Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет комп’ютерної інженерії та управління
(повна назва)

Кафедра електронних обчислювальних машин
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

Рівень вищої освіти другий (магістерський)

Система круїз-контролю на основі перепрограмованої FPGA

(тема)

Виконав:

студент II курсу, групи КСМм-21-1

Куліш Д. В.
(прізвище, ініціали)

Спеціальність

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

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

Освітня програма

Комп’ютерні системи та мережі
(повна назва освітньої програми)

Керівник: проф. Горбачов В. О.
(посада, прізвище, ініціали)

Допускається до захисту

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

2022 р.

Харківський національний університет радіоелектроніки

Факультет комп'ютерної інженерії та управління

Кафедра електронних обчислювальних машин

Рівень вищої освіти другий (магістерський)

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

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

Освітня програма Комп'ютерні системи та мережі
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
"____" _____ 20__ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студенту Кулішу Дмитру Віталійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Система круїз-контролю на основі перепрограмованої FPGA

затверджена наказом по університету від " 07 " листопада 2022 р. № 1453 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 13 грудня 2022 р.

3. Вхідні дані до роботи 1) програмна реалізація на основі реконфігурації
2) система на кристалі сімейства Virtex 7

4. Перелік питань, що потрібно опрацювати у роботі
1) Огляд принципів побудови сучасних систем круїз-контролю
2) Аналіз технологій сучасних реконфігурованих FPGA та сучасних систем круїз-контролю на основі реконфігурованої SoC
3) Огляд алгоритмів та технологій розробки для реалізації поставленої задачі
4) Розробка архітектури системи круїз-контролю та опис її програмної реалізації
5) Тестування розробленої системи круїз-контролю на симульованій FPGA
6) висновки.

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

Слайд-презентація – 12 слайдів

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

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

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Огляд сучасних систем круїз-контролю	08.11.22-11.11.22	
2	Аналіз технологій сучасних реконфігурованих	12.11.22-14.11.22	
	FPGA		
3	Вибір алгоритмів та технологій розробки	15.11.22-18.11.22	
4	Розробка архітектури системи	19.11.22-23.11.22	
5	Програмна реалізація системи	24.11.22-04.12.22	
6	Тестування реалізованої системи	05.11.22-06.12.22	
7	Оформлення матеріалів кваліфікаційної роботи	07.12.22-08.12.22	
8	Подання кваліфікаційної роботи керівникові та	09.12.22-10.12.22	
	її попередній захист		
9	Подання кваліфікаційної роботи на рецензу-	11.12.22-12.12.22	
	вання		

Дата видачі завдання 07 листопада 2022 р.

Студент _____
(підпис)

Керівник роботи _____ проф. Горбачов В. О.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 81 с., 45 рис., 1 табл., 2 дод., 24 джерел.

VHDL, FPGA, СИСТЕМА НА ЧІПІ, VIRTEX7, РЕКОНФІГУРАЦІЯ, СИСТЕМА КРУЇЗ-КОНТРОЛЮ.

Метою кваліфікаційної роботи є розробка системи круїз-контролю на основі перепрограмованої FPGA.

У ході виконання кваліфікаційної роботи було проведено аналіз сучасних систем круїз-контролю, розглянуто технологію реконфігурації та сімейство продуктивних систем на чіпі VIRTEX7. Було використано мову VHDL для реалізації системи, призначеної для підтримки встановленої швидкості автомобіля. Було проведено тестування системи в різних ситуаціях та за різних зовнішніх умов, які впливають на швидкість руху транспорту.

ABSTRACT

Master’s thesis: 81 pages, 45 figures, 1 table, 2 appendices, 24 sources.

VHDL, FPGA, SoC, VIRTEX7, RECONFIGURATION, CRUISE-CONTROL SYSTEM.

The purpose of the qualification work is to develop a cruise control system based on a reprogrammed FPGA.

In the course of the qualification work, an analysis of modern cruise control systems was carried out, reconfiguration technology and a family of productive systems on the VIRTEX7 chip were considered. VHDL language was used to implement a system designed to maintain the set speed of the vehicle. The system was tested in various situations and under various external conditions that affect the speed of traffic.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ІКК – інтелектуальний круїз-контроль

ПКК – прогнозований круїз-контроль

ПЛІС – програмована логічна інтегральна схема

ЧДР – часткова динамічна реконфігурація

ACC – адаптивний круїз-контроль (англ., adaptive cruise-control)

ASIC – інтегральна схема для конкретного застосування (англ., application-specific integrated circuit)

BRAM – блок оперативної пам'яті (англ., block random access memory)

CLB – конфігуруємий блок логіки (англ., configurable logic block)

DSP – цифровий сигнальний процесор (англ., digital signal processor)

FPGA – вентильна матриця, програмована у полем (англ., field-programmable gate array)

GPS – глобальна система позиціонування (англ., global positioning system)

LUT – таблиця підстановок (англ., look-up table)

SoC – система на чіпі (англ., system on a chip)

VHDL – дуже високошвидкісні інтегральні схеми (англ., very high speed integrated circuits)

ВСТУП

У наш час людині доводиться багато пересуватись: з дому до роботи, купити продукти в найближчому гіпермаркеті, з'їздити до батьків на вихідні. Іноді доводиться долати великі відстані, які важко пройти пішки і доводиться користуватися приватним автомобілем або громадським транспортом.

В той же час більшість побутових речей, якими користується людина включають в себе міні-комп'ютер, який полегшує користування, додає нові функції, тому не дивно, що сучасні транспортні засоби також стають все сильніше наповнені електронікою. Частка електроніки у вартості автомобіля значно зросла з 18% у 2000 р. до 40% у 2020 р. [2]

Сучасні автомобілі оснащуються великою кількістю електронних систем під управлінням мікрочіпів [1] (механізм спрацьовування подушки безпеки, система автоматичного гальмування, круїз-контроль, системи навігації та мультимедіа, помічники з паркування та ін.)

Одна з таких систем – це система круїз-контролю. Дана система допомагає підтримувати постійну швидкість автомобіля, вона реагує на зміну швидкості руху транспорту, що не залежить від самого транспорту, наприклад при підйомі вона збільшує подачу топлива або зменшує на спуску. Сучасні системи круїз-контролю навіть слідкують за швидкістю попереду ідучого автомобіля і контролює відстань між цим автомобілем та вашим.

Актуальність цієї атестаційної рооти заключається в стрімкому зростанні частки електроніки в трансортних засобах, круїз-контроль полегшує та робить зручнішим керування автомобілем.

Таким чином система круїз-контролю як одна з систем в сучасних автомобілях є актуальною в даний часі потребує практичної реалізації. Реалізація даної системи буде заключатися в програмуванні FPGA, що буде аналізувати кут натискання педалі газу і буде виконувати дії для підтримання постійної швидкості автомобіля.

1 АКТУАЛЬНІСТЬ ПРОБЛЕМИ ТА ПОСТАНОВКА ЗАВДАНЬ ДОСЛІДЖЕННЯ

1.1 Актуальність проблеми

В минулому сторіччі активно розроблялися електронні системи для транспортних засобів, які робили керування комфортнішим та безпечнішим. Наприклад такі системі як: система контролю стійкості автомобіля, антиблокувальна система гальм, система контролю тяги, круїз-контроль, адаптивний круїз-контроль та інщі.

Однією з найбільш поширених і помітних систем керування, доступних на сучасних автомобілях є круїз-контроль, який автоматично регулює поздовжню швидкість автомобіля за допомогою регулювання газу. Як правило, систему круїз-контролю автомобіля активує водій, який хоче підтримувати постійну швидкість під час тривалої їзди [3].

Частка електроніки в ціні автомобіля зростає з кожним роком і вже складає майже половину. Вже очевидна тенденція, що зростання кількості автомобільної електроніки продовжиться в найближчому майбутньому майбутнього . У наступне десятиліття очікуються значні успіхи у використанні силової електроніки, вдосконаленого керування системи та альтернативні концепції трансмісії.

Разом зі збільшенням частки електроніки, збільшується і кількість проданих автомобілей, експерти прогнозують збільшення кількості проданих автомобілей до 130 мільйонів за рік а кількість усіх автомобілей вже пересягнула планку в 1.2 мільярди. Враховуючи ці данні і те що система круїз-контролю вже перейшла з розряду передових технологій в розряд базових, можна вважати тему атестаційної роботи актуальною.

1.2 Принципи побудови сучасних систем круїз-контролю

Системи контролю швидкості розроблялися ще у 1900-х роках, але сучасний вигляд вони отримали лише в 40-х – 50-х роках минулого сторіччя. Одна з перших машин з такою системою була Chrysler Imperial, випущена в 1958 році, згодом американський виробник Cadillac дав цій системі назву круїз-контроль.

Круїз-контроль отримує сигнал швидкості від кабелю спідометра або інших датчиків. Транспортний засіб підтримуватиме потрібну швидкість, тягнучи трос дросельної заслінки, при цьому буде подаватися більше або менше горючої суміші.

Систему круїз-контроля можна розділити на декілька складових:

- датчики для спостереження за швидкістю автомобіля;
- датчики для спостереження за кутом нахилу педалі газу;
- деханізми керування подачею горючої суміші до двигуна автомобіля;
- донтроллер круїз-контролю, який буде проводити розрахунки і подавати сигнали для збільшення або зменшення кількості горючої суміші.

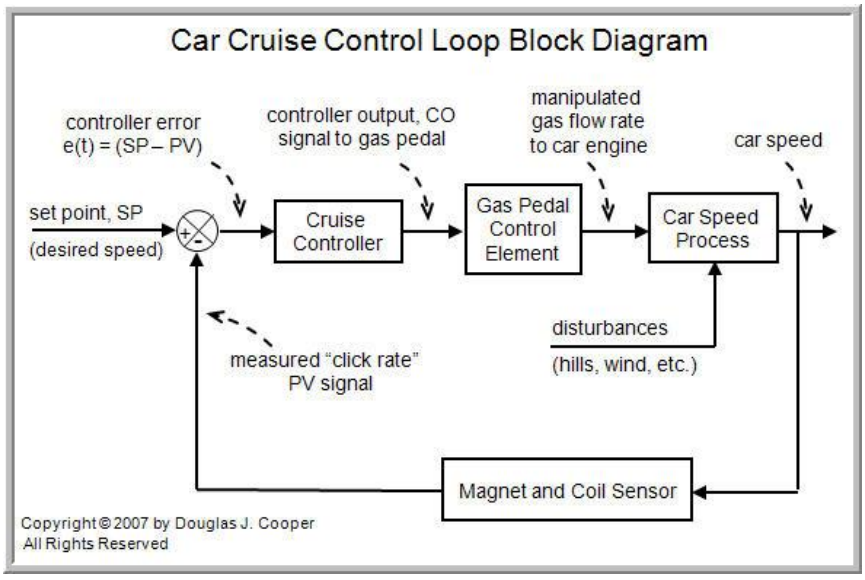

Рисунок 1.1 – Блокова діаграма схеми роботи системи круїз-контролю

Сучасні системи круїз-контролю мають назви прогностичний круїз-контроль та адаптивний круїз контролью. Прогнозний круїз-контроль поєднує в собі круїз-контроль із GPS і топографічними мапами, керує швидкістю автомобіля в різних типах місцевості. Технологія ПКК забезпечує максимальні переваги при їзді по пагорба. Адаптивний круїз-контроль регулює швидкість автомобіля залежно від швидкості транспорту, що рухається попереду, щоб підтримувати встановлену відстань[4].

1.3 Системи круїз-контролю на основі FPGA

Система круїз-контролю є вбудованою системою, з цього можна вивести, що ресурси даної системи є обмеженими. Потрібно враховувати енергоспоживання, габарити та ціну електронного пристрою, що буде реалізовувати роботу даної системи. Це спонукає нас використовувати технології FPGA, які пропонують гнучкі рішення для реалізації різних стадій, що складають всю систему. Крім того, FPGA дозволяють нам ефективно впроваджувати повну систему, де реалізуються завдання інтенсивної обробки сигналів як апаратні блоки та послідовні обчислення [5].

Система круїз-контролю на основі FPGA складається з самої FPGA, датчиків, механізму керування дросельної заслінки та каналів, через які передаються сигнали. Як видно на рисунку 1.2 сигнали приходять від датчиків, проходячи через посилювачі сигналу, інтегральна схема опрацьовує ці сигнали. Якщо система виявила зміну швидкості, без зміни положення педалі газу, наприклад автомобіль рухається в гору, то вона подає відповідний сигнал, що вказує як потрібно змінити положення дросельної заслінки.

Система на основі FPGA має можливість до масштабування, легкої інтеграції та є енергоефективною.

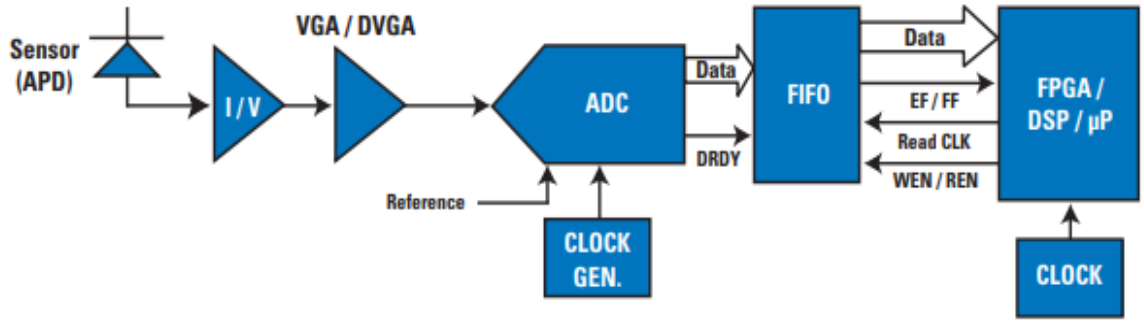

Рисунок 1.2 – Блокова діаграма отримання даних FPGA від датчиків

1.4 Постановка завдань дослідження

Метою кваліфікаційної роботи є розробка системи круїз-контролю на основі SoC сімейства Virtex 7. Основною задачею системи є підтримка заданої швидкості автомобіля, на яку впливають зовнішні умови, наприклад рух на пагорб, або з нього.

Для виконання поставленого завдання є ряд задач:

- розібрати особливості реалізації сучасних систем круїз-контролю;
- проаналізувати сучасні технології реконфігурації;
- розібрати алгоритми та технології для реалізації системи круїз-контролю;
- розробити архітектуру та програмну реалізації системи круїз-контролю;
- провести тестування розробленої системи;
- зробити аналіз результатів тестування розробленої системи.

2. ТЕХНОЛОГІЇ СУЧАСНИХ РЕКОНФІГУРОВАНИХ FPGA

2.1 Програмована вентильна матриця

FPGA або програмована вентильна матриця – це інтегральна схема, призначена для конфігурації користувачем або розробником вже після виготовлення. Конфігурація FPGA зазвичай вказується за допомогою мови опису апаратного забезпечення. Раніше для вказівки конфігурації використовувалися схеми для конкретної програми, але це трапляється все рідше через появу інструментів автоматизації проектування.

FPGA складається з логічних блоків, що конфігуруються (CLB). CLB знаходяться в комутаційній матриці (рисунок 1.3), яка визначає з'єднання входів і виходів для них. На кожному перетині провідників знаходиться шість ключів, що перемикають, керованих своїми комірками конфігураційної пам'яті. Відкриваючи одні та закриваючи інші, можна забезпечити різну комутацію сигналів між логічними блоками.

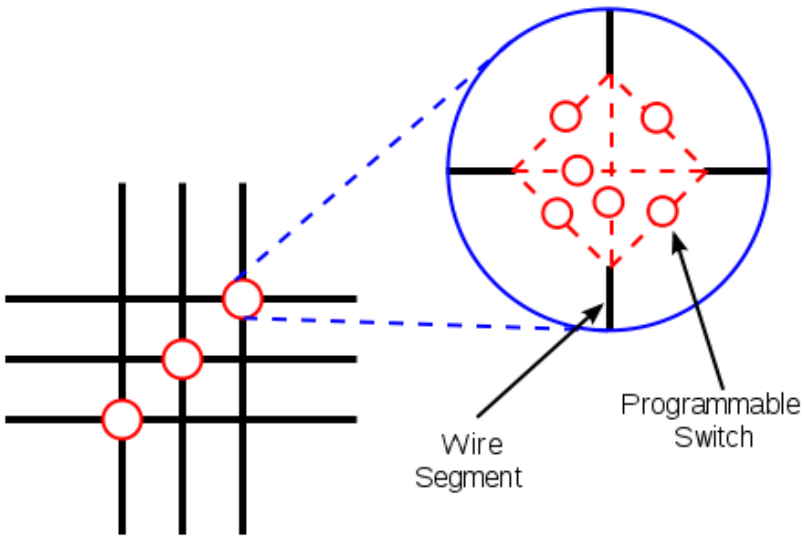

Рисунок 1.3 – Схема комутаційної матриці

CLB складаються з блоку, що задає булеву функцію від кількох аргументів, він має назву таблиця відповідності (LUT) та тригера. У сучасних FPGA LUT має шість входів. Вихід LUT подається на вихід CLB синхронно або асинхронно.

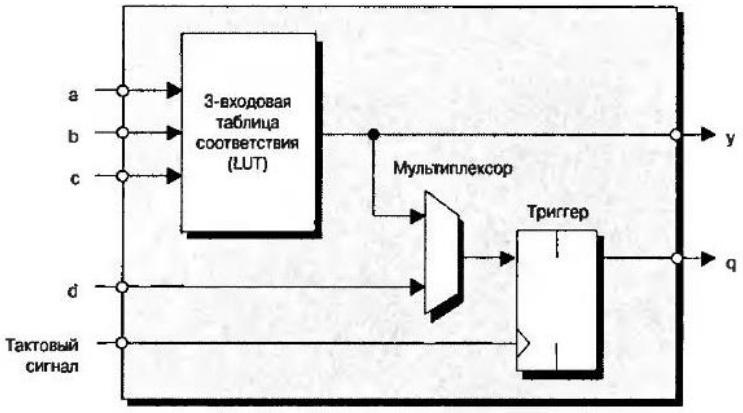

Рисунок 1.4 – CLB з 3 входами для LUT

LUT реалізує конкретну функцію. Наприклад на рисунку 1.5 показана програмна реалізація функції з 3 аргументами. Для реалізації функції будується таблиця відповідності, вихідні значення якої записуються в комірки пам’яті LUT.

