

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук та технологій
(повне найменування факультету)

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

Пояснювальна записка

до дипломного проєкту (роботи)

бакалавра

(ступінь вищої освіти (освітній ступінь))

на тему **РОЗРОБКА ПРИСТРОЮ КЕРУВАННЯ ДЛЯ**
ФІЛЬТРАЦІЇ МЕРЕЖЕВИХ ПАКЕТІВ ПО МАСЦІ

Виконав: студент 4 курсу, групи КНТ-510
спеціальності _____

123 «Комп'ютерна інженерія»

(код і найменування спеціальності)

Освітня програма (спеціалізація) _____

«Комп'ютерна інженерія»

ПІМШИН Б. О.

(ПРИЗВИЩЕ та ініціали)

Керівник ГОЛУБ Т.В.

(ПРИЗВИЩЕ та ініціали)

Рецензент ЄФИМЕНКО М.В.

(ПРИЗВИЩЕ та ініціали)

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.

“ ” 2024 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА

ПІМШИНА Богдана Олексійовича

(ПРІЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Розробка пристрою керування для фільтрації мережевих пакетів по масці

керівник проєкту (роботи) к. т. н., доцент, ГОЛУБ Тетяна Василівна

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

затверджені наказом вищого навчального закладу від “1” квітня 2024 року № 105

2. Строк подання студентом проєкту (роботи) 27 травня 2024 року

3. Вихідні дані до проєкту (роботи) Технічне завдання на розробку пристрою керування для фільтрації мережевих пакетів по масці із застосуванням ПЛІС та програмного забезпечення Quartus II

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
Аналіз технічного завдання. Вибір обладнання та пристроїв моделювання.
Розробка та тестування електричної принципової схеми.

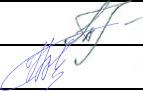
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

Схема електрична принципова

Структурна схема простого фільтра пакетів

Структурна схема пристрою сортування пакетів

6. Консультанти розділів проєкту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-3	ГОЛУБ Т.В., к.т.н., доцент		
нормоконтроль	ЩЕРБАК Н.В., ст. викл.		

7. Дата видачі завдання « 23 » лютого 2024 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Аналіз технічного завдання	05.03.2024 р.	
2	Вибір обладнання та технологій	05.04.2024 р.	
3	Моделювання та налаштування схем	12.04.2024 р.	
4	Тестування пристрою фільтрації та сортування	26.04.2024 р.	
5	Оформлення пояснювальної записки	08.05.2024 р.	
6	Оформлення графічного матеріалу	12.05.2024 р.	
7	Оформлення плакатів	20.05.2024 р.	

Студент (ка)


(підпис)

Богдан ПІМШИН
(Ім'я ПРИЗВИЩЕ)

Керівник проєкту (роботи)


(підпис)

Тетяна ГОЛУБ
(Ім'я ПРИЗВИЩЕ)

РЕФЕРАТ

ПЗ: 64 с., 1 табл., 20 рис., 3 ліст., 11 джерел.

БІТ, БРАНДМАУЕР, МАСКА, ОКТЕТ, ПІДМЕРЕЖА, ПРИСТРІЙ
КЕРУВАННЯ, ПЛІС, СХЕМА, ТЕСТУВАННЯ, ФІЛЬТРАЦІЯ, ХОСТ, AND, ASIC,
CIDR, ETHERNET, FPGA, HEADER, INTEL, IP, OR, PACKET, QUARTUS II,
SRAM, VHDL, XILINX

Мета роботи – проєктування пристрою керування для фільтрації мережевих пакетів по масці. Об’єкт розробки – процес проєктування пристрою для фільтрації мережевих пакетів. Предмет розробки – методи та засоби проєктування пристрою керування.

Перший розділ присвячено опису основних теоретичних положень системи, дослідженню та аналізу аналогічних сучасних систем, дослідженню апаратних та програмних засобів для розробки систем, а саме систем для фільтрації мережевих пакетів. Визначені основні задачі та завдання цієї роботи.

У другому розділі визначено та проведено розробку системи в цілому та її компонентів, зокрема було розроблено компаратор, кодер та демультиплексор. Наведено відповідні лістинги цих програмних частин.

Третій розділ присвячено моделюванню й тестуванню системи за допомогою програми Quartus II. Після розробки структурної схеми пристрою для фільтрації мережевих пакетів по масці було проведено моделювання імплементації системи на кристал цієї схеми з фіксацією результатів.

ЗМІСТ

Скорочення та умовні позначки	6
Вступ.....	7
1 Основні теоретичні положення розробки систем	8
1.1 Аналогічні сучасні системи.....	13
1.2 Апаратні засоби для розробки системи	17
1.3 Програмні засоби для розробки системи.....	25
1.4 Постановка завдання.....	29
2 Розробка структурних частин системи	31
2.1 Структура мережевого пакета повідомлення.....	31
2.2 Структура пристрою фільтрації.....	36
2.3 Розробка компаратора.....	39
2.4 Розробка кодера.....	41
2.5 Розробка демультіплексора.....	43
3 Розробка апаратної схеми системи.....	45
3.1 Розробка структурної схеми пристрою.....	45
3.2 Тестування схеми пристрою	47
Висновки	58
Перелік джерел посилання	59

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

- ПЛІС - програмована логічна інтегральна схема
- ASIC - інтегральна схема, спеціалізована для вирішення конкретного завдання
- CIDR - безкласова міждоменна маршрутизація, Classless Inter-Domain Routing
- IP - Інтернет протокол, Internet Protocol
- VHDL - мова опису апаратури інтегральних схем, Very high speed integrated circuits Hardware Description Language)

ВСТУП

У сучасну цифрову епоху важливість мережевої безпеки та захисту даних неможливо переоцінити. Зі збільшенням обсягів мережевого трафіку та зростанням кількості кіберзагроз виникає нагальна потреба в надійних пристроях контролю, здатних ефективно фільтрувати мережеві пакети на основі певних критеріїв.

Поточний стан справ у сфері фільтрації мережевих пакетів характеризується використанням програмних рішень, які часто є ресурсоємними і трудомісткими. Ці методи не завжди є надійними і можуть не забезпечувати необхідний рівень захисту для збереження конфіденційної інформації.

Для вирішення цієї проблеми пропонується використання спеціалізованого контрольного пристрою, розробці якого присвячено дану роботу. Для цього проведено детальний огляд та порівняння апаратних та програмних засобів для найбільш ефективного використання ресурсів та обрано оптимальні варіанти. Завдяки програмному забезпеченню Quartus II та мові VHDL стане можливим створення апаратного рішення, що забезпечить більш ефективний та надійний спосіб фільтрації мережевих пакетів.

Актуальність цієї розробки полягає в її потенціалі для підвищення мережевої безпеки та захисту даних. Реалізувавши апаратний пристрій керування, буде значно скорочено час обробки та звільнено ресурси, необхідні для фільтрації мережевих пакетів. Крім того, розробка цього пристрою керування може мати значний вплив на практичні аспекти мережевої безпеки. Результати, отримані в рамках цього проєкту, можуть сприяти розвитку технології апаратної мережевої фільтрації, а також забезпечити практичне рішення для поліпшення захисту даних в реальних додатках.

Загалом, розробка пристрою керування для фільтрації мережевих пакетів за маскою з використанням програмного забезпечення Quartus II та мови VHDL має великі перспективи для посилення мережевої безпеки, розширення технологічних можливостей та підвищення загальної ефективності заходів захисту даних.

1 ОСНОВНІ ТЕОРЕТИЧНІ ПОЛОЖЕННЯ РОЗРОБКИ СИСТЕМ

Фільтрація пакетів - це процес пропускання або блокування пакетів даних на мережевому інтерфейсі брандмауером на основі адрес джерела та призначення, портів або протоколів. Пакетний фільтр перевіряє заголовок кожного IP-пакета на основі набору правил і вирішує, чи не пропускати його (відкинути), чи дозволити (прийняти).

Більшість комп'ютерних мереж, включно з інтернетом, використовують пакети для передачі даних. Кожен пакет - це самодостатній пакет даних, який може маршрутизуватися мережею, і кожен пакет складається з двох частин. Заголовок пакета містить інформацію про IP-адресу джерела, IP-адресу призначення, протокол і порт. Корисне навантаження пакета містить дані, які потрібно передати. Фільтрація пакетів - це метод прийняття рішення про те, дозволити або відкинути кожен пакет, що проходить через брандмауер, на основі інформації в заголовку.

Брандмауери з фільтрацією пакетів часто розгортаються на кордонах мережі. Наприклад, вони можуть бути встановлені між корпоративною мережею та Інтернетом або між мережею серверів і клієнтами. Він також може існувати як програмне забезпечення на комп'ютері, що захищає його мережеве з'єднання.

Фільтрація пакетів є найпоширенішим типом брандмауера. Майже всі продукти брандмауерів підтримують фільтрацію пакетів.

Брандмауери, що фільтрують пакети, використовують правила, щоб вирішити, чи дозволити пакет даних, чи відкинути його. Ці правила встановлюються адміністратором брандмауера.

Фільтрація пакетів ґрунтується на інформації заголовка IP-пакета та інформації про пристрій брандмауера, наприклад, на наступному: IP-адреса джерела, IP-адреса призначення, порт-джерело, порт призначення, мережевий протокол, IP-прапори, інтерфейс брандмауера, напрямок, вхід або вихід.

Багато правил часто використовуються в поєднанні один з одним з певним пріоритетом для створення загальної політики.

Існує кілька різних типів брандмауерів з фільтрацією пакетів.

У статичному брандмауері всі правила встановлюються адміністратором. Правила не змінюються автоматично в залежності від умов, що змінюються.

Динамічний брандмауер може активувати або деактивувати правила залежно від умов, що змінюються. Ці правила можуть ґрунтуватися на таких факторах, як час або трафік. Наприклад, динамічний брандмауер може автоматично додавати погані IP-адреси до списку виключення під час розподіленої атаки на відмову в обслуговуванні.

Брандмауер без стану не зберігає жодної інформації про з'єднання в часі. Вони можуть приймати рішення лише на основі попередньо визначених правил та інформації, що міститься в IP-пакеті.

Брандмауер зі збереженням стану може зберігати інформацію в часі та зберігати список активних з'єднань. Це дозволяє брандмауеру приймати більш обґрунтовані рішення. Наприклад, якщо внутрішній клієнт надіслав запит на зовнішній сервер, брандмауер з підтримкою стану реєструє це в журналі, а потім, коли з відомого сервера надходить пакет-відповідь, брандмауер вважає його частиною чинного з'єднання і дозволяє його. З іншого боку, якщо зовнішній сервер надіслав вхідну відповідь без відповідного вихідного запиту, брандмауер відкидає його як зловмисний.

Фільтрація пакетів як система захисту мережі має свої переваги та недоліки.

Переваги: простота у використанні, економічно ефективна, швидкодія, не використовує багато обчислювальних ресурсів, детермінована, передбачувана поведінка.

Недоліки: не ефективний проти сучасних атак, можна обійти за допомогою підміни, не може приймати рішення на основі програми або автентифікації, списки правил з часом можуть стати великими й складними в управлінні [1].

В процесі фільтрації пакетів рішення про передавання даних приймається на основі маски підмережі.

За своєю суттю, маска підмережі є фундаментальним компонентом IP-адресації. Вона діє як механізм фільтрації, розділяючи IP-адресу на дві окремі

частини: мережеву адресу та адресу хоста. Визначивши маску мережі, можна визначити, яка частина IP-адреси представляє мережу, а яка ідентифікує пристрої в мережі.

Маска мережі складається з 32 бітів, зазвичай представлених у десятковій системі числення, і тісно пов'язана з IP-адресами.

Маска підмережі - більш поширений термін, який використовується для опису масок мереж в контексті поділу мереж на підмережі. Обидва терміни часто використовуються як взаємозамінні.

Щоб зрозуміти, як працює маска підмережі, дуже важливо розуміти двійкове представлення IP-адрес. У двійковому вигляді IP-адреса складається з 32 бітів, розділених на чотири октети (групи по вісім бітів у кожній). Маска мережі, також представлена у двійковому вигляді, складається з послідовних одиниць (1), за якими слідує послідовні нулі (0).

Виконуючи побітову операцію І між IP-адресою та маскою мережі, ефективно «маскуємо» частину хоста, залишаючи лише мережеву адресу. Це дозволяє пристроям у мережі визначати, чи знаходиться цільова IP-адреса в їхній власній мережі, чи її слід перенаправити в іншу мережу.

Обчислення маски мережі та маски підмережі може бути складним завданням, особливо для новачків у роботі з мережею. Однак за допомогою калькуляторів масок підмереж та IP-калькуляторів цей процес стає набагато простішим. Ці онлайн-інструменти дозволяють ввести вашу IP-адресу і бажану кількість підмереж або хостів, і вони автоматично генерують відповідну маску мережі або маску підмережі.

Маска мережі може бути виражена в різних нотаціях, таких як десятково-крапкова (наприклад, 255.255.255.0) або CIDR-нотація (наприклад, /24). Запис CIDR забезпечує стисле представлення маски мережі й широко використовується у сучасних мережах.

Існує 3 основні причини, чому рекомендується використовувати маску підмережі: ефективне використання адрес, підвищена безпека і спрощене керування мережею.

Ефективне використання адрес: маски підмережі відіграють важливу роль в ефективному використанні IP-адрес в мережі. Ретельно розробляючи підмережі та призначаючи відповідні маски підмереж, можна максимізувати розподіл адрес, гарантуючи, що кожна підмережа має достатньо адрес для розміщення своїх пристроїв, не витрачаючи при цьому цінний IP-простір даремно.

Визначаючи маску підмережі для підмережі, важливо дотримуватися балансу між кількістю доступних адрес хостів і кількістю необхідних підмереж. Вибір занадто обмежувальної маски може призвести до недостатньої кількості адрес хостів, тоді як надто широка маска може призвести до втрати цінних IP-адрес.

Підвищена безпека: використання мережевих масок для розділення мережі на підмережі забезпечує додатковий рівень безпеки. Розділяючи пристрої на основі їхніх мережевих адрес, можна впроваджувати більш деталізовані політики контролю доступу та ефективно керувати потоком мережевого трафіку. Це допомагає зменшити ризики безпеки та потенційний несанкціонований доступ.

Створюючи окремі підмережі для різних відділів, гостьових мереж або критично важливих об'єктів інфраструктури, можна застосувати більш суворі заходи безпеки і обмежити доступ між підмережами. Крім того, розгортання правил брандмауера і впровадження VLAN на основі мережевих масок може ще більше підвищити безпеку мережі.

Спрощене керування мережею: маски мереж спрощують управління мережею, дозволяючи логічно групувати пристрої на основі їх IP-адрес. Завдяки чітко визначеним підмережам і маскам мережі, усунення несправностей у мережі, впровадження змін і розширення мережі стають більш простими і організованими. Це призводить до підвищення операційної ефективності та зменшення часу простою.

Крім того, використання мережевих масок дозволяє ефективніше впроваджувати протоколи і політики маршрутизації. Рішення про маршрутизацію можна приймати на основі мережевої частини IP-адреси, забезпечуючи ефективне спрямування трафіку між підмережами [2].

Для фільтрації пакетів використовуються брандмауери різних типів, до яких відносяться апаратні та програмні рішення.

Апаратний брандмауер - це фізичний пристрій, розміщений між комп'ютером або мережею та його підключенням до Інтернету. Він працює незалежно від хост-пристрою, перевіряючи вхідний і вихідний трафік на відповідність встановленим правилам безпеки. Активно аналізуючи пакети даних, апаратний брандмауер може виявляти і блокувати загрози, забезпечуючи надійний бар'єр проти потенційних кібервторгнень.

Робота апаратного брандмауера передбачає його підключення безпосередньо між джерелом інтернету та цільовою мережею або системою. Після цього весь інтернет-трафік, як вхідний, так і вихідний, повинен проходити через цей пристрій. Перевіряючи кожен пакет даних, брандмауер приймає рішення на основі заздалегідь визначених політик безпеки. Шкідливий або підозрілий трафік блокується, тому до внутрішньої мережі потрапляють лише безпечні та легітимні дані. Загрози перехоплюються до того, як вони потрапляють у внутрішні системи, пропонуючи проактивний підхід до мережевої безпеки.

Програмний брандмауер - це брандмауер у формфакторі програмного забезпечення, а не фізичного пристрою, який може бути розгорнутий на серверах або віртуальних машинах для захисту хмарних середовищ.

Програмні брандмауери призначені для захисту конфіденційних даних, робочих навантажень і додатків у середовищах, де розгортання фізичних брандмауерів ускладнене або неможливе.

Програмні брандмауери використовують ту саму технологію захисту, що й апаратні брандмауери. Вони пропонують кілька варіантів розгортання, щоб відповідати потребам гібридних/мультихмарних середовищ і сучасних хмарних додатків. Програмні брандмауери можна розгорнути в будь-якій віртуалізованій мережі або хмарному середовищі.

Апаратний брандмауер - це окремий фізичний пристрій, розташований між мережею та підключеними до неї пристроями. Розгортання апаратного

брандмауера вимагає кваліфікованого персоналу для забезпечення належного налаштування та постійного управління.

З іншого боку, програмний брандмауер працює на сервері або віртуальній машині. Цей тип брандмауера працює на операційній системі, орієнтованій на безпеку, як правило, надбудованій над загальними апаратними ресурсами. Його часто можна швидко впровадити за допомогою хмарних засобів автоматизації.

Як апаратні, так і програмні брандмауери забезпечують необхідний захист мережевої безпеки, а їх вибір визначається конкретними вимогами та контекстом розгортання [3].

1.1 Аналогічні сучасні системи

Програмний брандмауер можна встановити на сервери або віртуальні машини для захисту мережі, оскільки це брандмауер з програмним формфактором, а не фізичний пристрій. Він інтегрується з широким спектром інших технологічних рішень для забезпечення безпеки, щоб надати компаніям будь-якого розміру більш надійний і комплексний захист.

Програмний брандмауер робить встановлення політик безпеки набагато простішим, швидшим і більш адаптивним. Може відстежувати вхідний і вихідний трафік на предмет можливої небезпеки або підозрілої поведінки користувачів. Підприємствам подобається зниження стартових витрат і відсутність обмежень на кількість гаджетів, які можна захистити. Цей тип брандмауера особливо важливий, оскільки він займає мало місця на комп'ютері і може бути віддалено розгорнутий на численних пристроях.

Основні переваги програмного брандмауера полягають у наступному:

- нижча вартість, зазвичай, програмні брандмауери надають безкоштовну пробну версію і мають низьку щомісячну вартість. Бюджетування є дуже важливим, оскільки витрати з часом зростатимуть внаслідок розширення.

- обмеження простору, програмні брандмауери не мають фізичної присутності, оскільки є програмним забезпеченням;

- просте встановлення, всі системи в мережі можуть бути охоплені професійними та корпоративними версіями, які мають багатокористувацьке ліцензування. Звичайні програмні брандмауери легко встановлюються і налаштовуються, і ви можете швидко оцінити відповідний рівень безпеки. Однак корпоративні версії вимагають набагато більше додаткових знань і навичок;

- корпоративне та професійне ліцензування, підприємство має доступ до багатокористувацьких ліцензій і можете працювати з широким спектром сторонніх рішень для захисту, використовуючи програмний брандмауер корпоративного рівня;

- періодичні оновлення, оновлення програмного брандмауера можна виконати на всіх пристроях одним натисканням кнопки;

- спостереження за безпекою, програмні брандмауери можуть відстежувати як вхідний, так і вихідний трафік на предмет потенційних загроз, оскільки вони створені для роботи з поточними рішеннями безпеки;

- гнучкість, брандмауер легко адаптується до мінливих вимог безпеки і може бути налаштований на гнучке або суворе застосування політик безпеки. Програмні брандмауери пропонують більше можливостей для налаштування;

- портативність, програмний брандмауер набагато легше видалити або оновити, і він подорожує разом з пристроєм, куди б він не перемістився.

