High-performance ASIC realization of orthogonal matching pursuit algorithm

Vu Quan Nguyen and Sang Yoon Park

Department of Electronic Engineering, Myongji University, 116 Myongji-ro, Cheoin-gu, Yongin-si, Gyeonggi-do 17058, Korea

Abstract: Compressive sensing (CS) is perceived as a breakthrough in sampling theory, as it proves that sparse signals can be reconstructed from fewer samples than required by the Shannon-Nyquist theorem. However, CS can hardly be applied to real-time applications because reconstruction algorithms are computationally demanding. To tackle this problem, in this paper, we propose a new high-speed architecture for implementing the orthogonal matching pursuit (OMP), which is one of the most popular algorithms for CS reconstruction. Specifically, a novel pipelined systolic architecture and an optimized scheduling strategy are proposed. From the synthesis results, we find that the proposed design takes 1.638 µs to reconstruct 16-sparse signal, which is 19.2 times faster than the existing VLSI implementation of the OMP.

Keywords: compressive sensing (CS), orthogonal matching pursuit (OMP), ASIC implementation, signal processing

Classification: Integrated circuits

References

1 Introduction

Over the past decade, compressive sensing (CS) has become popular in acquisition of sparse signals (i.e. signals that have only a few non-zero values) [1, 2, 3], as it allows for reconstruction of such a signal from much fewer measurements than required by the conventional sampling at the Nyquist rate. Therefore, CS makes it possible to dramatically reduce resource consumption, processing time, and power consumption required for data acquisition, transmission, and manipulation in diverse applications.

One of CS applications is in medical image acquisition. The sparsity property of a magnetic resonance image (MRI) made the CS technique a successful application in improving the image quality through the reduction of the measurement amount [1]. As for application in photography, CS allows enhancement in the acquisition speed of the single-pixel Terahertz camera [2]. In high-speed applications like the 1080p HD video, CS helps relieve the burden on data acquisition process, as the number of acquired datum is significantly fewer [3]. These high resolution signal processing applications require an ultrafast reconstruction to meet their real-time requirements.

Orthogonal matching pursuit (OMP) is one of the most widely used reconstruction algorithms for the CS. In [4], hardware solution is presented to accelerate the computation for OMP. Modified versions of the original OMP have also been proposed to speed up the algorithm in [5]. However, the OMP is not well suited for pipelined hardware architectures because of its iterative nature. It is also difficult to develop a high-throughput implementation of the OMP to meet real-time requirements, as the algorithm is based on matrix computations of high computational complexity.

In this paper, we propose solutions to these problems: 1) a novel pipelined systolic architecture for the OMP, 2) new processing elements that facilitate pipelining, and 3) a scheduling strategy for pipelined iterations. In the proposed design, all stages are designed to consume the same number of clock cycles in order to avoid cycle wasting and to maximize resource utilization.

The rest of this paper is organized as follows. Section 2 presents brief reviews of compressive sensing technique and orthogonal matching pursuit algorithm. Section 3 describes the proposed pipeline architecture with newly designed processing elements. Synthesis results and comparison between the proposed design...
and other existing implementations are discussed in Section 4. Finally, conclusions are given in Section 5.

2 Background review

2.1 Compressive sensing

Compressive sensing, since introduced by Candes et al. [6] in 2006, has been widely used in various applications of signal processing. CS is built upon the fundamental that signal can be represented in a proper domain or basis using only a few non-zero coefficients. Nonlinear optimization then can be used to recover such compressively sensed signals. The recovery is possible as long as the signal is sparse in the represented basis. Assume that signal \( f \) is sparse in the basis \( \Phi \) with the sparsity representation \( x \), then \( f \) can be described by the equation

\[
f = \Phi x,
\]

where \( f \in \mathbb{R}^N, \Phi \in \mathbb{R}^{N \times N} \) and \( x \in \mathbb{R}^N \). As most of the energy is distributed only on a few coefficients, the other ones can be zeros; yet the signal is not distorted. Hence, \( x \) will be mostly zero.

