Adv. Radio Sci., 7, 95–100, 2009
www.adv-radio-sci.net/7/95/2009/

Advances in Radio Science

A VLSI design concept for parallel iterative algorithms

C. C. Sun and J. Götze

Dortmund University of Technology, Information Processing Lab, Otto-Hahn-Str. 4, 44227 Dortmund, Germany

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. Implementing an iterative algorithm on a multiprocessor array, there is a trade-off between the performance/complexity of processors and the load/throughput of interconnects. This is due to the behavior of iterative algorithms. For example, we could 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. As an example we show that a 25×25 full Jacobi EVD array could be realized into one single FPGA device with the simplified μ-rotation CORDIC architecture.

1 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 (Gelsinger, 2008). Now, since the DSM 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 Computing) (Vangal et al., 2007; Wolf, 2004; Vitullo et al., 2008).

10 years ago, for 0.35 μm technology, design engineers were focusing on reducing the area size. Later, when it came to 0.13 μm technology they paid huge efforts to improve the signal delay and reduce the power consumption. As the VLSI manufacturing technology keeps shrinking down into 65 nm, the design methodology for nano-circuits poses new challenges: area requirements of the wire interconnections are increasing explosively in relation to the area of processor elements, bus transmission bottleneck in the million transistors SoC designs, and leakage current is now dominating the power consumption (Sainarayanan et al., 2007; Stine et al., 2007).

These changes bring us to analyze the impacts on parallel iterative algorithms as VLSI technology keeps evolving. As long as the convergence properties of the iterative algorithms are guaranteed, it is possible to modify/simplify the architecture during the iteration steps and reduce the computational complexity significantly with regard to the implementation. However, this simplification will usually cause an increased number of iterations for convergence. 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 parallel iterative algorithms.

Computing the Eigenvalue Decomposition (EVD) with the parallel Jacobi method is used as an example since the convergence of this methodology is very robust to modification of the processor elements. Finally, a VLSI design concept for parallel iterative algorithms is presented which takes into account the influence of the modifications on area, timing delay and power consumption.

The paper is organized as follows: in Sect. 2 we will first describe the design concepts for parallel iterative algorithms. After that, we will further clarify the definition of the serial and parallel Jacobi method, respectively, in Sect. 3. Then, in Sect. 4 the design issues of the Jacobi EVD array and their suitability for different hardware implementations are discussed, which lead to the simplified μ-rotation CORDIC processor. Section 5 shows the experimental and syntheses results. Section 6 concludes this paper.

Correspondence to: C. C. Sun
(chichia.sun@tu-dortmund.de)

Published by Copernicus Publications on behalf of the URSI Landesausschuss in der Bundesrepublik Deutschland e.V.

2 Design concept and implementation issues

A design concept for parallel iterative algorithms, is presented taking into consideration the influences of different VLSI technologies in terms of area, power and timing delay. Implementing an iterative algorithm on a multiprocessor array, there is a trade-off between the complexity of an iteration step (assuming that the convergence of the algorithm is retained) and the number of required iteration steps. For example, suppose we have a hardware platform, which requires an iteration step of the iterative algorithm to be executed K times in order to obtain the convergence. The iteration step is executed in parallel on the platform. If we simplify the processors in order to improve the logical utilization of the platform, the number of required iterations usually increase from K to $K+L$. It also means that the switch activity of interconnects between these processor elements is increasing due to the behavior of iterative algorithm. How to find a superior solution to balance the design criteria is the major issue of this paper, especially for low-power or limited-area devices.

In this paper, we selected the Jacobi EVD method as a typical iterative algorithm since the convergence of this methodology is very robust to modification of the processor elements (Brent and Luk, 1985; Gotze et al., 1993; Goetze and Hekstra, 1995; Klauke and Goetze, 2001). We have investigated the influences in DSM design with different sizes of multiprocessor arrays (i.e., 4×4, 16×16 and 25×25). After that, several modifications of the algorithm/processor were studied and their impacts on different FPGA devices were investigated (e.g., Xilinx Virtex series in 0.22 μm, 0.15 μm and 65 nm). According to these analyses, we present an efficient strategy to comply with the design criteria, especially in balancing the number of iterations and the computational complexity.

