ДОДАТОК А

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity Coder_Viterbi is
    Port (
            -----------Inputs-----------
             clk_for_in  : in std_logic;
             clk_for_out : in std_logic;
             I_In        : in std_logic;
             Q_In        : in std_logic;
             mode        : in std_logic;
             ----------Outputs-----------
             I_Out       : out std_logic;
             Q_Out       : out std_logic
           );

end Coder_Viterbi;

architecture Coder_Viterbi_Arch of Coder_Viterbi is

signal state : std_logic:='0';
signal Iz0, Iz1, Iz2, Iz3, Iz4, Iz5, Iz6 : std_logic:='0';
signal Qz0, Qz1, Qz2, Qz3, Qz4, Qz5, Qz6 : std_logic:='0';
signal IS1, IS2 : std_logic:='0';
signal QS1, QS2 : std_logic:='0';

begin

process (clk_for_in, clk_for_out)
begin
    if ( rising_edge(clk_for_in) ) then
        Iz0 <= I_In; Qz0 <= Q_In;
        Iz1 <= Iz0;  Qz1 <= Qz0;
        Iz2 <= Iz1;   Qz2 <= Qz1;
        Iz3 <= Iz2;   Qz3 <= Qz2;
        Iz4 <= Iz3;   Qz4 <= Qz3;
        Iz5 <= Iz4;   Qz5 <= Qz4;
        Iz6 <= Iz5;   Qz6 <= Qz5;


        if (mode = '1') then
            IS1 <= Iz0 xor Iz1 xor Iz2 xor Iz3 xor Iz6;
            IS2 <= Iz0 xor Iz2 xor Iz3 xor Iz5 xor Iz6;
            QS1 <= not(Qz0 xor Qz1 xor Qz2 xor Qz3 xor Qz6);
            QS2 <= not(Qz0 xor Qz2 xor Qz3 xor Qz5 xor Qz6);
        else
            IS1 <= Iz0 xor Iz1 xor Iz2 xor Iz3 xor Iz6;
            IS2 <= not(Iz0 xor Iz2 xor Iz3 xor Iz5 xor Iz6);
            QS1 <= Qz0 xor Qz1 xor Qz2 xor Qz3 xor Qz6;
            QS2 <= not(Qz0 xor Qz2 xor Qz3 xor Qz5 xor Qz6);
        end if;
        state <= '0';
```

```
        end if;


        if( rising_edge(clk_for_out)) then

            if( state = '0' ) then
                 I_Out <= IS1;
                 Q_Out <= QS1;
                 state <= '1';
            else
                 I_Out <= IS2;
                 Q_Out <= QS2;
            end if;
        end if;


    end process;

    end Coder_Viterbi_Arch;
```

ДОДАТОК Б

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity QPSK_coder is
    Port (
                Fclk : in std_logic;
                StrDIBit : in std_logic;
                StrDIBit2 : in std_logic;
                I_In : in std_logic;
                Q_In : in std_logic;

                StrIQ_Out : out std_logic;
                I_Out : out std_logic;
                Q_Out : out std_logic
          );

end QPSK_coder;


architecture Structure of QPSK_coder is

signal Fsm, StrDIBit2z : std_logic:='0';
signal Iok, Qok : std_logic:='0';
signal Iz1, Iz2, Iz3, Iz4, Iz5, Iz6 : std_logic:='0';
signal Qz1, Qz2, Qz3, Qz4, Qz5, Qz6 : std_logic:='0';
signal IS1, IS2 : std_logic:='0';
signal QS1, QS2 : std_logic:='0';
signal Ick, Qck : std_logic:='0';

begin


process (Fclk)
begin
if ( Fclk'event and Fclk = '1' ) then

 if( StrDIBit = '1' ) then

        Iok <= I_In xor Iok;
        Qok <= Q_In xor Qok;

        Iz1 <= Iok;    Qz1 <= Qok;
        Iz2 <= Iz1;   Qz2 <= Qz1;
        Iz3 <= Iz2;   Qz3 <= Qz2;
        Iz4 <= Iz3;   Qz4 <= Qz3;
        Iz5 <= Iz4;   Qz5 <= Qz4;
        Iz6 <= Iz5;   Qz6 <= Qz5;

        IS1 <= Iok xor Iz1 xor Iz2 xor Iz3 xor Iz6;
        IS2 <= not(Iok xor Iz2 xor Iz3 xor Iz5 xor Iz6);

        QS1 <= Qok xor Qz1 xor Qz2 xor Qz3 xor Qz6;
        QS2 <= not(Qok xor Qz2 xor Qz3 xor Qz5 xor Qz6);
```

```
            Fsm <= '0';

      end if;

      StrDIBit2z <= StrDIBit2;

      if( StrDIBit2z = '1' ) then

            if( Fsm = '0' ) then
                  Ick <= IS1;
                  Qck <= QS1;
                  Fsm <= '1';
            else
                  Ick <= IS2;
                  Qck <= QS2;
            end if;
      end if;


end if;
end process;


StrIQ_Out <= StrDIBit2z;
I_Out <= Ick;
Q_Out <= Qck;



end Structure;
```