Acquisition process of the signal vector \( f \) via linear transformation with measurement matrix \( \Psi \) results in measurement vector \( y \). That is

\[
y = \Psi f = \Psi \Phi x = \Theta x,
\]

with \( y \in \mathbb{R}^M, \Psi \in \mathbb{R}^{M \times N}, \Theta \in \mathbb{R}^{M \times N} \) and \( \Theta \) is reconstruction matrix. Since \( N > M \), Eq. (2) is an underdetermined system of linear equations. Minimization of the \( L^p \)-norm is the solution to such problems. Solution for Eq. (2) by \( L^2 \) minimization can be expressed as

\[
x = \arg \min_x \| \Theta x - y \|_2.
\]

The problem of Eq. (3) can be solved by greedy iterative methods. For hardware implementation, greedy methods such as matching pursuit (MP) and orthogonal matching pursuit (OMP) are more preferable for their simplicity, high speed of reconstruction and low implementation cost [7, 8, 9].

2.2 Orthogonal matching pursuit

Orthogonal matching pursuit is the derivative of the original matching pursuit algorithm, and the algorithm is summarized in Table I. The reconstruction matrix \( \Theta \) and measurement vector \( y \) are the inputs and the sparse reconstruction \( x \) is the output of the algorithm. OMP is iterative algorithm and the number of iterations depends on the signal sparsity \( K \). In each iteration, one column of \( \Theta \) which has the most correlation with the residual \( y \) is chosen and then its effect is removed to create new residual. A new value is reconstructed in each iteration, and the procedure is repeated until all the \( K \) values are reconstructed. The LS problem in Step 3 can be solved by

\[
x = (\Theta^T \Theta)^{-1} \Theta^T y,
\]

which involves finding the inverse of matrix. In this paper, modified method of Cholesky factorization, known as LDL decomposition, is used to compute matrix inversion as it avoids using square-root operation.
3 Proposed novel pipeline architecture of OMP

3.1 Hardware architecture

In the existing hardware implementation of the OMP [8, 9], a signal frame is generally processed as shown in Fig. 1, using three circuits: the column selector, least-square problem (LSP) solver, and residue calculator. The largest and most complicated one is the LSP solver as it performs four sequential operations on matrices: computation, decomposition, inversion, and composition. In the existing implementation, a high-speed solver has been constructed based on a multifunctional systolic array, in which each cell computes four totally distinctive independent functions and stores the value separately inside the cell for each function. This leads to a tremendous overhead in the area. Furthermore, data stored in each cell is required for the next steps, which means that the last iteration of the current frame must be completed before the first iteration of the next frame begins. Therefore, the existing approaches, in which the algorithms has been coarsely partitioned among hardware blocks, is characterized by a limited throughput.

In order to improve performance and save area, we propose a new 8-stages pipelined design of the OMP algorithm as shown in Fig. 2. Stages 0 and 1 in Fig. 2

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Orthogonal matching pursuit (OMP)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input:</td>
<td>reconstruction matrix $\Theta$, measurement vector $y$, sparsity $K$</td>
</tr>
<tr>
<td>Output:</td>
<td>sparse reconstruction $x$</td>
</tr>
<tr>
<td>Procedure:</td>
<td>1. Find the measurement column $i = \arg \max</td>
</tr>
<tr>
<td></td>
<td>2. Update the column set $\Theta_i = [\Theta_{i-1}, \theta_i]$ with initialized $\Theta_0 = \emptyset$.</td>
</tr>
<tr>
<td></td>
<td>3. Solve least-squares (LS) problem $x_i = \arg \min k = y - (\Theta_i \theta_i)^2$. This is to find out the value of non-zero location in $x$.</td>
</tr>
<tr>
<td></td>
<td>4. Compute the residual $r_i = y - \Theta_i x_i$. This is what remains after removing contribution of $\theta_i$.</td>
</tr>
<tr>
<td>Repeat step 1 to step 4 until $i = K$.</td>
<td></td>
</tr>
</tbody>
</table>

Fig. 1. The existing implementation of the OMP algorithm.

