Hindawi Publishing Corporation
VLSI Design
Volume 2014, Article ID 596103, 9 pages
http://dx.doi.org/10.1155/2014/596103

Research Article

Parallel Jacobi EVD Methods on Integrated Circuits

Chi-Chia Sun,[1] Jürgen Götze,[2] and Gene Eu Jan[3]

[1] *Department of Electrical Engineering, National Formosa University, Wunhua Road 64, Huwei 632, Taiwan*
[2] *Information Processing Lab, Technology University of Dortmund, Otto-Hahn-Strase 4, 44221 Dortmund, Germany*
[3] *Institute of Electrical Engineering, National Taipei University, University Road 151, San Shia District, New Taipei City 23741, Taiwan*

Correspondence should be addressed to Chi-Chia Sun; ccsun@nfu.edu.tw

Received 15 January 2014; Revised 22 May 2014; Accepted 26 June 2014; Published 20 July 2014

Academic Editor: Sungjoo Yoo

Design strategies for parallel iterative algorithms are presented. In order to further study different tradeoff strategies in design criteria for integrated circuits, A 10×10 Jacobi Brent-Luk-EVD array with the simplified μ-CORDIC processor is used as an example. The experimental results show that using the μ-CORDIC processor is beneficial for the design criteria as it yields a smaller area, faster overall computation time, and less energy consumption than the regular CORDIC processor. It is worth to notice that the proposed parallel EVD method can be applied to real-time and low-power array signal processing algorithms performing beamforming or DOA estimation.

1. Introduction

We are on the edge of many important developments which will require parallel data and information processing. The transmission systems are using higher and higher frequencies and the carrier frequencies are increasing to 10 GHz and above. Because of the smaller wavelength more antennas can be implemented on a single device leading to massive MIMO systems. Parallel VLSI architectures will be needed in order to provide the required computational power for 10 GHz and above, massive MIMO, and big data processing [1, 2].

In parallel matrix computation at the circuit level, implementing an iterative algorithm on a multiprocessor array results in a tradeoff between the complexity of an iteration step and the number of required iteration steps. Therefore, as long as the algorithm's convergence properties are guaranteed, it is possible to adjust the architecture, which can significantly reduce the complexity with regard to the implementation. Computing the parallel eigenvalue decomposition (EVD) as a preprocessing step to MUSIC or ESPRIT algorithm with Jacobi's iterative method is used as an important example as the convergence of this method is extremely robust to modifications of the processor elements [3–6].

In [7], it was shown that Brent-Luk-EVD architecture with a modified CORDIC for performing the plane rotation of the Jacobi algorithm can be realized in advanced VLSI design. Based on it, a Jacobi EVD array is realized by implementing a scaling-free microrotation CORDIC (μ-CORDIC) processor in this paper, which only performs a predefined number of CORDIC iterations. Therefore, the size of the processor array can be reduced for implementing a large-scale EVD array in parallel VLSI architectures. After that, several modifications of the algorithm/processor are studied and their impact on the design criteria is investigated for different sizes of EVD array (10×10 to 80×80). Finally, a strategy to comply with the design criteria is established, especially in terms of balancing the number of microiterations and the computational complexity. The proposed architecture is ideal for real-time antenna array applications, such as a flying object carrying an antenna array for beamforming or DOA estimation that would require a real-time, low-power, and efficient architecture for EVD, or joint time-delay and frequency estimation using a sensor network.

This paper is organized as follows. Serial and parallel Jacobi methods are described in Section 2. In Section 3, the design issues of the parallel Jacobi EVD array are discussed, leading to the simplification from a regular full CORDIC to the μ-CORDIC processor with an adaptive number of iterations. Section 4 shows the implementation results. Section 5 concludes this paper.

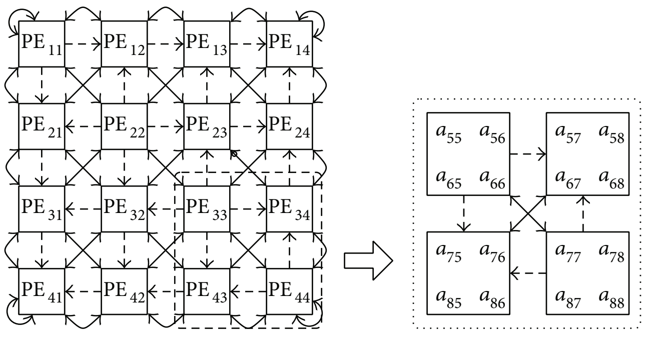


FIGURE 1: A 4×4 Brent-Luk-EVD array, where $n = 8$ for an 8×8 symmetric matrix [3].

2. Parallel Eigenvalue Decomposition

2.1. Jacobi Method. An eigenvalue decomposition of a real symmetric $n \times n$ matrix A is obtained by factorizing A into three matrices $A = Q \wedge Q^T$, where Q is an orthogonal matrix ($QQ^T = I$) and $\wedge$ is a diagonal matrix containing the eigenvalues of A. The Jacobi method approximates the EVD iteratively as follows:

$$A_{k+1} = Q_k A_k Q_k^T, \quad \text{with } k = 0, 1, 2, \ldots, \tag{1}$$

where Q_k is an orthonormal plane rotation by the angle θ in the (i, j) plane.

The plane rotations Q_k, where $k = 1, 2, 3, \ldots$, can be executed in various orders to obtain the eigenvalues. The most common order of sequential plane rotations $\{Q_k\}$ is called cyclic-by-row, meaning (i, j) is chosen as follows:

$$(i, j) = (1, 2)(1, 3) \cdots (1, n)(2, 3) \cdots (2, n) \cdots (n-1, n). \tag{2}$$

The execution of all $N = n(n-1)/2$ index pairs (i, j) is called a sweep. Matrix A will converge into a diagonal matrix $\wedge$ once k sweeps are applied, where $\wedge$ contains the eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$:

$$\lim_{k \to \infty} A_k = \operatorname{diag}[\lambda_1, \lambda_2, \ldots, \lambda_n] = \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & 0 & \lambda_n \end{bmatrix}. \tag{3}$$

2.2. Jacobi EVD Array. Instead of performing the plane rotations Q_k one by one in a cyclic-by-row order, they can be separated into multiple subproblems and executed in parallel on a $\log n$ dimensional multicore platform. Ahmedsaid et al. [3] first presented a parallel array based on Jacobi's method. It consists of $n/2 \times n/2$ PEs and each PE contains a 2×2 subblock of the matrix A. Figure 1 shows a typical 4×4 EVD array with 16 PEs. This Jacobi array can perform $n/2$ subproblems in parallel. Initially, each PE holds a 2×2 submatrix of A:

$$\mathrm{PE}_{pq} = \begin{pmatrix} a_{2p-1,2q-1} & a_{2p-1,2q} \\ a_{2p,2q-1} & a_{2p,2q} \end{pmatrix}, \tag{4}$$

where p and $q = 1, 2, \ldots, n/2$.