ДОДАТОК В

```
library IEEE;
 use IEEE.STD_LOGIC_1164.ALL;
 use IEEE.STD_LOGIC_ARITH.ALL;
 use IEEE.STD_LOGIC_UNSIGNED.ALL;
library UNISIM;
 use UNISIM.VComponents.all;

entity IQFormer is
 port (
 ----------Inputs---------
 clk_in    : in std_logic;
 Data_In   : in std_logic;
 rst            : in std_logic;
 ---------Outputs---------
 I_Out     : out std_logic;
 Q_Out     : out std_logic;
 clk_s_out : out std_logic   --1.5/25MHz
 );
end IQFormer;

architecture IQFormer_Arch of IQFormer is
component BUFG
              port(
              I: in STD_LOGIC;
              O: out STD_LOGIC
                     );
       end component;
signal state : std_logic := '0';

begin
 process(clk_in)
 begin
       if (rst='1')then
              I_Out <= '0';
              Q_Out <= '0';
              state <= '0';
       elsif (rising_edge(clk_in)) then
              if (state = '0') then
                     I_Out <= Data_In;
              else
                     Q_Out <= Data_In;
              end if;
              state <= not state;
       end if;
 end process;
U6: BUFG port map
                     (
                     I => state,
                     O => clk_s_out
                     );
end IQFormer_Arch;
```

ДОДАТОК Г

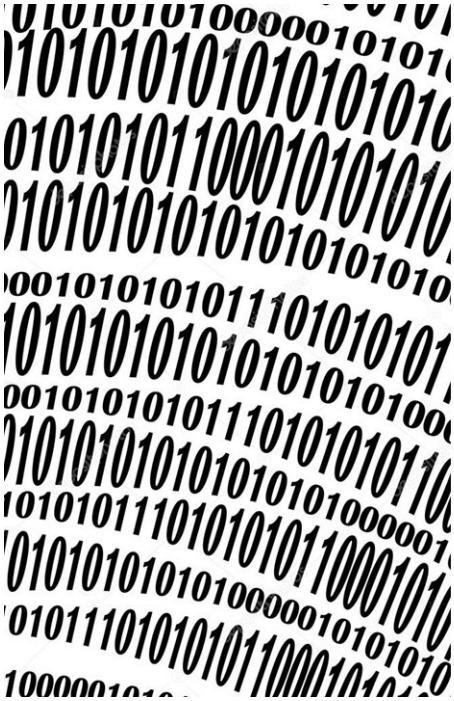
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
library UNISIM;
use UNISIM.VComponents.all;

entity Top_Transceiver is
 port (
 --------Inputs----------
 rst      : in std_logic;
 data_in        : in std_logic;
 clk_in   : in std_logic;
 mode     : in std_logic;
 ------TEST PINS---------
 I_t    : out std_logic;
 Q_t    : out std_logic;
 clk_s_t : out std_logic;
 -------Outputs----------
 I              : out std_logic;
 Q              : out std_logic
 );
end Top_Transceiver;

architecture Top_Arch of Top_Transceiver is


component IQFormer
 port(
 clk_in    : in std_logic;
 Data_In   : in std_logic;
 rst            : in std_logic;

 ---------Outputs---------
 I_Out     : out std_logic;
 Q_Out     : out std_logic;
 clk_s_out : out std_logic   --1.5/25MHz
 );
end component;
signal I_int : std_logic;
signal Q_int : std_logic;
signal clk_s : std_logic;

component Coder_Viterbi
 port(
        clk_for_in : in std_logic;
        clk_for_out : in std_logic;
        I_In : in std_logic;
        Q_In : in std_logic;
        mode : in std_logic;
        I_Out : out std_logic;
        Q_Out : out std_logic
         );
end component;
```

