Received: 13 February 2019 | Revised: 12 July 2019 | Accepted: 26 September 2019

DOI: 10.4218/etrij.2019-0067

ORIGINAL ARTICLE

ETRI Journal WILEY

High-throughput and low-area implementation of orthogonal matching pursuit algorithm for compressive sensing reconstruction

Vu Quan Nguyen[1] | **Woo Hyun Son**[2] | **Marek Parfieniuk**[3] | **Luong Tran Nhat Trung**[4] | **Sang Yoon Park**[2]

[1]Radio Management Solutions Division, OnPoom Co., Ltd., Seoul, Rep. of Korea

[2]Department of Electronic Engineering, Myongji University, Yongin, Rep. of Korea

[3]Institute of Informatics, University of Bialystok, Bialystok, Poland

[4]Engineering Department, Esilicon, Da Nang, Vietnam

Correspondence
Sang Yoon Park, Department of Electronic Engineering, Myongji University, Yongin, Rep. of Korea.
Email: sypark@mju.ac.kr

Funding information
This work was supported by Myongji University 'Assistant Professor Support Program'.

Massive computation of the reconstruction algorithm for compressive sensing (CS) has been a major concern for its real-time application. In this paper, we propose a novel high-speed architecture for the orthogonal matching pursuit (OMP) algorithm, which is the most frequently used to reconstruct compressively sensed signals. The proposed design offers a very high throughput and includes an innovative pipeline architecture and scheduling algorithm. Least-squares problem solving, which requires a huge amount of computations in the OMP, is implemented by using systolic arrays with four new processing elements. In addition, a distributed-arithmetic-based circuit for matrix multiplication is proposed to counterbalance the area overhead caused by the multi-stage pipelining. The results of logic synthesis show that the proposed design reconstructs signals nearly 19 times faster while occupying an only 1.06 times larger area than the existing designs for $N = 256$, $M = 64$, and $m = 16$, where N is the number of the original samples, M is the length of the measurement vector, and m is the sparsity level of the signal.

KEYWORDS

compressive sensing (CS), distributed arithmetic (DA), orthogonal matching pursuit (OMP), pipelined structure, signal reconstruction

1 | INTRODUCTION

During the past few years, compressive sensing (CS) has become appealing to researchers in various fields because of its wide applicability as well as its superiority to conventional signal processing techniques. The fundamental premise of CS theory is that signals can often be represented using only few non-zero coefficients in an appropriate basis. If a signal is sparse, then it can be sampled at a rate lower than the Nyquist rate, that is, lower than twice the highest frequency contents, and can still be accurately recovered [1,2]. Therefore, using CS reduces the resources needed for data acquisition and transmission, thereby dramatically reducing the required time and energy.

Among other advantages, CS allows reconstructing images from incomplete representations, supposing that an image is sparse in a consistent basis. For instance, a single-pixel Terahertz camera can exploit CS to reduce the effort required to acquire an image, because not all the pixels need to be sampled, thereby increasing the acquisition speed [3]. CS also allows improving the image quality by reducing the

number of measurements. In medical imaging, CS has been applied to remove motion artifacts, producing better results for dynamic magnetic resonance imaging [4].

Many software designs have been proposed for reconstructing compressively sensed signals. It is well known that the assistance of a graphics processing unit (GPU) can significantly improve the performance of a software implementation [5]. However, GPU-accelerated software-based solutions that run on general-purpose CPUs cannot reach high throughput, and they consume considerable power, which makes them unsuitable for real-time energy-efficient embedded systems.

Several hardware-level methods have been implemented to increase the speed of CS signal reconstruction. Septimus and Steinberg implemented the orthogonal matching pursuit (OMP), which is the most frequently used reconstruction algorithm, in a field-programmable gate array (FPGA). The speed of the algorithm was limited by a low operational clock frequency [6]. Mohsenin and others proposed an FPGA-based OMP architecture to reduce complexity by employing the QR-decomposition to solve a matching pursuit (MP) task [7]. Shi and others presented an hardware architecture based on the Cholesky factorization and a multifunctional systolic array [8]. However, the limited throughput and the demanded high sparsity make these designs impractical for real-time applications [9,10].

In the existing hardware designs for the MP and OMP algorithms, the least-squares problem (LSP) solver is the most time-consuming element. The LSP is solved only in the last stage of the MP algorithm, whereas in the OMP algorithm it has to be solved in every iteration. The LSP solver involves heavy matrix computations, especially matrix inversion, which dramatically influence the time performance. Furthermore, if the series of matrix computations is performed by a single hardware unit, then the area complexity can be reduced, but the overall throughput of the OMP processing is limited.

In this paper, we suggest a fine-grain pipelined architecture for the OMP to overcome the low-speed performance of conventional designs. LSP solving is implemented by using systolic arrays with four newly designed processing elements. We also propose two methods to implement the matrix multiplication of the LSP solver: (a) based on the sum-of-product principle (SoP), and (b) based on distributed-arithmetic (DA) matrix multiplication. The SoP-based design achieves an increased throughput by slightly extending the area, whereas the multiplierless DA-based design contributes to minimizing the area overhead related to the pipelining.

The remainder of this paper is organized as follows. Section 2 briefly reviews the backgrounds of CS and the OMP algorithm. Section 3 presents the proposed architectures, including the fine-grain pipelined design, the new systolic array structures, and the SoP- and DA-based matrix multiplication units. The results of logic synthesis are presented in Section 4, together with a comparison of the proposed and existing designs. Finally, Section 5 concludes the paper.

2 | BACKGROUND REVIEW

CS is a new signal processing technique for efficiently sampling and recovering signals. It allows a signal to be fully recovered after being sampled at a sub-Nyquist rate, if it is sparse in a certain domain or dictionary. For a signal to be sparse, its energy density must be mostly distributed over only a few points called sparseness support coefficients. The total number of these non-zero coefficients defines the level of signal sparsity. An additional condition for the reconstruction of the compressively sensed signal to be possible is the incoherence between the measurement matrix and the representation matrix. A lower coherence level means the signal recovery requires fewer samples.

In CS, the fundamental operation is the multiplication of the input signal by a measurement matrix. The precognition of the measurement matrix is the key for reconstructing the original signal from the compressively sensed signal. Assuming that a signal f has the sparse representation x in a basis $\mathbf{\Phi}$, it can be described by

$$f = \mathbf{\Phi} x, \tag{1}$$

where $f \in \mathbb{R}^N$, $\mathbf{\Phi} \in \mathbb{R}^{N \times N}$, and $x \in \mathbb{R}^N$. As most of the energy is concentrated in few coefficients, the remaining coefficients become zero without distorting the signal; consequently, x is mostly composed of zeros. The acquisition of the signal vector f is processed via a linear transformation of the measurement matrix $\mathbf{\Psi}$, resulting in the measurement vector y:

$$y = \mathbf{\Psi} f = \mathbf{\Psi}\mathbf{\Phi} x = \mathbf{\Theta} x, \tag{2}$$

where $y \in \mathbb{R}^M$, $\mathbf{\Psi} \in \mathbb{R}^{M \times N}$, $\mathbf{\Theta} \in \mathbb{R}^{M \times N}$, and $\mathbf{\Theta}$ is the reconstruction matrix. Because $N > M$ in order to reduce the number of samples, (2) becomes an underdetermined system of linear equations. One of the solutions to this problem is to minimize the L^p-norm as

$$x = \operatorname{argmin}_{x:\mathbf{\Theta}x=y} \|x\|_p. \tag{3}$$

The solution of (2) by L^2 minimization can be expressed as

$$x = \operatorname{argmin}_{x:\mathbf{\Theta}x=y} \|x\|_2. \tag{4}$$

The problem represented by (4) is known as the basis pursuit problem and can be solved by several approaches, such as convex relaxation, iterative thresholding, or greedy iterative methods. Greedy methods, such as MP and OMP, are preferable for hardware implementation because of their simplicity, high reconstruction speed, and low implementation cost.