A rotation angle has to be chosen in order to zero out the off-diagonal elements of the submatrix by solving a 2×2 symmetric EVD subproblem as shown in the following:

$$\begin{bmatrix} a'_{ii} & a'_{ij} \\ a'_{ji} & a'_{jj} \end{bmatrix} = R \cdot \begin{bmatrix} a_{ii} & a_{ij} \\ a_{ji} & a_{jj} \end{bmatrix} \cdot R^T, \tag{5}$$

where $R = \left[\begin{smallmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{smallmatrix}\right]$.

The maximal reduction $\{a'_{ij}, a'_{ji}\} = 0$ can be obtained by applying the optimal angle of rotation θ_{opt}:

$$\theta_{\mathrm{opt}} = \frac{1}{2} \arctan\left(\frac{2a_{ij}}{a_{jj} - a_{ii}}\right), \tag{6}$$

where the range of θ_{opt} is limited to $|\theta_{\mathrm{opt}}| \le \pi/4$.

This optimal angle θ_{opt}, which can annihilate the off-diagonal elements ($a_{2p-1,2q}$ and $a_{2p,2q-1}$), is computed using diagonal PEs in (6). Once these rotation angles are computed, they will be sent to the off-diagonal PEs. This transmission is indicated by the dashed lines in the vertical and horizontal direction in Figure 1. All off-diagonal PEs will perform a two-sided rotation with the corresponding rotation angles obtained from the row (θ_r) and column (θ_c), respectively.

Once these rotations are applied, the matrix elements are interchanged between processors as indicated by the diagonal solid lines in Figure 1, for the execution of the next $n/2$ rotations. One sweep needs to perform $n - 1$ of these parallel rotation steps. After several sweeps (iterations) are executed, the eigenvalues will concentrate in the diagonal PEs.

3. CORDIC Approach

3.1. Regular CORDIC. Within each PE, a simple way to solve the subproblem of (5) in VLSI for zeroing out the off-diagonal elements is to use the CORDIC algorithm. An orthogonal CORDIC rotator is defined as [8, 9]

$$\begin{aligned} x_{i+1} &= A_i \left[x_i - y_i \cdot d_i \cdot 2^{-i}\right] \\ y_{i+1} &= A_i \left[y_i + x_i \cdot d_i \cdot 2^{-i}\right] \\ z_{i+1} &= z_i - d_i \cdot \tan^{-1} 2^{-i} \\ A_i &= \sqrt{1 + 2^{-2i}} \quad i = 1, 2, 3, \ldots, n \end{aligned} \tag{7}$$

when $n \to \infty$, $A_n \cong 1.647$.

In the Cartesian coordinate system, the CORDIC orthogonal rotation mode can be used to compute (5) by separating the two-sided rotation into two parts, $G = [G_1^T; G_2^T] = A \cdot R^T$ and $R \cdot G$. $A \cdot R^T$ that is computed by

$$G_1 = \left[a_{ii}^r, a_{ij}^r\right]^T = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \cdot \left[a_{ii}, a_{ij}\right]^T,$$
$$G_2 = \left[a_{ji}^r, a_{jj}^r\right]^T = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \cdot \left[a_{ji}, a_{jj}\right]^T, \tag{8}$$

where the plane rotation with the desired rotation angle θ_{opt} is executed using two CORDIC rotators. The CORDIC processors apply n steps, usually $n = 32$ for single floating precision. A constant scaling value $K = 1/A_n = 0.6073$ is subsequently required to fix the rotated vectors $G_1 = [a_{ii}^r, a_{ji}^r]^T$ and $G_2 = [a_{ij}^r, a_{jj}^r]^T$ in order to retain the orthonormality. Similarly, these two CORDIC rotators can also be applied to compute $R \cdot G$:

$$\left[a'_{ii}, a'_{ji}\right]^T = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \cdot \left[a_{ii}^r, a_{ji}^r\right]^T,$$
$$\left[a'_{ij}, a'_{jj}\right]^T = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \cdot \left[a_{ij}^r, a_{jj}^r\right]^T. \tag{9}$$

Meanwhile, the angle θ_{opt} can also be determined by using the CORDIC orthogonal vector mode. The CORDIC rotates the input vector through whatever angle is necessary to align the resulting vector with the x-axis:

$$x_n = A_n \sqrt{x_0^2 + y_0^2}$$
$$y_n = 0 \tag{10}$$
$$z_n = z_0 - d_i \cdot \tan^{-1} 2^{-i}.$$

The CORDIC with an orthogonal vector mode can compute the arctangent result iteratively $\theta = \arctan(y/x)$, if the angle accumulator is initialized with zero ($z_0 = 0$).

In the VLSI design, two common approaches can be used to realize the CORDIC dependence flow graph in hardware: the folded (serial) or the parallel (pipelining) [10, 11]. Note that we limit our efforts to the conventional CORDIC iteration scheme, as given in (7). In Figure 2(a), the structure of a folded CORDIC PE is shown, which requires a pair of adders for plane rotation and another adder for steering the next angle direction (computing the following z_i and d_i). All internal variables are buffered in the registers separately until the iteration number is large enough to obtain the result. The signs of all three intermediate variables are fed into a control unit that generates the rotation direction flags d_i to steer the add or suboperations and keep track of the rotation angle z_i. For example, off-diagonal PE_{43} can directly apply the flags d_i from PE_{33} to (8)'s G_1 and PE_{44} to (8)'s G_2. After the rotation, the required scaling procedure can be obtained using the part of Figure 2(b) that fixes A_n, where two multiplexers are required to select the inputs into the barrel shifters. This folded dependence graph is typical for the orthogonal rotation mode and benefits in a small area in the VLSI design.