```
begin
 IQ_form: IQFormer
        port map(
               clk_in    => clk_in,
               Data_In   => data_in,
               rst            => rst,
               ---------Outputs---------
               I_Out     => I_int,
               Q_Out     => Q_int,
               clk_s_out => clk_s
               );
 I_t <= I_int;
 Q_t <= Q_int;
 clk_s_t <= clk_s;
 coding:  Coder_Viterbi
        port map(
               clk_for_in => clk_s,
               clk_for_out => clk_in,
               I_In => I_int,
               Q_In => Q_int,
               mode => mode,
               I_Out => I,
               Q_Out => Q
               );



end Top_Arch;
```

ДОДАТОК Д

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity modulator is
  port(
        I_in  : in STD_LOGIC;
        Q_in  : in STD_LOGIC;
        ----------------------------------------
        I_out : out STD_LOGIC_VECTOR(11 downto 0);
        Q_out       : out STD_LOGIC_VECTOR(11 downto 0)
      );
end modulator;


architecture modulator of modulator is
begin

 I_out <= x"80"     when (I_in = '0') else x"AAA";
 Q_out <= x"80"     when (Q_in = '0') else x"AAA";

end modulator;
```

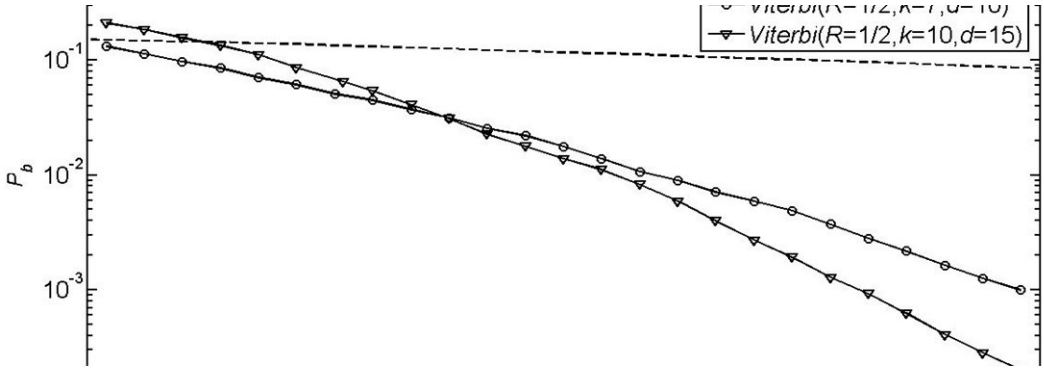
ДОДАТОК Ж



ДЕКОДЕР ВІТЕРБІ НА ПРОГРАМОВАНИХ ЛОГІЧНИХ ІНТЕГРАЛЬНИХ СХЕМАХ ДЛЯ СИСТЕМИ ЦИФРОВОГО НАЗЕМНОГО ТЕЛЕБАЧЕННЯ СТАНДАРТУ DVB-T (ТЕМА)

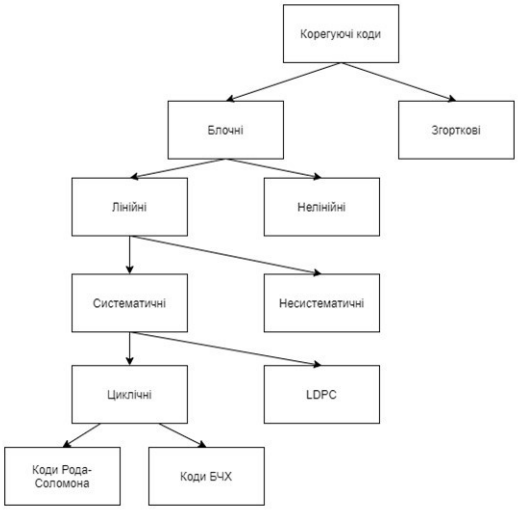
СКСМ-19-1

КРАВЦОВ КИРИЛО РОМАНОВИЧ



МЕТА РОБОТИ

При передачі даних по каналах зв'язку через пошкодження сигналів і впливу перешкод можливий помилковий прийом. Число виникаючих помилок визначається характеристиками каналу, і зазвичай їх неприпустимо багато. Методи боротьби з помилками різноманітні. Як відомо, будь-яка повторна передача даних веде до зменшення її швидкості, тому ефективніше використовувати методи боротьби з помилками без повторення. До них відносяться методи, які використовують коди, що виправляють помилки, зокрема, коди Ріда-Соломона, Боуза-Чоудхурі-хоквінгема (БЧХ), згорткові коди і багато інших.

