Design and implementation in VHDL code of the two-dimensional fast Fourier transform for frequency filtering, convolution and correlation operations

Juan M Vilardy¹, F. Giacometto, C. O. Torres and L. Mattos
Laboratorio de Óptica e Informática, Universidad Popular del Cesar, Sede balneario Hurtado, Valledupar, Cesar – Colombia
E-mail: vilardy.juan@unicesar.edu.co

Abstract. The two-dimensional Fast Fourier Transform (FFT 2D) is an essential tool in the two-dimensional discrete signals analysis and processing, which allows developing a large number of applications. This article shows the description and synthesis in VHDL code of the FFT 2D with fixed point binary representation using the programming tool Simulink HDL Coder of Matlab; showing a quick and easy way to handle overflow, underflow and the creation registers, adders and multipliers of complex data in VHDL and as well as the generation of test bench for verification of the codes generated in the ModelSim tool. The main objective of development of the hardware architecture of the FFT 2D focuses on the subsequent completion of the following operations applied to images: frequency filtering, convolution and correlation. The description and synthesis of the hardware architecture uses the XC3S1200E family Spartan 3E FPGA from Xilinx Manufacturer.

1. Introduction
The two-dimensional Discrete Fourier Transform (DFT 2D, [1]) allows to develop a large number of applications applied to images, among which are: filtering (using convolution in the frequency domain), image compression and encryption, and object or people recognition (with the correlation operation in the domain frequency), etc. The possibility of implementation of the above applications are due to the existence of a fast algorithm for calculating the DFT 2D, this algorithm is known as the Fast Fourier Transform, (FFT, [2]). The first FFT algorithms were published by Cooley and Tukey in 1965 [3-4].

One of the main features of the applications listed in the preceding paragraph that using the FFT for its operation, is the portability of the hardware and the execution speed and accuracy of calculating the FFT, these features are present in the development of the hardware architecture of this research paper. To meet the following requirements: computational speed and reduced area required for the Digital Signal Processing, (DSP) referred to above, there is a powerful hardware tool known as FPGA, which are arrays of reconfigurable logic gates by effect field. Among the numerous advantages provided by the use of FPGAs, three stand out in particular: the low cost of prototyping, conducting operations in parallel and the short production time [5].

¹ To whom any correspondence should be addressed.
This research article uses the hardware description language VHDL to develop and implement the FFT 2D; the VHDL files are generated directly from Simulink programming environment, using Embedded Matlab and HDL Simulink coder [6], these tools allow easy use of: complex signals, overflow, underflow and generation of test benches, among other facilities. All operations in these programming environments have Simulink fixed-point number representation [7].

The hardware architecture of the DFT 2D proposed in this paper is an initial work shows the extent of hardware architectures implemented in reference 8. The results report on the project of the code generation for the FFT 2D with fixed point numerical representation is organized as follows: section 2 presents the DFT 2D and FFT. In section 3 we have the hardware implementation of the FFT 2D and its respective simulation in ModelSim tool. The analysis of the results obtained with the hardware implementation of the FFT 2D are describe in section 4. Finally, the conclusions are presented in section 5.

2. DFT 2D and FFT

The DFT 2D is the sampling of the two-dimensional continuous Fourier transform, this is defined as:

$$X(u, v) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y) e^{-2\pi i xu} dx e^{-2\pi i yv} dy$$

(1)

The DFT 2D make the sampling from $X(u, v)$ into a samples or points set of size NxM, as follow:

$$X(k, s) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} x(n, m) W_{N}^{ns} W_{M}^{mk}$$

(2)

$$W_{N} = e^{-2\pi i / N}$$

(3)

And the inverse of $X(k, s)$, is:

$$x(n, m) = \frac{1}{MN} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} X(k, s) W_{N}^{-ns} W_{M}^{-mk}$$

(4)

From the definitions (1) and (2) shows that the Fourier transform in two dimensions can be divided into two Fourier transform 1D (one-dimensional); for the case in discrete, the DFT 2D can be calculated using the FFT first on rows and then for this result is applied to the FFT on columns or vice versa, initially applies the FFT on columns and then applies the FFT on the rows. With the previous result, calculate the DFT 2D is based on the FFT.

