Fast OMP algorithm and its FPGA implementation for compressed sensing-based sparse signal acquisition systems

Shirshendu Roy | Debiprasad P. Acharya | Ajit K. Sahoo

Electronics and Communication, National Institute of Technology, Rourkela, Odisha, India

Correspondence
Shirshendu Roy, Electronics and Communication, National Institute of Technology, Rourkela, Odisha, India. Email: shirshenduroy88@gmail.com

Abstract
Compressed sensing-based radio frequency signal acquisition systems call for higher reconstruction speed and low dynamic power. In this study, a novel low power fast orthogonal matching pursuit (LPF-OMP) algorithm is proposed for faster reconstruction of sparse signals from their compressively sensed samples and the reconstruction circuit consumes very low dynamic power. The searching time to find the best column is reduced by reducing the number of columns to be searched in successive iterations. A novel architecture of the proposed LPF-OMP algorithm is also presented here. The proposed architecture is implemented on field programmable gate array for demonstrating the performance enhancement. Computation of pseudoinverse in OMP is avoided to save time and storage requirement to store the pseudoinverse matrix. The proposed design incorporates a novel strategy to stop the algorithm without consuming any extra circuitry. A case study is carried out to reconstruct the RADAR test pulses. The design is implemented for \( K = 256 \), \( N = 1024 \) using XILINX Virtex6 device and supports maximum of \( K/4 \) iterations. The proposed design is faster, hardware efficient and consumes very less dynamic power than the previous implementations of OMP. In addition, the proposed implementation proves to be efficient in reconstructing low sparse signals.

1 | INTRODUCTION

High-frequency radio frequency (RF) signals, such as RADAR pulses, are sparse in nature in the transform domain. Exploiting this sparsity property, modern signal measurement systems use compressed sensing (CS) \([1,2]\) in place of other existing sampling techniques \([3]\) to acquire RF signals. CS-based acquisition systems can work with low speed analog-to-digital converters due to sampling at sub-Nyquist rate \([4]\). In CS-based sampling paradigm, random measurements are taken from the signal and then the original signal is recovered from the measurement samples using signal recovery algorithms. Orthogonal matching pursuit (OMP) \([5,6]\) is a well known recovery algorithm. Unlike the other greedy pursuit algorithms, OMP provides better performance with moderate computational complexity. OMP estimates a sparse signal by executing two steps in every iteration, viz., perform the atom searching (AS) and solve a least squares (LS) problem. In AS step, OMP identifies an atom or a column of the sampling matrix which gives maximum correlation with the current residual. Subsequently the signal is estimated by solving an LS problem.

The timing complexity of the AS step is very high as it is a linear function of the signal sparsity and the number of samples. Many techniques are reported in literature to reduce the timing complexity of AS step. In \([7]\) authors applied clustering algorithms to group the similar columns and reported a tree-based pursuit algorithm. But such algorithm has no reports of implementation. Researchers reported parallel selection of multiple columns to address the timing complexity problem in \([8,9]\). Multiple selection of columns reduces the timing complexity but with greater chances of choosing wrong columns.

Many implementations are reported based on either field programmable gate arrays (FPGAs) or application-specific integrated circuits. The LS problem is solved in different ways in current research works. The implementations of OMP reported in \([9–15]\) used modified Cholesky factorization \([16]\) to solve the LS problem. The LS problem is solved by lower-upper decomposition \([16]\) in \([17]\). QR decomposition \([16]\) is another powerful matrix factorization technique which is used

This is an open access article under the terms of the Creative Commons Attribution License, which permits use, distribution and reproduction in any medium, provided the original work is properly cited.

© 2021 The Authors. IET Circuits, Devices & Systems published by John Wiley & Sons Ltd on behalf of The Institution of Engineering and Technology.
in many implementations of OMP. The implementations in [18–22] preferred modified Gram–Schmidt algorithm [16] to perform QR factorization. Besides the matrix factorization-based solutions, matrix inversion bypass (MIB) technique reduces the computational complexity of OMP. An MIB-based implementation OMP is reported in [23]. Authors in [24] further reduce the complexity of OMP and achieved high signal-to-noise ratio.

Real-time CS-based sampling of high frequency periodic and RF pulses requires fast signal reconstruction. On the other hand, high digital overhead and high dynamic power consumption of the implementations of the recovery algorithms are other concerns in sampling practical sparse signals. In such applications, OMP implementations are required to have high speed of reconstruction, low hardware complexity and less power consumption. In addition, the hardware complexity must have less dependency on unknown signal sparsity. Previously in [25,26] we have presented hardware efficient architectures of OMP based on QR decomposition and incremental Gaussian elimination, respectively where same hardware resources are shared to perform different operations.

The classic OMP algorithm is reformulated to speed up the reconstruction process simultaneously reducing the dynamic power consumption. The proposed reformulated OMP is termed as low power fast OMP (LPF-OMP) algorithm. A novel architecture for the LPF-OMP algorithm is also proposed. The proposed hardware architecture is implemented on the FPGA platform for demonstration of performance. Low power devices are always desired in real-life portable applications where enhancing the battery life is one of the prime objectives. The speed enhancement is desirable in time critical applications. This algorithm can find use in portable RADAR systems, remote sensing systems like drones, unmanned underwater vehicles and many portable signal acquisition devices. This manuscript has the following major contributions:

1. The correlation step of the classic OMP algorithm is modified to search through different number of columns in successive iterations to reduce reconstruction time.
2. The OMP algorithm uses a partial evaluation of incremental QR decomposition by the modified Gram–Schmidt algorithm.
3. The computation of pseudoinverse is avoided which saves time and storage requirement to store pseudoinverse matrix.
4. A novel method to stop the algorithm is proposed which allows the design to be invariant to the signal sparsity.
5. The reformulated OMP algorithm in the aforesaid manner achieves low dynamic power consumption.

Organization of this manuscript is as follows: Section 2 describes the proposed LPF-OMP algorithm. First the classic OMP algorithm is briefed and then the modifications to that are illustrated. This section also describes the novel strategy to stop the algorithm and a strategy for estimation. In Section 3, the proposed architecture for hardware implementation is discussed in details. Section 4 describes the performance estimation of the LPF-OMP algorithm and its implementation. And finally the conclusions of this research are provided in Section 5.