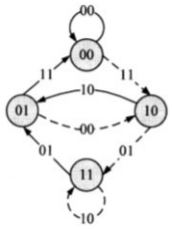


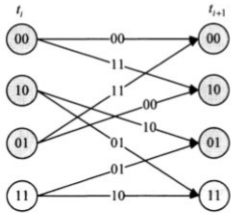
КОДИ

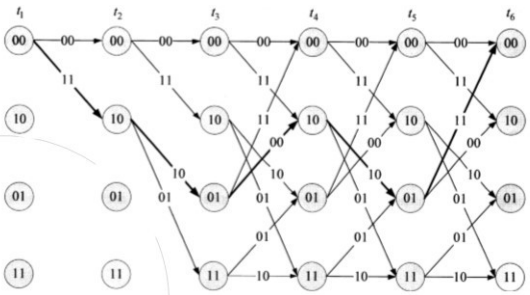
- Ріда-Соломона
- Боуза-Чоудхурі-хоквінгема
- LDPC
- Вітербі

ЗГОРТКОВЕ КОДУВАННЯ

Кодування згортковим кодом є лінійним перетворенням інформаційної послідовності, в загальному випадку, необмеженої довжини. При цьому введення надмірності полягає в тому, що кожна розглянута група кодових символів виявляється функцією не тільки поточної групи інформаційних символів, але і ряду попередніх груп інформаційних символів.

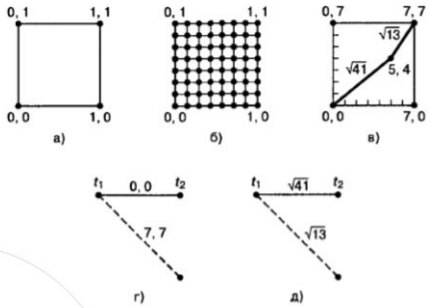






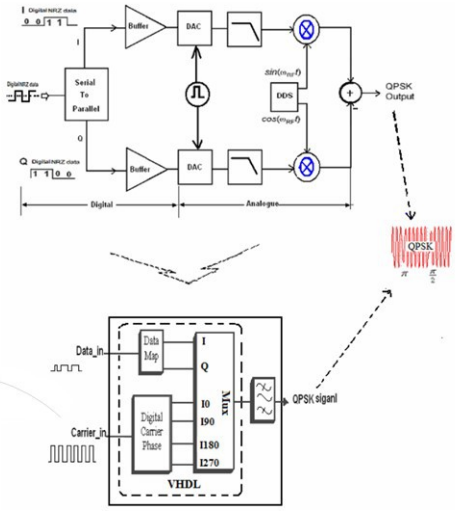
ДЕКОДУВАННЯ ЗГОРТКОВИХ КОДІВ

Декодування згорткових кодів за алгоритмом Вітербі. Сутність алгоритму декодування Вітербі складається в покроковому порівнянні всіх можливих шляхів (послідовності переданих символів) на гратчастої діаграмі декодера, аналогічної з спостережуваної послідовністю «жорстких» або «м'яких» рішень демодулятора.



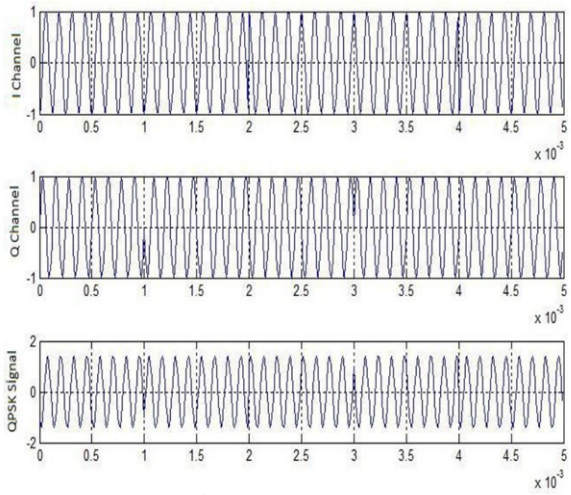
QPSK МОДУЛЯТОР

Принцип роботи модулятора QPSK – в квадратурно-фазовій маніпуляції (QPSK): два послідовні біти в послідовності даних групуються разом. Це зменшує бітову швидкість передачі сигналів (fb) і, отже, зменшує пропускну здатність каналу.

