

# Design And Implementation Of a Digital Direct Sequence Spread Spectrum (DSSS) System for Eight Users Using FPGA

**Dr. Kamal Aboutabikh**

Faculty of Informatics Engineering, Ittihad Private University, Damascus Syria.

## Abstract:

In this paper, we design a digital DSSS System using Cyclone II EP2C20F484C7 FPGA from ALTERA placed on education and development board DE-1 for the DSSS system with the transmitting and receiving sections according to the following parameters:

-Clock frequency of the system 10 KHz.

-Length of spreading pseudo-noise code (PNC): is (16) chips.

Type generation of spreading code: Walsh codes ( $H_{16}$ ).

-Length of data bit: is 3 bits with (16) chips for every one bit.

-spread operation: X

-Number of users: is eight.

-User 1 with Data (00) and pseudo-noise code:

$$\text{Code 1} = [1111111111111111]$$

$$1 \rightarrow +1, 0 \rightarrow -1 \Rightarrow \text{Code 1} = [1111111111111111]$$

-User 2 with Data (01) and pseudo-noise code:

$$\text{Code 2} = [1010101010101010]$$

$$1 \rightarrow +1, 0 \rightarrow -1 \Rightarrow \text{Code 2} = [1-11-11-11-11-11-11-11-11-11-11-11-11-11-11-11]$$

-User 3 with Data (10) and pseudo-noise code:

$$\text{Code 3} = [1100110011001100]$$

$$1 \rightarrow +1, 0 \rightarrow -1 \Rightarrow \text{Code 3} = [11-1-1-11-1-1-11-1-1-11-1-1-11-1-1-11]$$

-User 4 with Data (11) and pseudo-noise code:

$$\text{Code 4} = [1001100110011001]$$

$$1 \rightarrow +1, 0 \rightarrow -1 \Rightarrow \text{Code 4} = [1-1-111-1-111-1-111-1-111-1-111-1-111-11]$$

-User 5 with Data (00) and pseudo-noise code:

$$\text{Code 5} = [1111000011110000]$$

$$1 \rightarrow +1, 0 \rightarrow -1 \Rightarrow \text{Code 5} = [1111-1-1-1-1111-1-1-1-1111-1-1-1-1111-11]$$

-User 6 with Data (01) and pseudo-noise code:

$$\text{Code 6} = [1010010110100101]$$

$$1 \rightarrow +1, 0 \rightarrow -1 \Rightarrow \text{Code 6} = [1-11-1-11-1-111-1-1-1-11-1-1-11-1-11]$$

-User 7 with Data (10) and pseudo-noise code:

$$\text{Code 7} = [1100001111000011]$$

$$1 \rightarrow +1, 0 \rightarrow -1 \Rightarrow \text{Code 7} = [11-1-1-1-1111-1-1-1-1111-1-1-1-1111-11]$$

-User 8 with Data (11) and pseudo-noise code:

$$\text{Code 8} = [1001011010010110]$$

$$1 \rightarrow +1, 0 \rightarrow -1 \Rightarrow \text{Code 8} = [1 -1 -1 1 -1 1 1 -1 -1 1 -1 1 1 -1]$$

These codes must be orthogonal to each other, so they are chosen according to Welch's codes via the HADAMARD matrix, for a (16-bit) codes, the  $H_{16}$  array should be chosen.

**Keywords:** CDMA, DSSS, Walsh Codes, PNC, PNCG, FPGA.

## I. INTRODUCTION

The DSSS system is one of the Code Divisions Multiply Accesses channels (CDMA) and is used in wireless communication and Wi-Fi networks. It relies on spreading data bits across the spectrum according to a pseudo-noise code, assigning each user their own unique code.

Orthogonal codes must be used for the users to reduce interference between user signals.

Walsh codes are typically used to achieve orthogonality between the codes.

The most important requirement of the DSSS system is synchronization and coordination among all system components to ensure proper and normal operation.

The mathematical principle of spreading the spectrum using DSSS can be explained according to the diagram shown in figure (1), where the spectrum of the data signal is spread according to a PNCG and a mathematical spreading function at the transmitting side.

While on the receiving side, the process is the opposite, so that the spectrum is collected according to the same PNCG and the mathematical spreading function to obtain the initial spectrum of the data signal.

In paper [1], the DSSS is designed for pseudo-noise code generator with only (11) chips, while in this research we have (16) chips and it is possible to develop up to (64) chips.

In paper [2], the DSSS is designed for four users, while in this research we have eight users and (16) chips and it is possible to develop up to (64) chips.

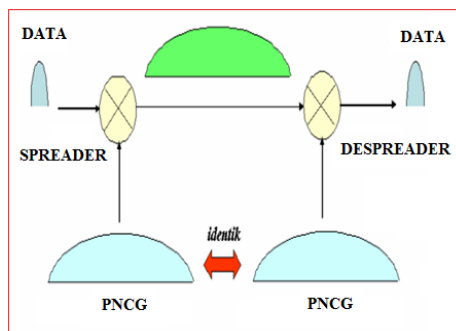


Figure (1) principle of spreading the spectrum using DSSS

## II. RESEARCH IMPORTANCE AND ITS OBJECTIVES

The purpose of the research is to design a digital DSSS system in the baseband signal domain to train students on digital design and to understand and comprehend DSSS system techniques.

This system consists of eight users, each having its own unique (16-bit) code and data bit sequence.

During transmission, the user's signals are combined after spreading according to each user's code and then transmitted as a combined signal.

In the receiver, there are eight channels for processing, with each channel dedicate to receiving and processing the signal of one user.

Each channel consists of a multiplier for the combined signal and the code of the user whose signal needs to be extracted.

After multiplication, the signal passes through a digital matched filter (DMF), which is a digital delay line consisting of (16) taps.

At the end of the delay line, there is a summer that sum all delayed signals, allowing the compression and relative of the desired user's data, and the same process applies for the other users.

### III. RESEARCH MATERIALS AND ITS WAYS

To design, and test the DSSS for eight users, the following tools and software are used:

- Cyclone II EP2C20F484C7 FPGA chip from ALTERA with highly accuracy, speed, and level specifications, placed on education and development board DE-1 [3].
- Transmitter for eight users each having its own unique 16-bit Walsh code, data bit sequence and receiver which is considered as highly accuracy digital matched filter designed on FPGA chips.
- VHDL programming language with Quartus II 9.1 design environment [4].
- Design Environment MATLAB R2008a.
- GDS-1052 digital oscilloscope with Free Wave program to take the results.
- PC computer for designing and injecting the design in the FPGA chip.

### IV. BLOCK DIAGRAM OF THE DSSS SYSTEM

In this research, the block diagram of the DSSS system consists of eight users, each with its own unique (16-bit) code and specific data bits.

The signal from the eight transmitters, after spreading according to their user codes, are combined to obtain the overall transmission signal. This signal is applied to the receiver, which is composed of eight receiving channels. Each channel is dedicated to processing signal of one user to recover its data bits after processing.

The combined signal is multiplied by the code of one user, then subjected to a multi-output digital delay line with by (16) taps, followed by summation and division by (16).

If the result is greater than (0), the received bit is considered (1), and if the result is less (0), the received bit is considered (0), as will be explained later.

The block diagram of the DSSS system is shown in the figure (2), the figure (3) also shows the block diagram of the DSSS system implemented in the Quartus 9.1 environment.

The transmitter and receiver operate with precise coordination and synchronization using clock pulses at a frequency 10KHz.

All signals written in red on the block diagram will be measured and tested in this laboratory platform using digital oscilloscope GDS-1052.

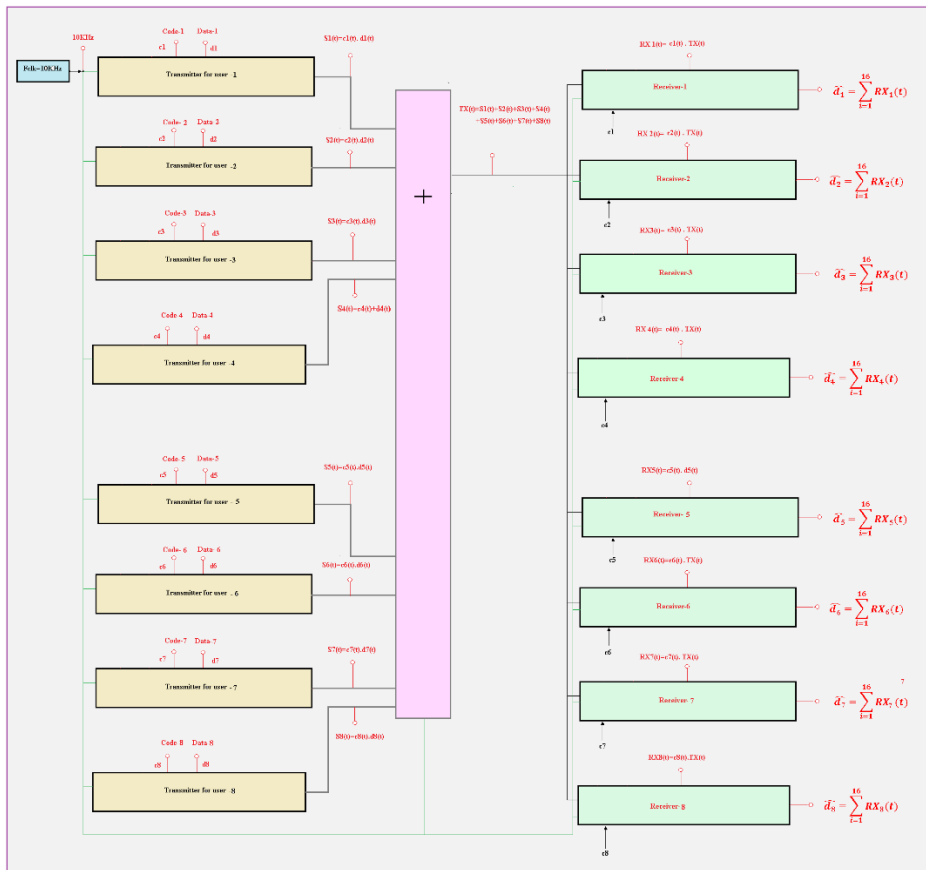


Figure (2): block diagram of the DSSS system for eight users.

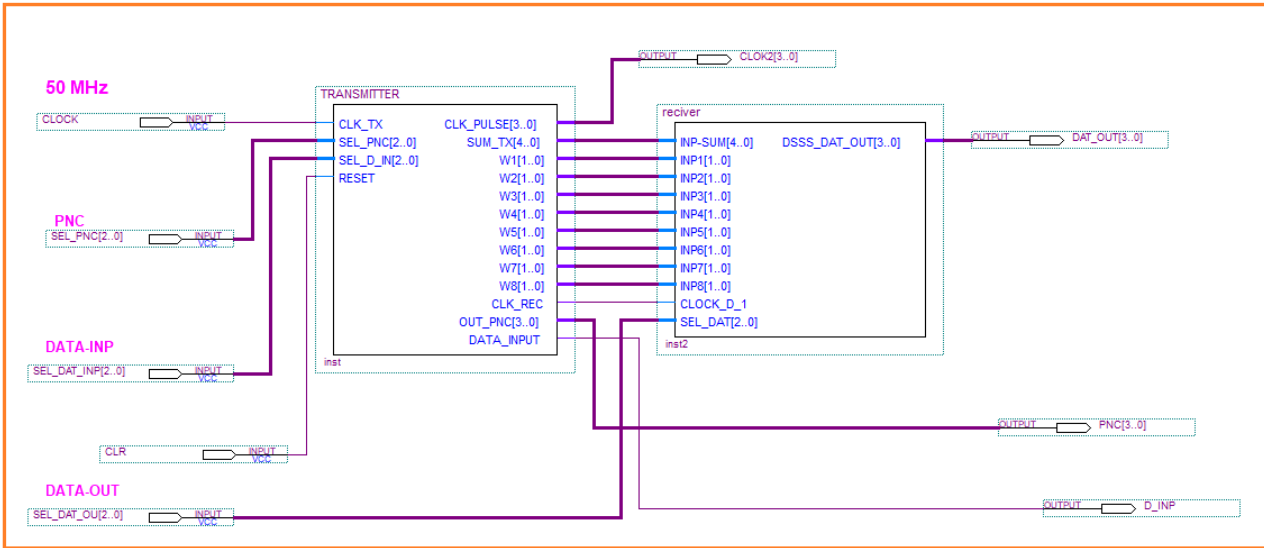


Figure (3): block diagram of the DSSS system for eight users in Quartus II 9.I design environment.

