VLSI Circuit Design Concept for Parallel Iterative Algorithms in Nanoscale

Chi-Chia Sun*† and Jürgen Götze*
*Dortmund University of Technology, Information Processing Lab,
Otto-Hahn-Str. 4, 44227 Dortmund, Germany.
†Tel: +49-231-7557018, Fax: +49-231-7557019
E-mail: chichia.sun@tu-dortmund.de

***Abstract*—Modern VLSI manufacturing technology has kept shrinking down to the nanoscale level with a very fast trend. Integration with the advanced nano-technology now makes it possible to realize advanced parallel iterative algorithms directly which was almost impossible 10 years ago. In this paper, we want to discuss the influences of evolving VLSI technologies for iterative algorithms and present design strategies from an algorithmic and architectural point of view. We can simplify the parallel implementation of the iterative algorithm (i.e., processor elements of the multiprocessor array) in any way as long as the convergence is guaranteed. However, the modification of the algorithm (processors) usually increases the number of required iterations which also means that the switch activity of interconnects is increasing. We implemented a 3×3 Jacobi EVD array with the μ–CORDIC PE in both 0.18μm and 45nm technologies in order to further study the trade-off between the performance/complexity of processors and the load/throughput of interconnects. Our experimental results show that using the μ–CORDIC PE is beneficial concerning the design criteria since it yields smaller chip area, faster overall computation timing and less energy consumption per operation than the Full CORDIC PE.**

***Index Terms*—EVD, SVD, CORDIC, Iterative Algorithm, Parallel Computing, MPSoC and VLSI Design.**

I. Introduction

Modern VLSI manufacturing technology has kept shrinking down to Deep Sub-Micron (DSM) with a very fast trend and Moore's law is expected to hold for the next 10 years [1], [2]. Now, since the nano-technology allows the integration of an ever-increasing number of IP macro-cells on a single silicon die, parallel multiprocessor platforms have received great attention and have been realized into several state-of-the-art applications (e.g., Dual-Core CPU, MPSoC and parallel processor arrays) [3]–[5]. However, as process technologies continue to shrink, and feature demands continue to increase, more and more challenges are coming up, such as timing delay of the global wire interconnection are increasing explosively in relation to the local processor elements, leakage power is becoming a major factor and bus transmission is a bottleneck in the million transistors SoC designs [6]–[8].

These challenges bring us to analyze their impact on parallel iterative algorithms as VLSI technology keeps evolving. As long as the convergence properties of iterative algorithms are guaranteed, it is possible to modify/simplify the architecture which is usually realizing one iteration step. This reduces the complexity with regard to the implementation significantly. However, this simplification will usually cause an increased number of iterations for convergence, i.e. the processor array has to execute its operation more often which means that the number of data transfer in the interconnections also increases. Therefore, it actually becomes a trade-off problem between the performance/complexity of the hardware, the load/throughput of interconnects and the overall energy/power consumption due to the behavior of iterative algorithms.

Computing the Eigenvalue Decomposition (EVD) with the parallel Jacobi method is selected as an example since the convergence of this method is very robust to modification of the processor elements [9]–[12]. In [13], we have presented a VLSI design concept for parallel iterative algorithms by implementing a μ–rotation CORDIC (μ–CORDIC) processor, which only performs a predefined number of CORDIC iterations (e.g., only one μ–rotation). Therefore the size of the processor is reduced (simplified rotation), such that a larger size of EVD array can be implemented in a single device. Here, in order to further study the design impacts in DSM level, a 3×3 EVD array (i.e., for computing the EVD of a 6×6 symmetric matrix) was implemented in both 0.18μm and 45nm technologies. According to our experimental results, we present a strategy to comply with the design criteria, especially concerning the number of iterations, the timing and the energy consumption per EVD operation.

This paper is organized as follows: In Section II we clarify the definition of the serial and parallel Jacobi methods, respectively. In Section III the design issues of the Jacobi EVD array are discussed, which lead to the proposed μ–CORDIC processor. Section IV shows the Matlab experiments and the ASIC implementation results. Section V concludes this paper.

