення точки анігіляції визначається як перетин ліній, що з'єднують пари детекторів, які зареєстрували співпадаючі (коінцидентні) події [14, 15].

Класична ПЕТ визначає положення точки анігіляції з точністю, обмеженою розмірами детекторів та геометрією системи. Метод time-of-flight PET (TOF-PET) додатково використовує різницю часів реєстрації двох гамма-квантів для локалізації точки анігіляції вздовж лінії з'єднання детекторів. При швидкості світла $c \approx 3\times10^8$ м/с різниця часів Δt відповідає зміщенню точки анігіляції на відстань $\Delta x = c\cdot\Delta t/2$ від центру. Для досягнення просторової роздільної здатності $\Delta x = 1$ см необхідна часова роздільна здатність $\Delta t \approx 67$ пс [16].

Сучасні TOF-PET сканери досягають часової роздільної здатності 200–400 пс, що відповідає просторовій роздільній здатності 3–6 см. Системи наступного покоління з роздільною здатністю 50–100 пс дозволять досягти субсантиметрової локалізації, що суттєво покращить якість зображення та зменшить дозу опромінення пацієнта. Розробка електроніки з такою роздільною здатністю є активною галуззю досліджень [17].

Системи лазерної локації та далекоміри використовують принцип вимірювання часу поширення світлового імпульсу для визначення відстані до об'єкта. Лазерний імпульс випромінюється у напрямку цілі, відбивається від неї та повертається до приймача. Час між випромінюванням та прийомом імпульсу (time-of-flight) пропорційний подвійній відстані до об'єкта:

$$d = c\cdot t/2, \tag{1.1}$$

де c — швидкість світла,

t — виміряний час [18].

За швидкості світла $c \approx 3\times10^8$ м/с похибка вимірювання часу в 1 нс відповідає похибці визначення відстані приблизно 15 см. Для топографічних застосувань та

автономних транспортних засобів, де вимагається сантиметрова точність, необхідна роздільна здатність TDC на рівні 50–100 пс. Супутникова лазерна локація (SLR — Satellite Laser Ranging) для високоточних геодезичних вимірювань потребує роздільної здатності 10–30 пс, що відповідає точності визначення відстані 1–5 мм [19, 20].

Телекомунікаційні системи нового покоління потребують точної синхронізації для забезпечення когерентної роботи розподілених вузлів. Мережі 5G та оптичні транспортні мережі використовують протоколи точної синхронізації часу (PTP — Precision Time Protocol, IEEE 1588), які забезпечують точність синхронізації на рівні наносекунд або субнаносекунд. Квантові комунікаційні мережі, що базуються на квантовому розподілі ключів (QKD — Quantum Key Distribution), потребують ще вищої точності синхронізації — на рівні десятків пікосекунд — для надійного розрізнення фотонів у часових вікнах [21, 22].

Промислова автоматизація та системи керування використовують високоточні вимірювачі часу для реєстрації подій у технологічних процесах, синхронізації розподілених систем керування та прецизійного позиціонування. Системи числового програмного керування (ЧПК) для високоточної обробки матеріалів потребують роздільної здатності на рівні наносекунд для точного вимірювання швидкості руху робочих органів. Системи неруйнівного контролю на основі ультразвукових методів використовують TDC для вимірювання часу поширення акустичних хвиль у матеріалах [23].

У табл. 1.1 систематизовано вимоги до часової роздільної здатності для різних галузей застосування систем реєстрації подій.

Окрім часової роздільної здатності, важливими характеристиками систем реєстрації подій є:

– діапазон вимірювання – максимальний часовий інтервал, який може бути виміряний;

– мертвий час – мінімальний інтервал між послідовними вимірюваннями, протягом якого система не здатна реєструвати нові події;

– точність – відхилення результату вимірювання від істинного значення;

– прецизійність – розкид результатів при повторних вимірюваннях одного інтервалу;

– лінійність – відхилення характеристики перетворення від ідеальної прямої;

– стабільність – зміна параметрів у часі та при варіаціях температури і напруги живлення [1].

Таблиця 1.1 — Вимоги до часової роздільної здатності для різних застосувань

Галузь застосування	Роздільна здатність	Кількість каналів	Діапазон
Фізика високих енергій (TOF)	10–100 пс	10^3–10^6	1–100 мкс
TOF-PET медична діагностика	50–200 пс	10^3–10^4	10–100 нс
Супутникова лазерна локація	10–30 пс	1–10	1–10 мс
LiDAR для автономних авто	100–500 пс	10^2–10^3	1–10 мкс
Квантові комунікації	20–100 пс	1–10	10–100 нс
Синхронізація мереж (PTP)	1–10 нс	1–100	необмежений
Промислова автоматизація	1–100 нс	10–100	1 мкс–1 с

1.2 Порівняльний аналіз методів вимірювання часових інтервалів

1.2.1 Метод лічильника

Еволюція методів вимірювання часових інтервалів охоплює понад півстоліття розвитку електронної техніки. Фундаментальний огляд методів вимірювання часових інтервалів з пікосекундною роздільною здатністю представлено у класичній роботі J. Kalisz [1], яка залишається базовим джерелом для дослідників у цій галузі. Подальший розвиток методів описано у монографії S.

Henzler [2] та численних оглядових статтях [24, 25]. Усі методи можна класифікувати на аналогові та цифрові, при цьому сучасні тенденції спрямовані на максимальне використання цифрових підходів завдяки їх стійкості до зовнішніх збурень, простоті інтеграції та можливості цифрової корекції систематичних похибок.

Найпростішим методом вимірювання часових інтервалів є підрахунок періодів опорного тактового сигналу. Сигнал START запускає лічильник, який інкрементується з кожним фронтом тактового сигналу, а сигнал STOP зупиняє підрахунок. Результат вимірювання N відповідає часовому інтервалу $T = N \cdot T_0$, де T_0 — період тактового сигналу. Роздільна здатність методу обмежена періодом тактового сигналу: для генератора з частотою f = 1 ГГц роздільна здатність становить $T_0 = 1$ нс [26].

Перевагами методу лічильника є простота реалізації, необмежений діапазон вимірювання (визначається лише розрядністю лічильника), малий мертвий час та висока лінійність. Основним недоліком є обмежена роздільна здатність, яка для практично досяжних тактових частот становить одиниці наносекунд. Для покращення роздільної здатності використовують метод усереднення: якщо вимірюваний інтервал асинхронний відносно тактового сигналу, багаторазове повторення вимірювання з наступним усередненням дозволяє зменшити квантування до будь-якого бажаного рівня. Проте цей підхід потребує значного часу вимірювання і неприйнятний для однократних подій.

1.2.2 Метод часового розтягування (time stretching)

Аналоговий метод, що полягає у перетворенні короткого вимірюваного інтервалу в пропорційно довший, який легше виміряти стандартним лічильником. Типова схема містить конденсатор, який заряджається великим струмом I_1 протягом вимірюваного інтервалу T_x, а потім розряджається малим струмом I_2. Коефіцієнт розтягування $K = I_1/I_2$ може досягати 100–1000, що дозволяє вимірювати інтервали з роздільною здатністю до 25 пс при використанні тактового генератора 100 МГц [27].

Недоліками методу є залежність від стабільності джерел струму, чутливість до температурних варіацій, значний мертвий час (пропорційний коефіцієнту розтягування) та обмежений діапазон вимірювання. Метод історично використовувався в ранніх TDC, але поступово витісняється цифровими методами.

1.2.3 Метод перетворення час-амплітуда (TAC)

Класичний аналоговий метод, в якому конденсатор лінійно заряджається постійним струмом протягом вимірюваного інтервалу. Напруга на конденсаторі $V = I \cdot T/C$ пропорційна часовому інтервалу T і перетворюється в цифровий код за допомогою АЦП. Метод TAC забезпечує високу роздільну здатність (до одиниць пікосекунд) та широкий динамічний діапазон, проте характеризується значним мертвим часом, обмеженим динамічним діапазоном без перемикання піддіапазонів та чутливістю до дрейфу параметрів аналогових компонентів [1, 28].

TAC залишається методом вибору для застосувань, де необхідна гранична роздільна здатність і допустимий значний мертвий час, наприклад у флуоресцентній спектроскопії з часовим розділенням. Для високошвидкісних застосувань TAC поступається цифровим методам.

1.2.4 Метод Верньє (Vernier method)

Цифровий метод, заснований на використанні двох осциляторів з близькими, але різними періодами T_1 та T_2 ($T_1 > T_2$). Сигнал START запускає осцилятор з періодом T_1, а сигнал STOP — осцилятор з періодом T_2. Оскільки $T_2 < T_1$, фронти другого осцилятора поступово «наздоганяють» фронти першого. Схема збігу (арбітр) фіксує момент, коли фази двох осциляторів стають однаковими. Кількість періодів N до моменту збігу пов'язана з вимірюваним інтервалом співвідношенням $T = N \cdot (T_1 - T_2)$. Роздільна здатність $\Delta T = T_1 - T_2$ може бути зроблена довільно малою шляхом зменшення різниці періодів [17, 18].

Метод Верньє забезпечує роздільну здатність до одиниць пікосекунд при використанні осциляторів з періодами порядку наносекунд. Діапазон вимірювання обмежений однозначністю методу та становить T_1 (один період повільнішого осцилятора) без додаткового грубого лічильника. Основними проблемами є

складність забезпечення стабільності різниці періодів осциляторів та необхідність швидкодіючого арбітра.

1.2.5 Метод ліній затримки з відводами (Tapped Delay Line — TDL)

Широко використовуваний цифровий метод, в якому вимірюваний сигнал поширюється ланцюгом послідовно з'єднаних елементів затримки. На виході кожного елемента (відводу) встановлено D-тригер, тактований опорним сигналом. Момент надходження опорного сигналу фіксує стан усіх тригерів, формуючи термометричний код, в якому кількість одиниць пропорційна часовому інтервалу між вимірюваним та опорним сигналами [3, 29, 30].

Роздільна здатність методу TDL дорівнює затримці одного елемента ланцюга τ. При використанні спеціалізованих елементів затримки в ПЛІС (carry chains) типова затримка елемента становить 10–50 пс залежно від технологічного процесу. Діапазон вимірювання дорівнює $N \cdot \tau$, де N — кількість елементів лінії затримки. Для вимірювання інтервалів, що перевищують період тактового сигналу, TDL комбінують з грубим лічильником (метод Nutt) [31].

Перевагами методу TDL є:

– повністю цифрова реалізація без аналогових компонентів;

– малий мертвий час, обмежений лише часом зчитування та обробки;

– можливість багатоканальної реалізації;

– стійкість до варіацій напруги живлення та температури при застосуванні методів калібрування.

Основним недоліком є нелінійність, спричинена неоднорідністю затримок елементів.

У табл. 1.2 наведено порівняння основних методів вимірювання часових інтервалів за ключовими характеристиками.

Аналіз наведених методів показує, що для реалізації на ПЛІС найбільш придатними є методи TDL та Верньє, оскільки вони є повністю цифровими і можуть ефективно використовувати спеціалізовані ресурси програмованих кристалів. Метод TDL забезпечує оптимальне співвідношення між роздільною

здатністю, мертвим часом та складністю реалізації, що робить його методом вибору для більшості застосувань.

Таблиця 1.2 — Порівняння методів вимірювання часових інтервалів

Метод	Роздільна здатність	Діапазон	Мертвий час	Складність
Лічильник	1–10 нс	необмежений	малий	низька
Time stretching	25–100 пс	1–100 нс	великий	середня
TAC + АЦП	1–25 пс	10–1000 нс	великий	висока
Верньє	1–50 пс	1–100 нс	середній	висока
TDL	10–100 пс	1–20 нс	малий	середня
TDL + лічильник (Nutt)	10–100 пс	необмежений	малий	середня

1.3 Особливості реалізації генераторів часових міток на ПЛІС порівняно з мікроконтролерами та спеціалізованими мікросхемами

1.3.1 Реалізація на мікроконтролерах

Вибір елементної бази для реалізації генератора часових міток суттєво впливає на досяжні характеристики пристрою, вартість розробки, час виходу на ринок та можливості подальшої модифікації. Розглянемо детально три основні підходи до реалізації: мікроконтролери, спеціалізовані інтегральні схеми (ASIC) та програмовані логічні інтегральні схеми (ПЛІС).

Мікроконтролери є найдоступнішим рішенням для простих застосувань з помірними вимогами до часової роздільної здатності. Сучасні мікроконтролери оснащені апаратними таймерами з роздільною здатністю від десятків до сотень наносекунд. Спеціалізовані сімейства, такі як Texas Instruments C2000 або

STMicroelectronics STM32 з модулями захоплення високої роздільної здатності (High-Resolution Timer — HRTIM), забезпечують роздільну здатність до 150–200 пс [32].

Основними перевагами мікроконтролерів є:

– низька вартість;

– наявність розвиненої екосистеми (засоби розробки, бібліотеки, документація);

– простота програмування;

– можливість реалізації складних алгоритмів обробки даних;

– широкий вибір периферійних інтерфейсів.

Проте мікроконтролери мають суттєві недоліки для застосувань з високими вимогами до часової роздільної здатності.

Послідовна природа обробки даних у мікроконтролерах обмежує кількість одночасно оброблюваних каналів. Переривання та обробка даних вносять недетерміновані затримки (jitter), які погіршують прецизійність вимірювань. Максимальна тактова частота обмежена енергоспоживанням та тепловиділенням. Апаратні таймери мають фіксовану архітектуру, яка не може бути оптимізована для конкретного застосування.

1.3.2 Реалізація на спеціалізованих інтегральних схемах (ASIC)

ASIC забезпечують найкращі характеристики серед усіх варіантів реалізації: роздільну здатність до одиниць пікосекунд; мінімальне енергоспоживання на канал; найвищу щільність інтеграції; оптимальну продуктивність для заданого застосування. Провідні виробники (Texas Instruments, Analog Devices, AMS, acam/ams) пропонують широку номенклатуру TDC ASIC для різних застосувань [33].

Наприклад, мікросхема TDC7201 від Texas Instruments забезпечує роздільну здатність 55 пс, діапазон вимірювання до 8 мс та споживання 2.5 мВт на канал. Мікросхеми сімейства TDC-GPX2 від ams OSRAM досягають роздільної здатності 10 пс при 8 каналах на кристалі.

Проте ASIC мають суттєві недоліки, які обмежують їх застосування: висока вартість розробки (мільйони доларів для сучасних технологічних процесів); тривалий цикл розробки (1–3 роки від специфікації до готового продукту); неможливість модифікації після виготовлення; економічна доцільність лише для масового виробництва (сотні тисяч і більше кристалів); ризики, пов'язані з помилками в проєкті.

