e-ISSN 3108-026X
**International Multidisciplinary Research Journal Reviews (IMRJR)**
A Peer-reviewed journal
Volume 2, Issue 12, December 2025 | DOI 10.17148/IMRJR.2025.021202

# Design And Implementation Of a Digital Pseudo-Noise Walsh codes Using FPGA

**Dr. Kamal Aboutabikh**

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

**Abstract**: This paper presents the design and implementation of Walsh code generators with lengths of (8), (16), and (32-bits) bits using shift registers on a Cyclone II EP2C20F484C7 FPGA from ALTERA placed on education and development board DE-1. Walsh codes, as orthogonal sequences, enable efficient multi-user direct-sequence spread spectrum (DSSS) communication by minimizing cross-user interference. The core contribution is a shift-register-based generator architecture that outputs Walsh codes in a synchronized manner with the system clock, enabling scalable multi-user support while maintaining low resource usage on the Cyclone II FPGA. The work includes HDL descriptions, synthesis considerations on DE-1, and verification through functional and timing tests, demonstrating reliable code generation across all specified lengths and straightforward extension to additional lengths if required. The results indicate that the proposed approach provides a compact, deterministic, and easily verifiable implementation suitable for education, prototyping, and small-scale DSSS experiments.

**Keywords:** DSSS, Walsh Codes, PNWCG, FPGA.

## I. INTRODUCTION

Field Programmable Gate Arrays (FPGAs) offer flexible, parallel, and reconfigurable platforms for prototyping digital signal processing systems. Walsh codes provide a complete set of orthogonal binary sequences widely used in direct-sequence spread spectrum (DSSS) systems to separate multiple users sharing the same channel without self-interference. This work focuses on implementing Walsh code generators of lengths (8) , (16) , and (32-bits) using shift registers as the primary building blocks, implemented on the Terasic DE-1 development board featuring a Cyclone II FPGA. The use of shift registers enables a compact, deterministic hardware realization where codes are produced serially or in parallel, synchronized to a common clock. The DE-1 board is chosen for its accessibility and its suitability for educational demonstrations and rapid prototyping of DSSS hardware architectures. The paper outlines a practical HDL design, discusses timing and resource considerations on DE-1, and demonstrates straightforward verification steps to validate orthogonally and timing.

In paper [1], the DSSS is designed for pseudo-noise code generator with only (11) chips, while in this research we have (32) chips and it is possible to develop up to 64 chips.

In paper [2], the DSSS is designed for pseudo-noise code generator with only (16) chips, while in this research we have (32) chips and it is possible to develop up to 64 chips.

## II. RESEARCH IMPORTANCE AND ITS OBJECTIVES

### Research Importance:

The design of Walsh code generators using shift registers on FPGA chips installed on the DE-1 development and education board holds great significance in the field of digital communication systems, particularly for applications relying on Direct Sequence Spread Spectrum (DSSS) technology. These codes provide orthogonally, which enables multiple users to share the communication channel without interference, enhancing both the efficiency and quality of communication. Utilizing FPGA allows for high design flexibility, rapid implementation, and compatibility with practical educational and research environments, making this research highly valuable for students and researchers in communications engineering and digital electronics. The DE-1 board serves as an ideal platform for development and education, offering advanced resources and programming capabilities to implement and demonstrate complex digital systems like Walsh code generators practically and clearly.

### Research Objectives:

-To design and implement Walsh code generators of length (8) ,(16), and (32-bits) using shift registers on FPGA, prioritizing efficient utilization of the DE-1 board resource.

- To realize accurate, synchronizing, and repeatable Walsh code generation that meets essential geometric properties such as orthogonality and independence among codes.

-To study the impact of code length on generator performance in terms of resource consumption, generation speed, and suitability for various multi-user communication application.

-To provide a practical and applicable model for educational and research purposes in designing DSSS systems on FPGA using shift registers.

**-** To evaluate and test the implementations on the DE-1 board to ensure compliance with specifications and analyze the generators' performance in real conditions through simulation and field testing.

These objectives enable a deep understanding of how to build and implement Walsh code generators on FPGA and enhance the ability of engineers and students to design advanced, practical communication systems.

## III. RESEARCH MATERIALS AND ITS WAYS

To design, and test the PNWCG for 8,16 32 bits, 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.I 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. THE WALSH CODES MATRIXES FOR LENGTH 8,16 AND 32 BITS

To generate Walsh codes that are mutually orthogonal and each (8 bits) long ,we use the Hadamard matrix (H8) of size (8x8), matrix (H16) of size (16x16) and matrix (H32) of size (32x32) shown in the matrix (1) , (3) and (5) [5].

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…..etc.

$$H_8 = \begin{bmatrix} H_4 & H_4 \\ H_4 & \overline{H_4} \end{bmatrix} =$$

$$= \begin{bmatrix} +1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 \\ +1 & -1 & +1 & -1 & +1 & -1 & +1 & -1 \\ +1 & +1 & -1 & -1 & +1 & +1 & -1 & -1 \\ +1 & -1 & -1 & +1 & +1 & -1 & -1 & +1 \\ +1 & +1 & +1 & +1 & -1 & -1 & -1 & -1 \\ +1 & -1 & +1 & -1 & -1 & +1 & -1 & +1 \\ +1 & +1 & -1 & -1 & -1 & -1 & +1 & +1 \\ +1 & -1 & -1 & +1 & -1 & +1 & +1 & -1 \end{bmatrix} \qquad (1)$$