Because $x(n)$ may be either real or complex, evaluating $X(k)$ requires on the order of $N$ complex multiplications and $N$ complex additions for each value of $k$. Therefore, because there are $N$ values of $X(k)$, computing an $N$-point DFT 1D requires $N^2$ complex multiplications and additions [2]. The basic strategy that is used in the FFT algorithm is one of “divide and conquer,” which involves decomposing an $N$-point DFT 1D into successively smaller DFTs 1D. To see how this works, suppose that the length of $x(n)$ is even (i.e., $N$ is divisible by 2). If $x(n)$ is decimated into two sequences of length $N/2$, computing the $N/2$-point DFT 1D of each of these sequences requires approximately $(N/2)^2$ complex multiplications and the same number of additions. Thus, the two DFTs require $2(N/2)^2=N^2/2$ complex multiplies and adds. Therefore, if it is possible to find the $N$-point DFT 1D of $x(n)$ from these two $N/2$-point DFT 1D in fewer than $N^2/2$ operations, a savings has been realized. These algorithms that use the division of the sequence $x(n)$ in time is known as algorithm of the FFT with decimation in time, there are algorithms of the FFT with decimation in frequency [2]. In this paper we use the algorithm of decimation in time. The most popular algorithms for the calculation of the FFT using radix-2 or radix-4 with decimation in time or frequency [2].

2.1. Radix-2 decimation in time FFT

When the transform of $N$ points is decimated into two transforms with $N/2$ points y these latter are decimated into transforms with $N/4$ points y and so on, the decimation may be continued until there are only two-point DFTs 1D, the resulting algorithm for the calculation of the FFT is called radix-2.
Computing an N-point DFT using a radix-2 decimation in time FFT is much more efficient than calculating the DFT directly. For example, if N=2\(^v\), there are v=log\(_2\)N stages of computation. Because each stage requires N/2 complex multiplies by the twiddle factors \(W_N^k\) and N complex additions, there are a total of (N/2)*log\(_2\)N complex multiplications and N*log\(_2\)N complex additions.

3. Hardware architecture

Figure 1 is a block diagram of the hardware architecture for the FFT 2D with the programmed embedded Matlab functions, this architecture calculate the FFT 2D for a square matrix of N size (N = 64). All embedded Matlab blocks were programmed in a way behavioral and concurrent using fixed point numerical representation, now we describe the functionality of the blocks of figure 1: FFT\(_{64}\) and control\_FFT\(_{2D}\).

Figure 1. FFT 2D hardware architecture Implementation in Simulink for VHDL code generation with Simulink HDL Coder.

In the previous section we show that the FFT algorithm is the basis for calculating the DFT 2D, in Figure 1 the block responsible for calculating the FFT is FFT\(_{64}\), this block calculates the 64-point complex FFT (if the input to transform is real, the imaginary part is taken as zero and form a complex signal) using radix-2 decimation in time [8]. The FFT\(_{64}\) block works in streaming mode (receives data continuously) and process when the start signal of the same block is in high or true logic, after a delay (D = 188 clock cycles for N = 64 points [8]) FFT\(_{64}\) block gives the results of the DFT 1D of N points and dvalid signal is set at true logic; all block components FFT\(_{64}\) process the image pixels in parallel. The image to transform has a NxN pixels or points size, and it is process as a column vector of N\(^2\) points, therefore the FFT\(_{64}\) block performs 2N DFT 1D, since the FFT is first applied to the columns and then this last result is applied to the FFT to the rows.

To control the calculation of the 2N FFT that needs the DFT 2D and the stored intermediate results is used the control\_FFT\(_{2D}\) block, this block generates the following signals: enabling, address reading and writing to external RAM where the intermediate results of the FFT calculated are saved (dout output port from FFT\(_{64}\) block is directly connected to the data port of external RAM), enabling for the FFT\(_{64}\) block (start of the FFT calculation of N points) and the signal that
controls *Switch* for selecting the source of points and apply the FFT (the source of the points can be the input image or intermediate results stored in the external RAM), so the control block concurrently manages several control processes.

The number of cycles it takes the architecture of figure 1 to calculate a DFT 2D of an image or NxN square matrix is defined by:

$$N_c = 2(N^2 + D) - (N + 1)$$

Where D is the delay of the block *FFT_64* to calculate an FFT that depends on N [8], if N=64 we have D=188 and NC=8503 clock cycles. The test images to the hardware architecture of the FFT 2D can be found in Figure 2 and Figure 3 shows the magnitudes and phases obtained from: the FFT 2D hardware architecture describe in this article and Simulink block that computes the FFT 2D, respectively.