Table 1. OMP algorithm
are for matrix multiplications of $\Theta^T r$, whose sizes are $N \times M$ and $M \times 1$, respectively. Each stage contains $N$ multiply-accumulators (MACs). In the first stage, $N$ elements of each column from the $N \times M$ matrix are fed into $N$ MACs to compute dot products (DPs) with the corresponding element from the $M \times 1$ matrix in every cycle. Stage 0 can be completed in $M/2$ cycles by computing the first half of the $N \times M$ matrix. The results from stage 0 are then fed into stage 1 to compute $N$ final DPs. Stage 1 also works in the same manner as stage 0 finishing in $M/2$ cycles. $N$ final DPs are then compared by using a tree comparator in stage 2, which consists of $N - 1$ two-input comparators with $\log_2 N$ layers. Each layer requires 1 cycle; thus, $\log_2 N$ cycles are required to complete stage 2. The LSP is solved in stages 3–6. Stages 3 and 4 are used to compute matrix multiplication $C = \Theta_i^T \Theta_i$, where the size of $\Theta_i$ is $M \times K$. As matrix $C$ is symmetric, only half the number of elements of $C$ need to be computed using $K(K + 1)/2$ MACs. Stage 3 computes the first half of the matrix multiplication and feeds the results into stage 4 with temporary values for final computation. Stages 3 and 4 can be completed in $M/2$ cycles. Stage 5 performs the LDL decomposition of matrix $C$ in $2K - 1$ cycles, in accordance with the following equations:

$$d_{mn} = c_{mn} - \sum_{k=1}^{m-1} l_{mk}^2 d_{kk}, \quad (5)$$

$$l_{mn} = \frac{1}{d_{mn}} \left( c_{mn} - \sum_{k=1}^{n-1} l_{mk} l_{nk} d_{kk} \right), \quad (6)$$

for $0 < m \leq K$, $0 < n < K$, and $m > n$. Stage 6 performs triangular matrix inversion and matrix composition for the LSP solver. Let us denote $A$ as the inverse matrix of $L$, that is, $A = L^{-1}$, then its $(m, n)$th entry can be calculated as

$$a_{mn} = -\sum_{k=n}^{m-1} l_{mk} a_{kn}, \quad (7)$$

in $K - 1$ cycles, and $a_{mn} = 1$ for $m = n$. Then, $C^{-1}$ can be computed as
As \( \mathbf{L} \) and \( \mathbf{D} \) are triangular and diagonal matrices, respectively, their multiplication can be simply computed by the accumulation of DPs of their appropriate elements. Because all the three matrices of \( \mathbf{A}^T \), \( \mathbf{D}^{-1} \), and \( \mathbf{A} \) have the same size of \( K \times K \), the matrix composition subpart takes \( K \) cycles to compute the final result. Thus, stage 6 requires \( 2K \) cycles in total. Stage 7 employs two matrix multipliers and one subtractor to compute the residual \( \mathbf{r} = \mathbf{y} - \mathbf{G}\mathbf{x} \). As each multiplier requires \( K \) cycles to execute the multiplication, the total number of cycles consumed by stage 7 becomes \( 2K \).

### 3.2 Processing elements

The structures of processing elements (PE) used from stage 0 to stage 7 are delineated in Fig. 3. Fig. 3(a) shows PEs used in stages 0, 3, and 7 to multiply matrices. Stages 1 and 4 compute matrix multiplication with accumulated values from stages 0 and 3, respectively, by using PEs shown in Fig. 3(b). PEs in Figs. 3(c) and (d) are used to compute the elements of diagonal matrix \( \mathbf{D} \) and elements on the lower-left half of triangular matrix \( \mathbf{L} \) for stage 5, respectively. As elements that lie on the diagonal line of \( \mathbf{L} \) will be 1 after inversion, only non-diagonal elements of \( \mathbf{L} \) need to be computed in the matrix inversion subpart of stage 6 by using PEs shown in Fig. 3(e). Finally, the matrix composition subpart of stage 6 is performed by PEs shown in Fig. 3(f). The detailed structure of each PE is described in the following subsections.
3.2.1 LSP solver - matrix multiplication