The sum of the products of any tow Walsh codes of length 8 bits equales zero according to the mathematical relaetionship [6]:

$$\sum_{i=1}^{8} W_i(a).W_i(b) = 0 \qquad (2)$$

Where $W_i(a)$ , $W_i(b)$ tow Walsh codes of length 8 bits, $(i = 1 \ldots 8)$ refers to each bit in the code.

$$H_{16} = \begin{bmatrix} H_8 & H_8 \\ H_8 & \overline{H_8} \end{bmatrix} =$$

$$= \begin{bmatrix}
+1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 \\
+1 & -1 & +1 & -1 & +1 & -1 & +1 & -1 & +1 & -1 & +1 & -1 & +1 & -1 & +1 & -1 \\
+1 & +1 & -1 & -1 & +1 & +1 & -1 & -1 & +1 & +1 & -1 & -1 & +1 & +1 & -1 & -1 \\
+1 & -1 & -1 & +1 & +1 & -1 & -1 & +1 & +1 & -1 & -1 & +1 & +1 & -1 & -1 & +1 \\
+1 & +1 & +1 & +1 & -1 & -1 & -1 & -1 & +1 & +1 & +1 & +1 & -1 & -1 & -1 & -1 \\
+1 & -1 & +1 & -1 & -1 & +1 & -1 & +1 & +1 & -1 & +1 & -1 & -1 & +1 & -1 & +1 \\
+1 & +1 & -1 & -1 & -1 & -1 & +1 & +1 & +1 & +1 & -1 & -1 & -1 & -1 & +1 & +1 \\
+1 & -1 & -1 & +1 & -1 & +1 & +1 & -1 & +1 & -1 & -1 & +1 & -1 & +1 & +1 & -1 \\
+1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\
+1 & -1 & +1 & -1 & +1 & -1 & +1 & -1 & -1 & +1 & -1 & +1 & -1 & +1 & -1 & +1 \\
+1 & +1 & -1 & -1 & +1 & +1 & -1 & -1 & -1 & -1 & +1 & +1 & -1 & -1 & +1 & +1 \\
+1 & -1 & -1 & +1 & +1 & -1 & -1 & +1 & -1 & +1 & +1 & -1 & -1 & +1 & +1 & -1 \\
+1 & +1 & +1 & +1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & +1 & +1 & +1 & +1 \\
+1 & -1 & +1 & -1 & -1 & +1 & -1 & +1 & -1 & +1 & -1 & +1 & +1 & -1 & +1 & -1 \\
+1 & +1 & -1 & -1 & -1 & -1 & +1 & +1 & -1 & -1 & +1 & +1 & +1 & +1 & -1 & -1 \\
+1 & -1 & -1 & +1 & -1 & +1 & +1 & -1 & -1 & +1 & +1 & -1 & +1 & -1 & -1 & +1
\end{bmatrix} \qquad (3)$$

The sum of the products of any tow Walsh codes of length 16 bits equales zero according to the mathematical relaetionship:

$$\sum_{i=1}^{16} W_i(a).W_i(b) = 0 \qquad (4)$$

Where $W_i(a)$ , $W_i(b)$ tow Walsh codes of length 16 bits, $(i = 1 \ldots .16)$ refers to each bit in the code.

$$H_{32} = \begin{bmatrix} H_{16} & H_{16} \\ H_{16} & \overline{H_{16}} \end{bmatrix} =$$

$$= \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 & -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 & -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 & +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 & -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 & -1 & +1 & +1 & -1
\end{bmatrix} \tag{5}$$

The sum of the products of any tow Walsh codes of length 32 bits equales zero according to the mathematical relaetionship:

$$\sum_{i=1}^{32} W_i(a).W_i(b) = 0 \qquad (6)$$

Where $W_i(a)$ , $W_i(b)$ tow Walsh codes of length 32 bits , $(i)$ refers to each bit in the code.

## V. BLOCK DIAGRAM OF THE DIGITAL PSEDO-NOISE WALSH CODE GENERATO FOR 8 BITS

For designing the hardware of pseudo-random code generator (PNWCG), (8-bit) shift registers are used with parallel output and shift input , where the binary code value is loaded into the rigester [7].
The last bit of register is connected to the shift input so that shifting begins with each clocl pulse , allowing the desired code to be obtaind.
Table (1) shows the values of (8) bits Walsh codes and the values recorden in the shift register of the PNWCG.

Table (1): PNWC for (8) bits

| code | values |
|---|---|
| c1=11111111 | 255 |
| c2=10101010 | 85 |
| c3=11001100 | 51 |
| c4=10011001 | 153 |
| c5=11110000 | 15 |
| c6=10100101 | 165 |
| c7=11000011 | 195 |
| c8=10010110 | 105 |

The block diagram of the PNWCG for (8) bits is shown in the figure (1) for c1,c2 and the block diagram of the PNWCG for (8) bits for c2 in the Quartus II 9.I design environment is shown in the figure (2).