2 | PROPOSED LOW POWER FAST OMP-BASED SPARSE SIGNAL RECONSTRUCTION

Real world signals can be sparse either in time or any other transformed domain. An m-sparse input signal \( x \in \mathbb{R}^{N \times 1} \) can also be written as \( x = \mathcal{D}s \), where \( \mathcal{D} \in \mathbb{R}^{N \times N} \) is the dictionary and \( s \) is the sparse representation of \( x \). CS-based reconstruction of signal \( x \) can be expressed as:

\[
y = \phi s + \eta
\]

where \( y \) is the measurement vector and \( \eta \) is the noise vector. The sampling matrix \( (\phi \in \mathbb{R}^{N \times N}) \) can be expressed as \( \phi = \psi \mathcal{D} \), where \( \psi \) is the sensing matrix which is used to take linear measurements from \( x \).

2.1 | Proposed LPF-OMP algorithm

The proposed LPF-OMP algorithm is shown in Algorithm 1. It differs from the original version of OMP algorithm in the technique of choosing suitable columns from the sampling matrix at each iteration. Initially two null sets \( \Lambda \) and \( \Omega \) are defined. The set \( \Lambda \) stores indices of the most correlated column selected at each iteration and the set \( \Omega \) stores the best set of columns. The constant parameters \( \varpi \) and \( \epsilon \) are also input to the algorithm. Signal estimation vector \( \hat{s} \) and the residual vector \( r \) are the outputs of OMP. Initial value of the residual vector \( r^0 \) is taken as \( y \).

**Algorithm 1 LPF-OMP algorithm for signal recovery**

**Objective:** Recovery of signal \( s \).

**Input:** Measurement vector \((y)\), sampling matrix \((\phi)\), maximum iteration \((I_{\text{max}})\), choice of \( \varpi \) and \( \epsilon \).

**Output:** Recovered signal \( \hat{s} \)

1: **Initialization** set \( r^0 = y \), \( \Lambda^0 = [ ] \), \( \Omega = [ ] \) and \( i = 0 \).
2: **while** stopping criteria is not true and \( i < I_{\text{max}} \) **do**
3: \[ \text{if } \text{mod}(i, \varpi) == 0 \] **do**
4: \[ \text{Identification: } \lambda = \arg \max_{j=1,\ldots,N} |\langle \phi_j, r^i \rangle| \]
5: \[ \Omega = \text{a set of best } W \text{ supports of } |\langle \phi_j, r^i \rangle| \]
6: \[ \text{else} \]
7: \[ \text{Identification: } \lambda = \arg \max_{j \in \Omega} |\langle \phi_j, r^i \rangle| \]
8: **end if**
9: Augment the index set \( \Lambda^{i+1} = \Lambda^i \cup \lambda \) and the matrix of chosen columns \( \hat{\phi}_{i+1} = [\hat{\phi}_1, \hat{\phi}_2] \)
10: **Estimation:** \( \hat{s}^{i+1} = \arg \min_{\tilde{s}} \| y - \hat{\phi}_{i+1} \tilde{s} \| \)
11: Compute the residual:\( r^{i+1} = y - \hat{\phi}_{i+1} \hat{s}^{i+1} \)
12: **Increment iteration counter:** \( i = i + 1 \)
13: **end while**
To denote the total number of iterations, \( I \) is used and \( i \) is a variable used for iteration count. The constant \( I_{\text{max}} \) stands for maximum iteration count and the parameter \( \epsilon \) is a threshold to determine stopping criteria.

In the classic OMP, the AS step finds the index of the column for which the magnitude of the inner product \( \langle \phi_i, r \rangle \) is maximum, where \( \phi_i \) is the \( i^{th} \) column of \( \phi \). This index and the respective column of the matrix \( \phi \) are augmented to \( \Lambda \) and \( \hat{\phi} \), respectively. In the estimation step, an LS equation is solved to estimate \( \hat{s} \). The solution to the LS equation is:

\[
\hat{s} = \hat{\phi}^+ y = ((\hat{\phi}^T \hat{\phi})^{-1} \hat{\phi}^T) y
\]

where \( \hat{\phi}^+ \) is the Moore-Penrose pseudoinverse of the under-determined matrix \( \hat{\phi} \). Residual is calculated at the final step. These steps are repeated until a halting criterion is true.

The AS step of the classic OMP algorithm is modified in this study to achieve high speed of reconstruction by reducing the timing complexity of AS step. In the iterations where the condition \( \text{mod}(i, m) = 0 \) is satisfied, a set \( \Omega \) is formed which contains the indices of the best \( W \) columns. Here, \( \Omega_w \) indicates the \( w^{th} \) index from set \( \Omega \). All other steps are evaluated in the same way as in the original OMP. In the next \( (w-1) \) iterations, only the columns which are indexed by \( \Omega \) are searched to find the best suitable column. This way the timing complexity of the AS step in the iterations which do not satisfy the above-mentioned condition, is reduced from \( N \) to \( W \). The constant parameter \( w \) can take values greater than one and can be tuned to produce better results. The size of the \( \Omega \) (which has \( W \) indices) depends on the value of \( w \). It is observed that the parameter \( W \) can be set to a minimum of \( 2w \) to produce an acceptable signal estimate.

In general, the power and speed are the trade-offs in design of integrated circuits (ICs). When a designer tries to improve one parameter the other gets degraded. Generally, speed improvements are done by using parallelism and more resources which in turn increase power consumption. Power improvement techniques at the circuit and architecture level reduce the speed. Power can also be significantly reduced when the algorithm is modified. Here we are not improving power at the cost of speed rather the algorithm is modified to achieve power reduction which also improves the speed due to involvement of less steps of computation.

### 2.2 Proposed stopping criteria

Conventionally there are three ways to stop the OMP algorithm.

1. Stopping after fixed \( m \) number of iterations.
2. Waiting until \( \|r\|_2 \) drops below a threshold level.
3. Stopping the algorithm when \( \max(\|r\|_2) \) drops below another threshold value.

Stopping after fixed iterations becomes impractical with no prior knowledge of signal sparsity. The other methods to stop the algorithm are based on progressive decay of residual. The second possibility demands extra circuitry for Euclidean norm computation. An alternative option is to make decision based on the value of \( \|r\|_2^2 \). The problem associated with the second and third option is the choice of threshold. They converge easily for noiseless situation. For compressed signals in presence of noise, the choice of threshold is very difficult. Also the length of the measurement vector is a key parameter to decide the threshold.