### V. BLOCK DIAGRAM OF THE TRANSMITTER DSSS SYSTEM

The block diagram of the DSSS system transmitter for eight users in the baseband frequency domain is shown in the figure (4).

The transmitting signal for the first, second, third and eighth users is given according to the following relations [5]:

$$s_i(t) = c_i(t) \cdot d_i(t) \quad (1)$$

Where:

$i=1,2,3,4,5,6,7,8$ ,  $s_i(t)$  signal of user  $i$ ,  $c_i(t)$  pseudo-noise code of user  $i$  and  $d_i(t)$  data of user  $i$ .

$$s_1(t) = c_1(t) \cdot d_1(t)$$

$$s_2(t) = c_2(t) \cdot d_2(t)$$

$$s_3(t) = c_3(t) \cdot d_3(t)$$

$$s_4(t) = c_4(t) \cdot d_4(t)$$

$$s_5(t) = c_5(t) \cdot d_5(t)$$

$$s_6(t) = c_6(t) \cdot d_6(t)$$

$$s_7(t) = c_7(t) \cdot d_7(t)$$

$$s_8(t) = c_8(t) \cdot d_8(t)$$

The transmitted sum signal is given according to the following relation:

$$TX(t) = \sum_{i=1}^{i=8} s_i(t) = \sum_{i=1}^{i=8} c_i(t) d_i(t) = c_1(t) \cdot d_1(t) + c_2(t) \cdot d_2(t) + c_3(t) \cdot d_3(t) + c_4(t) \cdot d_4(t) + c_5(t) \cdot d_5(t) + c_6(t) \cdot d_6(t) + c_7(t) \cdot d_7(t) + c_8(t) \cdot d_8(t) \quad (2)$$

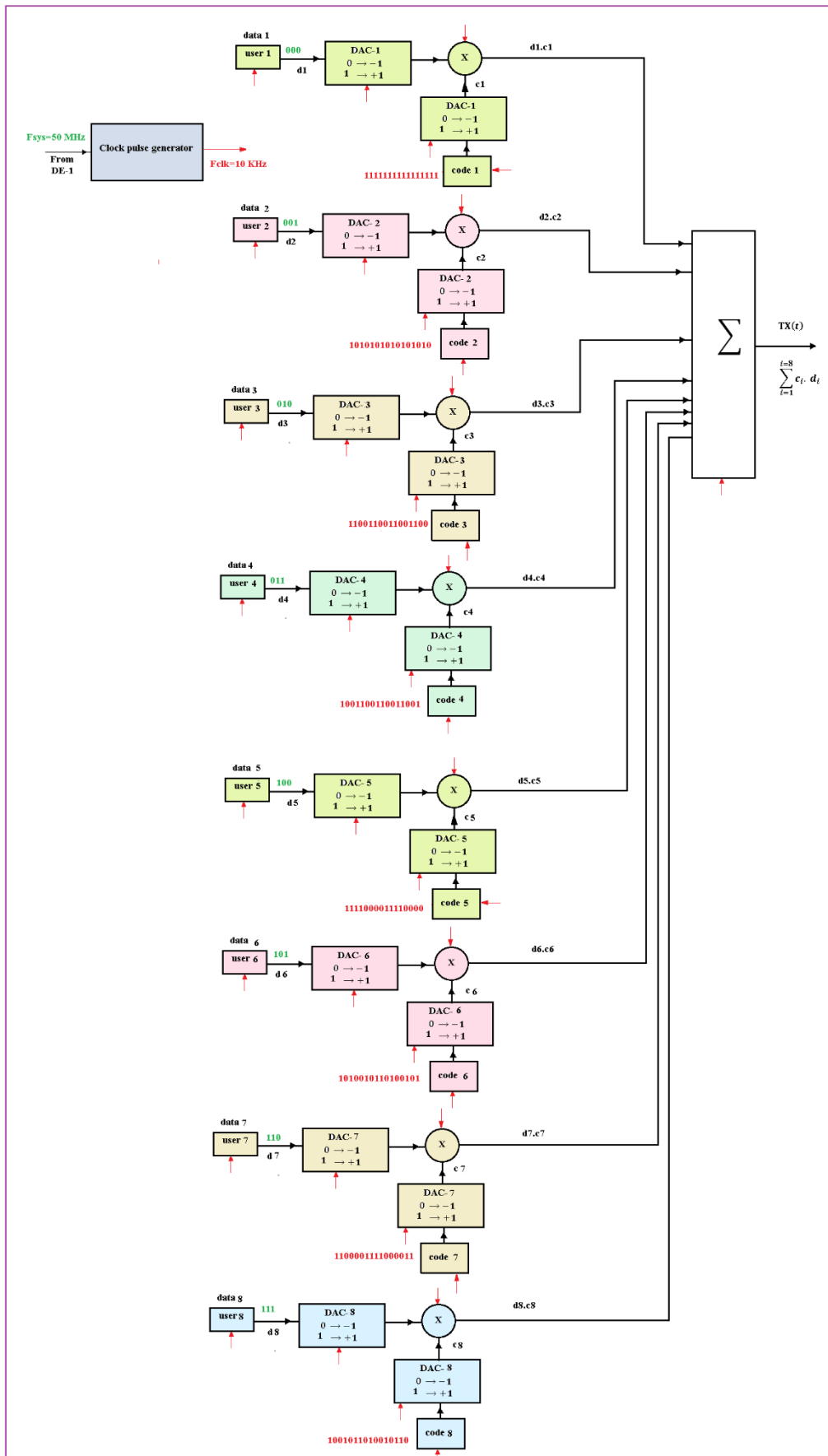


Figure (4): block diagram of the DSSS system transmitter for eight users.



<b>Cod e4</b>	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1
<b>data</b>	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
<b>code 4</b>	1	-	-	1	1	-	-	1	1	-	-	1	1	-	-	1	1	-	-	1	1	-	-	1	1	-	-	1	1	-	-	1	1	-	-	1
<b>data 4x code 4</b>	1	-	-	1	1	-	-	1	1	-	-	1	1	-	-	1	1	-	-	1	1	-	-	1	1	-	-	1	1	-	-	1	1	-	-	1

<b>Use r 5</b>	<b>Bit 1</b>																<b>Bit 0</b>																			
<b>Dat a (00)</b>	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
<b>Cod e5</b>	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1
<b>data</b>	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
<b>code 5</b>	1	1	1	1	-	-	-	-	1	1	1	1	-	-	-	-	1	1	1	1	-	-	-	-	1	1	1	1	-	-	-	-	1	1	1	1
<b>data 5x code 5</b>	-	-	-	-	1	1	1	1	-	-	-	-	1	1	1	1	-	-	-	-	1	1	1	1	-	-	-	-	1	1	1	1	-	-	-	-

<b>Use r 6</b>	<b>Bit 1</b>																<b>Bit 0</b>																			
<b>Dat a (01)</b>	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
<b>Cod e6</b>	1	0	1	0	0	1	0	1	1	0	1	0	0	1	0	1	1	0	1	1	0	0	1	0	1	1	0	1	0	0	1	0	1	0	1	
<b>Dat a6</b>	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
<b>code 6</b>	1	-	1	-	1	-	1	-	1	-	1	-	1	-	1	-	1	-	1	-	1	-	1	-	1	-	1	-	1	-	1	-	1	-	1	-
<b>data 6x code 6</b>	-	1	-	1	1	-	1	-	1	-	1	-	1	-	1	-	1	-	1	-	1	-	1	-	1	-	1	-	1	-	1	-	1	-	1	-

<b>Use r 7</b>	<b>Bit 1</b>																<b>Bit 0</b>																			
<b>Dat a (10)</b>	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
<b>Cod e7</b>	1	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	1	1
<b>data</b>	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
<b>code 7</b>	1	1	-	-	-	-	1	1	1	1	-	-	-	-	1	1	1	1	-	-	-	-	1	1	1	1	-	-	-	-	1	1	1	1	-	-
<b>data 7x code 7</b>	1	1	-	-	-	-	1	1	1	1	-	-	-	-	1	1	-	-	-	-	-	1	1	-	-	-	-	1	1	-	-	-	-	1	1	-

Use r 8	Bit 1																Bit 0															
Dat a (11)	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Cod e8	1	0	0	1	0	1	1	0	1	0	0	1	0	1	1	0	1	0	0	1	0	1	1	0	1	0	0	1	0	1	1	0
data	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
code 8	1	-	-	1	-	1	1	-	1	-	-	1	-	1	1	-	1	-	-	1	-	1	1	-	1	-	-	1	-	1	1	-
data 8x code 8	1	-	-	1	-	1	1	-	1	-	-	1	-	1	1	-	1	-	-	1	-	1	1	-	1	-	-	1	-	1	1	-

Us 1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-		
Us 2	-	1	-	1	-	1	-	1	-	1	-	1	-	1	-	1	1	-	1	-	1	-	1	-	1	-	1	-	1	-	1	-	
Us 3	1	1	-	-	1	1	-	-	1	1	-	-	1	1	-	-	-	-	1	1	-	-	1	1	-	-	1	1	-	-	1	1	
Us 4	1	-	-	1	1	-	-	1	1	-	-	1	1	-	-	1	1	-	-	1	1	-	-	1	1	-	-	1	1	-	-	1	1
Us 5	-	-	-	-	1	1	1	1	-	-	-	-	1	1	1	1	-	-	-	-	1	1	1	1	-	-	-	-	1	1	1	1	
Us 6	-	1	-	1	1	-	1	-	-	1	-	1	1	-	1	-	1	-	1	-	-	1	-	1	1	-	1	-	-	1	-	1	
Us 7	1	1	-	-	-	-	1	1	1	1	-	-	-	-	1	1	-	-	1	1	1	1	1	-	-	-	-	1	1	1	1	-	
Us 8	1	-	-	1	-	1	1	-	1	-	-	1	-	1	1	-	1	-	-	1	-	1	1	-	1	-	1	-	1	-	1	1	
su m	0	0	-	0	0	0	0	0	0	0	-	0	0	0	0	0	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

VI. BLOCK DIAGRAM OF THE PSEDO-NOISE CODE GENERATOR (PNCG)

To generate Walsh codes that are mutually orthogonal and each (16 bits) long ,we use the Hadamard matrix (H16) of size (16x16) shown in the matrix (3) [6].  
 The first row is taken as the code for the first user , the second row as the code for the second user , the thrid row as the code for the thrid user , and the eighth row as the code for the eighth user.

$$H_{16} = \begin{bmatrix} H_8 & H_8 \\ H_8 & -H_8 \end{bmatrix} = \begin{bmatrix} +1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 \\ +1 & -1 & +1 & -1 & +1 & -1 & +1 & -1 & +1 & -1 & +1 & -1 & +1 & -1 & +1 & -1 & +1 \\ +1 & +1 & -1 & -1 & +1 & +1 & -1 & -1 & +1 & +1 & -1 & -1 & +1 & +1 & -1 & -1 & +1 \\ +1 & -1 & -1 & +1 & +1 & -1 & -1 & +1 & +1 & -1 & -1 & +1 & +1 & -1 & -1 & +1 & +1 \\ +1 & +1 & +1 & +1 & -1 & -1 & -1 & -1 & +1 & +1 & +1 & +1 & -1 & -1 & -1 & -1 & +1 \\ +1 & -1 & +1 & -1 & -1 & +1 & -1 & +1 & +1 & -1 & -1 & +1 & -1 & -1 & +1 & -1 & +1 \\ +1 & +1 & -1 & -1 & +1 & +1 & +1 & +1 & -1 & -1 & -1 & -1 & -1 & -1 & +1 & +1 & +1 \\ +1 & -1 & -1 & +1 & +1 & -1 & -1 & +1 & -1 & +1 & +1 & -1 & -1 & -1 & +1 & +1 & -1 \\ +1 & +1 & +1 & +1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & +1 & +1 & +1 \\ +1 & -1 & +1 & -1 & -1 & +1 & -1 & +1 & -1 & +1 & -1 & +1 & -1 & +1 & -1 & +1 & -1 \\ +1 & +1 & -1 & -1 & -1 & -1 & +1 & +1 & -1 & -1 & +1 & +1 & +1 & +1 & -1 & -1 & -1 \\ +1 & -1 & -1 & +1 & -1 & +1 & +1 & -1 & -1 & +1 & +1 & -1 & +1 & -1 & -1 & -1 & +1 \end{bmatrix} \tag{3}$$

For designing the hardware of pseudo-random code generator (PNCG), (16-bit) shift registers are used with parallel output and shift input, where the binary code value is loaded into the register [7].