In practice, the angle accumulator is not required for the off-diagonal PEs. The d_i from (7) can be used to steer the rotators. Thus, the transmission on the vertical and horizontal dashed lines in Figure 1 will be replaced by a sequence of d_i flags, meaning that the off-diagonal computation efforts for computing the optimal angle θ_{opt} can be omitted.

3.2. Simplified μ-Rotation CORDIC. As the process technologies continue to shrink, it becomes possible to directly implement a Brent-Luk-EVD array with the Jacobi method [12, 13]. However, the size of the EVD array that can be implemented on the current configurable device with the regular CORDIC is still small, say, 4×4. Therefore, we must simplify the architecture in order to integrate more processors. A scaling-free μ-CORDIC for performing the plane rotation in (5) is used [5, 6], where the number of inner iterations is reduced from 32 iterations to only one iteration.

The definition of μ-CORDIC can be developed from (7) as

$$x_{i+1} = \widehat{m}\left[x_i - y_i \cdot d_i \cdot 2^{-i}\right]$$
$$y_{i+1} = \widehat{m}\left[y_i + x_i \cdot d_i \cdot 2^{-i}\right] \tag{11}$$
$$\widehat{m} = \sqrt{\cos^2\theta + \sin^2\theta} = 1 + \epsilon,$$

where $\widehat{m}$ is the required scaling factor per iteration and ϵ is the scaling error. The idea of the μ-CORDIC rotation is to reduce the number of iterations of the full CORDIC to only a few iterations. Meanwhile, the scaling error ϵ will be small enough to be neglected as long as the orthonormality is retained. Figure 3 shows four different methods for different sizes of μ-rotation angles and Table 1 shows a lookup table for the μ-CORDIC, listing 32 approximated rotation angles for each μ-rotation type, the required number of shift-add operations and its computation cycles. Note that the approximated angles are stored as two times of $\tan\theta$. When the rotation angle is very tiny (i.e., ϵ is tiny, too), Type I with only one iteration will comply with the limited working range $1 - 2^{-(n_m+1)} < \widehat{m} < 1 + 2^{-(n_m+1)}$, if the selected n_m ($n_m \in 1 \cdots 32$) is larger than 16. In Figure 3(a), a pair of shift-add operations realizing one iteration step is sufficient. Furthermore, it is scaling free when the angle $2 \times \tan\theta \leq 3.05176 \times 10^{-5}$. These orthonormal μ-rotations are chosen such that they satisfy a predefined accuracy condition in order to approximate the original rotation angles and are constructed by the least computation efforts.

Next, for the Type II rotation (as shown in Figure 3(b)), when n_m is selected from 8 to 15 for small angles, two pairs of shift-add operations are enough to retain the orthonormality. Moreover, when the n_m is selected from 5 to 7, Type III requires three μ-rotations. No scaling is required by Types I through III. Finally, for large rotation angles, the scaling errors cannot be omitted. Figure 3(d) shows the corresponding dependence flow graph for Type IV. Besides the rotation

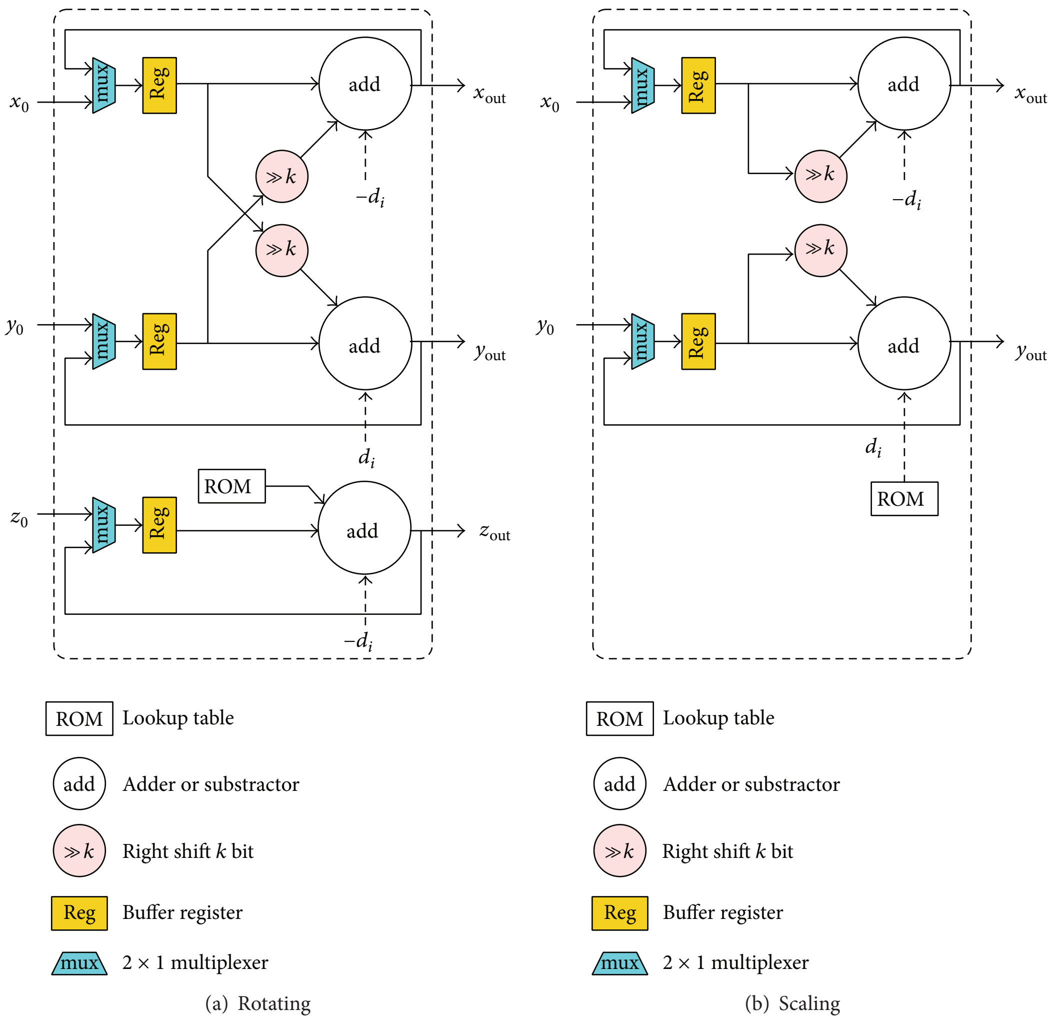


FIGURE 2: Flow graph of a folded CORDIC (recursive) processor.