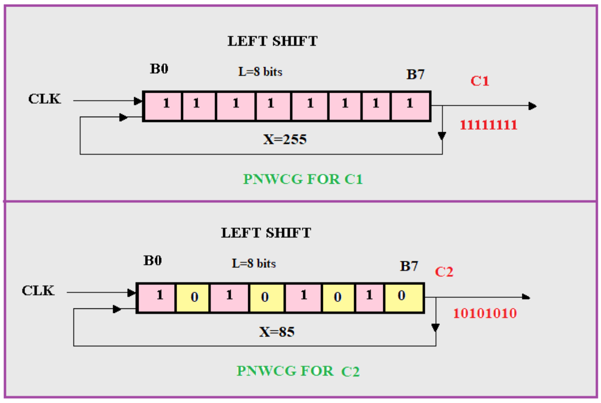

Figure (1): block diagram of the PNWCG for (8) bits for c1 and c2

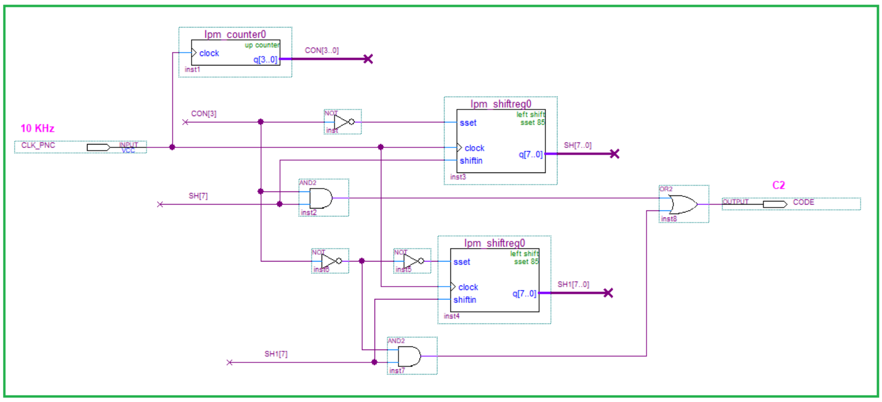

Figure (2): block diagram of the PNWCG for (8) bits for c2 in the Quartus II 9.I design environment.

The results of the PNWCG designed for (8) bits Walsh codes using an FPGA chip placed on the DE-1 Education and Development board are shown in the figure (3).

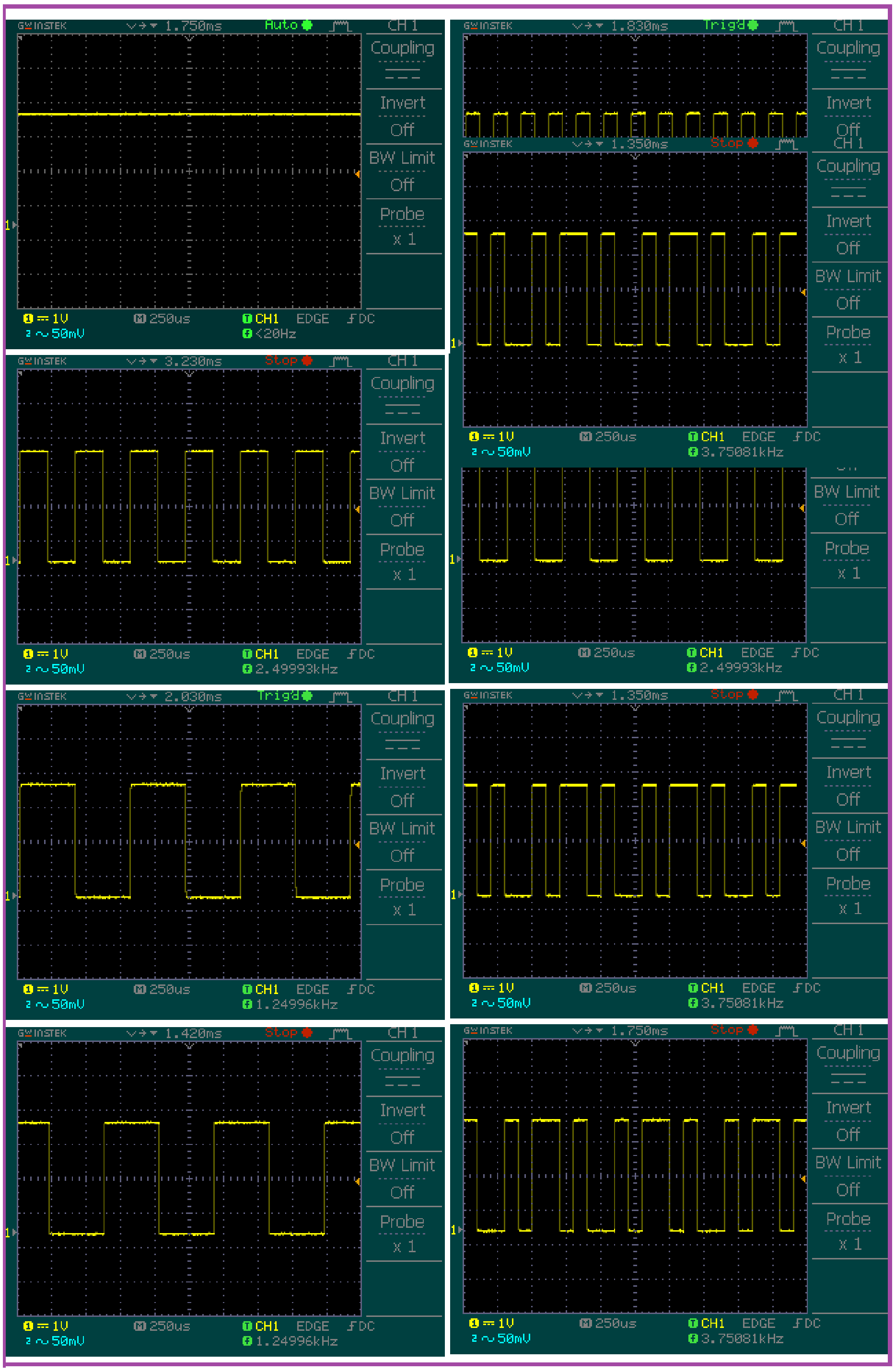

Figure (3):Walsh codes system DSSS for (8) bits: c1,c2,c3,c4,c5,c6,c7,c8 in the time domain

## VI. BLOCK DIAGRAM OF THE DIGITAL PSEDO-NOISE WALSH CODE GENERATOR FOR 16 BITS

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

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

Table (2) shows the values of (16) bits Walsh codes and the values recorden in the shift register of the PNWCG.