The last bit of register is connected to the shift input so that shifting begins with each clock pulse, allowing the desired code to be obtained.

The block diagram of the PNCG for eight users is shown in the figure (5).

To generate the code  $c_1=(1111111111111111)$ , the value (65535) is loaded into the first shift register.

To generate the code  $c_2=(1010101010101010)$ , the value (21845) is loaded into the second shift register.

To generate the code  $c_3=(1100110011001100)$ , the value (13107) is loaded into the third shift register.

To generate the code  $c_4=(1001100110011001)$ , the value (39321) is loaded into the fourth shift register.

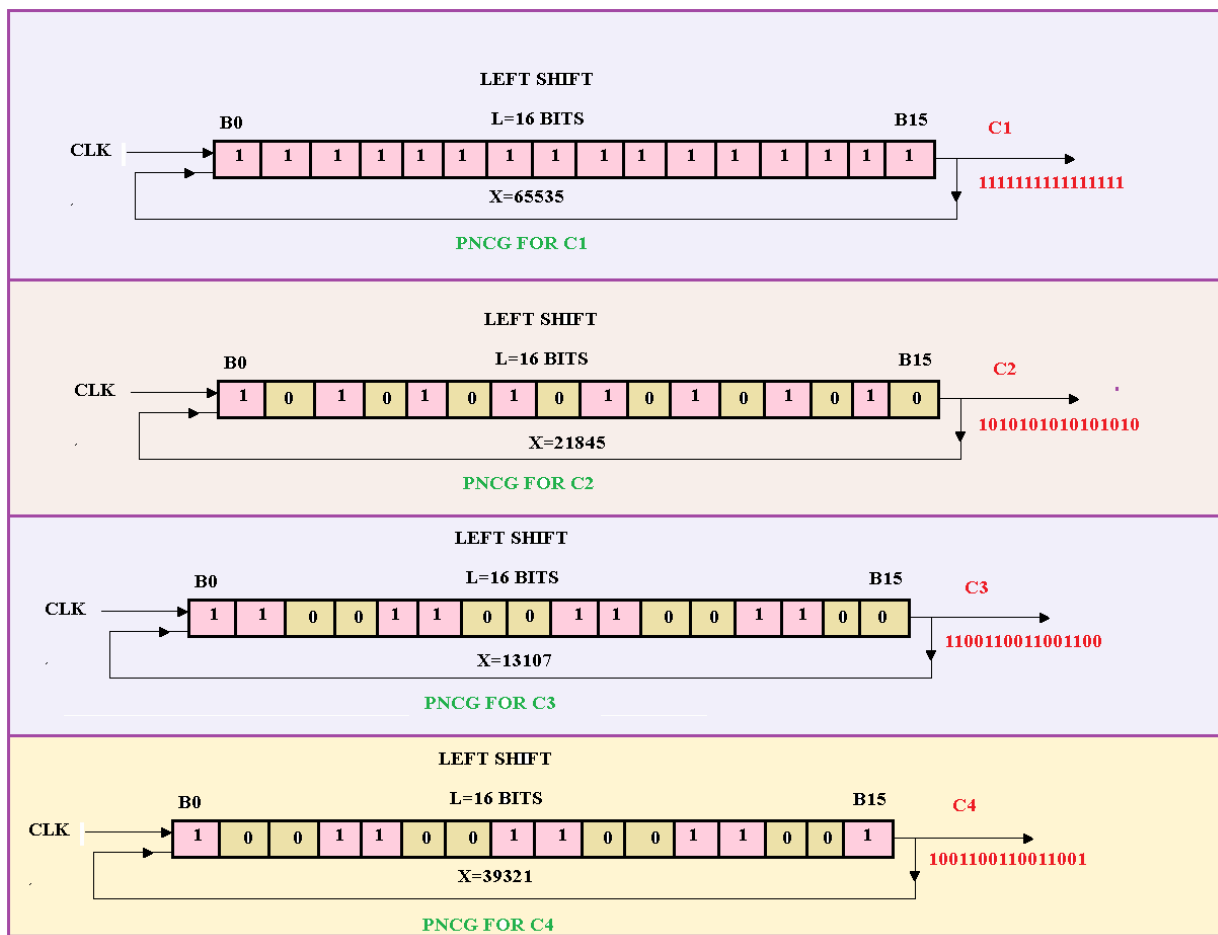
To generate the code  $c_5=(1111000011110000)$ , the value (3855) is loaded into the fifth shift register.

To generate the code  $c_6=(1010010110100101)$ , the value (42405) is loaded into the sixth shift register.

To generate the code  $c_7=(1100001111000011)$ , the value (50115) is loaded into the seventh shift register.

To generate the code  $c_8=(1001011010010110)$ , the value (26985) is loaded into the eighth shift register.

The block diagram of the PNCG implemented in the Quartus 9.I environment is shown in the figure (6).