The stopping criteria based on the decaying residual work well for signals having fixed amplitudes with Gaussian sensing matrix but does not guarantee reconstruction of practical signals which are approximately sparse. Also, the threshold value changes drastically with the type of signal. The stopping criterion used in this work is:

\[
\frac{\langle r, r \rangle}{\langle y, y \rangle} < \epsilon
\]

whenever the ratio in the left hand side is less than a predetermined threshold value, the algorithm stops.

### 2.3 Proposed signal estimation strategy

QR-based matrix factorization technique is used to solve the LS problem. The pseudoinverse is computed as:

\[
\hat{\phi}^+ = [R^{-1}0] Q^T
\]

where \( Q \in \mathbb{R}^{K \times 1} \) is an orthogonal matrix and \( R \in \mathbb{R}^{1 \times 1} \) is an upper triangular matrix. The signal estimation is computed as follows:

\[
\hat{s} = \hat{\phi}^+ y = R^{-1} Y
\]

where \( Y = \langle Q^T, y \rangle \), inner product between \( Q^T \) and \( y \). The signal estimation \( \hat{s} \) can be computed in two ways. One possible way is to compute the pseudoinverse and then multiply with \( y \). On the other hand, \( Y \) can be evaluated first and then the result is multiplied by inverse of \( R \). The later method saves the timing complexity of pseudoinverse computation and storage requirement to store pseudoinverse matrix. The second method is followed which eliminates the need for computation of pseudoinverse matrix.

### 3 Proposed architecture for LPF-OMP algorithm

Compared to our earlier study [25], a simpler and faster architecture is proposed here to implement LPF-OMP algorithm. The proposed architecture uses a single vector multiplication unit (VMU) to perform all the vector arithmetic operations involved in OMP algorithm. The proposed VMU
consists of two major blocks, viz., a multiply–accumulate (MAC) unit and an adder tree as shown in Figure 1. Here, DSP block-based MAC unit is used. Each MAC is controlled by a control signal \( f_{ni} \) that configures it to perform different operations. Dispatcher unit provides inputs to the VMU. The bank of MUXes (M1) selects between \( y \) and \( \phi \). A dedicated divider performs division operation in QR stage and computes reciprocal in matrix inversion stage. Memory block \( \phi_{mem} \) stores the matrix \( \phi \) and \( y_{mem} \) receives the measurement samples from the external world. Serial data from \( y_{mem} \) are captured by a register bank (RB). Memory block \( q_{mem} \) stores \( Q \) and also \( y \) to decrease the logic complexity. The block \( r_{mem} \) temporarily stores \( e_i \) and \( r^j \) where \( e_i \) represents the orthogonal columns of matrix \( Q \).

### 3.1 Sorting

The proposed scheme for evaluation of the AS step is shown in Figure 2. In this step, inner product \( \langle \phi_p, r^j \rangle \) is evaluated to search for the most correlated column. Initially \( y \) is written to the \( r_{mem} \) memory through M1 and the MAC unit. During this step, MAC performs the operation \( z = c - z \) when the control signal \( f_{ni} \) is “00”. The measurement vector \( y \) is also written into \( q_{mem} \) to reduce hardware complexity of the dispatcher unit. The dispatcher unit first selects \( r^j \) through both the port to compute \( \langle r^j, r^i \rangle \) and then selects \( r^i \) and \( \phi_j \) to find column which gives maximum correlation. The memory block \( \phi_{mem} \) is accessed in such a way that the columns of \( \phi \) can be read in parallel. These two inner products are evaluated in pipeline. During the evaluation of the inner products, the control input \( f_{ni} \) is “01” and MAC unit is configured to perform the operation \( z = a \times b \). The inner product \( \langle r^i, r^j \rangle \) is saved in a controlled register to evaluate the stopping criteria.

In the first iteration, \( N \) number of columns are searched to find the column which is most correlated to \( y^0 \) and also best \( W \) columns are identified for next \((w-1)\) iterations so that only \( W \) columns are needed to be searched. Now when the iteration count is equal to \( \omega \), again \( N \) number of columns are to be searched and the same process is to be repeated for next \((w-1)\) iterations. This way the timing complexity of the AS step is reduced.

Sort block (SB) gives the index of the most correlated column and also gives the indices of the best \( W \) columns. It also generates an \( index \) signal, which is used to start the QR factorization and other functions. The basic network (BN) block is designed to find the maximum of a serial input data stream and is shown in Figure 3. The BN block is the basic subblock of the SB. The proposed architecture of the SB is shown in Figure 4. There are \( W \) number of BN blocks connected in series to sort \( W \) elements. The absolute value of the inner product \( (ip)_j \) is fed to the SB. Indices of the columns are identified either from a counter which counts from 0 to \( N \) or from the set \( \Omega \) depending on the iteration count. Best \( W \) indices are stored in RAM3 memory and the value of \( index \) is stored in RAM2 memory in each iteration. The value of \( index \) is the index of the most correlated column.

Comparison is done in parallel with the multiplication and in \((W-1)\) clock cycles best \( W \) columns are identified. This sorting process is carried out in parallel with QR factorization in iterations where the condition is satisfied as shown in Algorithm 1. Timing complexity for correlation process is reduced due to the difference in number of columns searched in successive iterations. Total time complexity for the AS step for the proposed LPF-OMP is \( T_{AS} = N \times q + W \times (I-q) + 15I \) where latency for the inner product evaluation is 15 and \( q = \text{quo}(I-1, \omega) + 1 \). The function \( \text{quo}(I-1) \) returns only the value of quotient when \((I-1)\) is divided by \( \omega \). A comparison of

![Figure 1](image1.png) Proposed architecture of vector multiplication unit

![Figure 2](image2.png) Proposed scheme for the evaluation the atom searching step
the time complexity due to the AS step of three different versions of OMP is shown in Table 1.

### 3.2 Modified Gram–Schmidt-based QR decomposition

#### 3.2.1 Partial incremental QRD

The modified Gram–Schmidt algorithm is used here to perform the QR factorization. The LS step is solved by computing pseudoinverse matrix in the prior QR-based implementations [20–22] of OMP. Also, columns of Q are normalized by an extra normalizing step which involves computation of square root reciprocal. Previously in [25], we have avoided the normalization step but computation of pseudoinverse still degrades the performance of OMP. In this study, evaluation of QR factorization is simplified by avoiding the computation of pseudoinverse and also avoiding the normalization step.