itself, it requires two pairs of shift-add operations at the beginning of the flow graph, while 2 to 4 pairs of shift-add operations are required to fix the scaling factor $\widehat{m}$:

$$\widehat{m} = \left(1 + 2^{-2(k+1)}\right)\left(1 + 2^{-4(k+1)}\right)\cdots\left(1 + 2^{-2^{M}(k+1)}\right). \quad (12)$$

Note that the scaling costs $M = 2$ to 4 pairs of shift-add operations. In general, the cost of Type IV is bounded by $2 + M$ pairs of shift-add operations. For example, when the index k is 2, the scaling is

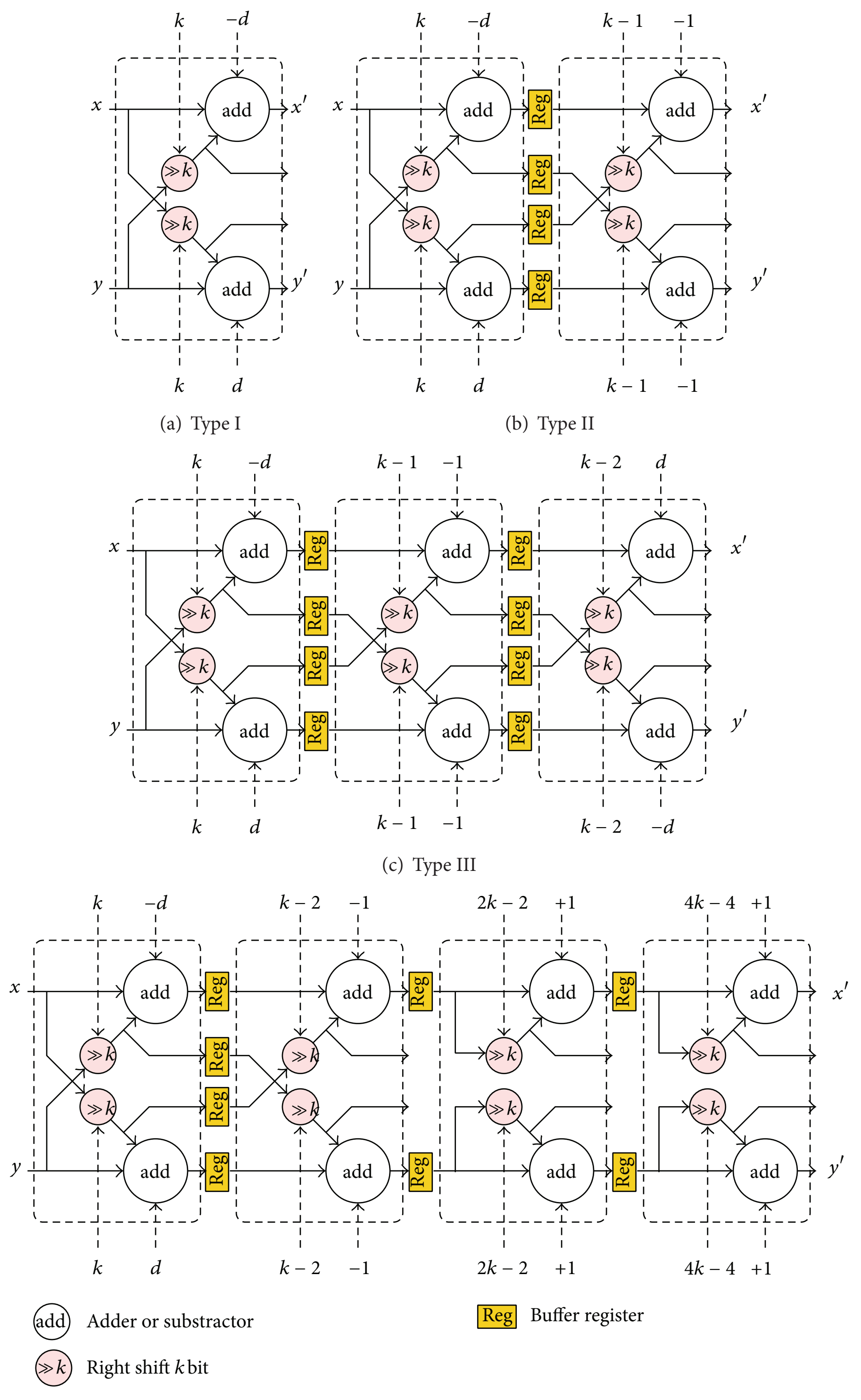
$$\widehat{m}_2 = \left(1 + 2^{-6}\right)\left(1 + 2^{-12}\right)\left(1 + 2^{-24}\right). \quad (13)$$

These four subtypes have three identical parts: Type I with one iteration, the scaling part of Type IV, and the second iteration of Type II. These three parts can be integrated together by using multiplexers to select the data paths, as shown in Figure 4, where 2 adders, 2 shifters, and 4 multiplexers are required [5].

3.3. Adaptive μ-CORDIC Iterations. To improve the computational efficiency, the μ-CORDIC has been modified to perform 6 iterations per cycle as CORDIC-6. As the global clock in a synchronous circuit is determined by the critical path, the maximum timing delay per iteration is 6 cycles (when the index k is 1, Type IV). Therefore, the inner iteration steps of the angles are repeated until they are close to the critical one. The required number of repetitions is quoted in Table 1. For example, when the rotation angle index k is 8, it will repeat three times from the index $k = 8$ to the index $k = 10$; when the rotation angle index k is 20, it is repeated six times from the index $k = 20$ to the index $k = 25$. On the other hand, we can adjust the number of iterations by selecting the average angle during the last sweep and name it as CORDIC-mean.