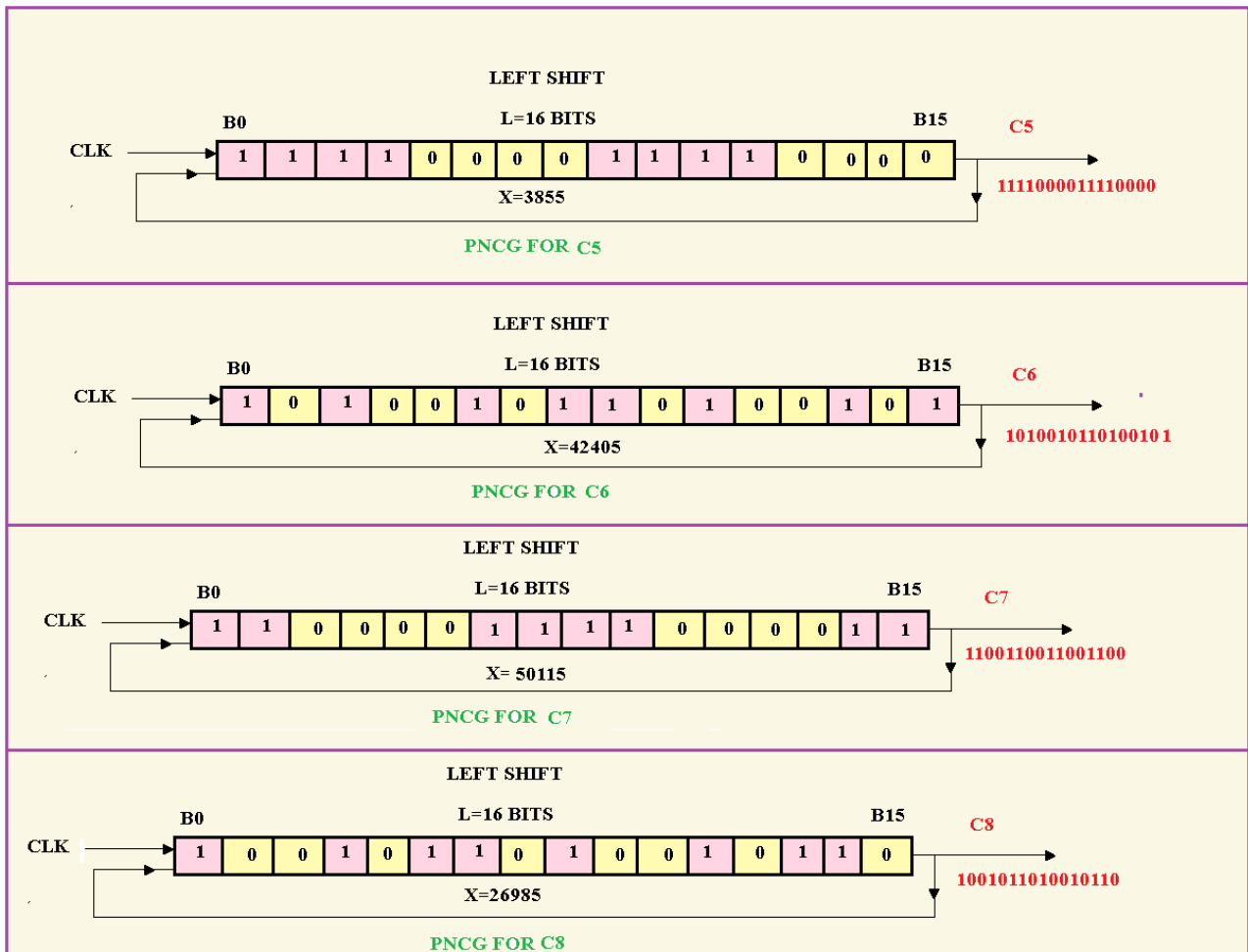


Figure (5): block diagram of the PNCG for eight users

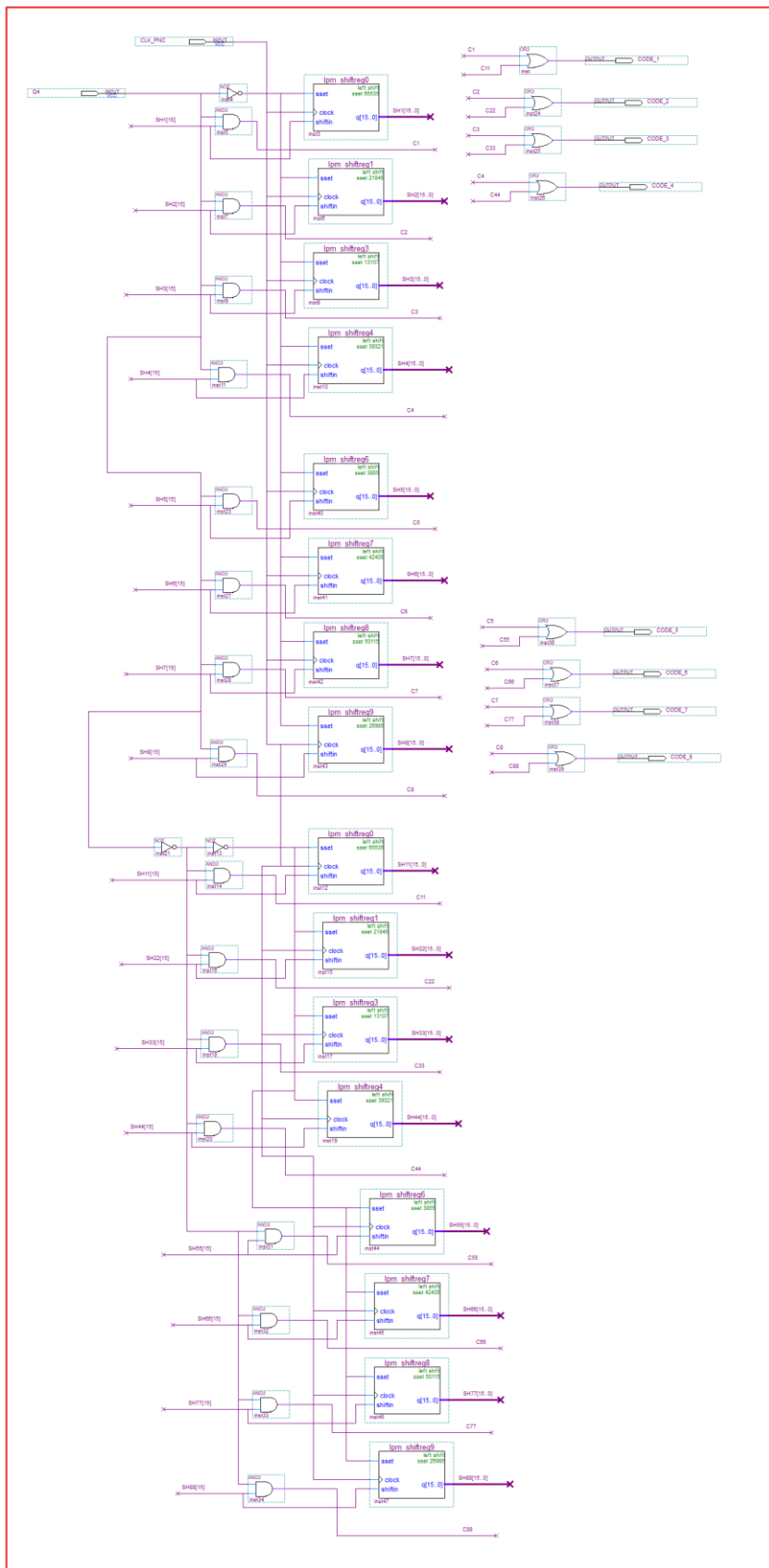


Figure (6): block diagram of the PNCG for eight users in Quartus II 9.1 design environment

The results of the PNCG designed for generating eight Walsh codes using an FPGA chip placed on the DE-1 Education and Development board are shown in the figure (7).

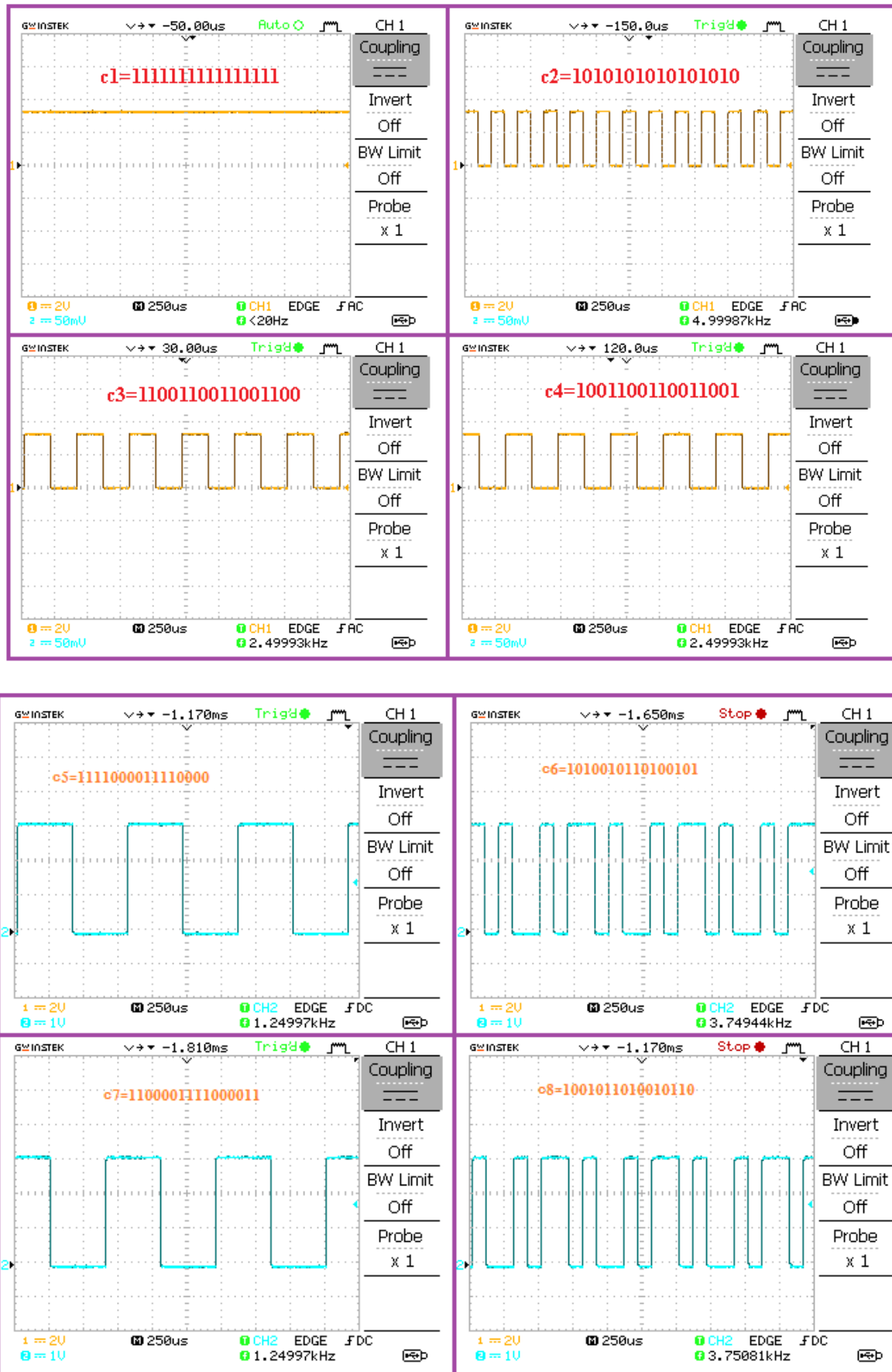


Figure (7): Walsh codes system DSSS: c1,c2,c3,c4,c5,c6,c7,c8 in the time domain

**VII. BLOCK DIAGRAM OF THE CLOCK AND DATA GERATOR**

To generate clock pulses with a frequency of (10 KHz) for all system components, a direct digital frequency synthesizer (DDFS) with an input frequency of (50 MHz) was used.

Frequency code for DDFS is calculated according to the mathematical formula [8]:

$$F_{CLK} = \frac{L \cdot F_{SYS}}{2^n} \Rightarrow L = \frac{2^n \times F_{CLK}}{F_{SYS}} \quad (4)$$

Where: (L) frequency code of DDFS, (n) bits number of phase accumulator DDFS, ( $F_{rys} = 50\text{MHz}$ ) the reference frequency from DE-1, ( $F_{olk} = 10\text{KHz}$ ) the clock frequency to be generated by DDFS.

$$L = \frac{2^n \times F_{CLK}}{F_{SYS}} = \frac{2^{32} \times 10}{50000} = 85899$$

To generate the data bits (d1, d2, d3, d4, d5, d6, d7, d8) for the eight users, a (5 –bit) binary counter operating from a (10 KHz) clock pulse was used.

The functional diagram of the clock pulse generator and data bits generator shown in the figure (8), and the functional diagram of the clock pulse generator in Quartus II 9.1 design environment shown in the figure (9).

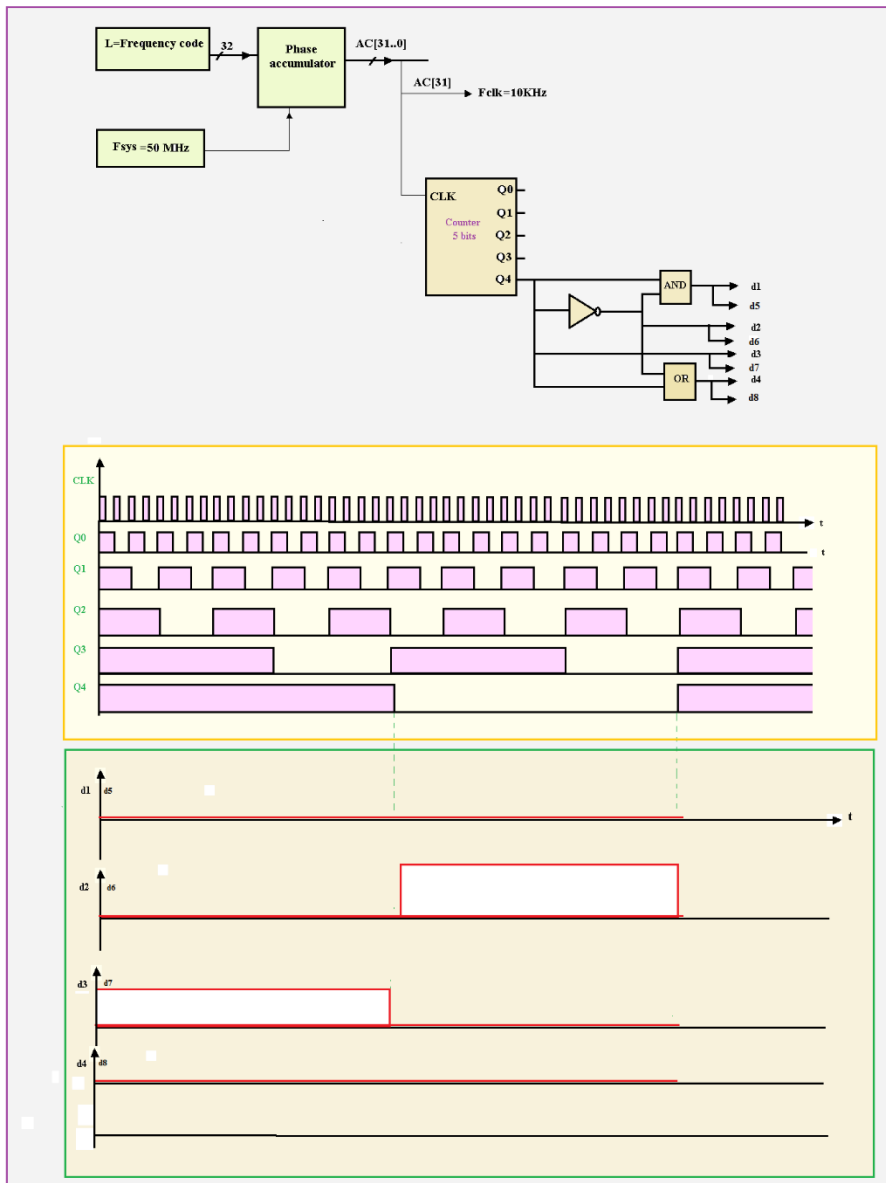


Figure (8): functional diagram of the clock pulse generator and data bits generator.

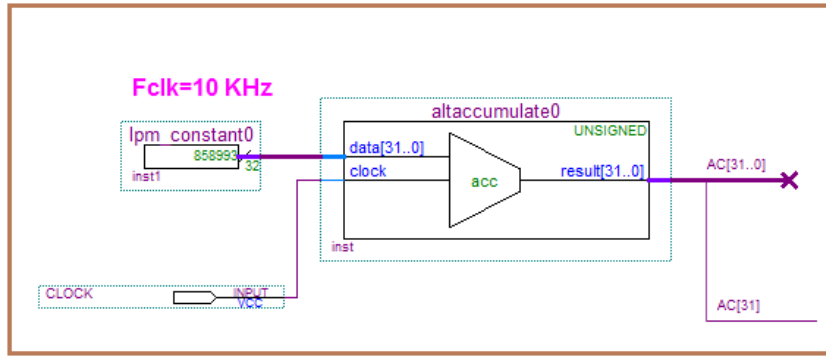


Figure (9): functional diagram of the clock pulse generator in the Quartus II 9.I design environment.

The practical designed results of the clock pulse generator and data bits generator using an FPGA chip located on the DE-1 board and using the Quartus 9.I design software environment are shown in figure (10) and figure (11).