The proposed design exploits the advantage of parallel processing as well as the dependency between PEs in order to accelerate the computational speed of the matrix multiplication. Fig. 4(a) shows the dataflow for matrix multiplication in stages 3 and 4, and Fig. 4(b) shows its detailed architecture. For each cycle, \( K(K+1)/2 \) PEs of each stage fetch corresponding data to compute the matrix multiplication. Two types of PEs for the first-half and the second-half multiplications denoted as PE\(_{fh}\) and PE\(_{sh}\), respectively, are employed to perform the MAC operations. It should be known that the parallel processing with \( K(K+1)/2 \) PEs leads to a fast matrix multiplication.

3.2.2 LSP solver - LDL decomposition

Diagonal matrix \( D \) and triangular matrix \( L \) can be calculated by Eqs. (5) and (6), respectively, where the rightmost summation in Eq. (6) can be expressed in the accumulation form as

\[
smnk = smnk - 1 + \sum_{k'=1}^{k} lmklnk'dkk;
\]

where

\[
smnk = \sum_{k'=1}^{k} lmk'llnk'dkk, \quad sme0 = 0.
\]

Eq. (9) can be realized by the systolic array with PE\(_{tm}\) for triangular matrix and PE\(_{dm}\) for diagonal matrix. Fig. 5(a) shows the dataflow in LDL decomposition, and Figs. 5(c) and (d) show architectures and connections between processing elements. Each PE\(_{tm}\) computes the value of the corresponding triangular element \( l_{mn} \).
using the fetched-in \( c_{in}, s_{in}, \) and \( d_{in}, \) and also computes the value of \( s_{out} \) according to Eq. (9). The inner register after the multiplexer selectively stores either \( l_{in} \) or \( l_{mn}, \) then transmits to the output \( l_{out} \) for the use by the next PE. \( \text{PE}_{\text{dm}} \) first computes the value of diagonal element by Eq. (5) with \( s_{in} \) received from \( \text{PE}_{\text{tm}} \) and then produces \( d_{out} \) and \( d_{out}^{-1}. \) Note that the registers to store \( l_{out}, s_{out}, d_{out}, \) and \( d_{out}^{-1} \) need to be updated properly by the enable signals. Fig. 5(b) shows an example for computing \( l_{41}, l_{42}, l_{43}, \) and \( d_{44} \) in consecutive cycles to demonstrate the dataflow.

### 3.2.3 LSP solver - matrix inversion

Dataflow for the inversion of triangular matrix \( L \) represented as Eq. (7) is shown in Fig. 6(a), and detail architecture and connection are described in Fig. 6(c). The inner register after the subtractor in each \( \text{PE}_{\text{inv}} \) is first initialized by \( l_{pn}, \) which is the same as \( l_{mn} \) from the previous LDL decomposition stage, then each \( \text{PE}_{\text{inv}} \) operates as a multiply-accumulator to compute \( a_{mn}, l_{out} \) and \( a_{out} \) are transferred to the right \( \text{PE}_{\text{inv}} \) and the lower \( \text{PE}_{\text{inv}} \), respectively for the subsequent computations. The inner registers to store \( l_{out} \) and \( a_{out} \) are updated properly by the control signal. Fig. 6(b) illustrates example dataflow between \( \text{PE}_{\text{inv}} \) for the computations of \( a_{41}, a_{42}, \) and \( a_{43}. \)
### 3.3 Time scheduling