---

**Algorithm 2**

Sparse signal estimation

**Input:** Augmented matrix $\hat{\phi}$, iteration count $i$.

**Output:** Signal estimation $\hat{s}$.

1: **Initialization:** The column vector $e_{imp}$ is set to null vector.
2: if $i = 1$ do
3: $e_i = \hat{\phi}_i$
4: else
5: if $i > 1$ do
6: for $j \leftarrow 1$ to $(i - 1)$ do
7: $R_{ji} = \langle e_j, \hat{\phi}_i \rangle$
8: $e_{imp} = e_{imp} + R_{ji} e_j$
9: end for
10: $e_i = \hat{\phi}_i - e_{imp}$
11: end if
12: end if
13: $R_{ii} = \langle e_i, \hat{\phi}_i \rangle$
14: $d_{ii} = \langle e_i, e_i \rangle$
15: $Y_{ii} = \langle e_i, y \rangle$
16: $B_{ii} = \frac{1}{d_{ii}}$
17: for $j \leftarrow (i-1)$ to 1 do
18: $V = R_{ji}$ row of $B^{(i-1)}$
19: $B_{ji} = -B_{ii} (V, R_i)$
20: end for
21: $s_{ji} = B_{ii}, Y_{ji}$
22: for $j \leftarrow (i-1)$ to 1
23: $s_{ji} = s_{ji}^{(i-1)} + B_{ji}, Y_{ji}$
24: end for

---

Steps 2–14 of the Algorithm 2 describe the QR decomposition. Actual QR factorization is not evaluated. In case of actual QR decomposition, Q has orthonormal columns and these columns are multiplied by $\hat{\phi}$ to generate R. But here, columns of Q are kept as orthogonal. The normalization of the columns of Q is required neither in computation of pseudoinverse, nor in estimation of $\hat{s}$, but for convenience the terms Q and R are retained.

### 3.2.2 Implementation

The above-mentioned QR decomposition is performed in six steps which are evaluated sequentially in each iteration. These steps are:

1. Compute $\langle e_i, \hat{\phi}_i \rangle$ which is $R_{ii}$
2. Perform division by $d_{ii}$
3. Multiply with $e_i$ and accumulate. Subtract the accumulated result from $\hat{\phi}_i$
4. Compute $\langle e_i, \hat{\phi}_i \rangle$ which is $R_{ii}$
5. Compute $\langle e_i, e_i \rangle$ and save it.
6. Compute $\langle e_i, y \rangle$ and save it.

Step by step evaluation of the QR factorization is shown in Figure 5. In iteration 1, $e_1 = \hat{\phi}_1$, so evaluation of first 3 steps is skipped. The RB $r_{mem}$ captures $\hat{\phi}_1$ through the MAC unit in iteration 1 to evaluate other functions. Step 1 is an inner product operation between $\hat{\phi}_1$ and previously saved $e_1$. This step produces the $i^{th}$ column of matrix R and saved in $R_{mem}$. In step 2, the divider divides the resulting inner product by previously saved $d_{ii}$. A temporary storage RAM2 is used to store the outputs of the divider. Depending upon the iteration count is lesser or greater than the overall VMU-divider path latency, divider outputs from RAM2 (op) are read instantly or after some clock periods, respectively. In step 3, first a scalar-vector multiplication is performed and then the result is subtracted from $\hat{\phi}_1$. The MAC unit performs the operation $z = (a \times b) + z$ when the input fu is “10”. Previously saved $e_k$ are again read from $q_{mem}$ and multiplied with the resulting value (op). The resulting vector $e_i$ is saved in $q_{mem}$. The RB $r_{mem}$ also holds the vector $e_i$ to evaluate steps 4, 5 and 6.

Steps 4, 5 and 6 compute inner products and are evaluated in pipeline, one after another. $\langle e_i, \hat{\phi}_i \rangle$ is equal to $R_{ii}$ which is saved in $R_{mem}$ for matrix inversion. The result of the step 5 ($d_{ii}$), is
The evaluation of $Y_i = \frac{1}{h_{e_1}y_i; h_{e_2}y_i; \ldots; h_{e_i}y_i^T}$, which is the step 6, is carried out along with the evaluation of steps 4 and 5 of QR decomposition. The measurement vector $y_i$ is read from $q_{mem}(u)$ and is multiplied with $e_i$ stored in $r_{mem}$. The resulting value is stored in $Y_{reg}$ which is a controlled register.

### Table 1: Timing complexity of AS step for different versions of OMP

| Design                | Timing complexity |
|-----------------------|-------------------|
| Classic OMP [14]      | $Nm$, $N$ columns are searched at each iteration |
| Improved OMP [9]      | $N(m/2 + 1)$, parallel selection of two columns |
| Proposed              | $T_{AS} = N \times q_i$, for mod$(i, w) = 0$ |
|                       | $W \times (I - q_i)$, otherwise |

Total timing complexity $T_{QR}$ to evaluate QR factorization for $K = 256$, $N = 1024$ is:

$$T_{QR} = 46I + (I - 2)(I - 3)/2 + (I - 1) + \sum_{i=1}^{I-27} i$$  \hspace{1cm} (6)

where the last term is for iterations greater than 27 which is the total latency for VMU-divider path.

### 3.3 Matrix inversion, computation of $\hat{s}$ and residual

Steps 16–20 of Algorithm 2 are for matrix inversion. The matrix $B$ is evaluated in the same way as depicted in [25]. In each iteration, $R_{mem}$ stores an evaluated column of $R$ and $B_{mem}$ stores the elements of matrix $B$. Rows of matrix $B$ are accessed in parallel and $R_{mem}$ is a RB. So, column of matrix $R$ from $R_{mem}$ and rows of matrix $B$ from $B_{mem}$ are fed to the VMU unit. Output from the VMU block is taken from the $ip_{mem}$ signal. The dedicated divider is used to compute the reciprocal of $R_{ij}$ in parallel. Extra registers are inserted to match the latencies of both the operations. The total time
complexity for computation of inverse is $I(I-1)/2$ where $I$ denotes the total number of iterations.

