ДОДАТОК А

Графічний матеріал кваліфікаційної роботи

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

КАФЕДРА ЕЛЕКТРОННИХ ОБЧИСЛЮВАЛЬНИХ МАШИН

КВАЛІФІКАЦІЙНА РОБОТА

«Пристрій цифрової обробки даних з використанням VHDL»

Студент гр. КІУКІ-21-1 **Островерх І.Д.**

Керівник **ас. Дяченко Д.О.**

Харків 2025

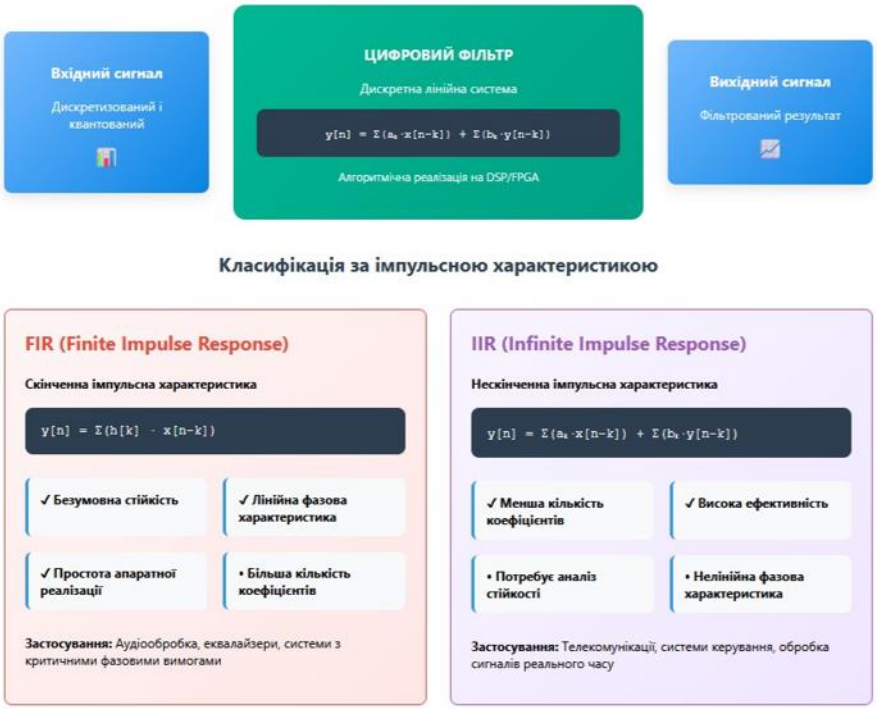
Мета та завдання кваліфікаційної роботи

Мета кваліфікаційної роботи: розробка пристрою, здатного здійснювати обробку аудіосигналів у реальному часі, а також забезпечення можливості регулювання посилення кожної частотної смуги відповідно до потреб користувача.

Завдання:

- теоретичний аналіз частотних характеристик фільтрів;
- розробка VHDL-коду окремих цифрових фільтрів (FIR);
- симуляція роботи еквалайзера у програмному середовищі;
- програмна реалізація;
- моделювання пристрою з даними.