Fig. 7 shows the time scheduling of the proposed design where $f$, $i$, and $s$ represent the frame, iteration, and stage indices, respectively. Each stage is designed to be completed in the same number of cycles (denoted as TS in Fig. 7), e.g., in 32 cycles for $M = 64$, $N = 256$, and $K = 16$. Specifically, stage 0 receives frame 0 for the first iteration of the OMP and completes the processing in 32 cycles. While the processed frame 0 is delivered to stage 1, frame 1 starts to be fed into stage 0. Frames 2–7 in order are fed into stage 0 for the first iteration in a similar way, and in the meantime, frame 0 travels through all the eight stages in $32 \times 8$ cycles and goes back to stage 0. Then, stage 0 takes frame 0 for the second iteration, rather than pulling a new frame 8, so that the first 8 frames are reconstructed first and the next 8 frames are reconstructed subsequently. The proposed pipelined systolic...
architecture can dramatically increase the throughput by reconstructing a frame every 32 cycles. On the other hand, in the existing design, a frame cannot be processed before finishing all iterations for the previous frame, so the throughput is much lower than in the proposed design.

### 4 Synthesis results and comparison

The proposed architecture of OMP has been coded in Verilog hardware description language and synthesized using the Synopsys Design Compiler with the CMOS standard cell library [10, 11]. In Table II, we show the synthesis results of the proposed designs and existing designs [8, 9] in terms of the maximum frequency, clock period, number of cycles (NOC) per frame, reconstruction time, area, and area-delay-product (ADP). The second and third column of Table II show the synthesis results with 90 nm CMOS library of the proposed and existing design [8] when the input word length is 16 bits, $M = 64$, $N = 256$, and $K = 16$. The proposed design operates at a maximum frequency of 312 MHz and completes the processing of every 256-sample frame in 32 3.2-nanosecond cycles. The number of iterations is 16 for a 16-sparse signal. Hence, the number of samples processed per second is $256/(32 \times 3.2 \times 10^{-9}) = 156.25 \times 10^6$ whereas the existing design of [8] can process $256/(474 \times 4.15 \times 10^{-9}) = 8.133 \times 10^6$ samples per second. As the input word length is set to 16, the proposed design can support a processing speed of up to $156.25 \times 10^6 \times 16 = 2.5 \times 10^9$ bits per second (bps), which is faster than the speed required for the reconstruction of a 1080p, 30 fps, 4:2:2 YCbCr HD video stream. The proposed design offers nearly 19 times higher throughput than the known approach, whereas the area overhead is
only 1.16 times; therefore, the area-delay product is 16.49 times lower than in the existing VLSI implementation of the OMP.

The proposed design has also been synthesized using different OMP parameters, i.e. \( M = 64, N = 256, \) and \( K = 8 \) and different technology library of 65 nm CMOS cell library \([11]\) in order to be compared with the previous OMP implementation of \([9]\), whose inversion matrix has been implemented by employing Q-R decomposition (QRD) process. Their synthesis results are listed in the fourth and fifth column in Table II. One of the significant drawbacks of \([9]\) is that it is not reconfigurable, i.e. it does not allow for the reconstruction of input signals with different values of sparsity. Additionally, the hardware implementation is only suitable for those signals which have high sparsity and hence, not applicable for practical use. Compared to OMP architecture in \([9]\), our proposed design requires 45.68 times less reconstruction time for each signal frame with only 6.8 times larger in area, which leads to 6.79 times less area-delay product.

### 5 Conclusion

By developing a novel pipelined systolic architecture, we have shown that the OMP can be implemented as a high-throughput VLSI circuit. The fine-grained pipeline design is constructed from stages which consume the same number of clock cycles, so that time and resource consumption are optimized. Additionally, the newly designed PEs which are specialized for matrix computation bring about a significant area reduction of the whole design. In particular, the proposed design, compared with the existing solutions, can offer a 19.2 times faster reconstruction time at the price of a marginal 1.16 times area overhead, also 45.68 times faster speed with 6.8 times area overhead for the reconstruction of 16-sparse and 8-sparse signals, respectively. The proposed design is capable of reconstructing a compressively sensed 1080p, 30 fps HD video stream in real time, therefore, can be used in the real-time CS applications with stringent performance requirements.
Acknowledgments

This work was supported by the IITP funded by the Ministry of Science and ICT (#2014-0-00501) and 2016 Research Fund of Myongji University.