4. Experimental Results

4.1. Matlab Simulation. The full CORDIC with 32 iteration steps, the μ-CORDIC with one iteration step, and two different adaptive modes have been tested using numerous



(a) Type I (b) Type II

(c) Type III

(d) Type IV

Figure 3: Four μ-CORDIC rotations.

random symmetric matrices A of size 8×8 to 160×160 (i.e., EVD array sizes range from 4×4 to 80×80). Figure 5(a) shows the average number of sweeps needed to compute the eigenvalues/eigenvectors for each size of the EVD array, where the sweep number increases monotonically.

When the Jacobi EVD array size is 10×10, the μ-CORDIC requires 12 sweeps while the full CORDIC only requires 6 sweeps per EVD computation. If we adjust the inner rotations to six times, the sweep number will be 10, smaller than the μ-CORDIC but more than the full CORDIC. Note that using

Table 1: The lookup table for μ-rotations CORDIC with 32-bit accuracy, showing the rotation type, the $2 \times \tan\theta$ angle, the required shift-add operations for rotation and scaling, the required cycle delay, and repeat number for CORDIC-6.

Index	Type	Angle	Shift-add		Cycle	
k		$2 \times \tan\theta$	rot.	sca.	cnt.	re.
1	IV	1.49070	4	8	6	1
2	IV	0.54296	4	6	5	1
3	IV	0.25501	4	6	5	1
4	IV	0.12561	4	4	4	1
5	III	$6.25841\,10^{-2}$	6	0	3	2
6	III	$3.12606\,10^{-2}$	6	0	3	2
7	III	$1.56263\,10^{-2}$	6	0	3	2
8	II	$7.81266\,10^{-3}$	4	0	2	3
9	II	$3.90627\,10^{-3}$	4	0	2	3
10	II	$1.95313\,10^{-3}$	4	0	2	3
11	II	$9.76563\,10^{-4}$	4	0	2	3
12	II	$4.88281\,10^{-4}$	4	0	2	3
13	II	$2.44141\,10^{-4}$	4	0	2	3
14	II	$1.22070\,10^{-4}$	4	0	2	4
15	II	$6.10352\,10^{-5}$	4	0	2	5
16	I	$3.05176\,10^{-5}$	2	0	1	6
17	I	$1.52588\,10^{-5}$	2	0	1	6
18	I	$7.62939\,10^{-6}$	2	0	1	6
19	I	$3.81470\,10^{-6}$	2	0	1	6
20	I	$1.90735\,10^{-6}$	2	0	1	6
21	I	$9.53674\,10^{-7}$	2	0	1	6
22	I	$4.76837\,10^{-7}$	2	0	1	6
23	I	$2.38419\,10^{-7}$	2	0	1	6
24	I	$1.19209\,10^{-7}$	2	0	1	6
25	I	$5.96046\,10^{-8}$	2	0	1	6
26	I	$2.98023\,10^{-8}$	2	0	1	6
27	I	$1.49012\,10^{-8}$	2	0	1	6
28	I	$7.45058\,10^{-9}$	2	0	1	5
29	I	$3.72529\,10^{-9}$	2	0	1	4
30	I	$1.86265\,10^{-9}$	2	0	1	3
31	I	$9.31323\,10^{-10}$	2	0	1	2
32	I	$4.65661\,10^{-10}$	2	0	1	1

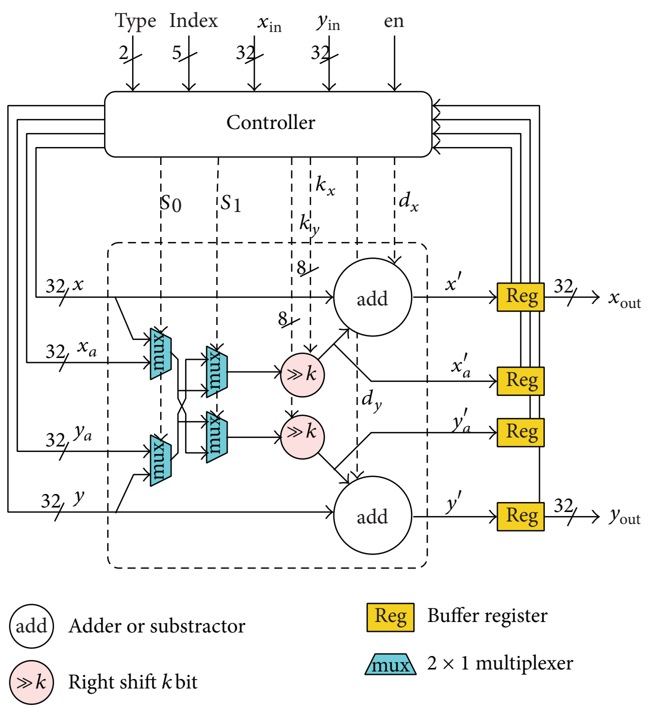


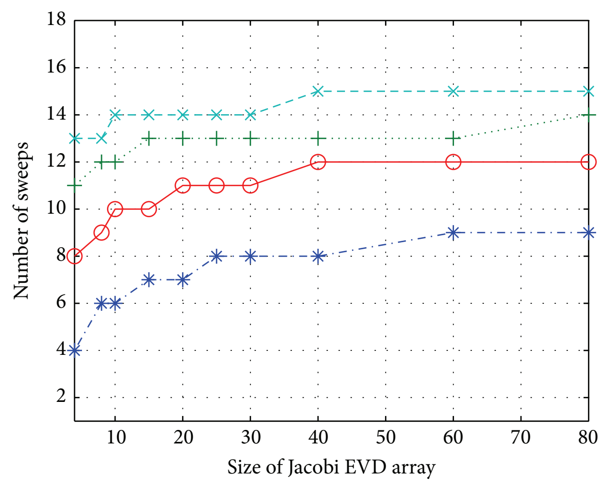
Figure 4: The block diagram of a scaling-free μ-CORDIC PE, including 2 adders, 2 shifters, and 4 multiplexers.

the average rotation angle to decide the rotation number as the CORDIC-mean seems to be an unwise method because it requires more sweeps. Although the μ-CORDIC requires double sweeps than the full CORDIC, it actually reduces the number of the inner CORDIC rotations, which results in improved computational complexity. For example, a 10×10 array with the Full CORDIC PE needs 6 sweeps × 32 inner CORDIC rotations and the CORDIC-6 needs 10 sweeps × 6 inner CORDIC rotations whereas the μ-CORDIC PE requires only 12 sweeps × 1 inner CORDIC rotation. In Figure 5(b), the average number of shift-add operations required for each rotation method for different sizes of EVD arrays is demonstrated whereas μ-CORDIC needs significantly fewer shift-add operations than others. The adaptive CORDIC-6 method can offer a compromise between the hardware complexity and the computational effort.

