An Architecture for Hardware Realization of Compressive Sensing Gradient Algorithm

Stefan Vujović, Miloš Daković, Irena Orović, Srdjan Stanković
Faculty of Electrical Engineering
University of Montenegro
Podgorica, Montenegro
e-mail: stefanv@ac.me

Abstract — An architecture for hardware realization of the Gradient algorithm for sparse signal reconstruction is proposed. Gradient algorithm is recently proposed and generally belongs to convex optimization class of algorithms. It is an iterative algorithm where missing samples are reconstructed by using a procedure of gradient-based concentration improvement. The proposed scheme assumes that sparse domain of signal is Discrete Fourier domain. It is interesting to note that this algorithm performs well even in the case of almost sparse signals. The proposed architecture could be modified easily for other transform domains. The scheme is composed of blocks that are suitable for FPGA implementation. Finally, this architecture gives a much deeper insight into the algorithm providing better understanding of this algorithm, which will facilitate its applications.

Keywords - Compressive sensing; gradient algorithm; hardware realization; signal reconstruction

I. INTRODUCTION

Sparse signals are those having small number of nonzero coefficients in a certain transform domain compared to the total number of coefficients. Although signal sparsity seems as a strict constraint, note that a wide class of signal interesting for practical applications such as radars, sonars, biomedicine, satisfy this property. In that sense, the Compressive sensing [1]-[7] is an area particularly interesting for these signals. It is the reason why it attracts attention of many researchers around the world. The first issue that arises in the compressive sensing theory [1] - [3], is related to the measurement matrix, which provides relation between signal $x$ of length $N$ and its $M$ measurements ($M<<N$). Measurements are taken by using a random matrix allowing high incoherence with transform basis. After an incomplete set of samples/measurements is acquired, a reconstruction algorithm has to be applied. For this purpose a few class of algorithms are developed: convex relaxations algorithms and greedy algorithms (based on matching pursuit, thresholding, etc) [8]-[12]. In this paper, recently proposed gradient algorithm [13] belonging to the convex group is realized. This algorithm is very efficient in the case of almost sparse signals [14]. This algorithm performs very well even in the case of natural images.

Having in mind that the main drawback of iterative algorithms lay in the fact that they are time consuming, the hardware implementation will make them much more attractive for practical applications. The idea of this paper is to provide an architecture for hardware realization of the gradient algorithm. Note that the gradient algorithm can be applied in different sparsity domains, using various measures of sparsity. However, here we will consider the discrete Fourier domain and $l_1$-norm as a measure of concentration.

The proposed architecture consists of several blocks, based on the standard components such as FFT [15], square root [16],[17], adders, multipliers, which make it suitable for FPGA implementation.

The paper is organized as follows. After introduction, the theoretical background about compressive sensing is provided in Section II. Section III is related to the Gradient algorithm for sparse signal reconstruction. The architecture for hardware realization is proposed and described in Section IV. The conclusion is given in Section V.

II. THEORETICAL BACKGROUND

Let $x$ be a discrete-time signal of length $N$ which have sparse presentation in some transformation domain. This sparse presentation will be denoted by $X$. The relation between these two domains can be expressed in matrix notation as:

$$X = Wx \quad (x = W^{-1}X),$$

where $W$ is $N\times N$ matrix, and $W^{-1}$ is its inverse. Let assume that we have just $M$ ($M<<N$) available samples of signal $x$ denoted as $y = x(n_i)$, where $i=1,2,...,M$. We can say that these $M$ samples are linear measurements of signal $X$ which are obtained as

$$y = AX.$$

In this case matrix $A$ is obtained from matrix $W^{-1}$ by eliminating rows corresponding to missing samples, i.e. preserving rows corresponding to available samples. The idea is to reconstruct the missing samples.