The signal estimation $\hat{s}$ can be computed by steps 21–24 of the Algorithm 2. In this estimation procedure, previously computed values $\hat{s}^{(i-1)}$ are used to compute the current estimation $\hat{s}^{(i)}$. It is not required to store all the components of vector $Y$. Only $\langle e_i, y \rangle$, evaluated at each iteration is saved in a single-controlled register $Y_{\text{reg}}$. The estimate $\hat{s}$ is stored in $\hat{s}_{\text{mem}}$ memory block. The proposed scheme for computation of signal estimation is shown in Figure 6. Time complexity for this function is $I(I+1)/2$.

To compute the residual, $\hat{\phi}$ is first multiplied with $\hat{s}$. Columns of $\hat{\phi}$ are accessed from $\phi_{\text{mem}}$ and memory RAM1 provides $\phi_{\text{mem}}$, the indices of those columns. $\hat{s}$ is accessed from $\hat{s}_{\text{mem}}$ serially. This scalar–vector multiplication is performed by the same MAC unit. After the multiplication and accumulation, operation $y$ is selected through the MUX M1 for subtraction. The product $\hat{\phi}\hat{s}$ is subtracted from $y$ by the MAC unit. Total timing complexity for residual computation is $I(I+1)/2$ for multiplication and accumulation and extra $I$ clock cycles to perform the subtraction.

The timing complexity ($T_{\text{add}}$) for matrix inversion, estimation of $\hat{s}$ and computation of residual for $K = 256$, $N = 1024$ can be expressed as:

$$T_{\text{add}} = 20I + I(I-1)/2 + \sum_{i=1}^{I-16} i$$ (7)

where the last term is for iterations greater than 16 which is the overall latency for the path from divider to signal estimation block.

### 3.4 Implementation of stopping criteria

Evaluation of the stopping criteria is avoided in the existing implementations. Determination of the stopping criteria can be easily achieved making the design invariant to the knowledge of sparsity. Determination of the stopping criteria doesn’t need any extra hardware except a comparator. In each iteration, $(r', r')$ is computed in pipeline with the function $(\phi_p, r')$. The inner product $(y, y)$ is saved in a register. The divider is shared to find the ratio in parallel with the computation of $(\phi_p, r')$.

The value of the threshold $\epsilon$ can be tuned close to zero for best results. A simplest choice of the parameter $\epsilon$ is of the order of $2^{-10}$. Stopping criteria for $(i-1)^{th}$ iteration are checked in parallel to the evaluation of the inner product function $(\phi_p, r')$ for $i^{th}$ iteration. Total time to determine the stopping criteria is $l_{ip} + l_{dv} + 1$, where $l_{ip}$ and $l_{dv}$ are latency for inner product and division, respectively. Moreover, only $I$ clock cycle is wasted to check the stopping criteria.

#### 4 PERFORMANCE OF THE LPF-OMP ALGORITHM AND ITS IMPLEMENTATION

##### 4.1 Reconstruction efficiency

Software analysis of the proposed LPF-OMP algorithm is carried out for random input signals which take values from the set $\{-1, 1\}$ for different sparsity values. The random Gaussian distribution-based sampling matrix is considered. A comparison of the performance of different versions of OMP is shown in Figure 7. Performance depends on the number of iterations allowed for a certain signal sparsity. If the algorithm runs for a fixed $m$ number of iterations, performance degrades for low sparse signals. Extra iterations improve the performance, however, it adds extra hardware. The proposed LPF-OMP incorporates a stopping criteria to improve the performance allowing maximum of $K/4$ iterations. Probability of success for the proposed LPF-OMP is better than the OMP with fixed iterations and also better than the improved OMP [9]. Simulation results are noted for different sparsity values and for each signal sparsity 100 simulations are carried out to calculate probability of success. Probability of success is measured in terms of the performance metric relative or normalized root mean square error (RMSE) which is expressed as:

$$RMSE = \frac{||\hat{x} - x||_2}{||x||_2}$$ (8)

If the RMSE value is of the order of $e^{-15}$ or lesser, then it is considered as a success.

OMP algorithm requires at least $O(m \log N)$ [6] number of measurement samples to recover a $m$-sparse signal. So, for a certain value $K$, the proposed LPF-OMP is capable of reconstructing signals having sparsity close to the maximum theoretical value. Table 2 shows a caparison of performances of different versions of OMP for a random signal with parameters $K = 256$, $N = 1024$ and $m = 36$. As the value of $w$ is increased, reconstruction time reduces significantly. But when the value of $w$ is 32, the reconstruction time is higher.
This is because, minimum searching time in each iteration is now 64 clock cycles which dominate. In the SB, higher value of $w$ increases the number of required comparators. The value of $w$ can be chosen based on two considerations: reconstruction speed requirement and number of permissible comparators.

The choice of $w$ and $W$ is crucial for optimum performance of the LPF-OMP algorithm. Several simulations have been carried out to qualitatively find the optimum value of $W$ as shown in Table 3. Here, each test set consists of 100 random input signals which are same as for Figure 7. Performance of classic OMP and LPF-OMP is compared for each test set. Performance is measured using probability of success as it is done for Figure 7. Parameters are chosen as $N = 1024$, $K = 256$ and simulations are stopped after $m = 36$ number of iterations. It can be seen from Table 3 that when $W \leq 2w$ the reconstruction performance is very poor. Reconstruction performance of LPF-OMP improves when $W \geq 2w$ and extra iterations are added as shown in Figure 7.

### 4.2 Recovery signal-to-noise ratio

The design metric recovery signal-to-noise ratio (RSNR) [25] is preferred here to estimate the design performance. RSNR is measured as:

$$RSNR = 20\log_{10}\left(\frac{1}{RMSE}\right)$$  \hspace{1cm} (9)

Various input signals are taken as input to the proposed design for parameters, $K = 256$, $N = 1024$ and $m = 36$. The algorithm is halted after $I = 36$ iterations to compare the design performance with other existing designs. Fixed point data width of the design is varied in XILINX platform and RSNR is measured for each data width. In order to consume minimum hardware resources and for acceptable RSNR requirement, 18-bit data width is used for the proposed design. RSNR performance is similar to the RSNR performance reported in [25]. Fractional part is represented by either 10 bits or by 12 bits. RSNR of 16.498 dB and 18.336 dB are achieved for 10 bits and 12 bits, respectively. The proposed design is targeted to the XILINX Vertex6 FPGA device for $K = 256$, $N = 1024$, $w = 8$ and supports maximum $I_{max} = K/4$ iterations.