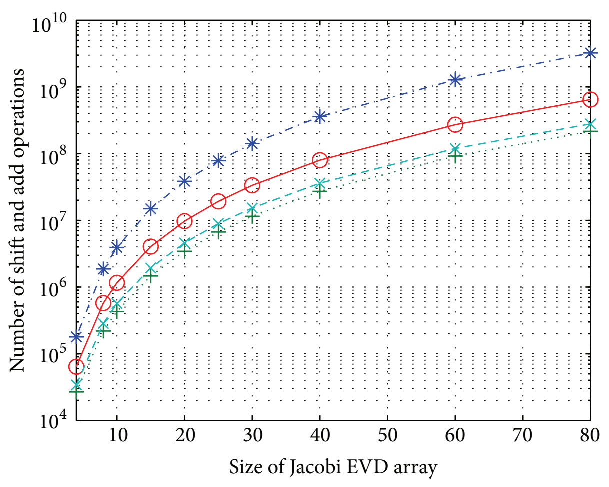
Figure 5(c) shows the off-diagonal Frobenius norm versus the sweep numbers for each array size of 80×80 with double floating precision. Each rotation method converges to the predefined stop criteria: $\|A_{\text{off}}\|_F \times 10^{-8}$. The $\|A_{\text{off}}\|_F$ is the Frobenius norm of the off-diagonal elements of A (i.e., $A_{\text{off}} = A - \text{diag}(\text{diag}(A))$).

Figure 5(d) shows the reduction of the off-diagonal Frobenius norm versus the sweep numbers for single floating precision. It can be noticed that the off-norms do not reach the convergence criteria, and each size of the EVD array has different stop criteria for each rotation method (default IEEE 754 single). Therefore, we can first analyze the Frobenius norm of the off-diagonal elements in Matlab and then observe it until it reaches its maximal reduction. Afterwards, a lookup table can be generated and directly assign these stop criteria to the target hardware circuit or IP component.

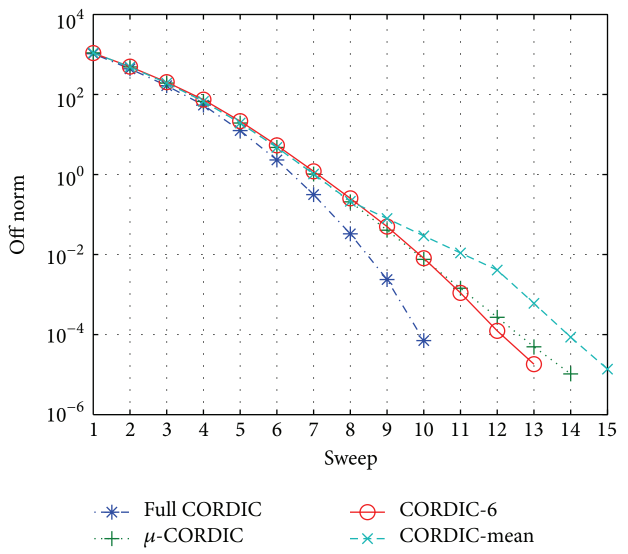
4.2. VLSI Implementation. The μ-CORDIC is modeled and compared to the folded Full CORDIC in VHDL with the resizing feature. These two methods have been integrated into parallel EVD arrays, with sizes 4×4 and 10×10, through a configurable interface separately. After that, they have been synthesized by using the Synopsys Design Compiler with the TSMC 45 nm standard cell library. Note that the word length is 32 bits with the IEEE 754 single floating precision for both CORDIC methods using the same floating point unit from



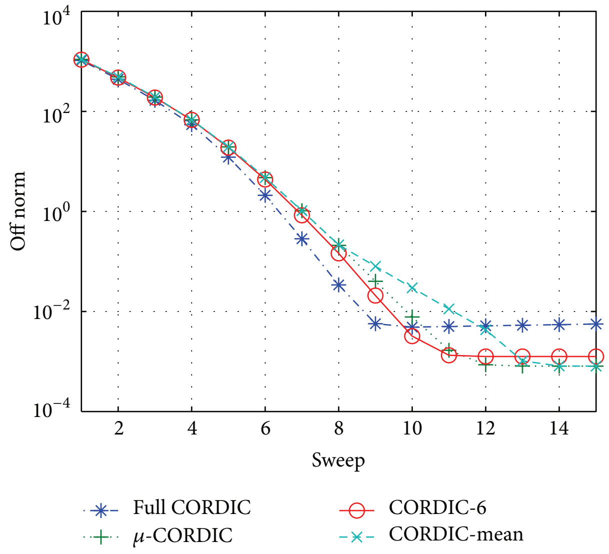
(a) The average number of sweeps versus array sizes



(b) The number of shift-add operations versus array sizes



(c) The required number of sweeps versus array off-diagonal norm (80 × 80 array with double precision)



(d) The required number of sweeps versus array off-diagonal norm (80 × 80 array with single precision)

FIGURE 5: Simulations of four rotation methods.

OpenCORE. Table 2 lists the synthesis results for area, timing delay, and power consumption.

As expected, the combinational logic area and the power consumption of the μ-CORDIC PE are much smaller than the Full CORDIC. Furthermore, in order to determine the time required to compute the EVD of a $n \times n$ symmetric matrix, it can be obtained by

$$T_{\text{total}} = T_{\text{delay}} \times K_{\text{iteration}} \times K_{\text{sweep}} \times [3(n-1) + \Delta + 3], \tag{14}$$

where $n = 8, 16, 20, 30, \ldots, 160$, $\Delta = n/2 - 1$.