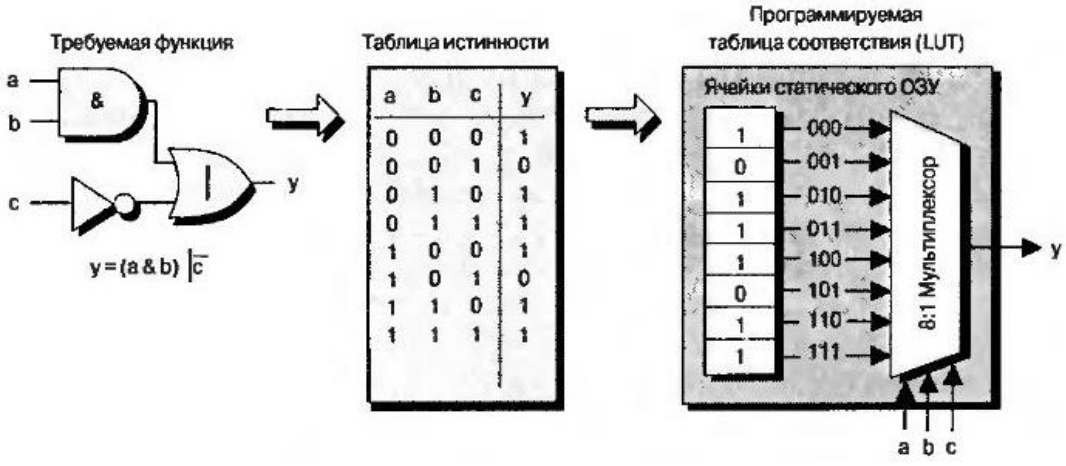

Рисунок 1.5 – Реалізація LUT

Значення кожної з комірки подається на вхід вихідного мультиплексора LUT, а вхідні аргументи булевої функції використовуються для вибору того чи іншого значення функції. CLB – найважливіший апаратний ресурс FPGA. Кількість CLB у сучасних кристалах FPGA може бути різною і залежить від типу та ємності кристала. Xilinx має кристали з кількістю CLB в межах приблизно від чотирьох тисяч до трьох мільйонів. Перший чіп даної компанії був випущений в 1985 році і мав всього 64 CLB.

Окрім блоків CLB FPGA може включати блоки внутрішньої пам’яті (BRAM), яка може досягати від 20 кбайт до 20 мбайт, блоки множення, блоки цифрової обробки сигналу (DSP). Зв'язуючи блоки CLB, DSP та BRAM, можна отримувати дуже ефективні схеми обробки даних.

FPGA використовуються у багатьох споживчих продуктах, таких як автомобілі, медичне обладнання, пристрої для обробки медіа інформації. Наприклад, у системах навігації, круїз-контролю, звуковідтворення автомобілів Mercedes Benz S-класу використовується більше десяти FPGA та PLD фірми Xilinx [6]. FPGA дозволяють швидше виводити вироби на ринок і спрощують налагодження та додавання нових можливостей на пізніх етапах життєвого циклу продукту.

Сучасні рішення на базі FPGA мають такі переваги, як гнучкість, масштабованість, висока продуктивність, низька затримка обчислень та енергетична ефективність. FPGA може бути налаштована для вирішення конкретної задачі.

2.2 Реконфігурація

Для конфігурації ПЛІС створюється конфігураційний бітовий потік, який на прикладі сімейства Xilinx Virtex II складається із заголовку й конфігураційної інформації. Остання має ієрархічну структуру – на першому рівні складається з трьох блоків, далі кожен блок поділяється на стовпці, що конфігурують множину базових компонентів ПЛІС. Наприклад, стовпці типу

IOI – конфігурують блоки регістрів, мультиплексорів та буфери з трьома станами лівого та правого боків пристроїв, CLB – конфігурують логічні блоки та блоки вводу/виводу верхнього та нижнього боків пристрою, BRAM та BRAM_INT – конфігурують частини, визначені користувачем в BlockRAM. Структура бітового потоку зображена на рис. 1.6. Можна розглядати повний бітовий потік, що створюється для конфігурації всієї мікросхеми, та частковий бітовий потік, що відповідає тільки тій області мікросхеми, що змінюється під час реконфігурації – часткової реконфігурації[14].

Рисунок 1.6 – Структура бітового потоку

Одна зі сторін задачі реконфігурації зводиться до виділення площини ПЛІС для розміщення реконфігурованих модулів, та розведення зв’язків між модулями системи. Для реалізації цього розглядають декілька способів. Один з них має назву реконфігурація стовпцями. Кожен стовпець реалізує певну кількість вертикальних фреймів, висота яких дорівнює висоті пристрою, а ширина одному логічному блоку CLB. Для Virtex II і Virtex II Pro Xilinx мінімальна область, що може бути реконфігурована, відповідає повному стовпцю ПЛІС. Цільова область реконфігурації складається з певної кількості стовпців і розмішується на всю висоту пристрою. Частковий бітовий потік включає тільки ті конфігураційні стовпці, що мають бути змінені. Виходячи з цього, час затрачений на часткову реконфігурацію, буде менший часу завантаження більшого за розміром повного бітового потоку. При цьому на розмір часткового бітового потоку звичайно впливає тип пристрою, а саме його розмір. Наприклад, розмір конфігураційного фрейму мікросхеми Virtex

II XC2V8000 складає 286 біт, а Virtex II XC2V40 – 26 біт. Виходячи з цього, час реконфігурації пропорційний довжині бітового потоку та кратний висоті стовпця мікросхеми, визначає один з недоліків розглянутої технології. Серед інших недоліків: непродуктивні витрати ресурсів ПЛІС та неефективне розміщення внутрішніх зв'язків між модулями зі значною розбіжністю розмірів реконфігурованої області і завантажуваного модуля.

Під час створення мікросхем сімейств, розпочинаючи з Virtex 4, з метою подолання недоліків реконфігурації стовпцями, з'явилася нова технологія реконфігурації, що отримала назву блочної або модульної реконфігурації. Організація бітового потоку також базується на фреймах, але їх структура тепер не залежить від висоти пристрою, а має стандартний розмір – 16 логічних блоків CLB в висоту і 1 CLB в ширину. Під час розводки ПЛІС та створення бітової послідовності відповідним програмним забезпеченням можна вибрати, який фізичний розмір на ПЛІС займатиме реконфігурований модуль. Відповідна реконфігурована область, таким чином, буде будь-якого прямокутного розміру, але кратного одному конфігураційному CLB-фрейму. В подальших сімействах простежується тенденція до зростання кількості логічних блоків у фреймі — 20 логічних блоків CLB у висоту і 1 CLB у ширину в мікросхемі Virtex 5, 40 CLB блоків у висоту – Virtex 6. Завдяки цьому також зменшується кількість фреймів для реконфігурації, а, отже, зменшується розмір бітового потоку і час здійснення реконфігурації

Найгнучкіший спосіб розміщення реконфігурованих модулів на площині ПЛІС передбачає, що кожен апаратний модуль реалізується прямокутними групами вузлів і розміри його не обмежені заздалегідь зумовленими розмірами частково реконфігурованої області. Під час генерування модуля, площу та співвідношення сторін можна оптимізувати відповідно до внутрішніх зв'язків модуля. Саме таку методологію пропонує так звана підтримка часткової динамічної реконфігурації в сучасних сімействах мікросхем ПЛІС Virtex 6, Virtex 7 – реконфігурація розділами.

Вона підтримується й на рівні інструментальних засобів, що вирішує проблему забезпечення однорідності вузлів та встановлення зв’язків між ними. Таке розміщення забезпечує найбільш ефективну реалізацію з точки зору використання ресурсів, розведення зв’язків, розміру бітової послідовності, часу реконфігурації.

Мікросхеми Virtex 4, Virtex 5 підтримують як модульний підхід, так і розподіл розділами, засобами спеціального програмного забезпечення. Сучасні пристрої Virtex 6, Virtex 7 підтримують найсучасніші технології ЧДР, що на підставі своєї затребуваності активно розвиваються виробниками ПЛІС.

Для зберігання часткових послідовностей використовується внутрішня пам’ять ПЛІС ціною обмеження на розмір і кількість послідовностей, котрі можуть бути збережені. Але така реалізація забезпечує більшу швидкодію вибірки даних, і простіший інтерфейс, ніж у зовнішньої пам’яті. Часткові конфігурації можна зберігати у зовнішній енергонезалежній пам’яті. Якщо враховувати співвідношення внутрішньої площі кристала до ціни, то кращим виходом буде зберігати неактивні бітові послідовності у зовнішній енергонезалежній пам’яті.

Розрізняють реконфігурацію часу зупинки (РЧЗ) та реконфігурацію часу виконання (РЧВ). Відповідно до назви, реконфігурація виконується або за зупинки системи або під час її роботи. Особливостями першого способу є зупинка системи та необхідність наявності зовнішніх засобів здійснення реконфігурації. Цей спосіб є значно дешевшим та простішим за реалізації. Другий спосіб реалізується за збереження працездатності системи під час реконфігурації та реалізації алгоритмів реконфігурації як частини системи.

Статична реконфігурація характеризується виконанням під дією зовнішніх алгоритмів управління та ініціалізацією запитом ззовні. Така реконфігурація за способом управління належить до пасивного типу, є повністю суб’єктивним процесом, відбувається на вимогу користувача системи та зазвичай застосовується для рішення користувацької задачі.

Динамічна реконфігурація характеризується динамічним або автоматичним режимом здійснення, незалежно від будь-яких зовнішніх впливів. Ініціалізація, так само як і сам процес реконфігурації, відбувається під час роботи системи під дією алгоритмів управління, що є частиною системи. Лише внутрішні фактори, що породжуються станом системи або логікою управління визначають процес динамічної реконфігурації.

За способом виконання реконфігурації розрізняють повну та часткову реконфігурацію. Повна реконфігурація (ПР) – це реконфігурація одного чи кількох фізичних модулів системи у повному об’ємі. Часткова реконфігурація (ЧР) – це процес реконфігурації частини фізичного модуля системи. Якщо перший спосіб відбувається за повної зупинки роботи фізичного модуля на час виконання реконфігурації, то у другому випадку модуль зможе виконувати певні функції,

Розвиток частково реконфігурованих систем – системи, що самореконфігурується (Self Reconfiguration, SR). Вони можуть змінювати власне апаратне забезпечення в динамічному режимі під дією внутрішніх алгоритмів управління, що надає їм надвисокої ефективності, гнучкості та автономності. Такі системи часто використовуються в співпроцесорах, збудованих на технології реконфігурації. Іншими прикладами їх використання є комп’ютерні мережі, обробка зображень, шифрувальні алгоритми.

В реконфігурованих обчислювальних системах найбільш ефективно використання динамічної реконфігурації. Це дозволить перебудовувати архітектуру системи автономно, в динамічному режимі. Це забезпечує, з одного боку, підвищення продуктивності реконфігурованих обчислювальних систем взагалі, а з іншого боку, приведення реконфігурованих обчислювальних систем до класу обчислювальних систем загального використання з досягненням високої реальної продуктивності обчислень для вирішення широкого класу задач.

Для реалізації найбільш продуктивних проектів доцільно використовувати сучасні пристрої, які підтримують ефективні технології проектування, з підтримкою часткової динамічної реконфігурації, та забезпечують необхідну інструментальну підтримку для спрощення проектування. Окрім того, політика компаній виробників щодо оновлення своїх продуктів, дотримується принципів ієрархічної спадковості, що призводить, в першу чергу, до вилучення застарілих рішень в нових апаратних та програмних продуктах. Серед актуальних на сьогодні мікросхем доцільно використання пристроїв сімейств Virtex 5 – Virtex 7 компанії Xilinx, сучасних пристроїв компаній Altera, розпочинаючи з 2010 року виробництва, зокрема Stratix V, Cyclone V.

ПЛІС серії Virtex 7 компанії Xilinx налаштовуються шляхом завантаження конфігурації для конкретної програми у внутрішню пам’ять. ПЛІС серії 7 можуть завантажуватися з зовнішнього енергонезалежного пристрою пам’яті або їх можна налаштувати за допомогою зовнішнього джерела, наприклад, мікропроцесору, процесору DSP, мікроконтролеру, ПК або тестер плати. Як процесори та периферійні пристрої процесорів, так і Xilinx FPGA можна перепрограмувати в системі необмежену кількість разів [7].

У сімействі ПЛІС серії 7 конфігураційна пам’ять використовується головним чином для реалізації логіки користувача, підключення та введення/виведення, але він також використовується для інших цілей. Наприклад, це використовується для визначення різноманітних статичних умов у функціональних блоках, таких як годинник плитки керування. Іноді програма вимагає зміни цих умов у функціональних блоках поки блок працює. Це можна досягти частковою реконфігурацією за допомогою портів JTAG, ICAPE2 або SelectMAP. Однак динамічна реконфігурація порту є складовою частиною багатьох функціональних блоків, що значно спрощує цей процес. Така конфігурація порти існують у CMT, MMCM/PLL, XADC, послідовних трансиверах і блоці PCIe.

Часткова динамічна реконфігурація вирішує проблему ініціалізації FPGA великого обсягу, які використовують інтерфейс PCI Express. Специфікація PCIe вимагає відповіді пристрою на запит від контролера протягом певного часу, який виявляється меншим, ніж час повного завантаження конфігурації деякі FPGA [8].

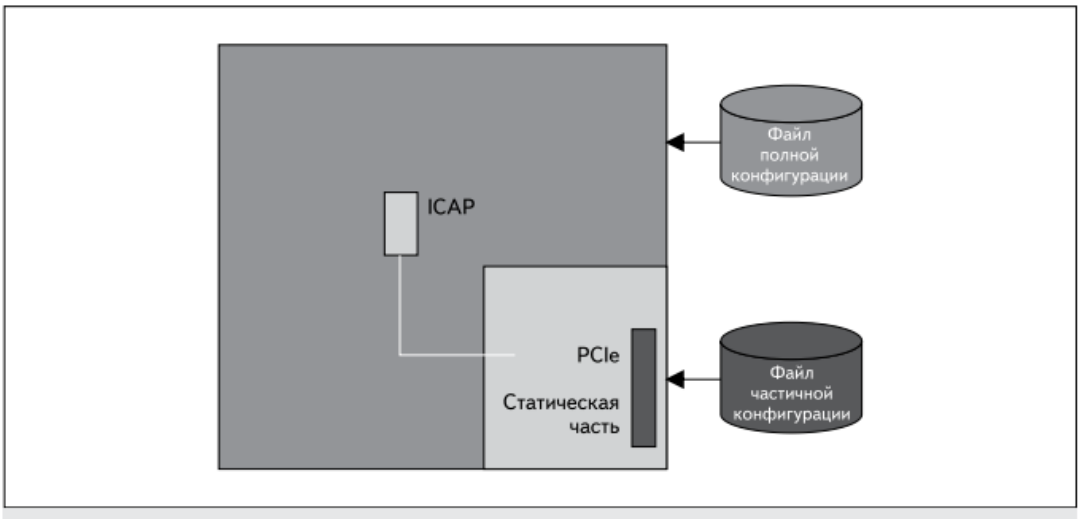

Рисунок 1.6 – Приклад використання часткової реконфігурації

Для цього можна завантажити частину проекту, що містить контролер PCI Express, що дасть можливість коректно ініціалізувати пристрій, а потім довантажити необхідні компоненти в режимі часткової реконфігурації без втрати працездатності PCI Express. На рисунку 1.6 представлений такий порядок завантаження.

Іншим прикладом ефективного використання часткової реконфігурації є динамічна зміна функцій, що виконуються окремими частинами ПЛІС. На рисунку 1.7 показаний приклад схеми багатопортового комутатора, кожен із портів якого може працювати в одному з трьох режимів. При цьому використання ПЛІС виявляється нераціональним, оскільки у конкретний момент часу кожен порт працює тільки в одному з режимів і приблизно 2/3 мікросхеми не задіяно.

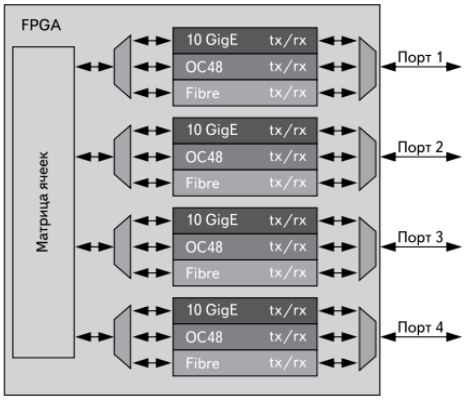

Рисунок 1.7 – Багатопортовий комутатор, що реалізує перемикання режимів роботи портів на основі однієї конфігурації FPGA

Альтернативний варіант реалізації такого пристрою показано на рисунку 1.8, де обсяг ПЛІС істотно менше, що дозволяє в кожний момент часу підтримувати лише один режим на кожному з портів.

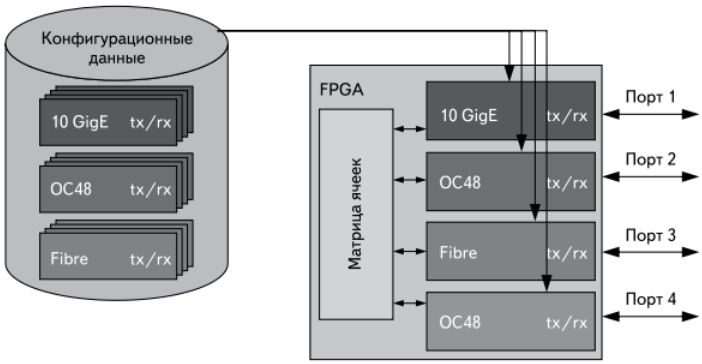

Рисунок 1.8 – Багатопортовий комутатор з режимами роботи окремих портів, що динамічно перезавантажуються.

При цьому часткові конфігурації зберігаються на зовнішньому носії та перезавантажуються у міру потреби без переривання роботи інших портів.

3. АНАЛІЗ СУЧАСНИХ СИСТЕМ КРУЇЗ-КОНТРОЛЮ НА ОСНОВІ РЕКОНФІГУРОВАНОЇ SOC

3.1 Системи на чіпі

Майже кожен автомобіль, який сьогодні сходить з конвеєра, використовує вбудованої технології в тій чи іншій формі. Більшість з вбудовані системи в автомобілях складаються з одного чіпа. Їхні компактні габарити дозволяють легко поміщати їх під тісний капот автомобіля. Ці системи можна використовувати для реалізації функцій, починаючи від регулювання підвіски відповідно до дорожніх умов до систем допомоги водію, наприклад круїз-контроль. Вбудовані системи також можуть зробити керування автомобілем без самого керування реальністю. Великі виробники автомобілів вже працюють над цією концепцією. Однією з таких технологій є круїз-контроль в автомобілях компанії Ford[9].