3 Eigenvalue decomposition

An Eigenvalue decomposition of a real symmertric $n \times n$ 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 which contains the eigenvalues of **A**.

3.1 Jacobi method

The cyclic-by-row Jacobi method computes the EVD of a $n \times n$ symmetric matrix iteratively by applying a sequence of orthonormal rotations to the left and the right of the matrix **A**, as shown in the following:

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

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

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

The order of sequential plan rotations $\{Q_k\}$ is called cyclic-by-row manner, if (i, j) is chosen as follows:

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

The execution of all $N = n(n-1)/2$ index pairs (i, j) is called a sweep. After several sweeps are applied, the matrix **A** will converge into a diagonal matrix $\wedge$, which contains the eigenvalues:

$$\lim_{k \to \infty} A_k = diag[\lambda_1, \lambda_2, \ldots, \lambda_n] = \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & 0 & \lambda_n \end{bmatrix}. \tag{4}$$

In practice we can observe the Frobenius norm of the off-diagonal elements until it is close to zero or perform a predefined number of sweeps which depends on the size of matrix **A**.

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

$$\begin{bmatrix} a'_{ii} & a'_{ij} \\ a'_{ji} & a'_{jj} \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} a_{ii} & a_{ij} \\ a_{ji} & a_{jj} \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}^T. \tag{5}$$

We can solve the subproblem and cause the maximal reduction $\{a_{i,j}, a_{j,i}\} = 0$ by applying an optimal angle of rotation θ_{opt}:

$$\theta_{\text{opt}} = \frac{1}{2} \arctan(\tau)\,, \tag{6}$$

where $\tau = \frac{2a_{ij}}{a_{jj} - a_{ii}}$, and the range of θ_{opt} is limited to $|\theta_{\text{opt}}| \leq \frac{\pi}{4}$.

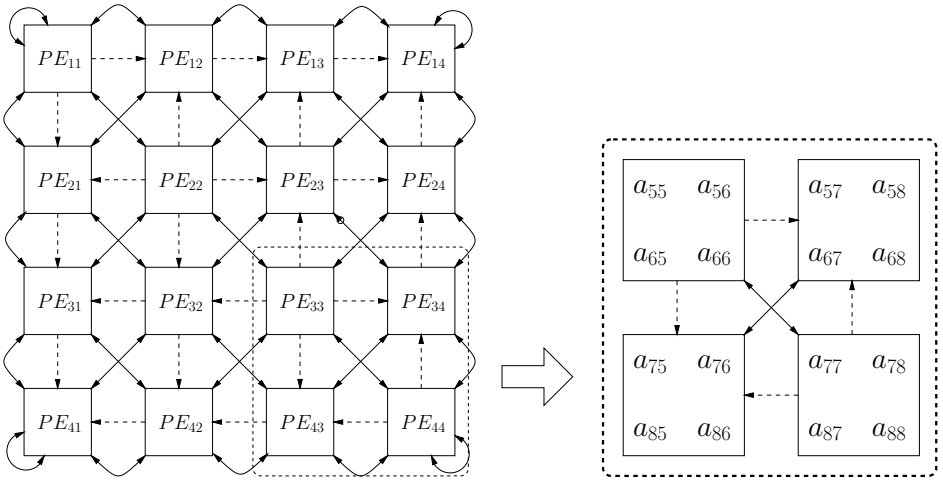

Fig. 1. A 4×4 EVD array, n=8.