The total timing delay per EVD operation is defined by the critical timing delay × the number of inner CORDIC rotations × average number of outer sweeps × size of the matrix A. It can be observed that the total operation time is dependent on the relationship between the inner CORDIC rotations and the outer sweeps. Therefore, one obtains a speedup by a factor of 21.4 by reducing the number of inner CORDIC rotations. Although the reduction of power consumption is less significant due to an extra μ-CORDIC's controller and multiplexers, it actually 6 consumes much less energy per EVD computation due to the shorter computation time. Note that the μ-CORDIC PE requires two inner iterations on average due to the different rotation cycles, from six to one inner iteration, as shown in Table 1. Figure 6 shows the energy consumption for sizes of the array from 4×4 to 80×80. Both rotation methods consume much less energy than the Full CORDIC, where the 6-CORDIC can obtain a factor of 40.9 and the μ-CORDIC can obtain a factor of 104.3 on average for energy reduction compared to the Full CORDIC.

In [14], a Jacobi single cycle-by-row EVD algorithm [15] has been implemented with a single CORDIC processor.

TABLE 2: Comparison of 4 × 4 and 10 × 10 Jacobi EVD arrays.

	PE array size	Full CORDIC 4 × 4	μ-CORDIC 4 × 4	Full CORDIC 10 × 10	μ-CORDIC 10 × 10
Area	Combinational	0.847 mm^2	0.296 mm^2	5.143 mm^2	1.829 mm^2
	Noncombinational	0.390 mm^2	0.123 mm^2	2.306 mm^2	0.833 mm^2
	Total	**1.237** mm^2	**0.419** mm^2	**7.449** mm^2	**2.662** mm^2
Power	Cell	62.283 mW	18.239 mW	388.379 mW	123.215 mW
	Net	0.465 mW	0.433 mW	2.993 mW	2.678 mW
	Leakage	11.909 mW	3.765 mW	86.136 mW	23.966 mW
	Total	**74.657** mW	**22.437** mW	**477.508** mW	**149.859** mW
Timing	Critical	4.454 ns	1.213 ns	4.286 ns	2.247 ns
	Frequency	224.5 MHz	824.4 MHz	233.3 MHz	445 MHz

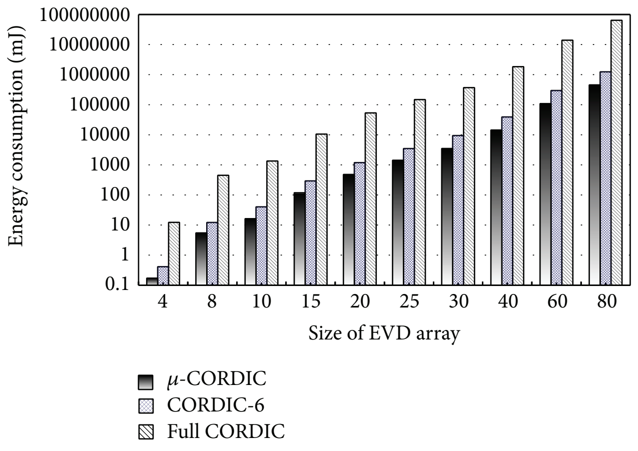


FIGURE 6: The energy consumption per EVD operation with each size of EVD array (operating at 100 MHz).

Since it requires a very complex controller and lookup tables, the throughput is not comparable with a real Brent-Luk's parallel EVD array [13]. In comparison to [13], Full CORDIC for Jacobi Brent-Luk-EVD parallel architecture is implemented in FPGA; however, current configurable device can only perform 4 × 4 EVD array. The experimental results show that performing the unitary rotation in CORDIC processor is a good solution. It required smaller area size, improved the overall computation time, and reduced the energy consumption. Furthermore, the unitary-rotation method can be also applied to other more efficient CORIDC architectures as long as the orthogonality is obtained during CORDIC iterations, such as pipeline CORDIC [16, 17], or implementing the rotators with better adder structures [18, 19].

As the process technologies continue to shrink, it becomes possible to directly implement a Brent-Luk-EVD array with the Jacobi method [12, 13]. However, the size of the EVD array that can be implemented on the current configurable device with the regular CORDIC is still small, say, 4×4. Therefore, we must simplify the architecture in order to integrate more processors. A scaling-free μ-CORDIC for performing the plane rotation in (5) is used [5, 6], where the number of inner iterations is reduced from 32 iterations to only one iteration.

5. Conclusions

The EVD was computed by the parallel Jacobi method, which was selected as an example for a typical iterative algorithm which exhibits very robust convergence properties. A configurable Jacobi EVD array with both Full CORDIC and μ-CORDIC is implemented in order to further study the tradeoff strategies in design criteria for parallel integrated circuits. The experimental results indicate that the presented μ-CORDIC method can reduce the size of the combinational logic, speed up the overall computation time, and improve the energy consumption.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgment

The authors would like to thank the National Science Council of Taiwan for the support under the contract of NSC 101-2218-E-150-001.

References

[1] S. Aggarwal and K. Khare, "CORDIC-based window implementation to minimise area and pipeline depth," *IET Signal Processing*, vol. 7, no. 5, pp. 427–435, 2013.

[2] H. M. Ahmed, J. Delosme, and M. Morf, "Highly concurrent computing structure for matrix arithmetic and signal processing," *IEEE Computer Magazine*, vol. 15, no. 1, pp. 65–82, 1982.

[3] A. Ahmedsaid, A. Amira, and A. Bouridane, "Improved SVD systolic array and implementation on FPGA," in *Proceedings of the IEEE International Conference on Field-Programmable Technology*, pp. 3–42, 2003.

[4] R. Andraka, "Survey of CORDIC algorithms for FPGA based computers," in *Proceedings of the ACM/SIGDA 6th International Symposium on Field Programmable Gate Arrays (FPGA '98)*, pp. 191–200, February 1998.

[5] I. Bravo, P. Jiménez, M. Mazo, J. L. Lázaro, and A. Gardel, "Implementation in FPGAS of Jacobi method to solve the eigenvalue and eigenvector problem," in *Proceedings of the International Conference on Field Programmable Logic and Applications (FPL '06)*, pp. 1–4, Madrid, Spain, August 2006.