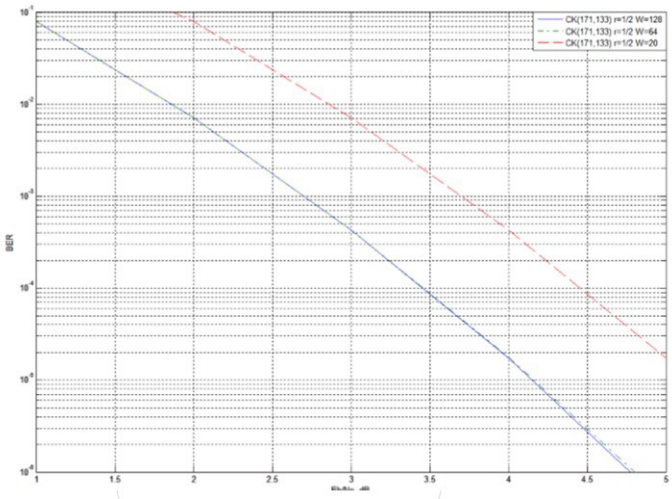


СИГНАЛ ПІСЛЯ МОДУЛЯЦІЙ

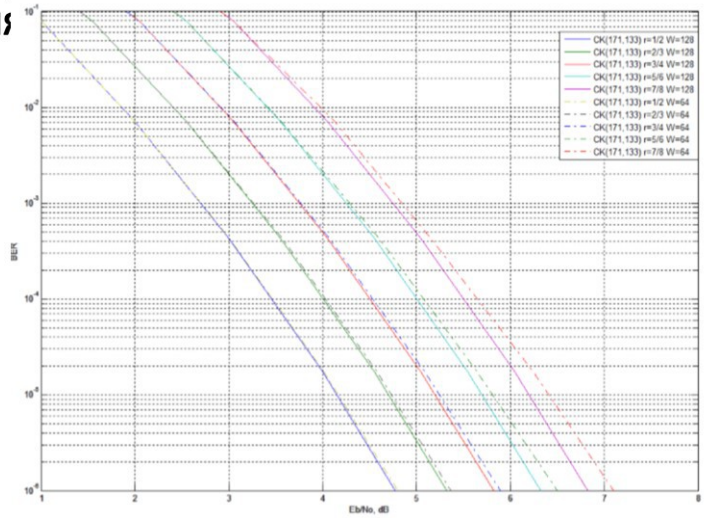
Дві двійкові послідовності модулюються окремо двома носіями φ1 (t) і φ2 (t), які знаходяться в квадратурі. Два модульованих сигнали, кожен з яких можна вважати сигналом BPSK, є підсумком для отримання сигналу QPSK. Фільтр на вихід модулятора обмежує спектр потужності сигналу QPSK в межах виділеної смуги. Це запобігає зменшенню енергії сигналу на сусідню каналів, а також усуває помилки поза діапазоном сигнали, що генеруються в процесі модуляції. У більшості реалізацій формування імпульсів здійснюється за адресою базової смуги для забезпечення належної ВЧ-фільтрації на вихід передавача.



ПЕРЕШКОДОСТІЙКІСТЬ ДЕКОДЕРА ВІТЕРБІ ПРИ РІЗНІЙ ГЛИБИНІ ДЕКОДУВАННЯ (ШВИДКІСТЬ КОДУ 1/2)



ПЕРЕШКОДОСТІЙКІСТЬ ДЕКОДЕРА ПРИ РІЗНИХ ШВИДКОСТЯХ ЗГОРТАЛЬНОЇ КОДИ І РІЗНІЙ ГЛИБИНІ ДЕКОДУВАННЯ



РЕЗУЛЬТАТИ КОМПІЛЯЦІЇ

Глибина декодування	Використана пам'ять	Всього пам'яті на ПЛІС	Глибина декодування	Використана пам'ять
60	57331	239616	60	57331
80	76459		80	76459
100	95572		100	95572
120	114702		120	114702