The OMP is a derivative of the original MP algorithm, which involves an LSP-solving step to compute the residual. The main difference between the two is that in the OMP all the coefficients are updated by orthogonally projecting them onto a set of selected atoms, improving the result of the algorithm. However, the LSP step needs to be solved in every iteration of the OMP and involves complicated matrix arithmetic. Thus, the computational complexity of OMP is much higher than that of the MP. Given that the complexity of each iteration increases, but the reconstruction output is enhanced, the OMP algorithm is more consistent than the MP with low-sparsity signals.

The OMP algorithm is summarized in Algorithm 1. The reconstruction matrix $\mathbf{\Theta}$ and measurement vector y are the inputs, whereas the sparse reconstruction x constitutes the output. The four steps of the OMP algorithm are performed iteratively. The number of iterations depends on m, that is, the sparsity level of the signal. In each iteration, the column of $\mathbf{\Theta}$ having maximum correlation with the residual y is selected. Then, its contribution is removed to create a new residual and a newly reconstructed value. The procedure from Steps 1 to 4 in Algorithm 1 is repeated until the number of iterations reaches the sparsity level m. Thus, all the non-zero values are reconstructed.

ALGORITHM 1 Describes the orthogonal matching pursuit (OMP) algorithm

INPUT: reconstruction matrix $\mathbf{\Theta}$, measurement vector y, sparsity m

OUTPUT: sparse reconstruction x

PROCEDURE:

1. Find the measurement column $\lambda_i = \text{argmax}|\langle r_{i-1}, \mathbf{\Theta}\rangle|$, which is maximally correlated with the residual y.
2. Update the chosen column set $\mathbf{\Theta}_i = [\mathbf{\Theta}_{i-1} \quad \theta_{\lambda_i}]$ with initial $\mathbf{\Theta}_0 = \emptyset$.
3. Solve the least-squares problem $x_i = \text{argmin}\|y - \mathbf{\Theta}_i x\|_2$ to find the values of non-zero elements of x.
4. Compute the residual $r_i = y - \mathbf{\Theta}_i x_i$ to remove the contribution of θ_{λ_i}.

Repeat Steps 1 to 4 until $i = m$.

The LS problem in Step 3 can be solved by

$$x = \left(\mathbf{\Theta}^{\mathrm{T}}\mathbf{\Theta}\right)^{-1}\mathbf{\Theta}^{\mathrm{T}}y = \mathbf{C}^{-1}\mathbf{\Theta}^{\mathrm{T}}y. \tag{5}$$

In our study, a modified Cholesky factorization method, known as the LDL decomposition, has been employed to find the inverse matrix $\mathbf{C}^{-1}$ in (5). The original matrix $\mathbf{C}$ can be computed as the product of three matrices:

$$\mathbf{C} = \mathbf{L}\mathbf{D}\mathbf{L}^{\mathrm{T}}. \tag{6}$$

The elements of the triangular matrix $\mathbf{L}$ and diagonal matrix $\mathbf{D}$ can be obtained from

$$d_{ii} = c_{ii} - \sum_{k=1}^{i-1} l_{ik}^2 d_{kk} \tag{7}$$

and

$$l_{ij} = \frac{1}{d_{jj}}\left(c_{ij} - \sum_{k=1}^{j-1} l_{ik} l_{jk} d_{kk}\right), \tag{8}$$

respectively. Then, the inverse matrix $\mathbf{C}^{-1}$ is computed as

$$\mathbf{C}^{-1} = \left(\mathbf{L}^{-1}\right)^{\mathrm{T}}\mathbf{D}^{-1}\mathbf{L}^{-1}. \tag{9}$$

Because $\mathbf{D}$ is a diagonal matrix, its inverse can be obtained by simply inverting every element on the diagonal. In contrast, the elements of matrix $\mathbf{A}$, the inverse of the triangular matrix $\mathbf{L}$ ($\mathbf{A} = \mathbf{L}^{-1}$), can be obtained by using the relation

$$a_{ij} = -\sum_{k=j}^{i-1} l_{ik} a_{kj}. \tag{10}$$

Note that all the elements on the diagonal become one after the inversion, that is, $a_{ii} = 1$.

3 | PROPOSED ARCHITECTURE

3.1 | Proposed fine-grain pipelined structure

Figure 1 shows the proposed hardware architecture for the OMP algorithm. It consists of three main computing units: the atom selection unit, the LSP solver, and the residual calculation unit. In the atom selection unit, the correlation between the reconstruction matrix $\mathbf{\Theta}$ and the newly updated residual r is computed. Then, the largest dot product is found by a tree comparator. The LSP solver performs four consecutive matrix computations: multiplication, decomposition, inversion, and composition, the details of which are summarized as follows:

- *Matrix multiplication*: Matrix $\mathbf{\Theta}_i$ needs to be multiplied by its transpose, $\mathbf{\Theta}_i^{\mathrm{T}}$ to compute $\mathbf{C}$, where $\mathbf{\Theta}_i \in \mathbb{R}^{M \times i}$ and $\mathbf{C} \in \mathbb{R}^{i \times i}$. Because $\mathbf{C}$ is a symmetric matrix, only its lower-left half needs to be computed. This function is performed by a multiplierless matrix computation unit, either DA- or SoP-based, as discussed in Sections 3.2 and 3.3.
- *Matrix decomposition*: In Figure 1, the matrix decomposition unit performs the LDL decomposition. It uses two specialized processing elements, namely PE_{1a} and PE_{1b},

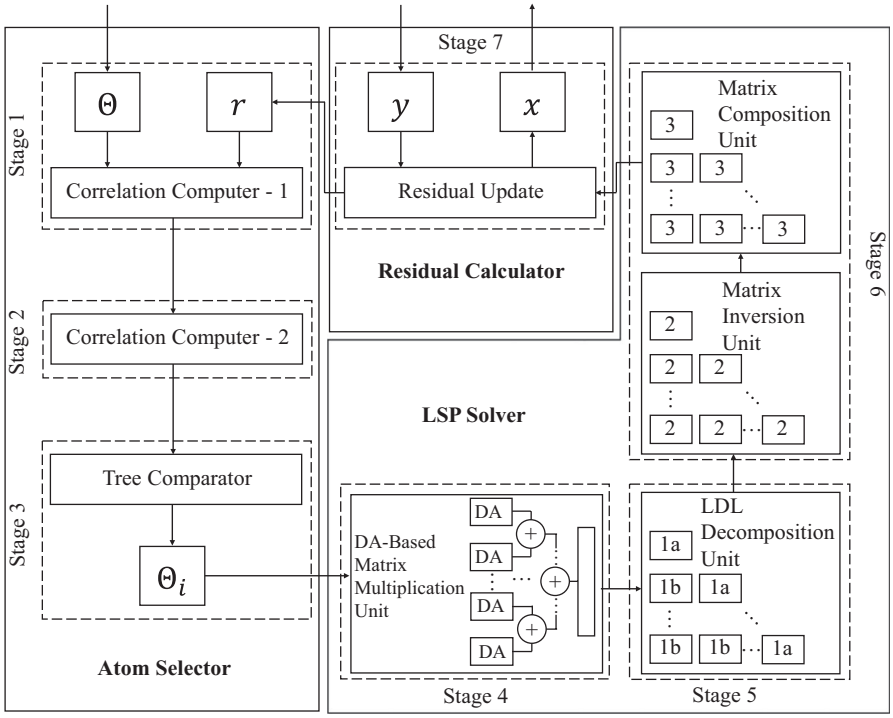


FIGURE 1 Proposed hardware implementation of orthogonal matching pursuit algorithm