### 4.3 Resource utilization

Hardware resources are shared and reused to reduce the digital overhead. The proposed architecture uses inbuilt DSP blocks to realize different arithmetic functions. Inbuilt block RAMs (BRAMs) are configured to be used as RAM or ROM. Memory utilization of the proposed design is depicted in Table 4. A parallel CORDIC [27] based divider with 10 pipeline stages is used and is shared in different steps. The resources used in this design are independent of the sparsity number but depend on the value of $I_{max}$ and $K$. The choice of $w$ is as crucial as count of comparators depends on the value of $w$.  

---

**TABLE 2** Comparison of the different versions of OMP

| OMP versions   | RMSE         | Size(Λ) | Recovery time (μs) |
|----------------|--------------|---------|--------------------|
| LPF-OMP        | $w = 2$      | 1.84e-15 | 36                 | 176.08             |
|                | $w = 4$      | 1.84e-15 | 36                 | 107.09             |
|                | $w = 8$      | 1.84e-15 | 36                 | 76.43              |
|                | $w = 16$     | 1.93e-15 | 38                 | 65.11              |
|                | $w = 32$     | 1.23e-15 | 39                 | 65.84              |
| Improved OMP [9] | 1.79e-15   | 38             | 170                 |
| Classic OMP*   | 8.925e-16   | 36             | 314.06              |

*Reconstruction time for classic OMP is estimated for the proposed design.

**TABLE 3** Validation of parameter $w$ based on probability of successful reconstruction

| Test sets | Classic OMP | LPF-OMP $w = 2$ | LPF-OMP $w = 4$ | LPF-OMP $w = 8$ |
|-----------|-------------|-----------------|-----------------|-----------------|
| 1         | 0.41        | 2               | 0.36            | NC              |
| 2         | 0.44        | 4               | 0.46            | 0.28            | NC              |
| 3         | 0.42        | 6               | 0.46            | 0.33            | NC              |
| 4         | 0.46        | 8               | 0.42            | 0.40            | 0.07            |
| 5         | 0.40        | 10              | 0.40            | 0.38            | 0.17            |
| 6         | 0.41        | 12              | 0.41            | 0.39            | 0.24            |
| 7         | 0.38        | 14              | 0.38            | 0.38            | 0.28            |
| 8         | 0.39        | 16              | 0.37            | 0.39            | 0.36            |

NC: Signal reconstruction failed.
The proposed design is compared with the other existing designs in Table 5. In comparison to the Cholesky-based architecture in [14], the proposed architecture uses less number of DSP and RAM blocks when targeted to the same FPGA device. The architecture depicted in [9] has implemented improved OMP considering complex sampling matrix. The proposed architecture considers real sampling matrix and very much hardware efficient than the architecture mentioned in [9]. Resource utilization of the Cholesky-based designs reported in [9,14] depends heavily on signal sparsity. The proposed design is hardware efficient although it supports maximum sparsity level of \( K/4 \) and resource utilization is less variant to the sparsity level. The MIB-based design reported in [23] uses very less DSP blocks and memory elements but its slice occupancy is quite high. Also, for \( N = 1024 \) the architecture depicted in [23] has very high reconstruction time. The proposed study has similar resource utilization compared to our previous study [25]. But the proposed architecture has less memory utilization as it avoids pseudoinverse computation. Also, the matrix multiplication unit is simplified to reduce power consumption and to achieve better reconstruction speed. If \( w = 2 \), then the proposed variation is similar to the improved OMP reported in [9] and the proposed algorithm is similar to classic OMP for \( w = 1 \).

### 4.4 | Reconstruction speed

The classic OMP algorithm is reformulated to achieve high speed of reconstruction. The total time complexity of the OMP algorithm is reduced in two ways. The first obvious

| Memory elements | Word per cycle | BRAM | Reg |
|-----------------|----------------|------|-----|
| \( y_{mem} \)   | 1              | 1    | \( n_{dxN} \) = 0 |
| \( \phi_{mem} \) | \( K \)        | \( K \) | \( NK \) = 0 |
| \( q_{mem} \)   | \( K \)        | \( K \) | \( IK \) = 0 |
| \( \tilde{s}_{mem} \) | 1              | 1    | \( I \) = 0 |
| \( ip_{mem} \)  | 1              | 1    | \( I \) = 0 |
| \( \text{RAM1}^*/\text{RAM2} \) | 1              | 1    | \( I \) = 0 |
| \( \text{RAM3}^* \) | 1              | 1    | \( W \) = 0 |
| \( B_{mem} \)   | 1              | \( I \) | \( I(I+1)/2 \) = 0 |
| \( M_{mem} \)   | 1              | \( I \) | \( 0 \) |
| \( r_{mem} \)   | \( K \)        | \( K \) | \( K \) |
| \( Y_{reg} \)   | 1              | 1    | \( 0 \) |

*Data width for \( \text{RAM1} \) and \( \text{RAM3} \) is 10 bit.

| Design          | [14]   | [23]   | [9]    | [25]   | [26]   | Proposed |
|-----------------|--------|--------|--------|--------|--------|----------|
| FPGA target     | xc6vsx240t | Kintex-7 | xc6vsx475t | xc6vsx240t | xc6vsx240t | xc6vsx240t |
| Algorithm       | OMP    | OMP    | Improved OMP | OMP    | OMP    | OMP      |
| Supported maximum sparsity (\( m \)) | 36     | Configurable | 36    | 36    | 36    | 64       |
| Signal size (\( N \)) | 1024   | 512    | 1024   | 1024   | 1024   | 1024     |
| Number of measurements (\( K \)) | 256    | Configurable | 256   | 256   | 256   | 256      |
| Maximum frequency (MHz) | 119.96 | 87     | 135.4  | 133.3  | 136.98 | 131.5    |
| Occupied slices | 6208(16\%) | \( \sim 92K \) | 7860(11\%) | 10902(28\%) | 10085(26\%) | 8525(22\%) |
| DSP core        | 589(76\%) | 93     | 1544(74\%) | 386(50\%) | 389(50\%) | 410(53\%) |
| RAMB            | 576(69\%) | 64 or 128 | 342(32\%) | 430(52\%) | 333(40\%) | 431(52\%) |
| Reconstruction time (\( \mu s \)) | 340    | 250    | 170    | 327    | 302.4  | 76.43    |
| Dynamic power (mW) | 3233   | \( N/A \) | 4370   | 2357   | 2066   | 1819     |

*Type of the RAM block is RAM36E1.