Програмні брандмауери мають ряд недоліків:

- швидкість або продуктивність хост-системи може постраждати, оскільки програмний брандмауер використовує її ресурси (процесор, пам'ять і т.д.);

- ціна придбання програмного брандмауера зростає зі збільшенням кількості комп'ютерів і систем у мережі. Крім того, недорогі або безкоштовні брандмауери часто неефективні в приватних і професійних умовах;

- на кожному комп'ютері необхідно встановлювати та оновлювати програму.

- продуктивність системи часто низька;

- ресурси системи витрачаються;

- несумісність з ігровими консолями або смарттелевізорами.

Апаратні брандмауери - це апаратні підключення до мережі, які фільтрують вхідний і вихідний трафік відповідно до заздалегідь встановлених критеріїв. Завдяки своїй незалежності від комп'ютера і відсутності споживання ресурсів, апаратні брандмауери є більш надійними, безпечними і швидкими, ніж програмні брандмауери. Однак вони дорожчі, складніші, їх важче налаштовувати та оновлювати. Якщо мережа велика або диверсифікована, встановлення та обслуговування апаратного брандмауера може потребувати професійної допомоги.

Апаратний брандмауер відстежує кожен інтернет-пакет до того, як він потрапляє на комп'ютер, і перевіряє його походження, IP-адресу та заголовок, щоб визначити, чи можна йому довіряти.

Більшість правил в апаратних брандмауерах є заздалегідь визначеними, вбудованими і покладаються на ці вбудовані правила; тому вони, як правило, не вимагають значних налаштувань.

Апаратні брандмауери гарантують, що жодні підозрілі дані не потраплять у мережу, і запобігають незаконному фізичному доступу. Тому апаратні брандмауери необхідні, доки існує фізичне обладнання.

Апаратний брандмауер має ряд переваг:

- швидкість, апаратні брандмауери, як правило, призначені для швидкої реакції і тому можуть справлятися з більшим навантаженням трафіку;
- контроль одного мережевого пристрою, Кожен комп'ютер, підключений до сервера, захищений одним апаратним брандмауером, що скорочує час і ресурси, необхідні для встановлення програмного забезпечення на кожній машині;
- просте керування вузлами, вузлом можна керувати ефективніше, якщо він ізольований від інших мережевих елементів і не завантажує та не уповільнює роботу інших програм;
- стандартні правила, набір попередньо налаштованих правил за замовчуванням, які при використанні з апаратним брандмауером можуть застосовуватися до всього трафіку;

- покращена безпека, апаратні брандмауери менш вразливі до деяких загроз, з якими можуть зіткнутися програмні брандмауери, коли комп'ютер скомпрометовано, оскільки вони мають власну, незалежну операційну систему;

- додаткові ресурси сервера, оскільки апаратний брандмауер є абсолютно окремим пристроєм від сервера, він надає доступ до додаткових ресурсів сервера, які в іншому випадку були б використані.

Налаштування корпоративних апаратних брандмауерів краще довірити досвідченим ІТ-фахівцям. Апаратний брандмауер має ряд недоліків:

- підвищена вартість, апаратні брандмауери зазвичай коштують дорожче, ніж програмні;

- потребує професійного налаштування, встановлення та налаштування цього типу брандмауерів вимагає певного досвіду;

- займають місце, апаратний брандмауер - це окремий фізичний об'єкт, який потребує місця як для себе, так і для кабелів, до яких він підключається;

- потребує моніторингу, апаратні брандмауери мають вбудований захист, але вони потребують постійного моніторингу та обслуговування.

Програмний брандмауер працює зсередини комп'ютера за допомогою програми, тоді як апаратний брандмауер - це фізичний пристрій на найпростішому рівні. Хоча їхні цілі схожі, вони підходять до них дещо по-різному, пропонуючи кожен з них унікальний набір переваг.

Хоча між апаратними і програмними брандмауерами є кілька незначних відмінностей, важливо знати про них, щоб вибрати той, який найкраще підійде під поставлені задачі.

Реалізація - одна з найважливіших відмінностей між апаратним і програмним брандмауером. Апаратний брандмауер постійно захищає всю мережу. З іншого боку, програмний брандмауер ефективний лише тоді, коли він розгорнутий на кожній системі, підключеній до мережі.

Апаратні брандмауери працюють у своїй операційній системі, тому вони не впливають на продуктивність, оскільки не використовують ресурси сервера. Крім того, апаратний брандмауер не захистить, якщо сервери будуть скомпрометовані

шкідливими даними. Один апаратний брандмауер можна використовувати для захисту багатьох підключених до нього пристроїв у всій мережі. Є можливість змінювати налаштування безпеки та налаштовувати будь-який пристрій, підключений до апаратного брандмауера.

Існують обмеження, пов'язані із захистом програмного брандмауера. Вони повинні бути встановлені на кожному комп'ютері в мережі. Якщо якийсь із пристроїв не готовий отримати оновлення, коли настане час оновити захист, його потрібно буде оновити вручну. Програмні брандмауери можуть виснажувати критично важливу пам'ять і обчислювальну потужність, оскільки їм не вистачає власних ресурсів, що впливає на зручність роботи користувачів і безпеку мережі.

Апаратні брандмауери є менш гнучкими, зручними та сконфігурованими, ніж програмні брандмауери, які дозволяють налаштувати параметри та правила відповідно до потреб. Крім того, їх встановлення та оновлення є більш доступними та простими. Однак програмні брандмауери менш безпечні, надійні та ефективні, ніж апаратні, оскільки вони залежать від продуктивності та безпеки вашого комп'ютера і можуть конфліктувати з іншими програмами або шкідливим програмним забезпеченням [4].

Таким чином, для розв'язання поставленої задачі є доцільною розробка апаратного брандмауера.

1.2 Апаратні засоби для розробки системи

Коректність роботи апаратного брандмауера забезпечується пристроєм керування, який відповідає за створення послідовності сигналів, необхідних для виконання команди, та послідовності команд, тобто програми. Команда в комп'ютері виконується за один або за декілька тактів, в кожному із яких виконується одна або декілька мікрооперацій. Кожна мікрооперація являє собою деяку елементарну дію передачі або пе-ретворення інформації, яка ініціюється

надходженням керуючого сигналу (мікрокоманди) на вхід керування відповідного пристрою. Прикладом може бути керуючий сигнал, який встановлює або очищує прапорець стану, керуючий сигнал запису до регістра, керуючий код на вході мультиплексора і т. д. Для реалізації команди необхідно на відповідні керуючі входи подати розподілену в часі послідовність керуючих сигналів.

Пристрій керування є одним з вузлів процесора. Процес функціонування процесора в часі складається з послідовності тактових інтервалів, в яких арифметико-логічний пристрій виконує операції над операндами та видає результати обробки. Виконання даних операцій арифметико-логічний пристрій здійснює на основі відповідних сигналів керування (мікрокоманд) з пристрою керування. Послідовність елементарних мікрокоманд пристрій керування формує на основі коду операції та службових сигналів стану з регістрової пам'яті процесора.

Відомі два основні методи побудови логіки формування керуючих сигналів. Перший з них виражається в тому, що для кожної команди процесора існує набір логічних схем, які в потрібних тактах збуджують відповідні сигнали керування. Такий принцип керування одержав назву "жорсткої" або "запаяної" логіки.

Другий метод, який дістав назву принципу мікропрограмного керування, передбачає формування керуючих сигналів за вмістом регістра мікрокоманд, в який мікрокоманди записуються із пам'яті мікрокоманд. Шляхом послідовного зчитування мікрокоманд із пам'яті в цей регістр організується потрібна послідовність керуючих сигналів [5].

Реалізувати пристрій керування можна на основі автономної інтегральної схеми, в якості якої може бути використані програмовані вентильні матриці (FPGA) та їх аналоги (ASIC і CPLD).

Оскільки центри обробки даних продовжують розвиватися, щоб задовольнити зростаючі вимоги сучасних додатків, програмовані вентильні матриці (FPGA) стали популярним, універсальним і ефективним рішенням для підвищення продуктивності.

Програмована вентильна матриця (FPGA) - це тип інтегральної схеми (ІС) або мікросхеми, яка налаштовується за допомогою масиву сконфігурованих логічних блоків і може бути переконфігурована після виготовлення.

CLB (Центральний логічний блок) - це основний логічний блок FPGA, який, як правило, містить комбінацію flip-flop і таблиць пошуку. Перекидні перемикачі забезпечують можливість зберігати та маніпулювати даними в послідовних логічних схемах, а таблиці пошуку використовуються для реалізації комбінаційної логіки.

FPGA можуть містити додаткові компоненти, такі як:

- інтерконектори - ресурси маршрутизації, які з'єднують різні логічні блоки;
- блоки вводу/виводу - компоненти, які забезпечують інтерфейс між зовнішнім світом і внутрішньою логікою FPGA;
- блоки цифрової обробки сигналів (DSP) - блоки, які полегшують виконання спеціалізованих функцій, таких як фільтрація або множення.

Виробники FPGA можуть реалізовувати власні цифрові схеми та змінювати конфігурацію компонентів відповідно до потреб кінцевого користувача.

Поряд з FPGA, ASIC і CPLD також є важливими компонентами в сучасних обчисленнях, які часто порівнюються в галузі. Знання відмінностей між ними - і те, який з них вибрати для певних застосувань - важливо як для початківців, так і для досвідчених ІТ-фахівців, які беруть участь у прийнятті обчислювальних рішень.

FPGA і ASIC - це два типи інтегральних схем зі схожою схемотехнікою, але вони відрізняються за призначенням і функціональністю. Загалом, ASIC розробляються на замовлення для конкретного застосування, тоді як FPGA є більш універсальними і можуть бути переобладнані для різних застосувань. CPLD менш складні, ніж FPGA і ASIC, і часто використовуються для простіших логічних задач.

FPGA (Field-Programmable Gate Array):

- найкраще підходить для: прототипування, розробки та додатків, що потребують частих змін дизайну;
- персональні налаштування: легко налаштовується та переналаштовується для широкого спектра застосувань;

- конфігурованість: можливість переналаштування після виготовлення;
- продуктивність: хороша продуктивність, але може не відповідати продуктивності ASIC через накладні витрати на конфігурацію;
- вартість: нижчі початкові витрати в порівнянні з ASIC;
- час розробки: швидший час розробки в порівнянні з ASIC;
- енергоспоживання: вище енергоспоживання за рахунок додаткових витрат на конфігурацію.

ASIC (Application-Specific Integrated Circuit):

- найкраще підходить для: побутова електроніка великих обсягів, спеціалізоване обладнання, критично важливі системи;
- персональні налаштування: легко налаштовується для конкретного застосування;
- конфігурованість: не підлягає зміні конфігурації після виготовлення;
- продуктивність: найвищий рівень продуктивності завдяки спеціальній конструкції;
- вартість: вимагає високих початкових витрат;
- час розробки: потребує більшого часу на розробку;
- енергоспоживання: найнижче енергоспоживання, оскільки дизайн оптимізовано під конкретні завдання.

CPLD (Complex Programmable Logic Device):

- найкраще підходить для: логіки склеювання, простих машин станів, завдання взаємодії та програм керування, які не вимагають високої складності;
- персональні налаштування: менш гнучкі та менш придатні для налаштування, ніж FPGA;
- конфігурованість: може запропонувати обмежену часткову реконфігурацію;
- продуктивність: зазвичай швидше, ніж FPGA;
- вартість: нижчі початкові витрати в порівнянні з ASIC;
- час розробки: швидший час розробки в порівнянні з ASIC;
- енергоспоживання: зазвичай споживає менше енергії, ніж FPGA.

Існує три типи FPGA: на основі статичної пам'яті з довільним доступом (SRAM), на основі флешпам'яті та з захистом від запобіжників.

FPGA на основі SRAM використовують енергозалежні комірки SRAM для зберігання даних конфігурації, які необхідно завантажувати кожного разу, коли FPGA вмикається. Вони зазвичай використовуються для створення прототипів, розробки та додатків, де потрібні часті зміни дизайну і тестування.

FPGA на основі флешпам'яті використовують енергонезалежні комірки флешпам'яті для зберігання даних конфігурації, що дозволяє FPGA зберігати свою конфігурацію і залишатися активними після увімкнення. FPGA на основі флешпам'яті найкраще підходять для додатків, які потребують миттєвої функціональності без необхідності перезавантаження даних конфігурації при кожному циклі живлення.

FPGA із захистом від запобіжників є енергонезалежними і залишаються під напругою після увімкнення живлення, але програмуються одноразово і не можуть бути скасовані або переконфігуровані. Ці характеристики роблять FPGA із захистом від запобіжників ідеальними для застосувань, які вимагають безпеки і не потребують додаткового перепрограмування, наприклад, в оборонній, аерокосмічній та інших галузях з високими вимогами до надійності.

FPGA надають багато переваг, розглянемо основні:

- універсальність, FPGA є дуже універсальними завдяки своїй програмованості та реконфігурованості, що дозволяє використовувати їх у широкому спектрі застосувань;

- швидкість та ефективність, FPGA здатні до паралельної обробки, що дозволяє їм пропонувати більш високі швидкості, більшу обчислювальну потужність, кращий час відгуку та загальну покращену продуктивність у порівнянні з іншими типами обладнання;

- вартість, порівняно з ASIC, FPGA забезпечують нижчі початкові витрати завдяки можливості перепрограмування знову і знову. Це особливо вигідно для малих і середніх обсягів виробництва, де початкові витрати на довгий цикл проектування і розробки ASIC можуть бути не виправданими;

- конфігурованість, основна перевага FPGA полягає в тому, що їх можна перепрограмувати. Ця якість дозволяє швидко і легко модифікувати FPGA відповідно до потреб користувача, на відміну від ASIC, які не можуть бути переконфігуровані після виготовлення;

- інтеграція, завдяки своїй гнучкій природі FPGA можна легко інтегрувати та швидко модернізувати у більші системи, ніж ASIC, які можуть вимагати додаткових інженерних витрат на оновлення.

Проблеми, пов'язані з використанням FPGA:

- вартість, хоча FPGA вимагають менших початкових витрат, ніж ASIC, через короткий час розробки, FPGA можуть бути дорожчими в довгостроковій перспективі при великосерійному виробництві;

- енергоспоживання, через можливість реконфігурації FPGA вони споживають більше енергії, ніж ASIC, оскільки їх конструкція дуже індивідуальна і оптимізована під конкретні завдання;

- складність, FPGA вимагають складного процесу програмування, що включає спеціалізовані знання апаратного дизайну, HDL, синтезу та інструментів розміщення і маршрутизації, особливо для великих і складних конструкцій;

- обмеження в часі, дані конфігурації необхідно завантажувати кожного разу при вмиканні FPGA, що може призвести до більш тривалого часу запуску в порівнянні з ASIC.

Типи мов програмування FPGA:

- verilog - це проста, легка для вивчення, текстова мова, яка широко прийнята в якості галузевого стандарту. Вона була розроблена, щоб нагадувати мову програмування C, частіше використовується в США і характеризується гнучким синтаксисом;

- vhdl, відома як сильно типізована мова опису апаратного забезпечення, що найкраще підходить для складних цифрових систем. Вона має синтаксис, що нагадує мову програмування Ada, частіше використовується в Європі і характеризується суворою логікою та розсудливістю;

- інші мови програмування для FPGA включають lucid, C та C++, python та фреймворки для штучного інтелекту, такі як TensorFlow та PyTorch.

Унікальна універсальність і обчислювальна потужність FPGA роблять їх чудовим рішенням для задоволення зростаючих потреб сучасної інфраструктури центрів обробки даних, що характеризуються збільшенням популярності хмарних обчислень, вибуховим зростанням обсягів створення даних і появою нових обчислювально-інтенсивних додатків, таких як машинне навчання [6].

Коли справа доходить до серій FPGA високого класу, Xilinx та Intel (колишня Altera) є двома провідними гравцями на ринку. Обидві компанії пропонують передові рішення FPGA з потужними функціями та можливостями. Порівняємо високопродуктивні серії ПЛІС від Xilinx та Intel за різними параметрами.

Відмінності в архітектурі та філософії проектування:

- високопродуктивні FPGA Xilinx базуються на власних архітектурах, таких як сімейства Virtex та UltraScale. Ці FPGA підкреслюють гнучкість, масштабованість і оптимізацію продуктивності за рахунок програмованих логічних ресурсів і ресурсів маршрутизації;

- висококласні FPGA Intel, відомі як сімейства Stratix і Arria, використовують власну архітектуру. Ці FPGA орієнтовані на високопродуктивні обчислення, з розширеними можливостями, такими як високошвидкісні приймачі, посилені процесори та блоки DSP.

Порівняння продуктивності:

- обчислювальна потужність. Високопродуктивні FPGA Xilinx та Intel пропонують подібну обчислювальну потужність, причому обидві компанії постійно розширюють межі продуктивності FPGA;

- швидкість. FPGA Xilinx та Intel забезпечують високошвидкісну продуктивність, але конкретні моделі та конфігурації FPGA можуть відрізнятися за тактовою частотою та досяжною швидкістю передачі даних;

- ефективність. Енергоефективність є вирішальним фактором для FPGA високого класу. Як Xilinx, так і Intel інвестують в технології оптимізації

енергоспоживання, але фактичне енергоспоживання може відрізнятися залежно від конкретних моделей FPGA і сценаріїв використання.

Інструменти розробки та екосистема програмного забезпечення:

- xilinx надає повний набір інструментів розробки, включаючи Vivado Design Suite та Vitis Unified Software Platform, які підтримують розробку, верифікацію та програмування FPGA;

- intel Quartus Prime Design Suite - це основний набір інструментів, пропонований Intel для розробки FPGA. Він включає широкий спектр функцій для проектування, моделювання та програмування FPGA Intel.

Доступність та підтримка інтерфейсів і протоколів:

- fpga Xilinx підтримують різні інтерфейси та протоколи, такі як PCIe, Ethernet, USB, HDMI тощо. Вони надають широкі IP-ядра та бібліотеки для безперешкодної інтеграції з різними системами;

- fpga Intel також пропонують підтримку широкого спектра інтерфейсів і протоколів, включаючи PCIe, Ethernet, USB та інші. Вони забезпечують багату екосистему IP-ядер і еталонних проєктів для легкої інтеграції.

Ціноутворення та економічні міркування:

- ціноутворення. Високопродуктивні FPGA Xilinx та Intel мають конкурентоспроможні ціни, але точна вартість може змінюватися залежно від таких факторів, як модель FPGA, функції та знижки за обсяг замовлення;

- загальна вартість володіння (ЗВВ): ЗВВ включає не тільки початкову вартість FPGA, але й такі фактори, як інструменти розробки, ліцензування інтелектуальної власності, енергоспоживання та підтримка. Важливо враховувати ЗВВ при оцінці висококласних FPGA Xilinx та Intel.

Спільнота користувачів та онлайн-ресурси:

- xilinx має велику та активну спільноту користувачів, з форумами, блогами та онлайн-ресурсами, які легко доступні для підтримки та обміну знаннями;

- спільнота користувачів FPGA Intel також добре розвинена, з форумами та ресурсами, призначеними для допомоги користувачам та обміну інформацією.