1.3.3 Реалізація на програмованих логічних інтегральних схемах (ПЛІС)

ПЛІС займають проміжну позицію між мікроконтролерами та ASIC, поєднуючи переваги апаратної реалізації з гнучкістю програмного підходу. Ключовими перевагами ПЛІС для реалізації генераторів часових міток є [24, 34]:

Архітектура ПЛІС дозволяє реалізувати множину незалежних каналів реєстрації, що працюють одночасно. На відміну від послідовної обробки в мікроконтролерах, усі канали ПЛІС функціонують паралельно без взаємного впливу на швидкодію. На одному кристалі середнього розміру можна реалізувати 64–128 каналів TDC з роздільною здатністю 10–30 пс [13].

Сучасні ПЛІС містять спеціалізовані елементи, оптимізовані для швидкого поширення сигналів — ланцюги прискореного перенесення (carry chains). Ці елементи призначені для реалізації швидких арифметичних операцій, але можуть використовуватися як елементи затримки з мінімальною тривалістю затримки 10–50 пс залежно від технологічного процесу [8, 9, 35].

Можливість перепрограмування ПЛІС дозволяє: оптимізувати архітектуру генератора для конкретного застосування; виправляти помилки без заміни апаратного забезпечення; додавати нову функціональність на існуючій платі; реалізувати адаптивні алгоритми, що змінюють конфігурацію в процесі роботи.

На одному кристалі ПЛІС можна реалізувати не лише генератор часових міток, але й: систему збору даних з буферною пам'яттю; попередню обробку та фільтрацію; високошвидкісні інтерфейси зв'язку (Ethernet, PCIe, USB); програмні процесори (soft-core) для складної обробки та керування [36].

Порівняно з ASIC, розробка на ПЛІС потребує значно менше часу та ресурсів. Типовий цикл розробки TDC на ПЛІС становить 3–6 місяців від специфікації до

працюючого прототипу, що особливо важливо для науково-дослідних проєктів та швидкого виходу на ринок комерційних продуктів.

Серед обмежень ПЛІС слід відзначити: вищу вартість кристала та енергоспоживання порівняно з ASIC масового виробництва; залежність характеристик від температури та напруги живлення, що потребує застосування схем калібрування; необхідність спеціалізованих знань для ефективного використання ресурсів ПЛІС; обмеження швидкодії порівняно з оптимізованими ASIC [28].

У табл. 1.3 наведено порівняння характеристик різних підходів до реалізації генераторів часових міток.

Таблиця 1.3 — Порівняння підходів до реалізації генераторів часових міток

Характеристика	Мікроконтролер	ASIC	ПЛІС
Роздільна здатність	150 пс – 10 нс	1–50 пс	5–100 пс
Кількість каналів	1–8	2–16	8–256
Вартість розробки	низька	дуже висока	середня
Час розробки	тижні	1–3 роки	місяці
Модифікованість	висока (ПЗ)	неможлива	висока
Енергоспоживання	низьке	мінімальне	середнє

Таким чином, ПЛІС є оптимальним вибором для розробки генераторів часових міток у застосуваннях, де необхідна висока роздільна здатність (десятки пікосекунд), багатоканальність та можливість швидкої модифікації. Це робить ПЛІС особливо привабливими для науково-дослідних проєктів, прототипування та серійного виробництва з помірними обсягами.

1.4 Аналіз архітектур ПЛІС та їх придатності для реалізації генераторів часових міток

1.4.1 Архітектура ПЛІС AMD/Xilinx

Ринок ПЛІС представлений переважно двома основними виробниками – компаніями AMD (яка придбала Xilinx у 2022 році) та Altera (яка була частиною Intel з 2015 по 2024 рік і відновила незалежність у 2025 році). Ці компанії разом контролюють понад 80% світового ринку ПЛІС [37]. Також на ринку присутні менші гравці: Lattice Semiconductor, Microchip (сімейство PolarFire), Efinix та інші, які займають нішеві позиції.

Базовим елементом ПЛІС компанії AMD (Xilinx) є конфігурований логічний блок (CLB – Configurable Logic Block), який містить два зрізи (slices). Кожен зріз включає таблиці перетворення (LUT — Look-Up Table), тригери (flip-flops) та ланцюги прискореного перенесення (carry chains). У сімействах 7-ї серії (Artix-7, Kintex-7, Virtex-7) та новіших (UltraScale, UltraScale+, Versal) використовуються 6-входові LUT, які можуть конфігуруватися як два 5-входові LUT з спільними входами [38].

Ключовим елементом для реалізації TDC є примітив CARRY4 (або CARRY8 в UltraScale), який забезпечує швидке поширення сигналу перенесення через групу з чотирьох (восьми) бітів. Кожен вихід CARRY4 може бути з'єднаний з тригером для реєстрації стану лінії затримки. Затримка одного елемента CARRY4 залежить від технологічного процесу: для 28-нм технології (7-series) типове значення становить 15–25 пс, для 16-нм (UltraScale+) – 10–15 пс [35, 39].

Сімейства ПЛІС Xilinx відрізняються обсягом логічних ресурсів, кількістю та типом вбудованих блоків пам'яті (Block RAM, UltraRAM), DSP-блоків, швидкісних послідовних приймачів-передавачів та інших спеціалізованих ресурсів. Для реалізації TDC найбільш релевантними є логічні ресурси (LUT, тригери, carry chains) та блоки пам'яті для буферизації результатів.

1.4.2 Архітектура ПЛІС Altera/Intel

Архітектура ПЛІС компанії Altera базується на адаптивних логічних модулях (ALM – Adaptive Logic Module), об'єднаних у логічні масиви (LAB – Logic Array Block). Кожен ALM містить два адаптивних LUT (ALUT), які можуть конфігуруватися у різних режимах: як два 4-входових LUT, один 6-входовий LUT, два 5-входових LUT тощо. Така гнучкість дозволяє ефективно використовувати логічні ресурси для різних типів функцій [40].

Сімейства Cyclone (бюджетний сегмент), Arria (середній сегмент) та Stratix/Agilex (високопродуктивний сегмент) використовують варіації цієї архітектури з різним співвідношенням логічних ресурсів, вбудованої пам'яті, DSP-блоків та швидкісних приймачів-передавачів. Ланцюги прискореного перенесення в ПЛІС Altera також оптимізовані для швидкого поширення сигналів і можуть використовуватися для побудови TDL.

1.4.3 Порівняння архітектур для реалізації TDC

Для реалізації генераторів часових міток на основі ліній затримки критичним параметром є затримка одного елемента ланцюга перенесення. У таблиці 1.4 наведено порівняння характеристик кількох поширених сімейств ПЛІС з точки зору придатності для реалізації TDC.

Таблиця 1.4 — Порівняння сімейств ПЛІС для реалізації TDC

Сімейство	Виробник	Технологія	Затримка carry	Досяжна роздільна здатність
Spartan-6	Xilinx	45 нм	50–80 пс	26–50 пс
Artix-7	Xilinx	28 нм	15–25 пс	10–20 пс
Kintex-7	Xilinx	28 нм	15–25 пс	5–15 пс
Cyclone V	Altera	28 нм	20–30 пс	15–30 пс
Kintex UltraScale	Xilinx	20 нм	10–15 пс	3–10 пс

Перехід до більш тонких технологічних процесів (16 нм, 7 нм) зменшує затримку елементів, що теоретично дозволяє підвищити роздільну здатність TDC.

Проте одночасно загострюються проблеми з нелінійністю характеристики та так званими «бульбашковими помилками» (bubble errors), які виникають через порушення монотонності термометричного коду внаслідок нерівномірності затримок та впливу трасування [39, 43].

Вибір конкретного сімейства ПЛІС для реалізації генератора часових міток залежить від комплексу факторів: вимог до роздільної здатності та прецизійності; необхідної кількості каналів; наявності додаткової функціональності (обробка, інтерфейси); бюджету проєкту; доступності засобів розробки та налагоджувальних плат. Для навчальних та дослідницьких проєктів оптимальним вибором є сімейства Artix-7 або Cyclone V, які забезпечують достатню продуктивність при помірній вартості.

1.5 Методи підвищення роздільної здатності TDC на базі ПЛІС

1.5.1 Метод Wave Union

Роздільна здатність базового TDC на основі лінії затримки з відводами обмежена затримкою одного елемента, яка для сучасних ПЛІС становить 10–50 пс. Для досягнення вищої роздільної здатності та прецизійності застосовують різноманітні методи інтерполяції, усереднення та корекції, які дозволяють подолати фундаментальні обмеження базової архітектури.

Метод Wave Union (хвильове об'єднання), запропонований дослідниками Fermilab [44], полягає у формуванні спеціального вхідного сигналу з кількома фронтами замість одиничного імпульсу. Кожен фронт вхідного сигналу генерує окремий перехід у термометричному коді, і фінальний результат отримується шляхом усереднення позицій усіх переходів.

Існує кілька варіантів методу Wave Union. У методі Wave Union A (WU-A) генерується імпульс з чотирма або більше фронтами (два наростаючих і два спадаючих), кожен з яких реєструється окремо. Ефективна роздільна здатність

покращується пропорційно кількості фронтів — для чотирьох фронтів роздільна здатність зменшується приблизно вчетверо. У методі Wave Union B (WU-B) замість формування складного вхідного сигналу використовується множинна реєстрація одного фронту з різними фазами тактового сигналу [45].

Перевагами методу Wave Union є можливість суттєвого підвищення роздільної здатності без збільшення довжини лінії затримки та кількості апаратних ресурсів. Недоліками є збільшення мертвого часу (необхідно дочекатися проходження всіх фронтів через лінію), ускладнення схеми декодування та можливість появи додаткових похибок при нерівномірних затримках фронтів.

1.5.2 Метод множинних ланцюгів (Multi-chain)

Метод множинних ланцюгів використовує кілька паралельних ліній затримки, фізично зміщених одна відносно одної. За рахунок різних затримок трасування вхідний сигнал досягає різних ланцюгів у різний час, що створює ефект субдискретизації. Результати вимірювання з усіх ланцюгів усереднюються для отримання фінального значення з підвищеною роздільною здатністю [8, 46].

Типова реалізація використовує 2–4 паралельних ланцюги, що дозволяє покращити роздільну здатність у 2–4 рази. Перевагою методу є низький мертвий час та висока пропускна здатність, оскільки всі ланцюги працюють паралельно. Недоліком є необхідність ретельного калібрування для компенсації різниці затримок між ланцюгами та підвищене споживання логічних ресурсів.

1.5.3 Верньє-осцилятор на основі кільцевих генераторів

Цей метод поєднує принцип Верньє з компактною реалізацією на ПЛІС за допомогою кільцевих генераторів (Ring Oscillator — RO). Два кільцевих генератори з близькими частотами f_1 та f_2 ($f_1 < f_2$) запускаються сигналами START і STOP відповідно. Лічильник фіксує кількість періодів N до моменту збігу фаз генераторів. Роздільна здатність визначається різницею періодів: $\Delta T = T_1 - T_2 = 1/f_1 - 1/f_2$ і може досягати одиниць пікосекунд [18, 47].

Перевагою методу є широкий діапазон вимірювання (визначається розрядністю лічильника) без використання грубого лічильника та компактна реалізація. Недоліками є залежність частот кільцевих генераторів від температури

та напруги живлення, що потребує застосування схем калібрування або стабілізації, а також порівняно великий мертвий час.

1.5.4 Двоступенева та багатоступенева інтерполяція

Комбінування різних методів у багатоступеневій архітектурі дозволяє поєднати переваги кожного з них [48]. Типова двоступенева архітектура складається з грубого вимірювача на основі стандартної лінії затримки (роздільна здатність 20–50 пс, діапазон — один період тактового сигналу) та точного вимірювача на основі Верньє-лінії або іншого методу (роздільна здатність 5–10 пс, вузький діапазон).

Перший ступінь визначає грубе положення події в межах періоду тактового сигналу, а другий ступінь уточнює результат. Такий підхід дозволяє оптимізувати використання ресурсів ПЛІС та досягти високої роздільної здатності при помірних апаратних витратах.

1.5.5 Методи корекції нелінійності

Нелінійність характеристики TDC є однією з основних проблем реалізації на ПЛІС, оскільки затримки елементів лінії неоднакові через технологічний розкид та нерівномірність трасування. Для корекції нелінійності використовують методи калібрування, які визначають реальні затримки кожного елемента та формують таблицю корекції [49].

Типовий метод калібрування полягає у подачі на вхід TDC сигналу з випадковим (пуассонівським) розподілом моментів відносно тактового сигналу. За тривалого часу спостереження кількість влучань у кожен бін повинна бути однаковою; відхилення від рівномірного розподілу вказує на ширину відповідного біну. На основі цих даних формується таблиця корекції, яка застосовується до результатів вимірювання в реальному часі.

1.6 Огляд мов опису апаратури та засобів проєктування

Проєктування цифрових систем на ПЛІС здійснюється з використанням мов опису апаратури (HDL — Hardware Description Language), які дозволяють формально описати структуру та поведінку цифрових схем. Найбільш поширеними HDL є VHDL та Verilog, обидві стандартизовані IEEE та підтримуються усіма основними виробниками ПЛІС та засобами проєктування [50].

VHDL (Very High-Speed Integrated Circuit Hardware Description Language, IEEE 1076) розроблена на замовлення Міністерства оборони США у 1980-х роках на основі мови програмування Ada. VHDL характеризується строгою типізацією, багатослівним синтаксисом та орієнтацією на формальну верифікацію. Ці властивості роблять VHDL популярною у критичних застосуваннях — аерокосмічній галузі, оборонній промисловості, телекомунікаціях та медичній техніці, де надійність коду є пріоритетом [51].

Verilog (IEEE 1364) створювався компанією Gateway Design Automation як мова моделювання з синтаксисом, близьким до мови програмування C. Лаконічний синтаксис та менша кількість обов'язкових конструкцій прискорюють написання коду, що робить Verilog популярним у комерційних проєктах та академічному середовищі. Проте слабка типізація може призводити до помилок, які важко виявити. SystemVerilog (IEEE 1800) розширює Verilog засобами верифікації, об'єктно-орієнтованого програмування та покращеною підтримкою синтезу [52].

Середовища розробки для ПЛІС надаються виробниками кристалів і включають повний набір інструментів для проєктування, верифікації та програмування. Компанія AMD (Xilinx) пропонує Vivado Design Suite для сімейств 7-series та новіших (UltraScale, Versal), а також ISE Design Suite для старіших кристалів (Spartan-3, Spartan-6). Компанія Altera (Intel) використовує Quartus Prime для всіх своїх сімейств ПЛІС.