The problem of missing samples reconstruction can be defined as:

$$\min \|x\|_0 \quad \text{subject to} \quad y = AX. \quad (1)$$

Since (1) is an NP-hard combinatorial approach, commonly $l_1$-norm is used instead od $l_0$-norm. Also, in practical signal
processing application $l_0$-norm is very sensitive to any small values, even to an error of quantization. The reconstruction task is then reformulated as:

$$\min \|X\|_0 \quad \text{subject to} \quad y = AX. \quad (2)$$

It is very important to note that solutions of (1) and (2) are the same if matrix $W$ satisfies restricted isometry property [2]. Here, we will focus on signals having sparse presentation in Discrete Fourier Domain (DFT). It means that $W$ is $N \times N$ DFT matrix with elements $W(n,k) = \exp(-2i\pi nk/N)$.

III. GRADIENT ALGORITHM FOR SPARSE SIGNAL RECONSTRUCTION

Recently proposed Gradient algorithm for sparse signal reconstruction is one of the convex optimization algorithms. It is based on $l_1$-norm minimization. The idea is to reconstruct signal samples in time-domain by minimizing concentration of sparse presentation of signal. Concentration of signal in sparse domain could be measured by some other concentration measures [18], but commonly $l_1$-norm is used.

Implementation of the algorithm [13],[14] is described in the sequel. Suppose that the signal $x(n)$ has some missing samples or some wrong values due to the measurement error. The positions of missing/wrong samples are denoted by $n_i$, $i=1,2,...,M$, where $M$ is total number of missing/wrong samples.

**Step 0:** Form the signal $y^{(0)}(n)$, where (0) means that this is the initial iteration of the algorithm, defined as $y^{(0)}(n)=x(n)$ for available samples, and 0 for missing samples.

**Step 1:** Then we form two signals $y^{(k)}_n(n)$ and $y^{(k)}_m(n)$ for each missing sample at $n_i$ in each next iteration as:

$$y^{(k)}_n(n) = y^{(k)}(n) + \Delta \delta(n-n_i)$$

$$y^{(k)}_m(n) = y^{(k)}(n) - \Delta \delta(n-n_i)$$

where $k$ is the iteration number.

**Step 2:** Estimation of the differential of the signal transform (here the DFT) measure is:

$$g(n_i) = \frac{1}{N} \left( \sum |DFT[y^{(k)}_n(n)] - \sum |DFT[y^{(k)}_m(n)] \right) \quad (3)$$

**Step 3:** A gradient vector $G$ with the same length as the signal $x(n)$ is formed. At the positions of available samples, this vector has value $G(n_i) = 0$. At the positions of missing samples its values are $g(n_i)$, calculated by (3).

**Step 4:** Correct the values of $y(n)$ iteratively as:

$$y^{(k+1)}(n) = y^{(k)}(n) - G(n).$$

By repeating the presented iterative procedure, the missing values will converge to the true signal values, producing the minimal concentration measure in the transformation domain.

IV. ARCHITECTURE FOR HARDWARE REALIZATION OF THE GRADIENT ALGORITHM

The architecture for hardware realization of the Gradient algorithm is given in Fig. 1. The “Correction block” is a crucial part of the architecture and it is a reason why this block is considered separately and presented in Fig. 2. Both block schemes will be explained in detail in this section.

An inputs to the architecture presented in Fig. 1 are two vectors: first one denoted by $y^0$ is signal having zeros at the position of missing samples, while the second one is flag vector $p$. This vector has the same length as the vector $y^0$. At the positions of missing samples this vector has value 1, while at the positions of available samples, it has zero values. Since this is an iterative algorithm, the iteration number is denoted by $k$. At the beginning, for $k=0$, a vector $y^0$ is stored in register denoted as reconstructed signal ($y^*$). This will also be the output after the required number of iterations is reached. In each iteration, vector $G$ is subtracted from $y^k$. In this way, iterative update of vector $y^k$ (in the register), expressed by $y^{(k+1)}(n) = y^{(k)}(n) - G(n)$, is achieved.
The vector $\mathbf{G}$ is calculated (in each iteration) in correction block based on $\mathbf{y}^k$, position vector $\mathbf{p}$, and constant $\Delta$. The constant is calculated as the maximal absolute value of signal $\mathbf{y}^0$.