Table (2): PNWC for (16) bits

| code | values |
|---|---|
| c1=1111111111111111 | 65535 |
| c2=1010101010101010 | 21845 |
| c3=1100110011001100 | 13107 |
| c4=1001100110011001 | 39321 |
| c5=1111000011110000 | 3855 |
| c6=1010010110100101 | 42405 |
| c7=1100001111000011 | 50115 |
| c8=1001011010010110 | 26985 |
| c9=1111111100000000 | 255 |
| c10=1010101001010101 | 43605 |
| c11=1100110000110011 | 52275 |
| c12=1001100101100110 | 26265 |
| c13=1111000000001111 | 61455 |
| c14=1010010101011010 | 23205 |
| c15=1100001100111100 | 15555 |
| c16=1001011001101001 | 38505 |

The block diagram of the PNWCG for (16) bits is shown in the figure (4) for c1,c2,c3,c4 and the block diagram of the PNWCG for (16) bits for c2 in the Quartus II 9.I design environment is shown in the figure (5).

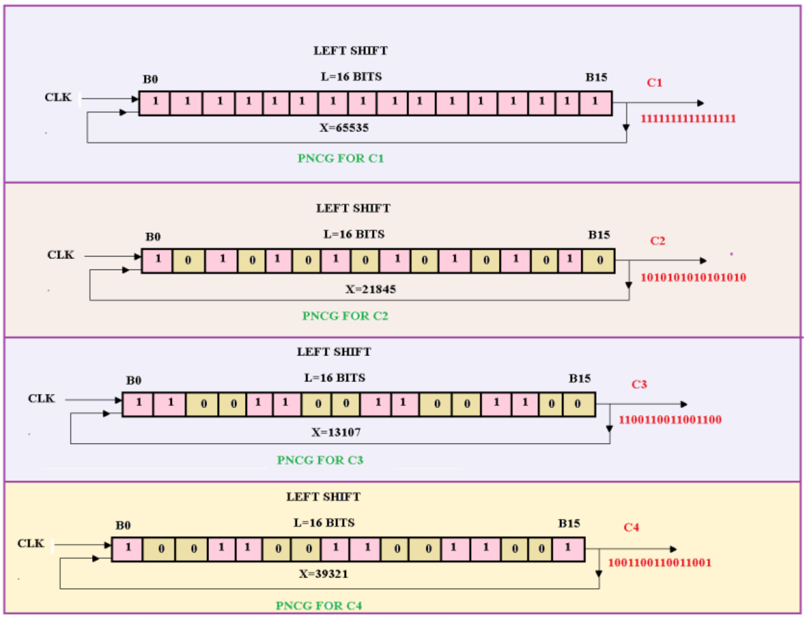

Figure (4): block diagram of the PNWCG for (16) bits for c1, c2, c3 and c4

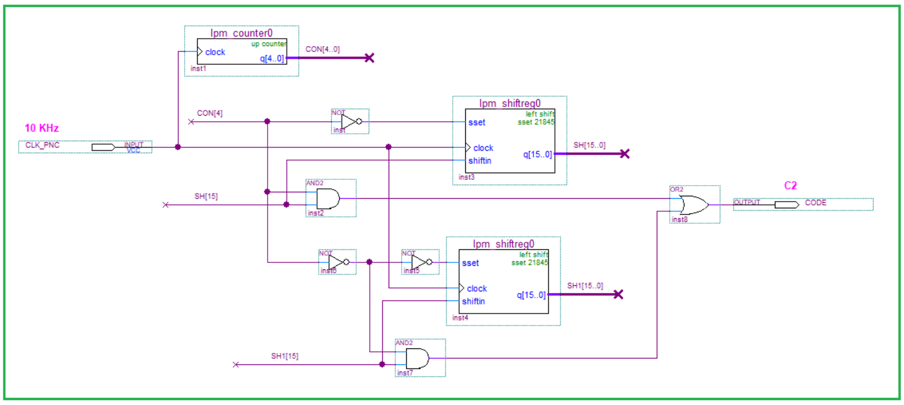

Figure (5): block diagram of the PNWCG for (16) bits for c2 in the Quartus II 9.I design environment.

The results of the PNWCG designed for (16) bits Walsh codes using an FPGA chip placed on the DE-1 Education and Development board are shown in the figure (6) and figure (7).