[6] R. P. Brent and F. T. Luk, "The solution of singular-value and symmetric eigenvalue problems on multiprocessor arrays," *SIAM Journal on Scientific and Statistical Computing*, vol. 6, no. 1, pp. 69–84, 1985.

[7] G. H. Golub and C. F. van Loan, *Matrix Computations*, Johns Hopkins University Press, Baltimore, Md, USA, 3rd edition, 1996.

[8] J. Götze and G. J. Hekstra, "An algorithm and architecture based on orthonormal μ-rotations for computing the symmetric EVD," *Integration, the VLSI Journal*, vol. 20, no. 1, pp. 21–39, 1995.

[9] J. Götze, S. Paul, and M. Sauer, "An efficient Jacobi-like algorithm for parallel eigenvalue computation," *IEEE Transactions on Computers*, vol. 42, no. 9, pp. 1058–1065, 1993.

[10] A. Hakkarainen, J. Werner, K. R. Dandekar, and M. Valkama, "Widely-linear beamforming and RF impairment suppression in massive antenna arrays," *Journal of Communications and Networks*, vol. 15, no. 4, pp. 383–397, 2013.

[11] S. Klauke and J. Götze, "Low power enhancements for parallel algorithms," in *Proceedings of the IEEE International Symopsium on Circuits and Systems*, pp. 234–237, 2001.

[12] Y. Liu, C.-S. Bouganis, and P. Y. K. Cheung, "Hardware architectures for eigenvalue computation of real symmetric matrices," *IET Computers and Digital Techniques*, vol. 3, no. 1, pp. 72–84, 2009.

[13] P. K. Meher and S. Y. Park, "CORDIC designs for fixed angle of rotation," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 21, no. 2, pp. 217–228, 2013.

[14] K. K. Parhi and T. Nishitani, *Digial Signal Processing for Multimedia Systems*, Marcel Dekker, 1999.

[15] S. Purohit and M. Margala, "Investigating the impact of logic and circuit implementation on full adder performance," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 20, no. 7, pp. 1327–1331, 2012.

[16] B. Ramkumar and H. M. Kittur, "Low-power and area-efficient carry select adder," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 20, no. 2, pp. 371–375, 2012.

[17] F. Rusek, D. Persson, B. K. Lau et al., "Scaling up MIMO: opportunities and challenges with very large arrays," *IEEE Signal Processing Magazine*, vol. 30, no. 1, pp. 40–46, 2013.

[18] C.-C. Sun and J. Götze, "VLSI circuit design concept for parallel iterative algorithms in nanoscale," in *Proceedings of the 9th International Symposium on Communications and Information Technology (ISCIT '09)*, pp. 688–692, Icheon, Republic of Korea, September 2009.

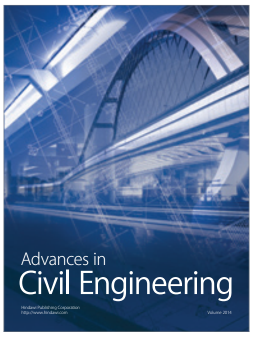
[19] J. S. Walther, "A unified algorithm for elementary functions," in *Proceedings of the Spring Joint Computer Conference*, pp. 379–385, 1971.

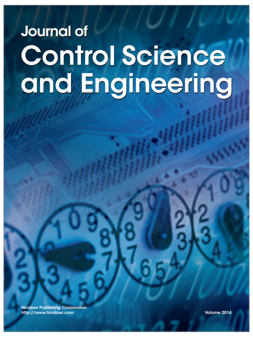
Journal of
Engineering

The Scientific World Journal

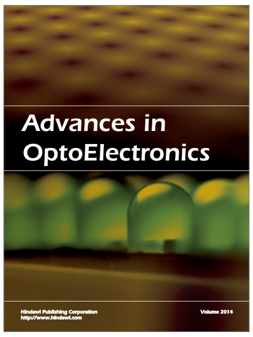
International Journal of
Rotating Machinery

Journal of
Sensors

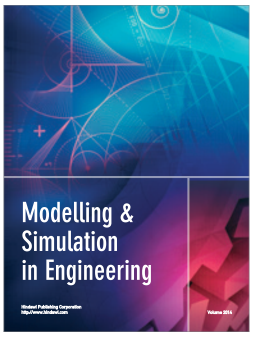
International Journal of
Distributed Sensor Networks

Advances in
Civil Engineering

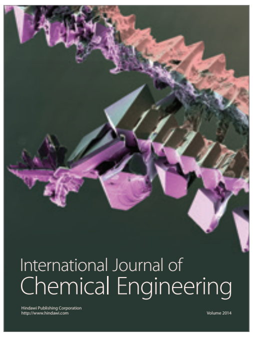
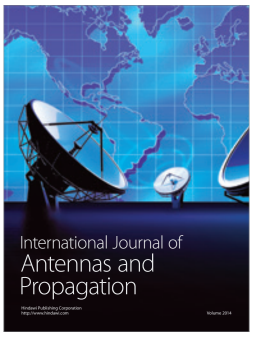

Journal of
Control Science and Engineering

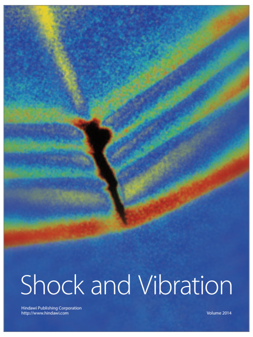
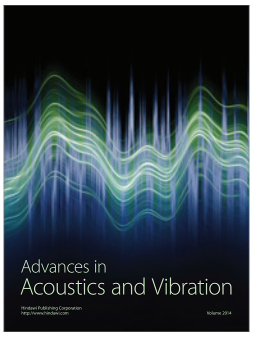
Journal of
Robotics

Journal of
Electrical and Computer Engineering

Advances in
OptoElectronics

VLSI Design

International Journal of
Navigation and Observation

Modelling & Simulation in Engineering

International Journal of
Aerospace Engineering

International Journal of
Chemical Engineering

International Journal of
Antennas and Propagation

Active and Passive
Electronic Components

Shock and Vibration

Advances in
Acoustics and Vibration