The core of the algorithm is implemented within the “Correction block”, and the architecture for hardware implementation is presented in Fig. 2. The inputs to the block are vector $\mathbf{y}$ (in Fig. 1. it is denoted as $\mathbf{y}^k$), position vector $\mathbf{p}$, and constant $\Delta$. For the position of each signal sample (1, 2, ..., $N$), two signals are formed. The first one is formed by correcting sample at the actual position for $+\Delta$, and the second one is formed by correcting the sample for $-\Delta$. For example, if input to this block is signal $\mathbf{y}$, then for the first signal sample at k-th iteration, we calculate the signals:

$$y_{11}^k(n) = y^k(n) + \Delta \delta(n-1),$$
$$y_{21}^k(n) = y^k(n) - \Delta \delta(n-1).$$

For the second sample the two signals are formed as:

$$y_{12}^k(n) = y^k(n) + \Delta \delta(n-2),$$
$$y_{22}^k(n) = y^k(n) - \Delta \delta(n-2).$$

This procedure is done for each signal sample. Further, each of the formed signals passes through the subsequent blocks. First of them is the fast Fourier transform (FFT) block with a known hardware realization in FPGA technology [15]. This block transforms signal $\mathbf{y}$ to the domain where it has sparse presentation. Since the DFT is complex valued, for each DFT coefficient, there are two parts, real and imaginary. In order to find the absolute value of each complex valued DFT coefficient, the real and imaginary parts are squared, added and the square root is calculated last. The square root implementation in FPGA technology is also studied in the literature [16], [17]. After these calculations are performed for each signal sample, the absolute values of the DFT
coefficients are added together for each $n$, which results in the signal transform measure. Then the differential of the signal transform (Step 3 in the algorithm) is calculated as a subtraction of appropriate transform measures and the result is multiplied by the corresponding element of vector $p$. Note that each value corresponding to position of available sample will be zero valued since vector $p$ has zeros at the positions of available samples. The other values corresponding to missing samples will be unchanged since they will be multiplied by 1. Values obtained after the multiplications are divided by signal length $N$. In this way a gradient approximation for each samples position is obtained as in (3). At the positions of available samples these approximations have zero value.

It is very important to emphasize that all described calculations are performed in parallel for each signal $y_k$. This means that the speed of algorithm will depend only on the blocks performing processing of a single line i.e. one sample processing. This is an advantage of the proposed architecture, and can perform fast reconstruction of signals where the time of reconstruction dominantly depends on speed of square root block (Fig.2.), since the operation latency is about 25 clock cycles, and the issue rate is 24 clock cycles for standard architecture [19]. The next block which needs the most time is FFT block (processing time of this block depends on the signal length), where number of clock cycles is of order $\log_2 N$. Generally, the number of clock cycles $P$ needed to perform Correction block calculation is $P=1+\log_2 N+3+1+50+1+1+1+3$ for adder, FFT block, square block, adder, sqrt block, adder, adder, subtractor, multiplier (multiplying with 1 or 0) and multiplier respectively. In addition to $P$, one more clock cycle is required for subtractor from Fig.1. In addition to aforementioned, 1+7 clock cycles (absolute value and finding of maximal value) which are performed just once at the beginning of the algorithm when $\Delta$ is calculated are required. Altogether, the total number of clock cycles is approximately $k(1+P)+1+7$, where $k$ is number of iterations. In order to limit the processing time needed for the reconstruction, a fixed number of iterations can be performed, for example 100 iterations would be enough in most of the applications.

V. CONCLUSION

An architecture for hardware realization of the Gradient algorithm for sparse signal reconstruction is proposed. The scheme is decomposed to the basic level of well-known implementation blocks. Since this algorithm adapts the values for missing samples iteratively, it is very important to reduce as much as possible the processing time of each iteration. Therefore, the parallel computing of gradient approximation for each sample is proposed. This means that the signal length will not affect the performance significantly since the main calculation will be done simultaneously for each signal sample. The length of signal affects only the FFT calculation. Further research will be focused to the optimization of the number of required FFT blocks in a way to reduce the number of components. The proposed scheme is suitable for FPGA realization, which will be the next step in our research.

REFERENCES