Article

FPGA-Based Hardware Implementation of a Stable Inverse Source Problem Algorithm in a Non-homogeneous Circular Region

José Jacobo Oliveros Oliveros * , José Rubén Conde Sánchez * , Carlos Arturo Hernández Gracidas * , María Monserrat Morín Castillo , José Julio Conde Mones *

Posted Date: 9 October 2023

doi: 10.20944/preprints202310.0381.v1

Keywords: Field-Programmable Gate Arrays (FPGAs); unrolling architectures; inverse source problem; Ill-posed problems; Tikhonov regularization; elliptic boundary problems; operational equations; Hilbert spaces.

Article

FPGA-Based Hardware Implementation of a Stable Inverse Source Problem Algorithm in a Non-Homogeneous Circular Region

José Jacobo Oliveros Oliveros [1,*,†], **José Rubén Conde Sánchez** [1,*,†], **Carlos Arturo Hernández Gracidas** [2,*,†], **María Monserrat Morín Castillo** [3,†] and **José Julio Conde Mones** [1,*,†]

1 Physical-Mathematical Science Faculty, BUAP, Puebla, Puebla, México. Av. San Claudio y 18 sur, Col. San Manuel, Edificio FM1-101B, Ciudad Universitaria, Puebla, Puebla, México, C.P. 72570; oliveros@fcfm.buap.mx, rconde@fcfm.buap.mx, jconde@fcfm.buap.mx

2 Physical-Mathematical Science Faculty, CONAHCYT-BUAP, Puebla, Puebla, México. Av. San Claudio y 18 sur, Col. San Manuel, Edificio FM1-101B, Ciudad Universitaria, Puebla, Puebla, México, C.P. 72570; cahernandezgr@conahcyt.mx

3 Electronic Science Faculty, BUAP, Puebla, Puebla, México. Av. San Claudio y 18 sur, Col. San Manuel, Edificio FCE1, Ciudad Universitaria, Puebla, Puebla, México, C.P. 72570; maria.morin@correo.buap.mx

* Correspondence: oliveros@fcfm.buap.mx, rconde@fcfm.buap.mx, cahernandezgr@conahcyt.mx, jconde@fcfm.buap.mx; Tel.: +52 (222) 2295500 ext. 2178 (J.J.O.O.); +52 (222) 2295500 ext. 2141 (J.R.C.S.); +52 (222) 2295500 ext. 2171 (C.A.H.G.); +52 (222) 2295500 ext. 2171 (J.J.C.M.)

† These authors contributed equally to this work.

Abstract: This work implements a stable algorithm in Field-Programmable Gate Arrays (FPGAs) for which two architectures (unrolling the loop) were developed and analyzed. The algorithm recovers sources located at the boundary separating two homogeneous media that make up a two-dimensional non-homogeneous region from measurements on the boundary of such region. The problem of recovering these sources is an ill-posed inverse problem as small errors in the measurement can produce important changes in the source location. Inverse source problems have many applications in different areas, such as engineering and medicine, making the proposed implementation important. The first architecture (mode one) allows considering different operating speeds, which is an advantage depending on whether we work with fast or slow signals. The second one (mode two) reduces resource consumption by exploiting the characteristics of the source identification algorithm, which is an advantage for multichannel problems such as inverse electrocardiography or electroencephalography. The architectures were tested on four FPGAs of the 7 Series of Xilinx: Spartan-7 xc7s100fgga484, Virtex-7 xc7v585tffg1157, Kintex-7 xc7k70tfbg484, and Artix-7 xc7a35tcpg236. The two FPGA implementations of the algorithm were validated using synthetic examples implemented in MATLAB. The results presented here can be extended to concentric spheres and complex geometries.

Keywords: Field-Programmable Gate Arrays (FPGAs); unrolling architectures; inverse source problem; Ill-posed problems; Tikhonov regularization; elliptic boundary problems; operational equations; Hilbert spaces

1. Introduction

In this paper, we propose two architectures based on Field Programmable Gate Arrays (FPGAs) [1] for implementing an inverse source identification algorithm. The inverse source identification problem that concerns us consists of determining the source from measurements produced by it on the exterior boundary. The inverse source problem has many applications in practical problems such as inverse electroencephalography and inverse electrocardiography, in which the study is done by operational equations defined in Hilbert spaces [2,3]. These operational equations are ill-posed in the sense of

Hadamard [4]. Here, the operational equation, from which the algorithm was obtained, was solved using the Fourier series. More precisely, we solved the normal equations obtained using the Tikhonov regularization method. Some inverse source problems lead us to algebraic systems of linear equations, which are obtained from the operational equations by discretization [5]. The matrices in the systems of equations are ill-conditioned, which must be considered when we solve the system with error on the right side because systems are unstable, which is a consequence of the ill-posedness of the operational equations. At this point, it is important to consider that inadequate handling of precision might worsen results as the solution found for data with errors might be too far from the one for data without errors, besides increasing other costs in hardware resources and critical paths. In [6], the authors made a wake-up call, using 2×2 matrices to show how ill-conditioning and precision can affect system design (resources, cost, etc.), and they illustrated the effect generated in the calculation of the inverse of an ill-conditioned matrix when its elements were approximated by truncation.

We consider a circular, non-homogeneous medium and that the sources are located on the separation interface of two homogeneous media, which compound the non-homogeneous media. Sources and measurements are correlated by a boundary value problem, which allows making an operational statement from where the ill-posedness (in the sense of Hadamard) is analyzed, and the source identification algorithm is obtained. We consider the algorithm given in [7], in which the inverse source problem was developed for sources located on the interface of two homogeneous media. The authors developed the algorithm using the technique of Fourier series for a circular geometry and the finite element method for a complex geometry. Since we are considering a circular geometry, we implemented the algorithm in an FPGA using the Fourier series technique, i.e., we used the trigonometric base to express the solution to the inverse problem. Ill-posedness is related to numerical instability, which can produce significant changes in the solution due to small measurement changes. The Tikhonov regularization method is employed to handle this numerical instability, which depends on a parameter called the Tikhonov regularization parameter, properly chosen in terms of the measurement error [5]. We chose the Tikhonov regularization parameter numerically.

Associated with the trigonometric base, there is an arithmetic kernel, which allows hardware resources to be reused in a repetitive type (interactive loop) system, reducing the number of operations [8]. The proposed architectures, labeled mode one and mode two, were tested on four FPGAs of the 7 Series of Xilinx [1]: Spartan-7 xc7s100fgga484, Virtex-7 xc7v585tffg1157, Kintex-7 xc7k70tfbg484, and Artix-7 xc7a35tcpg236. We report the performance of architectures mode one and mode two in terms of power and resource consumption.

The two architectures can also be cloned (or replicated), leading to a multichannel structure. This offers the possibility of applying these architectures to problems whose nature is multichannel; examples of this are the problems of identifying bioelectrical sources from electroencephalographic or electrocardiographic signals. In the first case, voltage measurements are taken on the scalp through electrodes, following different arrangements (10-20 being the most used). Up to a thousand measurements per second are recorded on each electrode, which implies applying the source identification algorithm the same number of times. In the case of electroencephalography, whose maximum frequency is 120Hz, it could be feasible to use the two architectures. However, mode two architecture consumes fewer hardware resources, which could contribute to the development of portable electroencephalographs.

This work is presented in the following way. In Section 2, the mathematical model that relates the sources with measurements is given. Furthermore, the forward and inverse problems are solved. The solution to the inverse problem gives the stable source identification algorithm; Section 3 gives numerical examples associated with forward and inverse problems. These examples are developed using the MATLAB software to validate the source identification algorithm given in Section 2; in Section 4, the source identification algorithm is implemented in an FPGA using the two proposed architectures. In Section 5, the hardware implementations are validated using the same examples developed in Section 3. Finally, the conclusions are given in Section 6.

2. Basic elements

This section presents the foundational components of the research. It serves as the cornerstone for the rest of the paper and provides the reader with the necessary background information to understand the main results and conclusions.

2.1. FPGA

An FPGA is an integrated circuit designed to have its operation configured after manufacturing, assembly, and even deployment of a product it is part of. FPGAs contain programmable logic blocks (capable of performing a range of operations, from simple logic gates to complex functions) and memory blocks. These blocks can then be connected using a hierarchy of reconfigurable interconnects. The combination of these elements allows the FPGA to perform in many varied applications.

Some advantages of FPGAs are:

1. FPGAs are more flexible than Complex Programmable Logic Devices (CPLDs), generally having a greater number of both logic blocks and programmable interconnects.
2. FPGAs have a lower development cost than Application-Specific Integrated Circuits (ASICs). While an ASIC can perform the same operations as an FPGA and is specific to the application, it cannot be reprogrammed.
3. FPGAs have a faster time-to-market and lower non-recurring engineering (NRE) cost than ASICs.

Three of the main characteristics of FPGAs, which are related to the speed of the process, are:

1. Throughput: it refers to the amount of data that is processed per clock cycle (bits/second).
2. Latency: it refers to the time between data input and processed data output (clock cycles).
3. Timing: it refers to the logic delays between sequential elements (frequency).

2.2. Inverse and Ill-Posed Problems

Inverse problems consist of finding an unknown property of an object or medium from observations of its response to a test signal. In contrast to forward problems, where there is information about the causes of a process in a medium, and the solution leads to the discovery of the effect produced, inverse problems have partial information about the results or effects produced in the medium by some unknown causes. The solution to an inverse problem is found by analyzing the partial information about the results.

We can consider forward problems as those in which we have information about the causes that describe a process in a medium and the solution to the problem leads us to discover the effect produced by such causes, whereas, in inverse problems, there is partial information about the results or effects produced in the medium by some unknown causes which must be found from the analysis of the results (effects). Thus, forward problems are cause-effect problems, while inverse problems are effect-cause problems [4].

Source identification problems are widely investigated in many research fields, and they are modeled as boundary value problems where the analysis of both the associated forward problem and its corresponding inverse problem must be considered. The inverse problem involves determining the source that produced the measurement on the boundary of the region. It appears in applications such as inverse electroencephalography, inverse electrocardiography, and inverse geophysics, where problems are modeled using partial differential equations.

An operational equation of the form $Ax = y$, where $A : X \longrightarrow Y$ and X, Y are Banach or Hilbert spaces, is well-posed if it satisfies the following conditions [4]:

1. For each $y \in Y$ a solution to the problem exists.
2. For each $y \in Y$ a solution to the problem is unique.
3. The solution x to the problem continuously depends on the initial data y.