Отже, Xilinx та Intel є основними гравцями на ринку FPGA високого класу, пропонуючи потужні та передові рішення. Xilinx робить акцент на гнучкості, в той час, як Intel фокусується на високопродуктивних обчисленнях. Обидві компанії забезпечують подібну обчислювальну потужність і швидкість, з різною енергоефективністю.

Xilinx пропонує Vivado Design Suite та Vitis Unified Software Platform, а Intel - Quartus Prime Design Suite. Вони підтримують широкий спектр інтерфейсів і протоколів з великою кількістю IP-ядер. Ціни, спільнота користувачів та онлайн-ресурси є конкурентоспроможними для обох продуктів. Вибір залежить від вимог проєкту, продуктивності, інструментів, інтерфейсів та бюджету. Xilinx та Intel продовжують впроваджувати інновації та обслуговувати різні галузі [7].

Хоча обидва виробники пропонують потужні інструменти для розробки програмного забезпечення для FPGA, вибір на користь Intel може бути обґрунтованим через те, що проєкт потребує оптимізації для пристроїв на чіпі Intel. Quartus Prime Design Suite від Intel також може виявитися більш ефективним використанням ресурсів.

1.3 Програмні засоби для розробки системи

Для проєктування та програмування пристроїв ПЛІС пропонуються наступні програмні засоби розробки: Quartus II, Aldec Active-HDL.

Quartus II - це програмний засіб розробки, що надається компанією Intel (раніше Altera) для проєктування та програмування пристроїв ПЛІС. Ось деякі переваги та недоліки використання Quartus II:

Переваги:

- Quartus II пропонує широкий спектр інструментів і функцій для проєктування ПЛІС, включаючи введення проєкту, синтез, моделювання та аналіз

часових параметрів. Він надає повний набір інструментів для підтримки всього процесу проектування;

- програма підтримує різноманітні пристрої FPGA від Intel, включаючи серії Cyclone, Arria та Startix. Це дозволяє проєктувальникам орієнтуватися на різні сімейства ПЛІС в залежності від вимог проєкту;

- високорівневе введення даних для проєктування, Quartus II підтримує різні методи введення проєкту, включаючи захоплення схеми, Verilog, VHDL і SystemVerilog. Проєктувальники можуть вибрати найбільш підходящий метод, виходячи зі своїх знань та уподобань;

- програма містить розширені інструменти синтезу та оптимізації, які допомагають підвищити продуктивність, зменшити використання ресурсів та дотриматись часових обмежень. Він пропонує варіанти оптимізації для площі, швидкості та енергоспоживання;

- хронометражний аналіз включає в себе інструменти для аналізу часових параметрів, щоб гарантувати, що проєкт відповідає необхідним часовим обмеженням і працює правильно на потрібній тактовій частоті. Це допомагає дизайнерам виявляти та розв'язувати проблеми синхронізації на ранніх стадіях процесу проєктування.

Недоліки:

- Quartus II може мати круту криву навчання для початківців або новачків у проєктуванні ПЛІС. Складність інструменту та велика кількість функцій можуть вимагати певного часу для повного розуміння та ефективного використання;

- обмежена підтримка сторонніх інструментів Quartus II у порівнянні з іншими програмами для проєктування ПЛІС. Це може обмежити дизайнерів у використанні певних інструментів або інтеграцій, яким вони надають перевагу;

- Quartus II може бути ресурсоємним з точки зору системних вимог та апаратних ресурсів. Для ефективної роботи складних проєктів або симуляцій може знадобитися потужна комп'ютерна система;

- витрати на ліцензування, Quartus II - це власницьке програмне забезпечення, яке вимагає ліцензії для повної функціональності. Залежно від версії та необхідних функцій, витрати на ліцензування можуть додати до загальних витрат на проєкт;

- Quartus II може мати обмеження з точки зору підтримки платформи, наприклад, сумісності з певними операційними системами або конфігураціями обладнання. Проєктувальники повинні переконатися, що їхня система відповідає вимогам інструменту.

Загалом Quartus II пропонує повний набір інструментів і функцій для проєктування ПЛІС, але він може мати деякі недоліки, пов'язані зі складністю, тривалістю навчання, використанням ресурсів, вартістю ліцензування та підтримкою платформ. Дизайнери повинні враховувати ці фактори при виборі Quartus II для своїх проєктів з проєктування ПЛІС.

Aldec Active-HDL - популярне інтегроване середовище розробки (IDE) для цифрового проєктування та верифікації з підтримкою проєктів ПЛІС. Ось деякі переваги та недоліки використання Aldec Active-HDL:

Переваги:

- Active-HDL надає потужні можливості моделювання, включаючи підтримку різних мов моделювання, таких як VHDL, Verilog, SystemVerilog, а також багатомовне моделювання. Це дозволяє розробникам моделювати складні цифрові проєкти та перевіряти їх функціональність;

- програма підтримує синтез та реалізацію пристроїв на ПЛІС різних виробників, включаючи Intel/Altera, Xilinx та Lattice. Розробники можуть орієнтуватися на різні сімейства ПЛІС і технології, виходячи з вимог свого проєкту;

- Active-HDL підтримує різні методи введення проєкту, включаючи захоплення схеми, введення тексту VHDL/Verilog і графічних блок-схем. Проєктувальники можуть вибрати найбільш підходящий метод введення проєкту на основі своїх уподобань і знайомства з ним;

- програма надає розширені засоби налагодження, які допомагають розробникам виявляти та усувати проблеми у своїх проєктах. Такі функції, як

перегляд осцилограм, масштабування і трасування сигналів, полегшують налагодження складних конструкцій і перевірку правильної поведінки;

- програма пропонує функції для спільної роботи та контролю версій, що дозволяє декільком членам команди одночасно працювати над одним проєктом. Він підтримує інтеграцію з системами контролю версій, такими як Git, для ефективного управління проєктами.

Недоліки:

- крива навчання, Active-HDL може потребувати певного часу на освоєння, особливо для початківців або дизайнерів, які є новачками у розробці ПЛІС. Функції та інтерфейс інструменту можуть потребувати певного часу для повного розуміння та ефективного використання;

- Active-HDL є комерційним інструментом, який вимагає ліцензії для повної функціональності. Залежно від версії та необхідних функцій, витрати на ліцензування можуть додати до загальних витрат на проєкт;

- обмежена підтримка сторонніх інструментів та плагінів у порівнянні з іншими програмами для проєктування ПЛІС. Це може обмежити дизайнерів у використанні певних інструментів або інтеграцій, яким вони надають перевагу;

- запуск складних проєктів або симуляцій в Active-HDL може вимагати потужної комп'ютерної системи з точки зору системних вимог та апаратних ресурсів для ефективної роботи;

- Active-HDL може мати обмеження щодо платформи, такі як сумісність з певними операційними системами або конфігураціями апаратного забезпечення. Дизайнери повинні переконатися, що їхня система відповідає вимогам інструменту.

В цілому, Aldec Active-HDL пропонує повний набір функцій для проєктування і верифікації ПЛІС, але він може мати деякі недоліки, пов'язані зі складністю, тривалістю навчання, вартістю ліцензування, підтримкою сторонніх інструментів, використанням ресурсів і підтримкою платформ. Розробники повинні враховувати ці фактори при виборі Active-HDL для своїх проєктів ПЛІС.

Aldec Active-HDL та Quartus II - це популярні інструменти для розробки і верифікації програмних засобів на базі FPGA. Однією з основних відмінностей є те, що Aldec Active-HDL є інтегрованою середовищем для симуляції та верифікації HDL коду, тоді як Quartus II - це програмне забезпечення для синтезу, розробки та налагодження FPGA. Іншою відмінністю є те, що Active-HDL підтримує ряд інших мов програмування, таких як VHDL, Verilog, SystemVerilog, а Quartus II в основному орієнтований на Verilog та VHDL.

Проаналізувавши переваги та недоліки обох програм було зроблено вибір на користь програми Quartus II, яка стане основним інструментом для розробки проєкту.

1.4 Постановка завдання

Розробка пристрою керування для фільтрації мережевих пакетів по масці є актуальним завданням у сучасному цифровому світі для забезпечення безпеки та ефективності мережі. Цей тип пристрою керування дозволяє фільтрувати вхідні та вихідні мережеві пакети на основі попередньо визначеної маски, забезпечуючи додатковий рівень захисту від несанкціонованого доступу та зловмисних атак.

Основною метою цієї роботи є розробка пристрою керування, який може ефективно фільтрувати мережеві пакети за допомогою маски, тим самим покращуючи продуктивність та безпеку мережі. Цей пристрій забезпечить детальний та ефективний метод управління мережевим трафіком, дозволяючи адміністраторам визначати конкретні критерії для дозволеного або заблокованого зв'язку.

Для досягнення цієї мети необхідно вирішити кілька завдань:

- визначення вимог. Першим кроком у розробці пристрою керування для фільтрації мережевих пакетів за маскою є чітке визначення вимог системи. Це

включає визначення типів пакетів, які потрібно фільтрувати, маску для фільтрації та бажаний рівень продуктивності та захисту;

- реалізація алгоритму фільтрації. Серцем пристрою керування є алгоритм фільтрації, який визначає, як вхідні та вихідні мережеві пакети фільтруються на основі попередньо визначеної маски. Це завдання передбачає розробку ефективного алгоритму, який може обробляти великий обсяг пакетів в режимі реального часу;

- тестування та оптимізація системи. Після того, як алгоритм фільтрації був реалізований, наступним завданням є тестування системи та оптимізація її продуктивності. Це включає проведення різноманітних тестових випробувань, щоб переконатись, що пристрій керування працює як задумано та відповідає встановленим вимогам;

- оцінка безпеки. Нарешті, безпека пристрою керування повинна бути оцінена для того, щоб забезпечити необхідний рівень захисту від несанкціонованого доступу та атак. Це завдання передбачає проведення аудитів безпеки та тестування на проникнення з метою виявлення та виправлення можливих вразливостей системи.

Виходячи з ретельного аналізу поточного стану проблеми, очевидно, що розробка керуючого пристрою для фільтрації мережевих пакетів за маскою є не тільки необхідною, але й дуже корисною для вирішення проблем, що виникають у зв'язку з сучасними кібернетичними загрозами. Основні рішення зосередитися на розробці спеціалізованого пристрою для цієї мети обґрунтовані необхідністю посилення заходів безпеки, підвищення продуктивності мережі та більшої гнучкості в управлінні мережевим трафіком.

2 РОЗРОБКА СТРУКТУРНИХ ЧАСТИН СИСТЕМИ

2.1 Структура мережевого пакета повідомлення

Мережевий пакет - це основна одиниця даних, які групуються разом і передаються комп'ютерною мережею, зазвичай мережею з комутацією пакетів, як-от Інтернет. Кожен пакет або фрагмент даних є частиною повного повідомлення і містить відповідну адресну інформацію, яка допомагає ідентифікувати комп'ютер-відправник і передбачуваного одержувача повідомлення.

Мережевий пакет складається з трьох частин: заголовка (header), корисного навантаження (payload) і причепа (trailer). Розмір і структура мережевого пакета залежать від базової структури мережі або протоколу, що використовується. Концептуально, мережевий пакет схожий на поштову посилку. У цьому випадку заголовок - це коробка або конверт, корисне навантаження - це вміст, а трейлер - це підпис. Заголовок містить інструкції, пов'язані з даними в пакеті.

Мережевий пакет передається, вибираючи найкращий доступний маршрут до місця призначення. Це маршрут, яким йдуть всі інші пакети в повідомленні, що робить мережевий трафік більш ефективним з точки зору балансування навантаження на різні частини обладнання. Наприклад, якщо під час передачі повідомлення виникає проблема з обладнанням, пакети перенаправляються через маршрутизатори, щоб забезпечити доставку всього повідомлення до місця призначення.

Загалом, більшість мереж сьогодні працюють на стеку TCP/IP, що дозволяє пристроям, підключеним до Інтернету, спілкуватися один з одним через різні мережі.

Мережевий пакет або одиниця даних проходить процес інкапсуляції, який додає до нього інформацію, коли він рухається до місця призначення, і позначає, де він починається і закінчується. Мережевий пакет складається з трьох частин.

Заголовок (header) - це початок або передня частина пакета. Будь-який пристрій обробки або прийняття, наприклад, маршрутизатор або комутатор, бачить заголовок першим.

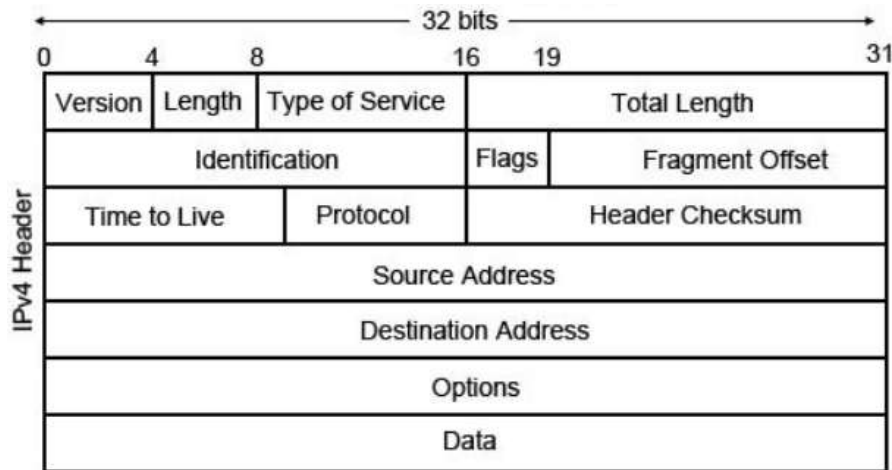


Рисунок 2.1 – Структура заголовка IP пакета

Наступні поля включені в заголовок протоколу IPv4:

- версія. Це поле вказує на формат інтернет-заголовка;
- довжина інтернет-заголовка (IHL). IHL - це довжина інтернет-заголовка в 32-бітних словах, які вказують на початок даних;
- тип послуги. Тут вказуються абстрактні параметри бажаної якості обслуговування;
- загальна довжина. Це довжина датаграми, виміряна в октетах, яка містить інтернет-заголовок і дані. Це поле дозволяє довжину дейтаграми до 65 535 октетів;
- ідентифікація. Відправник присвоює ідентифікаційне значення, щоб допомогти зібрати фрагменти дейтаграми;
- прапори. Перелік різних керуючих прапорів;
- зсув фрагмента. Це поле вказує, де в дейтаграмі знаходиться цей фрагмент. Зсув фрагмента вимірюється в одиницях восьми октетів, або 64 біт. Перший фрагмент має нульове зміщення;
- час життя (TTL). Поле TTL вказує на максимальний час, протягом якого дейтаграма може залишатися в інтернет-системі. Якщо це поле містить значення нуль, то дейтаграма повинна бути знищена;

- протокол. Це поле вказує на протокол наступного рівня, який використовується в частині даних інтернет-дейтаграми;
- контрольна сума заголовка. Контрольна сума дозволяє виявити пошкодження в заголовку IPv4-пакетів;
- адреса джерела. Це 32-бітна IP-адреса джерела;
- адреса призначення. Це 32-бітна IP-адреса призначення;
- параметри. Це поле є необов'язковим, і його довжина може бути змінною.

Корисне навантаження (payload). Це фактична інформація про дані, яку несе пакет до місця призначення. В IPv4 корисне навантаження доповнюється нульовими бітами, щоб гарантувати, що пакет закінчується на 32-бітній межі.

Причіп (trailer). Іноді деякі мережеві протоколи також додають до пакета кінцеву частину або ж трейлер. IP-пакет не містить трейлерів, але кадри Ethernet містять.

Пакети використовуються для ефективної та надійної передачі даних. Замість того, щоб передавати величезний файл як один блок даних, надсилання його меншими пакетами підвищує швидкість передачі. Пакети також дозволяють декільком комп'ютерам використовувати одне з'єднання.

Хоча можна передавати дані без використання пакетів, було б вкрай непрактично надсилати дані без попереднього розбиття їх на менші частини.

Основні переваги використання пакетів:

- для маршрутизації пакетів до місця призначення можна використовувати різні шляхи. Цей процес називається комутацією пакетів;
- якщо трапляється помилка, пакети можуть бути збережені і повторно передані пізніше;
- пакети використовують найкращий маршрут, доступний для доставки. Це дозволяє їм проходити через перевантажені ділянки мережі, не сповільнюючи їх у певному місці;
- для забезпечення безпечної доставки пакети можуть бути зашифровані [8].

IP-адреса – це ряд цифр, присвоєних пристрою, підключеному до Інтернету, подібно до адреси будинку. Комп'ютерна мережа використовує IP-адресу для

зв'язку з іншими комп'ютерами, вебсайтами та усіма частинами Інтернету. По суті, IP-адреси – це те, як комп'ютери в Інтернеті розпізнають один одного [9].

Маска підмережі - це 32-бітне число, створене шляхом встановлення бітів хоста на всі 0 і бітів мережі на всі 1. Таким чином, маска підмережі розділяє IP-адресу на мережеву та адресу хоста.

IP-адреса, маска підмережі та шлюз або маршрутизатор складають базову структуру - Інтернет-протокол, який використовується в більшості мереж для полегшення зв'язку між пристроями.

Коли організаціям потрібна додаткова підмережа, підмережування розділяє елемент хоста IP-адреси на підмережі. Мета масок підмереж - просто увімкнути процес створення підмереж. Слово "маска" застосовується тому, що маска підмережі використовує власне 32-бітне число для маскуванню IP-адреси.

32-розрядна IP-адреса однозначно ідентифікує окремий пристрій в IP-мережі. 32 двійкових біти розділені на хост і мережеву частини маскою підмережі, але вони також розбиті на чотири 8-бітових октети.

Оскільки двійкова система числення є складною, перетворюємо кожен октет таким чином, щоб вони були виражені в десятковій системі числення. Це призводить до характерного десятково-крапкового формату для IP-адрес - наприклад, 172.16.254.1. Діапазон значень у десятковій системі числення - від 0 до 255, оскільки саме так у двійковій системі числення відображаються адреси від 00000000 до 11111111.

Оскільки Інтернет повинен вміщати мережі всіх розмірів, існує схема адресації для різних мереж, заснована на тому, як розбиваються октети в IP-адресі. За трьома старшими або крайніми лівими бітами будь-якої IP-адреси можна визначити, до якого з п'яти різних класів мереж, від А до Е, належить ця адреса.

Мережі класу D зарезервовані для багатоадресної розсилки, а мережі класу Е не використовуються в Інтернеті, оскільки вони зарезервовані для досліджень, що проводяться Робочою групою з розробки інтернету IETF.

Маска підмережі класу А відображає частину мережі в першому октеті і залишає октети 2, 3 і 4 для мережевого менеджера, щоб він міг розділити її на хости

і підмережі за потребою. Клас А призначений для мереж з більш ніж 65 536 хостами.

У масці підмережі класу В перші два октети призначені для мережі, а решта адреси, 16 біт октетів 3 і 4, призначені для підмережі та хостів. Клас В призначений для мереж з 256 до 65 534 хостами.

У масці підмережі класу С мережева частина - це перші три октети, а хости і підмережі - лише решта 8 біт октету 4. Клас С призначений для невеликих мереж з кількістю хостів менше 254.

Мережі класів А, В і С мають природні маски або маски підмереж за замовчуванням:

- клас А: 255.0.0.0;
- клас В: 255.255.0.0;
- клас С: 255.255.255.0 .

Можна визначити кількість і тип IP-адрес, необхідних для певної локальної мережі, на основі маски підмережі за замовчуванням.