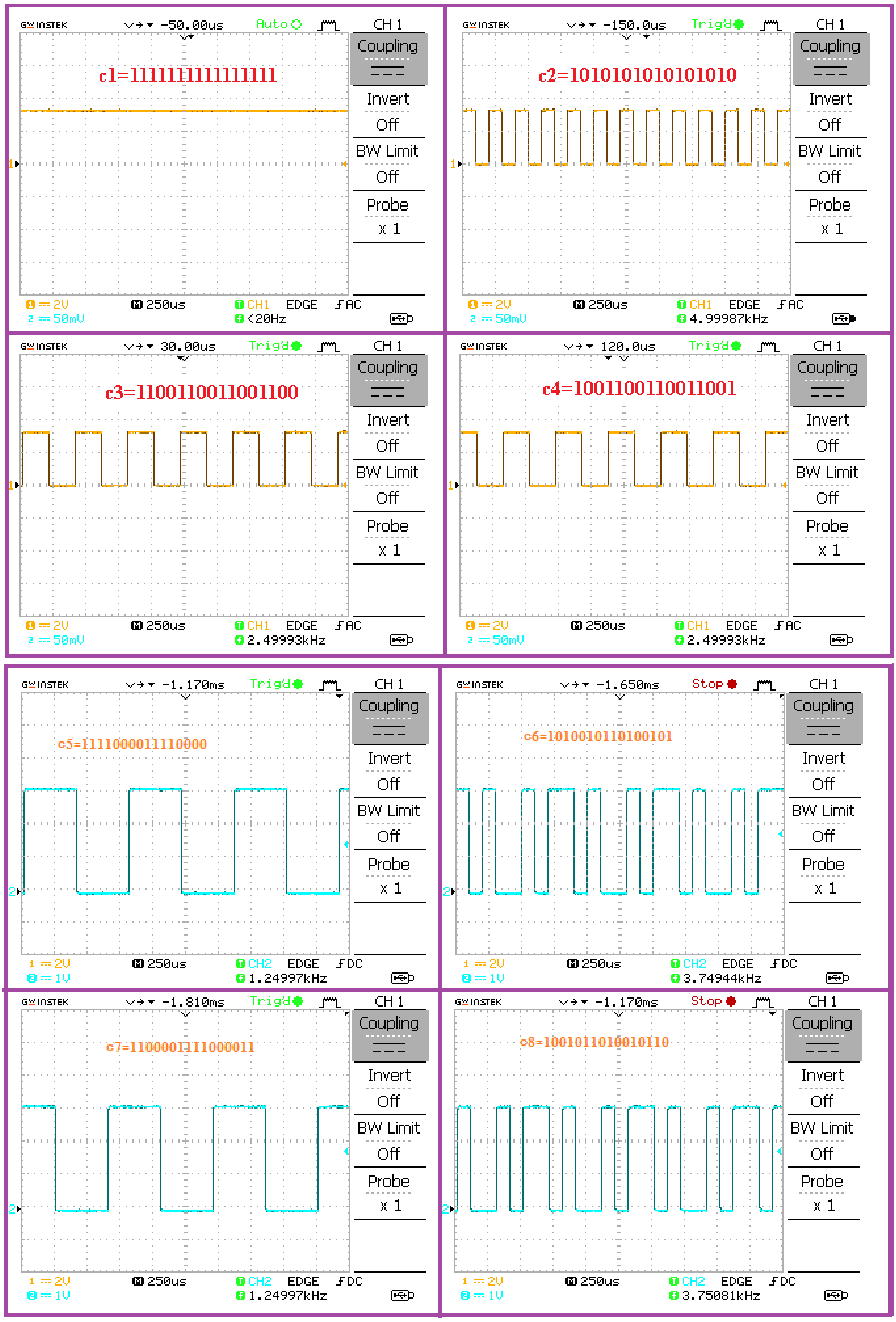

Figure (6):Walsh codes system DSSS for (16) bits: c1,c2,c3,c4,c5,c6,c7,c8 in the time domain