Типовий маршрут проєктування (design flow) включає наступні етапи: введення проєкту (HDL-код або схемне введення); функціональна симуляція для

перевірки логічної коректності; синтез — перетворення HDL-опису в нетліст з логічних елементів; імплементація — розміщення логічних елементів на кристалі (placement) та трасування з'єднань (routing); статичний часовий аналіз (STA) для перевірки виконання часових обмежень; генерація конфігураційного файлу; програмування ПЛІС [53].

Для функціональної верифікації використовуються HDL-симулятори: вбудовані у середовища розробки (Vivado Simulator, ModelSim-Altera) або незалежні (ModelSim/QuestaSim від Siemens EDA, Icarus Verilog — безкоштовний симулятор з відкритим кодом). Симулятори дозволяють моделювати поведінку проєкту на різних рівнях абстракції — від поведінкового опису до нетлісту з реальними затримками.

1.7 Висновки до розділу 1

Сучасні системи реєстрації подій у фізиці високих енергій, медичній діагностиці (TOF-PET), лазерній локації та телекомунікаціях потребують часової роздільної здатності від десятків пікосекунд до одиниць наносекунд. Найбільш вимогливі застосування (супутникова локація, фізика частинок) вимагають роздільної здатності 10–30 пс та прецизійності на тому ж рівні.

Порівняльний аналіз методів вимірювання часових інтервалів показав, що для реалізації на ПЛІС найбільш придатними є цифрові методи: ліній затримки з відводами (TDL) та Верньє. Метод TDL забезпечує оптимальне співвідношення між роздільною здатністю, мертвим часом та складністю реалізації. Для досягнення субнаносекундної роздільної здатності доцільно використовувати комбіновану архітектуру з грубим лічильником (метод Nutt) та точним інтерполятором на основі TDL.

Порівняння реалізації генераторів часових міток на мікроконтролерах, ASIC та ПЛІС показало, що ПЛІС забезпечують оптимальний баланс між

продуктивністю, гнучкістю та вартістю розробки. ПЛІС дозволяють реалізувати багатоканальні системи з паралельною обробкою, використовувати спеціалізовані ресурси (carry chains) для побудови ліній затримки та швидко модифікувати проєкт без зміни апаратного забезпечення.

Аналіз архітектур ПЛІС провідних виробників показав, що сучасні сімейства (Xilinx 7-series, Altera Cyclone V та новіші) містять спеціалізовані ланцюги прискореного перенесення з затримкою елемента 10–50 пс, що визначає базову роздільну здатність TDC. Для підвищення роздільної здатності та прецизійності застосовуються методи Wave Union, множинних ланцюгів, Верньє-осциляторів та багатоступеневої інтерполяції.

На основі проведеного аналізу для подальшої розробки обрано архітектуру генератора часових міток на основі вільного лічильника з синхронізатором вхідного сигналу та буфером FIFO. Для імплементації обрано ПЛІС сімейства Intel MAX V, що забезпечує достатню продуктивність (роздільна здатність 10 нс при тактовій частоті 100 МГц) при мінімальній вартості та компактному форм-факторі.

2 ПРОЄКТУВАННЯ ГЕНЕРАТОРА ЧАСОВИХ МІТОК

2.1 Формулювання технічних вимог до системи

На основі аналізу, проведеного в першому розділі, було сформульовано технічні вимоги до генератора часових міток для реєстрації подій. Система повинна забезпечувати точну фіксацію моменту надходження зовнішніх подій з можливістю подальшого зчитування та обробки зареєстрованих даних.

Основні функціональні вимоги до генератора часових міток включають наступне. По-перше, система повинна здійснювати безперервний підрахунок тактових імпульсів для формування часової бази. По-друге, необхідно забезпечити детектування зовнішніх подій за переднім або заднім фронтом вхідного сигналу з можливістю програмного вибору типу фронту. По-третє, система має зберігати

часові мітки зареєстрованих подій у внутрішньому буфері для подальшого зчитування. По-четверте, потрібно реалізувати механізм синхронізації асинхронних вхідних сигналів для запобігання метастабільності.

Щодо часових характеристик, генератор повинен працювати на тактовій частоті 100 МГц, що забезпечує роздільну здатність часових міток 10 нс. Розрядність лічильника часових міток має становити 32 біти, що дозволяє вимірювати інтервали тривалістю до 42,9 секунд без переповнення. Мінімальний інтервал між подіями, що коректно реєструються, визначається затримкою синхронізатора та становить не менше 30 нс.

Для зберігання часових міток використовується буфер FIFO глибиною 16 записів. Кожен запис містить 32-бітну часову мітку події. Буфер повинен мати прапорці стану: порожній (empty) та повний (full), а також забезпечувати можливість програмного очищення.

Таблиця 2.1 – Технічні характеристики генератора часових міток

Параметр	Значення	Примітка
Тактова частота	100 МГц	Визначає роздільну здатність
Роздільна здатність	10 нс	1 період такту
Розрядність лічильника	32 біти	Максимум 42,9 с
Глибина FIFO	16 записів	Конфігурується
Затримка синхронізації	3 такти	30 нс
Розрядність лічильника подій	8 біт	До 256 подій

2.2 Розробка архітектури генератора часових міток

Архітектура генератора часових міток базується на модульному принципі побудови, що забезпечує гнучкість конфігурації та можливість повторного

використання окремих компонентів. Система складається з чотирьох основних функціональних блоків: синхронізатора вхідного сигналу, детектора фронтів, лічильника часових міток та буфера FIFO.

Синхронізатор вхідного сигналу реалізований як триступеневий зсувний регістр, що забезпечує надійне придушення метастабільності при переході асинхронного сигналу через межу тактового домену. Використання трьох послідовних тригерів замість двох підвищує середній час напрацювання на відмову (MTBF) до значень, що перевищують термін експлуатації обладнання.

Таблиця 2.2 – Порти генератора часових міток

Порт	Напрямок	Розрядність	Опис
clk	Вхід	1	Тактовий сигнал 100 МГц
rst_n	Вхід	1	Асинхронне скидання (активний низький)
event_in	Вхід	1	Вхідний сигнал події
event_edge	Вхід	1	Вибір фронту: '0' – передній, '1' – задній
enable	Вхід	1	Дозвіл роботи генератора
clear_fifo	Вхід	1	Очищення буфера FIFO
timestamp_out	Вихід	32	Поточна часова мітка
event_count	Вихід	8	Кількість зареєстрованих подій
fifo_empty	Вихід	1	Прапорець порожнього FIFO
fifo_full	Вихід	1	Прапорець повного FIFO
event_valid	Вихід	1	Сигнал валідності події

Детектор фронтів аналізує синхронізований сигнал та формує короткі імпульси тривалістю один такт при виявленні переднього або заднього фронту. Вибір типу фронту здійснюється через керуючий вхід event_edge. При значенні '0'

детектується передній фронт (перехід з '0' в '1'), при значенні '1' – задній фронт (перехід з '1' в '0').

Лічильник часових міток являє собою 32-бітний двійковий лічильник, що безперервно інкрементується на кожному тактовому імпульсі при активному сигналі дозволу. Поточне значення лічильника доступне на виході timestamp_out і представляє собою абсолютний час з моменту скидання системи.

Буфер FIFO забезпечує тимчасове зберігання часових міток зареєстрованих подій. При виявленні події поточне значення лічильника автоматично записується в буфер. Реалізація базується на двопортовій пам'яті з окремими покажчиками запису та читання. Прапорці fifo_empty та fifo_full індикують стан буфера.

2.3 Проєктування блоку синхронізації вхідного сигналу

Синхронізація асинхронних сигналів є критично важливою задачею при проєктуванні цифрових систем на ПЛІС. Коли сигнал змінює свій стан в околі активного фронту тактового сигналу, виникає ризик захоплення тригером проміжного значення, що призводить до метастабільного стану. У метастабільному стані вихід тригера може мати невизначене значення протягом певного часу, що потенційно спричиняє помилки в роботі системи.

Для мінімізації ймовірності поширення метастабільності в системі використовується каскад послідовно з'єднаних тригерів. Кожен наступний тригер має додатковий час для стабілізації сигналу, що експоненційно зменшує ймовірність передачі невизначеного стану далі по ланцюгу. У розробленому генераторі застосовано триступеневий синхронізатор, реалізований як 3-бітний зсувний регістр.

Принцип роботи синхронізатора полягає в наступному. На кожному активному фронті тактового сигналу вміст регістра зсувається на одну позицію в напрямку старших бітів, а вхідний сигнал event_in захоплюється в молодший біт.

Таким чином, event_sync[0] містить значення входу із затримкою в один такт, event_sync[1] – із затримкою в два такти, а event_sync[2] – із затримкою в три такти. Для детектування фронтів використовуються біти event_sync[1] та event_sync[2], що забезпечує повну синхронізацію перед аналізом.

Затримка синхронізації становить три тактових періоди, що при частоті 100 МГц еквівалентно 30 нс. Ця затримка визначає мінімальний інтервал між подіями, що можуть бути коректно розрізнені системою. Для більшості практичних застосувань така затримка є прийнятною і не впливає на точність вимірювання часових інтервалів.

2.4 Проєктування детектора фронтів

Детектор фронтів призначений для формування короткого імпульсу тривалістю один тактовий період при зміні стану вхідного сигналу. Розроблений модуль підтримує детектування як переднього (наростаючого), так і заднього (спадаючого) фронту з можливістю динамічного вибору режиму роботи.

Детектування переднього фронту реалізовано за допомогою логічної операції: event_rising = event_sync[1] AND NOT event_sync[2]. Ця умова виконується, коли поточне синхронізоване значення дорівнює '1', а попереднє значення (затримане на один такт) дорівнює '0'. Аналогічно, детектування заднього фронту реалізовано як: event_falling = NOT event_sync[1] AND event_sync[2], що відповідає переходу з '1' в '0'.

Вибір активного фронту здійснюється мультиплексором, керованим сигналом event_edge. При event_edge = '0' на вихід event_detected передається сигнал event_rising, при event_edge = '1' — сигнал event_falling. Це дозволяє адаптувати генератор до різних типів датчиків та джерел подій без зміни апаратної конфігурації.

Вихідний сигнал event_detected має тривалість рівно один тактовий період, що гарантує однократну реєстрацію кожної події незалежно від тривалості імпульсу на вході event_in. Це важлива властивість, що забезпечує коректну роботу лічильника подій та буфера FIFO.

2.5 Проєктування лічильника часових міток

Лічильник часових міток є центральним елементом системи, що формує часову базу для реєстрації подій. Реалізований як 32-бітний двійковий лічильник з можливістю асинхронного скидання та синхронного дозволу роботи.

При активному сигналі скидання (rst_n = '0') лічильник встановлюється в нульовий стан. Після зняття скидання та при активному сигналі дозволу (enable = '1') лічильник інкрементується на кожному наростаючому фронті тактового сигналу. Поточне значення лічильника безперервно доступне на виході timestamp_out.

Вибір розрядності 32 біти обумовлений компромісом між діапазоном вимірювання та апаратними витратами. При тактовій частоті 100 МГц період переповнення лічильника становить 2^32 × 10 нс = 42,95 секунд. Для більшості застосувань цього діапазону достатньо, а при потребі вимірювання довших інтервалів можна використати програмний підрахунок переповнень або збільшити розрядність лічильника через параметр COUNTER_WIDTH. Лічильник реалізований з використанням типу unsigned з бібліотеки IEEE.NUMERIC_STD, що забезпечує коректну арифметику з автоматичним переповненням. При досягненні максимального значення (2^32 - 1) лічильник автоматично скидається в нуль і продовжує рахунок.

2.6 Проєктування буфера FIFO

Буфер FIFO (First-In-First-Out) призначений для тимчасового зберігання часових міток зареєстрованих подій. Використання буфера дозволяє накопичувати дані про декілька послідовних подій без втрати інформації, навіть якщо система обробки не встигає зчитувати дані в реальному часі.

Буфер реалізований на базі масиву регістрів глибиною 16 елементів, кожен з яких має розрядність 32 біти. Управління буфером здійснюється за допомогою двох покажчиків: fifo_wr_ptr (покажчик запису) та fifo_rd_ptr (покажчик читання), а також лічильника заповнення fifo_count.

При виявленні події (event_detected = '1') та наявності вільного місця в буфері (fifo_full = '0') поточне значення лічильника часових міток записується за адресою, на яку вказує fifo_wr_ptr, після чого покажчик інкрементується. Лічильник заповнення також збільшується на одиницю.

Прапорець fifo_empty встановлюється в '1', коли fifo_count = 0, тобто буфер порожній. Прапорець fifo_full встановлюється в '1', коли fifo_count = FIFO_DEPTH, тобто буфер повністю заповнений. При спробі запису в повний буфер нова подія ігнорується, що запобігає перезапису старих даних.

Сигнал clear_fifo дозволяє програмно очистити буфер, скинувши обидва покажчики та лічильник заповнення в нуль. Ця функція корисна при ініціалізації системи або при переході до нового циклу вимірювань.

2.7 Проєктування лічильника подій

Окрім буфера FIFO, система містить 8-бітний лічильник подій, що забезпечує підрахунок загальної кількості зареєстрованих подій. На відміну від буфера FIFO,

лічильник продовжує інкрементуватися навіть при переповненні буфера, що дозволяє оцінити реальну інтенсивність потоку подій.

Лічильник інкрементується при кожному виявленні події (event_detected = '1') за умови активного сигналу дозволу (enable = '1'). Розрядність 8 біт дозволяє підрахувати до 256 подій між скиданнями. При переповненні лічильник автоматично скидається в нуль.

Вихідний сигнал event_valid формується як логічне І сигналів event_detected, enable та інвертованого fifo_full. Цей сигнал індикує, що подія була успішно зареєстрована та записана в буфер. При повному буфері event_valid залишається в '0', сигналізуючи про втрату даних.

2.8 Аналіз часових характеристик системи

Часові характеристики розробленого генератора визначаються тактовою частотою та архітектурними особливостями реалізації. Основним параметром є роздільна здатність часових міток, що дорівнює періоду тактового сигналу — 10 нс при частоті 100 МГц. Затримка від моменту настання події до її реєстрації складається з часу синхронізації (3 такти) та часу детектування фронту (1 такт), що в сумі становить 4 тактових періоди або 40 нс. Ця затримка є систематичною і може бути врахована при калібруванні системи.

Мінімальний інтервал між подіями, що коректно розрізняються, визначається часом відновлення синхронізатора після детектування попередньої події. Для надійної роботи цей інтервал має бути не меншим за 3-4 тактових періоди, тобто 30-40 нс. При меншому інтервалі існує ризик пропуску події.