Прикладом IP-адреси і маски підмережі класу А може бути підмаска за замовчуванням класу А 255.0.0.0 і IP-адреса 10.20.12.2.

Підмережування - це метод логічного розділення однієї фізичної мережі на декілька менших підмереж.

Переваги підмереж включають в себе:

- зменшення обсягу ширококомовлення і, відповідно, мережевого трафіку;
- можливість роботи з дому;
- дозволяє організаціям подолати обмеження локальної мережі, такі як максимальна кількість хостів;
- мережева адресація.

Стандартним сучасним мережевим префіксом, який використовується як для IPv6, так і для IPv4, є нотація безкласової міждоменної маршрутизації (CIDR). IPv4-адреси, представлені в нотації CIDR, називаються мережевими масками, і вони визначають кількість бітів у префіксі адреси після похилої риски (/), що стоїть після роздільника.

Для призначення IP-адреси мережевому інтерфейсу з моменту появи CIDR існує два параметри: маска підмережі та адреса. Підмережі збільшують складність маршрутизації, оскільки в таблицях кожного підключеного маршрутизатора повинен бути окремий запис для представлення кожної локально підключеної підмережі [10].

2.2 Структура пристрою фільтрації

Система фільтрації інформації — це система, яка усуває надлишкову або небажану інформацію з інформаційного потоку за допомогою автоматичних або комп'ютерних методів і надає її користувачу.

Схема фільтрування пакетів даних використовується для запобігання проникненню в систему пакетів даних які не відповідають певним критеріям, в нашому випадку це маска. Структурна схема простого пристрою зображена на рисунку 2.2.

Схема фільтрування пакетів даних складається з двох основних компонентів: елемента порівняння (компаратор) та мультиплексора. Компаратор використовується для порівняння вхідних даних, в даному випадку – співпадіння IP-адрес пакету, що передається, та приймального пристрою. В рамках пристрою фільтрації мультиплексор використовується для дозволу передачі пакета до приймаючого пристрою.

На вхід подається деякий пакет даних та шаблон IP адреси з маскою. Далі виконується логічна операція “AND” для цих даних та в результаті формується код IP адреси мережі в яку прямує пакет та код IP адреси мережі шаблону. Компаратор порівнює ці адреси та робить певний висновок. Якщо мережі співпадають, то надається дозвіл для надсилання пакета даних до місця призначення.

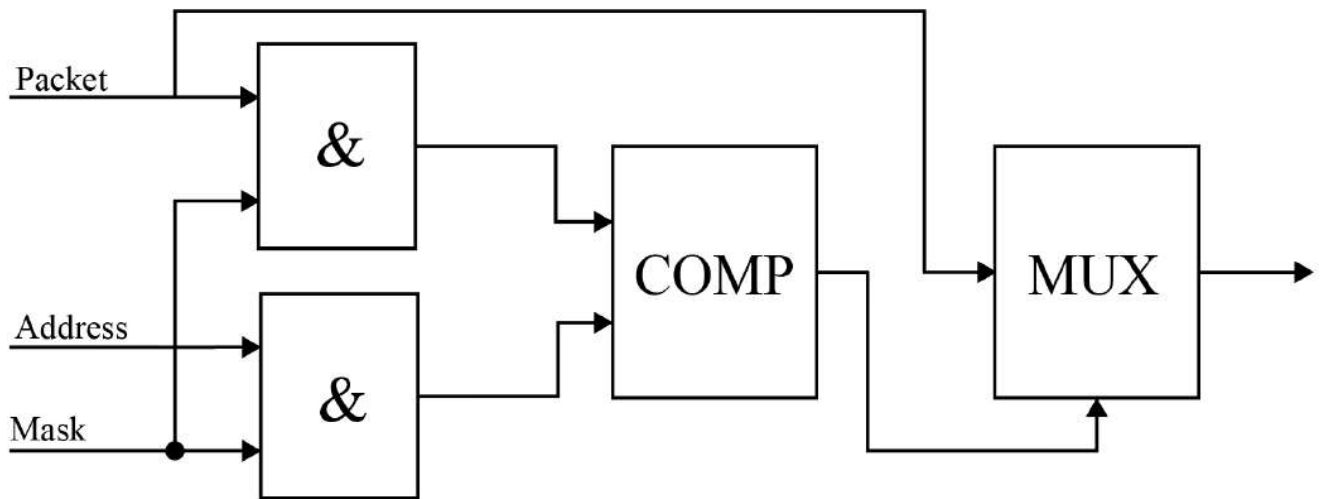


Рисунок 2.2 – Структурна схема простого фільтра пакетів

На рисунку 2.2 зображено структурну схему простого фільтра пакетів, але для реальних мереж такого фільтра буде замало, тому розглянемо більш складну схему, яка зображена на рисунку 2.3. Дана схема призначена для сортування пакетів за певними IP-адресами та масками.

Схема даного пристрою складається з трьох основних компонентів: компаратора, кодера та демультимплексора. Компаратор використовується для порівняння вхідних даних. Кодер призначено для задовування вихідного значення компаратора, який згенерував співпадіння кодів IP-адрес. Демультимплексор використовується для організації передачі пакета даних до відповідного виходу на основі результату задованого порту.

На вхід подається деякий пакет даних та чотири шаблони IP адрес з маскою. Далі виконується логічна операція “AND” для всіх цих даних та в результаті отримуємо задоване значення мережі, яке залежить від IP-адреси і маски підмережі в яку прямує пакет та чотири IP адреси мереж шаблонів. Компаратор порівнює ці адреси та робить певний висновок. Якщо одна з мереж в шаблоні співпадає з вказаною в пакеті мережею, то надається дозвіл для надсилання пакета даних до місця призначення. Рішення про напрямок пересилання пакета приймається на основі вихідного сигналу кодера, сформованого вихідними значеннями компараторів.

В залежності від номера компаратора формується відповідний код на виході кодера, який подається на вхід демультиплексора і визначає приймач пакета. Дозвіл на роботу демультиплексора визначається на основі логічного елементу “OR”: Якщо спрацював будь-який компаратор, то на виході логічного елементу формується логічна “1”.

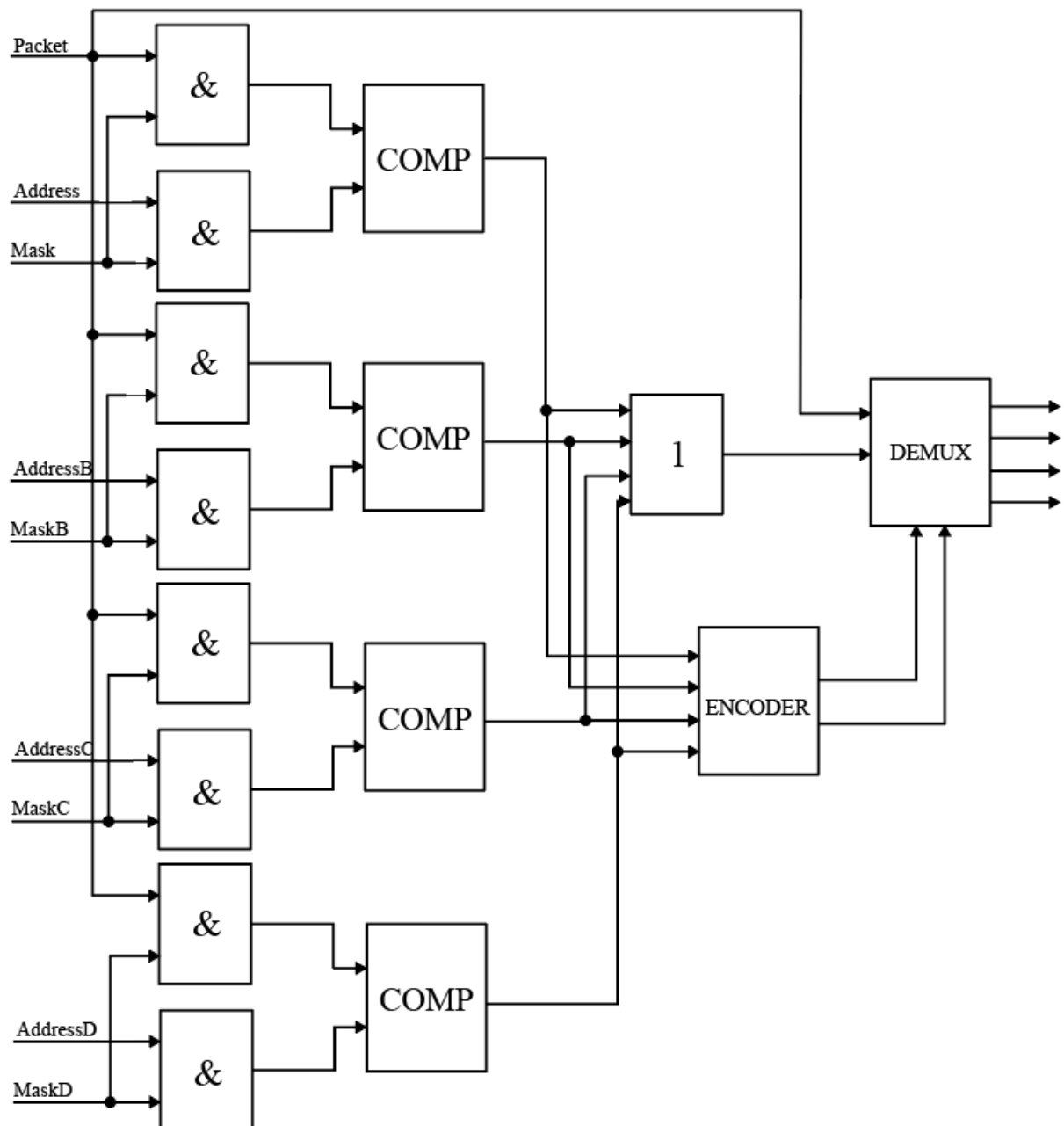


Рисунок 2.3 – Структурна схема пристрою сортування пакетів

В залежності від поставленої задачі і структури мережі можна використовувати як пристрій фільтрації пакетів, так і пристрій їх сортування.

2.3 Розробка компаратора

Компаратор – це елемент порівняння, який широко використовується в системах контролю та автоматичного керування. В рамках пристрою, що розробляється, компаратор використовується для порівняння кодів IP-адрес, вказаних в пакеті та тих, що належать приймаючому пристрою.

В структурній схемі компаратор має 2 входи даних, а саме, IP мережа пакета та IP мережа шаблону. Дані з цих двох входів порівнюються між собою. Компаратор має один вихід, дані якого залежать від результату порівняння вхідних даних, якщо вони однакові, то на виході буде логічна одиниця, якщо вхідні дані різні, то буде на виході логічний нуль.

Для даного проєкту було розроблено окремий компаратор, код якого наведено у лістингу 2.1.

Лістинг 2.1 – Фрагмент коду компаратора

```
ENTITY lpm_compare0 IS
    PORT
    (
        dataa          : IN STD_LOGIC_VECTOR (31 DOWNT0 0);
        datab          : IN STD_LOGIC_VECTOR (31 DOWNT0 0);
        aeb             : OUT STD_LOGIC ;
        aneb            : OUT STD_LOGIC
    );
END lpm_compare0;

ARCHITECTURE SYN OF lpm_compare0 IS
    SIGNAL sub_wire0    : STD_LOGIC ;
    SIGNAL sub_wire1    : STD_LOGIC ;

    COMPONENT lpm_compare
    GENERIC (
        lpm_representation : STRING;
        lpm_type           : STRING;
        lpm_width           : NATURAL
    );
    PORT (
        aeb : OUT STD_LOGIC ;
        aneb : OUT STD_LOGIC ;
        dataa : IN STD_LOGIC_VECTOR (31 DOWNT0 0);
        datab : IN STD_LOGIC_VECTOR (31 DOWNT0 0)
```

```

    );
    END COMPONENT;
BEGIN
    aeb    <= sub_wire0;
    aneb   <= sub_wire1;
    LPM_COMPARE_component : LPM_COMPARE
    GENERIC MAP (
        lpm_representation => "UNSIGNED",
        lpm_type => "LPM_COMPARE",
        lpm_width => 32
    )
    PORT MAP (
        dataa => dataa,
        datab => datab,
        aeb => sub_wire0,
        aneb => sub_wire1
    );
END SYN;

```

Цей VHDL код описує модуль компаратора, який використовується для порівняння двох вхідних сигналів, “dataa” та “datab”, 32-бітних цілих беззнакових чисел. Компаратор виводить два сигнали, “aeb” та “aneb”.

Компаратор працює наступним чином:

- порівнюються два вхідні сигнали “dataa” та “datab”;
- вихідний сигнал “aeb” встановлюється в 1, якщо “dataa” повністю співпадає з “datab”. В іншому випадку він дорівнює 0;
- вихідний сигнал “aneb” дорівнює 1, якщо “dataa” не співпадає “datab”. В іншому випадку він дорівнює 0.

Компаратор реалізовано за допомогою компонента LPM_COMPARE з бібліотеки LPM, який є попередньо зібраним компонентом, що надає простий спосіб реалізації функції порівняння.

Компонент налаштовується за допомогою наступних параметрів:

- lpm_representation. Цей параметр визначає представлення вхідних даних, яке в даному випадку має значення "UNSIGNED", що вказує на те, що вхідні дані є цілим беззнаковим числом;
- lpm_type. Цей параметр визначає тип виконуваного порівняння, який в даному випадку має значення "LPM_COMPARE", що вказує на те, що порівняння має бути виконано як просте порівняння;

- `lpm_width`. Цей параметр визначає ширину вхідних даних, яка у цьому випадку дорівнює 32, тобто вхідні дані мають ширину 32 біти.

2.4 Розробка кодера

Кодер (encoder) - це цифрова логічна схема, яка приймає набір двійкових вхідних сигналів і перетворює їх в один двійковий вихідний код. Кількість вихідних бітів, необхідних для представлення всіх можливих вхідних комбінацій, визначається кількістю вхідних бітів.

В рамках пристрою, що розробляється, кодер використовується для приймання та аналізу кодів компараторів та перетворення їх у відповідний код номера пристрою.

В структурній схемі кодер має чотири входи які приймають певні дані (нули або одиниці) від компараторів. На основі цих даних кодер виводить на 2 виходи дані.

Для даного проєкту було розроблено окремий кодер, VHDL код якого наведено у лістингу 2.2.

Лістинг 2.2 –Код пристрою encoder

```
library IEEE;
USE ieee.std_logic_1164.all;
entity Encoder is
    Port ( I1 : in  STD_LOGIC;
           I2 : in  STD_LOGIC;
           I3 : in  STD_LOGIC;
           I4 : in  STD_LOGIC;
           Y1 : out STD_LOGIC;
           Y2 : out STD_LOGIC);
end Encoder;
architecture Behavioral of Encoder is
begin
    process(I1, I2, I3, I4)
    begin
        if (I1 = '1' and I2 = '0' and I3 = '0' and I4 = '0') then
            Y1 <= '0';
```

```

        Y2 <= '0';
    elsif (I1 = '0' and I2 = '1' and I3 = '0' and I4 = '0') then
        Y1 <= '0';
        Y2 <= '1';
    elsif (I1 = '0' and I2 = '0' and I3 = '1' and I4 = '0') then
        Y1 <= '1';
        Y2 <= '0';
    elsif (I1 = '0' and I2 = '0' and I3 = '0' and I4 = '1') then
        Y1 <= '1';
        Y2 <= '1';
    end if;
end process;
end Behavioral;

```

Для реалізації логіки роботи кодера у VHDL-кодi використовується поведінковий процес. Інструкція процесу визначає блок коду, який виконується щоразу, коли змінюється сигнал у списку (I1, I2, I3 та I4). Усередині процесу є ряд операторів if-elsif, які визначають вихідні значення (Y1 і Y2) на основі вхідних значень. Таблиця істинності наведена в таблиці 2.1.

Таблиця 2.1 - Таблиця істинності кодера

Ввід	Y1	Y2
1000	0	0
0100	0	1
0010	1	0
0001	1	1
Інше	X	X

Як видно з таблиці істинності, кодер перетворює 4-бітне двійкове число у відповідний йому 2-бітовий двійковий код. Наприклад, якщо на вхід подано (1, 0, 0, 0), на виході буде (0, 0), що є двійковим кодом числа 0. Аналогічно, якщо на вхід подано (0, 0, 0, 1), на виході буде (1, 1), що є двійковим кодом числа 3.

2.5 Розробка демультиплексора

Демультиплексор - це комбінаційна логічна схема, призначена для перемикання однієї загальної вхідної лінії на одну з декількох окремих вихідних ліній [11].

В рамках пристрою, що розробляється, демультиплексор використовується для визначення відповідного порту виходу для пакета даних, використовуючи при цьому закодоване значення кодера.

В структурній схемі демультиплексор має чотири входи. Керуючий вхід працює таким чином, що дозволяє роботу демультиплексора, якщо на вході одиниця, або забороняє роботу, якщо на вході нуль. Два входи вибору в залежності від їх даних обирається порт виходу. Та вхід даних, який приймає пакет та чекає на подальшу роботу. Демультиплексор має чотири вихідних порти на які в подальшому буде передано пакет даних.

Для даного проєкту було розроблено окремий демультиплексор, VHDL код якого наведено у лістингу 2.3.

Лістинг 2.3 – Код демультиплексора

```

LIBRARY ieee;
USE ieee.std_logic_1164.all;
ENTITY demux IS
    PORT
    (
        data0x      : IN STD_LOGIC_VECTOR (159 DOWNT0 0);
        en          : IN STD_LOGIC;
        sel0        : IN STD_LOGIC ;
        sel1        : IN STD_LOGIC ;
        y1          : OUT STD_LOGIC_VECTOR (159 DOWNT0 0);
        y2          : OUT STD_LOGIC_VECTOR (159 DOWNT0 0);
        y3          : OUT STD_LOGIC_VECTOR (159 DOWNT0 0);
        y4          : OUT STD_LOGIC_VECTOR (159 DOWNT0 0)
    );
END demux;
ARCHITECTURE Behavioral OF demux IS
BEGIN
    PROCESS(data0x, sel0, sel1, en)
    BEGIN

```

```

IF en = '1' THEN
  IF sel0 = '0' AND sel1 = '0' THEN
    y1 <= data0x;
  ELSIF sel0 = '0' AND sel1 = '1' THEN
    y2 <= data0x;
  ELSIF sel0 = '1' AND sel1 = '0' THEN
    y3 <= data0x;
  ELSIF sel0 = '1' AND sel1 = '1' THEN
    y4 <= data0x;
  END IF;
END IF;
END PROCESS;
END Behavioral;

```

Демультимплексор працює наступним чином: Вхідний сигнал “data0x” є 160-бітним вектором. Два керуючих сигнали, “sel0” та “sel1”, мають по одному біту. Вони визначають, який вихідний сигнал має бути підключений до вхідного сигналу. Сигнал дозволу “en” - це один біт, який вмикає або вимикає демультимплексор. Якщо “en” дорівнює 1, демультимплексор виводить вхідний сигнал. Якщо “en” дорівнює 0, демультимплексор нічого не виводитиме. Вихідні сигнали “y1”, “y2”, “y3” та “y4” є 160-бітними векторами.

Оператор процесу перевіряє значення керуючих сигналів і дозволяє виведення відповідного вихідного сигналу. Наприклад, якщо “sel0” дорівнює 0 і “sel1” дорівнює 0, то вихідний сигнал “y1” буде підключено до вхідного сигналу. Якщо “sel0” дорівнює 0 і “sel1” дорівнює 1, то вихідний сигнал “y2” буде підключено до вхідного сигналу.