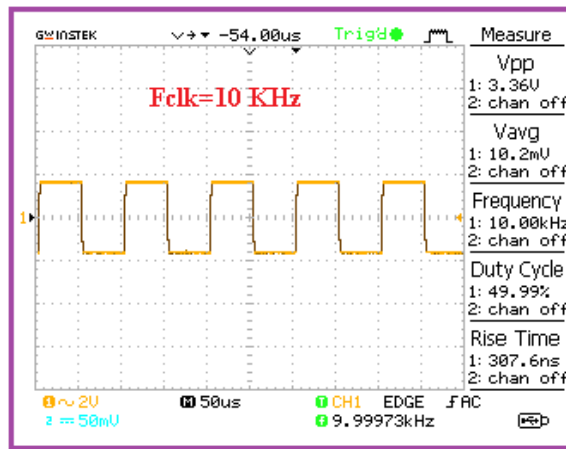


Figure (10):clock pulse with frequency 10 KHz

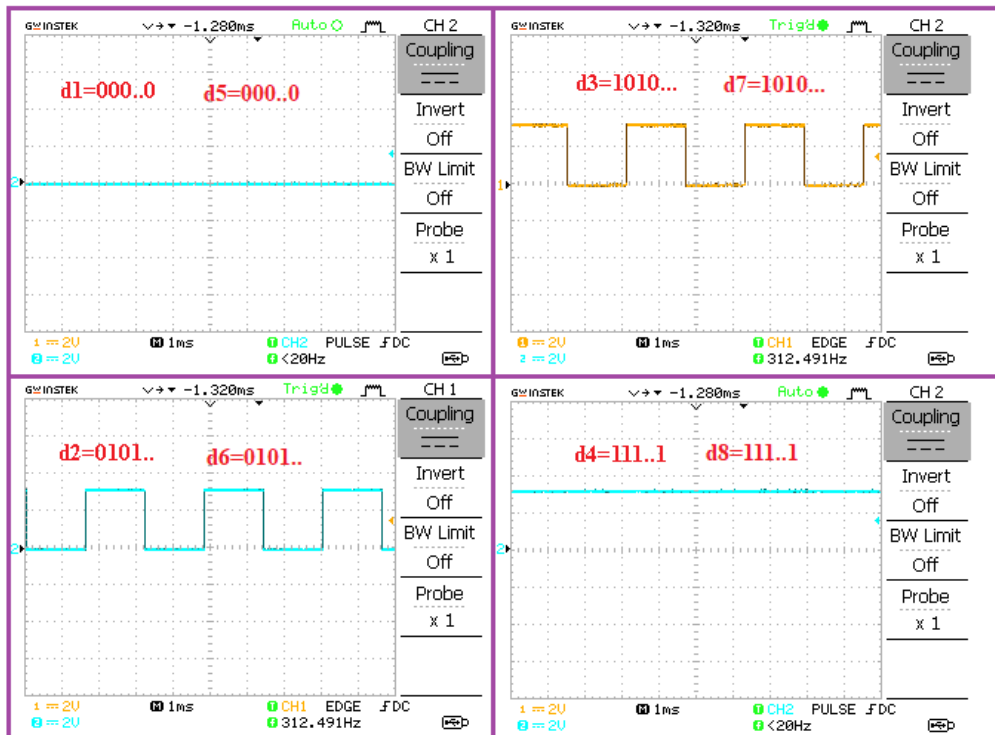
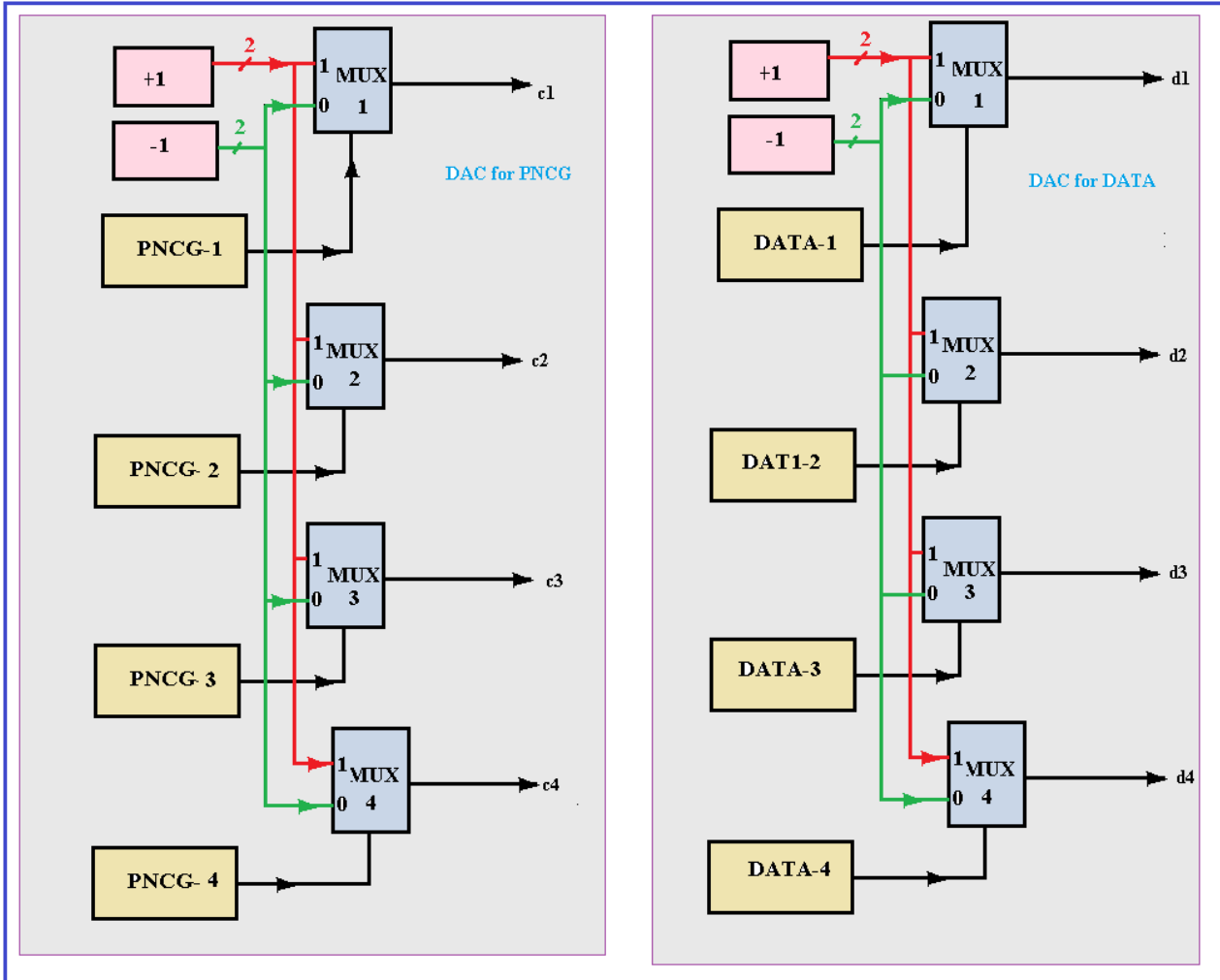


Figure (11):transmitting data bits d1,d2,d3,d4,d5,d6,d7,d8.

**VIII. BLOCK DIAGRAM OF THE DAC OF THE PNCG AND BINARY DATA GENERATOR**

The DAC's for PNCG and binary data generator is dedicated to converting the logic one to the value (+1) and converting the logic zero to (-1).

The block diagram of these converters is shown in the figure (12), and the block diagram in the Quartus II 9.I design environment is shown in the figure (13).



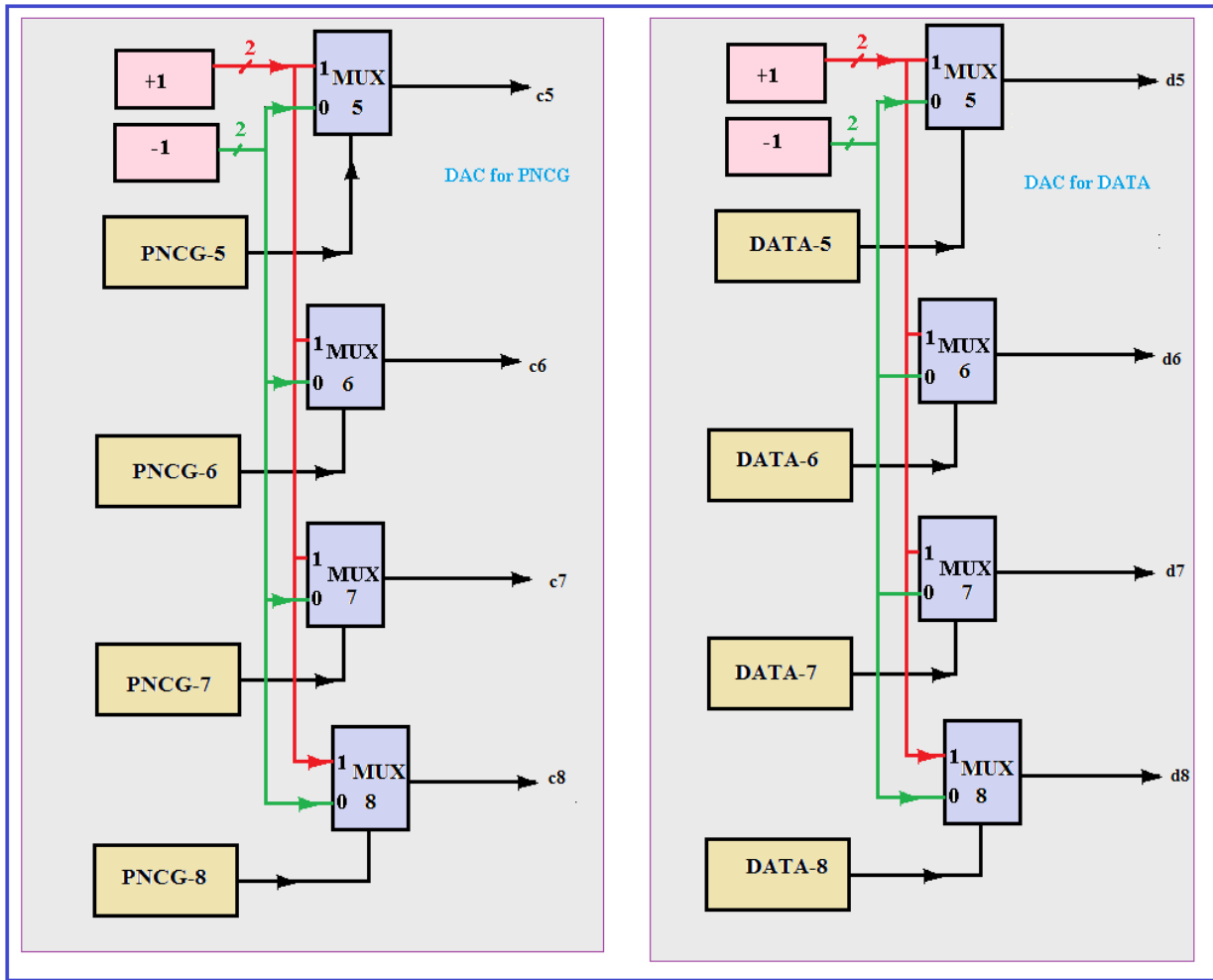
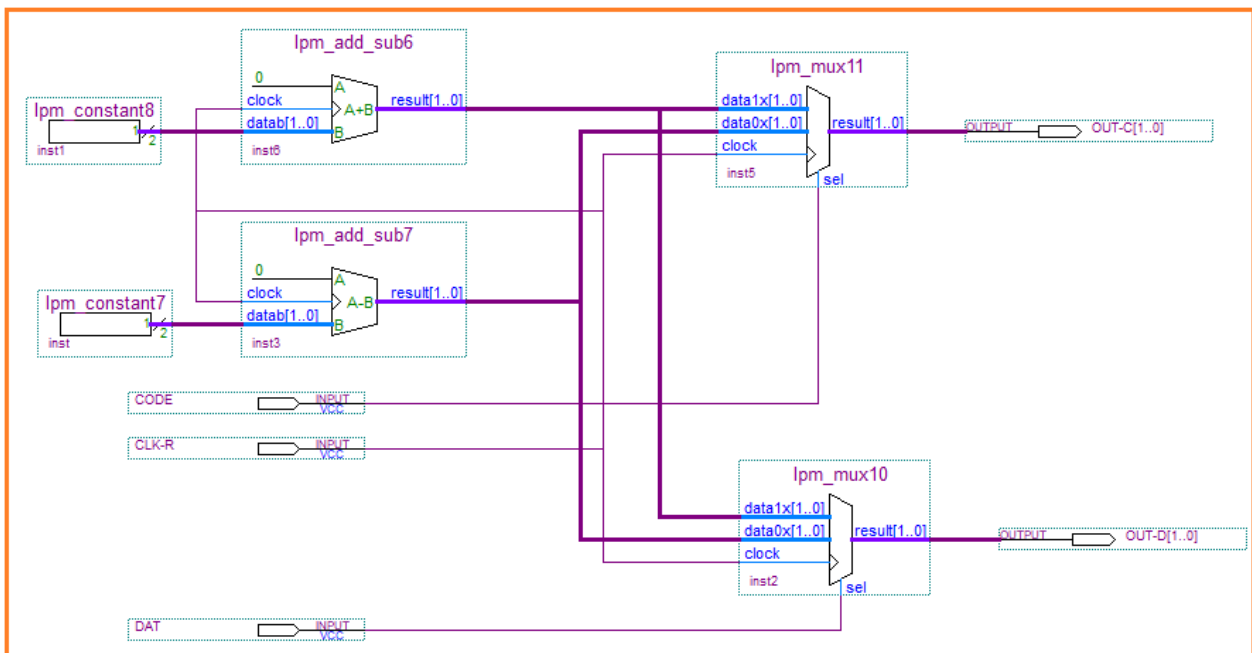