Problems that violate one or more of these conditions are referred to as ill-posed problems. Condition 3 is associated with numerical instability, i.e., small errors on the right side of the operational equation can result in significant changes in the location of the solution x. To address this numerical instability, regularization methods must be applied. In this work, we applied the Tikhonov regularization method, which involves choosing a regularization parameter in terms of the error [4,5].

2.3. Algorithm Implementation in FPGA

Some algorithms have been implemented in FPGAs [9–11].

However, the inverse source problem we study in this paper is ill-posed because it presents numerical instability, making it a challenging task to implement in hardware. To tackle this issue, we utilize the Tikhonov regularization method. In this work, we present the hardware implementation of two architectures for a source identification stable algorithm on an FPGA for the first time. Our approach shows that the algorithm can be parallelized, taking advantage of the unique capabilities of FPGAs. This implementation demonstrates the feasibility and efficiency of utilizing FPGAs for inverse source problems, which is an important and growing field [2,3,12–14]. Hence, the implementation developed in this work provides a new and promising solution for future research in this field.

3. Stable source identification algorithm

This section presents a mathematical model for a circular conductive region made up of two homogeneous media and defines the forward and inverse problems for the Superficial Boundary Value Problem (SBVP).

3.1. Mathematical model

In this work, a circular region Ω is considered as a conductive medium compound by two homogeneous media, as illustrated in Figure 1. Specifically, we consider that Ω consists of two concentric circles centered at the origin with radii R_1 and R_2. Ω_1 is the circle with radius R_1, and Ω_2 corresponds to the difference between the circle with radius R_2 and the circle with radius R_1, i.e., $\Omega = \Omega_1 \cup S_1 \cup \Omega_2$, where S_1 is the interface that separates regions Ω_1 and Ω_2. The conductivities of regions Ω_1 and Ω_2 are denoted by σ_1 and σ_2, respectively.

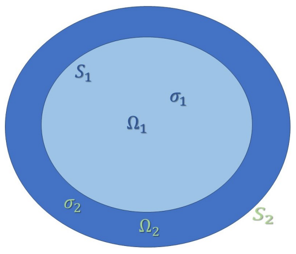

Figure 1. A circular conductive region composed of two homogeneous media. The source is located on the closed curve S_1, which separates the regions Ω_1 and Ω_2 with constant conductivities σ_1 and σ_2, respectively.

The boundary value problem to study the identification problem is given by [7]:

$$-\sigma_1 \Delta u_1 = 0, \quad \text{in} \quad \Omega_1, \tag{1}$$
$$-\sigma_2 \Delta u_2 = 0, \quad \text{in} \quad \Omega_2, \tag{2}$$
$$u_1 = u_2, \quad \text{on} \quad S_1, \tag{3}$$
$$\sigma_1 \frac{\partial u_1}{\partial n_1} = \sigma_2 \frac{\partial u_2}{\partial n_1} + g, \quad \text{on} \quad S_1, \tag{4}$$
$$\sigma_2 \frac{\partial u_2}{\partial n_2} = 0, \quad \text{on} \quad S_2, \tag{5}$$

where g is the source, $u_i = u|_{\Omega_i}$, $i = 1, 2$, and u represents the electric potential in Ω. The symbol Δ represents the Laplace operator, also denoted by ∇^2. Boundary condition (3) corresponds to the conductivity of the potential, and boundary condition (4) is associated with the jump in current flow due to the presence of the source. The conductivity of Ω^c is assumed to be zero, which leads to boundary condition (5). Using Green's formulas, we obtain the following compatibility condition:

$$\int_{S_1} g ds = 0. \tag{6}$$

Problem (1)-(5) is known as the Superficial Boundary Value Problem (SBVP) [7]. This problem has been used to study the inverse source problem in electroencephalography for cortical sources [2]. The following definitions are related to the SBVP.

Given g defined on S_1, the forward problem involves finding measurement $V = u|_{S_2}$, where u is the solution to the SBVP.

Given a function V defined on S_2, the inverse problem involves determining a source g defined on S_1, such that the solution u to the forward problem corresponding to g, satisfies that $u|_{S_2} = V$.

3.2. Forward problem

In this Section, the following functional spaces are considered:

$$L_2(S_i) = \left\{ h : S_1 \to \mathbb{R} : \langle h, h \rangle_{L_2(S_i)} = \int_{S_i} h^2(x) dx < \infty \right\}, i = 1, 2,$$

$$L_{2,\perp}(S_i) = \left\{ h \in L_2(S_i) : \langle h, 1 \rangle_{L_2(S_i)} = 0 \right\}, i = 1, 2,$$

$$L_2(\Omega) = \left\{ u : \Omega \to \mathbb{R} : \langle, uu \rangle_{L_2(\Omega)} = \int_{\Omega} h^2(x) dx < \infty \right\},$$

$$H^1(\Omega) = \{ u \in L_2(\Omega) : \text{its derivative is in } L_2(\Omega) \},$$

$$H^{1,\perp}(\Omega) = \left\{ u \in H^1(\Omega) : \langle u, 1 \rangle_{L_2(\Omega)} = 0 \right\},$$

where $\langle \cdot, \cdot \rangle$ is the inner product in the space indicated by the subscript.

To find the solution to the SBVP, we consider that

$$g(\theta) = \sum_{k=1}^{\infty} g_k^1 \cos k\theta + g_k^2 \sin k\theta, \tag{7}$$

where g_k^1 and g_k^2 are the Fourier coefficients of g. The solution to the forward problem is given by

$$\begin{aligned} V(\theta) &= A(g)(\theta) = u(R_2, \theta) \\ &= \sum_{k=1}^{\infty} a_k g_k^1 \cos k\theta + a_k g_k^2 \sin k\theta, \end{aligned} \tag{8}$$

where

$$a_k = \frac{2R_1^{k+1} R_2^k}{k[(\sigma_1 - \sigma_2)R_1^{2k} + (\sigma_1 + \sigma_2)R_2^{2k}]}. \tag{9}$$

The linear operator $A : L_{2,\perp}(S_1) \longrightarrow L_{2,\perp}(S_2)$ is defined as $A(g) := u|_{S_2}$, where $u \in H^{1,\perp}(\Omega)$ is the solution to the SBVP.

3.3. Stable algorithm for the inverse source problem

For the inverse problem, we consider that the exact (ideal) measurement $V(\theta) = \sum_{k=1}^{\infty} V_k^1 \cos k\theta + V_k^2 \sin k\theta$ is known, where V_k^1 and V_k^2 are the Fourier coefficients of V. Using Equation (8), we obtain the Fourier coefficients of the source, g.

$$\begin{aligned} g_k^i =& \frac{V_k^i}{a_k} = \frac{k(\sigma_1 - \sigma_2)}{2R_1}\left(\frac{R_1}{R_2}\right)^k V_k^i + \\ & \frac{k(\sigma_1 + \sigma_2)}{2R_1}\left(\frac{R_2}{R_1}\right)^k V_k^i, \quad i = 1, 2. \end{aligned} \tag{10}$$

In practice, the measurement has errors for various reasons, such as errors from the measurement device, truncation errors, and the application of filters to eliminate unwanted signals (signal contamination). A methodology to implement FIR filters on FPGAs, which are the basis of the discrete wavelet transform, is presented in [15]. The error in the measurement is reflected in its Fourier coefficients, which must be considered carefully since the inverse source problem is numerically unstable. More specifically, the error in the coefficients is reflected in the term $\left(\frac{R_2}{R_1}\right)^k$, $k = 1, 2, \ldots$ of the coefficients in Equation (10). Note that as these terms grow with increasing k, even a small error in the measurement V can result in significant changes in the location of the source. More precisely, if we know V_δ instead of V, and consider measurement $V_\delta(\theta) = \sum_{k=1}^{\infty} V_{k,\delta}^1 \cos k\theta + V_{k,\delta}^2 \sin k\theta$, with $\|V - V_\delta\|_{L_2(S_2)} \leq \delta$, the Fourier coefficients of the recovered sources are given by Equation (11),

$$\begin{aligned} g_{k,\delta}^i =& \frac{V_{k,\delta}^i}{a_k} = \frac{k(\sigma_1 - \sigma_2)}{2R_1}\left(\frac{R_1}{R_2}\right)^k V_{k,\delta}^i + \\ & \frac{k(\sigma_1 + \sigma_2)}{2R_1}\left(\frac{R_2}{R_1}\right)^k V_{k,\delta}^i, \quad i = 1, 2, \end{aligned} \tag{11}$$

However, the terms $\left(\frac{R_2}{R_1}\right)^k V_{k,\delta}^i$ amplify the errors, and the series of the recovered source may not converge. To address this numerical instability, the Tikhonov functional is used [5]:

$$J_{\alpha(\delta)}(g) = \frac{1}{2}\|A(g) - V\|_{L_2(S_1)}^2 + \frac{\alpha(\delta)}{2}\|g\|_{L_2(S_2)}^2, \tag{12}$$

where $\alpha(\delta) > 0$ is the Tikhonov regularization parameter, which can be chosen using the L-curve criterion [16], and $\|\cdot\|_{L_2(S_i)}^2$ denotes the norm of $L_2(S_i)$, for $i = 1, 2$. To find the unique minimum of $J_{\alpha(\delta)}$ we must solve the normal equations [5], given by:

$$[A^*A + \alpha(\delta)I](g) = A^*V, \tag{13}$$

where $A^* : L_{2,\perp}(S_2) \to L_{2,\perp}(S_1)$ is the adjoint operator of A, which is given by [7]:

$$A^*(h)(\theta) = \frac{R_2}{R_1}\sum_{k=1}^{\infty} a_k h_k^1 \cos k\theta + a_k h_k^2 \sin k\theta, \tag{14}$$

where $h(\theta) = \sum_{k=1}^{\infty} h_k^1 \cos k\theta + h_k^2 \sin k\theta$ and a_k is given by Equation (9).

After substituting into the normal Eqs. (13), we obtain the regularized solution:

$$g_{\alpha(\delta)}(\theta) = \sum_{k=1}^{\infty} A_k(\alpha) \left[V_{k,\delta}^1 \cos k\theta + V_{k,\delta}^2 \sin k\theta \right], \tag{15}$$

where

$$A_k(\alpha) = \frac{a_k R_2}{(a_k)^2 R_2 + \alpha(\delta) R_1}. \tag{16}$$

Remark: When $\alpha = 0$ in (15), the Fourier coefficients of the recovered source coincide with the coefficients given in Equation (11).