Демультимплексор можна використовувати для маршрутизації одного вхідного сигналу на один з чотирьох різних вихідних сигналів на основі значень керуючих сигналів.

3 РОЗРОБКА АПАРАТНОЇ СХЕМИ СИСТЕМИ

3.1 Розробка схеми пристрою

Логічні оператори - це спеціальні символи, які використовуються в програмуванні та логічних головоломках для комбінування умов і прийняття рішень. Вони діють як перемикачі, які керують потоком програми на основі того, чи є певні твердження істинними, чи хибними. Існує три основні логічні оператори:

- and. Цей оператор означає "обидва" або "все перераховане вище". Для того, щоб він був істинним, обидві умови, з'єднані оператором AND, повинні бути істинними. Наприклад, ворота з двома перемикачами: тільки якщо обидва перемикачі увімкнені, ворота дозволять прохід;

- or. Цей оператор означає "принаймні один" та "або". Поки одна з умов, з'єднаних оператором OR, істинна, все твердження вважається істинним. Наприклад, ворота з двома окремими перемикачами: якщо один з них увімкнено, ворота відкриваються;

- not. Цей оператор перевертає значення однієї умови. Якщо твердження істинне, NOT робить його хибним, і навпаки. Це як один перемикач, який змінює напрямок сигналу.

Ці базові оператори можна комбінувати для створення більш складних логічних виразів, що дозволяє програмам приймати тонкі рішення на основі декількох умов.

Базуючись на логічних операторах та розроблених пристроях (компаратор та мультиплексор) було створено структурну схему пристрою для фільтрації мережевих пакетів по масці (рисунок 3.1).

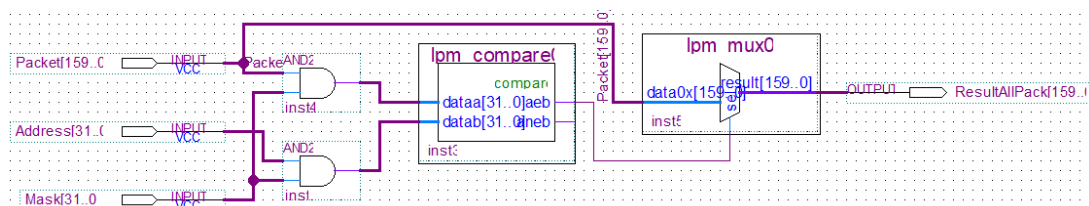


Рисунок 3.1 - Структурна схема пристрою фільтрації

На рисунку зображено 3 входи (input), вхід “Packet” має розмір 160 біт даних, він являє собою мережевий пакет даних який надходить з мережі. Два входи “Address” та “Mask” являють собою IP адресу та маску підмережі (кожен елемент розміром 32 біта), згідно з цими даних і відбувається фільтрація.

Пара логічних операторів “AND” слугують для того, щоб виокремити загальну IP адресу мережі для подальшої роботи пристрою.

Далі йде компаратор, який порівнює IP адресу мережі в яку прямує пакет даних з шаблоном IP адреси мережі фільтрації. Якщо адреса пакета даних та адреса шаблону співпадають, то на виході буде 1, якщо дані виявилися різними, то на виході буде 0.

Мультиплексор має 2 входи. Вхід “data0x” приймає повний пакет даних та очікує на подальші дії. Вхід “sel” слугує для того, щоб керувати дозволом на проходження пакета даних. Наприклад, при значенні “sel = 0” пакет даних буде відкинуто, або при значенні “sel = 1” пакет даних буде спрямовано на вихід.

Базуючись на логічних операторах та розроблених пристроях (компаратор, кодер та демультимплексор) було створено структурну схему пристрою сортування пакетів по масці (рисунок 3.2).

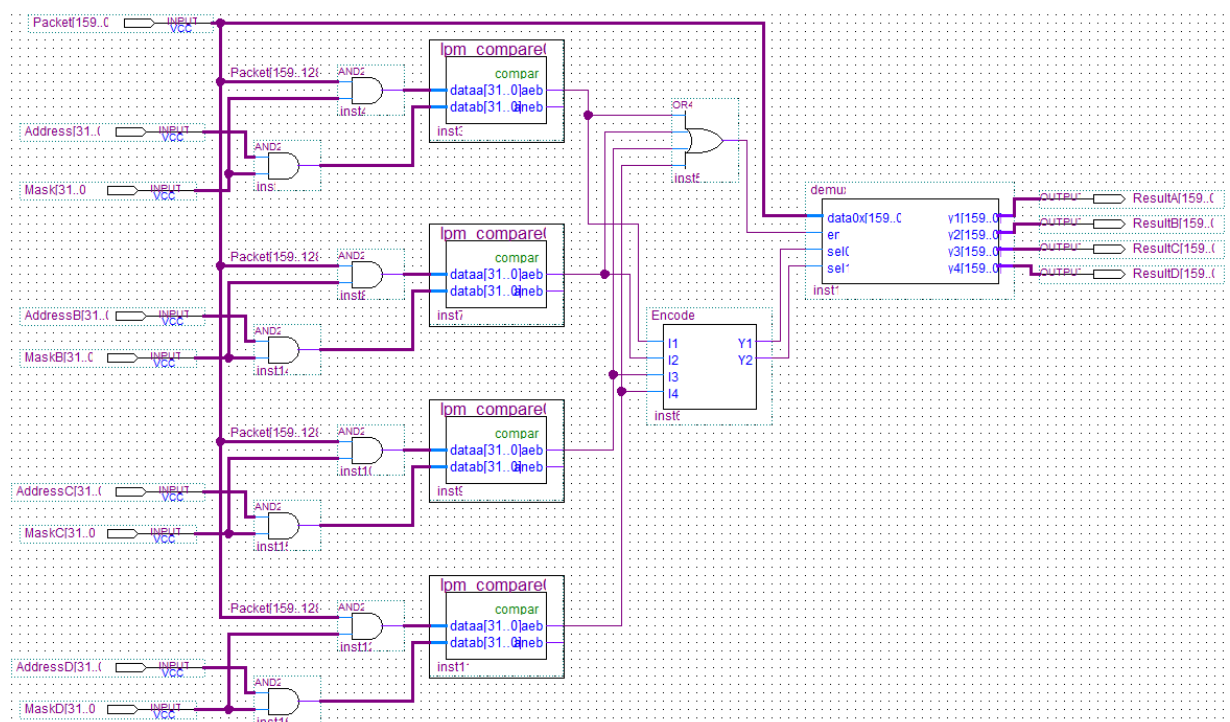


Рисунок 3.2 - Структурна схема пристрою сортування

На рисунку зображено 9 входів (input), вхід “Packet” має розмір 160 біт даних, він являє собою мережевий пакет даних який надходить з мережі. Чотири пари входів “Address, Mask” – “AddressD, MaskD” являють собою IP адресу та маску підмережі (кожен елемент розміром 32 біта), згідно з цими даними і відбувається фільтрація.

Чотири пари логічних операторів “AND” слугують для того, щоб виокремити загальну IP адресу мережі для подальшої роботи пристрою.

Далі йдуть 4 компаратори які порівнюють IP адресу мережі в яку прямує пакет даних з шаблоном IP адреси мережі фільтрації. Якщо адреса пакета даних та адреса шаблону співпадають, то на виході буде 1, якщо дані виявилися різними, то на виході буде 0.

Логічний оператор “OR” приймає виходи з чотирьох компараторів, якщо хоч один вихід передає одиницю, то на виході оператора також буде одиниця, якщо на всіх входах 0, то на виході буде 0, цей нуль повідомляє, що співпадіння даних не має, відповідно пакет даних потрібно відкинути.

Кодер приймає виходи з чотирьох компараторів і залежно від того на якому виході знаходиться одиниця виводить певні дані на виходи “Y1” та “Y2”.

Демультимплексор має 4 входи. Вхід “data0x” приймає повний пакет даних та очікує на подальші дії. Вхід “en” приймає дані від логічного оператора “OR”, якщо на вході 1, то демультимплексор отримує дозвіл на роботу, якщо 0, то демультимплексор не працює. Входи “sel0” та “sel1” слугують для того, щоб обрати на який саме вихід (y1-y4) буде направлено пакет даних. Наприклад, при значеннях “sel0 = 0” та “sel1 = 0” пакет даних буде направлено на вихід “y1”.

3.2 Тестування схеми пристрою

Тестування схеми, також відоме як перевірка або налагодження схеми, - це процес перевірки того, що цифрова схема або система функціонує за призначенням

і відповідає своїм специфікаціям. Мета тестування - виявити і виправити будь-які помилки або дефекти в дизайні до того, як він буде виготовлений або розгорнутий.

Для тестування схеми пристрою у програмі Quartus II було використано текстовий файл у форматі ASCII (з розширенням .vwf), який описує вхідні та вихідні вектори симуляції у вигляді графічних осцилограм.

Для кращої наочності протестуємо схему пристрою для фільтрації з використанням додаткових виходів. Часткова схема, яку буде протестовано, зображена на рисунку 3.3.

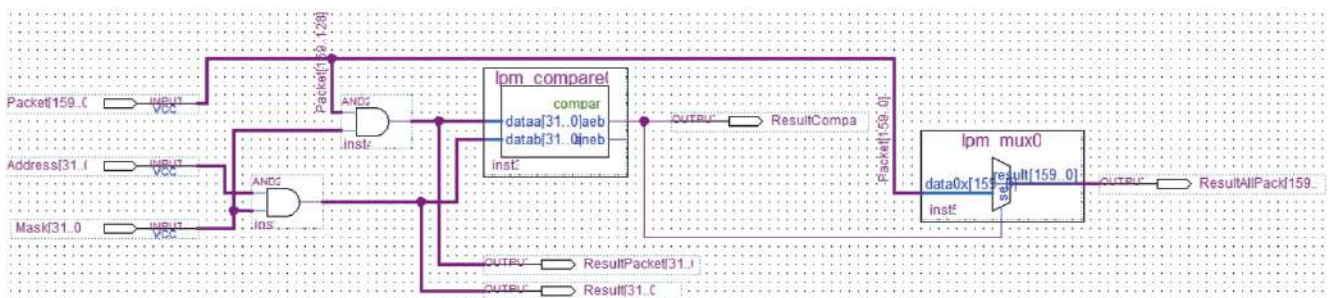


Рисунок 3.3 – Часткова схема пристрою для фільтрації

Виконуючи тестування схеми було використано такі вхідні дані:

- Packet: 192.168.10.10 - 11000000.10101000.00001010.00001010 (в двійк.);
- Address: 192.168.10.10 - 11000000.10101000.00001010.00001010 (в двійк.);
- Mask: 255.255.0.0 – 11111111.11111111.00000000.00000000 (в двійк.).

В результаті тестування було отримано дані, які зображено на рисунку 3.4. Вихідні порти “Result” та “ResultPacket” виводять код IP адреси мережі, в яку прямує пакет, та IP адреси мережі шаблону. Далі ці коди порівнюються компаратором, порт “ResultCompare” вивів одиницю, а це значить, що коди співпали. Далі ця логічна одиниця надходить до мультиплексора, який дозволяє передачу пакета даних на вихід.

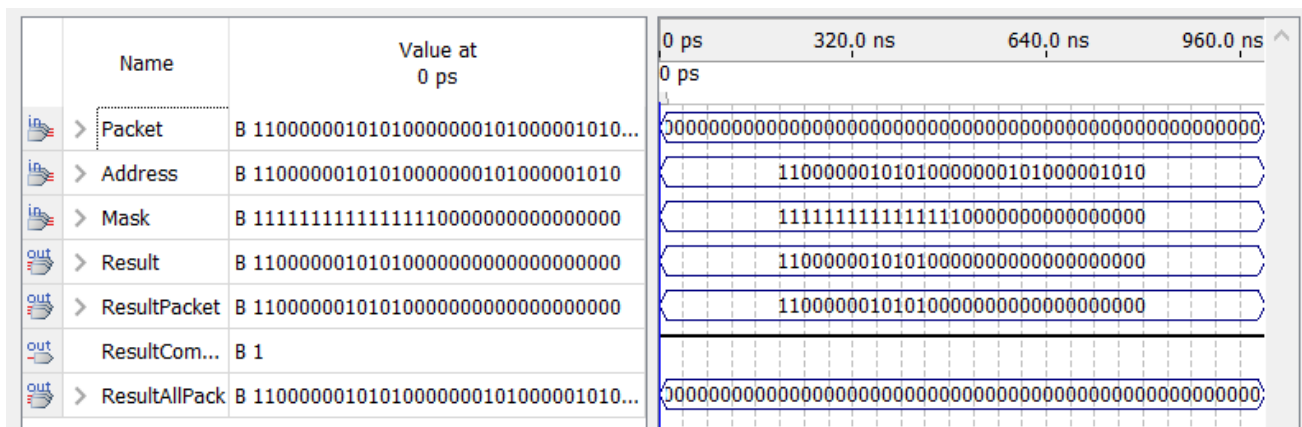


Рисунок 3.4 – Успішний результат тестування схеми

Розглянемо результат тестування схеми при таких вхідних даних:

- Packet: 192.16.1.10 - 11000000.00010000.00000001.00001010 (в двійк.);
- Address: 192.168.10.10 - 11000000.10101000.00001010.00001010 (в двійк.);
- Mask: 255.255.0.0 – 11111111.11111111.00000000.00000000 (в двійк.).

В результаті тестування було отримано дані, які зображено на рисунку 3.5. Вихідні порти “Result” та “ResultPacket” виводять код IP адреси мережі, в яку прямує пакет, та IP адреси мережі шаблону. Далі ці коди порівнюються компаратором, порт “ResultCompare” вивів нуль, а це значить, що коди виявились різними. Далі цей логічний нуль надходить до мультиплексора, який забороняє передачу пакета даних на вихід.

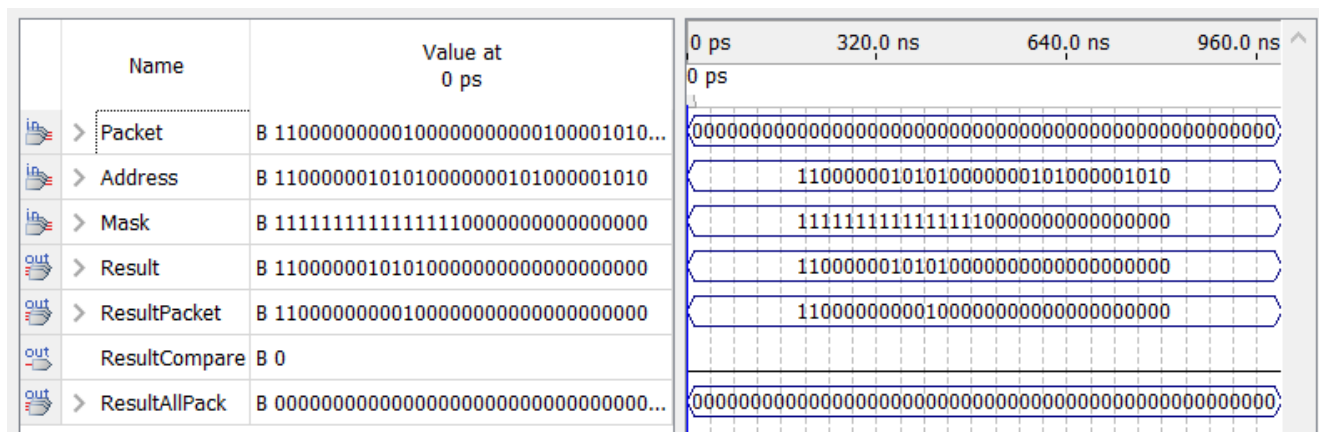


Рисунок 3.5 – Невдалий результат тестування схеми

Для кращої наочності протестуємо схему пристрою сортування частинами та з використанням додаткових виходів. Часткова схема, яку буде протестовано, зображена на рисунку 3.6. Ця схема містить логічні елементи “AND”, “OR”, компаратори та кодер.

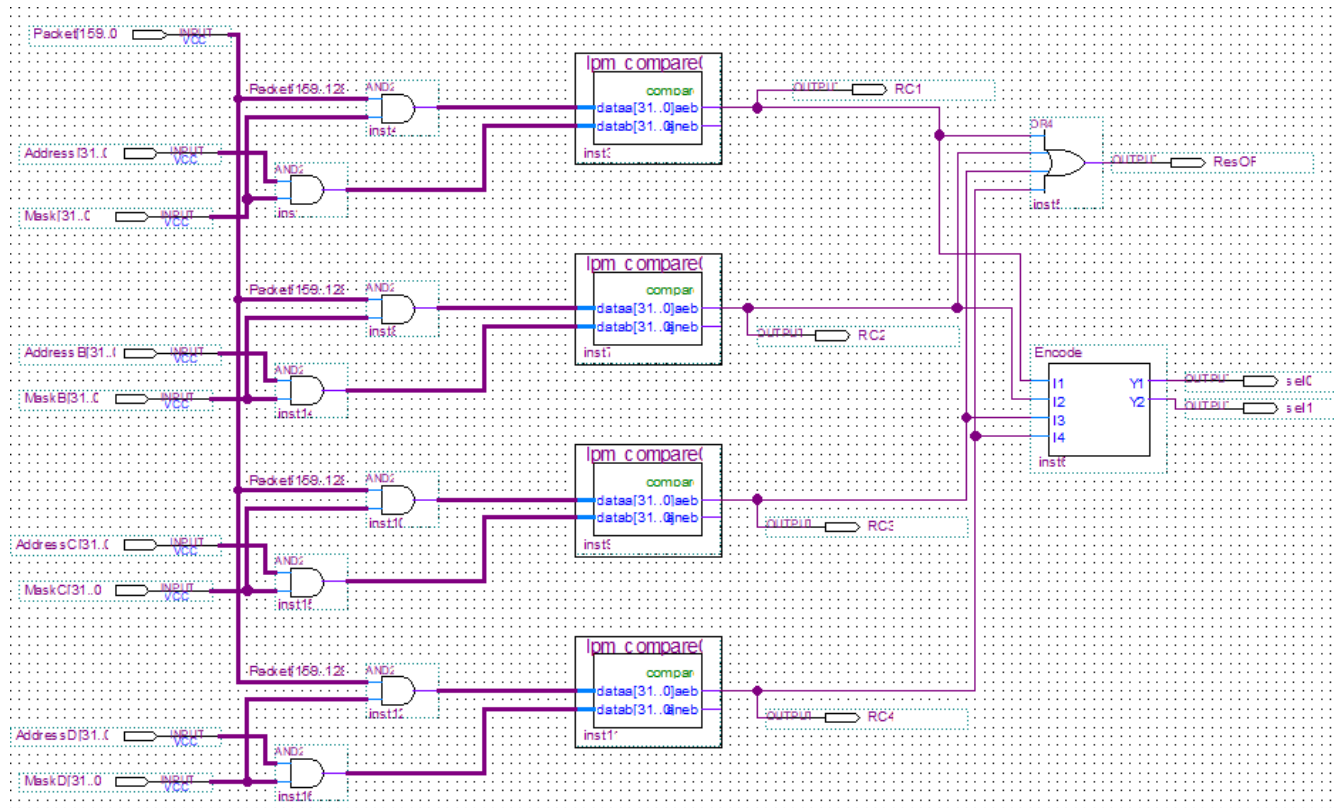


Рисунок 3.6 – Часткова схема пристрою сортування

Виконуючи тестування схеми було використано такі вхідні дані:

- Packet: 192.168.1.15 - 11000000.10101000.00000001.00001111 (в двійк.);
- Address: 192.168.128.128- 11000000.10101000.10000000.10000000 (в двійк.);
- Mask: 255.255.0.0 – 11111111.11111111.00000000.00000000 (в двійк.);
- AddressB: 200.66.68.1 - 11001000.01000010.01000100.00000001 (в двійк.);
- MaskB: 255.255.255.0 – 11111111.00000000.00000000.00000000 (в двійк.);
- AddressC: 172.168.3.1 - 10101100.10101000.00000011.00000001 (в двійк.);
- MaskC: 255.255.255.0 – 11111111.11111111.11111111.00000000 (в двійк.);
- AddressD: 192.16.1.1 - 11000000.00010000.00000001.00000001 (в двійк.);
- MaskD: 255.255.255.0 – 11111111.11111111.11111111.00000000 (в двійк.).