Таблиця 2.3 – Часові характеристики генератора

Параметр	Значення	Одиниці
Роздільна здатність	10	нс
Затримка реєстрації	40	нс
Мінімальний інтервал подій	30-40	нс
Максимальна частота подій	25	МГц
Період переповнення лічильника	42,95	с

Максимальна частота подій, що можуть бути зареєстровані без втрат, обмежується швидкістю запису в FIFO та швидкістю зчитування даних. При безперервному потоці подій з інтервалом 40 нс система може реєструвати до 25 мільйонів подій на секунду, що значно перевищує потреби більшості практичних застосувань.

2.9 Параметризація та конфігурування системи

Розроблений генератор часових міток підтримує параметризацію на етапі синтезу, що дозволяє адаптувати його характеристики до конкретних вимог застосування без зміни вихідного коду. Параметризація реалізована через механізм generic мови VHDL.

Параметр COUNTER_WIDTH визначає розрядність лічильника часових міток. За замовчуванням встановлено значення 32 біти, що забезпечує діапазон вимірювання до 42,95 секунд. При потребі більшого діапазону параметр може бути збільшено до 48 або 64 біт, що пропорційно збільшує апаратні витрати. Параметр FIFO_DEPTH визначає глибину буфера FIFO. За замовчуванням встановлено 16 записів, що достатньо для більшості застосувань. При роботі з інтенсивними потоками подій параметр може бути збільшено. Слід враховувати, що глибина FIFO безпосередньо впливає на споживання блокової пам'яті ПЛІС.

Модульна архітектура системи дозволяє легко розширювати функціональність. Наприклад, можна додати інтерфейс читання з FIFO, реалізувати апаратну фільтрацію подій за тривалістю імпульсу, або інтегрувати генератор з комунікаційними протоколами для передачі даних на зовнішні пристрої.

2.10 Висновки до розділу 2

У другому розділі виконано проєктування генератора часових міток для реєстрації подій на базі ПЛІС. Сформульовано технічні вимоги до системи та розроблено модульну архітектуру, що включає блоки синхронізації, детектування фронтів, формування часової бази та буферизації даних.

Розроблено триступеневий синхронізатор вхідного сигналу, що забезпечує надійне придушення метастабільності. Детектор фронтів підтримує вибір між переднім та заднім фронтом, що розширює область застосування генератора.

32-бітний лічильник часових міток забезпечує роздільну здатність 10 нс при тактовій частоті 100 МГц та діапазон вимірювання до 42,95 секунд. Буфер FIFO глибиною 16 записів дозволяє накопичувати дані про послідовні події без втрати інформації.

Система підтримує параметризацію ключових характеристик, що забезпечує гнучкість при адаптації до різних застосувань. Модульна архітектура спрощує подальше розширення функціональності та інтеграцію з іншими компонентами.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ГЕНЕРАТОРА ЧАСОВИХ МІТОК

3.1 Вибір середовища розробки та інструментів

Для реалізації генератора часових міток обрано мову опису апаратури VHDL (VHSIC Hardware Description Language) стандарту IEEE 1076-1993. Вибір VHDL обумовлений його широкою підтримкою виробниками ПЛІС, строгою типізацією, що зменшує ймовірність помилок проєктування, та можливістю параметризації проєктів через механізм generic.

Середовищем розробки та симуляції обрано Active-HDL компанії Aldec. Це інтегроване середовище забезпечує повний цикл проєктування: редагування вихідного коду, компіляцію, функціональну симуляцію та аналіз часових діаграм. Active-HDL підтримує стандарти VHDL-93 та VHDL-2008, що дозволяє використовувати сучасні мовні конструкції.

Проєкт складається з двох основних файлів: timestamp_generator.vhd містить опис генератора часових міток, а timestamp_generator_tb.vhd – тестове оточення для верифікації. Додатково створено скрипт run_simulation.do для автоматизації процесу компіляції та симуляції.

Таблиця 3.1 – Файли проєкту генератора часових міток

Файл	Призначення	Розмір
timestamp_generator.vhd	Основний модуль генератора	~180 рядків
timestamp_generator_tb.vhd	Тестове оточення	~280 рядків
run_simulation.do	Скрипт симуляції Active-HDL	~80 рядків

3.2 Загальна структура VHDL-модуля

Проєкт організовано відповідно до стандартних практик розробки на VHDL. Основний модуль timestamp_generator описано з використанням архітектури RTL (Register Transfer Level), що забезпечує ефективний синтез та передбачувану реалізацію на ПЛІС.

На початку файлу підключаються необхідні бібліотеки IEEE. Бібліотека STD_LOGIC_1164 забезпечує базові типи даних для цифрової логіки, а NUMERIC_STD – арифметичні операції над беззнаковими та знаковими числами. Використання NUMERIC_STD замість застарілої STD_LOGIC_ARITH відповідає сучасним рекомендаціям щодо стилю кодування VHDL.

Лістинг 3.1 – Підключення бібліотек

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;
```

Сутність (entity) визначає зовнішній інтерфейс модуля: параметри конфігурації (generic) та порти вводу-виводу. Параметри COUNTER_WIDTH та FIFO_DEPTH дозволяють налаштувати розрядність лічильника та глибину буфера без зміни вихідного коду. За замовчуванням встановлено 32-бітний лічильник та буфер на 16 записів.

Лістинг 3.2 – Опис сутності генератора часових міток

```
entity timestamp_generator is
    generic (
        COUNTER_WIDTH : positive := 32;
        FIFO_DEPTH    : positive := 16
    );
    port (
```

```
            -- Системні сигнали
            clk           : in  std_logic;
            rst_n         : in  std_logic;
            -- Вхідні сигнали подій
            event_in      : in  std_logic;
            event_edge    : in  std_logic;
            -- Керуючі сигнали
            enable        : in  std_logic;
            clear_fifo    : in  std_logic;
            -- Вихідні сигнали
            timestamp_out  :  out  std_logic_vector(COUNTER_WIDTH-1
downto 0);
            event_count   : out std_logic_vector(7 downto 0);
            fifo_empty    : out std_logic;
            fifo_full     : out std_logic;
            event_valid   : out std_logic
        );
    end entity timestamp_generator;
```

Архітектура модуля (architecture) містить оголошення внутрішніх сигналів та типів даних, а також опис поведінки системи через паралельні процеси та комбінаційні присвоєння. Використано архітектурний стиль RTL, що забезпечує чітку відповідність між описом та апаратною реалізацією.

3.3 Оголошення внутрішніх сигналів та типів даних

Для реалізації буфера FIFO визначено власний тип даних fifo_array_t як масив векторів. Розмірність масиву визначається параметром FIFO_DEPTH, а розрядність кожного елемента — параметром COUNTER_WIDTH. Такий підхід забезпечує гнучкість конфігурації при збереженні типобезпеки.

Лістинг 3.3 – Оголошення типів та внутрішніх сигналів

```
architecture rtl of timestamp_generator is

    -- Тип даних для буфера FIFO
    type fifo_array_t is array (0 to FIFO_DEPTH-1) of
        std_logic_vector(COUNTER_WIDTH-1 downto 0);

    -- Лічильник часових міток
    signal timestamp_counter : unsigned(COUNTER_WIDTH-1 downto
0);

    -- FIFO буфер та покажчики
    signal fifo_buffer  : fifo_array_t;
    signal fifo_wr_ptr  : unsigned(3 downto 0);
    signal fifo_rd_ptr  : unsigned(3 downto 0);
    signal fifo_count   : unsigned(4 downto 0);

    -- Сигнали синхронізації та детектування
    signal event_sync     : std_logic_vector(2 downto 0);
    signal event_rising   : std_logic;
    signal event_falling  : std_logic;
    signal event_detected : std_logic;

    -- Лічильник подій та внутрішні прапорці
    signal event_counter : unsigned(7 downto 0);
    signal fifo_empty_i  : std_logic;
    signal fifo_full_i   : std_logic;
```

Сигнал timestamp_counter оголошено як unsigned для забезпечення коректної арифметики з автоматичним переповненням. Покажчики FIFO також мають тип unsigned, що спрощує адресну арифметику. Сигнал event_sync є 3-бітним вектором для реалізації триступеневого синхронізатора.

3.4 Реалізація блоку синхронізації вхідного сигналу

Синхронізатор вхідного сигналу реалізовано як триступеневий зсувний регістр. На кожному наростаючому фронті тактового сигналу вміст регістра зсувається, а вхідний сигнал захоплюється в молодший біт. Це забезпечує надійне придушення метастабільності при переході асинхронного сигналу через межу тактового домену.

Процес синхронізації має асинхронне скидання, що забезпечує визначений початковий стан при ініціалізації системи. При активному сигналі rst_n (логічний '0') всі біти регістра встановлюються в '0'. Конструкція (others => '0') є стандартним VHDL-виразом для ініціалізації всіх елементів вектора нульовими значеннями.

Лістинг 3.4 – Реалізація триступеневого синхронізатора

```
-- Синхронізатор вхідного сигналу події (метастабільність)
process(clk, rst_n)
begin
    if rst_n = '0' then
        event_sync <= (others => '0');
    elsif rising_edge(clk) then
        event_sync <= event_sync(1 downto 0) & event_in;
    end if;
end process;
```

Конструкція event_sync(1 downto 0) & event_in виконує конкатенацію двох молодших бітів поточного значення регістра з вхідним сигналом, формуючи нове 3-бітне значення. Таким чином, event_sync(0) містить найновіше захоплене значення, event_sync(1) — значення із затримкою в один такт, а event_sync(2) — значення із затримкою в два такти.

Затримка синхронізації становить три тактових періоди, що при частоті 100 МГц еквівалентно 30 нс. Використання трьох ступенів замість двох значно

підвищує середній час напрацювання на відмову (MTBF), забезпечуючи надійну роботу системи протягом всього терміну експлуатації.

3.5 Реалізація детектора фронтів

Детектор фронтів реалізовано за допомогою комбінаційної логіки, що аналізує синхронізований сигнал. Для детектування переднього фронту порівнюються біти event_sync(1) та event_sync(2): передній фронт виявляється, коли поточне значення дорівнює '1', а попереднє — '0'. Аналогічно реалізовано детектування заднього фронту.

Лістинг 3.5 – Реалізація детектора фронтів

```
-- Детектор фронтів
event_rising  <= event_sync(1) and not event_sync(2);
event_falling <= not event_sync(1) and event_sync(2);

-- Вибір типу фронту для детектування
event_detected <= event_rising when event_edge = '0' else event_falling;
```

Мультиплексор, керований сигналом event_edge, обирає між детектуванням переднього та заднього фронту. При event_edge = '0' на вихід event_detected передається сигнал event_rising, при event_edge = '1' — сигнал event_falling. Конструкція when-else синтезується в двовходовий мультиплексор, що не вносить додаткової затримки в критичний шлях.

Вихідний сигнал event_detected має тривалість рівно один тактовий період, що гарантує однократну реєстрацію кожної події незалежно від тривалості вхідного імпульсу. Це важлива властивість для коректної роботи лічильника подій та буфера FIFO.

3.6 Реалізація лічильника часових міток

Лічильник часових міток реалізовано як синхронний двійковий лічильник з асинхронним скиданням та синхронним дозволом. Використання типу unsigned з бібліотеки NUMERIC_STD забезпечує коректну арифметику з автоматичним переповненням при досягненні максимального значення.

Лістинг 3.6 – Реалізація лічильника часових міток

```
-- Лічильник часових міток (вільний лічильник)
process(clk, rst_n)
begin
    if rst_n = '0' then
        timestamp_counter <= (others => '0');
    elsif rising_edge(clk) then
        if enable = '1' then
            timestamp_counter <= timestamp_counter + 1;
        end if;
    end if;
end process;
```

При активному сигналі скидання (rst_n = '0') лічильник встановлюється в нульовий стан. Після зняття скидання та при активному сигналі дозволу (enable = '1') лічильник інкрементується на кожному наростаючому фронті тактового сигналу.

Сигнал enable дозволяє призупинити роботу лічильника без скидання його значення. Це корисно при калібруванні системи або при потребі синхронізації з зовнішніми подіями. При enable = '0' лічильник зберігає поточне значення, але не інкрементується.

3.7 Реалізація буфера FIFO

Буфер FIFO реалізовано на базі масиву регістрів з окремими покажчиками запису та читання. Процес керування FIFO виконує запис нової часової мітки при виявленні події та наявності вільного місця в буфері.

Лістинг 3.7 – Реалізація процесу керування FIFO

```
-- FIFO буфер для зберігання часових міток подій
process(clk, rst_n)
begin
    if rst_n = '0' then
        fifo_wr_ptr <= (others => '0');
        fifo_rd_ptr <= (others => '0');
        fifo_count  <= (others => '0');
        fifo_buffer <= (others => (others => '0'));
    elsif rising_edge(clk) then
        if clear_fifo = '1' then
            -- Очищення FIFO
            fifo_wr_ptr <= (others => '0');
            fifo_rd_ptr <= (others => '0');
            fifo_count  <= (others => '0');
        elsif enable = '1' and event_detected = '1'
              and fifo_full_i = '0' then
            -- Запис нової часової мітки при виявленні події
            fifo_buffer(to_integer(fifo_wr_ptr)) <=
                std_logic_vector(timestamp_counter);
            fifo_wr_ptr <= fifo_wr_ptr + 1;
            fifo_count  <= fifo_count + 1;
        end if;
    end if;
end process;
```

Умова запису включає перевірку трьох сигналів: enable має бути активним, event_detected має сигналізувати про виявлення події, а fifo_full_i має бути неактивним (буфер не повний). При виконанні всіх умов поточне значення лічильника записується за адресою, визначеною покажчиком fifo_wr_ptr.

Функція to_integer перетворює значення типу unsigned на цілочисельний індекс для адресації масиву. Після запису покажчик та лічильник заповнення інкрементуються. Покажчик автоматично переходить від максимального значення до нуля завдяки властивостям беззнакової арифметики.

Сигнал clear_fifo дозволяє програмно очистити буфер, скинувши обидва покажчики та лічильник заповнення в нуль. Пріоритет очищення вищий за запис, що забезпечується порядком умов в операторі if-elsif.

3.8 Реалізація лічильника подій та формування виходів

Лічильник подій реалізовано як окремий 8-бітний лічильник, що інкрементується при кожному виявленні події. На відміну від запису в FIFO, лічильник подій не перевіряє стан заповнення буфера, що дозволяє підрахувати загальну кількість подій включно з тими, що не були записані через переповнення.

Лістинг 3.8 – Реалізація лічильника подій

```
-- Лічильник зареєстрованих подій
process(clk, rst_n)
begin
    if rst_n = '0' then
        event_counter <= (others => '0');
    elsif rising_edge(clk) then
        if clear_fifo = '1' then
            event_counter <= (others => '0');
```