### Table 4 | Estimation of memory elements

| Memory elements | Write | Read | BRAM | Reg |
|-----------------|-------|------|------|-----|
| \( y_{mem} \)  | 1     | 1    | \( n_{dxN} \) | 0   |
| \( \phi_{mem} \) | \( K \) | \( K \) | \( NK \) | 0   |
| \( q_{mem} \)  | \( K \) | \( K \) | \( IK \) | 0   |
| \( \tilde{s}_{mem} \) | 1     | 1    | \( I \) | 0   |
| \( ip_{mem} \)  | 1     | 1    | \( I \) | 0   |
| \( \text{RAM1}^*/\text{RAM2} \) | 1     | 1    | \( I \) | 0   |
| \( \text{RAM3}^* \) | 1     | 1    | \( W \) | 0   |
| \( B_{mem} \)   | 1     | \( I \) | \( I(I+1)/2 \) | 0   |
| \( M_{mem} \)   | 1     | \( I \) | 0     | \( I \) |
| \( r_{mem} \)   | \( K \) | \( K \) | 0     | \( K \) |
| \( Y_{reg} \)   | 1     | 1    | 0     | 1   |

- Data width for RAM1 and RAM3 is 10 bit.
reduction is taken place in the correlation step because of the change in the number of columns to be searched in successive iterations. Secondly, the timing complexity involved in computation of pseudoinverse is saved in estimating \( \hat{s} \). Extra two clock cycles are required for the first iteration: one clock cycle to store \( y \) and another to store \( e_i \) in \( r_{\text{mem}} \) temporarily at the initialization step and in the QR stage, respectively. Also, to evaluate the stopping criteria total \( I \) number of clock cycles are wasted. Interdependency of the steps is reduced by inserting buffers where it is necessary so that evaluation of a step can be started before full evaluation of its previous step. An estimation of the total reconstruction time of the proposed design for \( K = 256, N = 1024, w = 8 \) is \( T_{\text{ov}} = T_{\text{AS}} + T_{\text{QR}} + T_{\text{asb}} + (I + 2) \). Hence total 10051 number of clock cycles are required for \( I = 36 \) number of iterations.

The proposed architecture is faster in comparison to other designs as shown in Table 5. Thus the proposed technique can be adopted for real-time measurement of high frequency RF signals such as RADAR pulses. The proposed architecture achieves better reconstruction speed even for classic OMP as shown in Table 2. In comparison to our previous architectures [25,26], though the hardware performance is same the reconstruction speed is higher.

### 4.5 Power consumption

Power consumption of a circuit can be reduced from algorithm to device levels of abstraction. Modification at the algorithmic level has the capacity to achieve drastic power reduction as compared to the lower levels of abstraction in IC design. Dynamic power consumption is reduced by means of both algorithmic and architectural improvements. The power consumption of the proposed design is considerably low in comparison to other reported designs [9,14,25]. Xilinx XPower analyzer is used to estimate the power consumption of the design implemented in FPGA.

A considerable amount of dynamic power is reduced by algorithmic reformulation of the OMP algorithm. In case of classical OMP, \( \phi_{\text{mem}} \) block is active for \( N \) clock cycles in each iteration to perform the AS step. Thus a major fraction of total dynamic power is consumed in accessing \( \phi_{\text{mem}} \). The access time of \( \phi_{\text{mem}} \) in AS step for the proposed LPF-OMP is the same as the number of iterations is not same. In some of the iterations, \( \phi_{\text{mem}} \) is accessed for a shorter period of time and thus the dynamic power is reduced.

The first architectural modification to reduce power is sharing of resources. For example, a single VMU is used to perform several operations. The proposed architecture uses less hardware resources than the Cholesky-based implementations reported in [9,14]. Avoiding the pseudoinverse computation, a simpler VMU is proposed to reduce power consumption and to improve timing in comparison to [25]. Secondly, all the BRAMs including \( \phi_{\text{mem}} \) are selectively accessed. During evaluation of a step, BRAMs are selectively accessed when it is necessary. In Table 5, power consumption of the proposed design for \( m = 64 \) and \( m = 36 \) is almost same. This is because both the architectures have similar resource utilization.

### 4.6 A practical case for implementation

The reconstruction of high frequency RADAR pulses is studied here. The RADAR signals are sparse when represented with Gabor time-frequency dictionary. A general framework for CS-based capturing of unknown RADAR pulses is given in [25] where different types of Gaussian pulses are taken as test pulses. The OMP algorithm must incorporate a stopping criteria due to unavailability of information about sparsity. Signal strength and variety of RADAR pulses may not be fixed in such cases. Thus the stopping criteria should be robust and must be able to recover an unknown pulse in minimum number of iterations. Reconstruction of two types of RADAR test pulses, viz., a trapezoidal pulse and a Gaussian fifth derivative pulse are compared. The same value of threshold parameter \( \epsilon \) is used for both the cases. Figure 8 shows a comparison of reconstructed output of the FPGA implementation with the original signals for two different pulses with same halting threshold.

### 5 CONCLUSION

A novel LPF-OMP algorithm is proposed here to achieve low power and high reconstruction speed for measurement of high frequency RF pulses such as RADAR pulses. Both the performances are improved by reducing the number of columns to be searched in the AS step of successive iterations. Pseudoinverse computation is avoided for further acceleration in reconstruction time. A novel architecture is proposed to implement the LPF-OMP algorithm. The dynamic power
consumption of the proposed design is considerably low due to the algorithmic and architectural improvements. The proposed design proves to be faster than any other existing designs. Area is also reduced by reusing of the same VMU for every operation. The proposed design consumes less memory and DSP blocks for the same design parameters. A novel stopping criterion is proposed which is robust and does not consume any extra hardware. The proposed design is efficient for signals having low sparsity level as resource utilization varies marginally with the sparsity level.