3.2 Parallel Jacobi EVD array

The parallel array presented by Brent and Luk consists of $\frac{n}{2}\times\frac{n}{2}$ Processor Elements (PEs) and each PE contains a 2×2 sub-block of the matrix to be decomposed (Brent and Luk, 1985). Figure 1 shows a typical 4×4 EVD array with 16 PEs. This systolic Jacobi array can perform $\frac{n}{2}$ subproblem in parallel and each sweep requires $n-1$ steps. Initially each PE holds a 2×2 sub-matrix of A:

$$PE_{pq} = \begin{pmatrix} a_{2p-1,2q-1} & a_{2p-1,2q} \\ a_{2p,2q-1} & a_{2p,2q} \end{pmatrix}, \quad \text{where } p \text{ and } q = 1, 2, \cdots, \tfrac{n}{2}. \tag{7}$$

The 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 Fig. 1. All PEs will perform a two-sided rotation with the corresponding row (θ_r) and column (θ_c) rotation angles.

$$PE'_{pq} = Q(\theta_{\text{r}}) \cdot PE_{pq} \cdot Q(\theta_{\text{c}})^{\text{T}}, \quad \text{where } Q(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}. \tag{8}$$

One sweep needs to perform $n-1$ parallel rotation steps. After these rotations are applied, the local matrices are interchanged between processors as indicated by the solid lines in Fig. 1 for execution of the next sweep. We can use the CORDIC processor to realize the BLV EVD array (Walther, 1971; Volder, 1959; Parhi and Nishitani, 1999). It should be noticed that since we selected the CORDIC processor to approximate the rotation, we can transmit the $\tan\theta_{\text{opt}}$ directly instead of the angles (see Sect. 4). In this way, we can improve efficiency of the communication bus and make this systolic array more suitable for VLSI implementation.

4 Architecture considerations

In this section we will show the reasons why it is necessary to simplify the CORDIC architecture and how to achieve this goal. As the evaluation of the VLSI technology keeps shrinking down to the nanoscale, it is possible to implement a full Jacobi EVD array into one single FPGA device (Ahmedsaid et al., 2003). However if we still use the original full CORDIC processor which is generated by the Xilinx IP-CORE library (www.xilinx.com), only moderate parallelism can be obtained due to the limited FPGA configuration resources. For example, we could only realize a 6×6 multicore array at most in the biggest Xilinx FPGA device as shown in Table 2. Therefore, we must simplify the CORDIC architecture in order to fit the design criteria.

At first we have slightly modified a simplified scaling free μ-rotation CORDIC which was presented in Goetze and Hekstra (1995) as shown in Fig. 2. It is able to perform the single inner iteration efficiently. This simplified PE has 2 adders, 2 shifters and 4 multiplexers, and it reduces the number of inner iterations from 16 or 32 times for a full CORDIC with word length 16 and 32 bits, respectively, to only one or 6 inner iterations with the CORDIC circular rotation mode. However, decreasing the inner iterations will cause an increased number of outer sweeps because of the imprecise inner iterations. Therefore, the simplified CORDIC architecture can reduce the size of area but requires more sweeps. On the other hand, the full CORDIC architecture needs fewer sweeps but requires more area.

Table 1 gives a set $\mathcal{A}$ approximated rotation angles for a simplified 32-bits scaling free μ-rotation CORDIC PE. For

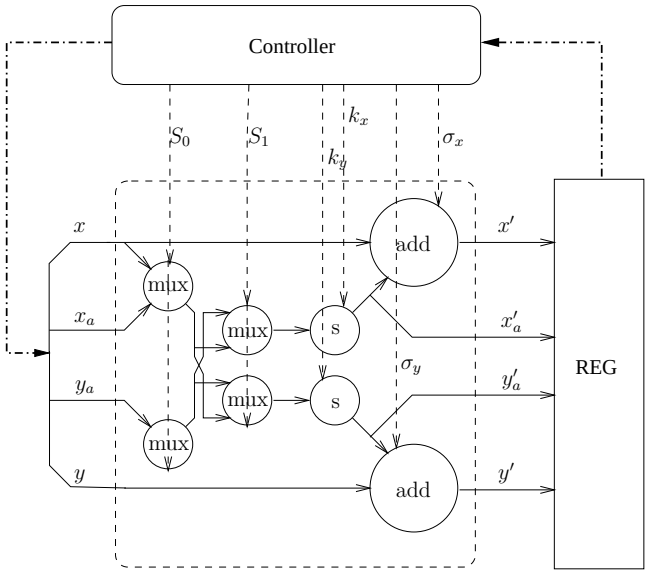

Fig. 2. The block diagram of a simplified CORDIC PE, including 2 adders, 2 shifters and 4 multiplexers.