in accordance with (7) and (8), respectively. The structure of the systolic array of the matrix decomposition unit is depicted in Figure 2A, and the structures of PE_{1a} and PE_{1b} are shown in detail in Figure 3A and 3B, respectively. Each PE_{1a} computes the elements of the diagonal matrix **D** using the elements of **C** obtained from the matrix multiplication unit and the s_{in} received from PE_{1b}. Each PE_{1b} computes the value of each element in the triangular matrix **L** using the fetched-in c_{in}, s_{in}, and d_{in}, and calculates the value of s_{out}. The inner register after the 2:1 multiplexer selectively stores either l_{in} or l_{ij} and, then, transmits to the next processing element. The matrix decomposition unit employs 136 PE s: 16 PE_{1a} and 120 PE_{1b} elements related to the diagonal and lower-left half of **L**, respectively. The matrix decomposition is completed in 31 clock cycles.

- *Matrix inversion:* Because **D** is a diagonal matrix, it can be inverted by simply computing the reciprocals of its diagonal elements. The division IP from the Synopsys DesignWare™ library is used for this purpose [11]. In contrast, **L** can be inverted according to (10) using the systolic array shown in Figure 2B. Because all the elements on the diagonal become one after the inversion, 120 PE_2 processing elements are necessary to invert **L**. The detailed architecture of PE_2 is shown in Figure 3C. First, after the subtractor, the inner register is initialized to l_p, which is equal to l_{ij} from the previous LDL decomposition stage. Then, each PE_2 operates as a multiply-accumulator to compute a_{ij}. The values of l_{out} and a_{out} in Figure 3C are transferred to the right and lower PE_2, respectively, for the subsequent computations. Note that the inversion is completed in 15 cycles.
- *Matrix composition:* The matrix composition unit computes $\mathbf{C}^{-1}$ from the outputs of the matrix inversion unit according to

$$c_{ij}^{-1} = -\sum_{k=1}^{m} l_{ki} l_{kj} d_{kk}^{-1}. \tag{11}$$

The systolic array with 136 PE_3 shown in Figure 2C is used to compute the matrix product of **L**, $\mathbf{L}^{-1}$, and **D**. The detailed architecture of the processing element is shown in Figure 3D. Because **L** and **D** are triangular and diagonal matrices, respectively, they can be multiplied by simply accumulating dot products. All three matrices have the same 1616 size. Thus, 16 cycles are sufficient to complete the operation.

After $\mathbf{C}^{-1}$ is obtained, the sparse representation x is multiplied by the set $\mathbf{\Theta}_i$ of selected atoms in order to compute the reconstructed signal. Then, the result is subtracted from the measurement vector y to update the residual for the next iteration.

In the proposed architecture, as shown in Figure 1, the LSP solver is composed of four consecutive sub-blocks, enabling simultaneous computations. All seven building blocks, namely the correlation computing unit, tree comparator, residual calculation unit, DA-based matrix multiplication unit, and the three systolic arrays for the LSP solver, use the same number of clock cycles, 32, when

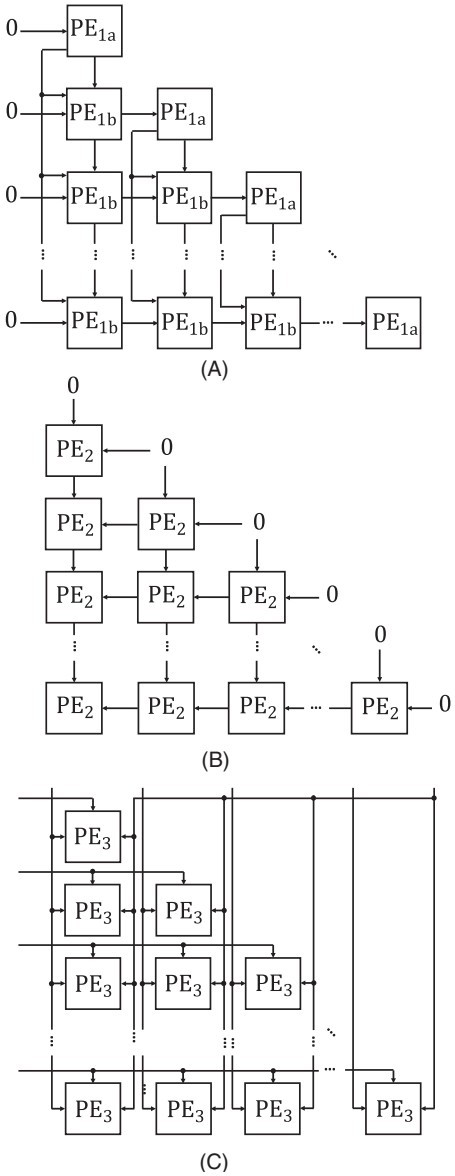


FIGURE 2 Structure of systolic arrays used in LSP solver (A) for LDL decomposition, (B) for matrix inversion, and (C) for matrix composition

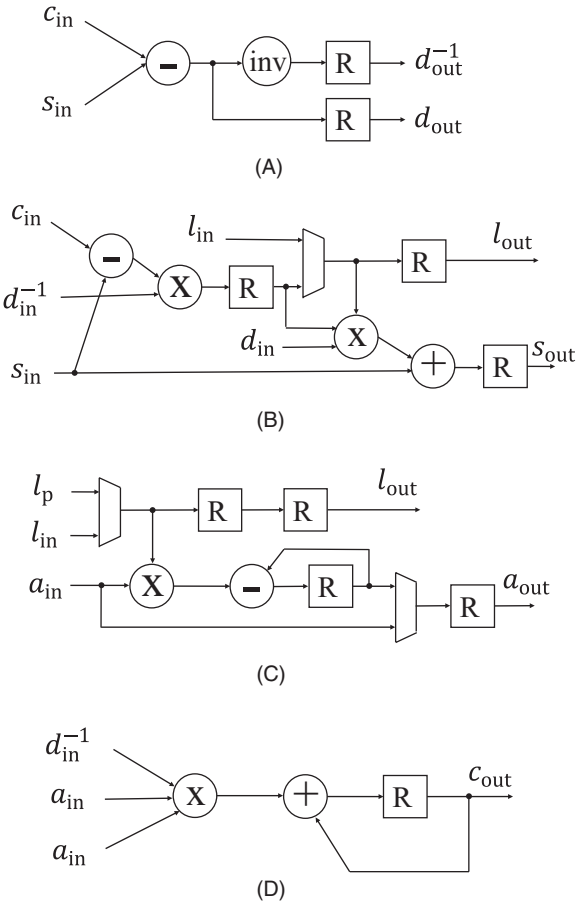


FIGURE 3 Structures of processing elements in LSP solver: (A) PE_{1a}, (B) PE_{1b}, (C) PE_2, and (D) PE_3

$N=256$, $M=64$, and $m=16$, where N is the size of the original signal, M is the measurement vector size, and m is the sparsity level. Therefore, the proposed seven-stage pipelined architecture can reconstruct an original frame of 256 samples from 64 compressively sensed signals every 32 cycles. The pipelined processing is one of the novel contributions of this paper.