Equation (15), obtained from the Tikhonov functional (12), provides us with a stable algorithm for recovering the regularized source $g_{\alpha(\delta)}$ from the measurement V_δ. This algorithm will be implemented on the FPGA, and in the next Section, we will illustrate algorithm (15) by implementing examples in MATLAB as a first step.

4. Numerical examples: MATLAB implementation

In this Section, we illustrate the source identification algorithm by considering the forward and inverse problems for different sources. We have developed MATLAB programs to validate the algorithm for its computational implementation. The following Section will describe the FPGA implementation of the source identification algorithm, and we will compare the hardware implementation with the MATLAB implementation using the same examples. The region Ω is described in Section 3.1, and it is shown in Figure 1.

To illustrate the algorithm, we built synthetic examples as follows:

1. We took some values for parameters σ_1, σ_2, R_1, and R_2, and defined a source g on S_1.
2. We solved the boundary value problem (1)-(5).
3. We computed the exact measurement $V = u|_{S_2}$, using Eq. (8) for $N = 16$, which was chosen by numerical tests.
4. To emulate the measurement (with error), we added an appropriate random error to the coefficients V_k^1 and V_k^2, where $k = 1, 2...$, using the *rand* function of MATLAB. Hence, we obtained coefficients $V_{k,\delta}^1$ and $V_{k,\delta}^2$, $k = 1, 2...$, of the measurement with error V_δ, which satisfies $\|V_\delta - V\|_{L_2(S_2)} \leq \delta$.
5. We obtained the regularized solution to the inverse problem by taking $N = 16$ in Eq. (15), i.e., we used
$$g_{\alpha(\delta),N}(\theta) = \sum_{k=1}^{16} A_k(\alpha) \left[V_{k,\delta}^1 \cos k\theta + V_{k,\delta}^2 \sin k\theta \right], \tag{17}$$
as an approximate (regularized) solution, where $A_k(\alpha)$ is given in (16). In the examples below, we considered $\delta = 0.1$ and $\alpha = 10^{-3}$ (which were chosen numerically).

Example 1. *In this example, we set $R_1 = 1$, $R_2 = 1.2$, $\sigma_1 = 3$, and $\sigma_2 = 1$, and consider the exact source g given by*

$$g(x, y) = f(x, y) - \frac{1}{m(S_1)} \int_{S_1} f(x, y) ds, \tag{18}$$

for all $(x, y) \in S_1$, where $f(x, y) = e^{-\frac{\|(x,y)-(a_1,a_2)\|^2}{2\beta^2}}$, for $(x, y) \in S_1$, $(a_1, a_2) = (0, 1) \in S_1$, $\beta^2 = 0.1$, and $m(S_1) = 2\pi R_1$. In polar coordinates (r, θ), g is given by

$$g(\theta) = f(R_1, \theta) - \frac{1}{2\pi} \int_0^{2\pi} f(R_1, \theta) d\theta, \text{ for all } \theta \in [0, 2\pi], \tag{19}$$

where $f(R_1, \theta) = e^{\frac{R_1^2 \cos(\theta - \theta_0)}{\beta^2}}$, for all $\theta \in [0, 2\pi]$, $\theta_0 = \frac{\pi}{2}$, and $\beta^2 = 0.1$.

We approximate the exact source g by its truncated Fourier series g_N using the first $N = 16$ terms. The Fourier coefficients g_k^1, g_k^2, $k = 1, 2, \ldots, N$ are obtained numerically using the *quadl* function of MATLAB. In this case, the exact measurement V is generated using Equation (17).

The relative error between the exact source g and the recovered source $g_{\alpha(\delta),N}$, denoted by $RE_{S_1}(g_{\alpha(\delta),N}, g)$, is given by

$$\begin{aligned} RE_{S_1}(g_{\alpha(\delta),N}, g) &= \|g_{\alpha(\delta),N} - g\|_{L_2(S_1)} / \|g\|_{L_2(S_1)} \\ &= 0.12. \end{aligned}$$

Figure 2 shows the exact measurement V and the measurement with error V_δ for $\delta = 0.1$.

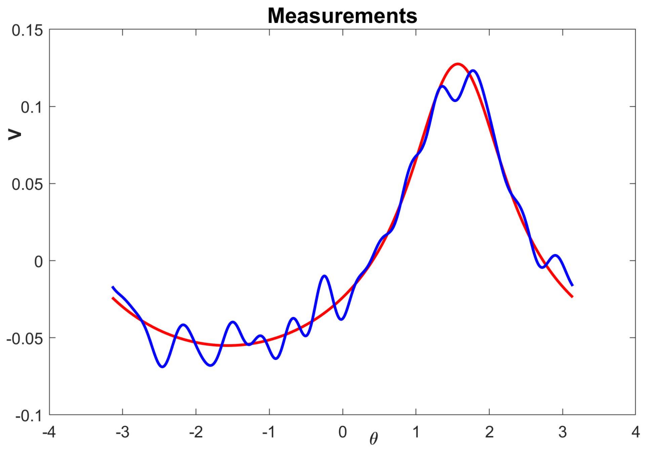

Figure 2. Exact measurement in red and recovered measurement in blue. The error in the exact measurement is obtained by adding a random error to the exact measurement using the *rand* function of MATLAB.

Figure 3 shows the plot of the exact source and the recovered source without regularization. We observe the necessity to apply regularization methods.

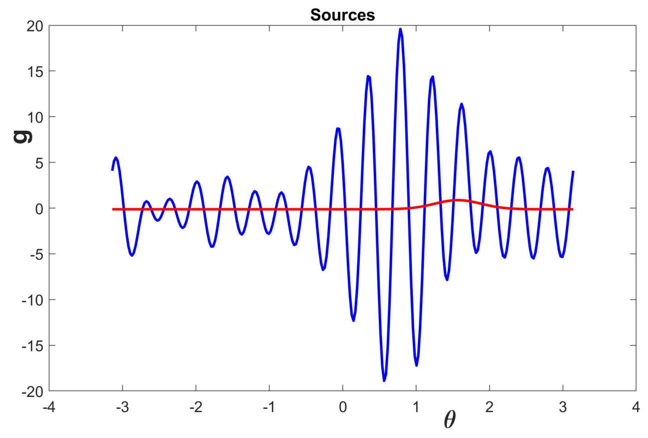

Figure 3. Exact source in red and recovered source without regularization in blue. In this case, the recovered source is obtained using the coefficients given by Equation (11), and it is far from the exact source due to the numerical instability of the inverse source problem.

Figure 4 shows the plot of the exact and the recovered source with regularization.

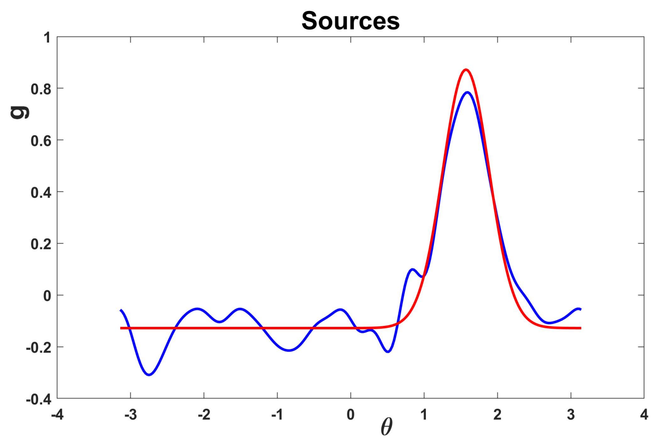

Figure 4. Exact source in red and recovered source, with regularization, in blue. In this case, the regularized source given by (17), which is obtained using Tikhonov regularization, allows us to handle numerical instability to get a good approximation of the exact source.

Example 2. *We consider the same values for R_1, R_2, σ_1, and σ_2 as in the previous example. We consider the following function, $g(x, y) = ye^x + e^y + x^2$ which in polar coordinates is given by $g(\theta) = sin(\theta)e^{cos(\theta)} + e^{sin(\theta)} + cos^2(\theta)$. Using Equation (8), we found the solution to the forward problem V, i.e., to find the ideal (exact) measurement V, we have to find the solution to problem (1)-(5), and then restrict it to the boundary S_2. The noisy data V_δ is obtained as in the previous example. Figure 5 shows the exact measurement V (in red), and the measurement with error V_δ (in blue).*

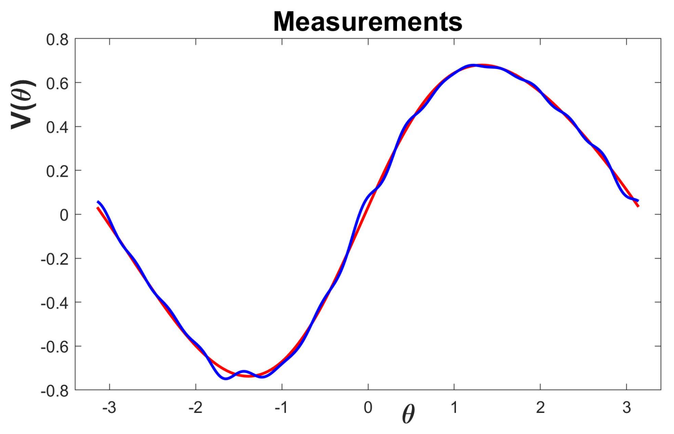

Figure 5. Exact measurement in red and measurement with error in blue. The latter is obtained by adding a random error to the exact measurement, which is done using the *rand* function of MATLAB.

Figure 6 shows the plot of the exact source and the recovered source without regularization. As in the previous example, we also observe the necessity of applying regularization methods.

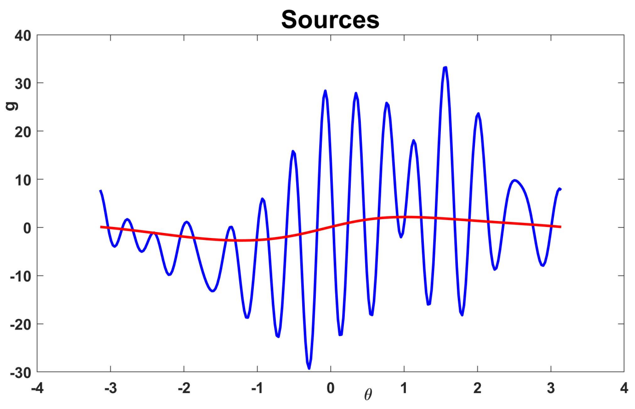

Figure 6. Exact source in red and recovered source, without regularization, in blue. In this case, the recovered source is obtained using the coefficients given by Equation (11), and it is far from the exact source due to the numerical instability of the inverse source problem.

