LOW POWER ENHANCEMENTS FOR PARALLEL ALGORITHMS

Stephan Klauke and Jürgen Götze

University of Dortmund, Information Processing Lab
Otto-Hahn-Str. 4, 44221 Dortmund, Germany
email: {stephan.klauke | juergen.goetze}@udo.edu
http://www-dt.e-technik.uni-dortmund.de

ABSTRACT

In this paper we present a general approach for reducing switching activity on the algorithmic level. We concentrate on iterative algorithms that are suitable for an implementation on parallel processor arrays. The reduction is substantially reached by avoiding operations that hardly contribute to the convergence of the implemented algorithm. Our general approach is exemplified on the implementation of a specific algorithm, i.e. the eigenvalue decomposition (EVD) of a real symmetric matrix.

1. INTRODUCTION

In recent years, power consumption has become a critical design concern for ICs/ASICs not only as a result of the preceding success of portable consumer electronics but also because of the increasing costs for packaging and cooling of ICs with high power consumption. In contrast the main design constraint in many applications (e. g. mobile communication) is given by the real-time requirements of the implemented signal processing tasks. Concurrent processing techniques had to be introduced early to gain the desired data rates needed for this kind of applications.

About two decades ago, systolic architectures were first introduced as an efficient way for the implementation of many algorithms in different fields of applications. Systolic arrays show a high degree of concurrency, they have a regular and modular structure and thus a regular and local interconnection scheme. These properties make them an ideal solution for applications with high throughput and large processing bandwidth [10, 9, 4]. As the systolic architecture itself is highly optimized in terms of parallelism and speed, the common architecture level low power strategy "trading area/speed for power" [14, 2, 11] fails to gain a significant power reduction.

1.1. Power reduction at the algorithm level

Since it is well known, that significant power reductions can be obtained on the algorithmic level [12, 14], we investigate possible power savings in parallel algorithms by avoiding operations that hardly contribute to the convergence of the implemented algorithm. Here, it is important to mention that the regularity of the implementation and the data flow is not changed by our modifications. That means the parallel algorithm only skips the specific operations, while keeping its overall structure.

Our general strategy is to ascertain parameters which allow to determine the contribution of subsequent operations to the overall result (resp. to the convergence) of an iterative algorithm. That means we specify some "look ahead" technique for determining unnecessary subsequent operations. Of course, the parameters must be easy to compute such that the power we save by neglecting the respective operations is much less than the power consumed by the parameter evaluation. In this context the term "operation" may be used not only for basic functions like addition or multiplication but could also be extended to some kind of computational module like a rotation or the computation of a couple of basic functions. This strategy is most efficient if it is possible to avoid operations whose results are processed by several other operations because these subsequent operations can be avoided without further cost.

A very profound knowledge of the inherent structure of the algorithm (parallelism, regularity) as well as the convergence behaviour is necessary in order to find the required parameters and to identify the possible power savings.

The algorithm we examine as an example of the described strategy is a Jacobi-like eigenvalue decomposition (EVD) of a real symmetric matrix, whose realization on a parallel processor array was presented by Brent and Luk [1].

2. EIGENVALUE DECOMPOSITION (EVD)

An eigenvalue decomposition of the real symmetric $n \times n$ matrix $\mathbf{A}$ is its factorization into the product of three matrices

$$\mathbf{A} = \mathbf{Q}\Lambda\mathbf{Q}^T \tag{1}$$

where $\mathbf{Q}$ is an orthogonal matrix ($\mathbf{Q}\mathbf{Q}^{\mathrm{T}} = \mathbf{I}$) and Λ is diagonal and contains the eigenvalues of $\mathbf{A}$.

C. G. J. Jacobi presented his iterative method for computing an EVD already in 1846 [8]. In each iteration he annihilated the off-diagonal element with the maximum absolute value using a two-sided plain orthogonal rotation. He repeated this procedure until the sum of the squares of all off-diagonal elements (referred to as the off-diagonal norm or "ODN" in the sequel) was diminished to a certain accuracy. This strategy ensures maximum convergence and thus a minimum number of iterations. We call this procedure Classic Jacobi.