3.2 | Matrix multiplication unit based on the sum-of-product principle

The matrix multiplication related to LSP solving, $\mathbf{C}=\mathbf{\Theta}^{\mathrm{T}}\mathbf{\Theta}$, can be expressed as

$$\begin{pmatrix} c_{11} & \cdots & c_{1i} \\ \vdots & \ddots & \vdots \\ c_{i1} & \cdots & c_{ii} \end{pmatrix} = \begin{pmatrix} \theta_{11} & \cdots & \theta_{1M} \\ \vdots & \ddots & \vdots \\ \theta_{i1} & \cdots & \theta_{iM} \end{pmatrix} \begin{pmatrix} \theta_{11} & \cdots & \theta_{i1} \\ \vdots & \ddots & \vdots \\ \theta_{1M} & \cdots & \theta_{iM} \end{pmatrix}. \tag{12}$$

The resultant matrix $\mathbf{C}$ can be obtained by separately computing its elements as

$$c_{pq} = \begin{bmatrix} \theta_{p1} & \cdots & \theta_{pM} \end{bmatrix} \begin{bmatrix} \theta_{q1} \\ \vdots \\ \theta_{qM} \end{bmatrix}, \tag{13}$$

for $p,q=1,2,\ldots,i$ and $p\le q$. The computation of (13) can be handled by the proposed SoP-based matrix multiplication unit. As shown in Figure 4, this consists of M partial product generators (PPGs), which correspond to M multipliers, and a $\log_2 M$-stage adder tree followed

by a serial-in-parallel-out (SIPO) shift register with $m(m+1)/2$ outputs, where M and m are the width of the reconstruction matrix and the sparsity level, respectively. In every bit cycle, each PPG computes the partial product of the corresponding pair of θ_p and θ_q. Because $\mathbf{C}$ is symmetric across its diagonal, only half of the matrix can be computed to save resources. Thus, PPGs require $m(m+1)/2$ clock cycles to compute all the partial products. Then, the adder tree sums all the M newly computed partial products to obtain each desired element of the resultant matrix $\mathbf{C}$. Afterwards, the SIPO shift register receives and stores the element values. At the end of each pipeline time slot, the SIPO register outputs all $m(m+1)/2$ values to Stage 5, that is, the LDL decomposition unit of the LSP solver.

3.3 | Matrix multiplication unit based on distributed arithmetic

The traditional approach for computing (13) is to use multiply-accumulators (MACs), where each MAC is responsible for computing one element of $\mathbf{C}$. Because only the lower-left half of $\mathbf{C}$ needs to be computed, $m(m+1)/2$ MACs are sufficient. The number of pipelined stages and MACs required is proportional to the size of $\boldsymbol{\Theta}$ and the value of m, respectively, which leads to a large extension of area and a significant increase of latency for the entire design. To resolve this issue, we propose a different structure for the matrix multiplication unit. It is based on DA, an efficient method for computing inner products and transforms based on multiply-accumulate operations. The result is calculated by accumulating the partial products of the serially shifted bits of a vector with the elements of another vector. The partial products are stored in a look-up table (LUT), which is addressed by a nibble of the shifted bits.

Equation (13) can be rewritten as a sum of partial matrix multiplications, where each operand takes the form of the multiplication between a row and a column vector as

$$\begin{aligned} c_{pq} &= \begin{bmatrix} \theta_{p1} & \cdots & \theta_{pM} \end{bmatrix} \begin{bmatrix} \theta_{q1} \\ \vdots \\ \theta_{qM} \end{bmatrix} \\ &= \begin{bmatrix} \theta_{p1} & \theta_{p2} & \theta_{p3} & \theta_{p4} \end{bmatrix} \begin{bmatrix} \theta_{q1} \\ \theta_{q2} \\ \theta_{q3} \\ \theta_{q4} \end{bmatrix} \\ &+ \begin{bmatrix} \theta_{p5} & \theta_{p6} & \theta_{p7} & \theta_{p8} \end{bmatrix} \begin{bmatrix} \theta_{q5} \\ \theta_{q6} \\ \theta_{q7} \\ \theta_{q8} \end{bmatrix} \\ &+ \begin{bmatrix} \theta_{pM-3} & \theta_{pM-2} & \theta_{pM-1} & \theta_{pM} \end{bmatrix} \begin{bmatrix} \theta_{qM-3} \\ \theta_{qM-2} \\ \theta_{qM-1} \\ \theta_{qM} \end{bmatrix}. \end{aligned} \tag{14}$$

Let us examine the first partial term in (14) as

$$\begin{aligned} r_1 &= \begin{bmatrix} \theta_{p1} & \theta_{p2} & \theta_{p3} & \theta_{p4} \end{bmatrix} \begin{bmatrix} \theta_{q1} \\ \theta_{q2} \\ \theta_{q3} \\ \theta_{q4} \end{bmatrix} = \\ &\theta_{p1}\theta_{q1} + \theta_{p2}\theta_{q2} + \theta_{p3}\theta_{q3} + \theta_{p4}\theta_{q4}, \end{aligned} \tag{15}$$

and assume that θ_{qk} for $k = 1, 2, 3, 4$ is a fractional two's complement number, with $-1 \le \theta_{qk} \le 1$, that is,

$$\theta_{qk} = -\theta_{qk,0}2^0 + \sum_{l=1}^{L-1} \theta_{qk,l}2^{-l},\ k = 1, 2, 3, 4, \tag{16}$$

where L is the word length of θ_{qk} and l indicates the lth bit, counting from the most significant bit (MSB). By substituting (16) into (15),

$$\begin{aligned} r_1 = \theta_{p1} &\sum_{l=0}^{L-1} \sigma(l)\,\theta_{q1,l}2^{-l} + \theta_{p2} \sum_{l=0}^{L-1} \sigma(l)\,\theta_{q2,l}2^{-l} \\ &+ \theta_{p3} \sum_{l=0}^{L-1} \sigma(l)\,\theta_{q3,l}2^{-l} + \theta_{p4} \sum_{l=0}^{L-1} \sigma(l)\,\theta_{q4,l}2^{-l}, \end{aligned} \tag{17}$$

where

$$\sigma(l) = \begin{cases} -1, & \text{for } l = 0, \\ 1, & \text{otherwise.} \end{cases} \tag{18}$$