ПЕРЕШКОДОСТІЙКІСТЬ ДЕКОДЕРА ПРИ РІЗНОЇ БІТНОСТІ М'ЯКИХ РІШЕНЬ

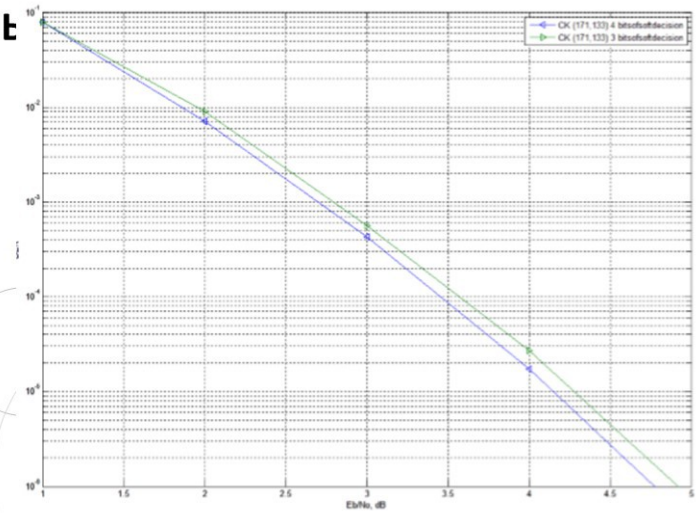
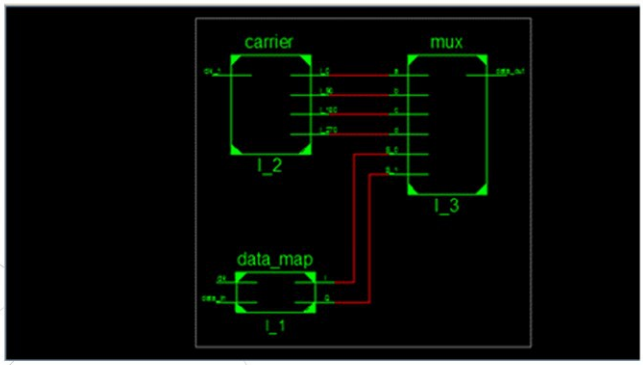
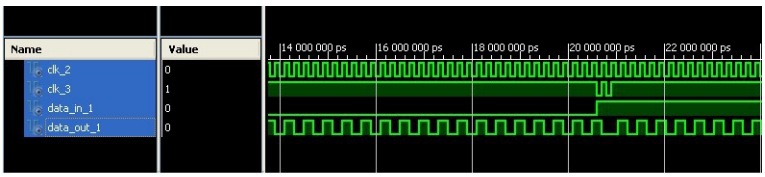
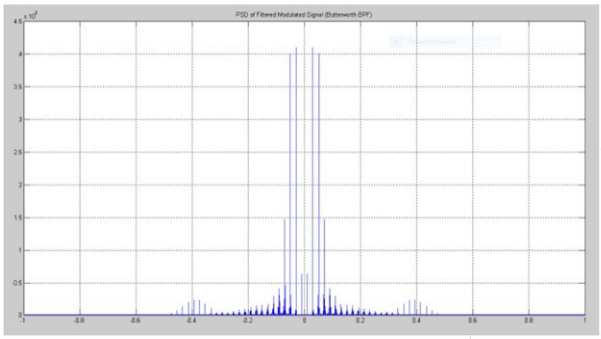


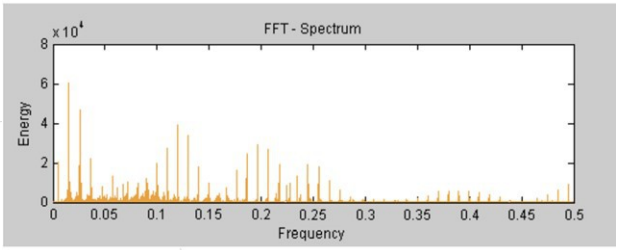
СХЕМА RTL ДЛЯ МОДУЛЯТОРА QPSK





ВИХІДНІ ДАНІ

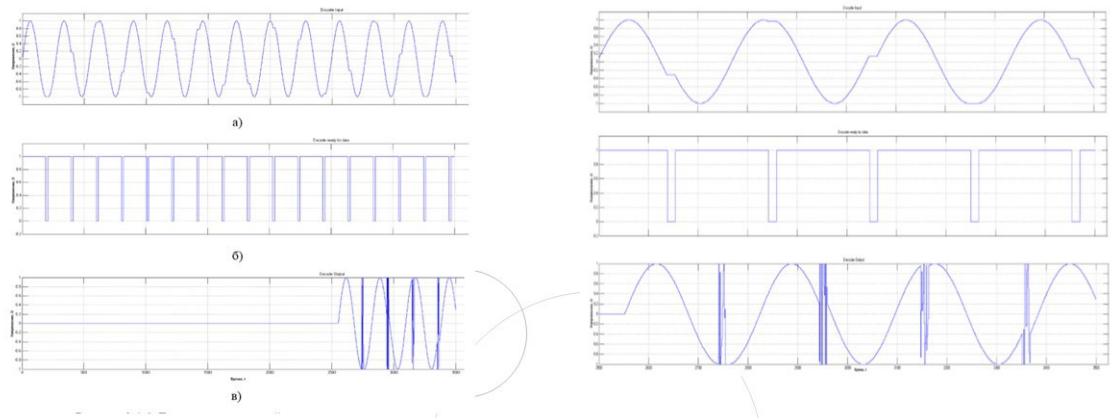




РЕЗУЛЬТАТИ КОМПІЛЯЦІЇ

Тип фільтру	Частота даних	Несуча частота	Вихідна частота	Тип фільтру
Butterworth BPF	500 Kbps	5 MHz	2 MHz	Butterworth BPF
Wavelet	500 Kbps	5 MHz	3 MHz	Wavelet
Butterworth LPF	500 Kbps	5 MHz	1 MHz	Butterworth LPF