a given accuracy n_m, this look-up table is constructed using the aforementioned four approximation methods in Goetze and Hekstra (1995). 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 cheapest possible method. It should be noticed that we have slightly modified the look-up table. First, since we only need the $\tan\theta$ for searching the optimal angle in Eq. (6), we can store $2\times\tan\theta$ instead of performing arctan operation to reverse the rotation angle in the look-up table. Second, we look into the critical path in Table 1. For angle index $k{=}{-}1$, it requires six cycles per iteration. In fact, the global clock in synchronous circuit is usually determined by the critical path, which also means that the maximum timing delay per iteration is 6 cycles. Therefore, in order to improve the computational balance, we repeat the inner iteration steps of the angles until they are close to the critical one. For example, when an optimal rotation angle index $k{=}{-}8$, it will repeat three times from the index -8 to the index -10. In this way, we can balance the overall computing overhead and improve the computational efficiency.

Figure 3 shows a block diagram of a 4×4 full Jacobi EVD array including one controller and 16 PEs. The shaded diagonal processors will first search the optimal rotation angle and then forward these angles to the off-diagonal PEs.

5 Experimental results

In this work, we have simulated four different cases of the cyclic–by–row parallel Jacobi EVD method in Matlab and on Xilinx FPGA respectively:

Table 1. The set $\mathcal{A}$ of μ-rotations for 32-bit accuracy, showing the method used, the $\tan\theta$ angle and the cost of rotation and scaling.

angle index	method	angle	cost (shift-add operations)		cycle	repeat
k		$2\times\tan\theta_k$	rot.	scl.	count	
−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

1. Full rotation CORDIC with 32 iteration steps.
2. Half rotation CORDIC with 16 iteration steps.
3. Simplified μ-rotation CORDIC with one single inner iteration step (μ-CORDIC).
4. Simplified μ-rotation CORDIC with 6 inner iteration steps (6-CORDIC).

5.1 Matlab simulation

At present we have tested with numerous random symmetric matrices **A** of size 4×4 to 50×50. Figure 4 shows the average number of Shift-Add operations needed to compute the eigenvalues for each size of EVD array. Apparently, both Full and Half CORDIC require much more effort than the simplified CORDIC. The 6-CORDIC requires a little more than the μ-CORDIC in average. On the other hand, we have

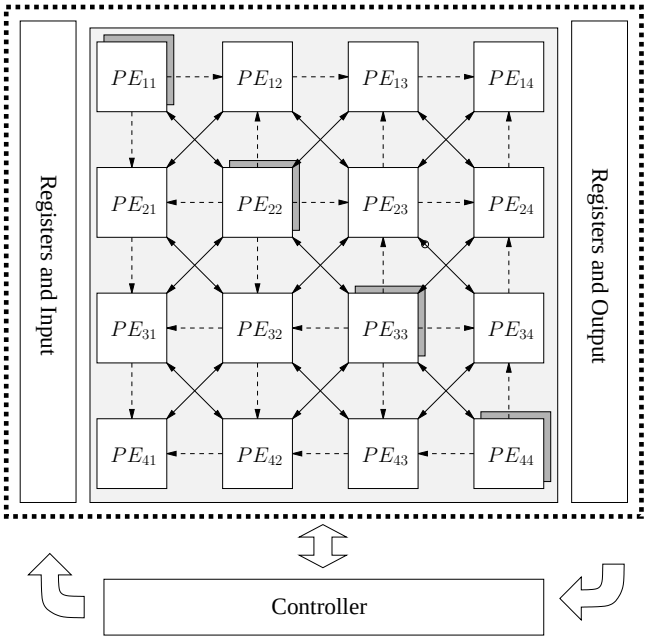

Fig. 3. The block diagram of a 4×4 Jacobi EVD array with 16 μ-rotation elements for FPGA implementation.