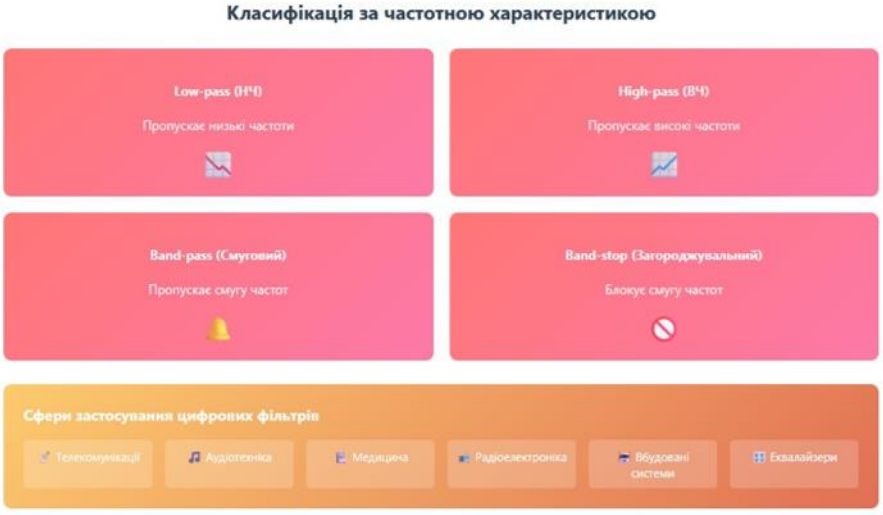
Цифрові фільтри



3

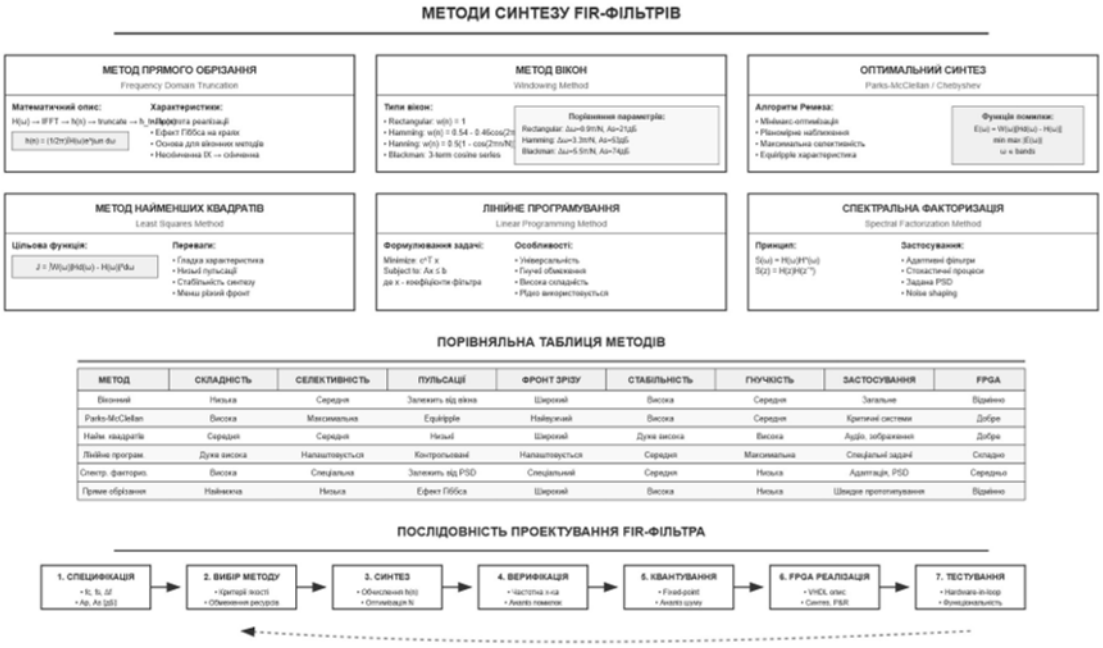
Класифікація та сфери застосування

Характеристика	FIR фільтри	IIR фільтри
Стійкість	Завжди стійкі	Потребують перевірки стійкості
Фазова характеристика	Лінійна фаза	Нелінійна фаза
Кількість коефіцієнтів	Більша	Менша
Обчислювальна складність	Вища	Нижча
Апаратна реалізація	Простіша	Складніша



4

Методи синтезу цифрових фільтрів



5

Огляд сучасних інструментів для проєктування фільтрів на VHDL

6

Основні методи проєктування

Метод віконних функцій

Принцип: Апроксимація ідеального фільтра за допомогою віконної функції

1. **Визначення ідеальної характеристики**
   Наприклад, для НЧ-фільтра: функція sinc
2. **Обрізання до скінченної довжини**
   Вибір порядку N фільтра
3. **Застосування віконної функції**
   Зменшення ефекту побічних лобів

```
h[n] = h_ideal[n] * w[n]
```

