Design and implementation in VHDL code of the discrete cosine transform over finite fields for symmetric convolution operation

J M Vilardy\textsuperscript{1}, L Barba\textsuperscript{1}, and C O Torres\textsuperscript{1}

\textsuperscript{1}Grupo de Óptica e Informática, Universidad Popular del Cesar, Valledupar, Colombia

E-mail: vilardy.juan@unicesar.edu.co

Abstract. The discrete cosine transform is a very important mathematical tool for image processing operations, such as filtering, recognition, segmentation, compression, etc. We present the design and implementation of the description and synthesis in VHDL code of the discrete cosine transform over the finite field with an order given by the Mersenne prime number 127, in order to perform the symmetric convolution operation. The development of the VHDL code for the discrete cosine transform and the symmetric convolution operations are performed using the HDL Coder in Simulink. The discrete cosine transform over finite field is defined using a new trigonometric transform proposed recently. The main aim for the development of the discrete cosine transform and the symmetric convolution hardware architectures over finite field is to integrate these architectures in applications of filtering, encryption and recognition of images.

1. Introduction

The discrete cosine transform (DCT) allows to implement several applications applied to images, for instances: filtering, encryption, compression and recognition [1].

The quantization of very large or too small numbers produce overflow or underflow problems, respectively. The finite fields are used with the purpose of avoiding these previous problems. The finite fields can perform multiplication of large integers, coding and decoding of digital data, detection and correction of errors in communication systems, estimation of movement, speech processing, filtering, compression and encryption of signals and images, among other applications [2–7]. Most of these applications that use finite fields are implemented by using FPGAs, due to the following unique characteristics of the FPGAs: high processing speed, digital signal processing in parallel, large amount of digital resources for the implementation of arithmetic operations and these FPGAs can be reprogrammed to change their functionality [8].

We present the design and implementation in VHDL code of the DCT over the finite field with an order given by the Mersenne prime number 127 (which is represented by $\mathbb{Z}_{127}$), using the HDL Coder of Simulink. We implement the hardware architectures in VHDL codes for the sum and multiplication operations over $\mathbb{Z}_{127}$. Later, we use the hardware architecture in VHDL code of the DCT in order to compute the symmetric convolution operation over $\mathbb{Z}_{127}$. Finally, we expect to use the proposed hardware architectures in this work for future development in applications of filtering, encryption and recognition of images.
2. Complex numbers and the discrete cosine transform (DCT) over finite fields

The complex numbers over finite fields are known as the Gaussian integers (GIs) [2, 9]. The set of GIs over the finite field $\mathbb{Z}_p = \{0, 1, 2, \ldots, p-1\}$ (where $p$ is a prime number) is given by $GI(p) = \{a + jb, \text{ where } a, b \in \mathbb{Z}_p\}$. The set $GI(p)$ will be a finite field when the prime number $p$ is given by $p \equiv 3 \pmod{4}$. The number of elements or the order of $GI(p)$ is $p^2$. The Mersenne prime numbers ($M_r = 2^r - 1$, where $r$ is a prime number) satisfy the congruence $p \equiv 3 \pmod{4}$, for example: $p = M_2 = 3, p = M_3 = 7, p = M_5 = 31, p = M_7 = 127$ and $p = M_{13} = 8191$.

The set $GI(p)$ with elements of unitary modulus is given by the elements $\gamma = a + jb \in GI(p)$ that satisfy: $a^2 + b^2 \equiv 1 \pmod{p}$. The finite field $GI(p)$ and its elements with unitary modulus are used for defining the DCT over finite fields [9].

Let $\gamma$ be an element with unitary modulus of $GI(p)$ and multiplicative order of $2N$, then the forward DCT over the finite field $GI(p)$ of the sequence or vector $x = (x_i)$, where $x_i \in \mathbb{Z}_p$ and $i = 0, 1, 2, \ldots, N-1$, is the sequence or vector $X = (X_k)$, where $X_k \in \mathbb{Z}_p$ and $k = 0, 1, 2, \ldots, N-1$, given by the Equation (1).

$$X_k = DCT\{x_i\} = \sum_{i=0}^{N-1} \sqrt{\frac{2}{N}} \beta_k x_i \cos_\gamma \left( k \left( i + \frac{1}{2} \right) \right), \quad \cos_\gamma(x) = \frac{\gamma^x + \gamma^{-x}}{2}, \quad (1)$$

where $x = k(i + \frac{1}{2})$ and $\beta_k$ is equal to $\sqrt{2^{-i}} \pmod{p}$ when $k = 0$ and 1 for $k \neq 0$. The inverse DCT over the finite field $GI(p)$ of the vector $X_k$ is defined in Equation (2).