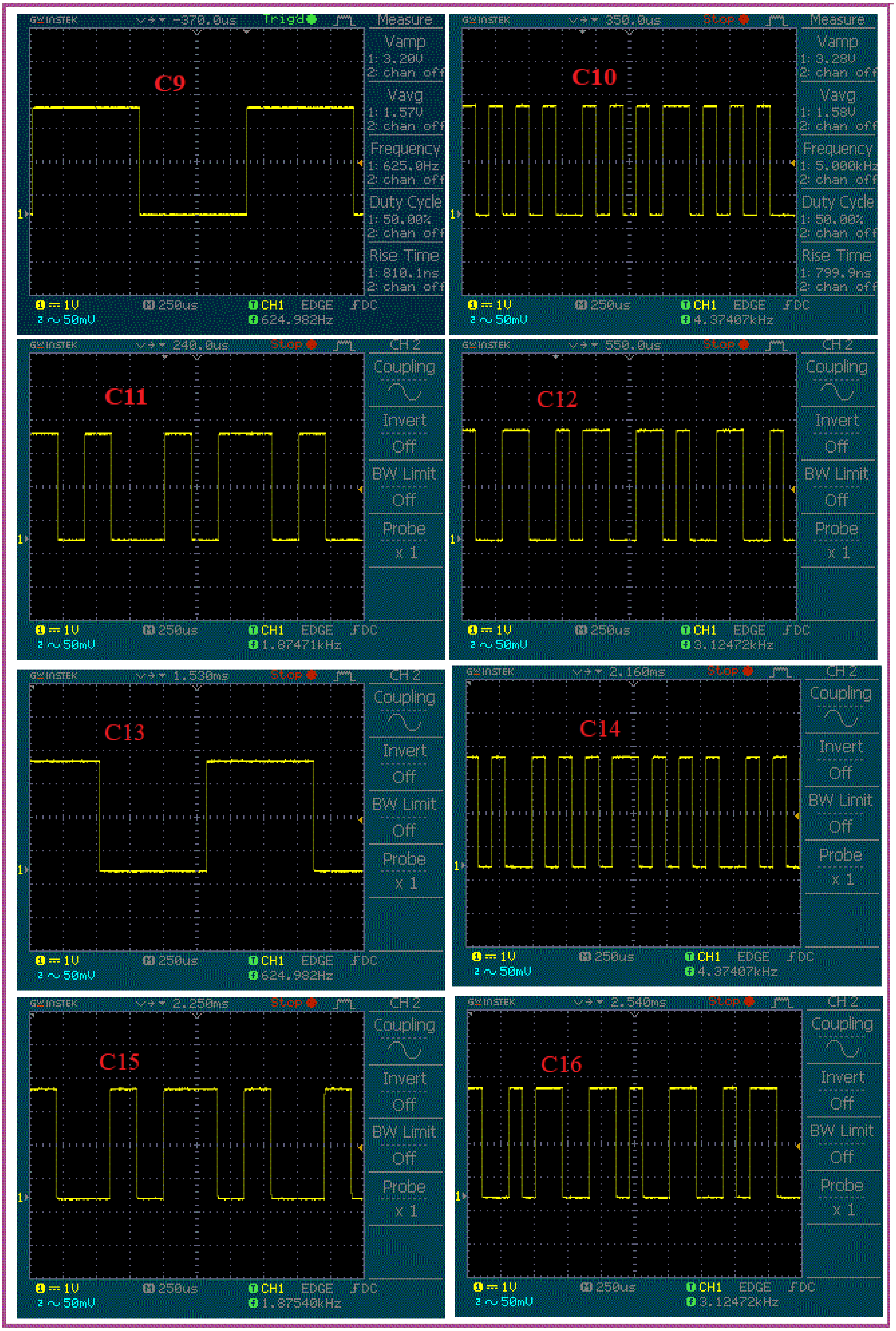

Figure (7):Walsh codes system DSSS: c9,c10,c11,c12,c13,c14,c15,c16 in the time domain

## VII. BLOCK DIAGRAM OF THE DIGITAL PSEDO-NOISE WALSH CODE GENERATOR FOR 32 BITS

For designing the hardware of pseudo-random code generator (PNWCG), (32-bit) shift registers are used with parallel output and shift input , where the binary code value is loaded into the rigester.
The last bit of register is connected to the shift input so that shifting begins with each clocl pulse , allowing the desired code to be obtaind.
Table (3) shows the values of (32) bits Walsh codes and the values recorden in the shift register of the PNWCG

Table (3): PNWC for (32) bits

| Code | Values |
|---|---|
| c1=11111111111111111111111111111111 | 4294967295 |
| c2=10101010101010101010101010101010 | 1431655765 |
| c3=11001100110011001100110011001100 | 858993459 |
| c4=10011001100110011001100110011001 | 2576980377 |
| c5=11110000111100001111000011110000 | 252654135 |
| c6=10100101101001011010010110100101 | 2779096485 |
| c7=11000011110000111100001111000011 | 3284386755 |
| c8=10010110100101101001011010010110 | 1768515945 |
| c9=11111111000000001111111100000000 | 16711935 |
| c10=10101010010101011010101001010101 | 2857740885 |
| c11=11001100001100111100110000110011 | 3425946675 |
| c12=10011001011001101001100101100110 | 1721329305 |
| c13=11110000000011111111000000001111 | 4027576335 |
| c14=10100101010110101010010101011010 | 1520786085 |
| c15=11000011001111001100001100111100 | 1019428035 |
| c16=10010110011010011001011001101001 | 2523502185 |
| c17=11111111111111110000000000000000 | 65535 |
| c18=10101010101010100101010101010101 | 2863289685 |
| c19=11001100110011000011001100110011 | 3435934515 |
| c20=10011001100110010110011001100110 | 1718000085 |
| c21=11110000111100000000111100001111 | 4042264335 |
| c22=10100101101001010101101001011010 | 1515890085 |
| c23=11000011110000110011110000111100 | 1010615235 |
| c24=10010110100101100110100101101001 | 2526439785 |
| c25=11111111000000000000000011111111 | 4278190335 |
| c26=10101010010101010101010110101010 | 1437248085 |
| c27=11001100001100110011001111001100 | 869059635 |
| c28=10011001011001100110011010011001 | 2573624985 |
| c29=11110000000011110000111111110000 | 267448335 |
| c30=10100101010110100101101010100101 | 2774162085 |
| c31=11000011001111000011110011000011 | 3275539260 |
| c32=10010110011010010110100110010110 | 1771476585 |

The block diagram of the PNWCG for (32) bits is shown in the figure (8), for c1,c2 and the block diagram of the PNWCG for (32) bits for c2 in the Quartus II 9.I design environment is shown in the figure (9).