СИМУЛЯЦІЯ В MATLAB

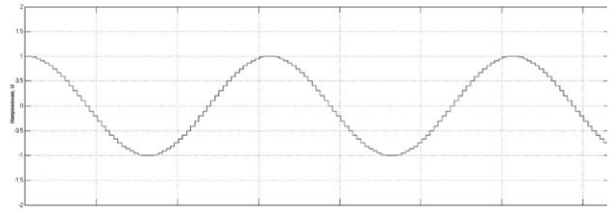


а)

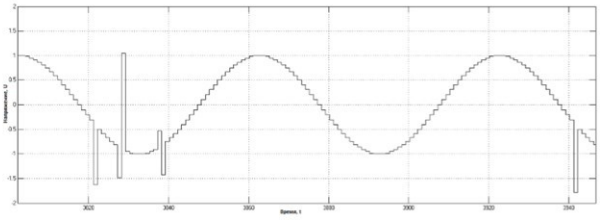
б)

в)

СИМУЛЯЦІЯ В MATLAB

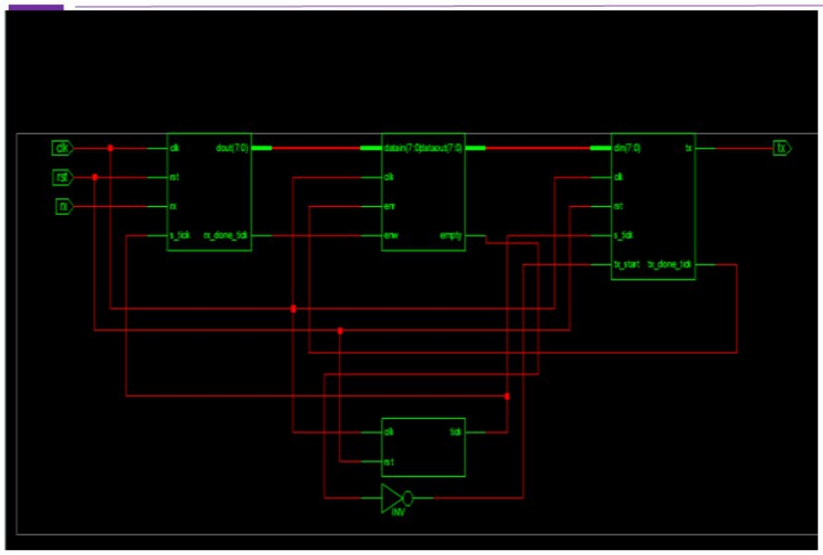


а)



б)

ВИХІДНИЙ RTL



ДОДАТОК З

ВИКОРИСТАННЯ ДЕКОДЕРУ ВІТЕРБІ В РНК БЛОКАХ НА ПЛІС

Кравцов К.Р.
Науковий керівник – к.т.н., доц. Філіппенко І.В.
Харківський національній університет радіоелектроніки
(61166, Харків, пр. Науки,14, каф. Автоматизації проектування
обчислювальної техніки, тел (057) 702-13-26
e-mail: kyrylo.kravtsov@nure.ua, тел. +380996655604

Convolutional and lattice codes are widely used in modern data transmission systems as noise-coding schemes. The popularity of these codes is due to the ability to decode them using the Viterbi algorithm.

Згорткові і гратчасті коди широко використовуються в сучасних системах передачі даних в якості схем завадостійкого кодування. Популярність цих кодів обумовлена можливістю їх декодування за допомогою алгоритму Вітербі, що забезпечує оптимальне декодування за критерієм максимальної правдоподібності при відносно невеликих й (в порівнянні з іншими класами кодів) обчислювальної складності.