Типи віконних функцій:

- Прямокутне
- Гаусове
- Хеммінга
- Блекмана
- Кайзера
- Ханна

Переваги

- Простота реалізації
- Швидкість розрахунку
- Гарантована стійкість
- Лінійна фаза

Недоліки

- Обмежений контроль параметрів
- Компроміс між різкістю та флуктуаціями
- Не оптимальні характеристики

Метод частотної вибірковості та оптимізації

Принцип: Безпосереднє визначення коефіцієнтів на основі бажаної частотної характеристики

1. **Задання бажаної характеристики**
   Табличний набір амплітуд на різних частотах
2. **Вибір критерію оптимізації**
   Мінімізація максимальної похибки (Чебишев)
3. **Ітераційний пошук коефіцієнтів**
   Алгоритм Ремеза (Паркса-Макклеллана)

```
min max |H(ω) - H_desired(ω)|
```

Алгоритм Паркса-Макклеллана:

Особливості:

- Мінімальна максимальна похибка (minimax)
- Рівномірний розподіл помилки
- Точний контроль параметрів фільтра
- Оптимальне використання коефіцієнтів

Переваги

- Точний контроль характеристик
- Оптимальні параметри
- Мінімальний порядок фільтра
- Контроль пульсацій

Недоліки

- Складність реалізації
- Більший час розрахунку
- Потребує спеціалізованих алгоритмів

7

Критерії методу проєктування. Застосування в еквалайзерах

Характеристика	Віконні функції	Паркс-Макклеллан	Сучасні методи
Складність реалізації	Низька	Середня	Висока
Час синтезу	Дуже швидкий	Швидкий	Повільний
Контроль параметрів	Обмежений	Точний	Дуже точний
Оптимальність	Субоптимальна	Оптимальна (minimax)	Глобально оптимальна
Порядок фільтра	Вищий	Мінімальний	Дуже мінімальний
Застосування	Загальне	Професійне	Спеціалізоване

Критерії вибору методу проєктування

Точність характеристики
Вимоги до точності амплітудної та фазової характеристики фільтра

Складність реалізації
Обчислювальні ресурси та час, доступні для синтезу

Апаратні обмеження
Пам'ять, швидкодія та енергоспоживання цільової платформи

Якість сигналу
Вимоги до збереження якості аудіосигналу та фазової лінійності

Застосування в аудіоеквалайзерах

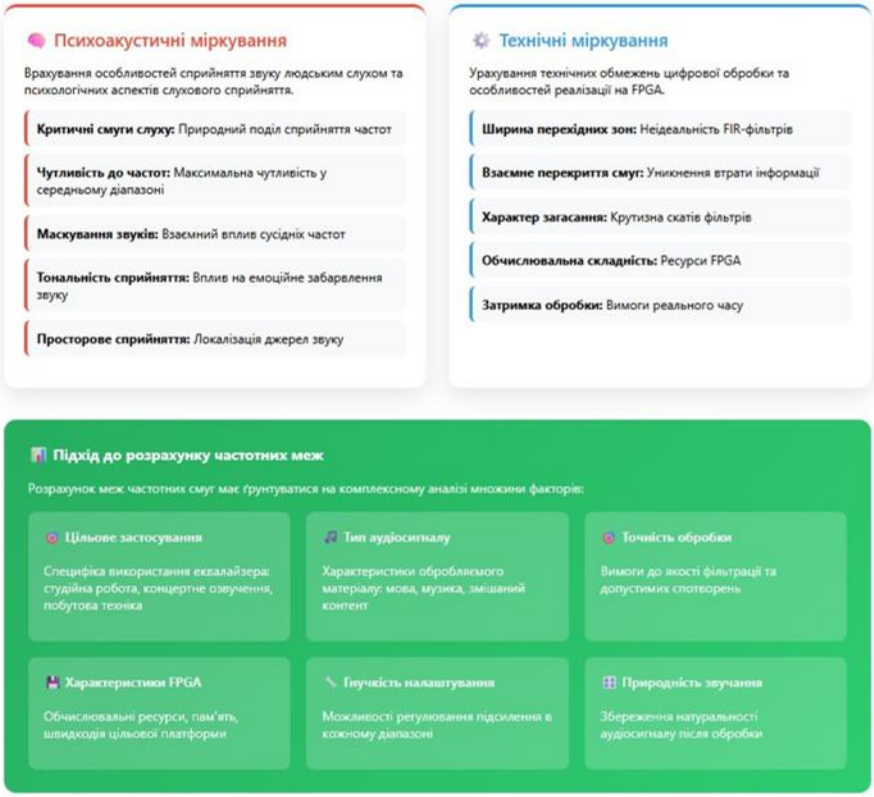
У практиці розробки аудіоеквалайзерів пріоритетом є збереження якості сигналу. Тому зазвичай перевага надається **методу віконних функцій із симетричними коефіцієнтами**, що гарантують:

- **Лінійну фазу** — Відсутність фазових спотворень
- **Стабільність** — Гарантована стійкість системи
- **Ефективність** — Швидкість обробки в реальному часі
- **Простота** — Легкість налаштування та реалізації

Для професійних застосувань з особливими вимогами до точності може використовуватися метод Паркса-Макклеллана або сучасні оптимізаційні підходи.

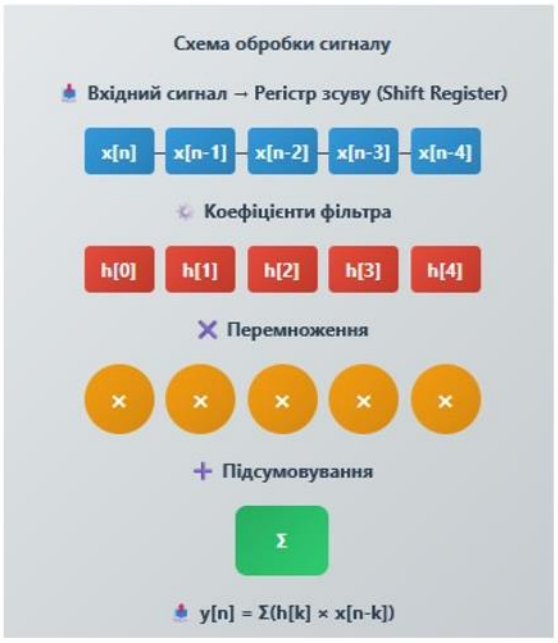
8

Визначення частотних меж



9

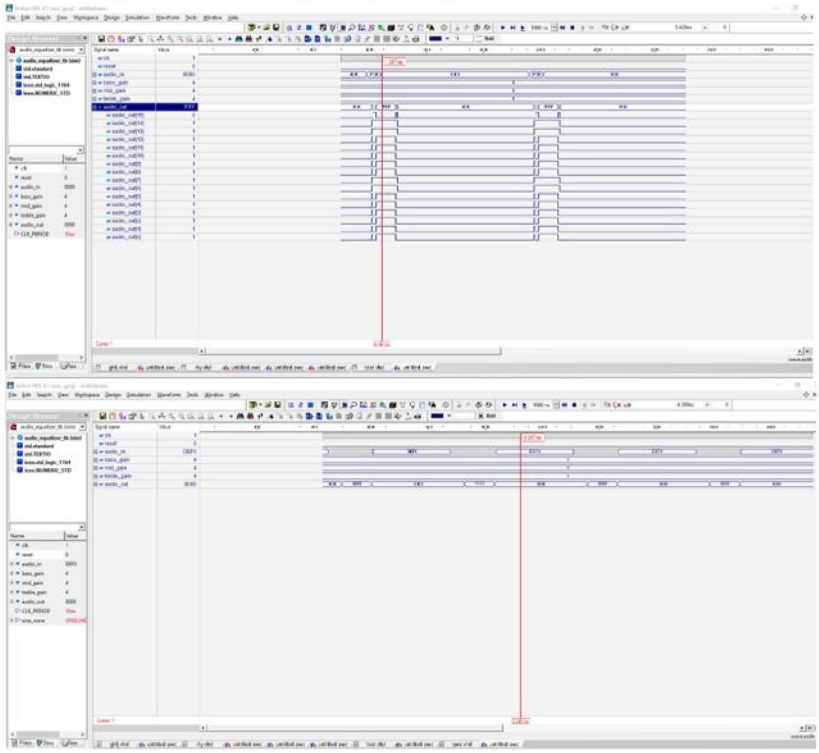
Реалізація FIR-фільтра у VHDL. Структура згортки FIR-фільтра. Фрагмент коду




```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
 entity audio_equalizer is
  port (
   clk : in std_logic;
   reset : in std_logic;
   audio_in : in signed(15 downto 0);
   bass_gain : in signed(3 downto 0);
   mid_gain : in signed(3 downto 0);
   treble_gain : in signed(3 downto 0);
   audio_out : out signed(15 downto 0)
  );
end audio_equalizer;
 architecture Behavioral of audio_equalizer is
  constant FIR_TAPS : integer := 32;
 type sample_array is array (0 to FIR_TAPS-1) of signed(15 downto 0);
  signal shift_reg : sample_array := (others => (others => '0'));
  -- FIR coefficients
 constant bass_coeff : sample_array := (
  to_signed(-2,16),to_signed(-1,16),to_signed(0,16),to_signed(3,16),
  to_signed(5,16),to_signed(8,16),to_signed(10,16),to_signed(12,16),
  others => to_signed(0,16)
 );
```