II. Parallel Eigenvalue Decomposition

An eigenvalue decomposition of a real symmetric n×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 which contains the eigenvalues of A. The Jacobi method iteratively computes the EVD as follows:

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

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

$$Q_k = \begin{array}{c} \begin{matrix} & & col\,i & & col\,j & & \\ & & \downarrow & & \downarrow & & \end{matrix} \\ \begin{bmatrix} 1 & 0 & & \cdots & & 0 \\ 0 & \ddots & & & & \\ & & \cos\theta_k & & \sin\theta_k & & \\ \vdots & & & \ddots & & \vdots \\ & & -\sin\theta_k & & \cos\theta_k & & \\ & & & & & \ddots & 0 \\ 0 & & & \cdots & & & 1 \end{bmatrix} \end{array} \begin{matrix} \\ \\ \leftarrow row\,i \\ \\ \leftarrow row\,j \\ \\ \\ \end{matrix} \tag{2}$$

The plane rotations Q_k, where $k = 1, 2, 3, \cdots, \infty$, can be executed in various orders in order to obtain the eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$ in a diagonal matrix $\wedge$ as shown in Eq. 3:

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

Instead of performing the plane rotation Q_k one by one in a cyclic–by–row order, we can separate it into multi-subproblems and execute them in parallel on a processor array.

A. Jacobi EVD Array

A parallel array for Jacobi's method was presented by Brent and Luk [9], it consists of $\frac{n}{2} \times \frac{n}{2}$ PEs and each PE contains a 2×2 sub-block of the matrix A. Fig. 1 shows a typical 4×4 EVD array with 16 PEs. This Jacobi array can perform $\frac{n}{2}$ subproblems in parallel. Initially each PE holds a 2×2 sub-matrix of A:

$$PE_{pq} = \begin{pmatrix} a_{2p-1,2q-1} & a_{2p-1,2q} \\ a_{2p,2q-1} & a_{2p,2q} \end{pmatrix}, \quad \text{where } p \text{ and } q = 1, 2, \cdots, \frac{n}{2}. \tag{4}$$

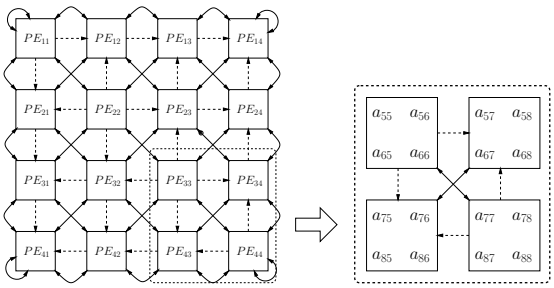

Fig. 1. A 4×4 EVD array, n=8.

We have to choose a rotation angle in order to decrease the off-diagonal elements of the submatrix by solving a 2×2 symmetric EVD subproblem as shown in the following:

$$\begin{bmatrix} a'_{ii} & a'_{ij} \\ a'_{ji} & a'_{jj} \end{bmatrix} = \mathrm{R} \cdot \begin{bmatrix} a_{ii} & a_{ij} \\ a_{ji} & a_{jj} \end{bmatrix} \cdot \mathrm{R}^T, \quad \text{where R} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}. \tag{5}$$

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

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

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

This optimal angel θ_{opt}, which is able to annihilate the off-diagonal elements ($a_{2p-1,2q}$ and $a_{2p,2q-1}$), is computed by diagonal PEs (i.e., PE_{11}, PE_{22}, PE_{33} and PE_{44}) using Eq. 6. After these rotation angles are computed, they will be sent to the off-diagonal PEs. This transmission is indicated by the dashed lines in vertical and horizontal direction in Fig. 1. All off–diagonal PEs will perform a two-sided rotation with the corresponding rotation angles which are obtained from the row (θ_r) and column (θ_c) respectively.

After these rotations are applied, the matrix elements are interchanged between processors as indicated by the solid lines in diagonal direction in Fig. 1 for execution of the next $\frac{n}{2}$ 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. It should be noticed that, in practice we can observe the Frobenius norm of the off-diagonal elements until it is close to zero using results of [12] or perform a predefined number of sweeps which depends on the order of matrix A.

III. Simplified Architecture

As the process technologies continue to shrink to the nanoscale level, it is possible to implement a full Jacobi EVD array into one single device [14], [15]. In the past we have presented a scaling–free μ–CORDIC method to replace the original Full CORDIC method for performing the plane rotation in Eq. 5 by reducing inner iteration numbers from sixteen to only one single step [11], [12]. The benefits we can get from the μ–CORDIC are that larger EVD array can be synthesized into a limited size of FPGA device and faster computational timing compared to the Full CORDIC can be obtained [13]. Here, we will implement this μ–CORDIC method as an ASIC and verify the real performance under different design criteria.