![Figure 2. The test images.](image)

![Figure 3. Magnitudes and phases of the FFT 2D obtained from the hardware implementation of this article and the FFT 2D Simulink block.](image)

To verify the accuracy of the DFT 2D with fixed point, the result is compared with the result that Simulink block computes for the FFT 2D using the Mean Square Error (MSE), the MSE can be defined by the energy difference between the images to check, so (N = 64):
The result of MSE for the magnitude and phase of the FFT 2D are: $8.7138 \times 10^{-5}$ and $7.1689 \times 10^{-4}$, respectively, demonstrating that the algorithm coded in fixed point has a good accuracy.

To the FFT 2D hardware architecture of Figure 1 after being programmed, its applies the tools available in the Simulink HDL coder, which are: compatibility checker code written in Simulink with respect to behavioral VHDL implementations available to the encoder, generation VHDL files from the codes programmed in Simulink and ultimately the generation of files test benches that allow the simulation of VHDL codes generated in the ModelSim simulation tool (these files have the same VHDL file name followed by the identifier generated _tb). The Simulink HDL coder automatically separates the real part of the imaginary part from complex numbers when generating VHDL files into two different signals and with respect to operations which involve such complex numbers, these operations are re-written with the two signals generated from each complex number.

To verify operation of the hardware architecture (both control components such as the calculation of the FFT) and possible speed requirements for a particular application that uses the FFT 2D, this is simulated with the ModelSim tool, loading in the latter tool the test benches generated by Simulink HDL coder, the results can be viewed in Figure 4.

![Figure 4](image_url)

**Figure 4.** Simulation of the FFT 2D hardware architecture with fixed point using ModelSim.

### 4. Results

The implementation of the hardware architecture of the FFT 2D in Figure 1 was successfully synthesized in XC3S1200E FPGA family Spartan 3E from Xilinx, with the programming tool Xilinx ISE Web Pack 12.1, and a working frequency of 50 MHz.

The XC3S1200E FPGA family Spartan 3E from Xilinx is in the Nexys2 development board that is distributed by the manufacturer Digilent, this development board has an external RAM: 128Mbit Micron M45W8MW16 Cellular RAM pseudo-static (DRAM device organized as an array of 8 MB for 16-bit lines), a 50 MHz oscillator, port: VGA, serial, PS/2, expansion, etc. [9].

Figure 5 shows the resource use of the FPGA for the implementation of the hardware architecture of the FFT 2D from Figure 1. Figure 5 shows that there are not over-mapping for any of the resources used in the FPGA, allowing the implementation of the hardware architecture of the FFT 2D in the selected FPGA device. Memory blocks RAMs dual (used 4 of the 28 available) and dedicated multipliers (MULT18X18SIOs, used 20 of the 28 available) are used exclusively for the FFT_64 block. The slices used for programmable hardware architecture were 76% of the slices available in the FPGA.

Equation (6) from section 3 determines the number of clock cycles (Nc) it takes for the hardware architecture described in this paper to calculate the FFT 2D of an NxN square image, for N=64, we have $D=188$ and $Nc=8503$ clock cycles, with a working frequency of 50 MHz, we have a clock cycle time of 20 ns and therefore the computing time of a FFT 2D for a NxN square image with $N = 64$ is
0.17006 ms, so in this way if you use the hardware architecture of the FFT 2D described in this paper to perform convolution or correlation in the frequency domain from two NxN square images, we have the triple computation time with respect to time for calculating the FFT 2D and it will increase to twice the size of the external RAM to store intermediate results of the convolution or correlation.

| Two_FFT_test_one Project Status (06/11/2010 - 19:00:43) |
|-----------------------------------------------------------|
| **Project File**: ise_two_dimensional_fft_mod.vhd          |
| **Parser Errors**: No Errors                             |
| **Module Name**: Two_FFT_test_one                         |
| **Implementation State**: Programming File Generated      |
| **Target Device**: xc3s1200e-4fg2120                       |
| **Product Version**: ISE 12.1                              |
| **Warnings**:                                              |
| **Design Goal**: Balanced                                 |
| **Routing Results**: All Signals Completely Routed        |
| **Design Strategy**: Xilinx Default (Unlocked)             |
| **Timing Constraints**: All Constraints Met                |
| **Environment**: System Settings                          |
| **Final Timing Score**: 0 (Timing Report)                 |