Then, (17) can be expanded to a sum of products as

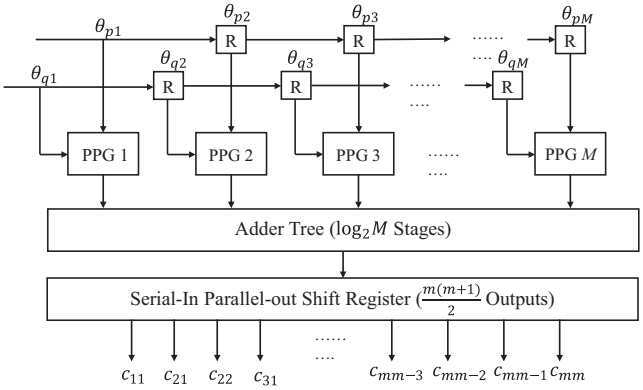


FIGURE 4 Proposed structure for sum-of-product-based matrix multiplication

$$
\begin{aligned}
r_1 = {} & \sigma(0)\left[\theta_{p1}\theta_{q1,0}+\theta_{p2}\theta_{q2,0}+\theta_{p3}\theta_{q3,0}+\theta_{p4}\theta_{q4,0}\right]2^{0} \\
& +\sigma(1)\left[\theta_{p1}\theta_{q1,1}+\theta_{p2}\theta_{q2,1}+\theta_{p3}\theta_{q3,1}+\theta_{p4}\theta_{q4,1}\right]2^{-1} \\
& +\sigma(2)\left[\theta_{p1}\theta_{q1,2}+\theta_{p2}\theta_{q2,2}+\theta_{p3}\theta_{q3,2}+\theta_{p4}\theta_{q4,2}\right]2^{-2} \\
& \vdots \\
& +\sigma(L-2)\left[\theta_{p1}\theta_{q1,L-2}+\theta_{p2}\theta_{q2,L-2}+\theta_{p3}\theta_{q3,L-2}+\theta_{p4}\theta_{q4,L-2}\right]2^{-(L-2)} \\
& +\sigma(L-1)\left[\theta_{p1}\theta_{q1,L-1}+\theta_{p2}\theta_{q2,L-1}+\theta_{p3}\theta_{q3,L-1}+\theta_{p4}\theta_{q4,L-1}\right]2^{-(L-1)},
\end{aligned}
\tag{19}
$$

where r_1 is the sum of L partial products. Each partial product is the sum of the products of the variables θ_p and one bit of θ_{qk}. Because $\theta_{qk,l}$ ($k=1,2,3,4$ and $l=0, \ldots, L-1$) is either 0 or 1, each partial product has $2^4=16$ possible values. It is undoubtedly more efficient to pre-compute all these values and store them in a LUT, rather than to compute them online. Each set of [$\theta_{q1,l}$ $\theta_{q2,l}$ $\theta_{q3.l}$ $\theta_{q4,l}$] bits can be used to address the LUT in order to read out a pre-computed partial product. The output of the LUT is accumulated during L cycles to obtain the value of r_1. The other partial matrix multiplications for (r_2, r_3, …, $r_{M/4}$) in (14) can be computed similarly.

The conventional hardware structure of DA for the computation of each partial product in (19) is shown in Figure 5, where a LUT with 16 partial products is followed by a shift accumulator. In each cycle, the data retrieved from the LUT are added to or subtracted from right-shifted accumulated data, and address bits are orderly shifted out from least significant bits (LSBs) to MSBs. The adder/subtractor is controlled by a sign bit, which is set to 1 only for the data fetched by the address of the MSB set; otherwise, it is set to 0.

The LUT in Figure 5 has to be implemented by using a random access memory (RAM), because different sets of partial products or different LUT contents should be used in the LUT for the calculation of r_1, r_2, r_3, …, $r_{M/4}$. However, in the proposed architecture, a DA table with 15 registers is used instead of storing 16 pre-computed values in a RAM-based LUT in order to adapt to the proposed pipelined design. Thus, all the partial products in the LUT can be retrieved in the same cycle. Let us denote the value of the i^{th} memory cell of the LUT as

$$
v_i = \sum_{j=0}^{3} \theta_{j+1} i_j, \quad 0 \le i \le 15, \tag{20}
$$

where θ_{j+1} indicates an element of the row vector of each partial matrix multiplication and i_j is the $(j+1)$th bit of the 4-bit binary representation of integer i. Then, the values of v_i can be pre-computed and stored in a register-based LUT as shown in Figure 6. Because the value corresponding to the address [0 0 0 0] is always zero, only 15 registers are required for the proposed DA-LUT. The 15 registered values of v_i can be updated by using only seven adders to allow multiplierless implementation and access in parallel. The address of the LUT can be used as a selection signal for a 16:1 multiplexor.

For $M=64$, 16 partial matrix multiplications are required in total. Therefore, the matrix multiplication unit is constructed from 16 DA-LUTs and a four-stage adder tree, as depicted in Figure 7. Note that A^k ($k=1,2,3, \ldots, 16$) represents the LUT address formed by each bit set [$\theta_{q4k-3,l}$ $\theta_{q4k-2,l}$ $\theta_{q4k-1,l}$ $\theta_{q4k,l}$], and T^k ($k=1,2,3, \ldots, 16$) denotes four elements consecutively shifted to the LUT, θ_{p4k-3}, θ_{p4k-2}, θ_{p4k-1}, and θ_{p4k}. A SIPO shift register receives the outputs of the adder tree, shifts them over 32 clock cycles, and then feeds them in parallel to the PE_1 processing elements of the matrix decomposition unit.

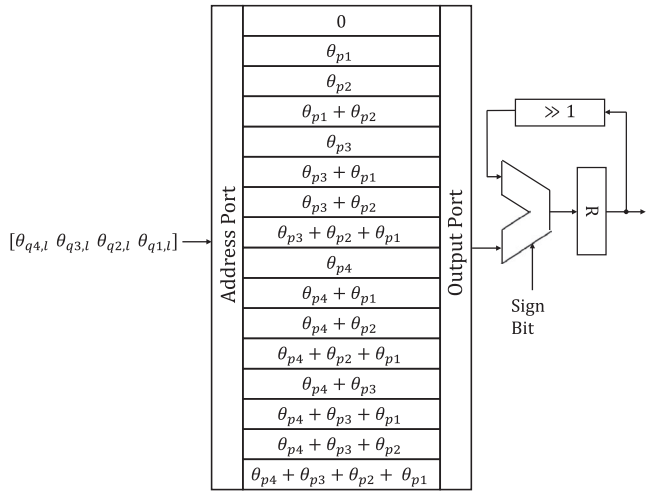


FIGURE 5 Conventional implementation of distributed arithmetic

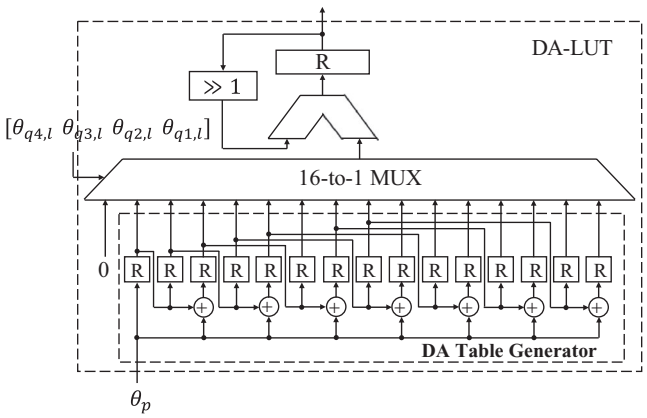


FIGURE 6 Proposed implementation of distributed arithmetic with register-based look-up table