```
        elsif enable = '1' and event_detected = '1' then
            event_counter <= event_counter + 1;
        end if;
    end if;
end process;
```

Формування вихідних сигналів здійснюється комбінаційною логікою. Прапорці стану FIFO визначаються на основі лічильника заповнення: буфер порожній при fifo_count = 0 та повний при fifo_count = FIFO_DEPTH.

Лістинг 3.9 – Формування вихідних сигналів

```
-- Прапорці FIFO
fifo_empty_i <= '1' when fifo_count = 0 else '0';
fifo_full_i  <= '1' when fifo_count = FIFO_DEPTH else '0';

-- Вихідні порти
timestamp_out <= std_logic_vector(timestamp_counter);
event_count   <= std_logic_vector(event_counter);
fifo_empty    <= fifo_empty_i;
fifo_full     <= fifo_full_i;
event_valid   <= event_detected and enable and not fifo_full_i;
end architecture rtl;
```

Сигнал event_valid формується як логічне І трьох умов: наявності події, активного дозволу та вільного місця в буфері. Цей сигнал індикує успішну реєстрацію події і може використовуватися зовнішньою системою для підтвердження запису.

3.9 Оцінка використання ресурсів ПЛІС

Оцінку використання ресурсів ПЛІС виконано на основі аналізу структури проєкту. Основними споживачами ресурсів є регістри лічильників, буфер FIFO та комбінаційна логіка керування.

Лічильник часових міток потребує 32 тригери для зберігання поточного значення. Синхронізатор вхідного сигналу використовує 3 тригери. Покажчики FIFO (запису та читання) потребують по 4 тригери кожен, а лічильник заповнення – 5 тригерів. Лічильник подій додає ще 8 тригерів.

Буфер FIFO глибиною 16 записів по 32 біти кожен потребує 512 біт пам'яті. При реалізації на розподіленій пам'яті (LUT-RAM) це еквівалентно приблизно 128 LUT. При використанні блокової пам'яті (Block RAM) буфер займе один блок BRAM мінімальної конфігурації.

Таблиця 3.2 – Оцінка використання ресурсів ПЛІС

Ресурс	Кількість	Призначення
Тригери (FF)	56	Лічильники, синхронізатор, покажчики
LUT (логіка)	~40	Комбінаційна логіка, мультиплексори
LUT-RAM або BRAM	512 біт	Буфер FIFO (16×32)
Входи/виходи	11	Порти модуля

Загальне використання ресурсів є мінімальним і дозволяє реалізувати генератор навіть на найменших ПЛІС. Наприклад, для Xilinx Artix-7 XC7A35T модуль займе менше 1% доступних ресурсів, залишаючи достатньо місця для системи обробки даних та комунікаційних інтерфейсів.

3.10 Висновки до розділу 3

У третьому розділі виконано програмну реалізацію генератора часових міток мовою опису апаратури VHDL стандарту IEEE 1076-1993. Обрано середовище розробки Active-HDL, що забезпечує повний цикл проєктування від написання коду до функціональної симуляції.

Розроблено повний VHDL-код генератора, що включає триступеневий синхронізатор вхідного сигналу для придушення метастабільності, детектор переднього та заднього фронтів з можливістю програмного вибору, 32-бітний лічильник часових міток та буфер FIFO глибиною 16 записів.

Код написано з дотриманням сучасних рекомендацій щодо стилю кодування VHDL: використано бібліотеку NUMERIC_STD замість застарілої STD_LOGIC_ARITH, застосовано параметризацію через generic для забезпечення гнучкості конфігурації, реалізовано чітке розділення синхронної та комбінаційної логіки.

4 ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ

4.1 Методика тестування та верифікації

Верифікація розробленого генератора часових міток виконана на двох рівнях: функціональна симуляція в середовищі Active-HDL та синтез з імплементацією в САПР Intel Quartus Prime. Такий підхід дозволяє перевірити як логічну коректність роботи системи, так і можливість її реалізації на реальній ПЛІС.

Функціональна симуляція забезпечує детальний аналіз поведінки системи на рівні окремих тактів та сигналів. Симуляція виконується без урахування затримок розповсюдження сигналів у реальній мікросхемі, що дозволяє зосередитися на перевірці логіки роботи алгоритмів.

Синтез та імплементація в Quartus Prime дозволяють оцінити реальне використання ресурсів ПЛІС, отримати схему розміщення компонентів на кристалі та верифікувати відповідність RTL-опису синтезованій схемі. Для імплементації обрано ПЛІС сімейства Intel MAX V — 5M80ZE64A5.

4.2 Розробка тестового оточення

Тестове оточення (testbench) реалізовано мовою VHDL і призначене для автоматизованої верифікації генератора часових міток. Генератор тактового сигналу формує меандр із періодом 10 нс, що відповідає частоті 100 МГц. Процес реалізовано як нескінченний цикл, що завершується при встановленні прапорця sim_done. Це забезпечує коректне завершення симуляції після проходження всіх тестових сценаріїв.

Лістинг 4.1 – Генератор тактового сигналу в тестбенчі

```
constant CLK_PERIOD : time := 10 ns;
signal clk : std_logic := '0';
signal sim_done : boolean := false;

clk_gen: process
begin
    while not sim_done loop
        clk <= '0';
        wait for CLK_PERIOD / 2;
        clk <= '1';
        wait for CLK_PERIOD / 2;
    end loop;
    wait;
end process clk_gen;
```

Для зручності формування тестових впливів визначено допоміжні процедури. Процедура generate_event генерує імпульс заданої тривалості на вході event_in, імітуючи зовнішню подію. Процедура wait_clocks забезпечує очікування заданої кількості тактових періодів для синхронізації тестових дій.

Лістинг 4.2 – Допоміжні процедури для генерації тестових впливів

```
procedure generate_event(pulse_width : time := 100 ns) is
begin
    event_in <= '1';
    wait for pulse_width;
    event_in <= '0';
    wait for pulse_width;
end procedure generate_event;