Figure 7 shows the exact and recovered sources when regularization is applied. The relative error between the exact source g and the recovered regularized source, $g_{\alpha(\delta),N}$, denoted by $RE_{S_1}(g_{\alpha(\delta),N}, g)$, is given by

$$RE_{S_1}(g_{\alpha(\delta),N}, g) = 0.054.$$

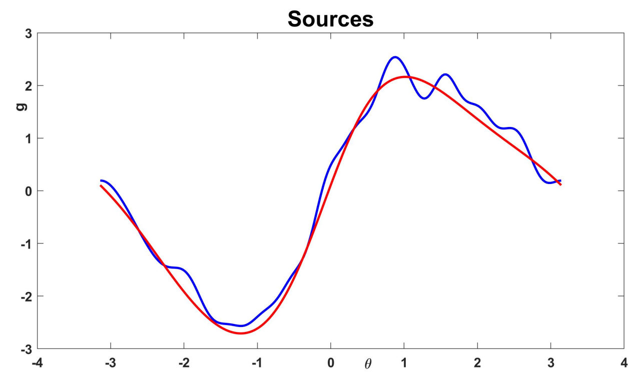

Figure 7. Exact source in red and recovered source with regularization in blue. In this case, the regularized source, given by Equation (17), allows us to obtain a stable approximation of the source.

Example 3. *We consider the same values for R_1, R_2, σ_1, and σ_2 as in the previous example. We consider the following square-wave function given in polar coordinates by*

$$g(\theta) = \begin{cases} -1, & \text{if } -\pi \leq \theta < 0, \\ 1, & \text{if } \quad 0 \leq \theta < \pi. \end{cases} \tag{20}$$

We found the solution V to the forward problem and the noisy data V_δ to the boundary S_2, as in the previous examples. Figure 8 shows the exact source g (red) and its approximation by Fourier series g_N (blue), truncating Fourier series (7) until the first $N = 16$ terms. Figure 9 shows the exact measurement V (in red) and the measurement with error V_δ (in blue).

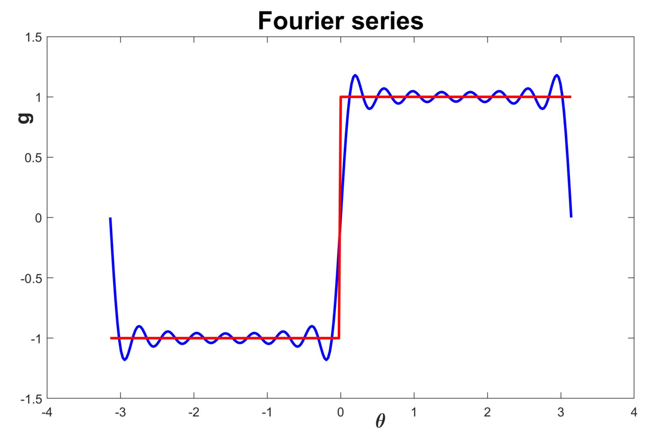

Figure 8. Plot of the exact square-wave function g (red) and its approximation by Fourier series (blue). We observe the Fourier series oscillations and the presence of the Gibbs phenomenon. This jump function is frequently used in an electronic signal.

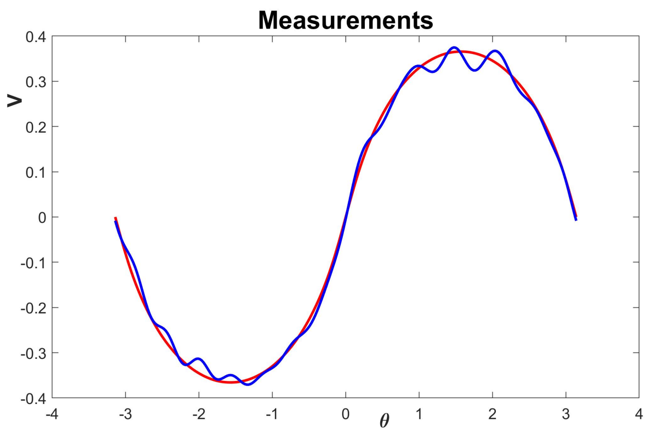

Figure 9. Exact measurement in red and measurement with error in blue. The measurement with error is obtained by adding a random error to the exact measurement, which is done using the MATLAB function *rand*.

Figure 10 shows exact and recovered sources without regularization. As in the previous examples, we can observe the importance of regularization methods to obtain stable solutions to the inverse problem.

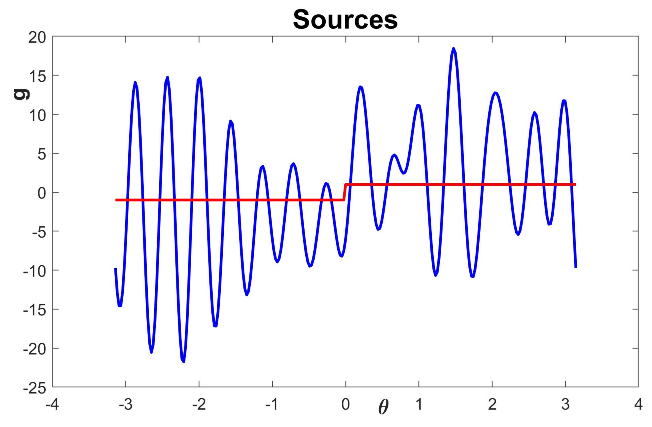

Figure 10. Exact source in red and recovered source without regularization in blue. In this case, the recovered source is obtained using the coefficients given by Equation (11), and it is far from the exact source due to the numerical instability of the inverse source problem.

Figure 11 shows the exact and recovered sources when regularization is applied. The relative error between the exact source g and the recovered source $g_{\alpha(\delta),N}$, denoted by $RE_{S_1}(g_{\alpha(\delta),N}, g)$, is given by

$$RE_{S_1}(g_{\alpha(\delta),N}, g) = 0.051.$$

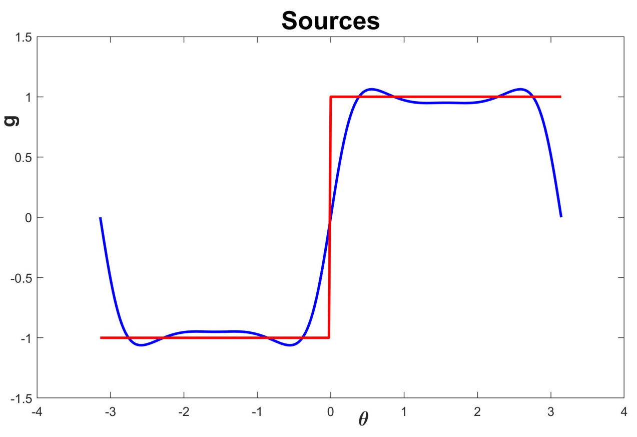

Figure 11. Exact source in red and recovered source with regularization in blue. The Tikhonov regularization method is used to handle numerical instability and obtain a good approximation to the exact source. The regularized solution acts as a smoothing filter.

5. FPGA implementation

The implementation of algorithms in hardware devices is crucial as it allows for optimal use of hardware resources. For instance, we can determine the number of coefficients in a series to achieve an appropriate approximation and the number of bits used to represent a real number (precision). We can also determine if the implementation can be duplicated for multichannel problems.

Analyzing the arithmetic operations in Equation (17), we conclude that a feedback implementation is appropriate, as it minimizes hardware resources. The operations in Equation (17) will be repeated as follows:

1. Perform the products $A_1V_1^1cos(\theta)$ and $A_1V_1^2sin(\theta)$. The two's complement format is chosen for the number representation, as the algorithm involves signed arithmetic operations.
2. Sum the results from the previous step.
3. Temporarily store the result.
4. Perform the products $A_2V_2^1cos(2\theta)$, $A_1V_2^2sin(2\theta)$ and add them to the temporary result.
5. Repeat the process until term $N = 16$.

The advantage of this implementation is that the arithmetic modules can be reused. The hardware implementation components include:

1. A double memory block ROM to store the coefficients $A_kV_k^1$ y $A_kV_k^2$.
2. A DDS Xilinx module (see [17]) to generate the values of $sin(k\theta)$ and $cos(k\theta)$ for $k = 1, ..., 16$.
3. Multiplexers to maintain synchrony in the Control Section.

Considering this, Figures 12 and 13 show the required hardware resources for architecture modes 1 and 2, respectively.

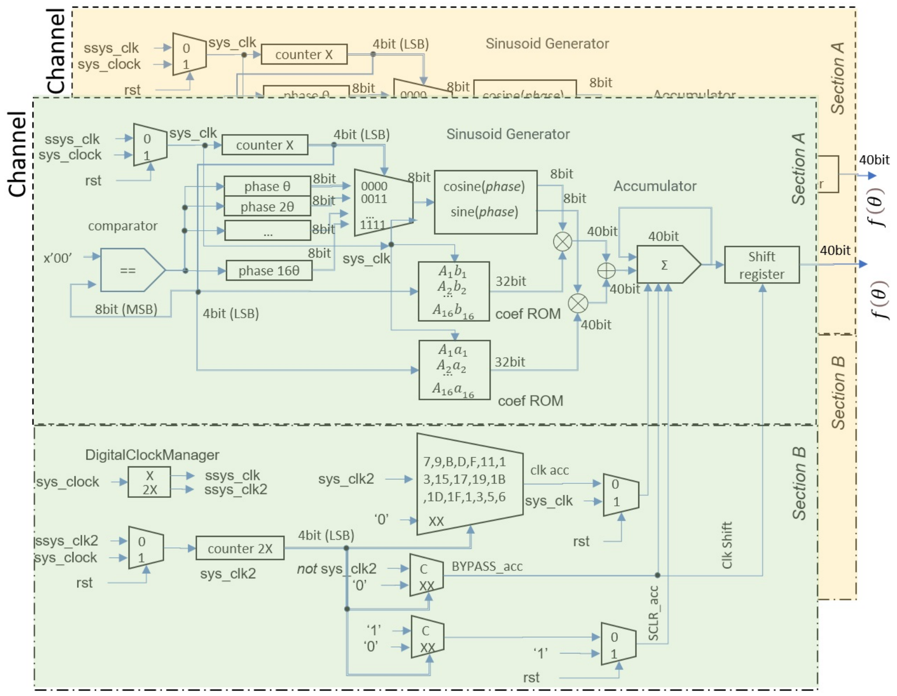

Figure 12. Feedback architecture Mode 1. In this mode, the control unit (Section B) is independent in each channel, as well as the phase generation and the processing unit. This allows for operation at different speeds.