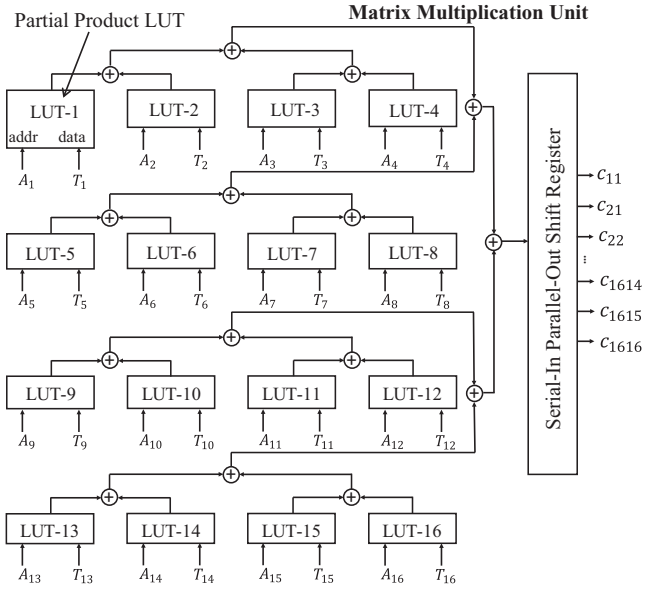


FIGURE 7 Design of the matrix multiplication unit

3.4 | Versatile design based on the proposed architecture

Many hardware implementations have been proposed for different CS applications. However, most of the known architectures have been designed for particular values of the input parameters N, M, and m, which are fixed and known in advance. Therefore, it can be difficult to adapt such constrained implementations to scenarios in which the values of N, M, and m are matched to the properties of a signal. To tackle this challenge, we propose a versatile design based on the architectures introduced in the previous sections. The design, which has been prototyped for reconstructing an m-sparse signal compressively sensed by using an N-by-M matrix, assumes that the reconstruction matrix can be as large as 1024×256 (the largest size of the reconstruction matrix used in state-of-the-art approaches) and the sparsity m of the signal can range up to 30.

The proposed versatile design consists of 11 stages, as shown in Figure 8. Each stage completes its task in maximum 32 clock cycles.

- Stages 1 and 2 compute the correlation between the reconstruction matrix $\mathbf{\Theta}$ and the newly updated residual r. Each stage is composed of four correlation computers, each of which can compute 32 partial multiplications in one pipeline time slot. As $M_{\max} = 256$, each stage can compute up to 128 partial multiplications, that is, half of the matrix multiplication, in 32 cycles.
- Stage 3, the tree comparator, compares N values from Stage 2 in $\log_2 N$ cycles. As $N_{\max} = 1024$, the maximum number of cycles required by stage 3 is 10.
- Stage 4 handles the matrix multiplication between $\mathbf{\Theta}_i$ and $\mathbf{\Theta}_i^{\mathrm{T}}$ to find $\mathbf{C}$ by using the previously proposed DA-based design. The output delay of Stage 4 depends on the word length L. If the design is prototyped for 16-, 24-, or 32-bit data, then Stage 4 requires 16, 24, or 32 cycles, respectively.
- Stages 5, 6, 7, and 8 perform the LDL decomposition to compute $\mathbf{L}$ and $\mathbf{D}$. As $m_{\max} = 30$, the four stages are designed to be able to decompose a 30-by-30 matrix. According to the computational properties of the systolic array–based design of the LDL decomposition, the number of rows in each stage is properly calculated, as shown in Figure 9.
- Stage 9 inverts $\mathbf{L}$ in m cycles, and Stage 10 composes $\mathbf{L}$, $\mathbf{L}^{-1}$, and $\mathbf{D}$ to find $\mathbf{C}^{-1}$ in $m-1$ cycles. As $m_{\max} = 30$, Stages 9 and 10 require 30 and 29 cycles at most, respectively.
- Stage 11 calculates and updates the residual r in m cycles.

3.5 | Implementation and synthesis results

We followed a typical front-end design flow for the design and implementation of the proposed OMP, that is, algorithmic level and bit-accurate modeling, register-transfer level (RTL) design, and logic synthesis. We used MATLAB for algorithmic level modeling, and a System C fixed point library for bit-accurate modeling, and the results have been used as the golden data for verifying the synthesized model of the proposed OMP.

In the proposed design, we could obtain a signal to noise ratio (SNR) higher than 60 dB with a 16-bit word length when $N = 256$ and $M = 64$. We included the ratio of correctly recovered signals as a function of the number of measurements M and of the sparsity level m in order to show accurate metrics of the proposed architectures. Figure 10A shows the ratio of correctly recovered signals as a function of m for different values of M when $N = 256$. A total of 1000 random signals was used, and the percentage of those correctly recovered was obtained from bit-accurate simulations. Note that for a specific number of measurements, the recovery percentage decreases as the sparsity level increases. It is also seen from Figure 10B, where the ratio of recovered signals is shown as a function of M for different values of m, that more measurements are required to retain the percentage of recovery as the sparsity level increases.

Two versions of the proposed architecture, respectively using a DA- and SoP-based matrix multiplication unit, have been coded in Verilog hardware description language (HDL) and synthesized by the Synopsys Design Compiler with the TSMC 90 nm standard CMOS library. The (N, M, m) parameters for the OMP are set to