Figure (12): block diagram of the DAC for PNCG and binary data generator



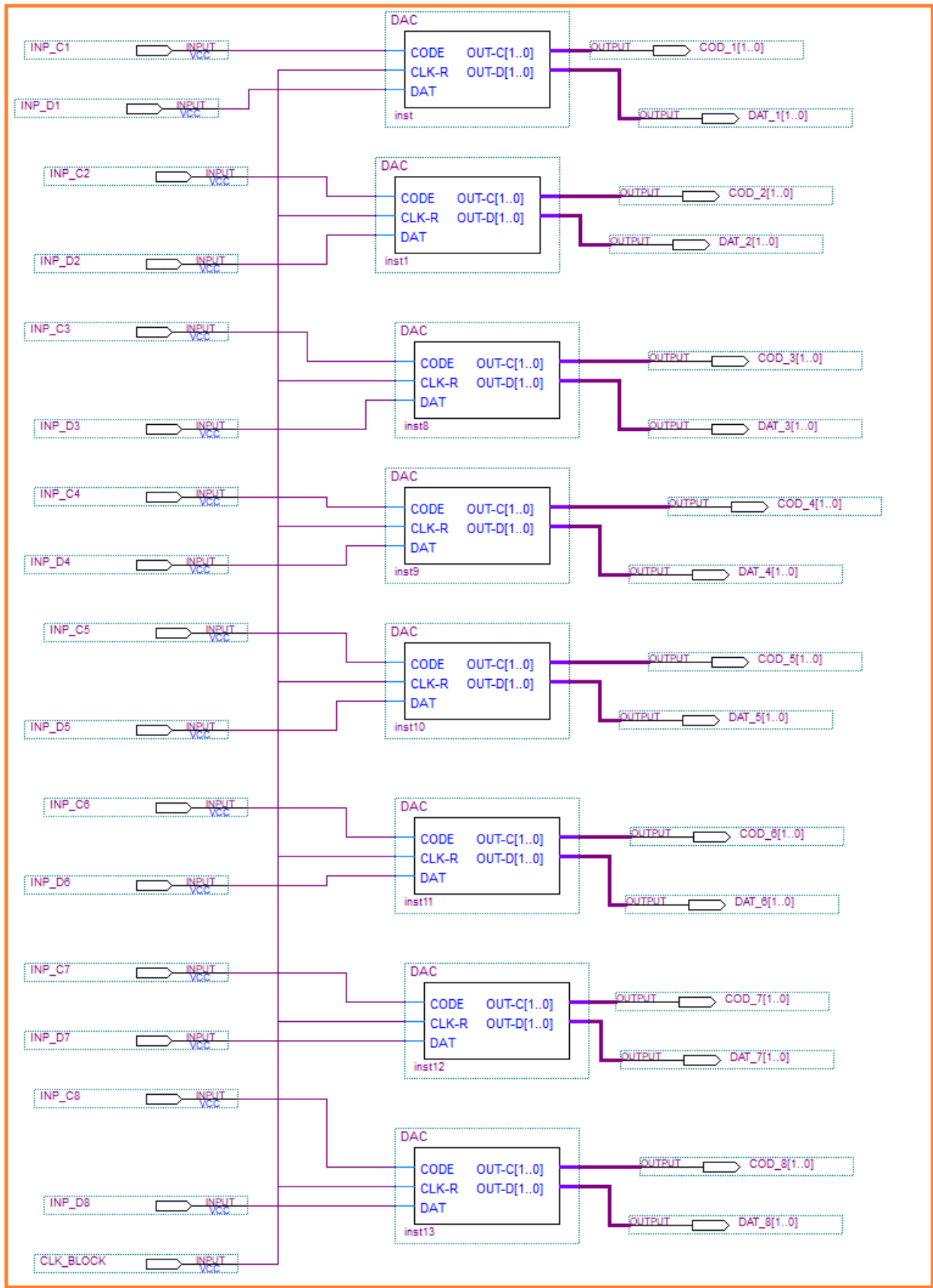


Figure (13): block diagram of the DAC for PNCG and binary data generator in Quartus II 9.1 design environment.

**IX. BLOCK DIAGRAM OF THE RECEIVER DSSS SYSTEM**

The received signal is given according to the following relation [9]:

$$RX(t) = TX(t) = \sum_{i=1}^{i=8} c_i(t) \cdot d_i(t) \quad (5)$$

To extract the desired user's signal (user j), the receiver signal is multiplied by this user's code  $c_{hi}(t)$  according to the following relation:

$$RX(t) \cdot c_j(t) = c_j(t) \cdot \sum_{i=1}^{i=8} c_i(t) \cdot d_i(t) \quad 6$$

$$RX(t) \cdot c_j(t) = c_j(t) \cdot c_1(t) \cdot d_1(t) + c_j(t) \cdot c_2(t) \cdot d_2(t) + c_j(t) \cdot c_3(t) \cdot d_3(t) + c_j(t) \cdot c_4(t) \cdot d_4(t) + c_j(t) \cdot c_5(t) \cdot d_5(t) + c_j(t) \cdot c_6(t) \cdot d_6(t) + c_j(t) \cdot c_7(t) \cdot d_7(t) + c_j(t) \cdot c_8(t) \cdot d_8(t)$$

$$c_j(t) \cdot c_i(t) = 0 : \text{if } i \neq j \quad (7)$$

$$c_j(t) \cdot c_i(t) = 1 : \text{if } i = j$$

$$\begin{aligned} \text{For } c_1(t) \Rightarrow RX(t) \cdot c_1(t) \\ = c_1(t) \cdot c_1(t) \cdot d_1(t) + c_1(t) \cdot c_2(t) \cdot d_2(t) + c_1(t) \cdot c_3(t) \cdot d_3(t) + c_1(t) \cdot c_4(t) \cdot d_4(t) \\ + c_1(t) \cdot c_5(t) \cdot d_5(t) + c_1(t) \cdot c_6(t) \cdot d_6(t) + c_1(t) \cdot c_7(t) \cdot d_7(t) + c_1(t) \cdot c_8(t) \cdot d_8(t) \end{aligned}$$

$$RX(t) \cdot c_1(t) = d_1(t) \quad (8)$$

$$\begin{aligned} \text{For } c_2(t) \Rightarrow RX(t)c_2(t) \\ = c_2(t) \cdot c_1(t) \cdot d_1(t) + c_2(t) \cdot c_2(t) \cdot d_2(t) + c_2(t) \cdot c_3(t) \cdot d_3(t) + c_2(t) \cdot c_4(t) \cdot d_4(t) \\ + c_2(t) \cdot c_5(t) \cdot d_5(t) + c_2(t) \cdot c_6(t) \cdot d_6(t) + c_2(t) \cdot c_7(t) \cdot d_7(t) + c_2(t) \cdot c_8(t) \cdot d_8(t) \end{aligned}$$

$$RX(t) \cdot c_2(t) = d_2(t) \quad (9)$$

$$\begin{aligned} \text{For } c_3(t) \Rightarrow RX(t)c_3(t) \\ = c_3(t) \cdot c_1(t) \cdot d_1(t) + c_3(t) \cdot c_2(t) \cdot d_2(t) + c_3(t) \cdot c_3(t) \cdot d_3(t) + c_3(t) \cdot c_4(t) \cdot d_4(t) \\ + c_3(t) \cdot c_5(t) \cdot d_5(t) + c_3(t) \cdot c_6(t) \cdot d_6(t) + c_3(t) \cdot c_7(t) \cdot d_7(t) + c_3(t) \cdot c_8(t) \cdot d_8(t) \end{aligned}$$

$$RX(t) \cdot c_3(t) = d_3(t) \quad (10)$$

$$\begin{aligned} \text{For } c_4(t) \Rightarrow RX(t)c_4(t) \\ = c_4(t) \cdot c_1(t) \cdot d_1(t) + c_4(t) \cdot c_2(t) \cdot d_2(t) + c_4(t) \cdot c_3(t) \cdot d_3(t) + c_4(t) \cdot c_4(t) \cdot d_4(t) \\ + c_4(t) \cdot c_5(t) \cdot d_5(t) + c_4(t) \cdot c_6(t) \cdot d_6(t) + c_4(t) \cdot c_7(t) \cdot d_7(t) + c_4(t) \cdot c_8(t) \cdot d_8(t) \end{aligned}$$

$$RX(t) \cdot c_4(t) = d_4(t) \quad (11)$$

$$\begin{aligned} \text{For } c_5(t) \Rightarrow RX(t)c_5(t) \\ = c_5(t) \cdot c_1(t) \cdot d_1(t) + c_5(t) \cdot c_2(t) \cdot d_2(t) + c_5(t) \cdot c_3(t) \cdot d_3(t) + c_5(t) \cdot c_4(t) \cdot d_4(t) \\ + c_5(t) \cdot c_5(t) \cdot d_5(t) + c_5(t) \cdot c_6(t) \cdot d_6(t) + c_5(t) \cdot c_7(t) \cdot d_7(t) + c_5(t) \cdot c_8(t) \cdot d_8(t) \end{aligned}$$

$$RX(t) \cdot c_5(t) = d_5(t) \quad (12)$$

$$\begin{aligned} \text{For } c_6(t) \Rightarrow RX(t)c_6(t) \\ = c_6(t) \cdot c_1(t) \cdot d_1(t) + c_6(t) \cdot c_2(t) \cdot d_2(t) + c_6(t) \cdot c_3(t) \cdot d_3(t) + c_6(t) \cdot c_4(t) \cdot d_4(t) \\ + c_6(t) \cdot c_5(t) \cdot d_5(t) + c_6(t) \cdot c_6(t) \cdot d_6(t) + c_6(t) \cdot c_7(t) \cdot d_7(t) + c_6(t) \cdot c_8(t) \cdot d_8(t) \end{aligned}$$

$$RX(t)c_6(t) = d_6(t) \quad (13)$$

$$\begin{aligned} \text{For } c_7(t) \Rightarrow RX(t)c_7(t) \\ = c_7(t) \cdot c_1(t) \cdot d_1(t) + c_7(t) \cdot c_2(t) \cdot d_2(t) + c_7(t) \cdot c_3(t) \cdot d_3(t) + c_7(t) \cdot c_4(t) \cdot d_4(t) \\ + c_7(t) \cdot c_5(t) \cdot d_5(t) + c_7(t) \cdot c_6(t) \cdot d_6(t) + c_7(t) \cdot c_7(t) \cdot d_7(t) + c_7(t) \cdot c_8(t) \cdot d_8(t) \end{aligned}$$

$$RX(t)c_7(t) = d_7(t) \quad (14)$$

$$\begin{aligned} \text{For } c_8(t) \Rightarrow RX(t)c_8(t) \\ = c_8(t).c_1(t).d_1(t) + c_8(t).c_2(t).d_2(t) + c_8(t).c_3(t).d_3(t) + c_8(t).c_4(t).d_4(t) \\ + c_8(t).c_5(t).d_5(t) + c_8(t).c_6(t).d_6(t) + c_8(t).c_7(t).d_7(t) + c_8(t).c_8(t).d_8(t) \end{aligned}$$

$$RX(t)c_8(t) = d_8(t) \quad (15)$$

To simplify and understand the receiving formation processes for all users ,we use the following tables:

For user 1																								
code 1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
sum	0	0	-8	0	0	0	0	0	0	0	-8	0	0	0	0	0	0	0	0	-8	0	0	0	
Cod e1 x sum	0	0	-8	0	0	0	0	0	0	0	-8	0	0	0	0	0	0	0	0	-8	0	0	0	
Sum all	-16												-16											
DA TA	$\frac{-16}{16} = -1 \Rightarrow \text{Data} = 0 \Rightarrow \text{Bit 1} = 0$												$\frac{-16}{16} = -1 \Rightarrow \text{Data} = 0 \Rightarrow \text{Bit 0} = 0$											

For user 2																								
code 2	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	
sum	0	0	-8	0	0	0	0	0	0	0	-8	0	0	0	0	0	0	0	0	-8	0	0	0	
Cod e2 x sum	0	0	-8	0	0	0	0	0	0	0	-8	0	0	0	0	0	0	0	0	8	0	0	0	
Sum all	-16												16											
DA TA	$\frac{-16}{16} = -1 \Rightarrow \text{Data} = 0 \Rightarrow \text{Bit 1} = 0$												$\frac{16}{16} = 1 \Rightarrow \text{Data} = 1 \Rightarrow \text{Bit 0} = 1$											

For user 3																								
code 3	1	1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	1	-1	
sum	0	0	-8	0	0	0	0	0	0	0	-8	0	0	0	0	0	0	0	0	-8	0	0	0	
Cod e3 x sum	0	0	8	0	0	0	0	0	0	0	8	0	0	0	0	0	0	0	0	-8	0	0	0	
Sum all	16												-16											
DA TA	$\frac{16}{16} = 1 \Rightarrow \text{Data} = 1 \Rightarrow \text{Bit 1} = 1$												$\frac{-16}{16} = -1 \Rightarrow \text{Data} = 0 \Rightarrow \text{Bit 0} = 0$											

For user 4																							
code 4	1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	1	-1	-1
sum	0	0	-8	0	0	0	0	0	0	0	-8	0	0	0	0	0	0	0	0	-8	0	0	0
Cod e4 x sum	0	0	8	0	0	0	0	0	0	0	8	0	0	0	0	0	0	0	0	8	0	0	0

Sum all	16																16															
DA TA	$\frac{16}{16} = 1 \Rightarrow \text{Data} = 1 \Rightarrow \text{Bit 1} = 1$																$\frac{16}{16} = 1 \Rightarrow \text{Data} = 1 \Rightarrow \text{Bit 0} = 1$															

For user 5																																								
code 5	1	1	1	1	-	-	-	-	1	1	1	1	-	-	-	-	1	1	1	1	-	-	-	-	1	1	1	1	-	-	-	-	1	1	1	1	-	-	-	-
sum	0	0	-8	0	0	0	0	0	0	0	0	-8	0	0	0	0	0	0	0	0	-8	0	0	0	0	0	0	0	0	0	0	0	0	-8	0	0	0	0	0	0
Code 5 x sum	0	0	-8	0	0	0	0	0	0	0	0	-8	0	0	0	0	0	0	0	0	-8	0	0	0	0	0	0	0	0	0	0	0	-8	0	0	0	0	0	0	0
Sum all	-16																-16																							
DA TA	$\frac{-16}{16} = -1 \Rightarrow \text{Data} = 0 \Rightarrow \text{Bit 1} = 0$																$\frac{-16}{16} = -1 \Rightarrow \text{Data} = -1 \Rightarrow \text{Bit 0} = 0$																							

For user 6																																							
code 6	1	-	1	-	-	1	-	1	1	-	1	-	-	1	-	1	1	-	1	-	-	1	-	1	1	-	1	-	-	1	-	-	1	-	1	1	-	1	
sum	0	0	-8	0	0	0	0	0	0	0	-8	0	0	0	0	0	0	0	0	-8	0	0	0	0	0	0	0	0	0	0	0	0	-8	0	0	0	0	0	0
Code 6 x sum	0	0	-8	0	0	0	0	0	0	0	-8	0	0	0	0	0	0	0	0	-8	0	0	0	0	0	0	0	0	0	0	0	-8	0	0	0	0	0	0	0
Sum all	-16																-16																						
DA TA	$\frac{-16}{16} = -1 \Rightarrow \text{Data} = 0 \Rightarrow \text{Bit 1} = 0$																$\frac{16}{16} = 1 \Rightarrow \text{Data} = 1 \Rightarrow \text{Bit 0} = 1$																						

For user 7																																								
code 7	1	1	-	-	-	-	1	1	1	1	-	-	-	-	1	1	1	1	-	-	-	-	1	1	1	1	-	-	-	-	1	1	1	1	-	-	-	-	1	1
sum	0	0	-8	0	0	0	0	0	0	0	-8	0	0	0	0	0	0	0	0	-8	0	0	0	0	0	0	0	0	0	0	0	-8	0	0	0	0	0	0	0	0
Code 7 x sum	0	0	8	0	0	0	0	0	0	0	8	0	0	0	0	0	0	0	0	8	0	0	0	0	0	0	0	0	0	0	0	-8	0	0	0	0	0	0	0	0
Sum all	16																-16																							
DA TA	$\frac{16}{16} = 1 \Rightarrow \text{Data} = 1 \Rightarrow \text{Bit 1} = 1$																$\frac{-16}{16} = -1 \Rightarrow \text{Data} = -1 \Rightarrow \text{Bit 0} = 0$																							

For user 8																																								
code 8	1	-	-	1	-	1	1	-	1	-	-	1	-	1	1	-	1	1	-	1	-	1	1	-	1	1	-	1	-	-	1	-	1	1	-	1	1	-	1	
sum	0	0	-8	0	0	0	0	0	0	0	-8	0	0	0	0	0	0	0	0	-8	0	0	0	0	0	0	0	0	0	0	0	-8	0	0	0	0	0	0	0	0
Code 8 x sum	0	0	8	0	0	0	0	0	0	0	8	0	0	0	0	0	0	0	0	8	0	0	0	0	0	0	0	0	0	0	0	8	0	0	0	0	0	0	0	0
Sum all	16																16																							
DA TA	$\frac{16}{16} = 1 \Rightarrow \text{Data} = 1 \Rightarrow \text{Bit 1} = 1$																$\frac{16}{16} = 1 \Rightarrow \text{Data} = 1 \Rightarrow \text{Bit 0} = 1$																							

The block diagram of the DSSS System Receiver for eight users in the baseband frequency domain is shown in the figure (14) and the block diagram of digital delay line in Quartus II 9.1 design environment is shown in the figure (15).

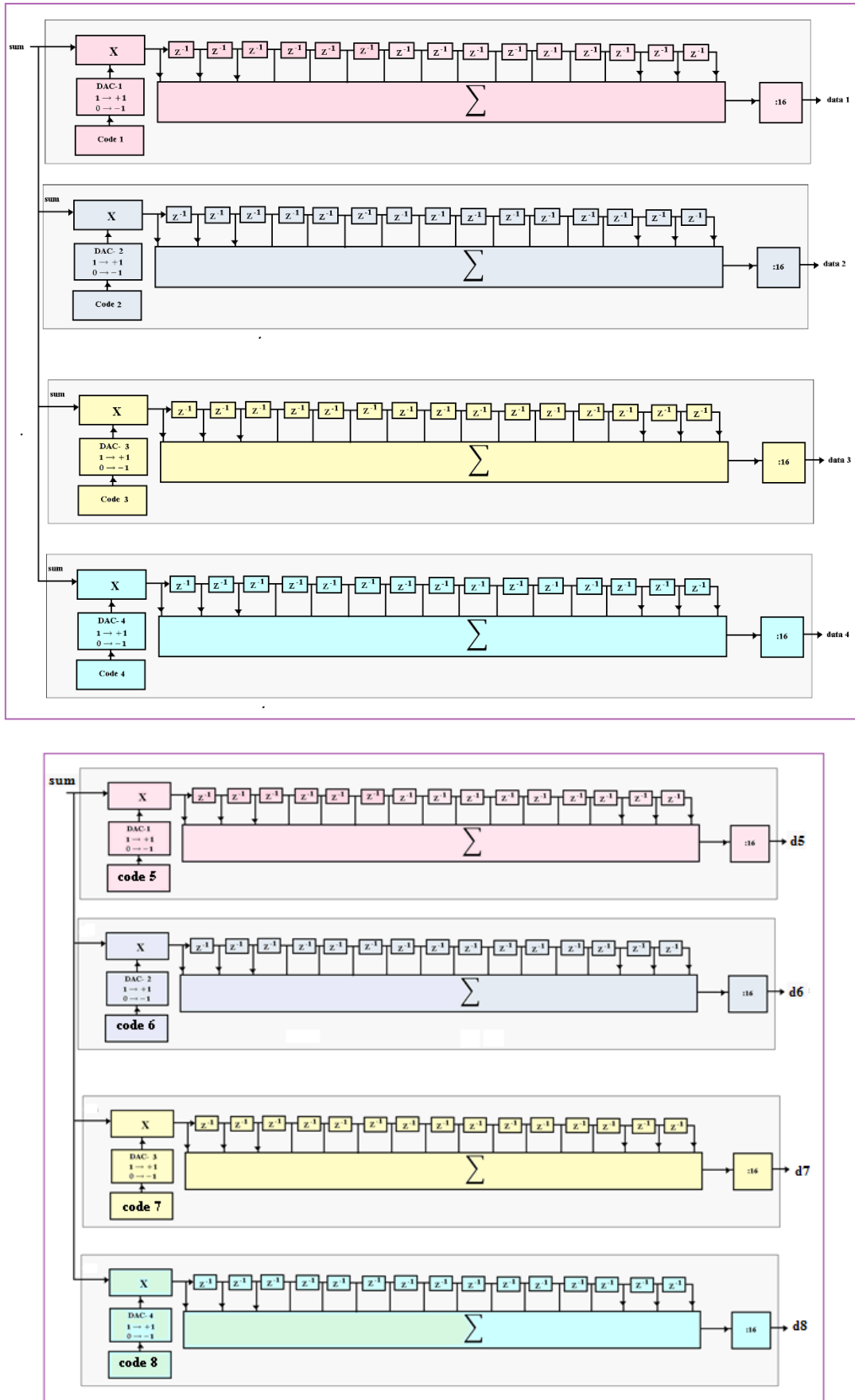


Figure (14): block diagram of the DSSS system receiver for eight users.