Сучасна система на чіпі (SoC) складається з кількох різних мікропроцесорних підсистем разом із пам'ятю та інтерфейсами введення/виведення. Процесором може бути спеціальний або стандартний мікропроцесор, або це може бути FPGA спеціалізована для виконання конкретної функції, такої як система круїз-контролю, автоматичного гальмування, допомоги з паркуванням, навігації чи мультимедія. Процесори з’єднані між собою за допомогою різних механізмів, включаючи спільну пам’ять і передачу повідомлень через спеціалізовані канали.

Системи допомоги водію складаються з центрального блоку та датчиків. Центральний блок включає в себе SoC та FPGA, що забезпечує високу продуктивність та ефективне енергоспоживання. Системи допомоги водієві – це системи та функції, які надають необхідну інформацію, автоматизують складні чи повторювані завдання з метою забезпечення загального підвищення безпеки для всіх учасників руху.

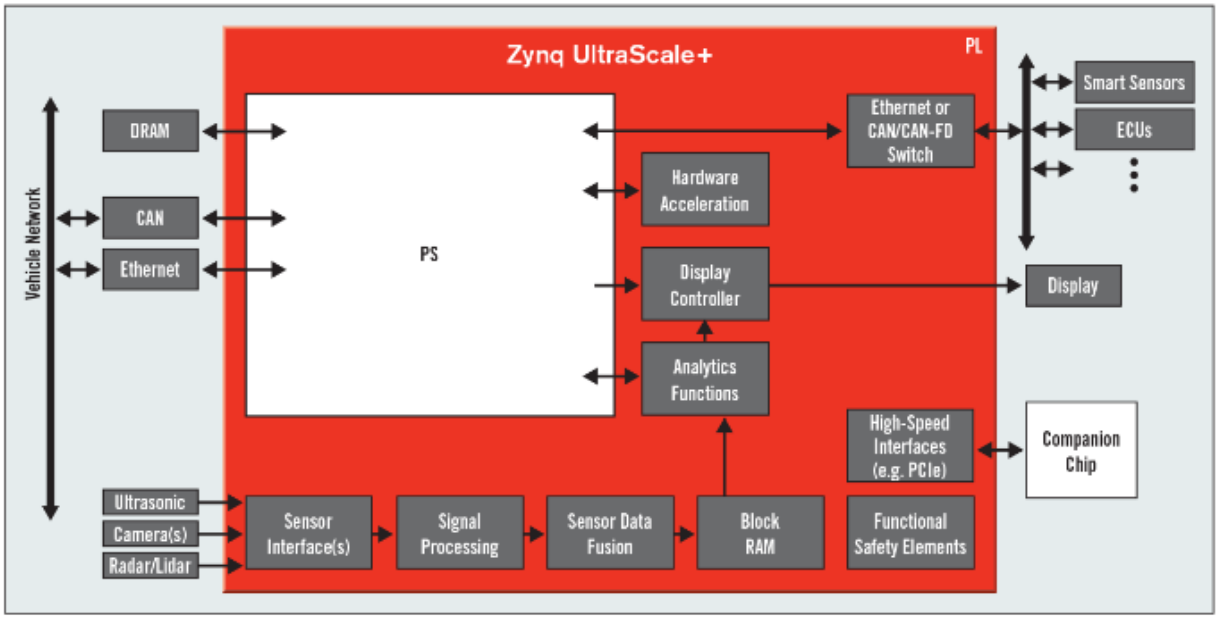

Рисунок 3.1 – SoC Zynq Ultrascale+ компанії Xilinx

Наприклад на рисунку 3.1 зображено SoC Zynq Ultrascale+ компанії Xilinx, до неї підключаються різні периферійні датчики та FPGA для забезпечення відповідних функцій, наприклад: круїз-контроль, автоматичне паркування, розпізнавання перешкод та дорожніх знаків, контроль сліпих зон тощо.

SoC використовуються в багатьох інших сферах, наприклад компанія Xilinx виготовляє різні SoC для компаній IBM, Kodak, Sony, Motorola, Toshiba, LG, вони працюють в космосі, їх використовує компанія NASA на своїх марсоходах.

Підхід із застосуванням FPGA дозволяє виробникам автомобільних систем отримати широкий спектр функціональних властивостей, які можуть бути доповнені та покращені з часом. Платформи на базі FPGA і SoC мають властивості масштабованості і можливістю модернізації, в тому числі віддалено. Програмовані "чіпи-хамелеони" посіли перше місце в щотижневому журналі "10 технологій, що змінять наше життя".

Сучасні системи круїз-контролю мають назви: адаптивний круїз-контроль, інтелектуальний круїз-контроль.

3.2 Адаптивний круїз-контроль

Адаптивний круїз-контроль (ACC) – передова система допомоги водієві, круїз-контроль для дорожніх транспортних засобів, який автоматично регулює швидкість автомобіля для підтримки безпечної дистанції до транспортних засобів, що їдуть попереду. Дана технологія також відома як динамічний круїз-контроль.

Управління базується на інформації з бортових датчиків. Такі системи можуть використовувати радар, лазерний датчик або налаштування камери, що дозволяє транспортному засобу гальмувати, коли він виявляє, що автомобіль наближається до іншого автомобіля попереду, а потім прискорюватися, коли це дозволяє дорожній рух.

Технологія ACC вважається ключовим компонентом майбутніх поколінь інтелектуальних автомобілів. Ця технологія покращує безпеку та зручність пасажирів, а також збільшує пропускну здатність доріг за рахунок підтримки оптимальної відстані між транспортними засобами та зменшення помилок водія. Адаптивний круїз-контроль не забезпечує повної автономності: система лише трохи допомагає водієві, але не керує автомобілем самостійно.

ACC на основі радара фокусується на двох основних параметрах:

- відстань;
- швидкість.

Це допомагає зменшити стрес водія. ACC допомагає підтримувати належну відстань між транспортними засобами з використанням регулювання швидкості.

Для різних швидкостей ACC корисний для підтримки дистанції між транспортними засобами, особливо низькошвидкісний ACC на основі радара дуже корисний, він допомагає працювати в пробках для підтримки короткої дистанції між транспортними засобами.

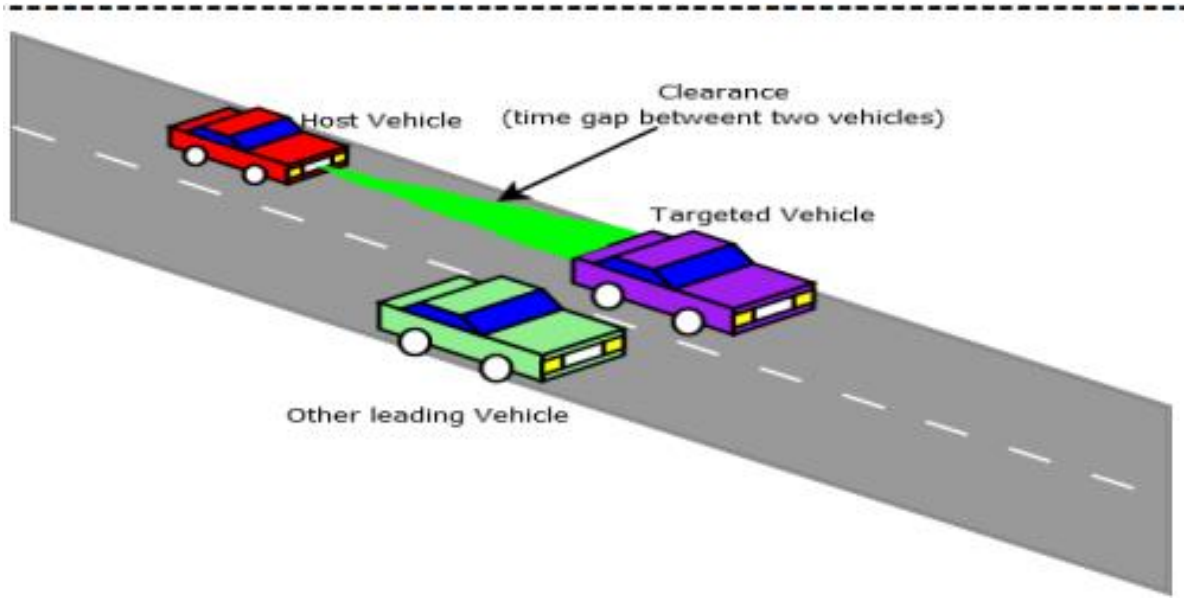

Рисунок 3.2 – Робота ACC

На рисунку 3.2 пояснюється робота ACC за допомогою радарного датчика. Тут автомобіль з радарним датчиком посилає хвилю і вимірює відстань між транспортними засобоми. Окрема дистанція між двома транспортними засобами відома як час до перешкоди.

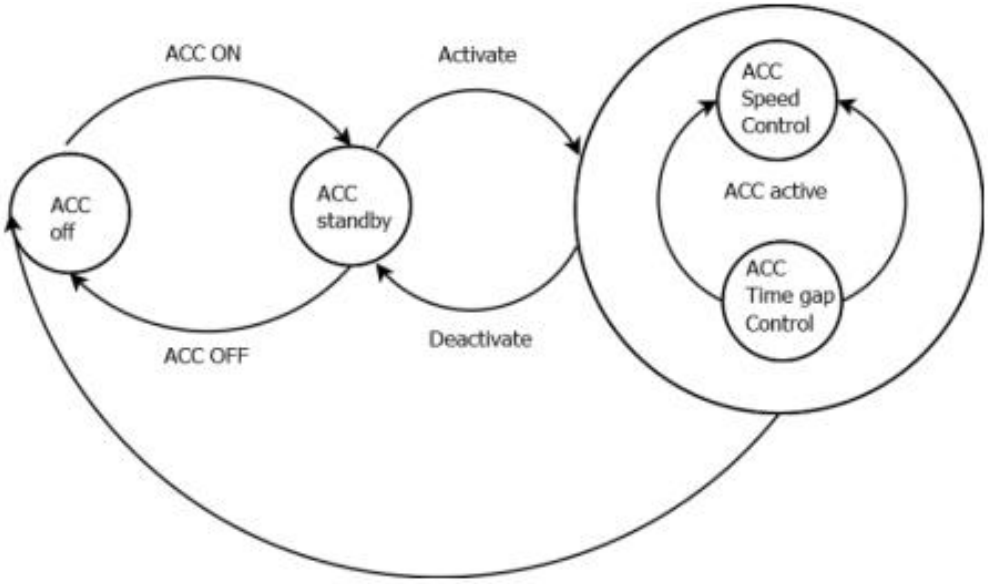

Рисунок 3.3 – Діаграма станів ACC

На рисунку 3.3. зображено структурну схему пристрою ACC. На цій діаграмі на початку ACC вимкнений, коли автомобіль заводится, стан стає увімкненим ACC і надсилає сигнали до автомобіля, під час надсилання інформація ACC знаходиться в режимі очікування. Цей режим очікування надсилає сигнали транспортному засобу та керує автомобілем за допомогою контролю швидкості та часового проміжку до лідируючого транспортного засобу [10]. Коли автомобіль зупиняється позаду лідируючого автомобіля, ACC вимикається або дезактивується.

Коли попереду немає автомобіля ACC працює за принципом швидкості. Коли транспортний засіб попереду рухається на меншій швидкості приблизно на 5 км/год, дросельна заслінка контролюється для дотримування дистанції між транспортними засобами. Якщо транспортного засобу попереду немає, система буде підтримувати автомобіль на початковій або поточній швидкості, а коли перешкода знаходиться перед транспортним засобом, вона буде зменшувати швидкість транспортного засобу під час зміни смуги руху, дороги тощо. ACC вдосконалений, щоб керувати автомобілем на помірній швидкості до 40 км/год і вище. Радар сканує передню частину транспортного засобу та перевіряє швидкість різних транспортних засобів, щоб підтримувати швидкість основного автомобіля і відстань. Зазвичай таке дотримування відстані призводить до більшого споживання палива та збільшує викиди CO2 тим більше, чим більша відстань між транспортними засобами відносно трафіку. ACC має верхній і нижній рівень контролю, якщо верхній контрольний рівень обчислює бажане прискорення, то нижній рівень визначає вхідний сигнал приводу дросельної заслінки як крутний момент. Верхній рівень складається з двох режимів:

- контроль швидкості;
- контроль відстані.

Для обчислення використовується перша частина контролера, тобто верхній рівень, необхідно розрахувати прискорення транспортного засобу, в той час час нижній рівень контролює положення дросельної заслінки для

керування двигуном, тобто визначає прискорення для підтримки дистанції та регулює швидкість головного автомобіля.

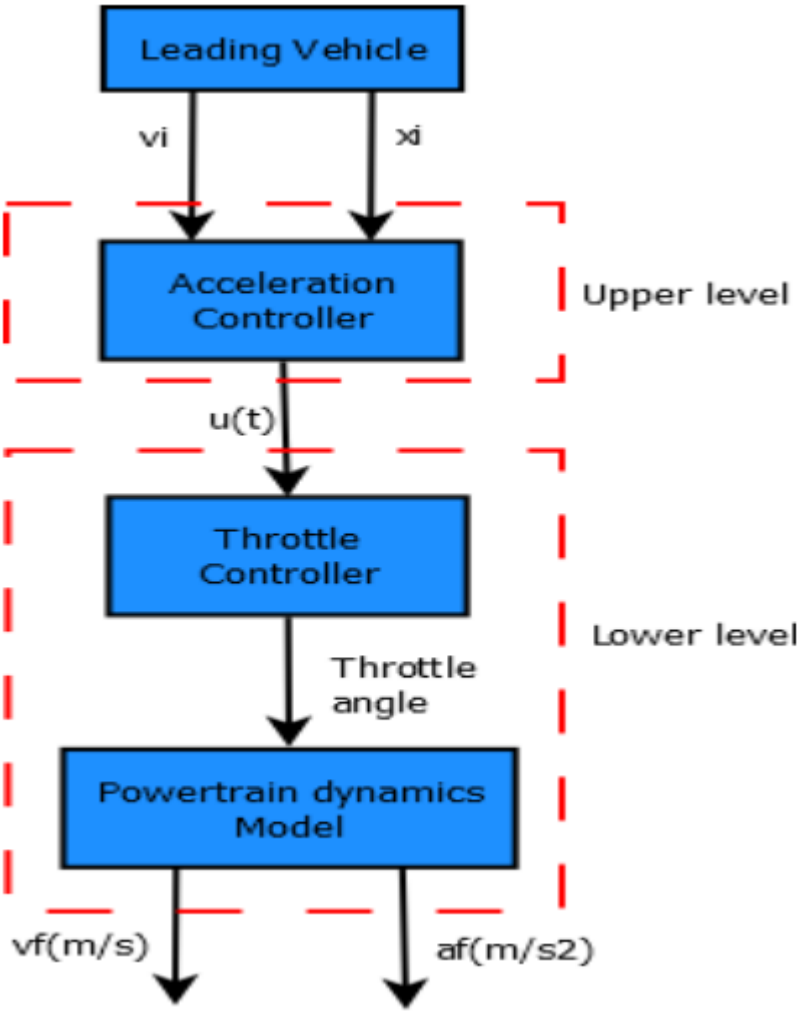

Рисунок 3.4 – Схема роботи ACC

Рисунок 3.4 описує роботу контролера верхнього рівня. Цей контролер працює на базі радарного датчика зчитування, щоб виявити інші транспортні засоби перед головним транспортним засобом, виміряти швидкість і відстань.

Тут vi і vf – відповідна швидкість лідируючого автомобіля і головного транспортного засібу. Фактична відстань між транспортним засобом, df:

$$df = xl - xf, \qquad (3.1)$$

де, xl – поздовжнє положення лідируючого автомобіля, xf – поздовжнє положення головного автомобіля.

ПІД-регулятор використовується для визначення кута дросельної заслінки. Для обчислення кута дросельної заслінки використовується наступне рівняння

$$a = a0 + K1\delta f + K2ef + \int (K3\delta f + K4ef)\, d\tau, \qquad (3.2)$$

де, a0 – контрольне значення рівноваги, $K1\delta f$ – інтегральна дія, $K2ef$ – похідна, K1-K4 – прибутки. Це допомагає обчислити кут, щоб забезпечити необхідне прискорення використовуючи інформацію від верхнього контролера. Вихід обчислюється за відсотком кута дросельної заслінки та динамічною моделлю трансмісії. Діапазон кута дросельної заслінки від 0° до 90°, тобто від 0% - 100% відповідно.

3.3 Інтелектуальний круїз-контроль

Інтелектуальний круїз-контроль (ІКК), або інтелектуальна система підтримки швидкості (ISA) – це система, яка гарантує, що швидкість транспортного засобу не перевищує безпечну або встановлену законом швидкість. У разі можливого перевищення швидкості водія можна сповістити або швидкість автоматично знизиться.

Інтелектуальний круїз-контрол використовує інформацію про дорогу для визначення місцевих обмежень швидкості. Інформацію можна отримати, знаючи положення автомобіля, враховуючи обмеження швидкості, відомі для цього положення, а також шляхом інтерпретації дорожніх особливостей, таких як знаки. Система ІКК призначена для виявлення та попередження водія, коли транспортний засіб увійшов у нову зону швидкості або коли діють інші обмеження швидкості відповідно до часу доби та умов. Багато систем ІКК також надають інформацію про небезпеку водіння та обмеження

швидкості та камери відеоспостереження на світлофорах. Мета ІКК – допомогти водієві підтримувати безпечну швидкість.