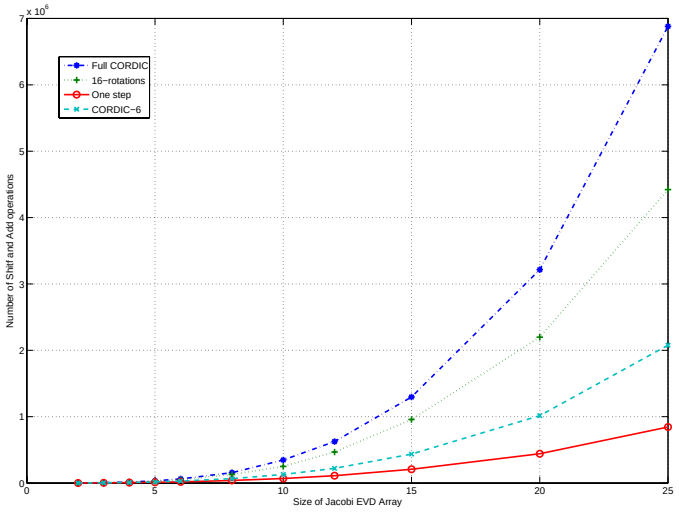

Fig. 4. Number of Shift-Add operations vs. Jacobi EVD array sizes for different CORDIC solutions.

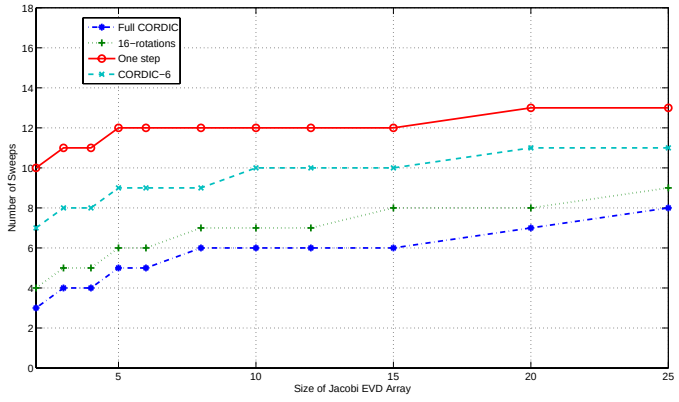

Fig. 5. The required number of sweeps vs. Jacobi EVD array sizes for different CORDIC solutions.

also simulated the number of the sweeps as shown in Fig. 5. Here, when the Jacobi EVD array's size is 20×20, the μ-CORDIC requires 13 sweeps which is almost twice than the Full CORDIC. Although the simplified μ-rotation CORDIC PE can improve the computational efficiency, it also increases the timing delay. The simplified 6-CORDIC not only requires less sweeps than the μ-CORDIC but also reduces the timing delay. Therefore, the simplified 6-CORDIC is actually a good compromise between the timing delay and the computational effort.

Consequently, from an algorithmic point of view, there is no doubt that we would rather realize the Jacobi method by utilizing the orthonormal simplified μ-rotation CORDIC method. However, when it comes to the VLSI circuit design (i.e., here we use VHDL for RTL design), things become totally different.

5.2 FPGA implementation

We have modeled a μ-rotation CORDIC PE in VHDL and compared with a full-pipeline CORDIC which is generated by the Xilinx Coregen automatically. Later, we synthesized these two CORDIC processors by Xilinx ISE into three different FPGA devices. It should be noticed that the word-length is 32 bits. Table 2 shows the syntheses results for Area, Timing Delay and the size of EVD array for each FPGA device (e.g., XCV1000-6FG680 0.22 μm, XC2V8000-5FF1517 0.15 μm and XC5VL330-2FF1760 65 nm). There are some important points that can be observed.

First of all, for the XCV1000-6FG680 0.22 μm, we are not able to implement a full EVD array directly because of the FPGA device could not provide enough configuration resources for implementation. Second, when the VLSI technology came to 0.15 μm, the FPGA device still can not provide enough hardware resources for regular CORDIC implementation. However, it is possible to implement a 14×14 EVD array with the presented μ-rotation CORDIC architecture. Although it needs more sweeps than the regular one, it enables significantly increased parallelism compared to the full CORDIC method. Finally, when the VLSI design keeps shrinking down into 65nm we are able to realize a 25×25 EVD array for solving the Eigenvalue problem of a 50×50 symmetric matrix **A** with the simplified μ-rotation CORDIC PE. Using the 6-CORDIC method allows three times the

Table 2. Area, Delay and the maximal size of EVD array of different Xilinx FPGA devices (i.e., XCV1000-6FG680, XC2V8000-5FF1517 and XC5VL330-2FF1760).