$$x_i = DCT^{-1}\{X_k\} = \sum_{k=0}^{N-1} \sqrt{\frac{2}{N}} \beta_i X_k \cos_\gamma \left( i \left( k + \frac{1}{2} \right) \right), \quad (2)$$

where $\beta_i$ is equal to $\sqrt{2^{-i}} \pmod{p}$ when $i = 0$ and 1 for $i \neq 0$. The forward and inverse DCT given by the two previous equations are unitary. If the sequences or vectors have a size of $1 \times N$, the forward and inverse DCT over the finite field $GI(p)$ can be computed using the multiplications of arrays specified in Equation (3).

$$X_k = x_i \ast FFDCT_{ik}, \quad x_i = X_k \ast FFDCT_{ik}^{-1}, \quad (3)$$

where the arrays $FFDCT_{ik}$ and $FFDCT_{ik}^{-1}$ of size $N \times N$, are represented in Equation (4).

$$FFDCT_{ik} = \sqrt{\frac{2}{N}} \beta_k \cos_\gamma \left( k \left( i + \frac{1}{2} \right) \right), \quad FFDCT_{ik}^{-1} = \sqrt{\frac{2}{N}} \beta_i \cos_\gamma \left( i \left( k + \frac{1}{2} \right) \right), \quad (4)$$

3. Design and implementation in VHDL code of the symmetric convolution operation using the DCT over finite fields

The design and implementation in VHDL code of the DCT over the finite field $Z_{127}$ and the unitary modulus element $\gamma = 106 + j103 \in GI(127)$ with a multiplicative order of 16, is implemented using the HDL Coder in Simulink [10]. All the signals are encoded with digital words in fixed point representation, using seven bits for the integer part, zero bit for the fraction part and no sign. The DCT over $Z_{127}$ uses the sum and multiplication modulo 127, for instance $c \equiv (a + b) \pmod{127}$ and $d \equiv (a \times b) \pmod{127}$, where $a, b, c$ and $d$ are positive integer numbers between 0 and 126. The description and synthesis of the hardware architecture for the DCT over $Z_{127}$ is implemented using several digital electronics components, such as: single-port Read
Only RAMs, adders, registers, multiplexers and finite state machines (FSMs). The subsystem of Simulink for computing the DCT over $\mathbb{Z}_{127}$ of a sequence with 8 samples, is called FFDCT_8.

The discrete convolution of $x_i$ and $h_i$ with $M$ and $N$ samples, is defined in Equation (5).

$$g_i = \sum_{k=0}^{N-1} x_k h_{i-k},$$ \hspace{1cm} (5)

where the sequence $g_i$ has $M + N - 1$ samples. The sum and multiplication operations are computed over the finite field $\mathbb{Z}_{127}$. When $x_i = (2 \ 1 \ 3 \ 4 \ 2 \ 1)$ with $M = 6$ samples and $h_i = (1 \ 2 \ 1)$ with $N = 3$ samples, the result of the command convolution (conv) of Matlab over $\mathbb{Z}_{127}$ applied to $x_i$ and $h_i$ is $g_i = (2 \ 5 \ 7 \ 11 \ 13 \ 9 \ 4 \ 1)$ with $M + N - 1 = 8$ samples.

The discrete convolution of Equation (5) can be computed using the symmetric convolution proposed in [11]. The DCT over the finite field $\mathbb{Z}_{127}$ can be used in order to compute the discrete convolution [9, 12]. Other way to compute the Equation (5) is presented in Equation (6).

$$g_i = DCT^{-1}\{DCT\{\hat{x}_i\} \times DCT\{\hat{h}_i\}\},$$ \hspace{1cm} (6)

where the forward and inverse DCTs over $\mathbb{Z}_{127}$ has 8 samples, $\hat{x}_i$ and $\hat{h}_i$ are extended sequences of $x_i$ and $h_i$, respectively [11]. The system of Simulink for computing the discrete convolution over $\mathbb{Z}_{127}$ of two sequences with 8 samples each one, is called convolution_z127_8 and it is presented in Figure 1. The block FFDCT_8 of Figure 1 can compute the forward and inverse DCT over $\mathbb{Z}_{127}$ for a sequence of 8 samples. In figure 1, the block multiplication_z127 is used to perform the multiplications of Equation (6) and the block control_convolution_z127_8 controls the flow of the input and output digital data when the discrete convolution is computed.