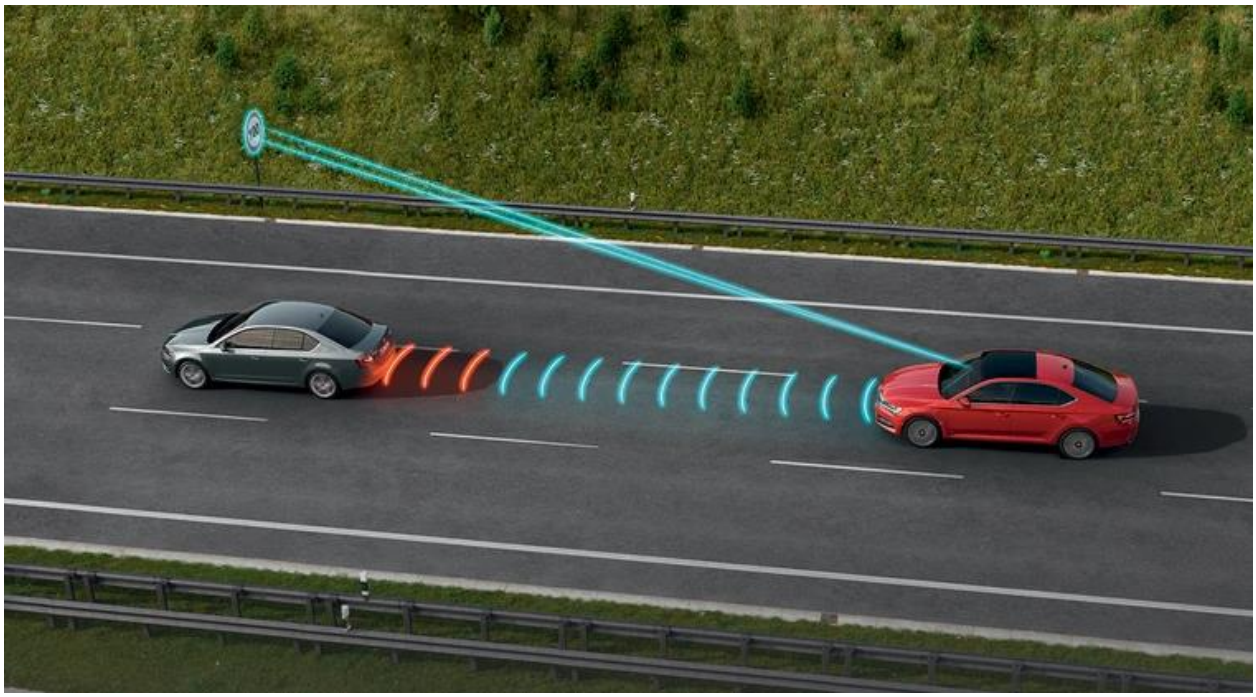

Рисунок 3.5 – Робота ІКК

Інтелектуальний круїз-контроль – еволюція адаптивного круїз-контролю. Дана система може автоматично регулювати встановлену швидкість автомобіля в залежності від обмежень швидкості на дорожнії знаках, наприклад коли автомобіль з’їзджає з автомагістралі до міста, система сповільнює автомобіль, і навпаки підвищує швидкість при виїзді з міста.

Для визначення місцевих обмежень швидкості на дорозі та швидкості транспортного засобу доступні чотири типи технології:

- позиційні системи;
- радіомаяки;
- оптичне розпізнавання;
- розрахунок контрольних точок.

В позиційних системах використовується GPS, яка базується на мережі

супутників, які постійно передають радіосигнали. GPS-приймачі вловлюють ці передачі та порівнюють сигнали від кількох супутників, щоб точно визначити місцезнаходження приймача з точністю до кількох метрів. Це робиться шляхом порівняння часу, коли сигнал було надіслано із супутника, з моментом, коли його було прийнято приймачем. Оскільки орбітальні траєкторії супутників відомі, приймач може виконати обчислення на основі відстані до кількох орбітальних супутників і, отже, визначити своє положення.

Технологія оптичного розпізнавання зосереджена на розпізнаванні знаків швидкості, дорожньої розмітки та придорожніх об'єктів [11]. Ця система вимагає, щоб транспортний засіб проїхав знак швидкості або подібний покажчик, а дані про знак або покажчик були зареєстровані сканером або системою камер. Коли система розпізнає знак, дані про обмеження швидкості отримуються та порівнюються зі швидкістю автомобіля. Система використовує це обмеження швидкості, доки не виявить знак швидкості з іншим обмеженням.

Розрахунок контрольних точок (КТ) використовує механічну систему, пов’язану з вузлом приводу автомобіля, щоб передбачити шлях, яким проходить транспортний засіб. Вимірюючи обертання опорних коліс з часом, можна зробити досить точну оцінку швидкості транспортного засобу та пройденої відстані. Розрахунок контрольних точок вимагає, щоб транспортний засіб починав з відомої фіксованої точки. Потім, об’єднавши дані про швидкість і відстань із такими факторами, як кут повороту керма та зворотний зв’язок від спеціалізованих датчиків, він може побудувати шлях, яким пройшов автомобіль. Накладаючи цей шлях на цифрову карту, система КТ приблизно знає, де знаходиться транспортний засіб, яке місцеве обмеження швидкості та швидкість, з якою рухається транспортний засіб. Потім система може використовувати інформацію, надану цифровою картою, щоб попереджати про майбутні та надавати попередження, якщо обмеження швидкості перевищено.

Розрахунок контрольних точок схильний до кумулятивних помилок вимірювання, таких як коливання між припущеним окружністю шин порівняно з фактичним розміром. Інші похибки вимірювань накопичуються, коли транспортний засіб рухається по плавним поворотам, якщо інерційні датчики недостатньо чутливі, щоб виявити поворот, або через електромагнітний вплив на магнітний потік компасів, наприклад, через проїзд під лініями електропередач або під час руху через сталевий міст або через підземні переходи та дорожні тунелі.

Принциповим обмеженням ІКК є те, що це може призвести до руху з обмеженням швидкості, а не місцевими умовами. Дорожні особливості, такі як повороти та схили, можуть вимагати нижчої швидкості, ніж указане максимальне обмеження швидкості.

Системи ІКК можуть служити бортовими реєстраторами даних транспортного засобу, зберігаючи інформацію про місцезнаходження автомобіля та його продуктивність для подальшої перевірки та управління автопарком. Як остаточний висновок можна зазначити, що ІКК може мати переваги в безпеці дорожнього рух. ІКК можна розглядати як систему підтримки безпеки дорожнього руху, але не можна розглядатися як єдиний спосіб зменшити порушення швидкості. Використання ІКК також має бути спрямоване на інші аспекти безпеки дорожнього руху, політики щодо зниження швидкості: краще розуміння водіями зон швидкості та краще запровадження обмежень швидкості у зв'язку з використанням дороги[12].

3.4 Використання технології реконфігурації в системі круїз-контролю.

Адаптивний круїз-контроль є вбудованою системою, що працює в реальному часі. Для даної системи доцільно використовувати високопродуктивні інтегральні схеми, наприклад сімейство virtex 7 – вони включають можливість швидкого обміну даними та швидкої обробки цих даних системою, що є необхідною умовою для систем, критичних до часу.

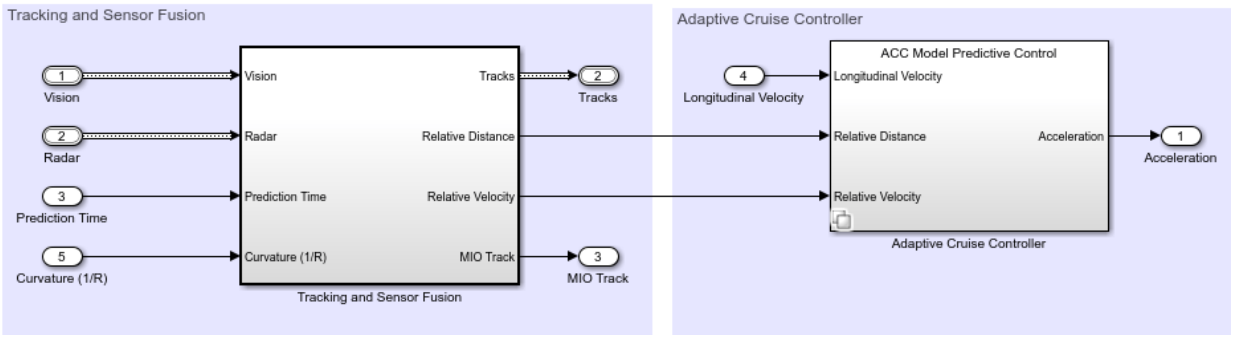

Рисунок 3.6 – Структурна схема роботи адаптивного круїз-контролю

Усі програмовані користувачем функції всередині пристроїв Xilinx FPGA та SoC контролюються комірками енергозалежної пам'яті, які необхідно налаштувати під час увімкнення. Ці комірки пам'яті спільно відомі як конфігураційна пам'ять. Вони визначають рівняння LUT, маршрутизацію сигналу, стандарти напруги IOB та всі інші аспекти конструкції. Архітектури Xilinx FPGA і SoC мають конфігураційну пам’ять, організовану в кадри, які розміщуються на пристрою. Ці кадри є найменшими адресованими сегментами пристрою. Реконфігуровані фрейми побудовані на цих конфігураційних фреймах, і це мінімальні будівельні блоки для виконання динамічної реконфігурації[23]. Базові регіони в ПЛІС серії virtex 7:

- CLB: 50 у висоту на 1 в ширину;
- DSP48: 10 у висоту на 1 в ширину;
- Блок RAM: 10 у висоту на 1 у ширину.

Швидкість конфігурації безпосередньо залежить від розміру часткового BIT файлу і пропускної здатності конфігураційного порту.

Таблиця 3.1 – Пропускна здатність конфігураційного порту для virtex 7

Configuration Mode	Max Clock Rate	Data Width	Maximum Bandwidth
1	2	3	4
ICAP	100 MHz	32 bit	3.2 Gb/s
SelectMAP	100 MHz	32 bit	3.2 Gb/s

Продовження таблиці 3.1

1	2	3	4
Serial Mode	100 MHz	1 bit	100 Mb/s
JTAG	66 MHz	1 bit	66 Mb/s

Технологія FPGA забезпечує гнучкість програмування та перепрограмування без повторного виготовлення зі зміненою конструкцією. Часткова реконфігурація розширює цю гнучкість, дозволяючи модифікувати робочу FPGA, проектування проходить шляхом завантаження часткового файлу конфігурації, як правило, часткового файлу BIT. Після того як файл BIT конфігурує FPGA, часткові BIT-файли можна завантажити для зміни реконфігурованих областей в FPGA без шкоди для цілісності додатків, що працюють на частині пристрою, які не перенастроюються[24].

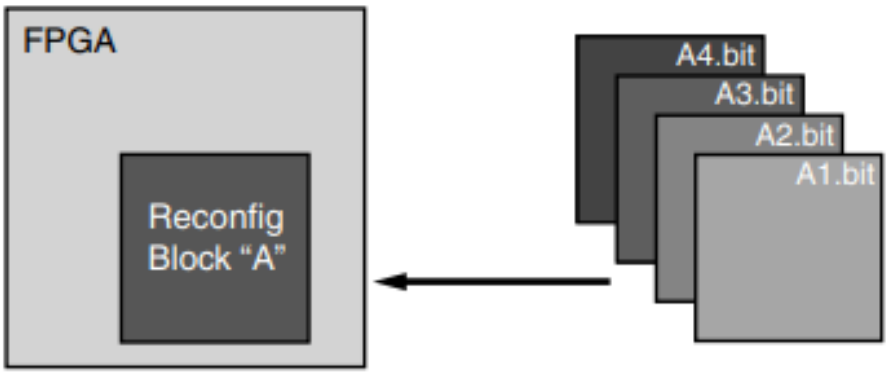

Рисунок 3.7 – Зображення часткової реконфігурації

Як показано на рисунку 3.7, функція, реалізована в реконфігурованому блоці A модифікується завантаженням одного з декількох BIT файлів. Логіка FPGA поділяється на два типи: логіка, що реконфігурується, та статична логіка. Сіре поле в блоці FPGA представляє статичну логіку і блок позначений реконфігурованим блоком A представляє логіку, що реконфігурується. Логіка, що реконфігурується заміняється новими даними з часткового BIT файлу.

Існує багато причин, чому можливість динамічно змінювати схему FPGA пристрою є перевагою:

- покращує відмовостійкість системи;
- зменшує розмір пристрою FPGA необхідний для реалізації поставленої функції, разом зі зменшенням ціни і енергоспоживання;
- надає гнучкості в виборі алгоритму чи доступного протоколу для застосунку;
- дозволяє використовувати нові технології при розробці безпеки;
- покращує конфігуровані розрахунки.

В додаток до зменшення розміру, ваги, енергоспоживання та ціни часткова реконфігурація дозволяє використовувати нові типи розробки, що є неможливими для реалізації без неї.

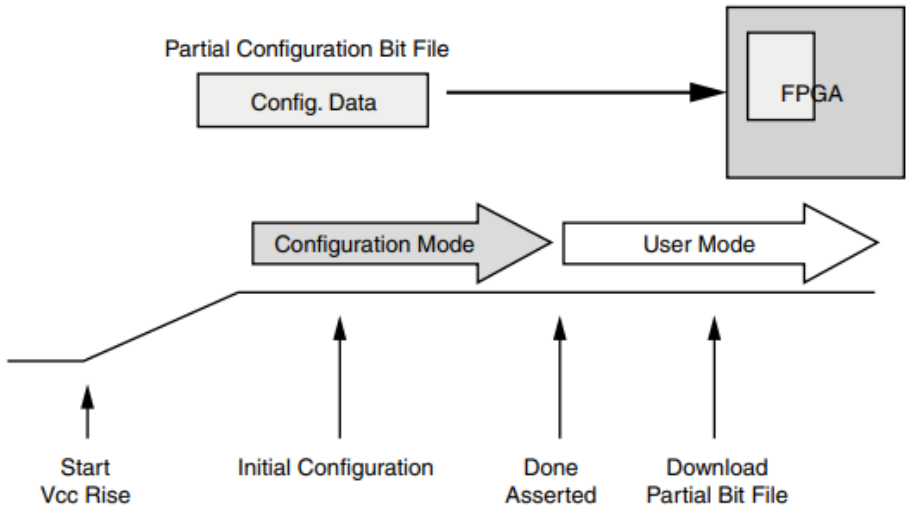

Рисунок 3.8 – Етапи часткової динамічної реконфігурації

Часткова динамічна реконфігурація робить систему круїз-контролю більш гнучкою, дозволяє динамічно адаптувати алгоритм до змін зовнішніх умов та характеристик транспортного засобу. ЧДР дозволяє створювати системи, що самореконфігуруються, враховуючи безліч факторів для збільшення безпеки та комфорту використання системи.

4 РЕАЛІЗАЦІЯ СЦЕНАРІЮ ПІДТРИМУВАННЯ ПОСТІЙНОЇ ШВИДКОСТІ

4.1 Математична модель

Система круїз-контролю автомобіля – це поширена система зворотного зв'язку, яка зустрічається в повсякденному житті. Система намагається підтримувати постійну швидкість у присутності збурень, в основному викликаних змінами ухилу дороги. Контролер компенсує ці невідомі, вимірюючи швидкість автомобіля та регулюючи дросель [13].

Для моделювання системи розглянемо з блок-схему на рисунку 4.1. Нехай v буде швидкість автомобіля і v_r бажана швидкість. Контролер, який як правило, має пропорційно-інтегральний тип, приймає сигнали v і v_r і генерує керуючий сигнал u, який надсилається на привід, що контролює положення дросельної заслінки. Дросель, у свою чергу, контролює крутний момент T генеруємий двигуном, який передається через шестерні та колеса, створюючи силу F, яка рухає автомобіль. Існують збурювальні сили F_d через зміни ухилу дороги, опору крученню та аеродинамічних сил.

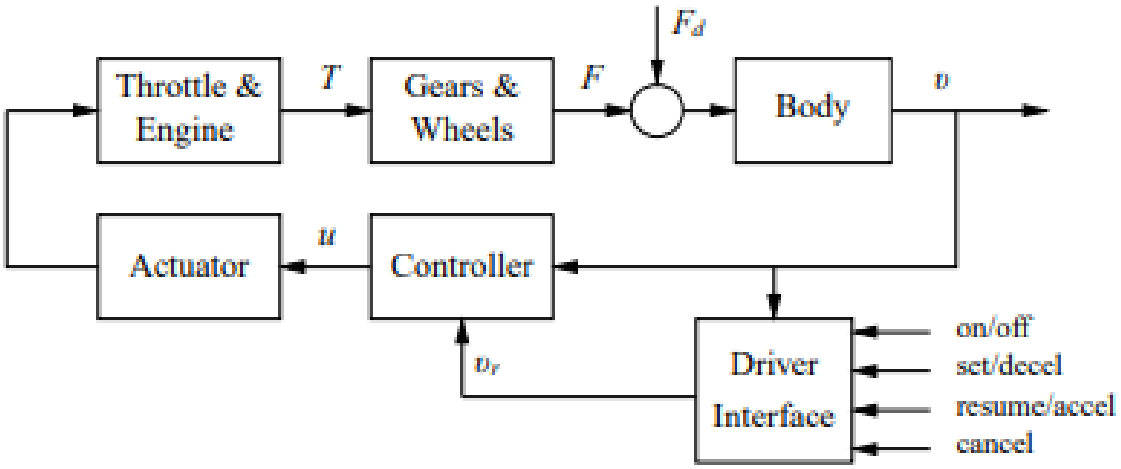

Рисунок 4.1 – Блокова діаграма системи круїз-контролю

Система складається з багатьох окремих компонентів:

- двигун;
- трансмісія;
- привід дросельної заслінки;
- колеса;
- кузов автомобіля.

Для розробки математичної моделі потрібно почати з балансу сил які діють на кузов автомобіля. Нехай v – швидкість автомобіля, m – загальна маса, F – сила породжена контактом коліс із дорогою, а F_d – сила збурення через силу тяжіння, тертя та аеродинамічного опору. Рівняння руху автомобіля таке

$$m\frac{dv}{dt} = F - F_d. \tag{4.1}$$

Сила F створюється двигуном, крутний момент якого пропорційний швидкості впорскування палива, який сам по собі пропорційний сигналу керування $0 \leq u \leq 1$, який контролює положення дросельної заслінки. Крутний момент також залежить від частоти обертання двигуна ω. Представлення крутного моменту при повному газі дається кривою крутного моменту

$$T(\omega) = T_m\left(1 - \beta\left(\frac{\omega}{\omega_m} - 1\right)^2\right), \tag{4.2}$$

де максимальний крутний момент T_m досягається при частоті обертання двигуна ω_m. Нехай n передавальне число і r радіус колеса. Частота обертання двигуна пов'язана зі швидкістю через вираз

$$\omega = \frac{n}{r}v = a_n v, \tag{4.3}$$

і рушійну силу можна записати як

$$F = \frac{nu}{r}\, T\,(\omega) = \; a_n u\, T(a_n v). \tag{4.4}$$

Обернене a_n має фізичну інтерпретацію як ефективний радіус колеса. На рисунку 3.2 показано крутний момент як функцію частоти обертання двигуна та швидкості автомобіля. На рисунку 4.2 показано, що ефект передачі полягає в тому, щоб згладити криву крутного моменту, щоб повний крутний момент можна було отримати майже у всьому діапазоні швидкостей.