В результаті тестування було отримано дані, які зображено на рисунку 3.7. Вихідні порти RC1-RC4 призначені для перевірки виводу стану компаратора. Як видно з рисунка порт RC1 вивів одиницю, це означає, що дані які порівнювались виявились однаковими. На портах RC2-RC4 отримано нуль, виходить що результат порівняння негативний, внесені дані не співпадають. Далі ці значення надходять до кодера та логічного елемента “OR”. ResOr це вихідний порт який демонструє результат виконання “OR”. Вихідні порти sel0 та sel1 виводять результат обробки інформації яка надійшла з компараторів до кодера. Потім всі ці значення прямують до демультіплексора, який вирішує як і куди направити пакет даних.

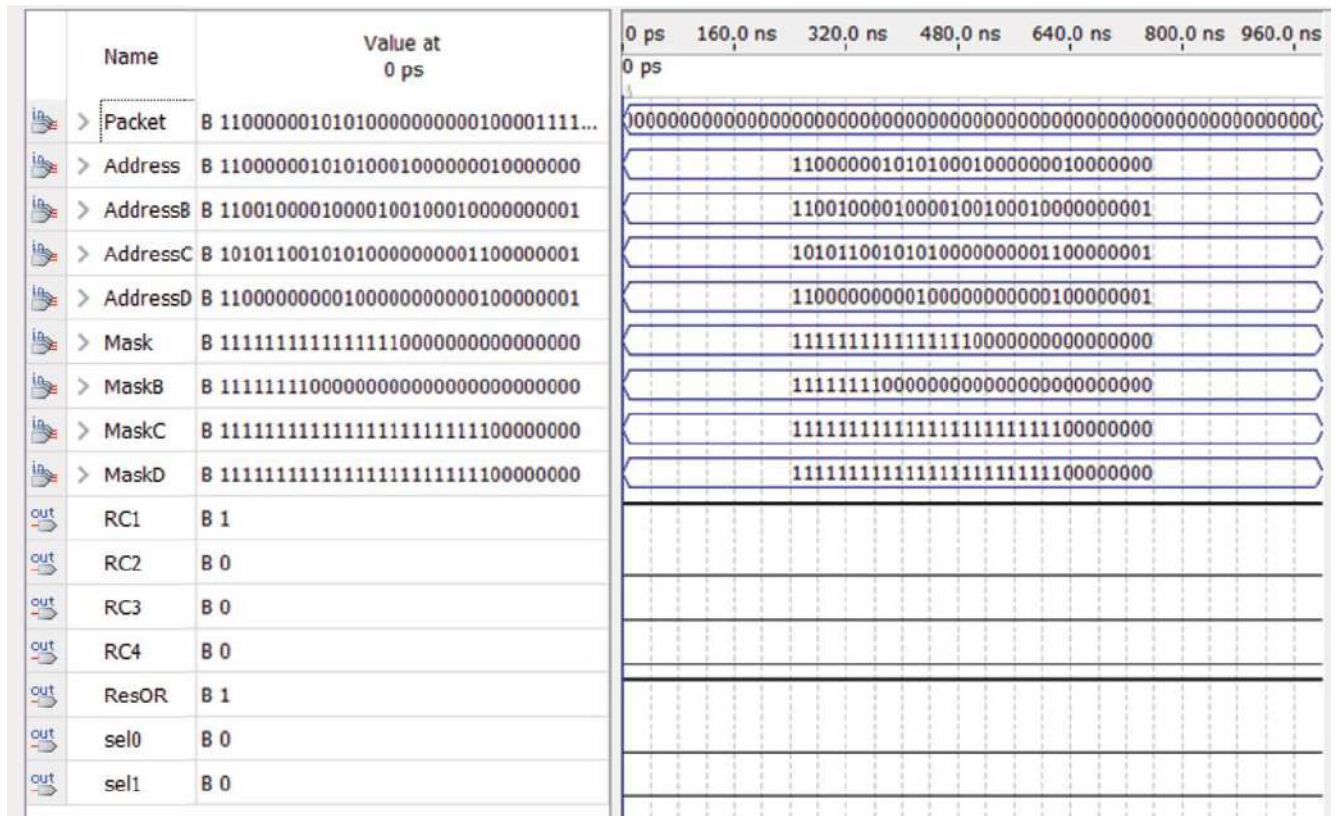


Рисунок 3.7 – Результат тестування схеми

Протестуємо другу частину схеми пристрою сортування з використанням додаткових виходів. До другої частини схеми відноситься демультіплексор та його входи та виходи. Друга частина схеми зображена на рисунку 3.8.

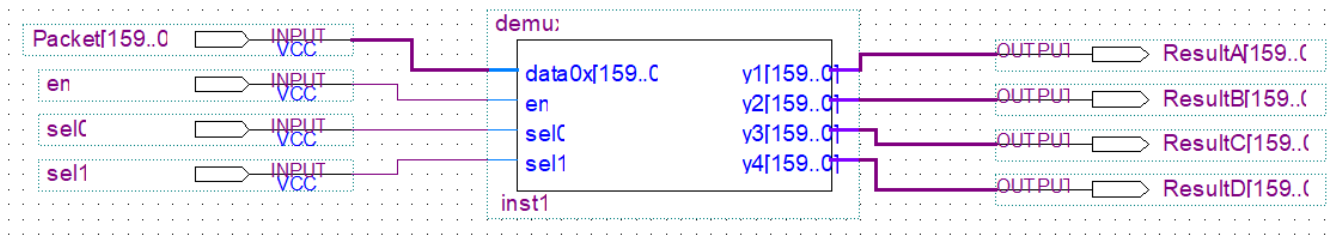


Рисунок 3.8 – Друга частина схеми пристрою сортування

Виконуючи тестування схеми було використано такі вхідні дані:

- Packet: 192.168.128.3 - 11000000.10101000.10000000.00000011 (в двійк.);
- En: 1;
- Sel0: 0;
- Sel1: 0.

В результаті тестування було отримано дані, які зображено на рисунку 3.9. Вхідний керуючий сигнал “en” має значення логічної одиниці, це значить, що демультимплексор отримає дозвіл на подальшу роботу. Вхідні сигнали вибору “sel0” та “sel1” мають значення логічних нулів. Демультимплексор приймає ці значення та на їх основі робить вибір куди саме спрямувати пакет даних. В даному випадку пакет даних було спрямовано на вихідний порт “ResultA”.

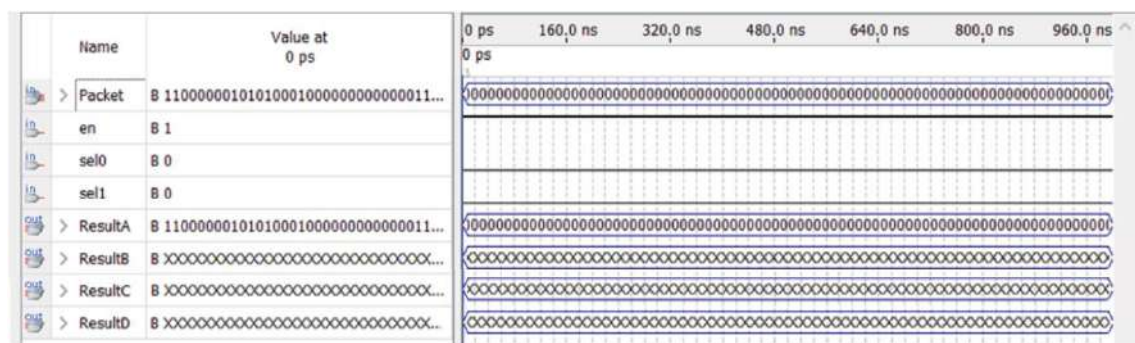


Рисунок 3.9 – Результат тестування схеми при sel0=sel1=0

Виконуючи тестування схеми було використано такі вхідні дані:

- Packet: 192.168.128.3 - 11000000.10101000.10000000.00000011 (в двійк.);
- En: 1;
- Sel0: 1;
- Sel1: 0.

В результаті тестування було отримано дані, які зображено на рисунку 3.10. Вхідний керуючий сигнал “en” має значення логічної одиниці, це значить, що демультиплексор отримає дозвіл на подальшу роботу. Вхідні сигнали вибору “sel0” має значення одиниці, а “sel1” має значення нуля. Демультиплексор приймає ці значення та на їх основі робить вибір куди саме спрямувати пакет даних. В даному випадку пакет даних було спрямовано на вихідний порт “ResultC”.

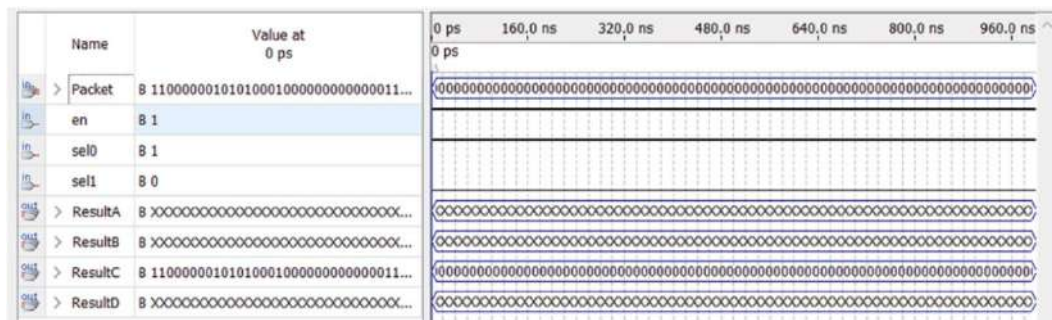


Рисунок 3.10 – Результат тестування схеми при sel0=1 та sel1=0

Для більш детального аналізу працездатності схеми, було проведено тестування пристрою в цілому на різних типах вхідних даних. Розглянемо результати тестування повної схеми, які зображено на рисунку 3.11.

Виконуючи тестування схеми було використано такі вхідні дані:

- Packet: 192.168.1.15 - 11000000.10101000.00000001.00001111 (в двійк.);
- Address: 192.168.128.128- 11000000.10101000.10000000.10000000 (в двійк.);
- Mask: 255.255.0.0 – 11111111.11111111.00000000.00000000 (в двійк.);
- AddressB: 200.66.68.1 - 11001000.01000010.01000100.00000001 (в двійк.);
- MaskB: 255.255.255.0 – 11111111.00000000.00000000.00000000 (в двійк.);
- AddressC: 172.168.3.1 - 10101100.10101000.00000011.00000001 (в двійк.);
- MaskC: 255.255.255.0 – 11111111.11111111.11111111.00000000 (в двійк.);
- AddressD: 192.16.1.1 - 11000000.00010000.00000001.00000001 (в двійк.);
- MaskD: 255.255.255.0 – 11111111.11111111.11111111.00000000 (в двійк.).

Вихідні порти ResultA-D призначено для виводу цілого пакета даних за результатами фільтрації. В цьому випадку пакет було передано до вихідного порту ResultA, всі інші залишились незалучені. Це сталося через те, що пристрій дізнався

куди прямує пакет, порівняв це значення з усіма шаблонами та знайшов збіги завдяки яким пакет даних було направлено на відповідний вихідний порт.

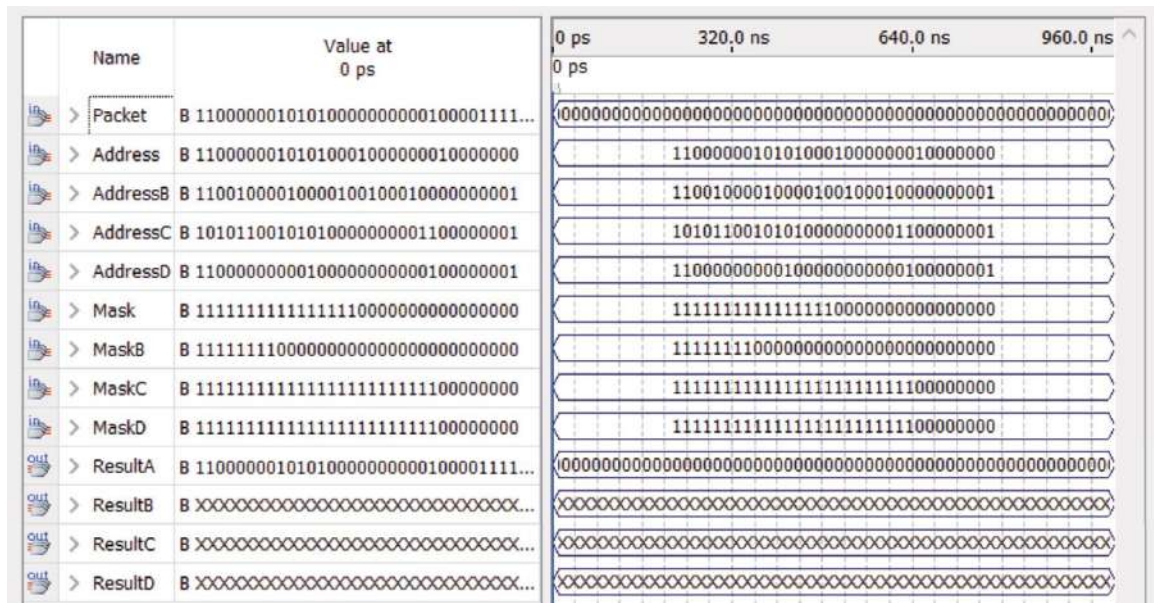


Рисунок 3.11 – Результат тестування схеми з виводом в порт ResultA

Розглянемо результати тестування Результат тестування схеми з виводом в порт ResultA пристрою з іншими параметрами (рис 3.12-15).

За результатом роботи пристрою, пакет було передано до вихідного порту ResultB, всі інші залишились незадіяні.

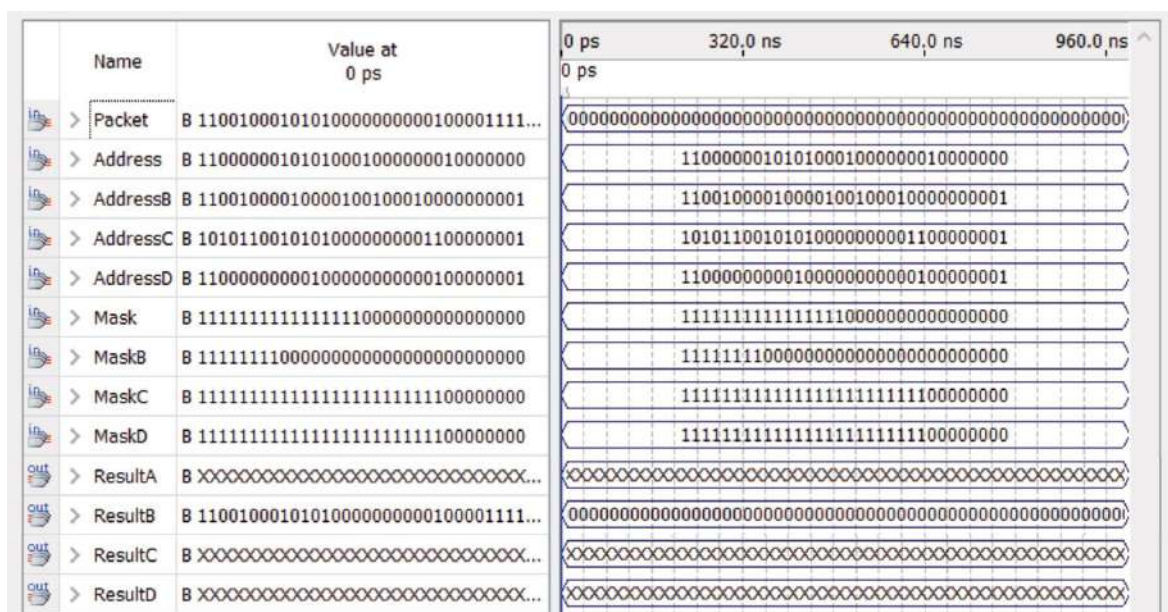


Рисунок 3.12 – Результат тестування схеми з виводом в порт ResultB

За результатом роботи пристрою, пакет було передано до вихідного порту ResultC, всі інші залишились незалучені.

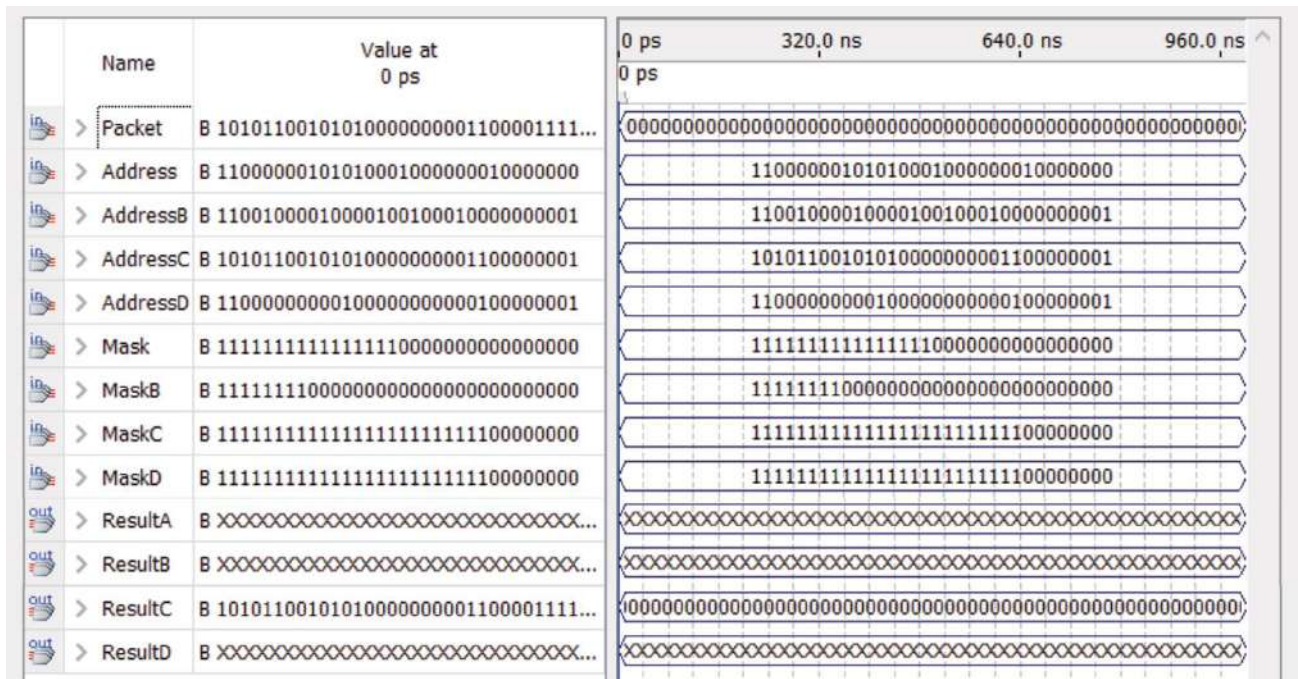


Рисунок 3.13 – Результат тестування схеми з виводом в порт ResultC

За результатом роботи пристрою, пакет було передано до вихідного порту ResultD, всі інші залишились незадіяні.