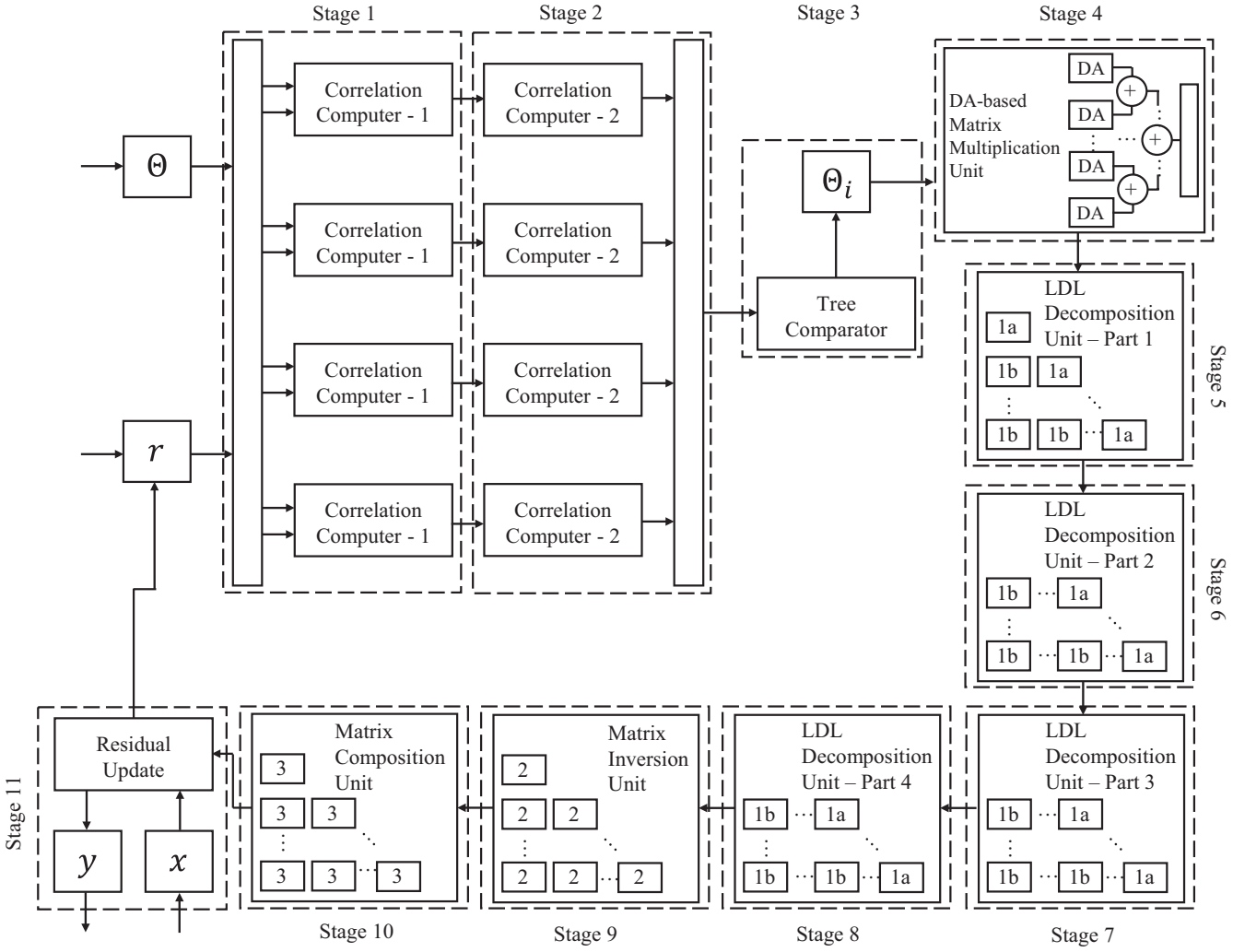


FIGURE 8 Proposed design; it is reconfigurable for wide ranges of N, M, and m

(256, 64, 16). Table 1 shows the results of the logic synthesis for the proposed architectures. The design with the DA- and SoP-based matrix multiplication units can operate at the maximum frequencies of 312 MHz and 333 MHz, respectively, taking 512 cycles in both cases to complete the reconstruction of a 256 sample frame. Because the required number of iterations is 16 when the sparsity level is 16, it takes $16 \times 512 = 8192$ cycles to complete the reconstruction of a sparse signal. Our designs may occupy a larger area than the existing ones because of the fine-grain pipelining. However, the area overhead can be significantly counterbalanced by using the proposed register-based DA-LUT. Specifically, the area of the matrix multiplication unit based on DA-LUT is only 0.128 mm^2, whereas the common approach based on MACs requires an area of 0.778 mm^2. Therefore, the total area of the proposed design, which is 8.215 mm^2, can compete with known architectures. Besides, with a

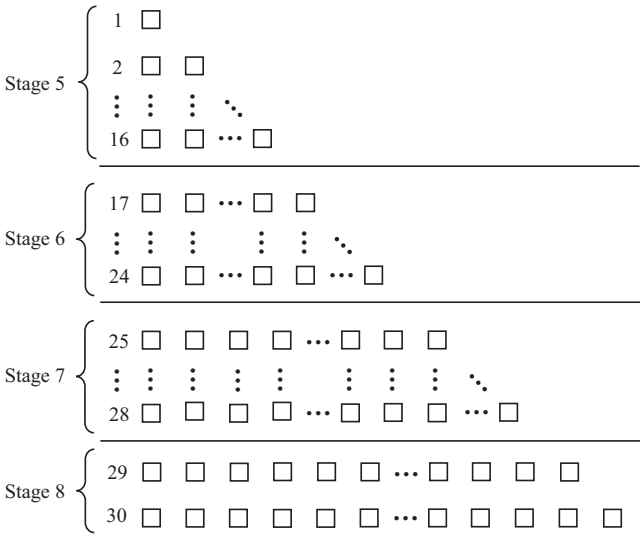


FIGURE 9 Arrangement of processing elements in Stages 5, 6, 7, and 8

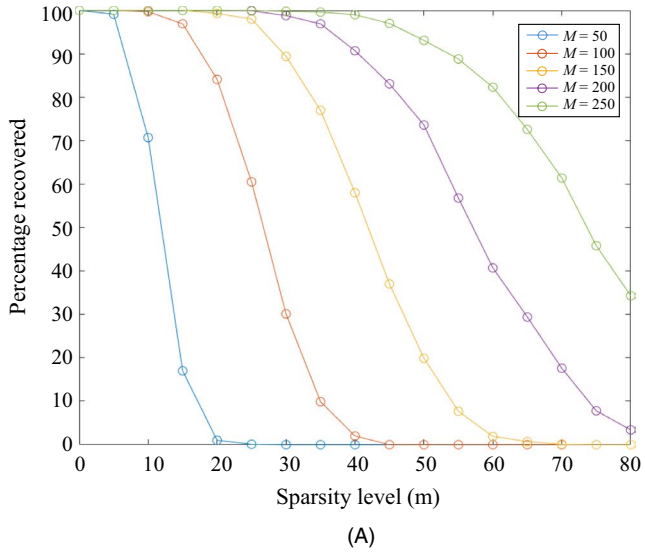


FIGURE 10 The percentage of input signals correctly recovered when $N = 256$ (A) as a function of the sparsity level m for different number of measurements, (B) as a function of the number of measurements M for different sparsity levels

marginal area overhead, the design with the SoP-based unit achieves an even higher throughput than that of the design with the DA-based unit.

We also implemented a design in which the matrix multiplication unit is constructed from two systolic arrays. The synthesis results for three of our designs are listed in Table 2. The corresponding parts of each design occupy nearly the same area, except for the atom selector. When coupling the atom selector and the matrix multiplication unit of the LS solver, the design with systolic array–based matrix multiplication requires more registers and multiplexers for data transfer. Therefore, the area of this part of the design is slightly larger than that of the designs with DA- and SoP-based matrix multiplications.

TABLE 1 Performance summary of proposed architectures for $N = 256$, $M = 64$, and $m = 16$ based on synthesis results and using TSMC 90-nm library

Design	DA-based design	SoP-based design
Maximum operating frequency (MHz)	312	333
Number of cycles per iteration (cycles)	512	512
Number of samples processed per time unit (sample/s)	156.25×10^6	166.67×10^6
Area (mm^2)	8.215	8.297
Delay per iteration (ns)	1638.4	1536
Area-delay product ($mm^2 \times ns$)	13 459.5	12 744.2

The multiplierless DA- and SoP-based units are 83.57% and 73.18% smaller than the systolic-array-based unit, while the total areas of the LS solvers are reduced by 29.59% and 25.91%, respectively. Table 3 compares the very large scale integration (VLSI) implementation of the proposed architectures and that of the existing architectures for compressive sensing reconstruction. All the architectures have been designed for a measurement matrix size of $(N, M) = (256, 64)$ and compared in terms of technology, sparsity, operating frequency (MHz), reconstruction time (μs), and availability of pipelining. The proposed designs offer the lowest reconstruction times among the designs in Table 3. In addition, the proposed architecture can start reconstructing a new sparse signal every 32 clock cycles, whereas the existing designs cannot start processing an incoming sparse signal before reconstruction of the previous one is completed. Therefore, the proposed design with DA-based unit can reconstruct $256 \times 312 \times 10^6 / (32 \times 16) = 156 \times 10^6$ samples per second, and the design with SoP-based unit can reconstruct $256 \times 333 \times 10^6 / (32 \times 16) = 166.5 \times 10^6$ samples per second. On the other hand, existing designs such as the one by Shi and others [8] can only process $256 / (45.42 \times 10^{-6}) = 5.636 \times 10^6$ samples per second, providing much lower throughputs than the proposed designs.