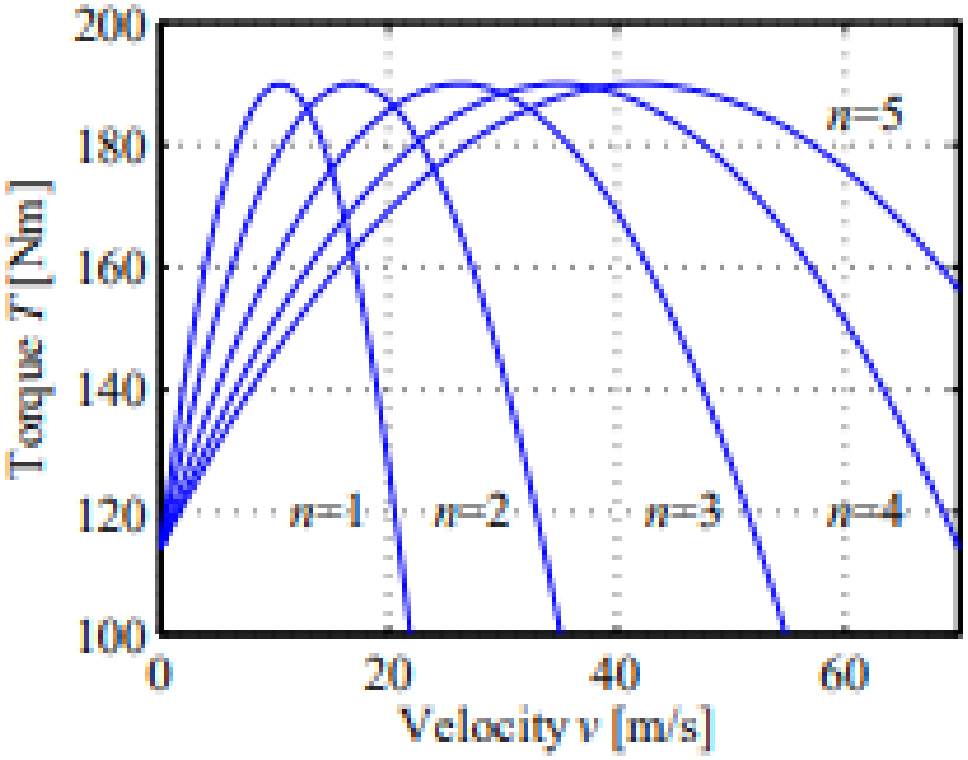

Рисунок 4.2 – Швидкість автомобіля на різних передачах

Збурювальна сила F_d складається з трьох основних компонентів: F_g – сили тяжіння, F_r – сили тертя кочення і F_a – аеродинамічний опір. Нехай нахил дороги буде θ, сила тяжіння

$$F_g = \text{mg sin } \theta, \tag{4.5}$$

як показано на рисунок 4.3, де g = 9,8 м/с^2 гравітаційна стала. Модель тертя кочення є

$$F_r = mgC_r sgn(v), \tag{4.6}$$

де C_r є коефіцієнтом тертя кочення, а sgn(v) є знаком v: +1. -1 або 0, якщо v = 0. Типове значення коефіцієнта тертя кочення C_r = 0,01.

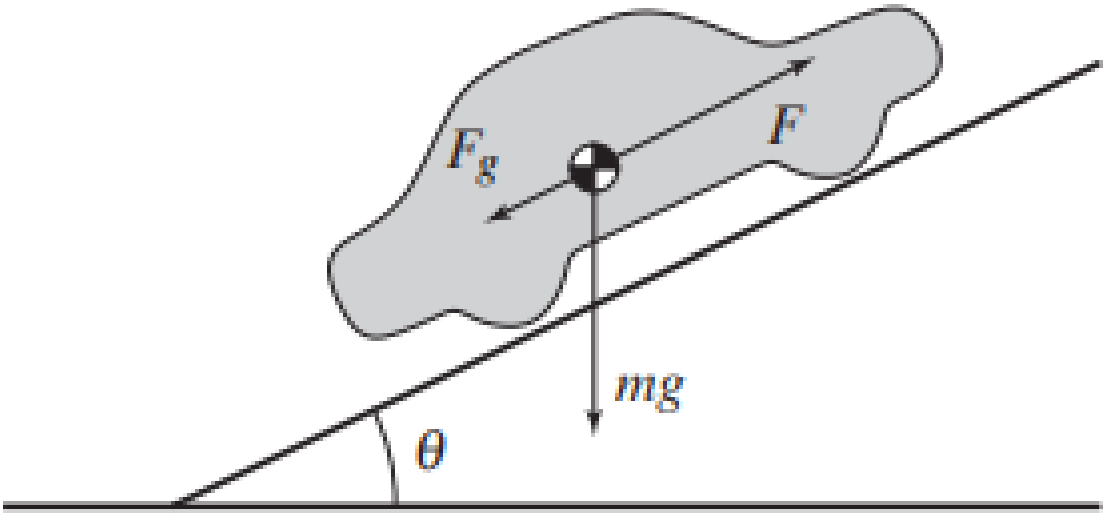

Рисунок 4.3 – Дія сил гравітації

Аеродинамічний опір пропорційний квадрату швидкості:

$$F_a = \frac{1}{2}\rho\, C_d A v^2, \tag{4.7}$$

де ρ – густина повітря, C_d – залежний від форми коефіцієнт аеродинамічного опору а A — лобова площа автомобіля.

З цих формул можна вивести формулу для моделі автомобіля

$$m\frac{dv}{dt} = a_n u\, T(a_n v) \; - \; mgC_r sgn(v) - \frac{1}{2}\rho\, C_d A v^2 - \; \text{mg sin } \theta, \quad (4.8)$$

де функція T задана рівнянням 4.2. Модель з формули 4.8 є динамічною системою першого порядку. Станом є швидкість автомобіля v, яка також є виходом. Вхід – це сигнал u, який контролює положення дросельної заслінки, а збурення – це сила F_d , яка залежить від ухилу дороги. Система є нелінійною через криву крутного моменту, силу гравітації, нелінійний характер тертя кочення і аеродинамічний опір. Параметри також можуть відрізнятися; наприклад, маса автомобіля залежить від кількості пасажирів і вантажу, що перевозиться в ньому автомобіль.

Треба додати до цієї моделі контролер зворотного зв'язку, який намагається регулювати швидкість автомобіля за наявності зовнішніх збуджень. Будемо використовувати пропорційно-інтегральний регулятор, який має вигляд

$$u(t) = \; k_p e(t) + \; k_i \int_0^t e(\tau) d\tau. \quad (4.9)$$

Сам контролер може бути реалізований як динамічна система введення/виведення шляхом визначення стану регулятора s та реалізації диференціального рівняння

$$\frac{ds}{dt} = \; v_r - v, \;\; u = \; k_p(v_r - v) + k_i s, \quad (4.10)$$

де vr – це бажана швидкість. Інтегратор гарантує, що в усталеному стані помилка буде доведена до нуля, навіть якщо є збурення або помилки моделювання. Рисунок 4.4 показує реакцію замкнутої системи, що складається з рівнянь 4.8 і 4.10, коли вона стикається зі зміном нахилу дороги. На рисунку показано, що навіть якщо пагорб настільки крутий, що дросель змінює невеликий кут відкриття дросельної заслінки до майже

повного газу, найбільша похибка швидкості становить менше 1 м/с, а бажана швидкість відновлюється через невеликий час.

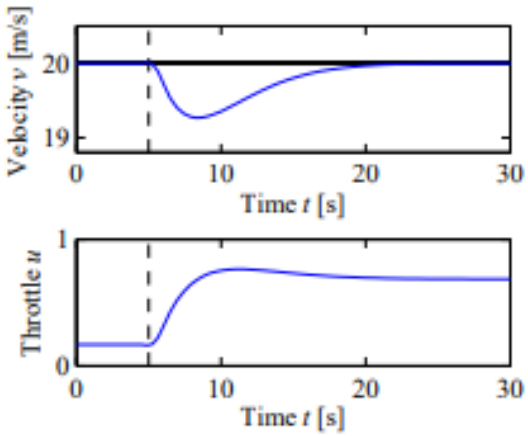

Рисунок 4.4 – Реакція системи на зміну кута нахилу дороги

4.2 Вибір системи на кристалі

При виборі системи на кристалі, або system on a chip (SoC), на якій буде запускатись програмний код, потрібно орієнтуватися на продуктивність та технології реконфігурації. Для реалізації найбільш продуктивних проектів доцільно використовувати сучасні пристрої, які підтримують ефективні технології проектування, з підтримкою часткової динамічної реконфігурації, та забезпечують необхідну інструментальну підтримку для спрощення проектування [14]. В атестаційній роботі було обрано SoC сімейства Virtex-7 компанії Xilinx.

Сімейство Virtex-7 було представлено в червні 2010 року на 28-нм технологічним процесі і, як повідомляється, забезпечує подвійне підвищення продуктивності системи при зниженні на 50 відсотків потужності порівняно з пристроями попереднього покоління Virtex-6. Крім того, Virtex-7 подвоює пропускну здатність пам’яті порівняно з попереднім поколінням Virtex FPGA з продуктивністю інтерфейсу пам’яті 1866 Мбіт/с і понад двома мільйонами логічних комірок. Особливості сімейства Virtex-7:

- включає три підродини оптимізовані для різного застосування;
- до двох мільйонів логічних комірок;
- швидкість передачі до 28.05 Гбіт/сек;
- об'єм пам'яті до 96Мбіт;
- блок обробки аналогових сигналів;
- призначена для побудови систем провідного та бездротового зв'язку, радарів, цивільних систем обробки інформації.

Крім того Сімейство Virtex-7 пропонує сучасні технології реконфігурації, наприклад часткову динамічну реконфігурацію. Часткова реконфігурація – це можливість динамічно змінювати блоки логіки шляхом завантаження часткових бітових файлів, у той час як решта логіки продовжує працювати без перерв. Технологія часткової реконфігурації Xilinx дозволяє розробникам змінювати функціональні можливості на льоту, усуваючи необхідність повністю переконфігурувати та повторно встановлювати зв’язки, значно підвищуючи гнучкість, яку пропонують FPGA. Використання часткової реконфігурації може дозволити розробникам перейти до меншої кількості пристроїв або менших пристроїв, зменшити енергоспоживання та покращити можливість оновлення системи. Ефективніше використовуйте кремній, завантажуючи лише ті функції, які потрібні в будь-який момент часу.

4.3 Вибір технології для реалізації системи

Традиційно розробник має в основному наступні три варіанти реалізації апаратної платформи керування машиною:

- цифровий сигнальний процесор (DSP);
- реконфігурована FPGA;
- інтегральна схема, спеціалізована для вирішення конкретного завдання (ASIC).

ASIC забезпечують найвищу продуктивність, оскільки їх конструкція може бути оптимально налаштована по відношенню до вимоги програми. Однак витрати на рішення на основі ASIC є найвищими, частково через невелику кількість вироблених готових схем. На іншому боці – рішення на основі DSP економічно ефективні, але забезпечують неоптимальну швидкість обробки за рахунок виконання програмного забезпечення [15].

Між цими двома крайнощами реконфігурована FPGA забезпечує компроміс, який підходить для багатьох застосувань. Протягом останніх років розвиток технології FPGA просунувся з точки зору використовуваних ресурсів і кількісті комірок, швидкісті обробки, споживання енергії та ціни. Таким чином, найсучасніші FPGA є альтернативами звичайним мікроконтролерам і ASIC для розробки програмного продукту.

Сучасні системи на основі FPGA поєднують багато переваг відносно DSP і ASIC. Це включає швидкий цикл розвитку, високу гнучкість і багаторазове використання, помірні витрати, легка модернізація. Крім того, сучасні FPGA дозволяють інтегруватися з програмними процесорамив. Тобто FPGA може використовувати типові можливості процесора.

У порівнянні з DSP реконфігуровані FPGA мають ряд переваг

- низьку затримку. Низька затримка критично важлива при розробці систем, що працюють в реальному часі, де час між введенням данних і отримкою вихідного сигналу повинен бути якомога коротший. В цьому FPGA набагато кращі, ніж DSP;

- підключення джерела даних. До FPGA можливо підключити будь-яке джерело даних, таке як мережевий інтерфейс або датчик, безпосередньо до контактів мікросхеми. В той час як для DSP підключення відбувається через стандартизовані шини даних;

- енергоефективність. FPGA за рахунок конфігурованої схеми є більш енергоефективною ніж DSP.

Як мінус у порівнянні FPGA та DSP можна привести більш складну та дорожчу розробку. Мова для опису інтегральніх схем більш складна для

розробки, ніж мови прграмування по типу C++, Java. Конфігурація FPGA займає більше часу ніж компіляція програмного коду для DSP.

З цього можна зробити висновок, що для систем, що працюють в реальному часі, по типу системи круїз-контролю, доцільніше використовувати FPGA, хоч це буде затратніше в розробці.

ASIC має ті самі переваги щодо DSP, що і FPGA, і навіть є більш ефективною, але за рахунок технології реконфігурації FPGA має ряд переваг, основні: масштабованості і можливістю модернізації в майбутньому.

Реконфігурація поділяється на:

- повну;
- часткову.

Повна реконфігурація – це реконфігурація одного чи кількох фізичних модулів системи у повному об’ємі. Часткова реконфігурація – це процес реконфігурації частини фізичного модуля системи. Якщо перший спосіб відбувається за повної зупинки роботи фізичного модуля на час виконання реконфігурації, то у другому випадку модуль зможе виконувати певні функції, часткова реконфігурація відбувається в режимі часу виконання. Повна реконфігурація за застосування в SoC відповідає статичній реконфігурації. Як окремий випадок повної динамічної реконфігурації розглядають реконфігурацію реального часу – з розширенням та вдосконаленням технології FPGA стає можливо зменшити час, необхідний на виконання реконфігурації, що дозволяє виконувати повну реконфігурацію між різними стадіями чи режимами роботи FPGA. В мультикристальних SoC як повна, так і часткова реконфігурація виконується як в статичному, так і в динамічному режимах.

Сімейство Virtex 7, обране як база для розроблюваної системи, підтримує сучасні технології часткової динамічної реконфігурації. Дана реконфігурація пропонує проводити реконфігурацію розділами, що дозволяє оптимізувати реконфігуровані модулі [16]. Дана технологія підтримується й на рівні інструментальних засобів, що вирішує проблему забезпечення

однорідності вузлів та встановлення зв'язків між ними. Таке розміщення забезпечує найбільш ефективну реалізацію з точки зору використання ресурсів, розведення зв'язків, розміру бітової послідовності, часу реконфігурації.

4.4 Алгоритми обчислень, застосовані при розробці

Орієнтуючись на рівняння 4.10 систему круїз-контролю можна реалізувати з використанням суматорів та помножувачів, як показано на рисунку 4.5. Так як система працює в реальному часі, дані елементи потрібно реалізувати на базі реконфігурування. Дані елементи повинні підтримувати n-розрядні операції, де n – кількість бітів, потрібних для арифметичних операцій для знаходження вихідного сигналу системи.

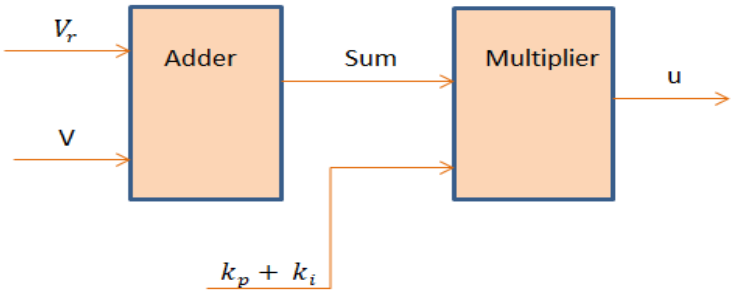

Рисунок 4.5 – Модель системи круїз-контролю

Сумматор можна реалізувати, використовуючи кілька повних суматорів для додавання n-розрядних чисел. Кожен повний суматор вводить Cin, який є Cout попереднього суматора. Вихід попереднього суматора передається як вхід до наступного суматора.

Повний суматор додає двійкові числа та враховує введені та виведені значення. Однорозрядний повний суматор додає три однорозрядні числа, які записуються як A, B і Cin, A і B є операндами, а Cin перенесено з попереднього етапу.[17] Вихідний перенос і сума зазвичай представлені

сигналами Cout і S, я показано на рисунку 4.6, де сума дорівнює 2Cout + S. Найпоширенішою реалізацією є S = A ⊕ B ⊕ Cin, де Cout = (A · B) + (Cin · (A ⊕ B)).

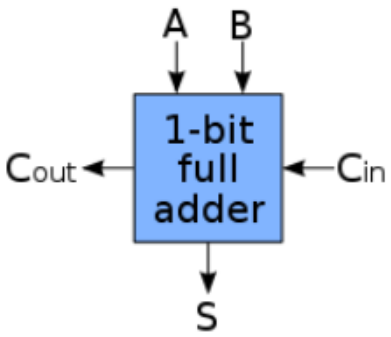

Рисунок 4.6 – Повний суматор

Алгоритм для помножувача зв’язаний з суматором та зсувом бітів, покроково його можна записати так:

- записати множене та множник;
- помножити молодший біт на множник;
- зсунути множник на один біт вліво;
- повторити попередні кроки;
- додати всі отримані числа.

Приклад алгоритма для помножувача зображено на рисунку 4.7

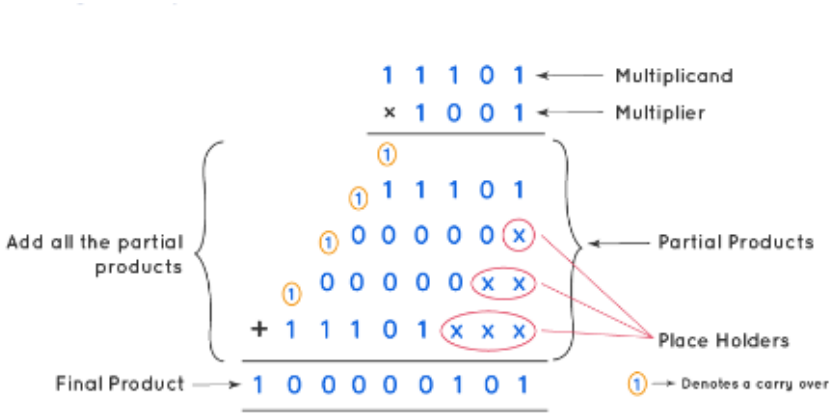

Рисунок 4.7 – Алгоритм роботи помножувача

Згідно з рівнянням 4.10 для роботи системи потрібно реалізувати інтегратор. Електронний інтегратор — це форма фільтра низьких частот першого порядку. Матиматичну формулу можна описати рівнянням

$$y(t) = \int_0^t x(\tau)d\tau + y_0, \qquad (4.11)$$

де x(t) – вхідна функція від часу, y(t) – вихідна функція від часу – результат інтегрування за час від 0 до t, k– коефіцієнт пропорційності, зворотний часу, y_0 – початкове значення вихідної змінної в момент часу t = 0 [18].

Інтегратор реалізується за допомогою конденсаторів, схема показана на рисунку 4.8, Вихідна напруга повільно збільшується на виході конденсатора, поки не досягне значення вхідної напруги.