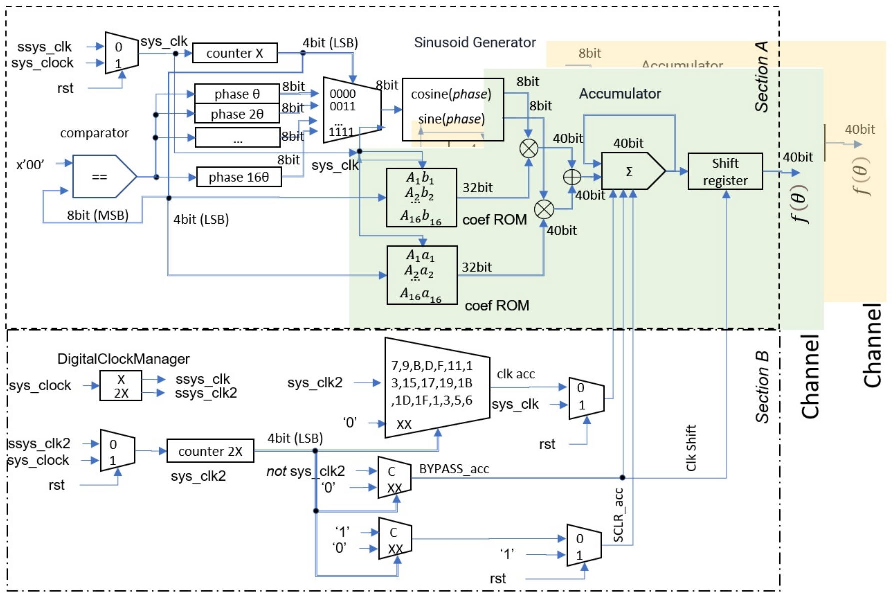

Figure 13. Feedback architecture Mode 2. In this mode, the Control Section is shared with the algorithm operations Section, resulting in a coupled synchronization of the trigonometric base operations

The implementation results presented were obtained using VHDL code, synthesized in the Vivado IDE c2018.2.2 (64 bit) and tested in the Xilinx® 7 FPGA series, which comprises four FPGA families that address the complete range of system requirements, ranging from low-cost, small form factor, cost-sensitive, high-volume applications, to ultra high-end connectivity bandwidth, logic capacity, and signal processing capability for the most demanding high-performance applications: Spartan-7 xc7s100fgga484, Virtex-7 xc7v585tffg1157, Kintex-7 xc7k70tfbg484, and Artix-7 xc7a35tcpg236. Tests were carried out using a 100MHz base clock.

The instrumentation developed for the implementation of the algorithm consists of 48 coefficients, which are involved in expression (13) with $N = 16$. Each coefficient is 16 bits long. So, there are 16 coefficients for A_k, V_k^1, and V_k^2. The DDS component was configured with a partition of 8 bits for the phase and amplitude of the sinusoidal functions. The product requires 40 bits since we have 32-bit and 8-bit numbers. Finally, the accumulated sum and the resulting output both require 40 bits.

5.1. Architecture description

Figures 12 and 13 show the architecture for mode 1 and mode 2, respectively. Both architectures contain two channels, each of which is composed of:

1. Section A: This section contains the arithmetic operations.
2. Section B: This section contains the control of the operations in Section A, which synchronizes the pipeline operations.

The design of the architecture consists of three blocks, each of which contains:

1. The trigonometric base.
2. Linear combination of the elements of the base.
3. Control module.

Trigonometric base. The counterX module performs the following three fundamental actions through its four least significant bits (LSB):

1. Acts as an 8-bit selector control in the 16-to-1 multiplexer.
2. Increments the consecutive value from 1 to 16 in resolution.
3. Synchronizes the addressing and reading of the ROM memory.

To evaluate the functions $cos(\theta)$ and $sin(\theta)$ and address the ROM memory, a latency of two cycles is used. This maintains synchronization between the DDS module and the ROM memories.

Linear combination of the trigonometric base. The following latencies are required for different operations:

1. A two-cycle latency for the DDS module to reflect the sine and cosine values on the data bus and for the memory ROM module to reflect the coefficients simultaneously on the data bus. Both modules work in parallel.
2. A one-cycle latency for each operation (product and sum). Those operations are performed in series.
3. A one-cycle latency to store the result in the accumulator.
4. A latency of 16 cycles to obtain $g(\theta)$.

Thus, a new value of $g(\theta)$ is calculated every 16 cycles. With the main clock of the used FPGA operating at 100 MHz, each point in the partition has an output frequency of 6.25 MHz.

Construction of the Control Section. The control section, shown in Section B of Figures 12 and 13, corresponds to the synchronization of the algorithm. Its design allows for efficient use of throughput, which provides the amount of data processed in each clock cycle. The pipeline plays a crucial role in this Section, as the input phase generation data must be synchronized to get the output data, which corresponds with a new value of $g(\theta)$. A flux output data is generated in this form, which must be aligned. The processing stages of the control section must be applied in each clock cycle.

The Control Section is utilized to manipulate the phase shifts of the main clock to ensure the overall synchronization of the architecture. The design of this Control Section operates using two clock signals, which are generated by the Digital Clock Manager (DCM) block. The signal *sys_clk* is used for constructing the *Trigonometric Base* and *Linear combinations of the Trigonometric base* Sections. To maintain the synchronizations of the accumulator and shift register elements, *ssys_clk*2 must be twice as fast as *ssys_clk*. Through a multiplexer array, *ssys_clk*2 serves as a control element for resetting the accumulator content, the clock signal, and the bypass signal. The bypass signal allows for the result of the sum to be passed on to the shift register. This results in the first phase of the sum being obtained.

We developed two architectures, labeled Mode 1 and Mode 2, which are identical in the case of a single channel. For multiple channels, the Mode 2 architecture takes advantage of the characteristics of the algorithm to reduce hardware consumption. This reduction can be observed in Section A of Figure 13, where the generation of the functions $\sin kx$ and $\cos kx$, where $k = 1, 2, ...$, is shared across different channels. The Control module, shown in Section B, is shared by both architectures. The Mode 1 architecture allows for the operation at different speeds, which is beneficial when working with either fast or slow signals. The Mode 2 architecture reduces resource consumption by exploiting the characteristics of the source identification algorithm.

5.2. Resource description

We report the resource consumption of the used Xilinx 7 series FPGAs. The resources are:

1. LUTs (Lookup Tables): These contain the logical elements that determine the output from one or multiple inputs. They are essentially truth tables created from the description of the VHDL program.
2. FFs (Flip Flops): Sequential logical elements with one bit of memory.
3. RAM blocks: Each block has a storage capacity of 32 K-bit.
4. DSP blocks: Specialized blocks for product, sum, and accumulation operations for signed numbers in two's complement format. These operations are called: multiply-accumulate (MAC).
5. Power consumption: Determines the energy consumption of the system.

Section B of Figure 12 shows that the channel blocks share control, which synchronizes the MAC operations, phase generation, and the values of the trigonometric functions that make up the base. From Section B in Figure 12, it can be seen that the channel blocks share control, which synchronizes MAC operations, phase generation, and sine and cosine values (the base). Combining the common control operations with sine and cosine generation reduces the number of hardware resources.

Figures 14 and 15 show the resources used in the proposed architectures. In the case of one channel, the two architectures have the same resource consumption. Architecture Mode 2 presents a lower resource consumption for more than one channel.

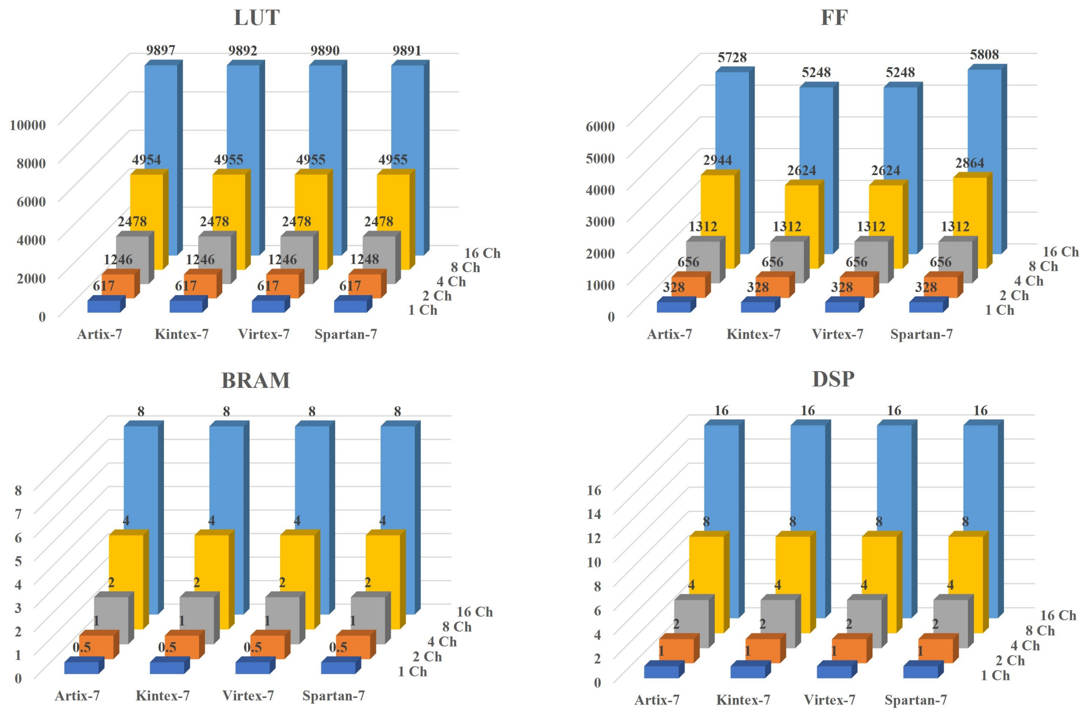

Figure 14. Resources mode 1. We observe that as the number of channels increases, the resource consumption linearly increases in all FPGAs, and there is no difference in consumption.