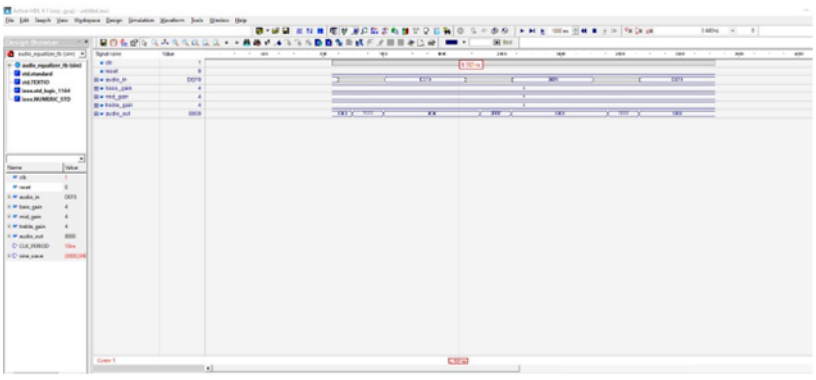
10

VHDL-реалізація розробленого пристрою



11

Реалізація фільтру на VHDL



12

Висновки

У ході виконання роботи було здійснено повноцінне проєктування цифрового трисмугового еквалайзера із використанням мови опису апаратури VHDL. Основна мета полягала у створенні функціональної моделі пристрою цифрової обробки аудіосигналів, який дозволяє регулювати підсилення в межах трьох частотних діапазонів: низьких, середніх та високих частот. Усі етапи розробки від визначення частотних характеристик до моделювання логіки системи — було реалізовано у віртуальному середовищі.

Результатом роботи стала структурно завершена VHDL-модель, яка включає модулі фільтрації на основі FIR-фільтрів, блоки підсилення та захисту від перевантаження сигналу. Особливу увагу приділено побудові масиву затримок, реалізації згортки, правильному вибору коефіцієнтів, а також інтеграції частотних каналів у єдину функціональну схему. Було проведено програмне моделювання у симуляторі, що дозволило перевірити коректність логіки, дослідити часові характеристики сигналів та переконатися у відповідності результатів очікуваним.

На основі отриманих результатів можна зробити висновок, що реалізована система забезпечує базову функціональність цифрового еквалайзера і є придатною для подальшої апаратної реалізації на FPGA. Модель демонструє стабільну роботу у середовищі симуляції, чітко реагує на зміну коефіцієнтів підсилення та ефективно обмежує сигнал у разі перевищення допустимого діапазону. Попри відсутність фізичного прототипу, проєкт дає повне уявлення про алгоритмічну структуру пристрою та відкриває перспективи для його впровадження в реальному апаратному середовищі.