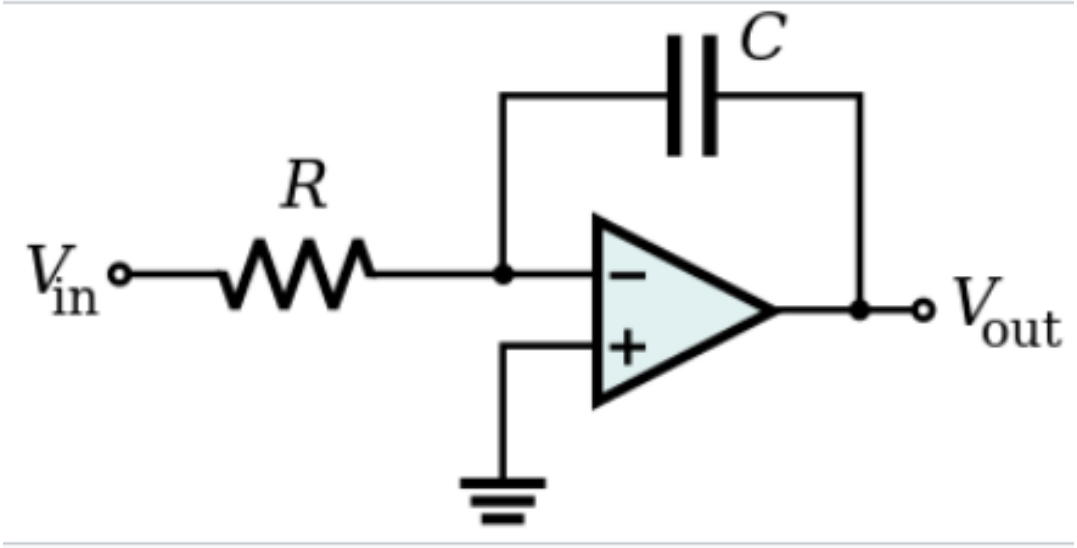

Рисунок 4.8 – Аналоговий інтегратор

Для роботи системи круїз-контролю потрібно реалізувати цифровий інтегратор. В цих інтеграторах вхідний і вихідний сигнали представлені у вигляді цифрових кодів. За своєю суттю є суматорами із накопиченням. На псевдокоді їхню роботу можна описати так:

Вихід_інтегратора = Вихід_інтегратора + Вхід * Інтервал, (4.12)

де інтервал вибірки – час від моменту отримання попереднього значення до моменту отримання поточного значення. Не обов'язково, щоб інтервал вибірки був справжнім часом.

4.5 Вибір технологій розробки інтегральних схем

В ході роботи потрібно розробити програму для FPGA, для цього було обрано мову VHDL – мова опису апаратури інтегральних схем. VHDL – це мова для опису цифрових електронних систем призначена для задоволення низки потреб у процесі проектування. [19].

Переваги мови VHDL:

- дозволяє моделювати і перевіряти поведінку необхідної системи до того, як проект буде використовуватися на реальному обладнанні;
- дозволяє описувати паралельну систему;
- проект VHDL є багатоцільовим. Будучи створеним один раз, блок розрахунків може використовуватися в багатьох інших проектах
- проект VHDL є переносним. Будучи створеним для однієї елементної бази, проект обчислювального пристрою може бути перенесений на іншу елементну базу;
- великою перевагою VHDL порівняно з Verilog є те, що VHDL має систему повного типу. Розробник можуть використовувати систему типів для написання більш структурованого коду.

При виборі середовищу розробки потрібно орієнтуватися на обрану мову, в нашому випадку VHDL, набір технологій для розробки, та підтримку розробниками. Існує декілька популярних середовищ розробки, що підтримують мову VHDL:

- Active-HDL від компанії Aldec;
- Vivado Design Suite компанії Xilinx;

- Incisive компанії Cadence.;
- Mentor Graphics ModelSim;
- EDA Playground;
- VHDL Simili;
- Freehd.

Для реалізації системи круїз-контролю на мові VHDL було обрано середу розробки Active-HDL.

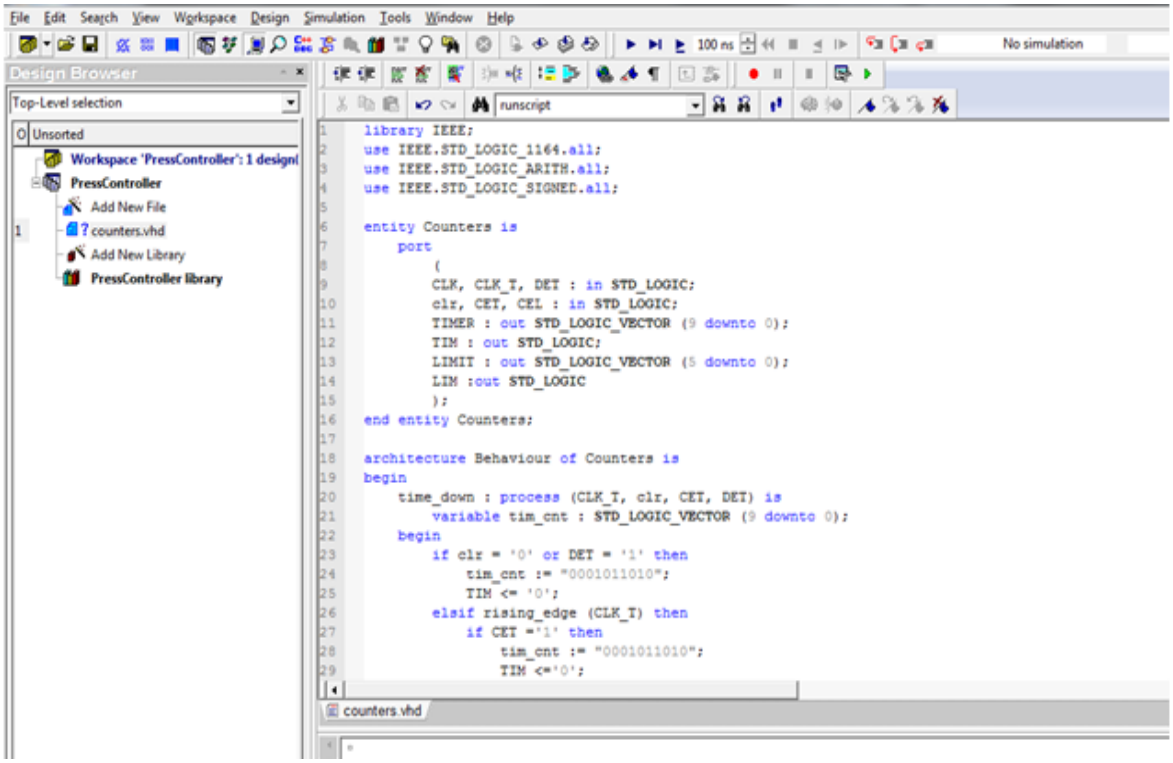

Рисунок 4.9 – Середовище розробки Active-HDL

Інтегроване середовище проектування Active-HDL включає повний набір інструментів HDL і графічного дизайну, а також змішаний мовний симулятор на рівні RTL/шлюзу для швидкого розгортання та перевірки конструкцій FPGA. Active-HDL підтримує провідні в галузі пристрої FPGA від Intel, Lattice, Microchip, Quicklogic, Xilinx та інших [20].

5 ПРОГРАМНА РЕАЛІЗАЦІЯ СЦЕНАРІЮ ПІДТРИМУВАННЯ ПОСТІЙНОЇ ШВИДКОСТІ

5.1 Розробка архітектури програмного продукту

Система круїз-контролю є вбудованою системою, що працює в реальному часі, тому було вирішено реалізовувати дану систему на основі скінченного автомату.

Скінченний автомат або кінцевий автомат – це математична модель обчислень. Це абстрактна машина, яка може перебувати в одному зі скінченної кількості станів у будь-який момент часу. Скінченний автомат може переходити з одного стану в інший у відповідь на деякі вхідні дані, перехід від одного стану до іншого називається переходом. Скінченний автомат визначається списком його станів, початковим станом і вхідними даними, які запускають кожен перехід[21]. Скінченні автомати бувають двох типів – детерміновані кінцеві автомати та недетерміновані кінцеві автомати.

Кінцевий автомат K визначається як набір

$$K = (A, Z, W, \delta, \lambda, a_1), \tag{5.1}$$

де $A = \{ a_1 ,..., a_Q \}$ – множина станів;

$Z = \{ z_1 ,..., z_N \}$ – множина вхідних сигналів;

$W = \{ w_1 ,..., w_M \}$ – множина вихідних сигналів;

δ – функція переходів, що визначає стан автомата в момент часу t+1 залежно від стану автомата та вхідного сигналу в момент часу t, інакше кажучи,

$$a_s = \delta (a_m, z), \tag{5.2}$$

де am стан автомата на момент часу t, z – вхідний сигнал у момент часу t, as– стан автомата на момент часу t+1, λ – функція виходів. Якщо функція λ за станом автомата a_m та вхідного сигналу z_n визначає значення вихідного сигналу w_m.

Скінчений автомат для системи круїз-контролю можна зобразити графом, як показано на рисунку 5.1

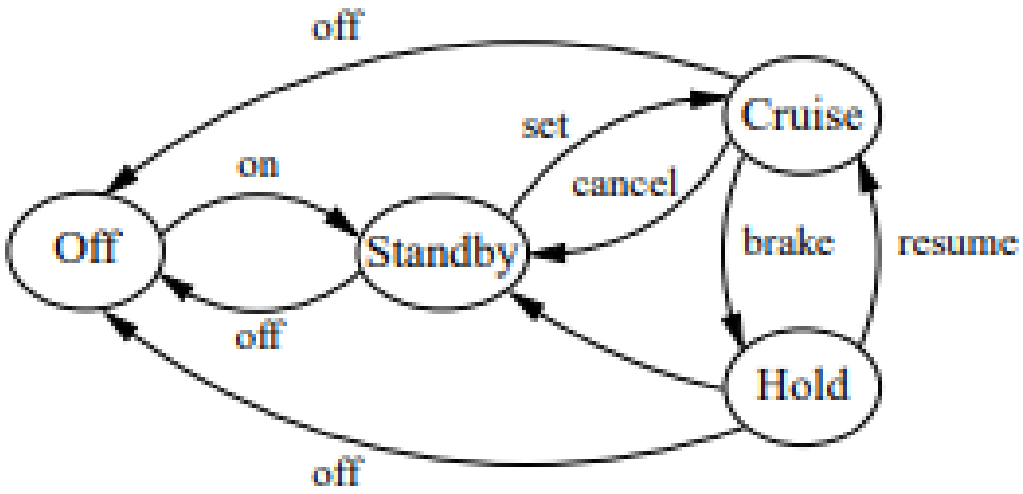

Рисунок 5.1 – Скінчений автомат для системи круїз-контролю

В початковому стані система вимкнена. При подачі сигналу на ввімкнення система переходить в стан очікування, де очікує на сигнал встановлення бажаної швидкості, або на сигнал вимкнення. Користувач може задати бажану швидкість, яка буде подана на вхід системи. Коли користувач задав бажану швидкість, система переходить в стан круїз-контролю. Якщо поточна і бажана швидкість співпадають, система переходить в стан утримування. Якщо поточна швидкість змінюється відносно бажаної, система визначає вихідний сигнал на дросельну заслінку, щоб збільшити чи зменшити поточну швидкість, на яку впливають зовнішні фактори. З усіх станів система може перейти в стан вимкнення і не буде регулювати швидкість.

Для реалізації системи потрібно розробити сумматор та помножувач. Суматор можна розробити на основі повного суматора, для цього потрібно поєднати n суматорів, де n – розрядність. Схема повного суматора зображена на рисунку 5.2.

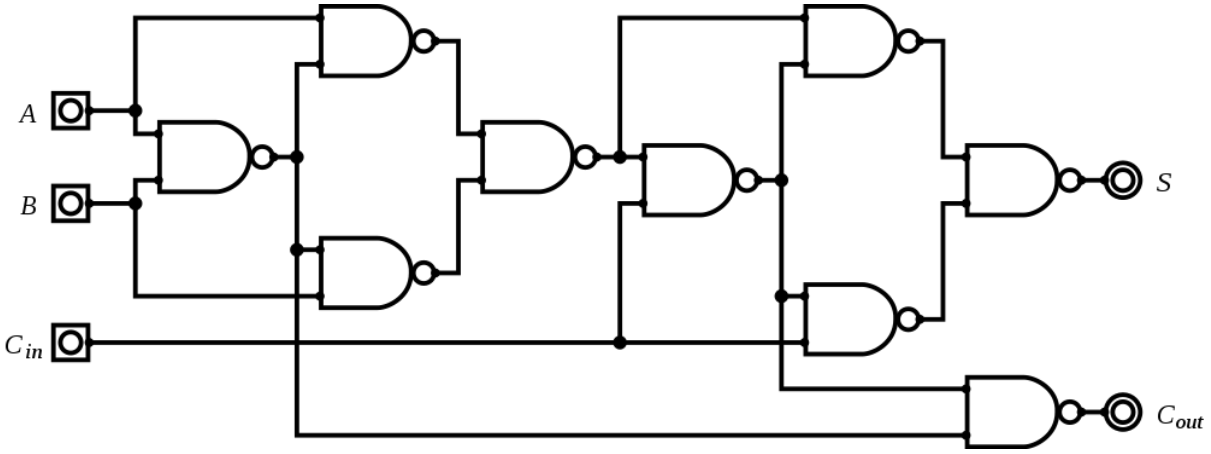

Рисунок 5.2 – Схема повного суматора

Далі повні суматори поєднуються в логічний блок, як на рисунку 5.3. Вихід переносу n-ного повного суматору є входом переносу n+1 повного суматору. Кожен повний суматор на виході має суму для n-ного біта [22].

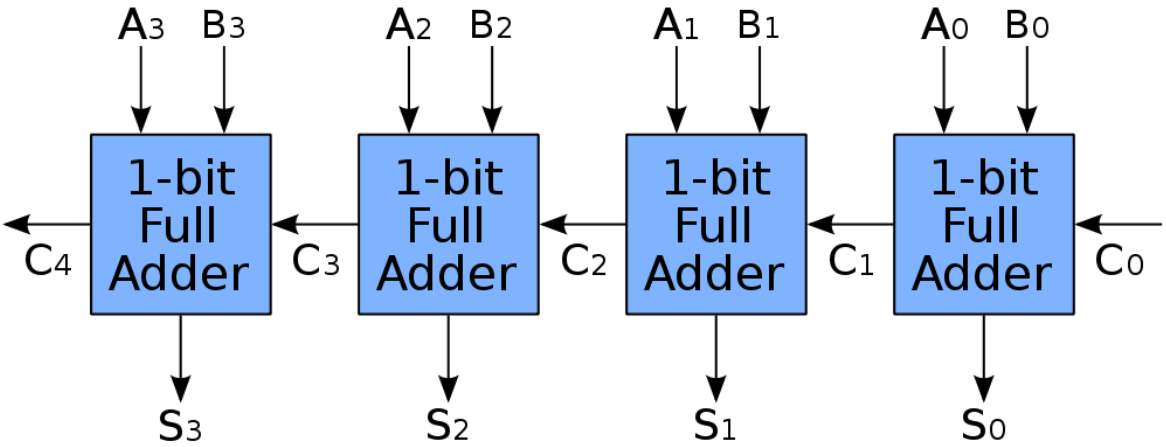

Рисунок 5.2 – 4-х бітний суматор

Даний суматор можна використати для реалізації помножувача, потрібно вихід суматора приєднати до його входу разом з множеним, зсунутим на і позицій вліво, на і ітерації потрібно додавати накоплену суму та оттимане число, або нулі, якщо і-тий біт множника має нульове значення. Через n ітерацій можна отримати шукане число.

Для реалізації інтегратора можна використати суматор з додаванням мультиплексора, як показано на рисунку 5.3.

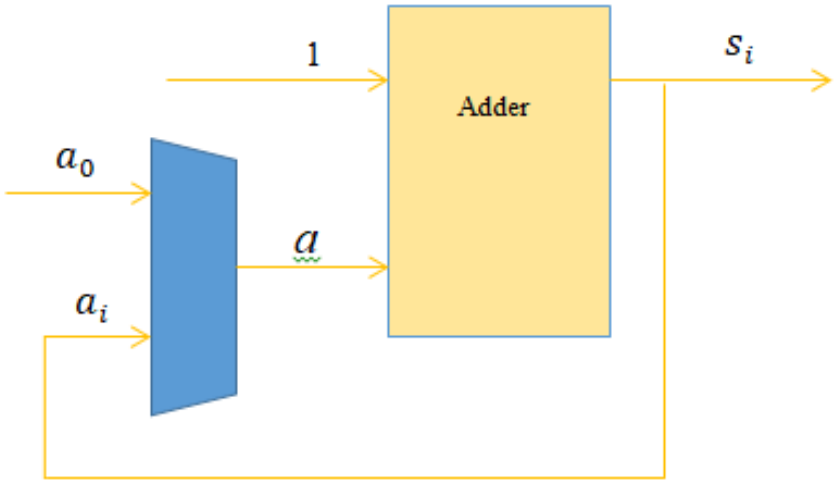

Рисунок 5.3 – Реалізація інтегратора

На вході суматора подається одиниця і вихід суматора на і-тій ітерації. На початковій ітерації на вхід подається початкове значення, яке буде інтегратуватися до певного значення. На і-тій ітерації на виході суматора маємо і-те значення інтегрування.

5.2 Опис програмної реалізації розробленого продукту

Програмну реалізацію системи круїз контролю можна розбити на пункти:

- реалізація суматора;

- реалізація помножувача;
- інтеграція попередніх пунктів;
- реалізація станів системи згідно графу на рисунку 5.1.

Для реалізації суматора потрібно реалізувати повний суматор та циклічно проходити по бітам числа, враховуючи біт переносу.

Реалізація повного суматора приведена в лістингу 5.1. На вхід подаються біти, які потрібно додати та біт переносу з попереднього повного суматору. Перший повний суматор на вході переносу отримує нуль. На виході отримуємо суму бітів та біт переносу.

Лістинг 5.1 – Реалізація повного суматора (файл Cruise_control.vhd)

```
entity fullAdder is
  port (c1, a1, a2 : in  bit;
        c2, s2     : out bit);
end fullAdder;
architecture fullAdder_arch of fullAdder is
begin
  s2 <= ((not c1) and (not a1) and a2) or
        ((not c1) and a1 and (not a2)) or
        (c1 and (not a1)and (not a2)) or
        (a1 and a2 and c1);
  c2 <= (a1 and c1) or (a2 and c1) or (a1 and a2);
end fullAdder_arch;
```

Для реалізації суматора для n розрядного числа потрібно поєднати n повних суматорів, як показано на рисунку 5.2. Реалізація суматора приведена в лістингу 5.2.