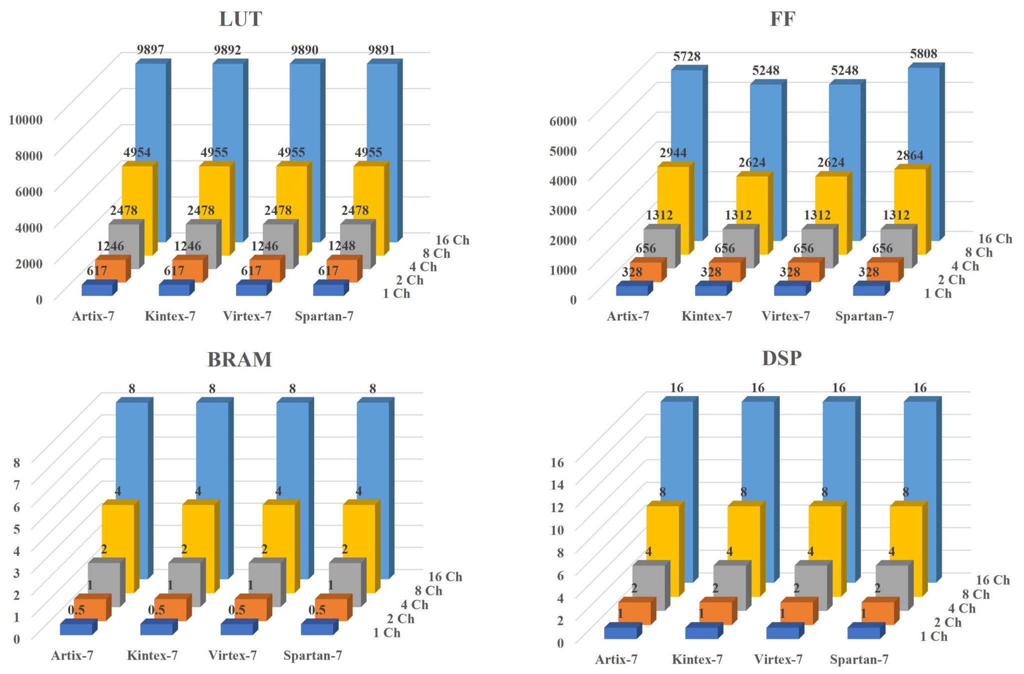

Figure 15. Resources mode 2. We observe that the resource consumption is lower for LUT, FF, and BRAM compared to mode one as the number of channels increases. In particular, there is no increase in BRAM consumption as the architecture uses the same resources. The DSP consumption is the same for both architectures.

Figure 16 shows the results of the following relation:

$$\frac{\text{mode 1- mode 2}}{\text{mode 1}} = 1 - \frac{\text{mode 2}}{\text{mode 1}}. \tag{21}$$

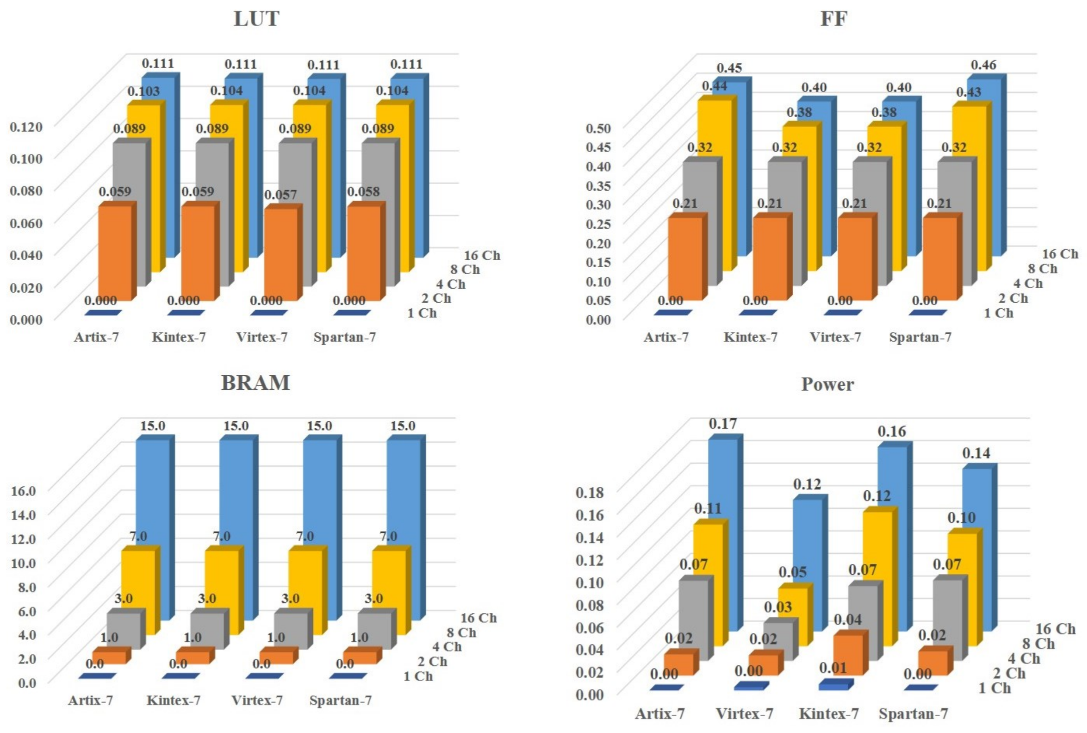

Figure 16. Using Equation (21), we calculate the percentage of resources consumed by mode 2 compared to mode 1. As the number of channels increases, this percentage also increases. For LUTs, the percentage is nearly the same. For FF, both Kintex-7 and Virtex-7 have the same performance.

The relation gives the percentage of resources that mode 2 consumes compared to mode 1. For two channels, 5.9% fewer LUT resources are obtained for Artix and Kintex. For four channels, 8.9% fewer LUT resources are obtained for all FPGAs. For 16 channels, 11% fewer LUT resources are obtained for all FPGAs.

For two channels, 21% fewer FF resources are obtained for all FPGAs. For 16 channels, 40% to 46% fewer FF resources are obtained. For 16 channels, 15 times less BRAM consumption is obtained, as architecture mode 2 uses only a DDS module. For 16 channels, 17%, 12%, 16%, and 14% less power consumption is obtained in Artix, Virtex, Kintex, and Spartan, respectively.

Figure 17 shows the power consumption. Mode 2 has lower current consumption. Artix-7 has the lowest consumption for both modes.

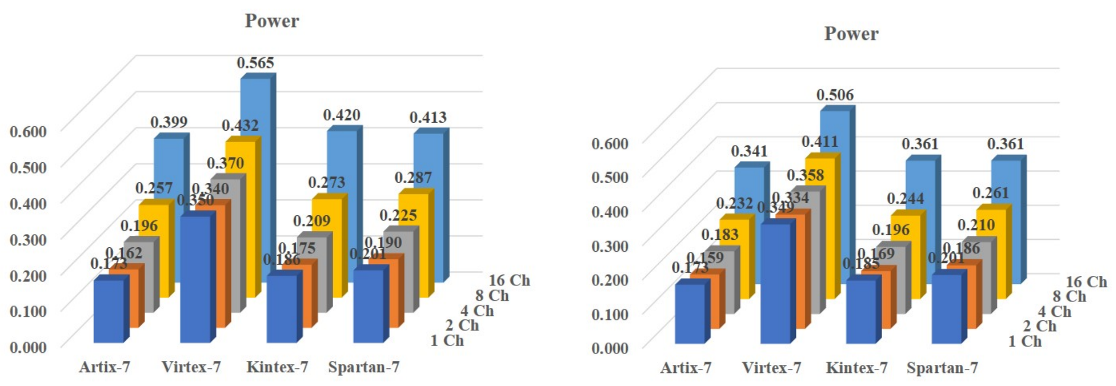

Figure 17. Left: Power mode 1. Right: Power mode 2. We observe that mode 2 has lower current consumption. Of both modes, Artix-7 has the lowest consumption.

6. Validation of the hardware implementation

In this section, we present the results of the FPGA implementation of the same examples given in Section 4. We compare them with the MATLAB implementation to validate the FPGA implementation.

Figures 18–22 show the implementation of the examples from Section 4. We obtain similar results to the MATLAB implementation. In Figures 23–25, we present the graphics of the error between the recovered sources using MATLAB and the FPGA.

Figures 19, 21 and 26 show the recovered source without regularization provided by the FPGA implementation. We can see that the recovered sources are far away from the exact source, as in the MATLAB implementation. Figures 18, 20 and 22 show the recovered source provided by the FPGA implementation using regularization. We emphasize that we obtain similar results as in the MATLAB implementation.

We define the Norm of the Maximum Absolute Error (NMAE) between the regularized recovered source $g_{\alpha,N}$ and the recovered implemented source $g^I_{\alpha,N}$ as follows:

$$\text{NMAE} = \max_{\theta}\left\{\left|g_{\alpha,N}(\theta) - g^I_{\alpha,N}(\theta)\right|\right\}.$$

We define the Maximum Absolute Error (MAE) as the following function:

$$\text{MAE}(\theta) = \left|g_{\alpha,N}(\theta) - g^I_{\alpha,N}(\theta)\right|, \text{for all } \theta \in [-\pi,\pi].$$

Figures 23–25 show the MAE between the sources recovered using MATLAB and the sources recovered using the FPGA implementation. We observe that the NMAE is less than 10^{-3}. This result confirms that the FPGA implementation of the algorithm is reliable.