At first we have slightly modified the μ–CORDIC PE which was presented in [11] as shown in Fig. 2. It is able to perform the single inner iteration efficiently. This μ–CORDIC PE has 2 adders, 2 shifters and 4 multiplexers, and it reduces the number of inner iterations, which is 32 for a Full CORDIC (word-length W=32bits) to only one inner iteration. However, decreasing the inner iterations will cause an increased number

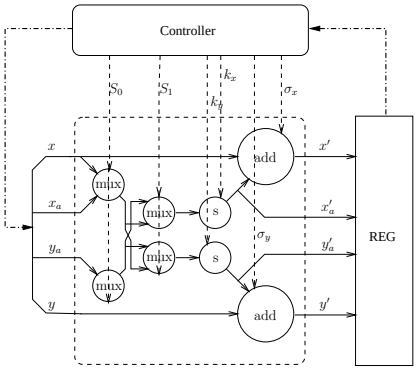

Fig. 2. The block diagram of a scaling–free μ–CORDIC PE, including 2 adders, 2 shifters and 4 multiplexers.

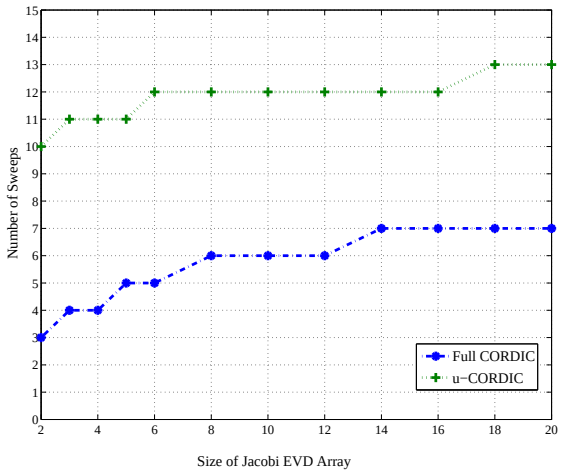

Fig. 3. The required number of sweeps vs. array sizes for μ–CORDIC and Full CORDIC rotation methods.

of outer sweeps because of the imprecise inner iterations. Therefore, the μ–CORDIC architecture can reduce the area but requires more sweeps. On the other hand, the Full CORDIC architecture needs fewer sweeps but requires more area.

Table I shows a Look–Up Table (LUT) for the presented μ–CORDIC PE. This LUT table lists 32 approximated angles for scaling–free μ–rotation in four types, the required number of shift–add operations and its computation cycles. More detailed information about the μ–CORDIC can be found in [11], [13].

IV. Experimental Results

In this paper, we have first simulated the μ–CORDIC and the Full CORDIC for a parallel Jacobi EVD method in Matlab and implemented four ASIC designs for further detailed comparison.

A. Matlab Simulation

We have used the Full CORDIC with 32 iteration steps for the μ–CORDIC with one iteration step by numerous random symmetric matrices A from the size 4×4 to 40×40 (i.e., EVD array size from 2×2 to 20×20.). Fig. 3 shows the average number of sweeps needed to compute the eigenvalues for each size of EVD array.

When the Jacobi EVD array's size is 10×10, the μ–CORDIC requires 12 sweeps while the Full CORDIC only requires 6 sweeps. Although the μ–CORDIC requires more sweeps per EVD computation than the Full CORDIC, it actually reduces the number of the inner CORDIC rotations which overall results in an improved computational complexity. For example, a 10×10 array with the Full CORDIC PE needs 6 sweeps × 32 inner CORDIC rotations, while the μ–CORDIC PE only requires 12 sweeps × 1 inner CORDIC rotation. Consequently, from an algorithmic point of view, there is no doubt that we would rather realize the Jacobi method by utilizing the μ–CORDIC method. However, when it comes to ASIC implementation, other criteria become also important.

B. ASIC Implementation

We have modeled the presented μ–CORDIC PE in Verilog HDL and compared it with a Full CORDIC PE which is

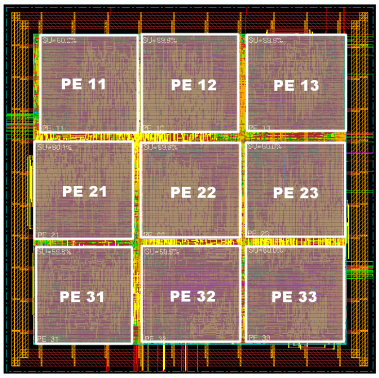

Fig. 4. Final layout view of a 3×3 Jacobi EVD array with the μ–CORDIC PE in FreePDK 45nm library.

available in open source [16]. Later, we implemented these two methods into a 3×3 EVD array (6×6 matrix) separately and synthesized them by Design Compiler with TSMC 0.18μm and FreePDK 45nm cell libraries [7]. It should be noticed that the word-length is 32 bits. At the end, we used the SoC Encounter to perform Place & Route of these two designs and estimated the power consumption by PrimeTime PX with thousand random patterns. The final version of the chip ready for fabrication with 9 μ–CORDIC PEs in 45nm technology is shown in Fig. 4. It should be noticed that since the FreePDK library did not provide any IO pads, we also did not add IO pads to the TSMC 0.18μm's implementation and treated it as a macro design. Table II lists the final layout results for area, timing delay and power consumption. There are some important points that can be observed.