The VHDL codes for the DCT and the convolution over $\mathbb{Z}_{127}$ for the two sequences with 8 samples each one, are generated using the HDL Coder of Simulink applied to the system convolution_z127_8 of Figure 1. The sequence $\hat{x}_i$ is the input port input_data of Figure 1, the DCT of the sequence $\hat{h}_i$ is previously stored in the block control_convolution_z127_8 (typically, this sequence is known as the kernel or filter) and it is used to compute the multiplications of Equation (6). The output port output_data corresponds to the sequence $g_i$ with 8 samples.

**Figure 1.** Complete architecture for the system of Simulink convolution_z127_8.
The VHDL codes were generated for the board FPGA Atlys. The Simulink allows the co-simulation with the hardware of the board FPGA Atlys using the ethernet port [10]. The result of the co-simulation with the hardware of the board FPGA Atlys for the input data \( \hat{x}_i = (0 \ 2 \ 1 \ 3 \ 4 \ 2 \ 1 \ 0) \), the output data \( g_i = (2 \ 5 \ 7 \ 11 \ 13 \ 9 \ 4 \ 1) \) and the sequence \( \hat{h}_i = (1 \ 2 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0) \) are presented in Figure 2 and Figure 3.

**Figure 2.** Input data \( x_i \) (input data from Figure 1) for the co-simulation with the hardware of the board FPGA Atlys using the subsystem of Simulink convolution \( z_{127} \).

**Figure 3.** Output data \( g_i \) (output data from Figure 1) for the co-simulation with the hardware of the board FPGA Atlys using the subsystem of Simulink convolution \( z_{127} \). The convolution operation is between two sequences over \( Z_{127} \).

4. Conclusions
We have presented the design and implementation in VHDL code of the DCT and the discrete convolution of two sequences with 8 samples each one over \( Z_{127} \), using the HDL Coder of Simulink. The computing time of the developed hardware architecture for the discrete convolution of two sequences was 3.7 \( \mu s \), which is very fast. The finite fields allowed suitable hardware architectures over FPGA boards, because the mathematical structure of the finite fields is very related to the behavior or operation of the digital electronics components. Finally, all the developed hardwares architectures in this work can be integrated as basic operations in applications of filtering, encryption and recognition of images.

Acknowledgments
This research has been funded by the Universidad Popular del Cesar.
References

[1] K Ramamohan Rao and P Yip 2014 *Discrete cosine transform: Algorithms, advantages, applications* (San Diego: Academic Press, Inc.)

[2] Schroeder M 2009 *Number theory in science and communication: with applications in cryptography, physics, digital information, computing, and self-similarity* (Berlin: Springer)

[3] Bhattacharya M, Creutzburg R and Astola J 2004 Some historical notes on number theoretic transform *Proc. Int. TICS Workshop on Spectral Methods and Multirate Signal Processing, SMMS 2004* (Vienna) ed Astola J, Egiazarian K, Saramki T pp 1–8

[4] Kim K, Lee H and Kim S 2018 *Proc. of the 7th International Conference on Emerging Databases (Busan)* (Singapore: Springer) p 30

[5] Vilardy J, Jimenez C and Perez R 2017 *J. Phys.: Conf. Ser.* 850 012012 1–7

[6] Perez R, Vilardy J and Jimenez C 2017 *J. Phys.: Conf. Ser.* 792 012046 1–7

[7] Perez R, Vilardy J and Torres C 2017 *J. Phys.: Conf. Ser.* 792 012047 1–5

[8] Deschamps J P, Bioul G J and Sutter G D 2006 *Synthesis of arithmetic circuits: FPGA, ASIC and embedded systems* (Weinheim: Wiley)

[9] Lima J B 2008 *Trigonometria sobre corpos finitos: novas definições e cenários de aplicação* (Recife: Universidade Federal de Pernambuco)

[10] Giacometto F, Vilardy J, Torres C and Mattos L 2011 *J. Phys.: Conf. Ser.* 274 012042 1–13

[11] Martucci S A 1994 *IEEE Transactions on Signal Processing* 42 1038–1051

[12] Lima J B 2015 *Electron. Lett.* 51 1570–1572