		XCV1000, 0.22 μm	XC2V8000, 0.15 μm	XC5VL330, 65 nm
6-CORDIC×3	Area	454/24.576 LUTs	464/93.184 LUTs	332/207.360 LUTs
	Delay	12.506 ns (79.9 MHz)	8.802 ns (113.6 MHz)	3.934 ns (254.2 MHz)
	EVD	7×7	14×14	25×25
	Matrix	14×14	28×28	50×50
Full CORDIC	Area	5.938/24.576 LUTs	5.938/93.184 LUTs	5.938/207.360 LUTs
	Delay	14.977 ns (66.8 MHz)	7.295 ns (137.1 MHz)	3.52 ns (284 MHz)
	EVD	2×2	4×4	6×6
	Matrix	4×4	8×8	12×12

matrix size of the full CORDIC. Therefore, utilizing the Full CORDIC would cause a partition problem and the processor array would require handling the partition sequentially. This requires an external memory and a more complicated control routine.

6 Conclusions

In this paper, we presented a design concept for parallel iterative algorithms when the VLSI design keeps evolving into nanoscale. For iterative algorithms we are able to simplify/modify the PEs as long as the convergence is guaranteed, such that the parallelism of the implementation can be increased. This is paid for by an increased number of iterations. Computing the EVD by the parallel Jacobi algorithm was used as an example. We have synthesized it into three different Xilinx FPGA devices. The experimental results show that we can realize a 25×25 full Jacobi EVD array into Xilinx XC5VL330 65nm FPGA device. In future work we will investigate the influences of the interconnects, i.e., with advancing VLSI technology the simplified PEs become smaller and smaller in comparison with the interconnection structure of the processor array. This fact requires that the varying importance of interconnects must be incorporated into the design concept.

References

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

Brent, R. P. and Luk, F. T.: The Solution of Singular-Value and Symmetric Eigenvalue Problems on Multiprocessor Arrays, SIAM Journal on Scientific and Statistical Computing, 6, 69–84, 1985.

Gelsinger, P.: Moore's Law: "We See No End in Sight,", Tech. rep., Intel Chief Technology Officer, http://websphere.sys-con.com/node/557154, 2008.

Goetze, J. and Hekstra, G.: An Algorithm and Architecture Based on Orthonormal Micro-Rotations for Computing the Symmetric EVD, INTEGRATION – The VLSI Journal, 20, 21–39, 1995.

Gotze, J., Paul, S., and Sauer, M.: An Efficient Jacobi-Like Algorithm for Parallel Eigenvalue Computation, IEEE Transactions on Computers, 42, 1058–1065, 1993.

Klauke, S. and Goetze, J.: Low Power Enhancements for Parallel Algorithms, in: IEEE International Symopsium on Circuits and Systems, 2001.

Parhi, K. K. and Nishitani, T.: Digial Signal Processing for Multimedia Systems, MARCEL DEKKER, New York, 1999.

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

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

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

Vitullo, F., L'Insalata, N. E., Petri, E., Saponara, S., Fanucci, L., Casula, M., Locatelli, R., and Coppola, M.: Low-Complexity Link Microarchitecture for Mesochronous Communication in Networks-on-Chip, IEEE Transactions on Computer, 57, 1196–1201, 2008.

Volder, J.: The CORDIC trigonometric computing technique, IRE Trans. Electron. Comput., EC-8, 330–334, 1959.

Walther, J.: A unified algorithm for elementary functions, in: Proc. Spring Joint Comput. Conf., vol. 38, pp. 379–385, 1971.

Wolf, W.: The future of multiprocessor systems-on-chips, in: Annual ACM IEEE Design Automation Conference, pp. 681–685, 2004.