**Device Utilization Summary**

| Logic Utilization                  | Used       | Available | Utilization | Note(s) |
|------------------------------------|------------|-----------|-------------|---------|
| Number of Slice Flip Flops         | 4,570      | 17,344    | 26%         |         |
| Number of 4 input LUTs             | 10,107     | 17,344    | 58%         |         |
| Number of occupied Slices          | 6,658      | 8,672     | 76%         |         |
| Number of Slices containing only related logic | 6,658 | 6,658 | 100% | |
| Number of Slices containing unrelated logic | 0 | 6,658 | 0% |         |
| Total Number of 4 input LUTs       | 10,106     | 17,344    | 62%         |         |
| Number used as logic               | 9,832      |           |             |         |
| Number used as a route-thru        | 769        |           |             |         |
| Number used as Shift registers     | 275        |           |             |         |
| Number of bundled IOs              | 162        | 250       | 64%         |         |
| Number of RAM16s                   | 4          | 28        | 14%         |         |
| Number of BUF/GMUXs                | 1          | 24        | 4%          |         |
| Number of MULT18K18SIOs            | 20         | 28        | 71%         |         |
| Average Fanout of Non-Connect Nets | 2.43       |           |             |         |

**Figure 5.** XC3S1200E FPGA resources used by the implementation of the hardware architecture of the FFT 2D from Figure 1.

5. Conclusions

A hardware architecture for the calculation of the FFT 2D with fixed point numerical representation was developed with VHDL code generation using Simulink HDL Coder and embedded Matlab. The use of resources for the selected FPGA in this research article was high (76% of the slices available), so the selected FPGA is not very suitable for DSP operations that involve a lot of mathematical calculations (as is the FFT 2D) and therefore we recommend a FPGA with higher capacity resources and best dedicated components, an alternative may be the FPGA family Xilinx’s Virtex-5. The implementation shown in the paper have a good accuracy around the order of $10^{-5}$ for the magnitude and phase of the FFT 2D compared with the result of the Simulink block that computes the FFT 2D, and also we simulate the hardware architecture of the FFT 2D with the ModelSim tool, with the simulation is verified the correct functioning of the hardware architecture proposed in this research article in both: the numerical results and the timing aspects. Finally, there is an easily programming of: complex signals, registers, RAMs dual memories, finite state machines and handling of overflow and underflow, with the Fixed-Point Toolbox of Matlab, Embedded Matlab Function and Simulink HDL Coder, achieving generate the hardware architecture described in this paper in a time less than three weeks, according to experiences of the authors of this research article. The hardware architecture of the FFT 2D synthesized and implemented in this paper can be used in DSP applications such as those mentioned in the introduction to this article.
6. References

[1] Jahne B 2003 (Digital Image Processing) Ed. Springer-Verlag. (Germany)

[2] Hayes M 2001 (Digital Signal Processing) Ed. McGraw Hill, (United States of America), p. 262-286.

[3] Cooley J W and Tukey J W, 1965 An algorithm for machine calculation of complex Fourier series, Math. Comput., vol. 19, p. 297-301.

[4] Cooley J W, Tukey J W and Welch P D, 1969 The Finite Fourier Transform, IEEE Trans. on Audio and Electroacoustics, vol. AU-17, no. 2.

[5] Moraleda U. 2001 Diseño de Hardware Digital con VHDL, p. 23-87.

[6] Simulink® HDL Coder™ 1 User Guide, http://www.mathworks.com/access/helpdesk/help/pdf_doc/slddcode/slddcode_ug.pdf

[7] Fixed-Point Toolbox™ 2 User Guide, http://www.mathworks.com/access/helpdesk/help/pdf_doc/fixedpoint/FPTUG.pdf

[8] Yaruro Y and Vilardy J 2009 Descripción y Sintesis de Arquitecturas Hardware de La Transformada Rapida de Fourier en VHDL para Sistemas de Comunicaciones Basados en La Tecnica de Multiplexacion por Division de Frecuencias Ortogonales (OFDM Estandar IEEE 802.11a) School of electronic Engineering (Universidad Popular del Cesar Press)

[9] Digilent Nexys2 Board Reference Manual, http://www.digilentinc.com/Data/Products/NEXYS2/Nexys2_rm.pdf