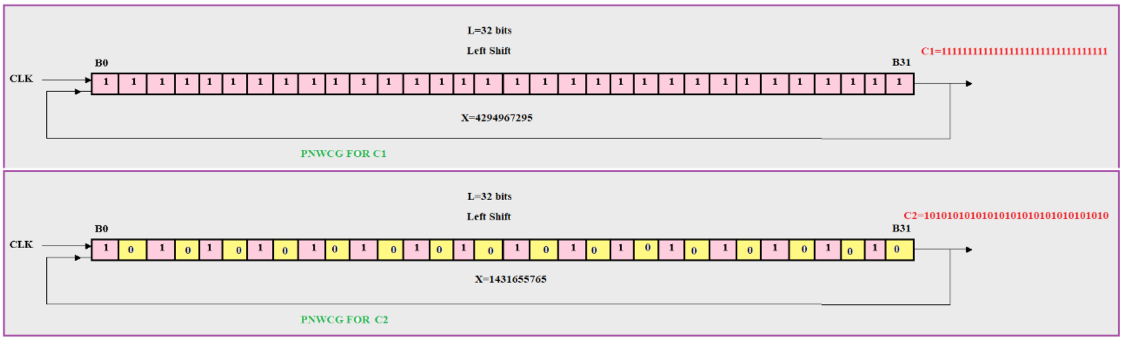

Figure (8): block diagram of the PNWCG for (32) bits for c1 and c2

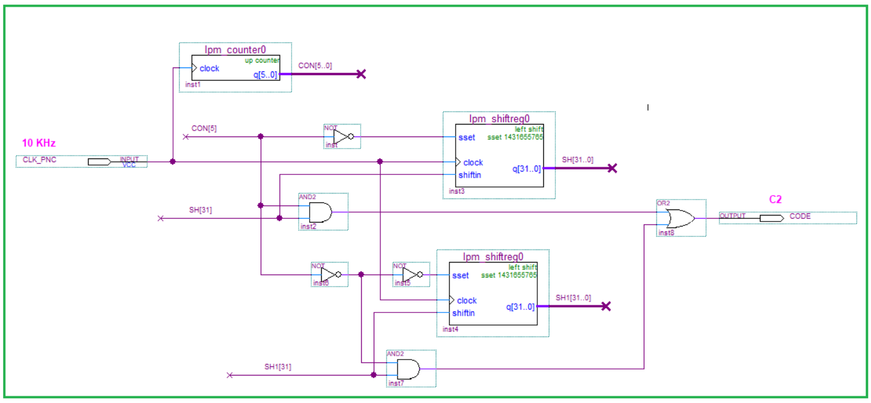

Figure (9): block diagram of the PNWCG for (32) bits for c2 in the Quartus II 9.I design environment.

The results of the PNWCG designed for (32) bits Walsh codes using an FPGA chip placed on the DE-1 Education and Development board are shown in the figure (10)

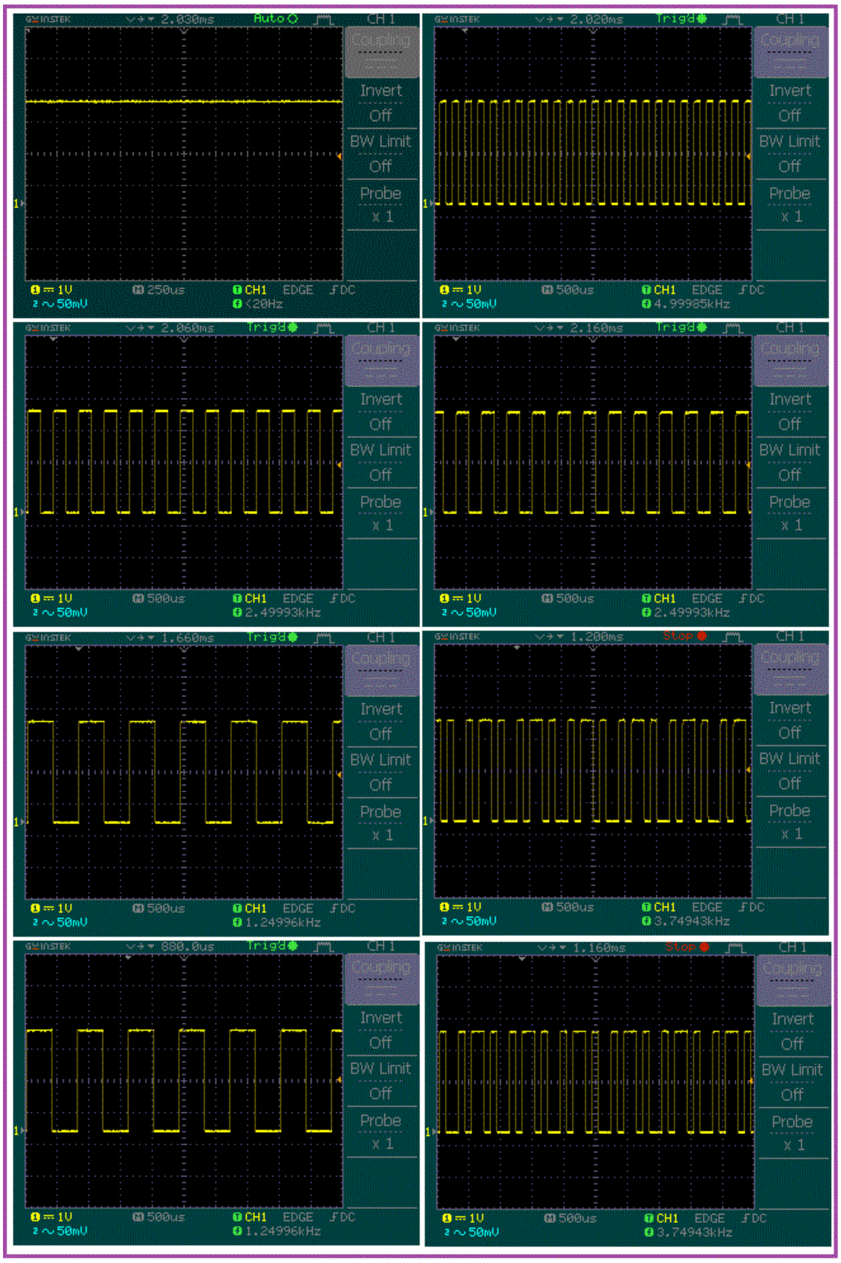

Figure (10):Walsh codes system DSSS for (32) bits: c1,c2,c3,c4,c5,c6,c7,c8 in the time domain

## VIII. BLOCK DIAGRAM OF THE CLOCK AND DATA GERATOR

To generate clock pulses with a frequency of (10 KHz) for all components PNWCG , 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]:

$$F_{CLK} = \frac{L.F_{SYS}}{2^n} \Longrightarrow L = \frac{2^{\mathrm{n}} \text{ x } F_{CLK}}{F_{SYS}} \qquad (7)$$

Where : (L) frequency code of DDFS , (n=32) bits number of phase accumulator DDFS ,($F_{sys}$ =50MHz) the reference frequency from DE-1, ($F_{clk}$ =10KHz) the clock frequency to be generated by DDFS.