Було розглянуто архітектуру і деталі апаратної реалізації розробленого блоку декодера Вітербі для згорткового коду з базовим темпом кодування 1/2, довжиною кодового обмеження K = 7, заданого породжують поліномами (133, 177). Цей згортковий код широко поширений і використовується в системах бездротового широкосмугового доступу на основі стандарту IEEE 802.16 (Fixed WiMAX), а також в системах бездротового зв'язку IEEE 802.11a, b, g, n (Wi-Fi), цифрового телебачення (DVB-T, H)та багатьох інших системах. У реалізованому модулі підтримуватимуться пакетний і безперервний режими роботи. У пакетному режимі для завершення процедури кодування використовується доповнення нулями (zerotailing). Крім базового темпу кодування 1/2, можлива підтримка швидкостей кодування 2/3, 3/4, 5/6 з використанням процедури виколювання. Параметри цілочисельних операцій, що виконуються всередині блоку що розробляється, параметризовані і можуть бути змінені перед процедурою логічного синтезу, надаючи розробнику компроміс між продуктивністю, займаною площею на кристалі і пропускною спроможністю модуля.

Розглянутий блок декодераВітербі був спроектований для створення прототипу бездротової системи зв'язку на ПЛІС, проте також може бути використана як блок в спеціалізованих НВІС.

Для цього блоку були отримані оцінки його характеристик при реалізації на ПЛІС Stratix II фірми Altera. Дані показники наведені в табл. 1.

Як видно, розроблений модуль декодера Вітербі забезпечує пропускну здатність до 220 Мбіт/с, що є достатнім для його застосування в більшості сучасних систем передачі, включаючи системи бездротового зв'язку WiMAX на основі стандарту IEEE 802.16.

Таблиця 1 – Характеристики при реалізації на ПЛІС

Назва параметру	Значення
Максимальна тактова частота, МГц	220
Число еквівалентних логічних елементів	5960
Необхідний об'ємпам'яті, біт	98816
Затримка обчислень	4D

Концепція, що виникла і активно розвивається протягом останніх кількох років, покликана забезпечити задовільні час розробки та верифікацію інтегральних схем з постійно зростаючою складністтю. У відповідності з цією концепцією розробку кінцевої інтегральної системи можна уявити як розробку окремих блоків, їх верифікацію з подальшим об'єднанням на одному кристалі.

Сучасні пристрої бездротового зв'язку для мобільних пристроїв є характерними прикладами, що включають в себе безліч блоків, які реалізують різноманітні функції аналогової і цифрової обробки сигналів. В роботі була розглянута реалізація трьох блоків для систем бездротового зв'язку: блоку швидке перетворення Фур'є, декодера Вітербі і апаратного емулятора бездротової лінії зв'язку. Ці блокизастосовні для розробки і досліджень сучасних систем бездротового широкосмугового доступу типу WiMAX і призначені для їх подальшої інтеграції в ПЛІС і спеціалізованих інтегральної систем.

Список використаних джерел:

1. Мальцев А.А., Масленников Р.О., Тхора А.В., Пестрецов В.А., Шилов М.С. «СФ-блок швидкого перетворення Фур'є для бездротових систем зв'язку на основі стандарту IEEE 802.16 e Mobile WiMAX», Праці конференції «Проблеми розробки перспективних мікро- і наноелектронних систем» - 2008 (МЕМ-2008), Москва, 200, 6 с.

2. Maltsev, A. Khoryaev, A. Lomayev, R. Maslennikov, M. Shilov, V. Pestretsov, A. Sevastyanov, «Pseudostoring Hardware Link Level Emulatorfor System Level Simulations of WiMAX-based Systems», InProc. ICT Mobile Summit 2008, Stockholm, Sweden, 8 p.

3. Maltsev, A. Khoryaev, A. Lomayev, R. Maslennikov, M. Shilov, A. Sevastyanov, «Real Time Hardware-Software Emulator of MEMBRANE Multihop Wireless Network», submit tedto 2nd Int. Conf. Simulation Tools and Techniques – Simutools 2009., 8 p.

№	Позначення	Найменування	Дод. відомості
		Текстові документи	
1	ГЮІК.ХХХХХХ.006 ПЗ	Пояснювальна записка	53 с.
		Графічні матеріали	
2		Слайд-презентація	17 слайдів
		Носії інформації	
3		Компакт-диск (CD)	1 шт.

Змін.	Арк.	Номер докум.	Підп.	Дата	ГЮІК.ХХХХХХ.006 ВД			
Розроб.		Кравцов К.Р..			Моделі та методи проектування завадостійких систем бездротової передачі даних з використанням ПЛІС Відомість атестаційної роботи	Літ	Аркуш	Аркушів
Перевір.		Філіппенко І.В.				У		
Н.контр.		Рожнова Т.Г.		22.12		ХНУРЕ Кафедра АПОТ		
Затв.		Чумаченко С.В.		22.12				