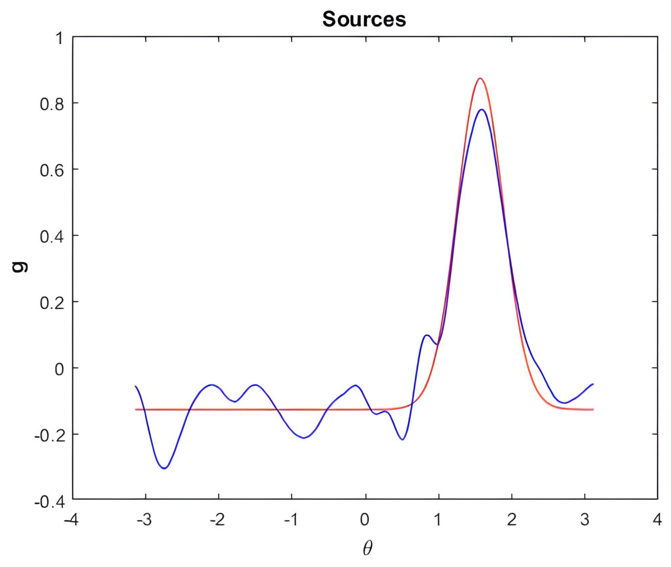

Figure 18. Exact source is shown in red, and the result provided by the FPGA implementation using Equation (17) (with regularization) is shown in blue. Similar results are obtained as in the MATLAB implementation, thus validating the FPGA implementation.

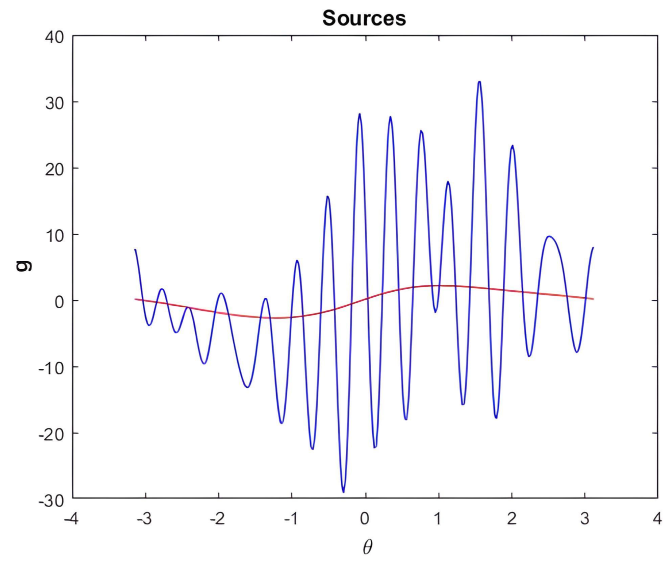

Figure 19. Exact source in red. In blue, the approximate source obtained by the implementation in the FPGA using Equation (11) (without regularization). Similar results as in the MATLAB implementation are obtained. Thus, we validate the FPGA implementation. The recovered source is obtained by applying the algorithm to the measured data.

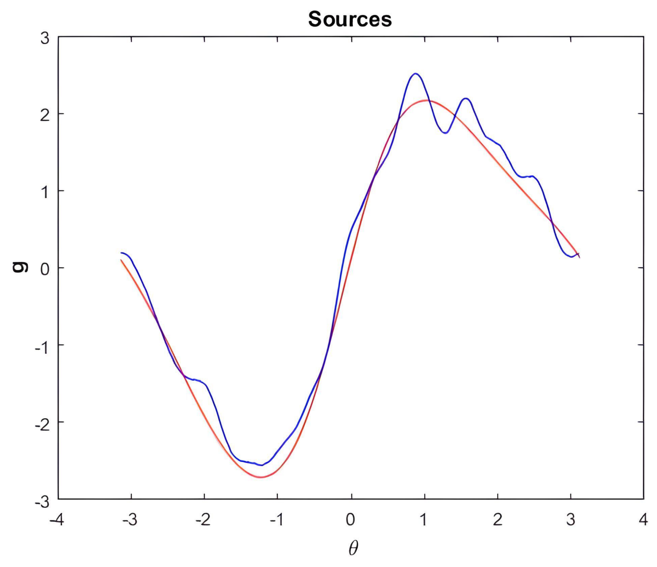

Figure 20. Exact source in red. Source recovered by the FPGA implementation using Equation (17) (with regularization) in blue. Similar results as in the MATLAB implementation are obtained. Thus, we validate the FPGA implementation.

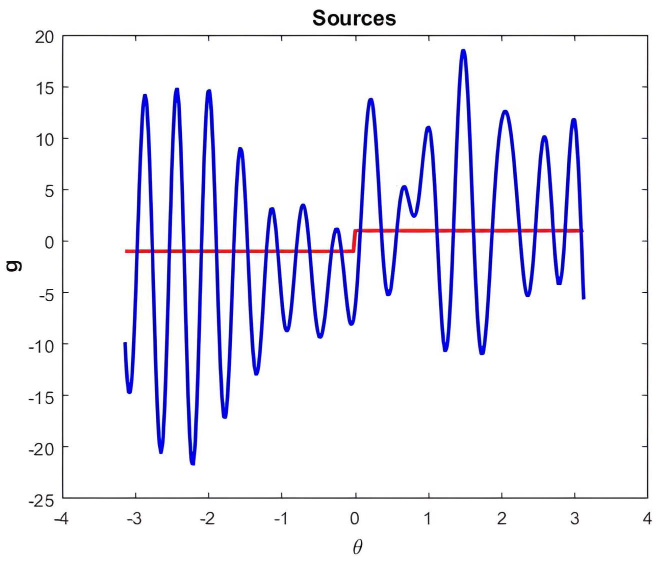

Figure 21. Exact source in red. In blue, the approximate source obtained by the FPGA implementation using Equation (11) (without regularization). Similar results as in the MATLAB implementation are obtained. Thus, we validate the FPGA implementation.

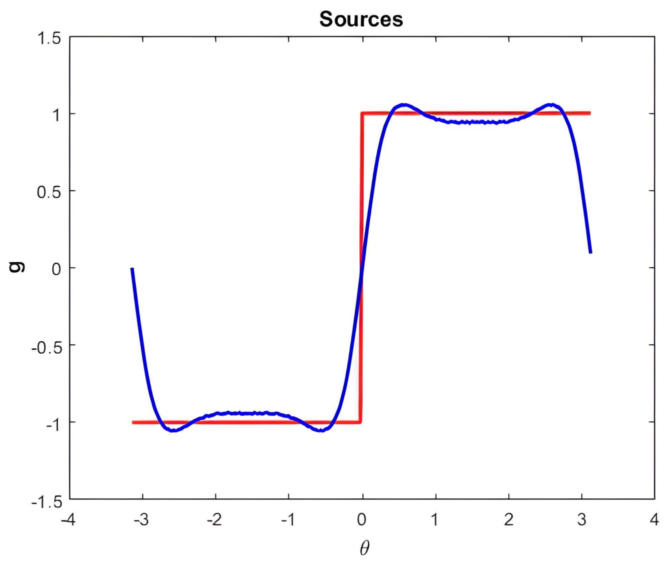

Figure 22. Exact source in red. Result provided by the FPGA implementation using Equation (17) (with regularization) in blue. Similar results as in the MATLAB implementation are obtained. Thus, we validate the FPGA implementation.

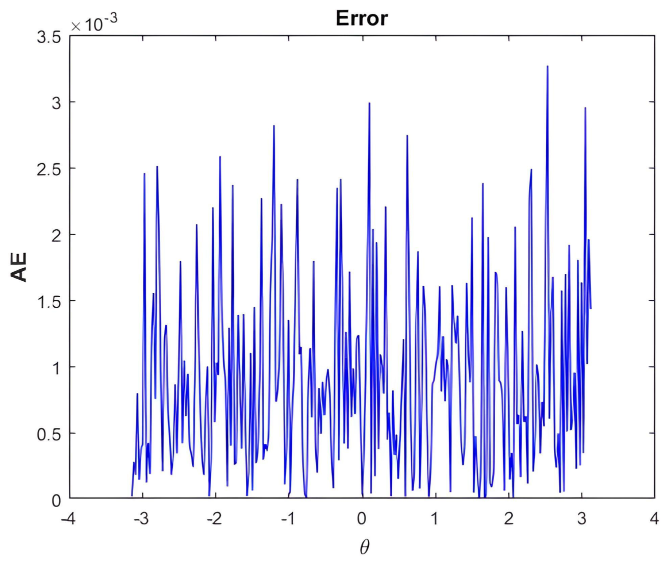

Figure 23. MAE between the source recovered using MATLAB programs and the one recovered by the FPGA implementation (note the scale in the y-axis). The NMAE is less than 10^{-3}, which shows that the FPGA implementation gives almost the same results as the MATLAB implementation.

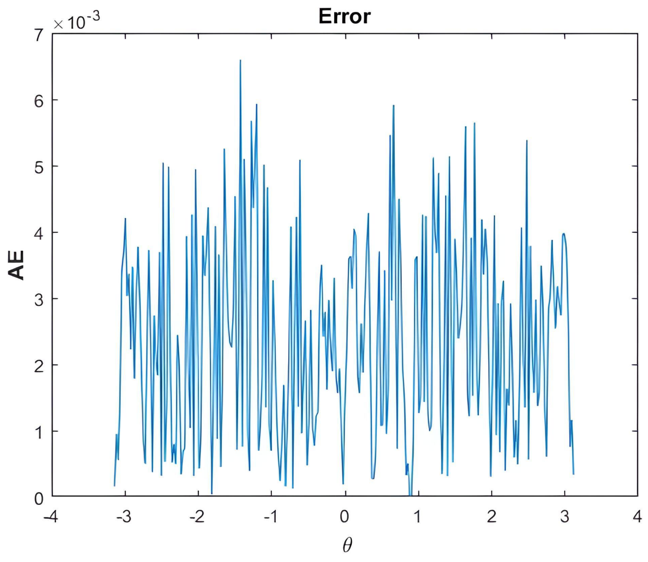

Figure 24. MAE between the source recovered using MATLAB programs, and the one recovered by the FPGA implementation. Note the scale. The NMAE is less than 10^{-3}, which shows that the FPGA implementation gives almost the same results as the MATLAB implementation.