ACKNOWLEDGMENTS
There is no funding agency involved in this manuscript. It is a result of research work that was carried out at the Department of electronics and communication engineering in NIT Rourkela.

ORCID
Shirshendu Roy  https://orcid.org/0000-0001-5274-1036

REFERENCES
1. Eldar, Y.C., Kuttnerok, G.: Compressed Sensing: Theory and Applications. Cambridge University Press (2012).
2. Candès, E.J., Wakin, M.B.: An introduction to compressive sampling. IEEE Signal Process. Mag. 25(2), 21–30 (2008).
3. Zhao, Y., Hu, Y.H., Liu, J.: Random triggering-based sub-nyquist sampling system for sparse multiband signal. IEEE Trans. Instrum. Meas. 66(7), 1789–1797 (2017).
4. Daponte, P., et al.: Analog-to-information converters in the wideband RF measurement for aerospace applications: current situation and perspectives. IEEE Instrum. Meas. Mag. 20(1), 20–28 (2017).
5. Pati, Y.C., Rezaifar, R., Krishnaprasad, P.S.: Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition. In: 1993 Conference Record of the Twenty-Seventh Asilomar Conference on Signals, Systems and Computers, Pacific Grove, pp. 40–44. IEEE (1993).
6. Tropp, J.A., Gilbert, A.C.: Signal recovery from random measurements via orthogonal matching pursuit. IEEE Trans. Inf. Theory. 53(12), 4655–4666 (2007).
7. Jost, P., Vanderghynst, P., Frossard, P.: Tree-based pursuit: Algorithm and properties. IEEE Trans. Signal Process. 54(12), 4685–4697 (2006).
8. Cheng, Y.C., Tsai, P.-Y., Huang, M.-H.: Matrix-inversion-free compressed sensing with variable orthogonal multi-matching pursuit based on prior information for ECG signals. IEEE Trans. biomed. Circ. Syst. 10(4), 864–873 (2016).
9. Liu, S., Lyu, N., Wang, H.: The implementation of the improved OMP for AIC reconstruction based on parallel index selection. IEEE Trans. Very Large Scale Integr. VLSI Syst. 26(2), 319–328 (2018).
10. Septimius, A., Steinberg, R.: Compressive sampling hardware reconstruction. In: Proceedings of 2010 IEEE international Symposium on Circuits and Systems (ISCAS), Paris, pp. 3316–3319. IEEE (2010).
11. Blache, P., Rabah, H., Amira, A.: High level prototyping and fpga implementation of the orthogonal matching pursuit algorithm. In: 2012 11th International Conference on Information Science, Signal Processing and Their Applications (ISSPA), 1336–1340. IEEE (2012).
12. Shi, W., et al.: An extensible and real-time compressive sensing reconstruction hardware for wbans using omp. In: 2013 IEEE 10th International Conference on ASIC, 1–4 (2013).
13. Ren, F., et al.: A single-precision compressive sensing signal reconstruction engine on FPGAs. In: 2013 23rd International Conference on Field Programmable Logic and Applications (FPL), 1–4. IEEE (2013).
14. Rabah, H., et al.: FPGA implementation of orthogonal matching pursuit for compressive sensing reconstruction. IEEE Trans Very Large Scale Integr VLSI Syst. 23(10), 2209–2220 (Oct 2015).
15. Ren, F., Marković, D.: A configurable 12237 ks/s 12.8 mw sparse approximation engine for mobile data aggregation of compressively sampled physiological signals. IEEE J. of Solid-State Circ. 51(1), 68–78 (2016).
16. Golub, G.H., Van Loan, C.F.: Matrix Computations', 3. JHU Press (2012).
17. Kulkarni, A., Mohsenin, T.: Low overhead architectures for OMP compressive sensing reconstruction algorithm. IEEE Trans. Circ. Syst. I Regul. Pap. 64(6), 1468–1480 (2017).
18. Stanislaus, J.L.V.M., Mohsenin, T.: Low-complexity FPGA implementation of compressive sensing reconstruction. In: 2013 International conference on computing, networking and communications (ICNC), pp. 671–675 (2013).
19. Stanislaus, J.L.V.M., Mohsenin, T.: High performance compressive sensing reconstruction hardware with QRD process. In: 2012 IEEE International Symposium on Circuits and Systems, 29–32 (2012).
20. Yu, Z., et al.: Fast compressive sensing reconstruction algorithm on FPGA using orthogonal matching pursuit. In: 2016 IEEE International Symposium on Circuits and Systems (ISCAS), 249–252. IEEE (2016).
21. Bai, L., et al.: High-speed compressed sensing reconstruction on FPGA using OMP and AMP. In: 2012 19th IEEE International Conference on Electronics, Circuits, and Systems (ICECS 2012), 53–56 (2012).
22. Knoopy, B., et al.: Rapid digital architecture design of orthogonal matching pursuit. In: 2016 24th European Signal Processing Conference (EUSIPCO), 1857–1861 (2016).
23. Huang, G., Wang, L.: An FPGA-based architecture for high-speed compressed signal reconstruction. ACM Trans. Embed. Comput. Syst. 16(3), 89:1–89:23 (2017).
24. Jiang, J., Huang, Y.: A high-SNR projection-based atom selection OMP processor for compressive sensing. IEEE Trans. Very Large Scale Integr. VLSI Syst. 24(12), 3477–3488 (2016).
25. Roy, S., Acharya, D.P., Sahoo, A.K.: Low-complexity architecture of orthogonal matching pursuit based on QR decomposition. IEEE Trans. Very Large Scale Integr. VLSI Syst. 27(7), 1623–1632 (2019).
26. Roy, S., Acharya, D.P., Sahoo, A.K.: Incremental Gaussian elimination approach to implement OMP for sparse signal measurement. IEEE Trans. Instrum. Meas. 69(7), 4067–4075 (2020). https://doi.org/10.1109/TIM.2019.2947118.
27. Volder, J.E.: The chord trigonometric computing technique. IEEE Transactions on Electronic Computers. EC-8(3), 330–334 (1959).

How to cite this article: Roy S, Acharya DP, Sahoo AK. Fast OMP algorithm and its FPGA implementation for compressed sensing-based sparse signal acquisition systems. IET Circuits Devices Syst. 2021;1–11. https://doi.org/10.1049/cds2.12047