Лістинг 5.2 – Реалізація суматора (файл Cruise_control.vhd)

```
entity adder is
  generic (N : natural := 16);
  port (a, b : in  bit_vector (0 to N-1);
        s    : out bit_vector (0 to N-1);
        c    : out bit);
end adder;
architecture adder_arch of adder is
  component fullAdder
```

```
    port(c1, a1, a2 : in  bit;
         c2, s2     : out bit);
  end component;
  signal c_in : bit_vector (0 to N);
begin
  c_in(0) <= '0';
  adder : for i in 0 to N-1
  generate
    all_bit : fullAdder port map (c1 => c_in(i), a1 => a(N-1-i),
a2 => b(N-1-i),c2  => c_in(i+1), s2 => s(N-1-i));
  end generate adder;
  c <= c_in(N);
end adder_arch;
```

Суматор працює для n-розрядного числа. Код проходить по усім бітам, передає їх значення на вхід повного суматора, отримує суму та біт переносу. Після проходу по усім бітам на виході отримуємо вихідне значення, що є сумою двох вхідних чисел. При тестуванні додаємо два 16-розрядних числа: 45 та 36, в бітовому вигляді "0000000001000101" та "0000000000110110" відповідно. На виході маємо число 7B, "0000000001111011" відповідно, що підтверджує правильну роботу системи, рисунок 5.4.

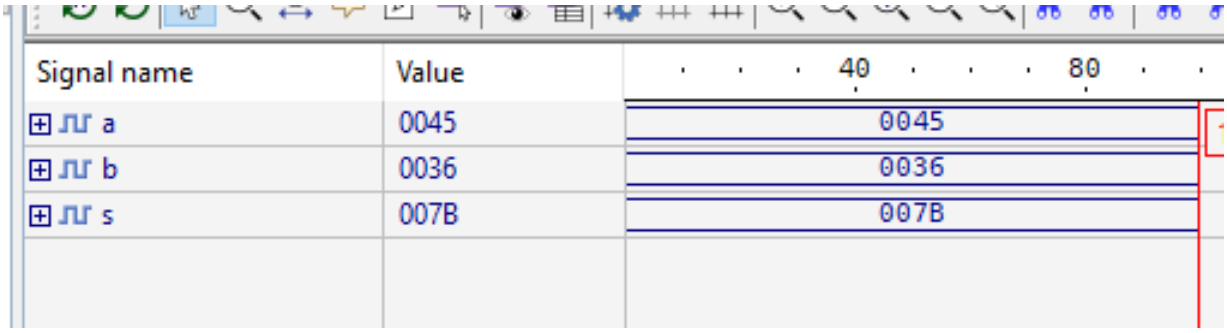

Рисунок 5.4 – Тестування 16-розрядного суматора

Як сказано раніше отриманий суматор можна застосувати для реалізації помножувача. В лістингу програмний код циклічно проходить по бітам множеного, якщо значення біта одиниця, то до змінної додається множене зі зміщенням на і позицій вліво.

Лістинг 5.3 – Реалізація посножувача (файл Cruise_control.vhd)

```
entity multiplier is
  generic (N : natural := 16);
  port (a, b : in  bit_vector (0 to N-1);
          s    : out bit_vector (0 to N-1) );
end multiplier;
architecture multiplier_arch of multiplier is
  component adder
    port(a1, a2 : bit_vector (0 to N-1);
         s2     : bit_vector (0 to N-1));
  end component;
begin
variable a_shift, sum : bit_vector (0 to N-1);
variable needAdd : bit;
     a_shift := to_stdlogicvector(a);
     sum := "0000000000000000";
     for i in 0 to N-1 loop
     a_shift := to_stdlogicvector(a sll i);
          needAdd := b(i);
          add: if needAdd = '1' generate
          all_bit : adder port map (sum, a_shift, s);
          end generate add;
          end if;
     end loop;
     s <= sum;
end multiplier_arch;
```

Ітогове значення акумулюється в змінній, після останньої ітерації знайдене число виводиться на вихід елементу. В тестовій програмі проводиться множення 16-розрядних чисел 45 та 36, на виході отримуємо число E8E.

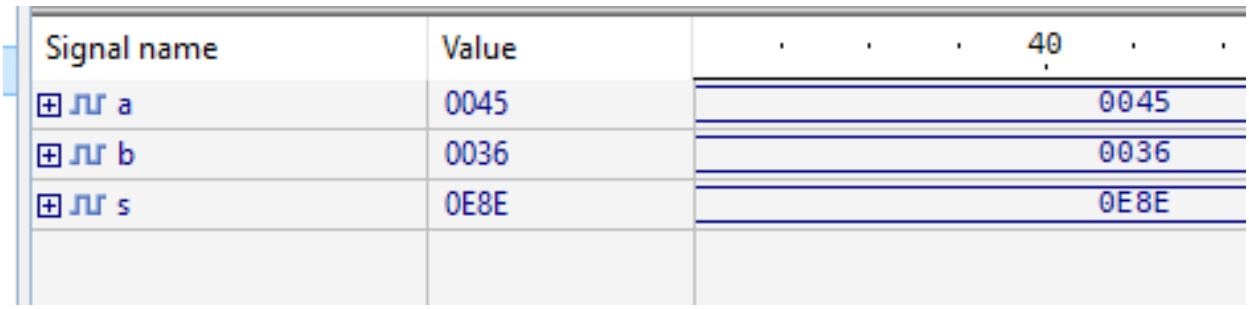

Рисунок 5.5 – Тестування 16-розрядного помножувача

Використовуючи розроблемний суматор можна реалізувати інтегратор. Елемент повільно збільшує втхідне значення від початкового до бажаного.

Лістинг 5.4 – Реалізація інтегратора (файл Cruise_control.vhd)

```
entity integrator is
  generic (N : natural := 16);
  port (a, b : in  bit_vector (0 to N-1);
           s    : out bit_vector (0 to N-1)
           );
end integrator;
architecture integrator_arch of integrator is
  component adder
    port(a1, a2 : bit_vector (0 to N-1);
         s2     : bit_vector (0 to N-1));
  end component;
  signal iteration : bit := '1'
begin
process is
variable sum : bit_vector (0 to N-1);
begin
     sum := a;
     adder : for i in 0 to N-1
     generate
    all_bit : fullAdder port map (sum => a, a2 => iteration, s2
=> s);
     end generate adder;
     s <= sum;
     wait on a;
end process;
```

Як видно на рисунку 5.6 інтегратор інтеграційно збільшує вихідне значення від 50 до 60.

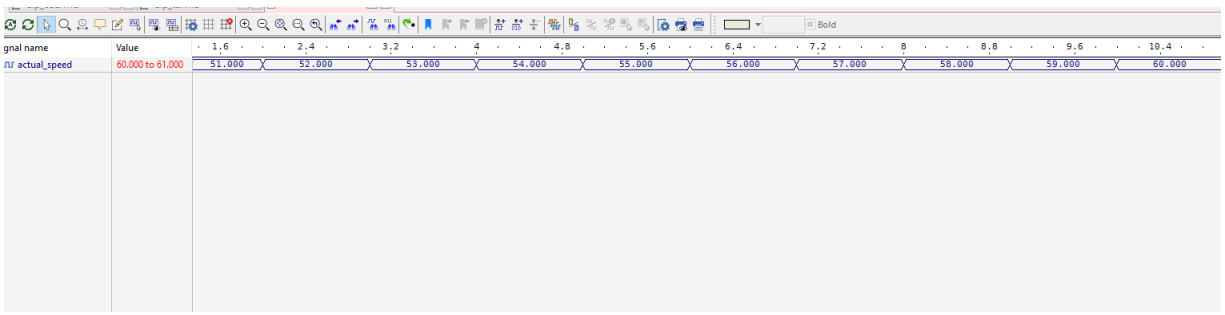

Рисунок 5.6 – Тестування інтегратора

Графік ітерації приведено на рисунку 5.7.

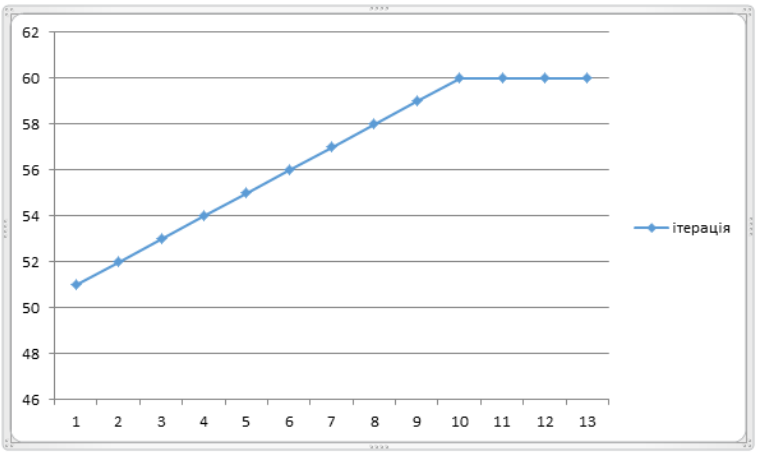

Рисунок 5.7 – Графік інтеграції

Інтегратор дозволяє вихідне значення системи привести до інтеграційного виду. Приклад приведено на рисунку 5.8. Це дозволяє змінювати швидкість більш плавно, система буде більш комфортною для водія і пересування буде більш плавним.

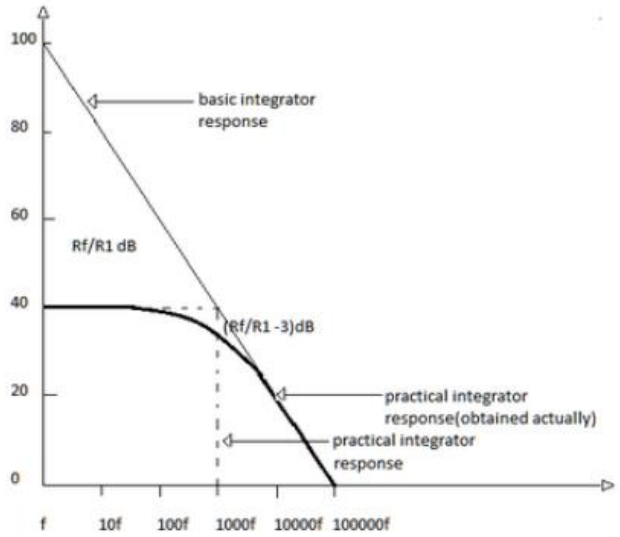

Рисунок 5.8 – Робота інтегратора

Поєднуючи раніше реалізовані елементи, можна реалізувати рівняння 4.10. В лістингу 5.5 показано реалізацію рівняння, що забеспечує роботу круїз контролю.

Лістинг 5.5 – Реалізація системи круїз-контролю (файл Cruise_control.vhd)

```
entity cruise_control is
  generic (N : natural := 16);
  port (v, vr : in  bit_vector (0 to N-1);
          u    : out bit_vector (0 to N-1)
          );
end cruise_control;
architecture cruise_control_arch of cruise_control is
  component multiplier
    port(v, vr : bit_vector (0 to N-1);
         u     : bit_vector (0 to N-1));
  end component;
begin
process is
variable temp_u : bit_vector (0 to N-1);
begin
     temp_u := "0000111010001110";
     add: generate
     all_bit : adder port map (v, vr, temp_u);
     end generate add;
     add: generate
     all_bit : multiplier port map (temp_u, k);
     end generate add;
all_bit : integrator port map (temp_u, k);
     end generate add;

     u <= temp_u;
     wait on v, vr;
end process;
end cruise_control_arch
```

При тестуванні розробленої системи, подамо на вхід поточну та бажану швидкість. Поточна швидкість 58 кілометрів за годину, бажана 60 кілометрів за годину. Як показано на рисунку 5.5 система збільшує сигнал на дросельну заслонку, що призводить до збільшення швидкості транспорту. Після досягнення бажаної швидкості система підтримує дану швидкість. Як показано на рисунку точність системи є +/- 0.5 кілометрів за годину.

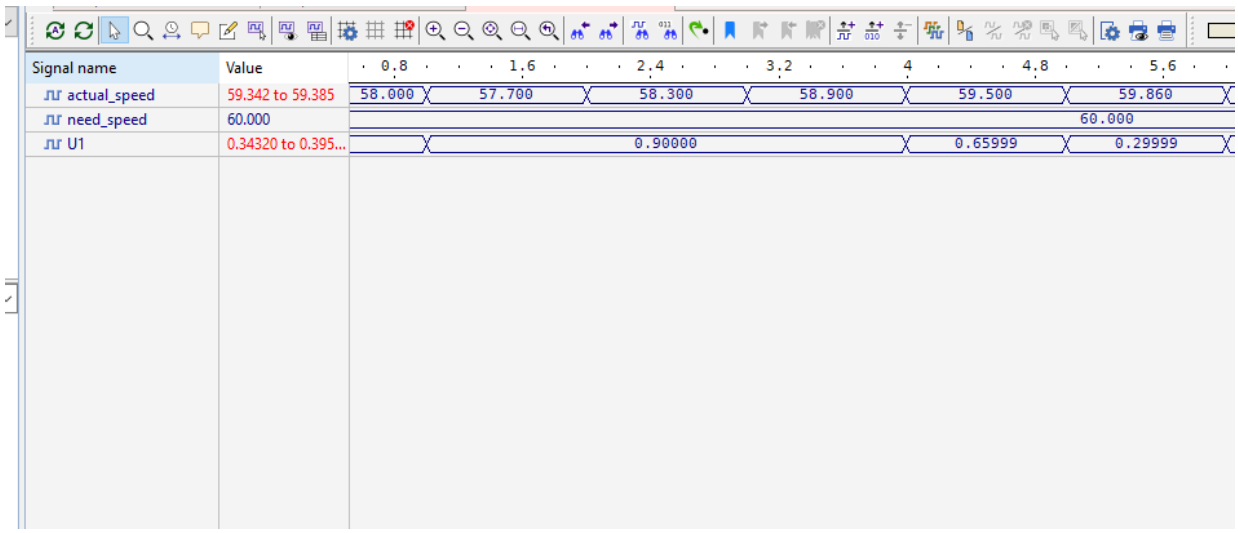

Рисунок 5.9 – Робота системи круїз-контролю

Збільшення швидкості транспорту можна зобразити графіком на рисунку 5.10. Поточна швидкість поступово збільшується, поки не стане такою як бажана.

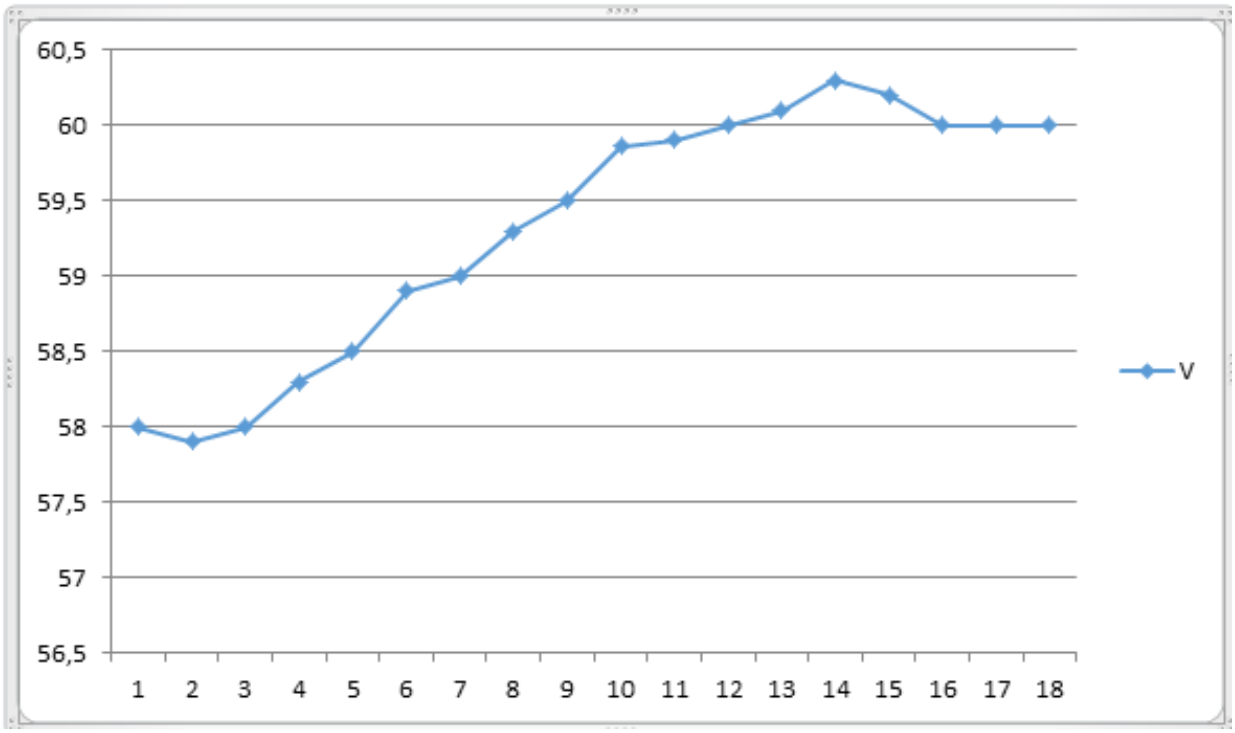

Рисунок 5.10 – Графік зміни швидкості

Для системи залишилось додати підтримку різних станів. Для системи є 4 стани:

- вимкнено;
- чекання;
- круїз-контроль;
- підтримка швидкості.

В початковому стані система вимкнена, після ввімкнення система очикує на задавання бажаної швидкості, після цього система регулює швидкість, коли досягнено бажаної швидкості система підтримує її. Для реалізації станів потрібно додати перевірку стану, як показано в лістингу 5.6.

Лістинг 5.6 – Перевірка поточного стану системи (файл Cruise_control.vhd)

```
IF input = '1' or '2' THEN
  u <= u;
ELSIF input = '3' THEN
  --робота системи круїз-контролю
Else
  u <= u;
end if;
```

В початковому стані система не впливає на швидкість транспорту. В стані очикування система очікує на встановлення бажаної швидкості і також не впливає на сигнал на дросельну заслінку. В стані круїз-контроль система змінює поточну швидкість, наближуючи її до бажаної. Коли бажану швидкість досагнуто, система підтримує поточну швидкість, очікуючи на її зміну.

5.3 Тестування розробленої системи

Розроблена система повинна працювати в різних сценаріях. Перший сценарій – це коли система вимкнена. Як показано на рисунку 5.11 вихідний сигнал на дросельну заслінку не змінюється, поточна та бажана швидкісті не змінюються, бо в цьому сценарії автомобіль рухається по рівній дорозі.