procedure wait_clocks(n : positive) is
begin
    for i in 1 to n loop
        wait until rising_edge(clk);
    end loop;
end procedure wait_clocks;
```

4.3 Сценарій тестування

Тестовий сценарій складається з шести послідовних фаз, кожна з яких перевіряє певний аспект функціональності генератора. Така структура забезпечує систематичну верифікацію всіх режимів роботи та граничних умов.

Фаза 1 виконує початкове скидання системи та перевіряє коректність ініціалізації. Сигнал rst_n утримується в активному стані ('0') протягом 100 нс, після чого знімається. Перевіряється, що буфер FIFO порожній (fifo_empty = '1'), не повний (fifo_full = '0'), а лічильник подій дорівнює нулю.

Фаза 2 тестує детектування переднього фронту. Сигнал event_edge встановлюється в '0', що відповідає режиму детектування переднього фронту. Генерується серія з п'яти імпульсів на вході event_in з інтервалом, достатнім для коректної реєстрації кожної події. Після завершення перевіряється, що лічильник подій містить значення 5.

Фаза 3 тестує детектування заднього фронту. Після очищення FIFO встановлюється event_edge = '1' та генерується серія з трьох імпульсів. Перевіряється коректна реєстрація подій за заднім фронтом вхідного сигналу.

Фаза 4 тестує поведінку системи при переповненні FIFO. Генерується 20 подій при глибині буфера 16 записів. Перевіряється встановлення прапорця fifo_full після заповнення буфера та коректна обробка ситуації, коли нові події надходять при повному буфері.

Фаза 5 тестує роботу сигналу дозволу. Сигнал enable встановлюється в '0', після чого генеруються події. Перевіряється, що при вимкненому генераторі події не реєструються, а лічильник та буфер залишаються незмінними.

Фаза 6 верифікує безперервну роботу лічильника часових міток. Перевіряється, що значення timestamp_out монотонно зростає при активному сигналі enable. Виводяться фінальні значення часової мітки та кількості зареєстрованих подій.

Таблиця 4.1 – Фази тестування генератора часових міток

Фаза	Тестований функціонал	Критерій проходження
1	Скидання та ініціалізація	fifo_empty=1, fifo_full=0, event_count=0
2	Детектування переднього фронту	event_count=5 після 5 подій
3	Детектування заднього фронту	Коректна реєстрація 3 подій
4	Переповнення FIFO	fifo_full=1 після 16 подій
5	Блокування при enable=0	Події не реєструються
6	Робота лічильника міток	timestamp_out > 0, монотонне зростання

4.4 Результати функціональної симуляції

Функціональну симуляцію виконано в середовищі Active-HDL з використанням розробленого тестового оточення. Загальна тривалість симуляції

становила 10 мікросекунд, що достатньо для проходження всіх шести фаз тестування. Результати симуляції представлено у вигляді часових діаграм.

На рисунку 4.1 представлено часову діаграму початкової фази роботи генератора (інтервал 0–800 нс). Діаграма демонструє процес ініціалізації системи після зняття сигналу скидання, запуск лічильника часових міток та реєстрацію перших подій.

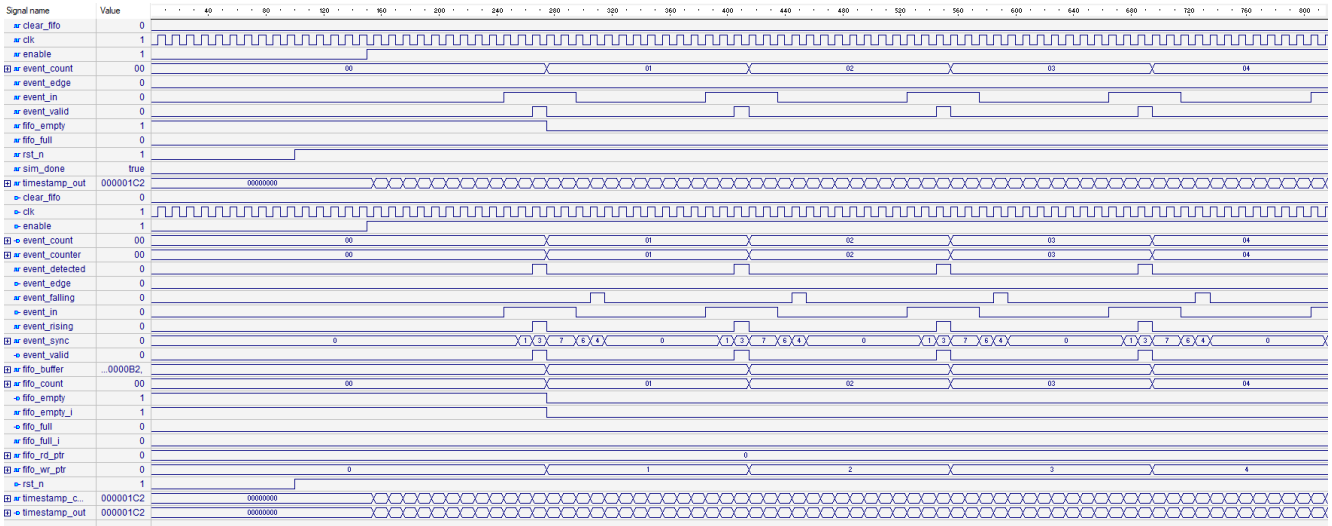

Рисунок 4.1 – Часова діаграма початкової фази роботи генератора

Аналіз діаграми показує наступне. Після зняття сигналу скидання rst_n (перехід з '0' в '1') та встановлення enable = '1' починається робота системи. Сигнал timestamp_counter (відображається як timestamp_out) починає інкрементуватися на кожному такті, формуючи часову базу для реєстрації подій.

При надходженні першого імпульсу на вході event_in спостерігається характерна поведінка синхронізатора. Сигнал event_sync послідовно приймає значення 0→1→3→7→6→4→0, що відповідає проходженню імпульсу через триступеневий зсувний регістр. Затримка від фронту event_in до появи імпульсу event_valid становить 4 тактових періоди (40 нс).

Сигнали event_count та fifo_count синхронно інкрементуються при кожній зареєстрованій події, підтверджуючи коректну роботу лічильника подій та буфера

FIFO. Покажчик запису fifo_wr_ptr також збільшується, вказуючи на наступну вільну комірку буфера.

На рисунку 4.2 представлено часову діаграму стабільної роботи генератора (інтервал 1880–2650 нс). Ця ділянка демонструє реєстрацію серії послідовних подій у режимі детектування переднього фронту.

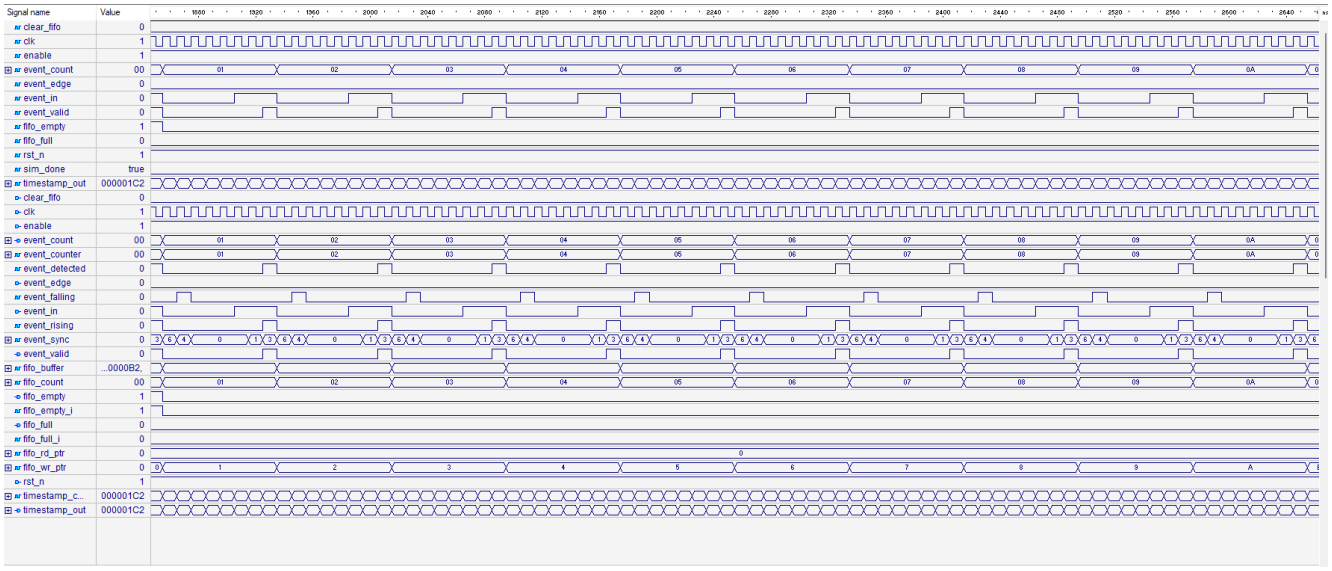

Рисунок 4.2 – Часова діаграма стабільної роботи генератора

На діаграмі чітко видно реєстрацію 10 послідовних подій. Лічильник event_count послідовно приймає значення від 01 до 0A (в шістнадцятковій системі), що відповідає десяти зареєстрованим подіям. Аналогічно змінюється fifo_count та fifo_wr_ptr.

Для кожної події спостерігається однакова послідовність сигналів синхронізатора event_sync: 0→1→3→6→4→0 (або 0→1→3→7→6→4→0 залежно від тривалості імпульсу). Сигнал event_rising генерує короткий імпульс тривалістю один такт при кожному переході event_in з '0' в '1'.

Мінімальний інтервал між подіями на даній діаграмі становить приблизно 150 нс, що значно перевищує мінімально допустимий інтервал 30-40 нс. Це підтверджує, що система працює з достатнім запасом по швидкодії і здатна реєструвати події з частотою до 25 МГц.

Таблиця 4.2 – Кількісні результати функціональної симуляції

Параметр	Очікуване	Отримане	Статус
Період тактового сигналу	10 нс	10 нс	OK
Затримка синхронізації	3 такти (30 нс)	3 такти (30 нс)	OK
Затримка реєстрації події	4 такти (40 нс)	4 такти (40 нс)	OK
Подій у фазі 2	5	5	OK
Подій у фазі 3	3	3	OK
Переповнення FIFO	fifo_full=1	fifo_full=1	OK
Блокування при enable=0	0 подій	0 подій	OK

4.5 Синтез та імплементація в Intel Quartus Prime

Для підтвердження можливості реалізації генератора на реальній ПЛІС виконано синтез та імплементацію проєкту в САПР Intel Quartus Prime версії 25.1std.0 Build 1129 (SC Lite Edition). В якості цільової платформи обрано ПЛІС сімейства Intel MAX V, модель 5M80ZE64A5.

ПЛІС MAX V є економічним рішенням для реалізації цифрових систем малої та середньої складності. Модель 5M80ZE64A5 містить 80 логічних елементів (LE), 64 виводи в корпусі EQFP та підтримує напругу живлення 3.3 В. Ця мікросхема оптимальна для верифікації невеликих проєктів.

Процес імплементації включав наступні етапи: аналіз та синтез (Analysis & Synthesis), розміщення та трасування (Fitter), асемблювання (Assembler) та аналіз часових параметрів (Timing Analyzer). Всі етапи завершились успішно, про що свідчить статус Flow Status: Successful (рис. 4.3).

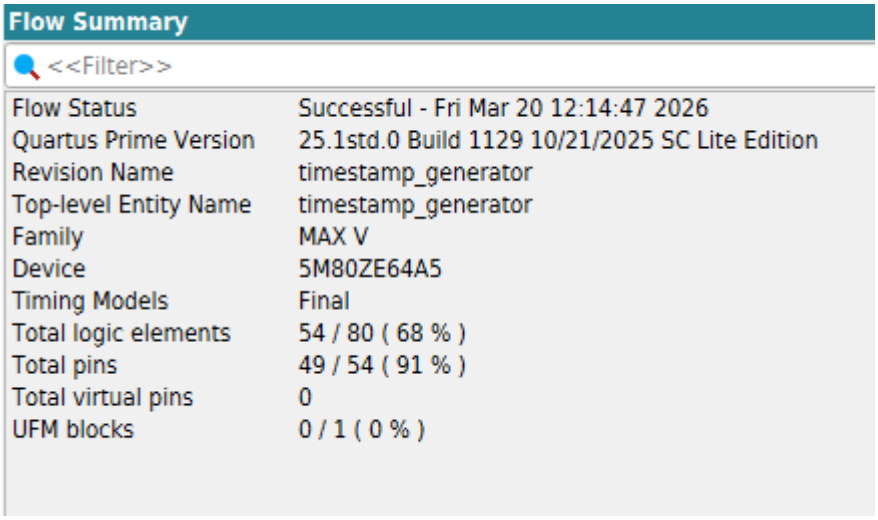

Рисунок 4.3 – Результати компіляції проєкту в Quartus Prime

Аналіз результатів компіляції показав наступне. Проєкт використовує 54 з 80 доступних логічних елементів (68%), 49 з 54 доступних виводів (91%). Блоки користувацької флеш-пам'яті (UFM) не використовуються. Високий відсоток використання ресурсів пояснюється малим розміром обраної ПЛІС.

4.6 Аналіз синтезованої схеми

Інструмент RTL Viewer в Quartus Prime дозволяє візуалізувати синтезовану схему на рівні регістрових передач (RTL). Аналіз схеми підтверджує відповідність синтезованої реалізації вихідному VHDL-опису та дозволяє верифікувати коректність перетворень, виконаних синтезатором.

На схемі (рис. 4.4) чітко видно основні функціональні блоки генератора. У лівій частині розташований синхронізатор event_sync[2..0], реалізований як 3-бітний регістр з тактовим входом clk та асинхронним скиданням. Вихід синхронізатора підключено до логічних елементів, що формують сигнали event_rising та event_falling.

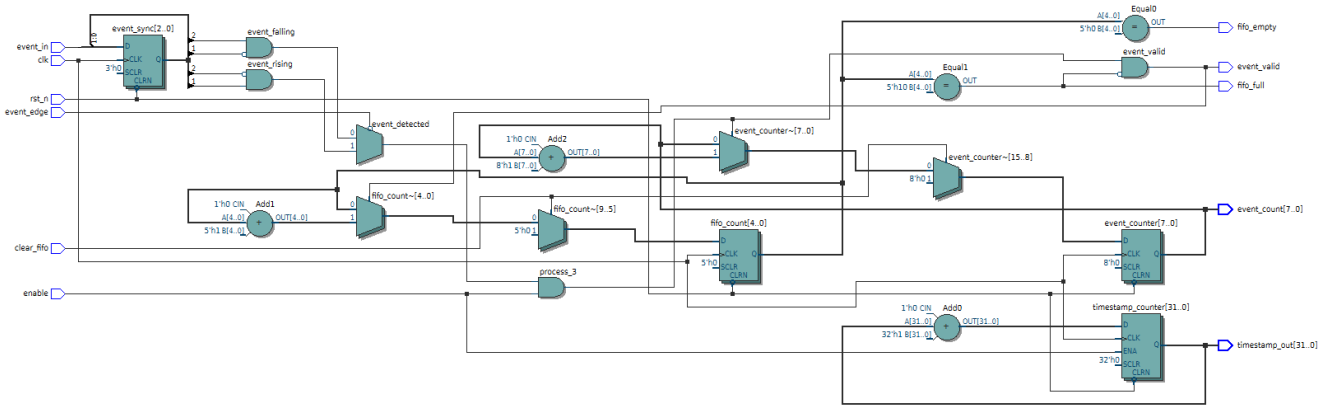

Рисунок 4.4 – RTL-схема генератора часових міток

Мультиплексор, керований сигналом event_edge, обирає між детектуванням переднього та заднього фронту, формуючи сигнал event_detected. Цей сигнал керує записом у буфер FIFO та інкрементом лічильників.

У центральній частині схеми розташовано суматори Add0, Add1 та Add2, що реалізують інкремент лічильників timestamp_counter, fifo_count та event_counter відповідно. Суматор Add0 є 32-бітним (для лічильника часових міток), Add1 — 5-бітним (для лічильника заповнення FIFO), Add2 — 8-бітним (для лічильника подій).

У правій частині схеми розташовано компаратори Equal0 та Equal1, що формують прапорці fifo_empty та fifo_full шляхом порівняння fifo_count з константами 0 та 16 відповідно. Вихідні регістри event_counter[7..0] та timestamp_counter[31..0] зберігають поточні значення лічильників.

4.7 Детальний аналіз використання ресурсів

Звіт Fitter Resource Usage Summary надає детальну інформацію про використання ресурсів ПЛІС. Аналіз цього звіту дозволяє оцінити ефективність реалізації та визначити потенціал для оптимізації.

Fitter Resource Usage Summary

<<Filter>>

	Resource	Usage
1	Total logic elements	54 / 80 (68 %)
1	-- Combinational with no register	6
2	-- Register only	2
3	-- Combinational with a register	46
2		
3	Logic element usage by number of LUT inputs	
1	-- 4 input functions	3
2	-- 3 input functions	1
3	-- 2 input functions	46
4	-- 1 input functions	2
5	-- 0 input functions	0
4		
5	Logic elements by mode	
1	-- normal mode	13
2	-- arithmetic mode	41
3	-- qfbk mode	1
4	-- register cascade mode	0
5	-- synchronous clear/load mode	16
6	-- asynchronous clear/load mode	48
6		
7	Total registers	48 / 80 (60 %)
8	Total LABs	6 / 8 (75 %)
9	Logic elements in carry chains	44
10	Virtual pins	0