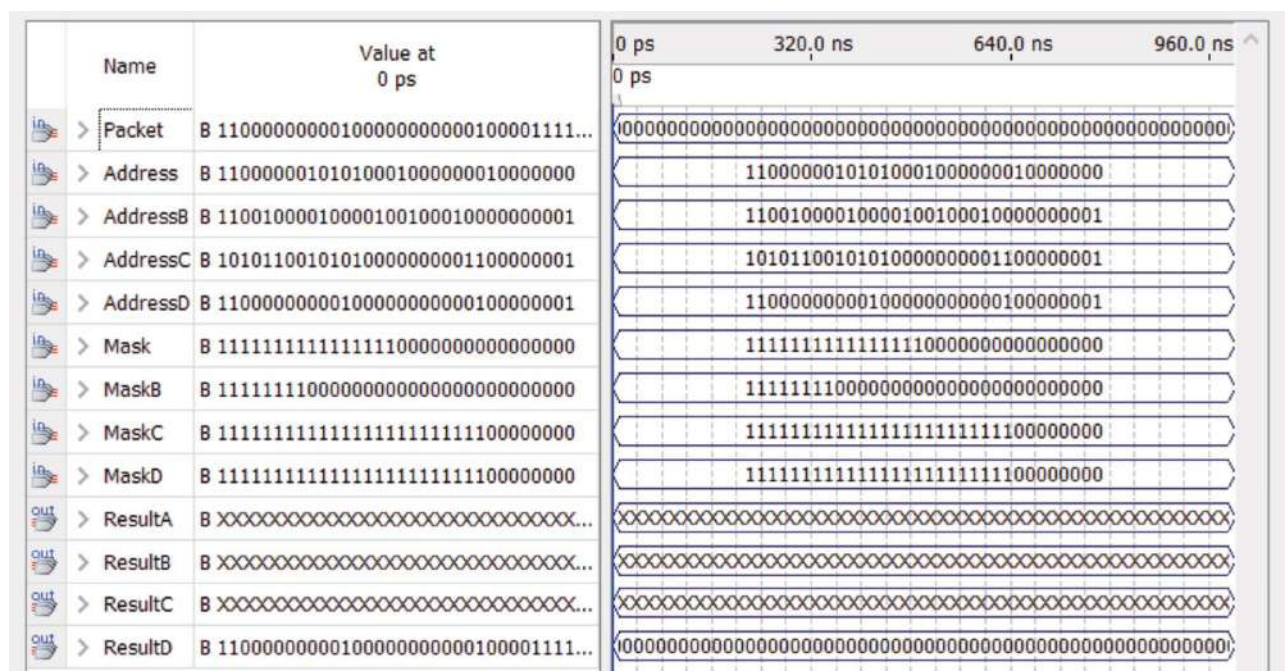


Рисунок 3.14 – Результат тестування схеми з виводом в порт ResultD

При використанні таких вхідних даних, пакет не буде передано на жоден вихід ResultA-D, через те, що пакет даних не задовольняє жодної з умов фільтрації.

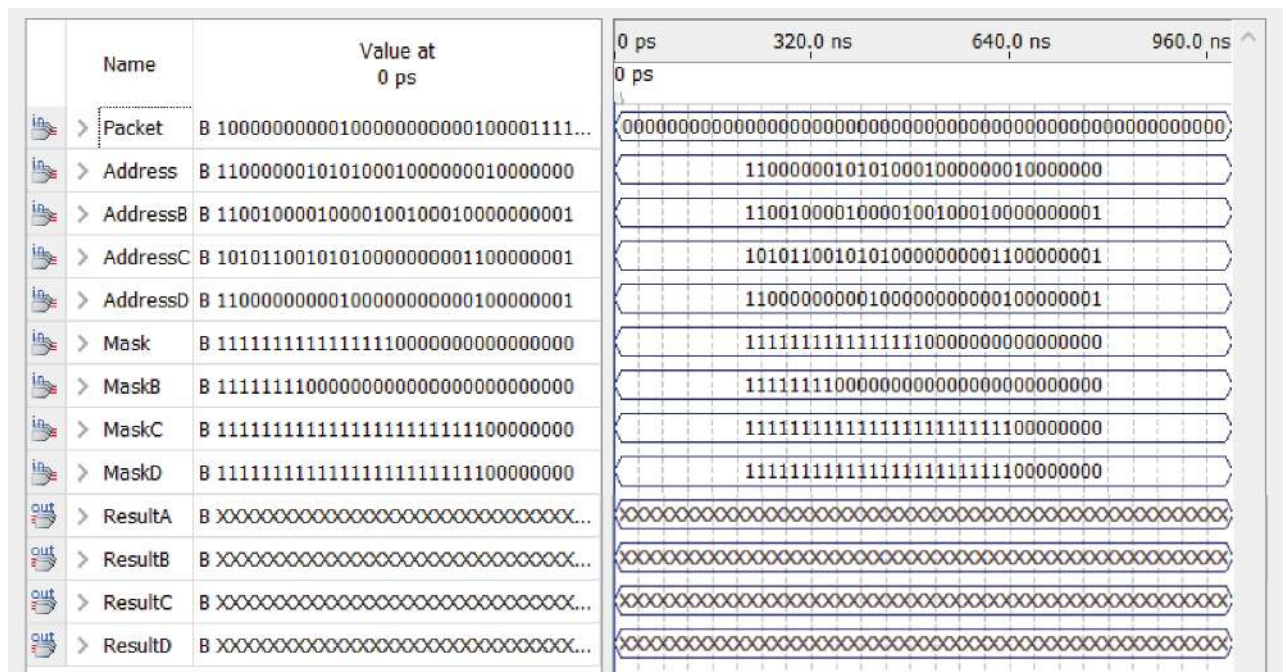


Рисунок 3.15 – Результат тестування схеми з відкиданням пакета

Результати тестування пристрою керування для фільтрації мережевих пакетів за маскою демонструють, що схема працює у відповідності з поставленою задачею. Виходи схеми підтверджують, що пристрій забезпечує фільтрацію мережевих пакетів на основі наданих шаблонів, пропускаючи лише дозволені пакети та відкидаючи несанкціоновані.

Розглянемо використані ресурси пристрою фільтрації та сортування (топологію, звіт імплементації).

Топографія мікросхеми – це зафіксоване на матеріальному носії просторово-геометричне розташування сукупності елементів інтегральної мікросхеми та зв'язків між ними.

Звіт імплементації з'являється після компіляції проекту (рис. 3.16).

Flow Summary		Flow Summary	
Flow Status	Successful - Sat Jun 01 08:36:43 2024	Flow Status	Successful - Sat Jun 01 08:44:46 2024
Quartus II 64-Bit Version	13.1.0 Build 162 10/23/2013 SJ Web Edition	Quartus II 64-Bit Version	13.1.0 Build 162 10/23/2013 SJ Web Edition
Revision Name	Filter	Revision Name	Filter
Top-level Entity Name	Filter	Top-level Entity Name	Filter
Family	Cyclone IV GX	Family	Cyclone IV GX
Total logic elements	245 / 149,760 (< 1 %)	Total logic elements	155 / 109,424 (< 1 %)
Total combinational functions	245 / 149,760 (< 1 %)	Total combinational functions	155 / 109,424 (< 1 %)
Dedicated logic registers	0 / 149,760 (0 %)	Dedicated logic registers	0 / 109,424 (0 %)
Total registers	0	Total registers	0
Total pins	449 / 508 (88 %)	Total pins	423 / 508 (83 %)
Total virtual pins	0	Total virtual pins	0
Total memory bits	0 / 6,635,520 (0 %)	Total memory bits	0 / 5,621,760 (0 %)
Embedded Multiplier 9-bit elements	0 / 720 (0 %)	Embedded Multiplier 9-bit elements	0 / 560 (0 %)
Total GXB Receiver Channel PCS	0 / 8 (0 %)	Total GXB Receiver Channel PCS	0 / 8 (0 %)
Total GXB Receiver Channel PMA	0 / 8 (0 %)	Total GXB Receiver Channel PMA	0 / 8 (0 %)
Total GXB Transmitter Channel PCS	0 / 8 (0 %)	Total GXB Transmitter Channel PCS	0 / 8 (0 %)
Total GXB Transmitter Channel PMA	0 / 8 (0 %)	Total GXB Transmitter Channel PMA	0 / 8 (0 %)
Total PLLs	0 / 8 (0 %)	Total PLLs	0 / 8 (0 %)
Device	EP4CGX150DF31C8	Device	EP4CGX110DF31C7
Timing Models	Final	Timing Models	Final

Рисунок 3.16 – Звіт імплементації для пристрою а) фільтрації (зліва) та б) сортування (справа)

Топологія мікросхеми наведена на рисунку 3.17.

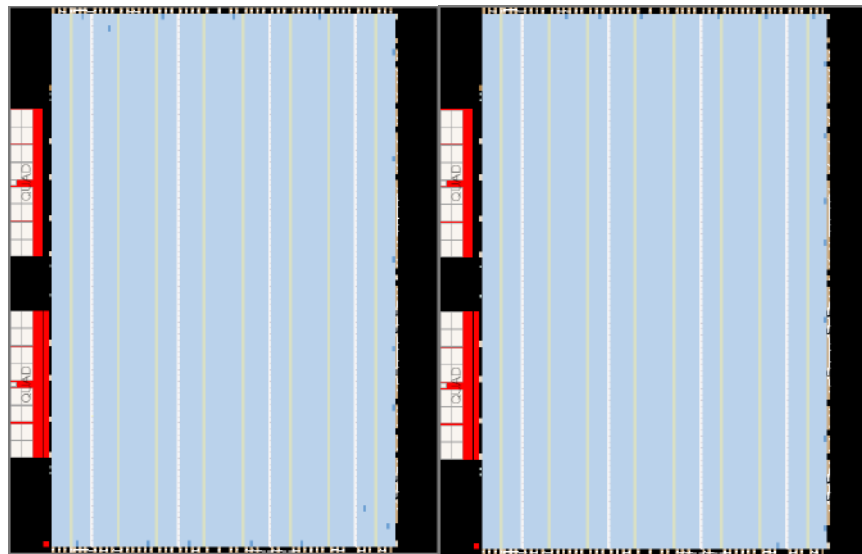


Рисунок 3.17 – Топологія мікросхеми для пристрою а) фільтрації (зліва) та б) сортування (справа)

Таким чином, видно, що розроблений пристрій сортування на кристалі займає менше 1% доступного простору.

За результатами звіту з імплементації та візуалізації топології мікросхеми видно, що даного кристалу буде достатньо для реалізації пристрою сортування для підмережі значних розмірів. Це надає можливість сортувати пакети паралельно за менший проміжок часу, що є перспективним при його використанні.

ВИСНОВКИ

Результати, отримані в рамках цього проєкту, продемонстрували ефективність та результативність апаратного підходу до фільтрації мережевих пакетів. Завдяки програмному забезпеченню Quartus II та мові VHDL стане можливим створення пристрою керування, який забезпечить надійну та своєчасну фільтрацію мережевих пакетів на основі заданих критеріїв, таких як маски.

Відповідність результатів робочому завданню та вимогам підтверджує надійність та практичність розробки пристрою керування. Потенційні області застосування цієї технології дуже широкі - від підвищення мережевої безпеки в різних галузях промисловості до вдосконалення заходів захисту даних в системах критичної інфраструктури.

Крім того, можливість впровадження отриманих результатів у реальних сценаріях є цілком реальною. Апаратна природа контрольного пристрою гарантує, що він може бути легко інтегрований в існуючі мережеві системи, забезпечуючи реальне рішення для посилення захисту даних і мережевої безпеки.

Таким чином, розробка пристрою керування для фільтрації мережевих пакетів за маскою з використанням комбінації передових технологій та методологій виявилася значним досягненням. Отримані результати відповідають початковим цілям і вимогам проєкту, а потенціал для реалізації та застосування цих результатів у практичних сценаріях є багатообіцяючим. Цей проєкт є цінним внеском у сферу мережевої безпеки та захисту даних, що має далекосяжні наслідки для майбутнього глобальних мереж.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

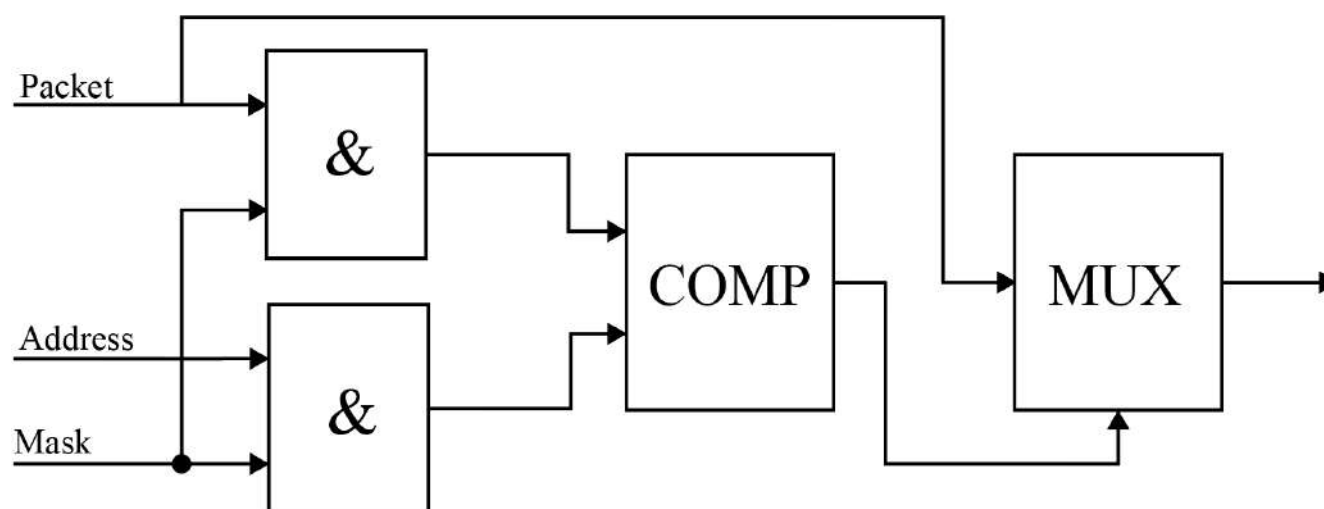
1. Wright G. What is packet filtering? – TechTarget Definition. Networking. URL: <https://www.techtarget.com/searchnetworking/definition/packet-filtering> (дата звернення: 26.04.2024).
2. Understanding netmask: a comprehensive guide to network subnetting | teltonika. Teltonika Networks. URL: <https://teltonika-networks.com/newsroom/understanding-netmask-a-comprehensive-guide-to-network-subnetting> (дата звернення: 27.04.2024).
3. Types of firewalls defined and explained. Palo Alto Networks. URL: <https://www.paloaltonetworks.co.uk/cyberpedia/types-of-firewalls> (дата звернення: 02.05.2024).
4. Firewall types and selection. hardware vs software - zenarmor.com. Zenarmor - Agile Service Edge Security. URL: <https://www.zenarmor.com/docs/network-security-tutorials/hardware-firewall-vs-software-firewall> (дата звернення: 01.05.2024).
5. Розділ 8 Пристрій керування. StudFiles. URL: <https://studfile.net/preview/3907471/page:108/> (дата звернення: 24.04.2024).
6. What is an FPGA? Definition, types, programming, and more. Data Center Knowledge | News and analysis for the data center industry. URL: <https://www.datacenterknowledge.com/hardware/what-fpga-definition-types-programming-and-more#close-modal> (дата звернення: 26.04.2024).
7. Xilinx vs intel high-end FPGA series comparison. FPGA Insights. URL: https://fpgainsights.com/fpga/xilinx-vs-intel-high-end-fpga-series-comparison/#Comparison_of_Xilinx_and_Intel_High-End_FPGA_Series (дата звернення: 27.04.2024).
8. Yasar K., Zola A. What are network packets and how do they work?. Networking. URL: <https://www.techtarget.com/searchnetworking/definition/packet> (дата звернення: 27.05.2024).

9. Що таке IP-адреса і для чого вона потрібна? – Trium. Купити бу комп'ютери в Україні – магазин комп'ютерної техніки Trium | Продаж БУ Комп'ютерів. URL: <https://trium.com.ua/43-shcho-take-ip-adresa-i-dlia-choho-vona-potribna> (дата звернення: 27.05.2024).

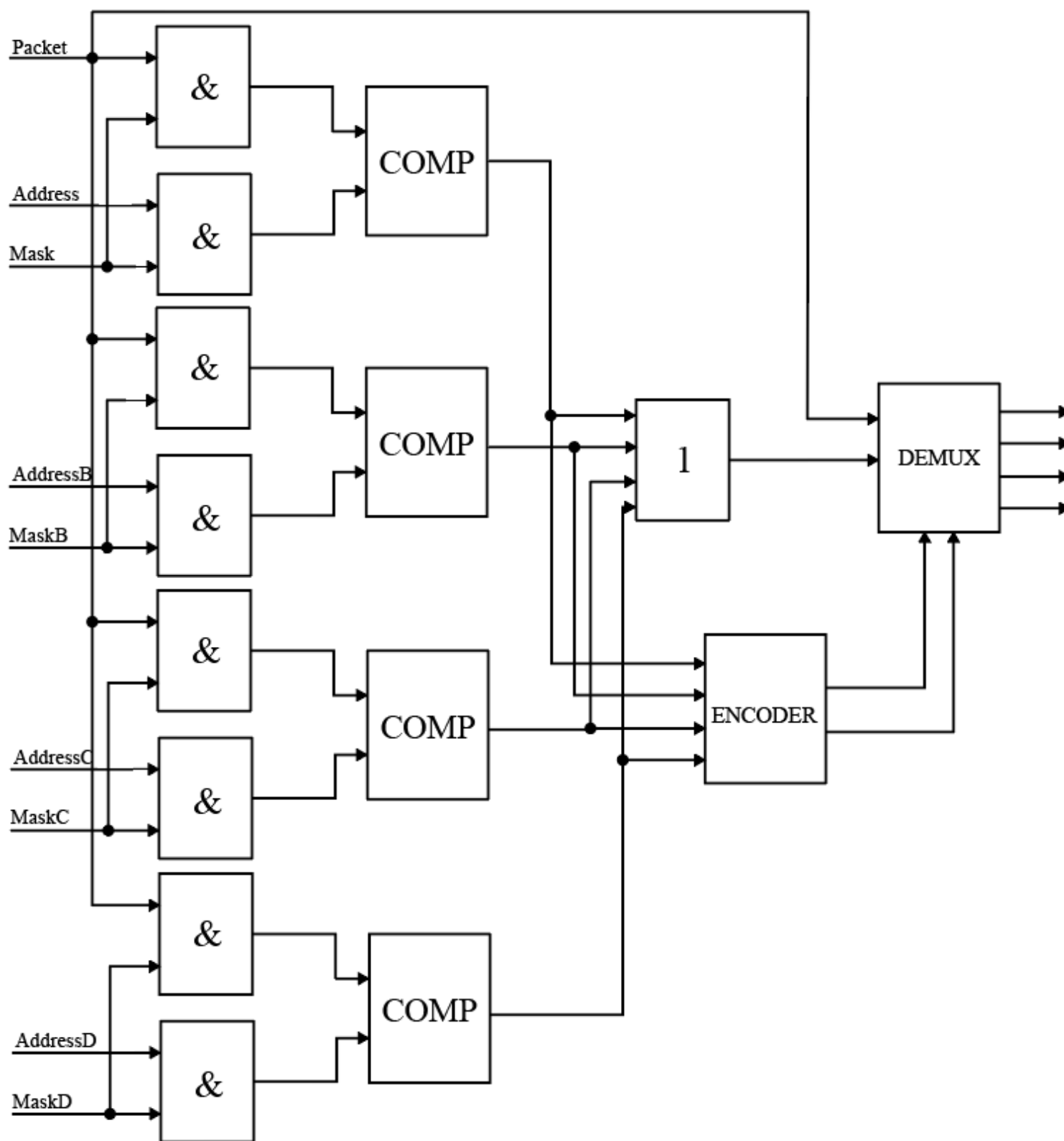
10. What is subnet mask? Definition & faqs | avi networks. Avi Networks. URL: <https://avinetworks.com/glossary/subnet-mask/#:~:text=A%20subnet%20mask%20is%20a,the%20network%20and%20host%20addresses.> (дата звернення: 27.05.2024).