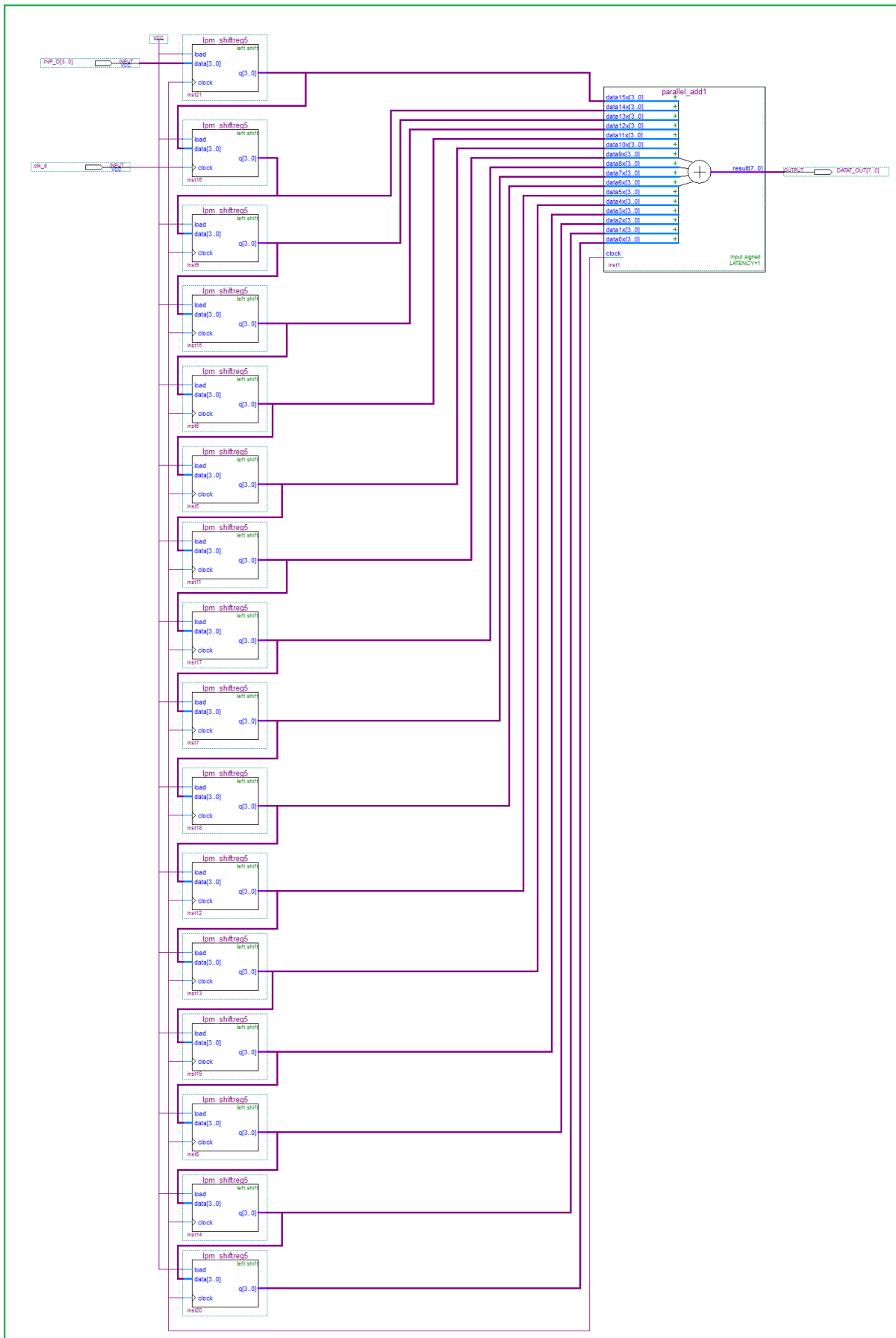


Figure (15): block diagram of digital delay line in Quartus II 9.1 design environment

The block diagram of the multipliers of the sum signal and the Walsh code signals for eight users in the baseband frequency domain in Quartus II 9.I design environment is shown in the figure (16).

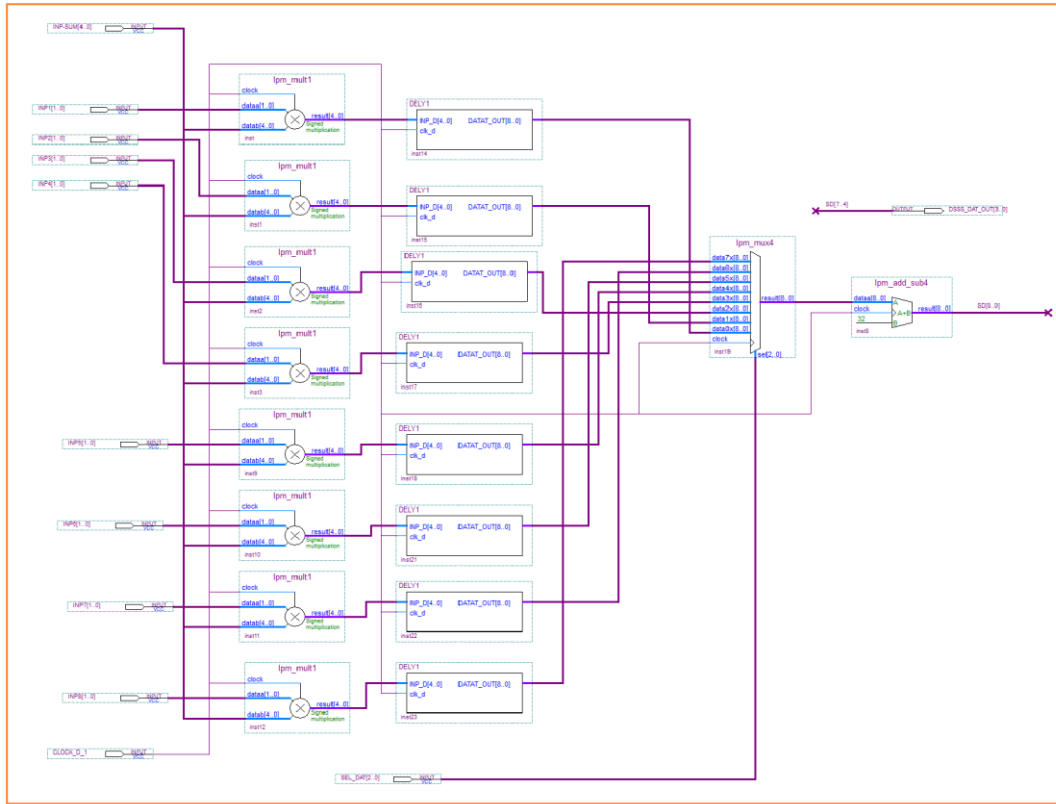


Figure (16): The block diagram of the multipliers of the sum signal and the Walsh codes for eight users in Quartus II 9.I.

The practical designed results of the data receiver DSSS system using an FPGA chip located on the DE-1 board and using the Quartus 9.I design software environment are shown in figure (17).

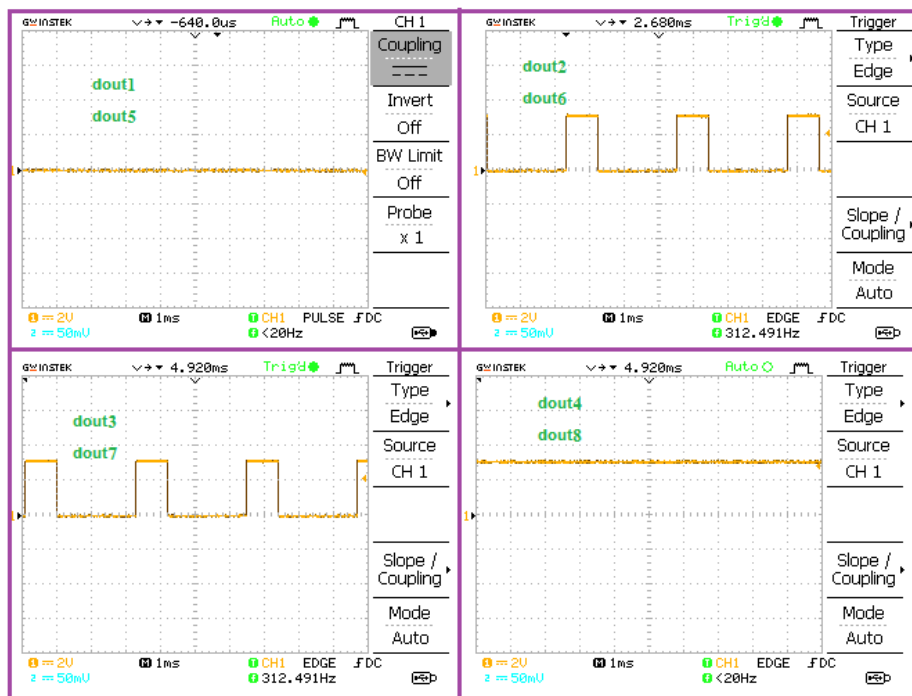


figure (17) ): receiving data bits  $\hat{d}_1(t), \hat{d}_2(t), \hat{d}_3(t), \hat{d}_4(t), \hat{d}_5(t), \hat{d}_6(t), \hat{d}_7(t), \hat{d}_8(t)$

## X. CONCLUSION AND RESULTS

Based on the theoretical analysis and practical application conducted in this research, some important points can be drawn, summarized as follows:

- The practical results obtained are consistent with the theoretical results, which proves the effectiveness and accuracy of digital designs.
- The use of FPGAs in the design process allows for easy modification and improvement of the design and its specifications upon request, simply by changing the software design.
- The number of users in the transmission can be increased to (12, 16), or more, increasing the system's capacity.
- The length of the user code can be increased, which increases the system's ability to distinguish between users, thus improving the quality of transmission and reception, reducing interference between signals, and improving the signal-to-noise ratio at the output of the digital delay line, which acts as a signal compressor.

## REFERENCES

- [1]. Hamsa A. ABDULLAH , Raya K. MOHAMMED, DSSS-DRMC BASED SECURE TRANSMISSION WITH FPGA IMPLEMENTATION ,U.P.B. Sci. Bull., Series C,Vol. 83, Issue 1, 2021 ISSN 2286-3540.
- [2]. Dr. Kamal Aboutabikh , “Design And Implementation Of a Digital Direct Sequence Spread Spectrum (DSSS) System For Four Users Using FPGA”, International Multidisciplinary Research Journal Reviews (IMRJR) , Volume 2, Issue 10, October 2025.
- [3]. [www.altera.com](http://www.altera.com).
- [4]. Volnei A. Pedroni, “Circuit Design With VHDL”, MIT Press Cambridge, Massa- chusetts London, England (2004) 364.
- [5]. M .Ali , "Implementation of a Software Defined Spread Spectrum modem" , 2012.
- [6]. Theodore K.F , Haddamard Matrices :Construction Using Number Theory and Algebra ,2019.
- [7]. Afaq Ahmad, Sayyid Samir Al-Busaidi and Mufeed Juma Al-Musharafi .On Properties of PN Sequences
- [8]. Generated by LFSR – a Generalized Study and Simulation Modeling. Indian Journal of Science and Technology-2013.
- [9]. GOLDBERG B. 1999- “Digital Frequency Synthesis Demystified”, LLH Technology Publishing, united states, 334.
- [10]. Don Torrieri, "Principles of Spread –Spectrum communication system", 2005.

## BIOGRAPHY



**Dr. Kamal Aboutabikh** holds a PhD in communication engineering in 1988 from the USSR, university of communication in Leningrad, holds a degree assistant professor in 2009 from Aleppo university.

Lecturer at Department of Biomedical Engineering, Al Andalus University for Medical Sciences-Syria, Tishreen University-Syria, Corduba Private University- Syria, Kassala University-Sudan and Ittihad Private University- Syria.

Published a lot of research’s in the field of digital communication and digital signal processing in the universities of Syria and in the European and Indian journals.

Working in the field of programming FPGA by using VHDL and design of Digital Filters.