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

To generate the codes walsh with lengths of (8) , (16) , and (32-bits) , a (10 KHz) clock pulse was used. the functional diagram of the clock pulse generator in Quartus II 9.I design environment shown in the figure (11).

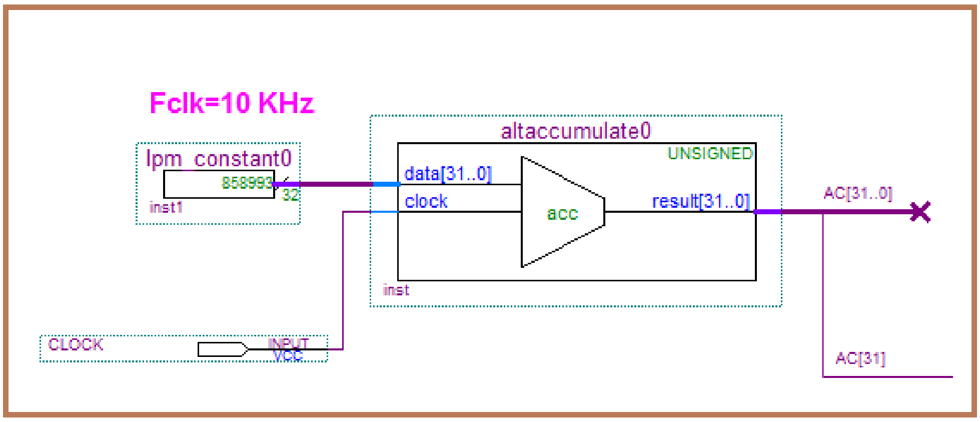

Figure (11): functional diagram of the clock pulse generator in the Quartus II 9.I design environment.

The practical designed results of the clock pulse generator using an FPGA chip located on the DE-1 board and using the Quartus 9.I design software environment are shown in figure (12).

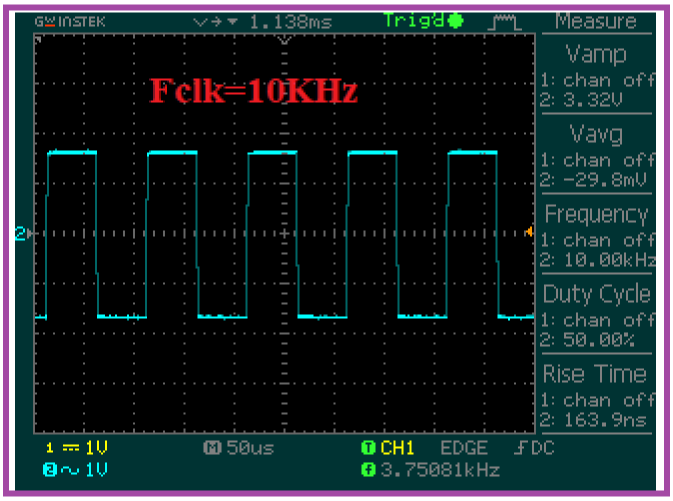

Figure (12):clock pulse with frequensy 10 KHz

## IX. CONCLUSION AND RESULTS

In this study, a Walsh code generator with lengths of 8, 16, and 32 bits was designed and implemented using shift registers and digital circuits programmed on an FPGA platform with the DE-1 development board. The clock pulse generator was designed based on Direct Digital Frequency Synthesis (DDFS) operating at a frequency of 10 kHz, providing high flexibility in frequency generation for encryption and digital communication applications**.**

**Main results:**
Accuracy and stability: Testing confirmed that the design generates Walsh codes of various lengths with high accuracy and frequency stability, ensuring reliable performance in communication and encryption applications.

Flexibility: The system can easily switch code lengths, enhancing customization and adaptability using only FPGA programming resources.

Time efficiency: Simulation and hardware implementation results showed that the generation time and synchronization checks are suitable for practical applications with low power consumption compared to traditional systems.

Scalability and development ease: Thanks to the FPGA's open digital architecture, new features and improvements can be integrated without full redesign, such as error correction or additional code sets.

**Conclusions:**
The combination of shift registers with FPGA digital circuits provides an effective and reliable solution for generating Walsh codes with configurable lengths, meeting high performance encryption system requirements.

The DDFS-based clock pulse generator ensures frequency flexibility with stable and precise timing required for communication systems.

The implemented system is suitable for use in encryption, communication, and digital signal processing, where fast and accurate customized waveforms are essential.

## 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.
[4]. Volnei A. Pedroni, "Circuit Design With VHDL", MIT Press Cambridge, Massa- chusetts London, England (2004) 364.
[5]. Theodore K.F, Haddamard Matrices: Construction Using Number Theory and Algebra ,2019.
[6]. M. Ali, "Implementation of a Software Defined Spread Spectrum modem", 2012.
[7]. Afaq Ahmad, Sayyid Samir Al-Busaidi and Mufeed Juma Al-Musharafi. On Properties of PN Sequences Generated by LFSR – a Generalized Study and Simulation Modeling. Indian Journal of Science and Technology-2013.
[8]. GOLDBERG B. 1999- "Digital Frequency Synthesis Demystified", LLH Technology Publishing, united states, 334.

## 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.