11. URL: https://www.electronics-tutorials.ws/combination/comb_3.html (дата звернення: 27.05.2024).

Структурна схема пристроїв



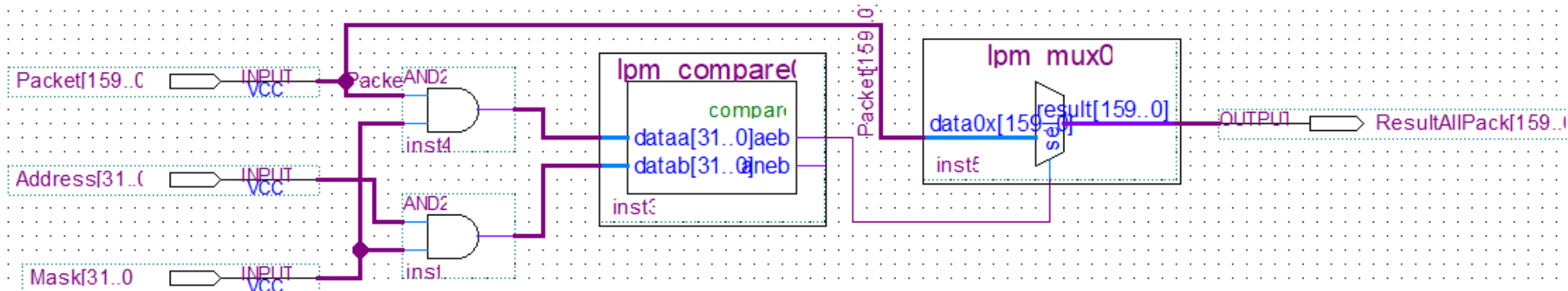
Структурна схема простого фільтра пакетів



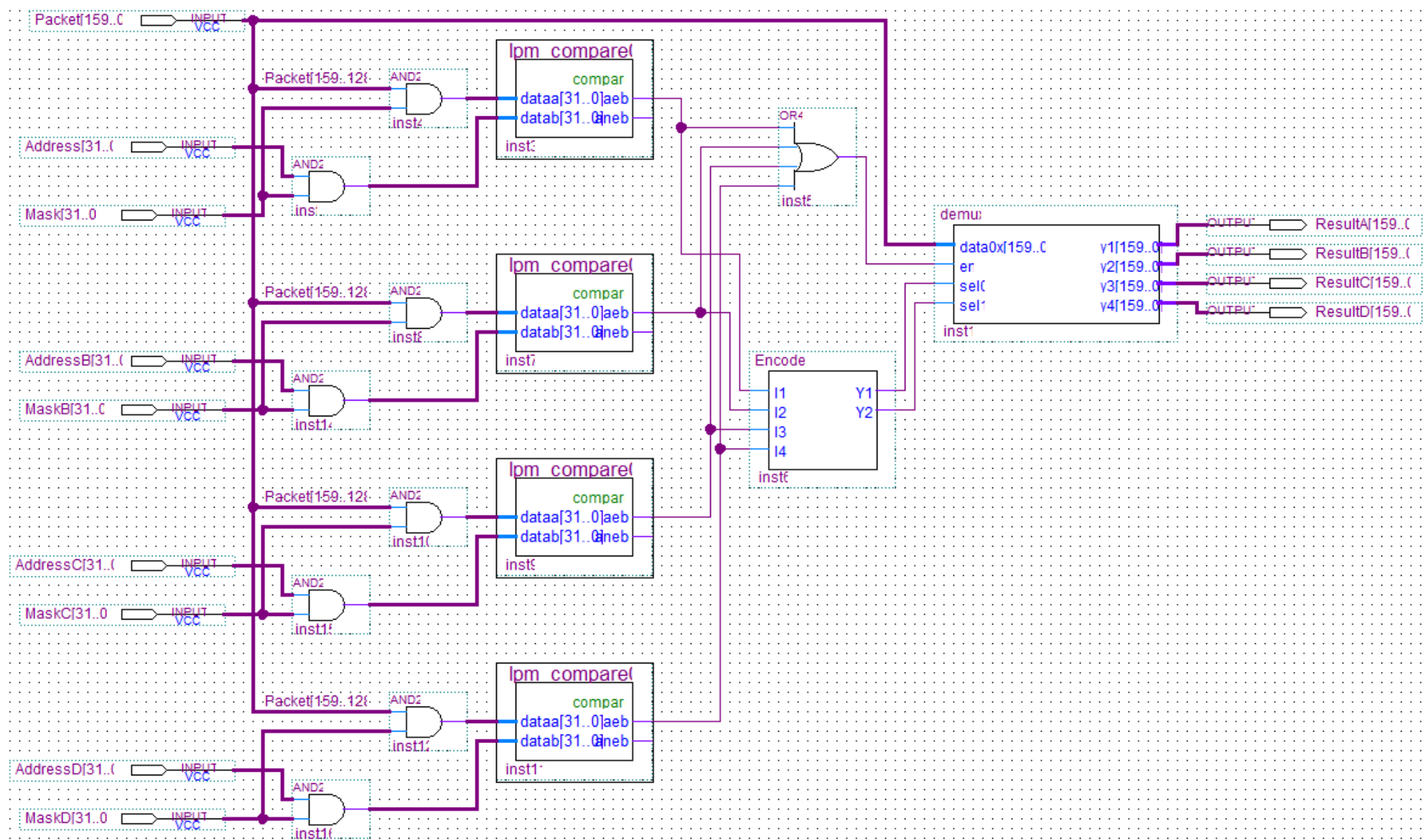
Структурна схема пристрою сортування пакетів

					13.02070849.00009 ПЛІ				
					Розробка пристрою керування для фільтрації мережеских пакетів по масці	Лист.		Маса	Масшт.
Зм.	Лист.	№ докум.	Підпис	Дата					
Розроб.		Пімшин Б.О.							
Перевір.		Голуб Т.В.							
Т.контр.						Лист.		Листів	
						НУ «Запорізька політехніка»			
Н.контр.		Щербак Н.В.				КНТ-510			
Затвердж.		Кудерметов Р.К.							

Апаратна схема пристроїв



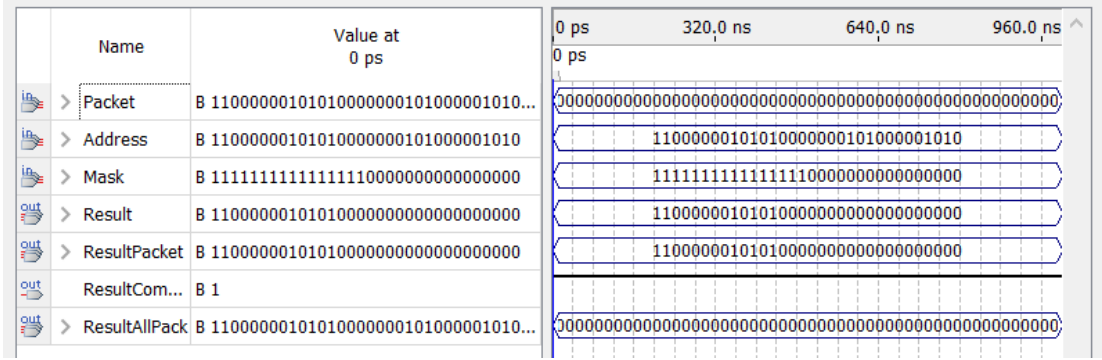
Структурна схема пристрою фільтрації



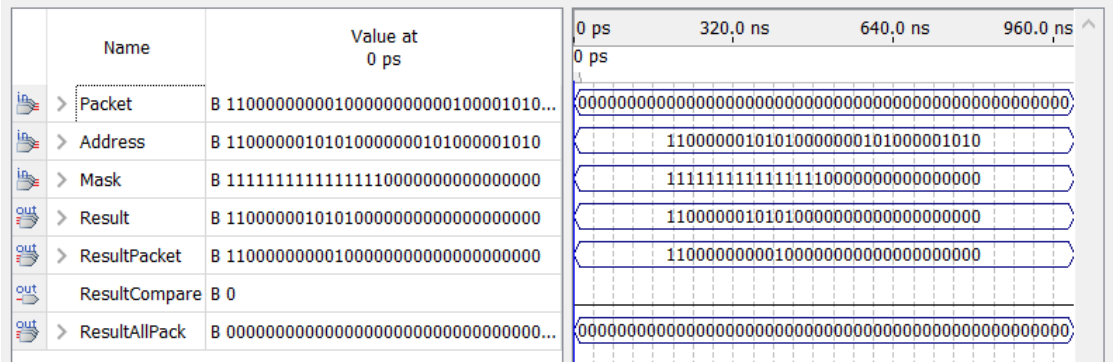
Структурна схема пристрою сортування

					13.02070849.00009 ПЛ2						
					Розробка пристрою керування для фільтрації мережесих пакетів по масці	Лист.			Маса	Масшт.	
Зм.	Лист.	№ докум.	Підпис	Дата							
Розроб.		Пімшин Б.О.									
Перевір.		Голуб Т.В.									
Т.контр.					НУ «Запорізька політехніка» KHT-510	Лист.			Листів		
Н.контр.		Щербак Н.В.									
Затвердж.		Кудерметов Р.К.									

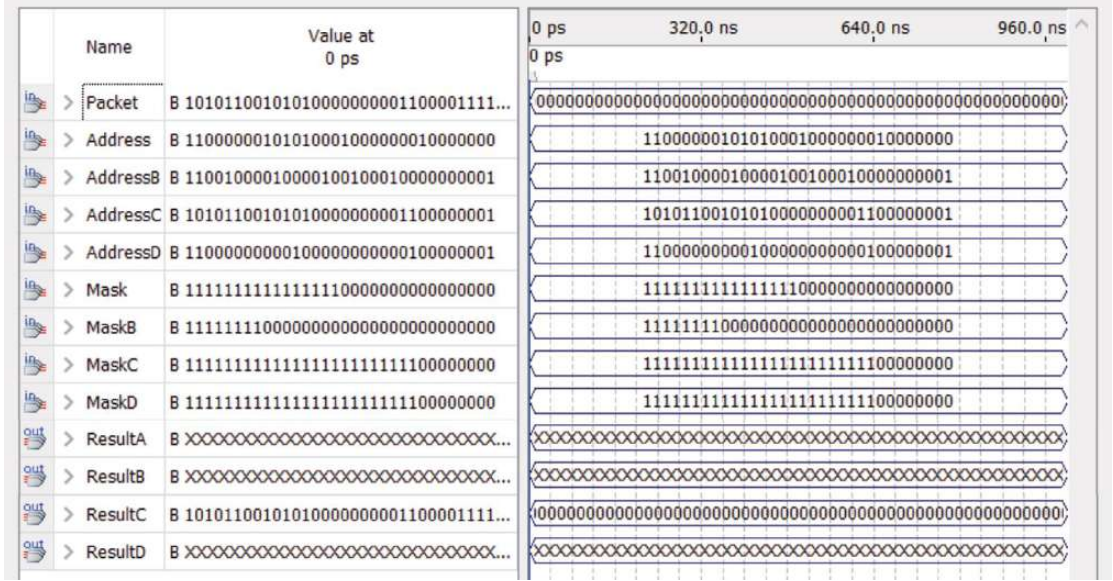
Результати тестування схем



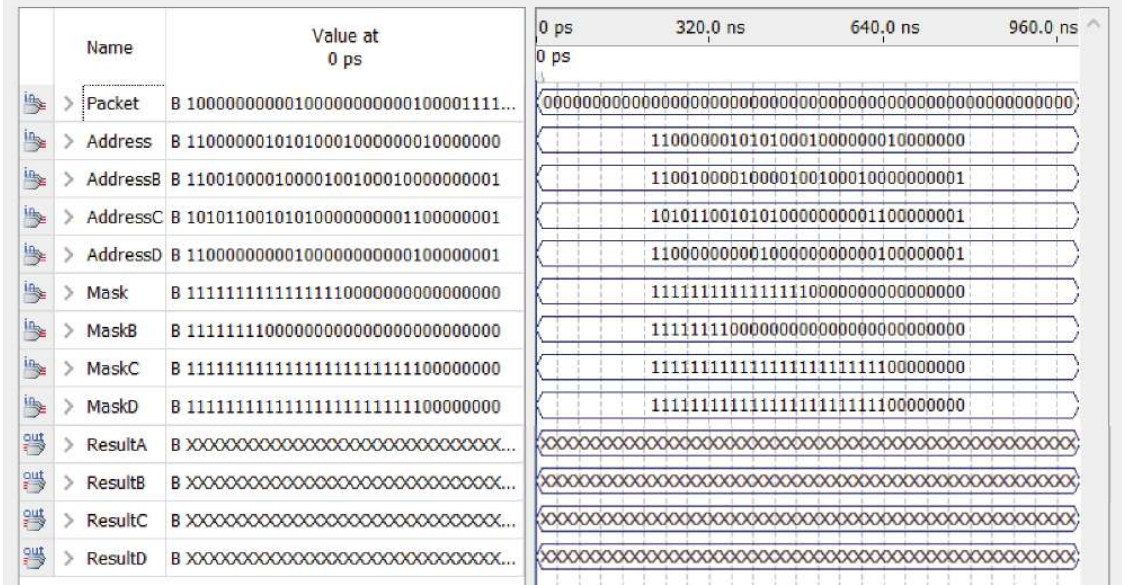
Успішний результат тестування схеми



Невдалий результат тестування схеми

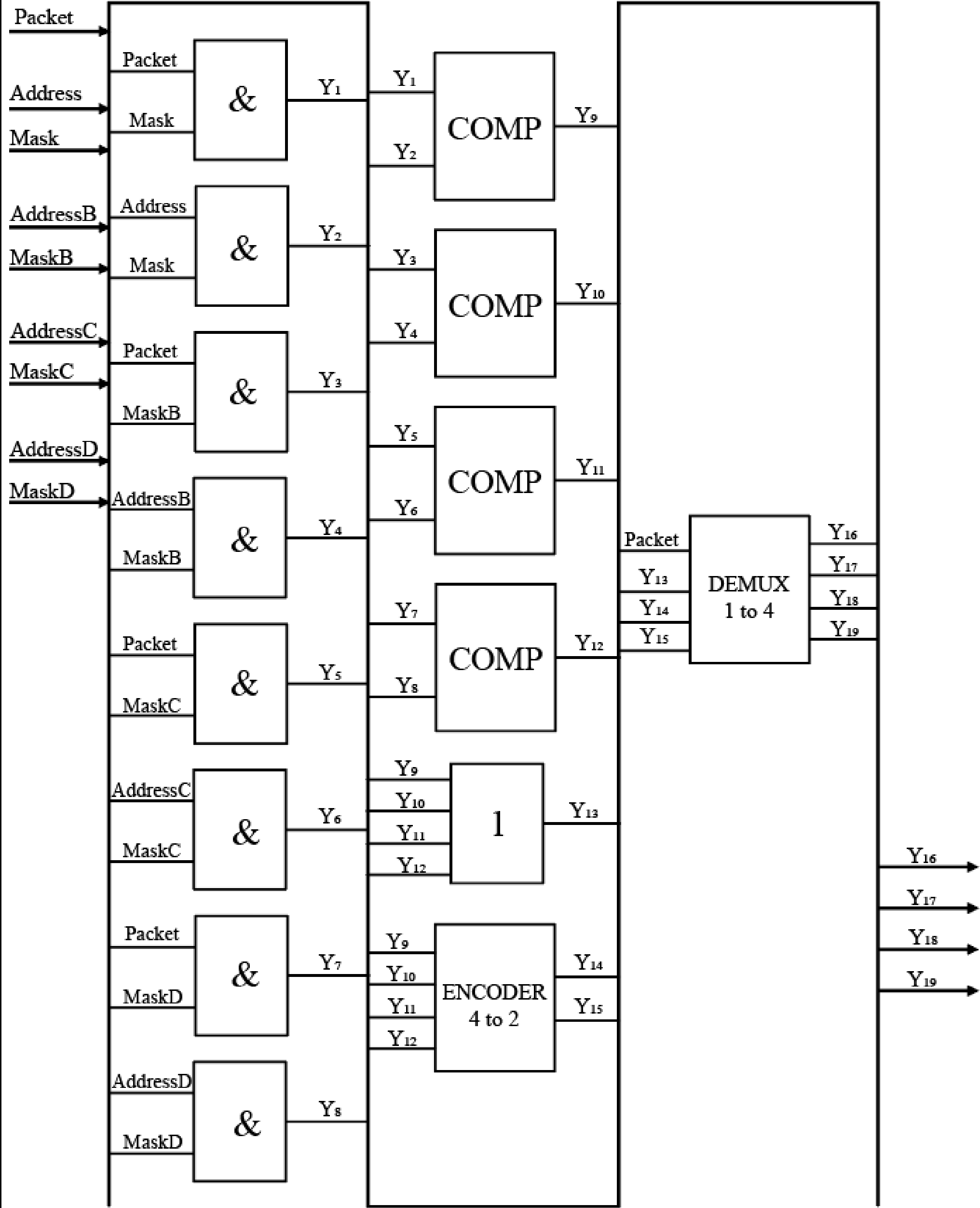


Результат тестування схеми з виводом в порт ResultC



Результат тестування схеми з відкиданням пакета

					13.02070849.00009 ПЛЗ						
					Розробка пристрою керування для фільтрації мережевих пакетів по масці	Лист.		Маса		Масшт.	
						Лист.		Листів			
						НУ «Запорізька політехніка» КНТ-510					
Зм.	Лист.	№ докум.	Підпис	Дата							
Розроб.		Пімшин Б.О.									
Перевір.		Голуб Т.В.									
Т.контр.											
Н.контр.		Щербак Н.В.									
Затвердж.		Кудерметов Р.К.									



					13.02070849.00009 E3							
					Пристрій керування для фільтрації мережесих пакетів по масці Схема електрична принципова	Лист.			Маса		Масшт.	
Зм.	Лист.	№ докум.	Підпис	Дата								
Розроб.		Пімшин Б.О.										
Перевір.		Голуб Т.В.										
Т.контр.												
						Лист. 1			Листів 1			
Н.контр.		Щербак Н.В.				НУ «Запорізька політехніка» KHT-510						
Затвердж.		Кудерметов Р.К.										