First of all, the combinational logic area and the core size of the μ–CORDIC PE is less than the Full CORDIC as expected. Furthermore, in order to determine the time required to compute the EVD of a 6×6 symmetric matrix, we obtain:

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

TABLE I
THE LOOK–UP TABLE FOR μ–ROTATIONS WITH 32-BIT ACCURACY, SHOWING THE ROTATION TYPE USED, THE $2\times \tan\theta$ ANGLE, THE SHIFT–ADD OPERATION COST OF ROTATION AND SCALING, THE REQUIRED CYCLE DELAY [13].

			TSMC 0.18μm	FreePDK 45nm
μ–CORDIC PE	Area	Cell	1.796 mm^2	0.324 mm^2
		Net	0.804 mm^2	0.070 mm^2
		Core	1.797 mm^2	0.339 mm^2
	Power	Cell	98.0 mW	52.4 mW
		Net	45.8 mW	27.5 mW
		Leakage	0.0053 mW	1.2 mW
		Total	143.8 mW	81.1 mW
	Timing	Critical	12.314 ns	5.012 ns
		Frequency	72 MHz	166 MHz
Full CORDIC PE	Area	Cell	2.199 mm^2	0.370 mm^2
		Net	0.899 mm^2	0.071 mm^2
		Core	2.199 mm^2	0.393 mm^2
	Power	Cell	95.0 mW	31.9 mW
		Net	44.2 mW	21.6 mW
		Leakage	0.0069 mW	1.4 mW
		Total	139.2 mW	54.9 mW
	Timing	Critical	12.111 ns	6.132 ns
		Frequency	72 MHz	125 MHz

TABLE II
AREA, DELAY AND POWER CONSUMPTION RESULTS OF A 3×3 JACOBI EVD ARRAY WITH TSMC 0.18μM AND FREEPDK 45NM TECHNOLOGY LIBRARIES.

$$\mathrm{T}_{total} = \mathrm{T}_{delay} \times \mathrm{K}_{iteration} \times \mathrm{K}_{sweep} \times (\mathrm{n}-1). \quad (7)$$

The total timing delay per EVD operation is defined as " the critical timing delay $\times$ the number of inner CORDIC rotations $\times$ average number of outer sweeps $\times$ size of the matrix". We can observe that the total operation timing is dependent upon the relation between the inner CORDIC rotations and the outer sweeps. For instance, computing the EVD on the 3×3 EVD array with Full CORDIC PE in 45nm technology, requires 6.132×32×4×(6-1)≅3924.5ns. Using the μ–CORDIC PE only requires 5.012×2×11×(6-1)≅551.3ns. Therefore, we can obtain a benefit for speed up by a factor of 7.1 from reducing the number of inner CORDIC rotations. Although the power consumption is a little higher due to the faster clock frequency, it actually consumes much less energy per EVD computation because of the shorter computational timing. It should be noticed that the μ–CORDIC PE requires two inner iterations in average due to the different rotation cycles from six to one inner iterations as shown in Table I. Fig. 5 shows the energy consumption results, where the clock cycle for 45nm is from 6ns to 20ns and for 0.18μm is from 14ns to 20ns. For instance, when the clock cycle is 14 ns, the μ–CORDIC PE reduces the energy consumption by a factor of

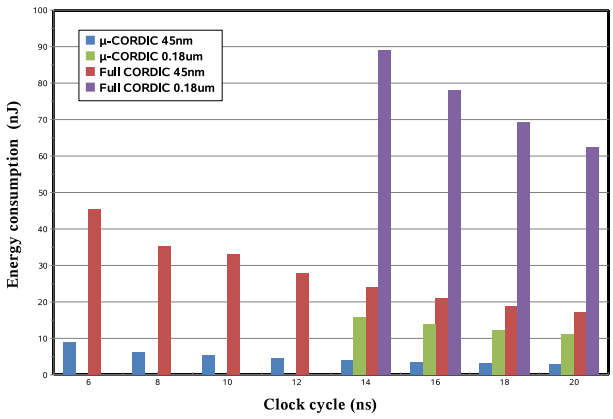

Fig. 5. The energy consumption per 3×3 EVD operation with different clock cycles.