4 | CONCLUSION

We have proposed an efficient architecture for high-throughput and low-area hardware implementation of the OMP algorithm for CS reconstruction. The throughput is dramatically enhanced by the proposed pipelined structure, which is constructed on top of the iterative nature of the OMP algorithm. We have also suggested a register-based DA scheme for the matrix multiplication unit. The synthesis results show that the proposed design consumes 6.26% more area but is characterized by 94.47%

TABLE 2 Area (mm^2) comparison of designs with systolic array-, DA-, and SoP-based units for matrix multiplication based on synthesis results and using TSMC 90-nm library

Design	Design with SA-based matrix multiplication unit	Design with SoP-based matrix multiplication unit	Design with DA-based matrix multiplication unit
Atom selector	6.6897	6.5511	6.5511
LS solver			
Matrix multiplication 1	0.3720	0.2087	0.1279
Matrix multiplication 2	0.4061	0.2087	0.1279
LDL decomposition	1.0333	1.0333	1.0333
Triangular matrix inversion	0.3251	0.3251	0.3251
Matrix composition	0.0615	0.0615	0.0615
Residual calculator	0.0250	0.0250	0.0250

Note: SA: systolic array, SoP: sum-of-product, and DA: distributed arithmetic.

TABLE 3 Comparison of VLSI implementations of proposed and existing architectures

Algorithm	Technology (nm)	(N, M)	Sparsity	Operating Frequency (MHz)	Reconstruction Time (μ s)	Pipelining
OMP [8]	130	(256, 64)	16	167	45.42	No
R-SGP [9]	90	(256, 64)	8	150	61.97	No
MCMIB [10]	90	(256, 64)	12	141	72.25	No
Proposed (DA-based design)	90	(256, 64)	16	312	26.26	Yes
Proposed (SoP-based design)	90	(256, 64)	16	333	24.60	Yes

less area-delay-product than state-of-the-art designs for $N = 256$, $M = 64$, and $m = 16$. With its superior processing speed, the proposed design promises to bring about a significant contribution in various applications, in particular real-time ones, in which high-speed paradigms for information and signal processing are required.

ACKNOWLEDGMENTS

This work was supported by Myongji University 'Assistant Professor Support Program'. The authors are grateful to the School of Engineering at Myongji University for the technical support and management.

ORCID

Sang Yoon Park https://orcid.org/0000-0002-7168-6455

REFERENCES

1. L. D. Donoho, *For most large underdetermined systems of linear equations, the minimal 1-norm solution is also the sparsest solution*, Comm. Pure Appl. Math. **59** (2006), 797–829.
2. M. Davenport, *The Fundamentals of compressive sensing*, SigView, 2013.
3. W. L. Chan et al., *A single-pixel terahertz imaging system based on compressed sensing*, Appl. Phys. Lett. **93** (2008), 121105.
4. U. Gamper, P. Boesiger, and S. Kozerke, *Compressed sensing in dynamic MRI*, Magn. Reson. Med. **59** (2008), 365–373.
5. T. R. Braun, *An evaluation of GPU acceleration for sparse reconstruction*, in Proc. SPIE, Signal Process, Sensor Fusion, Target Recognit. XIX, Orlando, FL, USA, 2010, pp. 769715:1–15.
6. A. Septimus and R. Steinberg, *Compressive sampling hardware reconstruction*, in Proc. IEEE Internat. Symp. Circuits Syst., Paris, France, 2010, pp. 3316–3319.
7. J. L. V. M. Stanislaus and T. Mohsenin, *High performance compressive sensing reconstruction hardware with QRD process*, in Proc. IEEE Internat. Symp. Circuits Syst., Seoul, Rep. of Korea, 2012, pp. 29–32.
8. W. Shi et al., *An extensible and real-time compressive sensing reconstruction hardware for WBANs using OMP*, in Proc. IEEE Internat. Conf. ASIC (ASICON), Shenzhen, China, Oct. 2013, pp. 1–4.
9. Y. M. Lin et al., *Low-complexity stochastic gradient pursuit algorithm and architecture for robust compressive sensing reconstruction*, IEEE Trans. Signal Process. **65** (2017), 638–650.
10. J. W. Jhang and Y. H. Huang, *A high-SNR projection-based atom selection OMP processor for compressive sensing*, IEEE Trans. Very Large Scale Integr. VLSI Syst. **24** (2016), 3477–3488.
11. Synopsis Inc., *DesignWare Building Block IP*, Document Overview, DWBB_201312.2, I-2013.12-SP2, 2014.

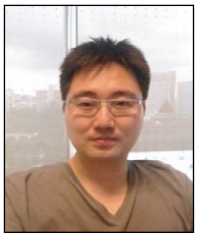
AUTHOR BIOGRAPHIES

Vu Quan Nguyen received his BS degree in electronics and telecommunications engineering from Da Nang University of Science and Technology, Da Nang, Vietnam, in 2014, and his MS degree in electronic engineering from Myongji University, Yongin, Rep. of Korea, in 2018. Since 2018, he has been working as a digital engineer for Radio Management Solutions Division, OnPoom Co., Ltd., Seoul, Rep. of Korea. His main research interests are digital signal processing and system-on-chip design.mediaResourceGroup>

Woo Hyun Son received his BS degree in electronics engineering from Myongji University, Yongin, Rep. of Korea, in 2018. He is currently working on his MS degree in electronic engineering at Myongji University. His research interests include design and implementation of low-power and high-performance digital signal processing systems.

Marek Parfieniuk received his MSc and PhD degrees in computer science, both with distinction, from the Bialystok University of Technology, in 2000 and 2007, respectively. He also received a habilitation degree in computer science from the Czestochowa University of Technology, in 2015. From 2000 to 2003, he worked for Computerland S.A. as an enterprise software developer. From 2000 to 2018, he was an assistant lecturer and then an assistant professor at the Faculty of Computer Science, Bialystok University of Technology, Poland. Since 2018, he has been with the Institute of Informatics, University of Bialystok, where he is now an associate professor. His research interests include filter banks and transforms for digital signal processing: the theory behind them, their design methods, software and hardware implementation techniques, and their applications, especially in data compression. He is also interested in multimedia standards, distributed systems, parallel computing, and knowledge management. He has authored about 100 publications, including three books, and two successful patent applications. He has served as a reviewer for prestigious journals published by the IEEE and Elsevier. He is a senior member of the IEEE and a member of the ACM.

Luong Tran Nhat Trung received his BS degree in electronics and telecommunications engineering from Da Nang University of Science and Technology, Da Nang, Vietnam, in 2016, and his MS degree in electronic engineering from Myongji University, Yongin, Rep. of Korea, in 2019. Since 2019, he has been working as a model engineer for the Engineering Department, Esilicon, Da Nang, Vietnam. His main research interests include high-throughput and low power digital circuit design.



Sang Yoon Park received his BS degree in electrical engineering and his MS and PhD degrees in electrical engineering and computer science from Seoul National University, Seoul, Rep. of Korea, in 2000, 2002, and 2006, respectively. In 2007 he joined the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore, as a research fellow. From 2008 to 2014, he was a research scientist with the Institute for Infocomm Research, Singapore. Since 2014, he has been with the Department of Electronic Engineering, Myongji University, Yongin, Rep. of Korea, where he is currently an associate professor. His research interests include design of dedicated and reconfigurable architectures for low-power and high-performance digital communication systems.