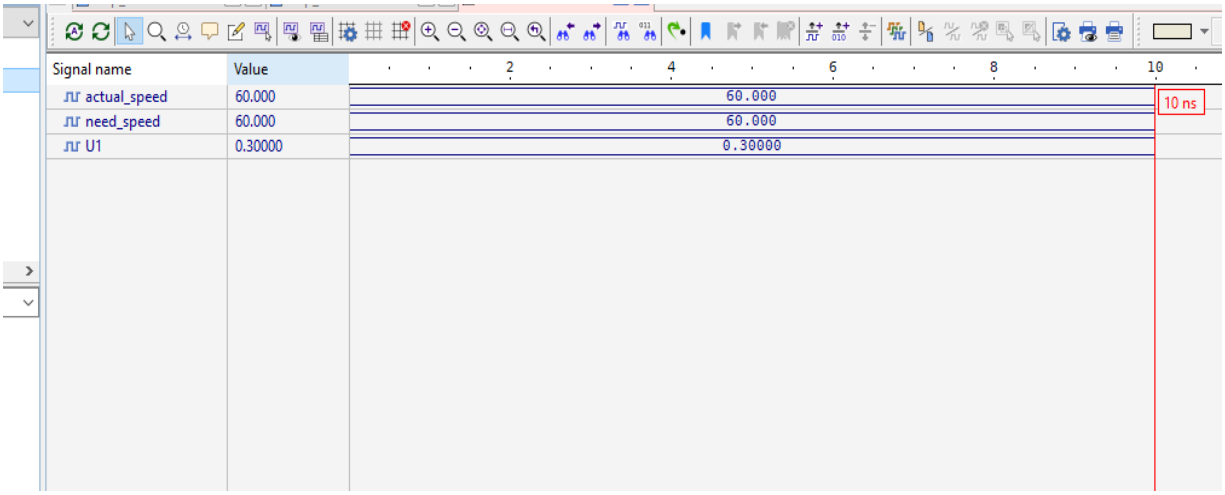

Рисунок 5.11 – Робота системи в вимкненому стані

Наступний сценарій – реакція системи в вимкненому стані при пересуванні по дорозі під певним кутом. На рисунку 5.12 видно, що сигнал на дросельну заслінку не зміінюється, але збільшився кут нахилу дороги, тому поточна швидкість поступово зменшується.

Рисунок 5.12 – Реакція системи в вимкненому стані на похилу дорогу

Далі треба перевірити реакцію системи на ці умови в вімкненому стані. Виставимо поточну швидкість як 59 кілометрів за годину а бажану 61

кілометр за годину. На рисунку 5.13 продемонстровано роботу системи на рівній дорозі, поточна швидкість наближується до бажаної з точністю 0.5 кілометрів за годину.

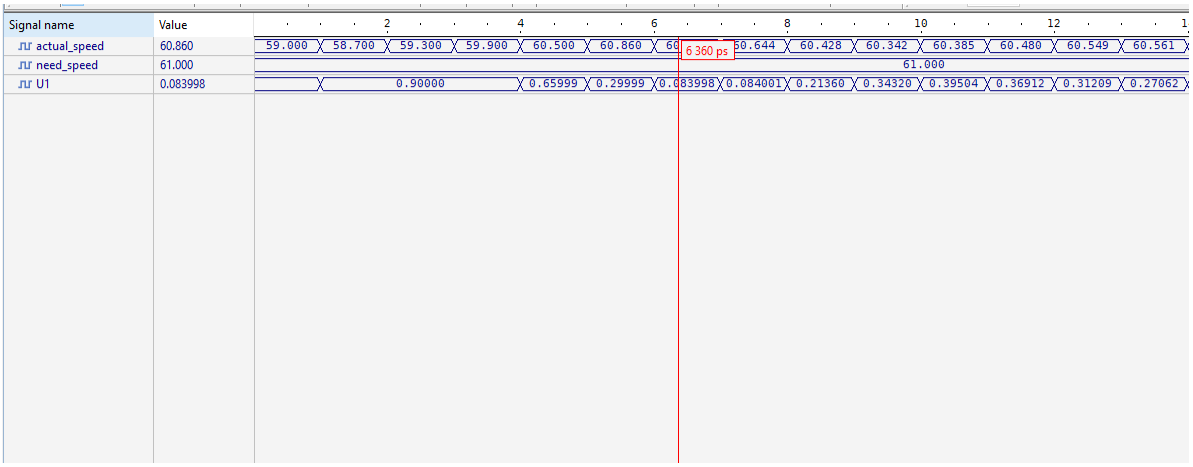

Рисунок 5.13 – Робота системи на рівній дорозі

Коли транспорт їде по похилій дорозі, сила тяжіння починає впливати на поточну швидкість транспорту. Як видно з рисунку 5.14 при зміні кута нахилу дороги поточна швидкість починає зменшуватись, система круїз-контролю реагує на це збільшенням сигналу, що подається на дросельну заслінку, що призводить до повернення поточної швидкості до початкового стану.

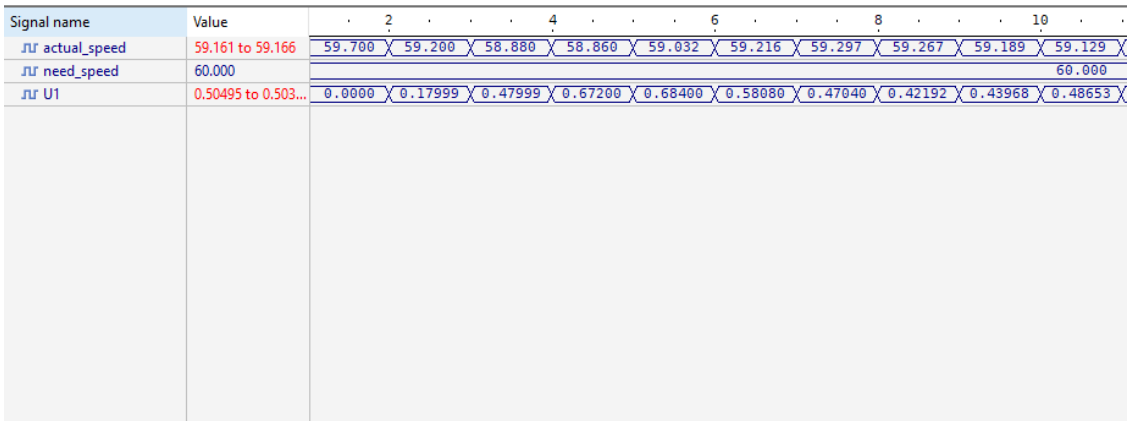

Рисунок 5.14 – Реакція системи на збільшення кута нахилу дороги

При тестуванні стану очікування система при досяганні бажаної швидкості підтримує задану швидкість. Як видно з рисунку 5.15 система після досягнення бажаної швидкості переходить в стан підтримки цієї швидкості.

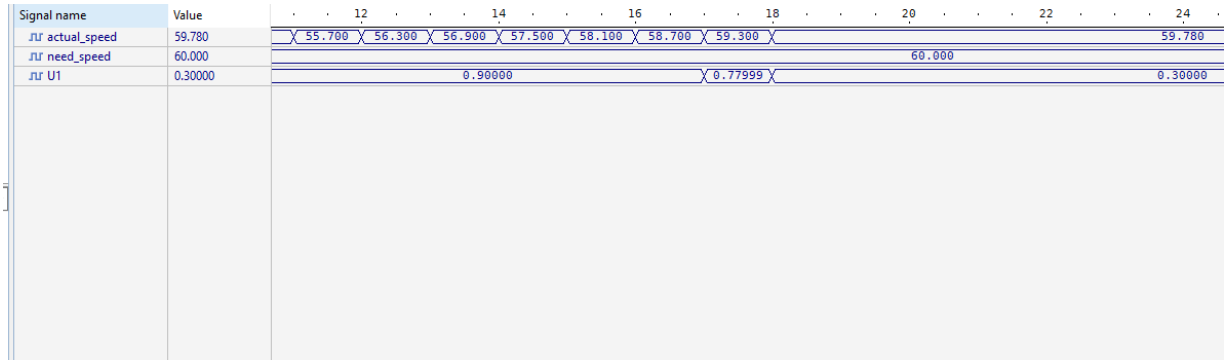

Рисунок 5.15 – Реакція системи на збільшення кута нахилу дороги

Треба перевірити реакцію системи, коли бажана швидкість менша ніж поточна. Як бачимо з рисунку 5.16 система досягає бажану швидкість з точністю 0.5 кілометрів за годину.

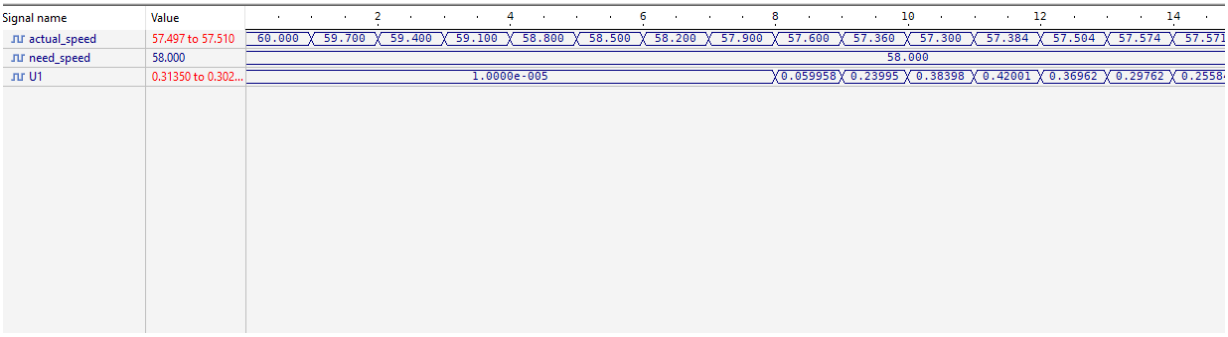

Рисунок 5.16 – Робота системи при зменшенні бажаної швидкості

Коли транспорт рухається з похилої площини, його поточна швидкість зюільшується. Як видно з рисунку 5.17 швидкість починає збільшуватись, але система реагує на це і відповідно зменшує сигнал на дросельну заслінку.

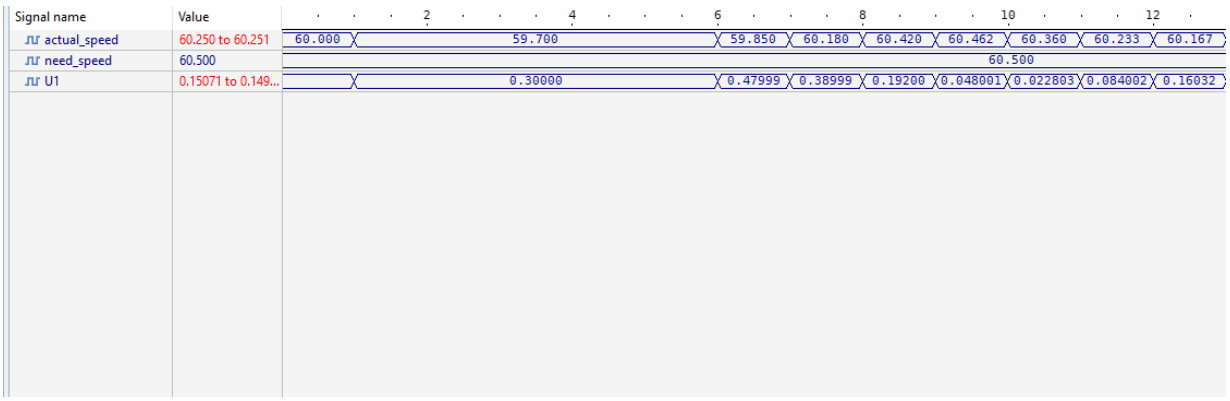

Рисунок 5.17 – Робота системи при русі з пагорба

Треба перевірити, як система відреагує на вимкнення під час роботи. Як видно з рисунку 5.18 система починає регулювати швидкість, через 5 наносекунд систему вимикають, через це автомобіль втрачає швидкість.

Рисунок 5.18 – Реакція системи на вимкнення під час роботи

Під час тестування було протестовано систему в різних ситуаціях, з отриманих результатів можна зробити вивід, що система адекватно реагує на зміну швидкості автомобіля і підтримує бажану швидкість за різних зовнішніх умов.

ВИСНОВКИ

В ході виконання кваліфікаційної роботи було проведено аналіз поставленої задачі, розглянуто сучасні приклади використання програмованих логічних матриць та систем на чіпі для реалізації систем допомоги водієві.

Було розглянуто принцип побудови сучасних систем круїз-контролю, в тому числі з використанням FPGA. В ході роботи було розглянуто технології побудови сучасних FPGA, системи круїз-контролю з використанням систем на чіпі.

Було розроблено математичну модель руху транспортного засобу, де було описано зовнішні сили, які впливають на швидкість його руху. В процесі аналізу математичної моделі було виведено функцію для роботи системи круїз-контролю

Було описано програмну реалізацію системи круїз-контролю, з використанням технології реконфігувації на базі SoC сімейства Virtex7.

При перевірці розробленої системи було проведено ряд експериментів в різних умовах. Проаналізовані результати показали, що система коректно підтримує встановлену швидкість транспорту як при русі на пагорб, так і з нього.

Існує декілька шляхів розвитку розробленої системи, наприклад можливо розвити звичайну систему круїз-контролю до системи прогностичного круїз-контролю, в якій можливо використати технологію GPS та топологічні мапи, щоб заздалегіть передбачати зовнішні умови, які будуть впливати на швидкість транспортного засобу.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Куліш Д. В. Розробка системи круїз-контролю з використанням телекомунікаційних мереж/ Д.В. Куліш, В.О. Горбачов // Проблеми інформатизації – 2022. – Т. 10. № 2.

2. Рачков С. А. Фактори розвитку світової автомобільної промисловості / С. А. Рачков // Вісник євразійської науки. - 2022. - Т. 14. – № 1

3. Automotive control systems / A. Galip Ulsoy, Huei Peng, Melih Cakmakci // Cambridge University, New York, USA 2012. – 406 с.

4. Peter Slowik Automation in the long haul: Challenges and opportunities of autonomous heavyduty trucking in the United States / Peter Slowik, Ben Sharpe // International council on clean transportation. - 2018 - 30 с.

5. FPGA Implementation of Embedded Cruise Control and Anti-Collision Radar [Електронний ресурс] / Sébastien Le Beux, Philippe Marquet, Ouassila Labbani, Jean-Luc Dekeyser // HAL, Dubrovnik, Croatia 2006. – 9 с. – Режим доступу до ресурсу: https://hal.inria.fr/inria-00079057

6. Девід М. Харріс Цифрова схемотехніка та архітектура комп'ютера / Девід М. Харріс, Сара Л. Харріс; - М. :Вид-во. Morgan Kaufman, 2013.–1684 с.

7. Тарасов І. Нові можливості САПР Xilinx / Ілья Тарасов // Компоненти і технології. – Т.12. – 2010. – с. 34-36.

8. 7 Series FPGAs Configuration User Guide [Електронний ресурс] // Xilinx. – 2022. – с. 176. – Режим доступу до ресурсу: https://docs.xilinx.com/v/u/en-US/ug470_7Series_Config

9. A.L.Suseela Embedded Systems in Real Time Applications, Design & Architecture [Електронний ресурс] / A.L.Suseela, V. L. Kumar// Ubiquity. – 2005. - Т. 6, № 28. – Режим доступу до ресурсу: http://www.acm.org/ubiquity

10. Prashik Shende Radar Based Adaptive Cruise Control / Prashik Shende, Vignesh Kumar // International Research Journal of Engineering and Technology. – 2020. – Т. 7. – Issue 6.

11. Eichner M. L. Integrated Speed Limit Detection and Recognition from Real-Time Video [Електронний ресурс] / M. L Eichner, T.P. Breckon // IEEE Intelligent Vehicles Symposium. – 2019. – с. 626–631 doi:10.1109/IVS.2008. 4621285. ISBN 978-1-4244-2568-6. S2CID 12477544.

12. Sven Vlassenroot Driving with intelligent speed adaptation / Sven Vlassenroot, Steven Broekx, Johan De Mol // Transportation Research. – 2007. – с.267-279.

13. Karl Johan Aström Feedback Systems : an introduction for scientists and engineers / Karl Johan Aström, Richard M. Murray // Princeton University, Princeton, New Jersey. – 408 с.

14. Клименко І. А. Класифікація реконфігурованих обчислювальних систем [Електронний ресурс] / І. А. Клименко, М. В. Рудницький // Вісник Вінницького політехнічного інституту. – 2014. – Т. 5. – с 120-128. – Режим доступу: http://nbuv.gov.ua/UJRN/vvpi_2014_5_21.

15. R. Joost Advantages of FPGA-based multiprocessor systems in industrial applications / R. Joost, R. Salomon // 31st Annual Conference of IEEE Industrial Electronics Society. – 2005 – Т. 1. – с 6. – DOI: https://doi.org/ 10.1109/IECON.2005.1568946

16. Partial Reconfiguration in the ISE Design Suite [Електронний ресурс]. — Xilinx, 2014. – Режим доступу: http://www.xilinx.com/tools/partial-reconfiguration.htm.

17. M Morris Mano Digital Logic And Computer Design / M Morris Mano // Pearson College. – 1979. – Т1. – 612 с.

18. Paul Horowitz The Art of Electronics / Paul Horowitz, Winfield Hill // Cambridge University Press. – 2015. – Т3. – 525 с.

19. Peter J. Ashenden The Designer's guide to VHDL [Електронний ресурс] / Peter J. Ashenden // Elsevier. – 2008. – Т3. – 910 с. – DOI: https://doi.org/10.1016/B978-012088785-9.00031-9.

20. Документація Active-HDL [Електронний ресурс] – Режим доступу до ресурсу: https://www.aldec.com/en/support/resources/documentation

21. Miro Samek Practical Statecharts in C/C++ / Miro Samek // CMP Books, San Francisco, USA. – 2002. – 405 c.

22. Burgess Fast Ripple-Carry Adders in Standard-Cell CMOS VLSI [Електронний ресурс] / Burgess, Neil // 20th IEEE Symposium on Computer Arithmetic. – 2011. – с. 103–111. Режим доступу: https://ieeexplore. ieee.org/document/5992115

23. Vivado Design Suite User Guide [Електронний ресурс]. — Xilinx, 2022. – Режим доступу: https://docs.xilinx.com/v/u/2020.1-English/ug909-vivado-partial-reconfiguration

24. Partial Reconfiguration User Guide [Електронний ресурс]. — Xilinx, 2013. – Режим доступу: https://www.xilinx.com/content/dam/xilinx/ support/documents/sw_manuals/xilinx14_5/ug702.pdf