ДОДАТОК Б

Програмний код

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity audio_equalizer is
  port (
    clk         : in  std_logic;
    reset       : in  std_logic;
    audio_in    : in  signed(15 downto 0);
    bass_gain   : in  signed(3 downto 0);
    mid_gain    : in  signed(3 downto 0);
    treble_gain : in  signed(3 downto 0);
    audio_out   : out signed(15 downto 0)
  );
end audio_equalizer;

architecture Behavioral of audio_equalizer is

  constant FIR_TAPS : integer := 32;
  type sample_array is array (0 to FIR_TAPS-1) of signed(15
downto 0);

  signal shift_reg : sample_array := (others => (others =>
'0'));

  -- FIR coefficients
  constant bass_coeff : sample_array := (
    to_signed(-2,16),to_signed(-
1,16),to_signed(0,16),to_signed(3,16),

to_signed(5,16),to_signed(8,16),to_signed(10,16),to_signed(12,16
),
    others => to_signed(0,16)
  );

  constant mid_coeff : sample_array := (
    to_signed(-1,16),to_signed(-
2,16),to_signed(4,16),to_signed(10,16),

to_signed(12,16),to_signed(10,16),to_signed(4,16),to_signed(-
2,16),
    to_signed(-1,16),others => to_signed(0,16)
  );

  constant treble_coeff : sample_array := (
    to_signed(10,16),to_signed(-5,16),to_signed(-
```

```
4,16),to_signed(0,16),
    to_signed(4,16),to_signed(-5,16),to_signed(10,16),
    others => to_signed(0,16)
  );

  signal bass_sum, mid_sum, treble_sum : signed(31 downto 0);
  signal bass_scaled, mid_scaled, treble_scaled : signed(31
downto 0);
  signal mixed_out : signed(31 downto 0);

begin

  FIR_Process: process(clk, reset)
    variable bass_accum, mid_accum, treble_accum : signed(31
downto 0);
    variable gain_bass_32, gain_mid_32, gain_treble_32 :
signed(31 downto 0);
  begin
    if reset = '1' then
      shift_reg <= (others => (others => '0'));
      audio_out <= (others => '0');
    elsif rising_edge(clk) then
      shift_reg(0) <= audio_in;
      for i in 1 to FIR_TAPS-1 loop
        shift_reg(i) <= shift_reg(i-1);
      end loop;

      bass_accum   := (others => '0');
      mid_accum    := (others => '0');
      treble_accum := (others => '0');

      for i in 0 to FIR_TAPS-1 loop
        bass_accum   := bass_accum   + (shift_reg(i) *
bass_coeff(i));
        mid_accum    := mid_accum    + (shift_reg(i) *
mid_coeff(i));
        treble_accum := treble_accum + (shift_reg(i) *
treble_coeff(i));
      end loop;

      bass_sum   <= bass_accum;
      mid_sum    <= mid_accum;
      treble_sum <= treble_accum;

      -- Sign-extend gain inputs
      gain_bass_32   := resize(bass_gain, 32);
      gain_mid_32    := resize(mid_gain, 32);
      gain_treble_32 := resize(treble_gain, 32);

      -- Scaling without division (for testing output)
      bass_scaled   <= resize(bass_accum * gain_bass_32, 32);
      mid_scaled    <= resize(mid_accum * gain_mid_32, 32);
      treble_scaled <= resize(treble_accum * gain_treble_32,
```

```
32);

      mixed_out <= bass_scaled + mid_scaled + treble_scaled;

      -- Clipping
      if mixed_out > 32767 then
        audio_out <= to_signed(32767,16);
      elsif mixed_out < -32768 then
        audio_out <= to_signed(-32768,16);
      else
        audio_out <= mixed_out(15 downto 0);
      end if;

    end if;
  end process;

end Behavioral;
```