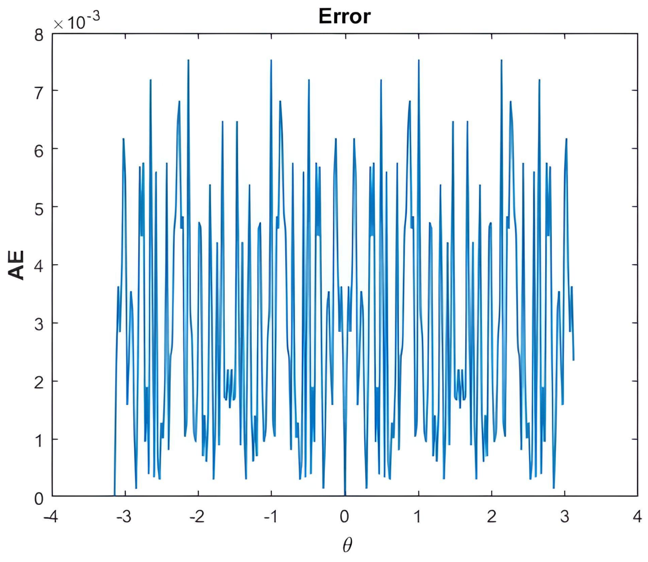

Figure 25. MAE between the source recovered using MATLAB programs, and the one recovered by the FPGA implementation. Note the scale. The NMAE is less than 10^{-3}, which shows that the FPGA implementation provides results that can be compared to the MATLAB implementation.

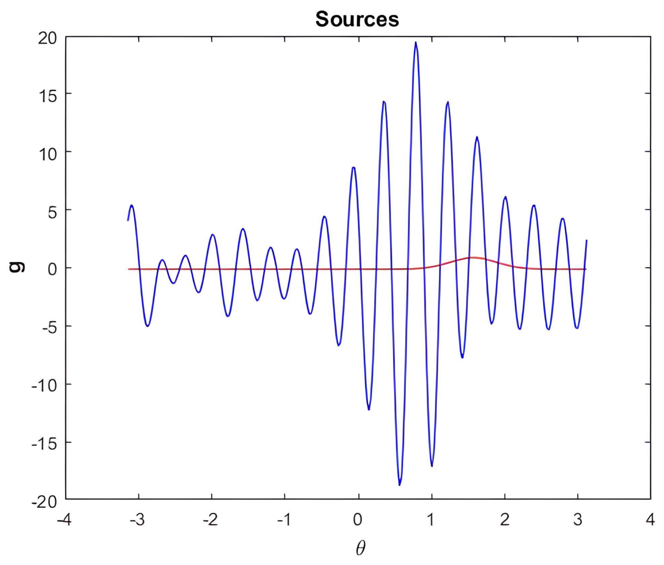

Figure 26. Exact source in red. In blue, the approximate source obtained by the FPGA implementation using Equation (11) (without regularization). We obtained similar results to the MATLAB implementation, thus validating the FPGA implementation.

7. Discussion

The source identification problem arises in many applications, such as inverse electroencephalography, inverse electrocardiography, and inverse geophysics. An FPGA is an integrated circuit that allows the implementation of algorithms and other complex mathematical operations. In this work, we implemented a source identification algorithm considering a non-homogeneous medium made up of two homogeneous media, and sources located on the interface separating the homogeneous media. The algorithm is expressed as a Fourier series expansion considering the circular geometry.

Since the source identification problem presents numerical instability, we used the Tikhonov regularization method and a cut-off of the Fourier expansion to handle the numerical instability. The Tikhonov regularization parameter and the truncation term were chosen through numerical testing. This work presents a hardware architecture with two designs to implement the source identification algorithm. These architectures were tested on four FPGAs from the 7 Series of Xilinx: Spartan-7 xc7s100fgga484, Virtex-7 xc7v585tffg1157, Kintex-7 xc7k70tfbg484, and Artix-7 xc7a35tcpg236. Each architecture contains two channels, each of which is composed of two sections: one for the arithmetic operations and one for the control of the operations. We report the resource consumption of the used 7 Series of Xilinx FPGAs. These resources include LUTs, RAM blocks, FFs, DSP blocks, and power consumption. One of the developed architectures could be used for problems with a larger number of channels. One example of this case is inverse electroencephalography, in which many inverse source problems must be solved per second.

8. Conclusion

This paper presented two FPGA implementations of a stable source identification algorithm in a circular, non-homogeneous region expressed in Fourier series expansions. The first architecture (mode 1) is an extended implementation that considers all operations involved in the algorithm, while the second architecture (mode 2) takes advantage of the algorithm's properties to apply a pipeline system and reuse hardware resources, resulting in improved performance. The implementation was tested using synthetic examples and produced comparable numerical results for both architectures. The error between the exact source and the recovered source was found to be less than 10^{-3}, which is considered an acceptable result. One example tested involved a square function source, where the recovered source showed smoothing properties, as demonstrated by the absence of variation in the Fourier series expansion. With its reduced hardware resource consumption, the second architecture could be utilized for multichannel problems. The results presented here can be extended to concentric spheres and complex geometries. In these cases, the trigonometric base must be changed and implemented. In the case of spheres, the base is the spherical harmonics. We can choose a base generated for the finite element method or the differences method for complex geometries. So, the architectures developed for implementing the inverse source algorithm could allow the generation of portable devices for problems in different areas such as engineering and medicine.

Author Contributions: Conceptualization, J.J.O.O., C.A.H.G., M.M.M.C., J.R.C.S. and J.J.C.M.; methodology, J.J.O.O., C.A.H.G., J.R.C.S. and J.J.C.M.; software, J.J.O.O., J.R.C.S. and J.J.C.M.; validation, J.J.O.O., J.R.C.S. and J.J.C.M.;; formal analysis, J.J.O.O., J.R.C.S., C.A.H.G., M.M.M.C. and J.J.C.M.; investigation, J.J.O.O., J.R.C.S., C.A.H.G., M.M.M.C. and J.J.C.M.; resources, J.J.O.O., J.R.C.S., C.A.H.G., M.M.M.C. and J.J.C.M.; data curation, J.J.O.O., J.R.C.S. and J.J.C.M.; writing—original draft preparation, J.J.O.O., J.R.C.S., C.A.H.G., M.M.M.C. and J.J.C.M.; writing—review and editing, J.J.O.O., J.R.C.S., C.A.H.G., M.M.M.C. and J.J.C.M.; visualization, J.J.O.O., J.R.C.S., C.A.H.G., M.M.M.C. and J.J.C.M.; supervision, J.J.O.O., C.A.H.G., J.R.C.S. and J.J.C.M.; project administration, J.J.O.O., J.R.C.S., C.A.H.G., M.M.M.C. and J.J.C.M.; funding acquisition, J.J.O.O., J.R.C.S., C.A.H.G., M.M.M.C. and J.J.C.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Council of Science and Technology in Mexico (CONACYT), VIEP-BUAP, and PRODEP-SEP.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Xilinx, 7 Series FPGAs Data Sheet: Overview, DS180 (v2.6.1), 8 September 2020. Available online: https://docs.xilinx.com/v/u/en-US/ds180_7Series_Overview (accessed on 10 April 2023).
2. Morín-Castillo, M.M.; Netzahualcoyotl-Bautista, C.; Conde-Mones, J.J.; Oliveros-Oliveros, J.J.; Santillán-Guzmán, A. Stable identification of sources associated with epileptic focus on the cerebral cortex, *Rev. Mex. Ing. Biomed.* **2019**, *40*, 1–14.
3. Morín-Castillo M.M.; Arriaga-Hernández J.; Cuevas-Otahola B.; Oliveros-Oliveros, J.J. Analysis of Dipolar Sources in the Solution of the Electroencephalographic Inverse Problem, *Mathematics*, **2022**, *10*, 1–22.
4. Gockenbach, M.S. *Linear inverse problems and Tikhonov regularization*, The Carus Mathematical Monographs, 32, Publisher: The Mathematical Association of America, 2016.
5. Kirsch, A. *An Introduction to the mathematical theory of inverse problems*, 2nd ed.; Vol 120, Publisher: Appl. Math. Sci., Springer, New York, 2011.
6. Algredo-Badillo, I; Conde-Mones J.J.; Hernández-Gracidas C.A.; Morín-Castillo M.M.; Oliveros-Oliveros J.J.; Feregrino-Uribe, C. An FPGA-based analysis of trade-offs in the presence of ill-conditioning and different precision levels in computations, *PLoS One*, **2020**, *15(6)*, 1–26.
7. Morín, M.M.; Netzahualcoyotl, C.; Oliveros, J.J.; Conde J.J.; Juárez, H. Stable identification of sources located on separation interfaces of two different homogeneous media, *Adv. Differ. Equ. Control Process.* **2019**, *20(1)*, 53–97.
8. Hennessy, J.L.; Patterson D.A. *Computer architecture: a quantitative approach*; 5th ed.; Publisher: Elsevier, 2011.
9. lsuwailemo, A.M. Real-time FPGA-based Image Enhancement Using Histogram Projection Technique for Uncooled Infrared Imagers, *J. King Saud Univ., Eng. Sci.*, **2007**, *21*, 15–22.
10. Nayak, M.R.; Bag, J.; Sarkar, S.; Sarkar, S.K. Hardware implementation of a novel water marking algorithm based on phase congruency and singular value decomposition technique, *Int. J. Electron. Commun.* **2017**, *71*, 1–8.
11. Saric, R.; Jokic, D.; Pokvic, L.G.; Badnjecvic, A. FPGA-based real-time epileptic seizure classification using Artificial Neural Network, *Biomed. Signal Process. Control*, **2020**, *62(2)*, 1–10.

12. Conde Mones, J.J.; Estrada Aguayo, E.R.; Oliveros Oliveros, J.J.; Hernández Gracidas, C.A.; Morín Castillo, M.M. Stable Identification of Sources Located on Interface of Nonhomogeneous Media, *Mathematics*, **2021**, *9(16)*, 1–23.
13. Arias-Cruz, J.A.; Morín-Castillo, M.M.; Oliveros-Oliveros, J.J.; and Gutiérrez-Arias, J.E. Stable identification of sources located on the cerebral cortex from EEG over the scalp, *Rev. Mex. Fis.* **2023**, *69*, 1–8.
14. Badia, E.A.; Duong, T.H. Some remarks on the problem of source identification from boundary measurements, *Inverse Probl.* **1998**, *14*, 883–891.
15. Harize, S.; Benouaret, M.; Doghmane, N. A methodology for implementing decimator FIR filters on FPGA, *Int. J. Electron. Commun.* **2013**, *67(12)*, 993–1004.
16. Hansen, P.H. The L-curve and its use in the numerical treatment of inverse problems. In *Computational Inverse Problems in Electrocardiography*, P. Johnston (Ed.), WIT Press, Southampton, 2001, pp. 119–142.
17. Xilinx, DDS Compiler v6.0, PG141, 21 January 2021. Available online: https://docs.xilinx.com/v/u/en-US/pg141-dds-compiler (accessed on 10 April 2023).