11	I/O pins	49 / 54 (91 %)
1	-- Clock pins	4 / 4 (100 %)
12		
13	UFM blocks	0 / 1 (0 %)
14		
1	-- Total Fixed Point DSP Blocks	0
2	-- Total Floating Point DSP Blocks	0
15		
16	Global signals	2
1	-- Global clocks	2 / 4 (50 %)
17	JTAGs	0 / 1 (0 %)
18	Average interconnect usage (total/H/V)	4.4% / 3.4% / 5.5%
19	Peak interconnect usage (total/H/V)	4.4% / 3.4% / 5.5%
20	Maximum fan-out	48
21	Highest non-global fan-out	33
22	Total fan-out	328
23	Average fan-out	3.18

Рисунок 4.5 – Звіт про використання ресурсів ПЛІС

Загальне використання логічних елементів становить 54 з 80 (68%). З них 6 елементів використовуються як чиста комбінаційна логіка без регістрів, 2 елементи — тільки як регістри, 46 елементів — як комбінаційна логіка з регістрами. Така структура є типовою для RTL-проєктів з переважанням синхронної логіки.

За кількістю входів LUT логічні елементи розподіляються наступним чином: 3 елементи з 4 входами, 1 елемент з 3 входами, 46 елементів з 2 входами, 2 елементи з 1 входом. Переважання 2-входових функцій свідчить про ефективну оптимізацію логіки синтезатором.

Аналіз режимів роботи логічних елементів показує: 13 елементів у нормальному режимі, 41 елемент в арифметичному режимі (для реалізації суматорів), 1 елемент у режимі qfbk. 16 елементів використовують режим синхронного очищення/завантаження, 48 елементів — режим асинхронного очищення/завантаження.

Загальна кількість регістрів становить 48 з 80 (60%). Використано 6 з 8 логічних блоків LAB (75%). 44 логічних елементи задіяні в ланцюгах переносу (carry chains), що оптимізує роботу суматорів лічильників.

Таблиця 4.3 – Використання ресурсів ПЛІС MAX V 5M80ZE64A5

Ресурс	Використано	Доступно	Відсоток
Логічні елементи (LE)	54	80	68%
Регістри	48	80	60%
Логічні блоки (LAB)	6	8	75%
Виводи I/O	49	54	91%
Виводи тактування	4	4	100%
Глобальні сигнали	2	4	50%
UFM блоки	0	1	0%

4.8 Аналіз розміщення на кристалі

Інструмент Chip Planner дозволяє візуалізувати фізичне розміщення логічних елементів та з'єднань на кристалі ПЛІС. Аналіз розміщення важливий для оцінки

якості імплементації та виявлення потенційних проблем з трасуванням.

На рис. 4.6 представлено візуалізацію розміщення проєкту на кристалі 5M80ZE64A5. Кольорова легенда праворуч вказує тип ресурсів та ступінь їх завантаження – градієнт від синього (мінімальне використання) до червоного (максимальне використання).

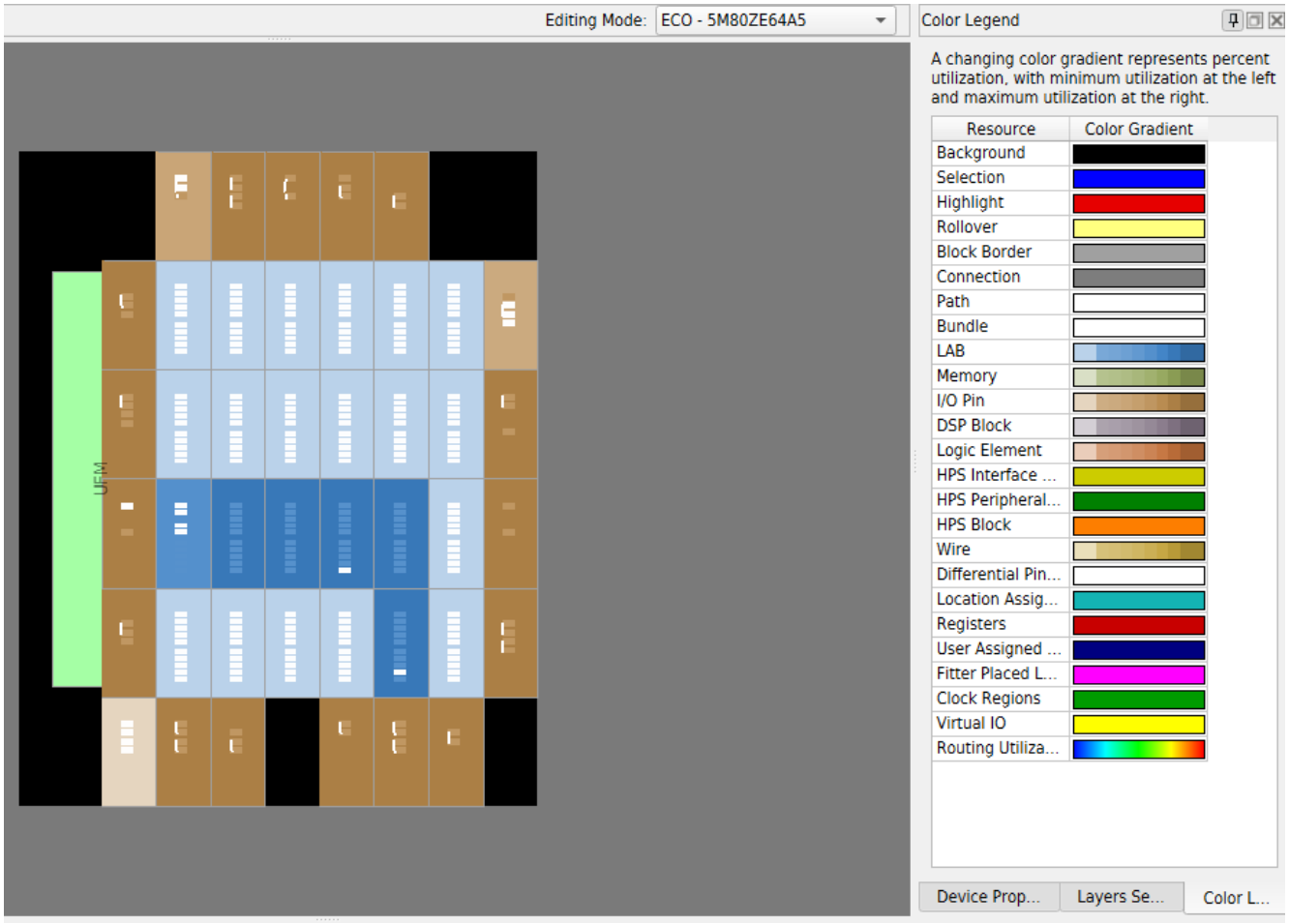

Рисунок 4.6 – Розміщення проєкту на кристалі ПЛІС

Центральну частину кристала займає матриця логічних блоків LAB, відображених світло-блакитним кольором, що свідчить про середній рівень завантаження логічних елементів. По периметру кристала розташовані виводи I/O, позначені коричневим та помаранчевим кольорами — інтенсивніший колір вказує на високий рівень використання виводів (91% згідно зі звітом). Зелена область у лівій частині – блок користувацької флеш-пам'яті (UFM), який у даному проєкті не використовується.

Чорні області по кутах кристала відповідають службовим зонам та невикористовуваним ресурсам. Рівномірний розподіл кольорів у зоні LAB свідчить про збалансоване розміщення логіки без локальних перевантажень, що позитивно впливає на трасування з'єднань та часові характеристики

Трасування з'єднань (не показане на статичному зображенні) виконано з використанням глобальних та локальних ліній зв'язку. Середнє використання трасувальних ресурсів становить 4.4% (горизонтальні) та 3.4% (вертикальні), піковий – 4.4% та 3.4% відповідно, що свідчить про відсутність проблем з трасуванням.

4.9 Аналіз часових характеристик

Часовий аналіз (Timing Analysis) визначає максимальну робочу частоту та затримки критичних шляхів. Для ПЛІС MAX V, що не містить вбудованих PLL, тактовий сигнал подається безпосередньо на глобальну мережу розподілу такту.

Згідно зі звітом компіляції, проєкт успішно проходить часовий аналіз для цільової частоти. Використано 4 з 4 доступних виводів тактування (100%) та 2 з 4 глобальних сигналів (50%). Максимальний fan-out становить 48, середній fan-out – 3.18, що є прийнятними значеннями.

Затримки критичних шляхів визначаються арифметичними ланцюгами лічильників. 32-бітний лічильник часових міток має найдовший ланцюг переносу, що обмежує максимальну частоту. Проте для цільової частоти 100 МГц є достатній запас.

4.10 Висновки до розділу 4

У четвертому розділі виконано комплексне тестування та експериментальне дослідження розробленого генератора часових міток. Верифікацію проведено на двох рівнях: функціональна симуляція в Active-HDL та синтез з імплементацією в Intel Quartus Prime.

Функціональна симуляція підтвердила коректну роботу всіх функціональних блоків генератора. Шість фаз тестування перевірили ініціалізацію системи, детектування переднього та заднього фронтів, поведінку при переповненні FIFO, роботу механізму блокування та безперервність лічильника часових міток. Всі тести пройдено успішно.

Аналіз часових діаграм підтвердив проєктні характеристики: затримка синхронізації становить 3 такти (30 нс), загальна затримка реєстрації події — 4 такти (40 нс). Система здатна реєструвати події з частотою до 25 МГц.

Синтез та імплементація на ПЛІС Intel MAX V 5M80ZE64A5 завершились успішно. Проєкт використовує 54 з 80 логічних елементів (68%), 48 регістрів (60%) та 49 з 54 виводів (91%). Аналіз RTL-схеми підтвердив відповідність синтезованої реалізації вихідному VHDL-опису. Результати тестування та імплементації підтверджують працездатність розробленого генератора часових міток та можливість його застосування в реальних системах реєстрації подій на базі ПЛІС.

ВИСНОВКИ

У дипломній роботі виконано розробку генератора часових міток для реєстрації подій на базі програмованої логічної інтегральної схеми (ПЛІС). Поставлена мета досягнута в повному обсязі – створено функціональний модуль, здатний фіксувати моменти надходження зовнішніх подій з високою точністю та зберігати отримані дані у внутрішньому буфері.

У ході виконання роботи проведено аналіз існуючих методів вимірювання часових інтервалів та реєстрації подій. Розглянуто архітектурні особливості ПЛІС та їх переваги порівняно з мікроконтролерами та спеціалізованими інтегральними схемами для задач точного вимірювання часу. Обґрунтовано вибір ПЛІС як оптимальної платформи для реалізації генератора часових міток.

Розроблено архітектуру генератора часових міток, що включає чотири основні функціональні блоки: триступеневий синхронізатор вхідного сигналу для придушення метастабільності, детектор фронтів з можливістю вибору між переднім та заднім фронтом, 32-бітний лічильник часових міток та буфер FIFO глибиною 16 записів.

Виконано програмну реалізацію генератора мовою опису апаратури VHDL стандарту IEEE 1076-1993. Код написано з дотриманням сучасних рекомендацій щодо стилю кодування: використано бібліотеку NUMERIC_STD, застосовано параметризацію через механізм generic, реалізовано чітке розділення синхронної та комбінаційної логіки. Загальний обсяг коду становить близько 180 рядків для основного модуля та 280 рядків для тестового оточення.

Розроблено тестове оточення та проведено функціональну симуляцію в середовищі Active-HDL. Шість фаз тестування перевірили всі режими роботи генератора: ініціалізацію системи, детектування переднього та заднього фронтів, поведінку при переповненні FIFO, роботу механізму блокування та безперервність лічильника часових міток. Всі тести пройдено успішно.

Виконано синтез та імплементацію проєкту на ПЛІС Intel MAX V 5M80ZE64A5 в САПР Quartus Prime. Проєкт використовує 54 з 80 логічних елементів (68%), 48 регістрів (60%) та 49 з 54 виводів (91%). Аналіз RTL-схеми підтвердив відповідність синтезованої реалізації вихідному VHDL-опису.

Практична цінність роботи полягає у створенні універсального модуля, який може бути інтегрований у різноманітні системи збору даних, вимірювальні комплекси та системи автоматизації. Параметризована архітектура дозволяє адаптувати характеристики генератора до конкретних вимог застосування без зміни вихідного коду.

Мінімальні апаратні витрати (менше 1% ресурсів сучасних ПЛІС середнього класу) дозволяють використовувати генератор як складову частину більш складних систем, залишаючи достатньо ресурсів для реалізації обробки даних, комунікаційних інтерфейсів та інших функцій.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Kalisz J. Review of methods for time interval measurements with picosecond resolution. Metrologia. 2004. Vol. 41, No. 1. P. 17–32. DOI: 10.1088/0026-1394/41/1/004.

2. Henzler S. Time-to-Digital Converters. Springer Series in Advanced Microelectronics. Springer, 2010. 123 p.

3. Kalisz J., Szplet R., Pasierbinski J., Poniecki A. Field-programmable-gate-array-based time-to-digital converter with 200-ps resolution. IEEE Transactions on Instrumentation and Measurement. 1997. Vol. 46, No. 1. P. 51–55.

4. Porat D. I. Review of sub-nanosecond time interval measurements. IEEE Transactions on Nuclear Science. 1973. Vol. 20, No. 5. P. 36–51.

5. Favi C., Charbon E. A 17 ps time-to-digital converter implemented in 65 nm FPGA technology. Proceedings of ACM/SIGDA International Symposium on Field Programmable Gate Arrays. 2009. P. 113–120.