6.1 in 45nm and by a factor of 5.6 in 0.18μm, respectively. In short, the μ–CORDIC can not only achieve less combinational logic cell area and faster frequency, but also lower power consumption compared to the Full CORDIC.

V. Conclusions

For iterative algorithms we are able to simplify/modify the PEs as long as the convergence is guaranteed. This is paid for by an increased number of iterations. Computing the EVD by the parallel Jacobi algorithm was used as an example. We implemented a 3×3 Jacobi EVD array as an ASIC with the Full CORDIC and the presented μ–CORDIC PE to explain our design concept for parallel iterative algorithms. The experimental results showed that the presented μ–CORDIC method can reduce the size of the combinational logic, speed up the overall computational timing and improve the energy consumption due to the less inner iteration steps.

In future work, instead of performing only one iteration, we will try to repeat it to k iterations per cycle and name it as CORDIC-k ($k = 1, 2, 3, \cdots$,word-length), which may offer a compromised solution between the number of sweeps (iterations) and computational complexity. On the other hand, we will extend the array from the size of 3×3 to larger arrays by parameterizing the IP Core. Later, a detailed comparison between area, timing delay, power and energy consumption for an EVD application (e.g., subspace based frequency estimation or DOA estimation) with different parallel multiprocessor platforms should also be performed and further discussed. Furthermore, the concept of exchanging inner and outer iterations in parallel iterative algorithms should be investigated for other iterative algorithms and their applications.

References

[1] P. Gelsinger, "Moore's Law: "We See No End in Sight,"," Intel Chief Technology Officer, Tech. Rep., May 2008. [Online]. Available: http://websphere.sys-con.com/node/557154

[2] J. R. Powell, "The quantum limit to moore's law," in *Proceedings of the IEEE*, vol. 96, no. 8, Aug 2008.

[3] S. Vangal, J. Howard, G. Ruhl, S. Dighe, H. Wilson, J. Tschanz, D. Finan, P. Iyer, A. Singh, T. Jacob, S. Jain, S. Venkataraman, Y. Hoskote, and N. Borkar, "An 80-Tile 1.28TFLOPS Network-on-Chip in 65nm CMOS," *Solid-State Circuits Conference, 2007. ISSCC 2007. Digest of Technical Papers. IEEE International*, pp. 98–589, Feb. 2007.

[4] W. Wolf, "The future of multiprocessor systems-on-chips," in *Annual ACM IEEE Design Automation Conference*, June 2004, pp. 681–685.

[5] F. Vitullo, N. E. L'Insalata, E. Petri, S. Saponara, L. Fanucci, M. Casula, R. Locatelli, and M. Coppola, "Low-Complexity Link Microarchitecture for Mesochronous Communication in Networks-on-Chip," *IEEE Transactions on Computer*, vol. 57, no. 9, pp. 1196–1201, Sep. 2008.

[6] K. S. Sainarayanan, C. Raghunandan, and M. Srinivas, "Delay and Power Minimization in VLSI Interconnects with Spatio-Temporal Bus-Encoding Scheme," in *IEEE Computer Society Annual Symposium on VLSI*, March 2007, pp. 401–408.

[7] J. E. Stine, I. Castellanos, M. Wood, J. Henson, F. Love, W. R. Davis, P. D. Franzon, M. Bucher, and S. Basavarajaiah, "FreePDK: An Open-Source Variation-Aware Design Kit," in *IEEE International Conference on Microelectronic Systems Education*, 2007, pp. 173–174.

[8] T. Williams, "The future is low power and test," in *13th European Test*. Synopsys Inc., Boulder, CO, May 2008, p. 4.

[9] 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, January 1985.

[10] 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.

[11] J. Götze and G. Hekstra, "An Algorithm and Architecture Based on Orthonormal Micro-Rotations for Computing the Symmetric EVD," *INTEGRATION - The VLSI Journal*, vol. 20, pp. 21–39, 1995.

[12] S. Klauke and J. Götze, "Low Power Enhancements for Parallel Algorithms," in *IEEE International Sympopsium on Circuits and Systems*, 2001.

[13] C.-C. Sun and J. Götze, "A VLSI Design Concept for Parallel Iterative Algorithms," in *Kleinheubacher Tagung*, Sep. 2008, to be published in Advances in Radio Science.

[14] A. Ahmedsaid, A. Amira, and A. Bouridane, "Improved SVD systolic array and implementation on FPGA," in *IEEE International Conference on Field-Programmable Technology (FPT)*, Dec 2003, pp. 3–42.

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

[16] "The OPENCORES Website," www.opencores.org, 2008.