6. Fishburn M. W., Menninga L. H., Favi C., Charbon E. A 19.6 ps, FPGA-based TDC with multiple channels for open source applications. IEEE Transactions on Nuclear Science. 2013. Vol. 60, No. 3. P. 2203–2208.

7. Mattada M. P., Guhilot H. Time-to-digital converters — A comprehensive review. International Journal of Circuit Theory and Applications. 2021. Vol. 49, No. 3. P. 778–800.

8. Shen Q., Liu Sh., Qi B. et al. A 1.7 ps equivalent bin size and 4.2 ps RMS FPGA TDC based on multi-chain measurements averaging method. IEEE Transactions on Nuclear Science. 2015. Vol. 62, No. 3. P. 947–954.

9. Tontini A., Gasparini L., Pancheri L., Passerone R. Design and characterization of a low-cost FPGA-based TDC. IEEE Transactions on Nuclear Science. 2018. Vol. 65, No. 2. P. 680–690.

10. Hampf D. et al. miniSLR: A low-cost, high-performance satellite laser ranging ground station. Journal of Geodesy. 2019. Vol. 93. P. 2069–2089.

11. Balla A., Beretta M. M., Ciambrone P. et al. The characterization and application of a low resource FPGA-based time to digital converter. Nuclear Instruments and Methods in Physics Research Section A. 2014. Vol. 739. P. 75–82.

12. Song J., An Q., Liu S. A high-resolution time-to-digital converter implemented in field-programmable-gate-arrays. IEEE Transactions on Nuclear Science. 2006. Vol. 53, No. 1. P. 236–241.

13. Hari Prasad K., Chandratre V. B. A high-density, 129-channel time-to-digital converter in FPGA for trigger-less data acquisition systems. Nuclear Instruments and Methods in Physics Research Section A. 2023. Vol. 1057. Article 168727.

14. Moses W. W. Recent advances and future advances in time-of-flight PET. Nuclear Instruments and Methods in Physics Research Section A. 2007. Vol. 580, No. 2. P. 919–924.

15. Hong K. J., Kim E., Yeom J. Y. et al. FPGA-based time-to-digital converter for time-of-flight PET detector. Nuclear Science Symposium Conference Record. 2012. P. 1234–1237.

16. Prasad K. H., Chandratre V. B. Calibration Techniques in ASIC and FPGA Based Time-to-Digital Converters. Lecture Notes in Electrical Engineering. Springer, 2022. P. 195–210.

17. Jovanović G. S., Stojčev M. K. Vernier's delay line time-to-digital converter. Scientific Technical Review. 2009. Vol. 59, No. 3-4. P. 30–37.

18. Cui K., Ren Z., Li X. et al. A high-linearity, ring oscillator-based, Vernier time-to-digital converter utilizing carry chains in FPGAs. IEEE Transactions on Nuclear Science. 2017. Vol. 64, No. 1. P. 697–704.

19. Degnan J. J. Satellite laser ranging: current status and future prospects. IEEE Transactions on Geoscience and Remote Sensing. 1985. Vol. 23, No. 4. P. 398–413.

20. McCarthy J. Timekeeping requirements for time-of-flight measurement. Journal of Physics: Conference Series. 2020. Vol. 1690. Article 012121.

21. Zhang J. et al. Advances in synchronization techniques for quantum networks. Nature Physics. 2023. Vol. 19. P. 1052–1061.

22. White Rabbit Collaboration. Sub-nanosecond timing distribution with White Rabbit. Nuclear Instruments and Methods A. 2016. Vol. 830. P. 235–242.

23. Industrial automation timing requirements. IEC 61784-2:2019.

24. Chulkov V. A. Interpolating time-to-digital converters. Optoelectronics, Instrumentation and Data Processing. 2008. Vol. 44. P. 567–575.

25. Garzetti F., Corna N., Lusardi N., Geraci A. Time-to-digital converters for FPGAs: A systematic approach. IEEE Access. 2021. Vol. 9. P. 85515–85534.

26. Nutt R. Digital time intervalometer. Review of Scientific Instruments. 1968. Vol. 39, No. 9. P. 1342–1345.

27. Rahkonen T., Kostamovaara J. The use of stabilized CMOS delay lines for the digitization of short time intervals. IEEE Journal of Solid-State Circuits. 1993. Vol. 28, No. 8. P. 887–894.

28. Mäntyniemi A., Rahkonen T., Kostamovaara J. A CMOS time-to-digital converter with better than 10 ps single-shot precision. IEEE Journal of Solid-State Circuits. 2002. Vol. 37, No. 6. P. 727–734.

29. Szplet R., Kalisz J., Szymanowski R. Interpolating time counter with 100 ps resolution on a single FPGA device. IEEE Transactions on Instrumentation and Measurement. 2000. Vol. 49, No. 4. P. 879–883.

30. Wang J., Liu S., Zhao L. et al. The 10-ps multitime measurement averaging TDC implemented in an FPGA. IEEE Transactions on Nuclear Science. 2011. Vol. 58, No. 4. P. 2011–2018.

31. Wang Y., Liu C. A 3.9 ps RMS precision time-to-digital converter using ones-counter encoding scheme in a Kintex-7 FPGA. IEEE Transactions on Nuclear Science. 2017. Vol. 64, No. 10. P. 2713–2718.

32. Texas Instruments. High-Resolution Pulse Width Modulator (HRPWM) Reference Guide. SPRUGE9E, 2021.

33. Texas Instruments. TDC7201 Time-to-Digital Converter for Time-of-Flight Applications. Datasheet SNAS647D, 2020.

34. Unsalan C., Tar B. Digital System Design with FPGA: Implementation Using Verilog and VHDL. McGraw-Hill Education, 2017. 480 p.

35. Qin X. et al. A high resolution time-to-digital converter based on a carry-chain and DSP48E1 adders in a 28-nm FPGA. Review of Scientific Instruments. 2020. Vol. 91, No. 2. Article 024708.

36. Bourdeauducq S. A 26 ps RMS time-to-digital converter core for SPARTAN-6 FPGAs. 2013. arXiv:1303.6840.

37. Xilinx. FPGA market analysis report. Annual Report, 2023.

38. Xilinx. 7 Series FPGAs Configurable Logic Block User Guide. UG474, 2016.

39. Chen H. et al. A 64-channel precision time-to-digital converter with average 4.77 ps RMS implemented in a 28 nm FPGA. 2024. arXiv:2412.18642.

40. Intel. Cyclone V Device Handbook. Volume 1: Device Interfaces and Integration, 2020.

41. Bourdeauducq S. A 26 ps RMS time-to-digital converter core for SPARTAN-6 FPGAs. arXiv:1303.6840, 2013.

42. Szplet R., Jachna Z., Kwiatkowski P., Rozyc K. A 2.9 ps equivalent resolution interpolating time counter. Measurement Science and Technology. 2013. Vol. 24, No. 3. Article 035904.

43. Lusardi N. et al. Quantization noise and temperature effects on TDC in FPGAs. IEEE Transactions on Nuclear Science. 2019. Vol. 66, No. 7. P. 1297–1304.

44. Wu J., Shi Z. The 10-ps wave union TDC: Improving FPGA TDC resolution beyond its cell delay. IEEE Nuclear Science Symposium Conference Record. 2008. P. 3440–3446.

45. Wu J. Several key issues on implementing delay line based TDCs using FPGAs. IEEE Transactions on Nuclear Science. 2010. Vol. 57, No. 3. P. 1543–1548.

46. Wang Y., Liu C. A 4.2 ps time-interval RMS resolution time-to-digital converter using a bin decimation method in an UltraScale FPGA. IEEE Transactions on Nuclear Science. 2016. Vol. 63, No. 5. P. 2632–2638.

47. Daigneault M.-A., David J. P. A high-resolution time-to-digital converter on FPGA using dynamic reconfiguration. IEEE Transactions on Instrumentation and Measurement. 2011. Vol. 60, No. 6. P. 2070–2079.

48. Szplet R., Czuba A. Two-stage clock-free time-to-digital converter based on Vernier and tapped delay lines in FPGA device. Electronics. 2021. Vol. 10, No. 18. Article 2190.

49. Zhang M. et al. A high-throughput Vernier time-to-digital converter on FPGAs with improved resolution using a bi-time interpolation scheme. Applied Sciences. 2022. Vol. 12, No. 15. Article 7674.

50. IEEE Standard VHDL Language Reference Manual. IEEE Std 1076-2019.

51. IEEE Standard for Verilog Hardware Description Language. IEEE Std 1364-2005.

52. IEEE Standard for SystemVerilog. IEEE Std 1800-2017.

53. Xilinx. Vivado Design Suite User Guide: Design Flows Overview. UG892, 2023.

ДОДАТОК А
СЛАЙДИ ПРЕЗЕНТАЦІЇ

Рисунок А.1 – Титульний слайд

Рисунок А.2 – Актуальність та мета роботи

Постановка задачі та технічні вимоги

Завдання роботи:

- аналіз методів вимірювання часових інтервалів
- проєктування архітектури генератора
- реалізація мовою VHDL
- верифікація: симуляція та синтез

Параметр	Значення
Тактова частота	100 МГц
Роздільна здатність	10 нс
Розрядність лічильника	32 біти (до 42,95 с)
Глибина буфера FIFO	16 записів
Затримка синхронізації	30 нс

3 / 13

Рисунок А.3 – Постановка задачі та технічні вимоги

Аналіз методів вимірювання часових інтервалів

Метод	Роздільна здатність	Особливість
Лічильник	1–10 нс	Простий, необмежений діапазон
ТАС + АЦП	1–25 пс	Аналоговий, великий мертвий час
Вернье	1–50 пс	Цифровий, складна реалізація
TDL (лінія затримки)	10–100 пс	Цифровий, ресурси ПЛІС (carry chains)

Для реалізації на ПЛІС найбільш придатні цифрові методи — TDL та Вернье.

4 / 13

Рисунок А.4 – Аналіз методів вимірювання часових інтервалів

Рисунок А.5 – Вибір елементної бази

Рисунок А.6 – Архітектура генератора часових міток

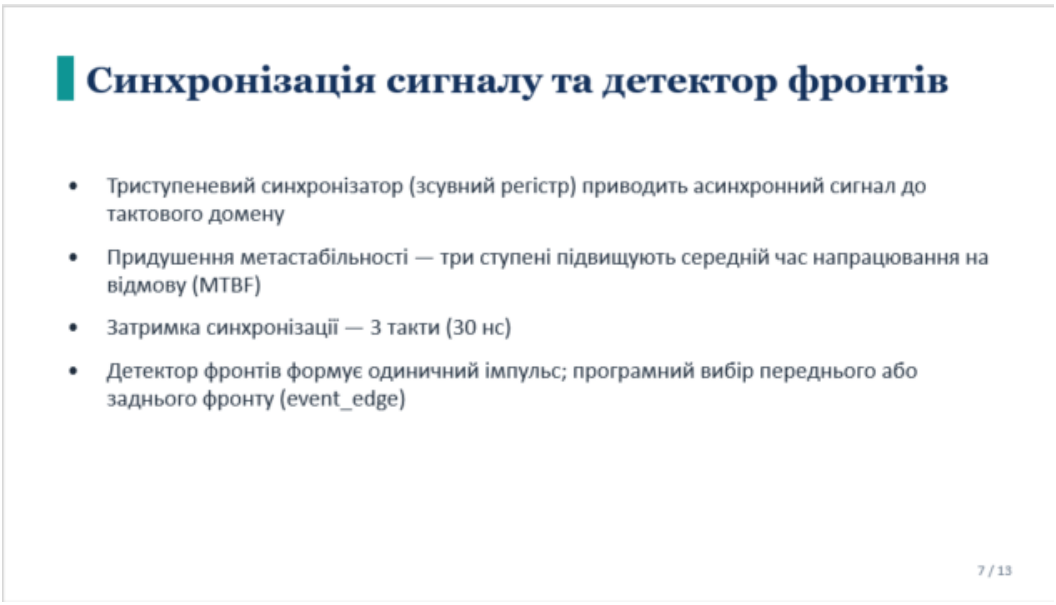

Рисунок А.7 – Синхронізація сигналу та детектор фронтів

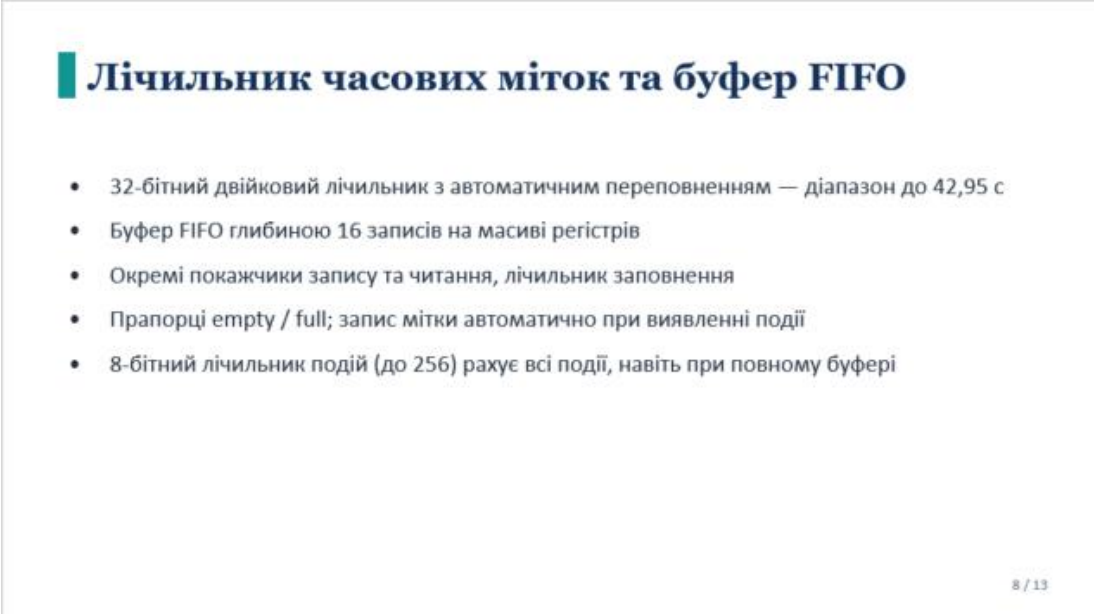

Рисунок А.8 – Лічильник часових міток та буфер FIFO

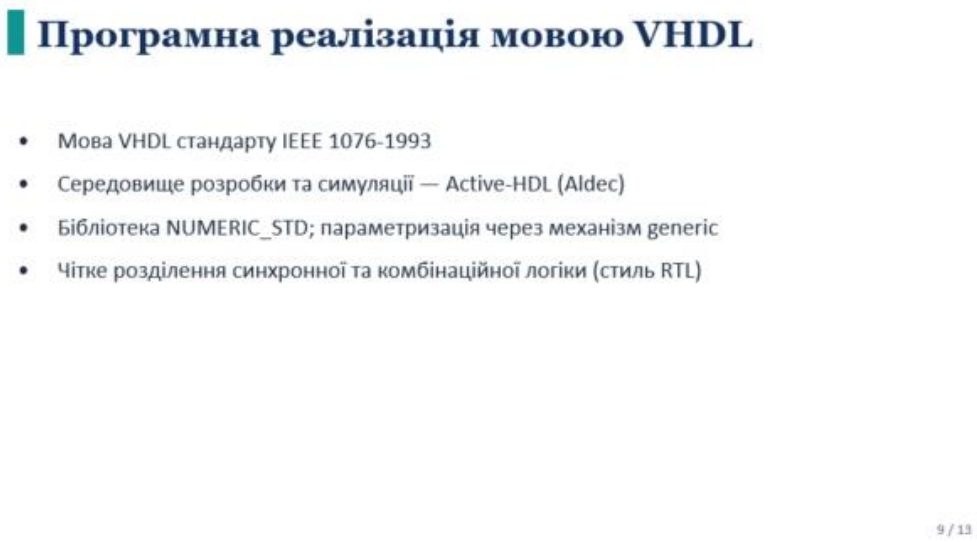

Рисунок А.9 – Програмне реалізація мовою VHDL

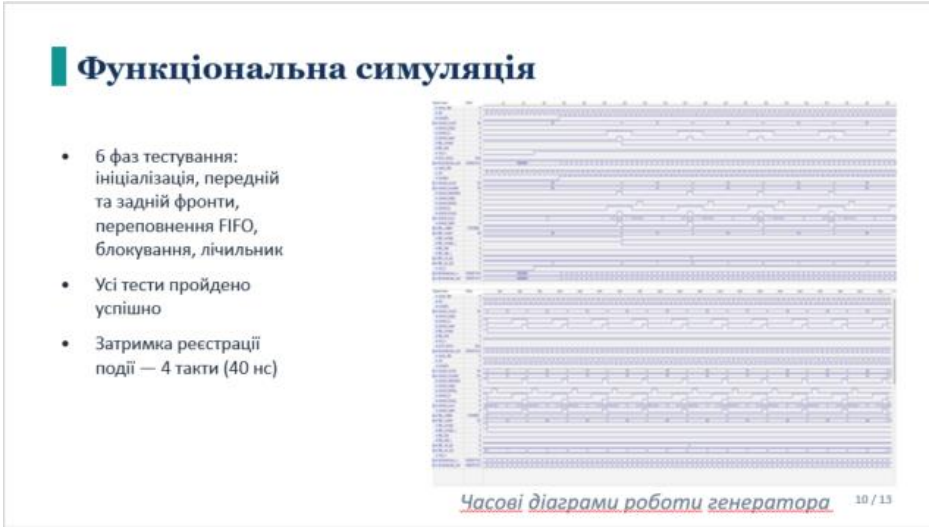

Рисунок А.10 – Функціональна симуляція

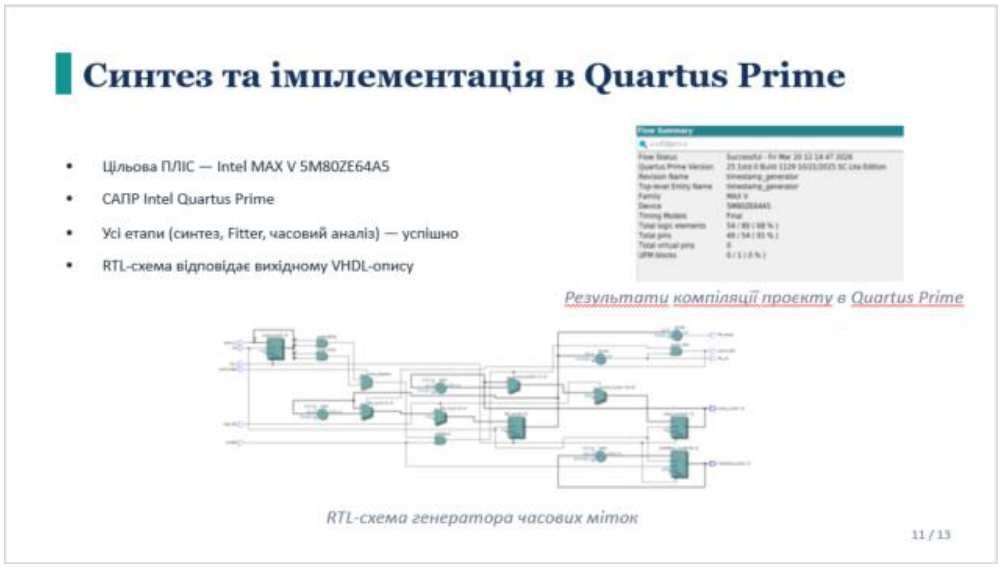

Рисунок А.11 – Синтез та імплементація в Quartus Prime

Рисунок А.12 – Результати розміщення на кристалі

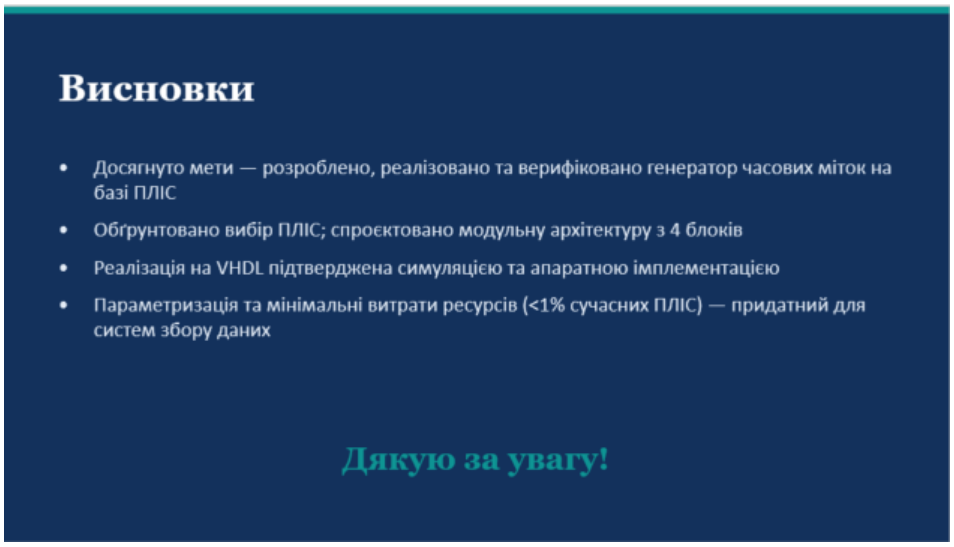

Рисунок А.13 – Висновки