Because the maximum search performed in the classic Jacobi algorithm is very inefficient to implement on a sequential computer, cyclic Jacobi methods were proposed early. One of the most straightforward and regular variants is the cyclic-by-row method, where the pivot is chosen from left to right, row by row. This ordering is repeated through several "sweeps" (each with $n(n-1)/2$

single iteration steps) until all off-diagonal elements are annihilated to a certain accuracy. Brent and Luk [1] proposed another cyclic ordering scheme for annihilating the off-diagonal elements. Their ordering allows it to perform $\lfloor n/2 \rfloor$ rotations simultaneously and thus to map them on a parallel processor array which performs a whole sweep of Jacobi iterations in $(n-1)$ time steps.

Unfortunately, it is the regularity that causes the disadvantage, that convergence slows down and thus the number of iterations goes up. In fact the cyclic-by-row algorithm takes up to twice the number of iteration steps to decompose a matrix containing uniformly distributed data compared to the Classic Jacobi.

2.1. A systolic EVD array

The systolic array presented by Brent and Luk [1] consists of $\lceil \frac{n}{2} \times \frac{n}{2} \rceil$ processing elements (PEs) each containing a 2×2 sub-block of the matrix to be decomposed. Figure 1 shows an array for a 6×6 matrix with 9 processing elements.

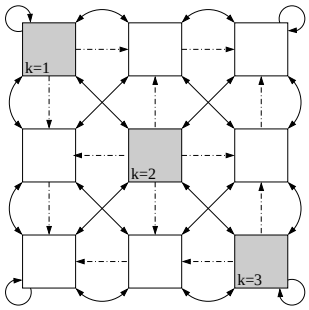

Figure 1: SVD/EVD array proposed by Brent and Luk.

The data processing in the array is done in two main steps: In a first step the PEs on the diagonal compute the cosine and sine of the angle θ that is needed to annihilate their off-diagonal elements. The four data elements in one diagonal PE are taken as a matrix $\mathbf{A}_k$ given by

$$\mathbf{A}_k = \begin{pmatrix} a_{11} & a_{12} \\ a_{12} & a_{22} \end{pmatrix} . \quad (2)$$

The angle θ_k is set to zero if $a_{12} = 0$. Otherwise the sine and cosine value of θ_k is computed as follows:

$$\sigma_k = \frac{2a_{12}}{a_{22} - a_{11}} \quad (3)$$

$$\tan\theta_k = \frac{\operatorname{sign}\sigma_k}{|\sigma_k| + \sqrt{1+\sigma_k^2}} \quad (4)$$

$$\cos\theta_k = \frac{1}{\sqrt{1+\tan\theta_k}} \quad (5)$$

$$\sin\theta_k = \tan\theta_k \cos\theta_k \quad (6)$$

After the computation the results are transmitted along the corresponding row (θ_r) and column (θ_c). This transmission is indicated by the dashed lines in figure 1.

In a second step all PEs perform a two sided (2×2) transformation with the appropriate angles, described by

$$\mathbf{B} = \mathbf{Q}(\theta_r)^T \mathbf{A} \mathbf{Q}(\theta_c) \quad (7)$$

with

$$\mathbf{Q}(\theta) = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} . \quad (8)$$

With this two steps $\lfloor n/2 \rfloor$ rotations are completed. It needs $(n-1)$ further time steps to complete a whole sweep of iteration steps, that means to apply one rotation to each off-diagonal element. Before the next sweep can begin the PEs have to exchange their data elements as indicated by the solid lines in figure 1.

3. THE ALGORITHM LOW POWER STRATEGY

Our general approach to lower the power consumption of a systolic array is to reduce average switching activity by avoiding operations that do not contribute significantly to the convergence of the whole algorithm. In order to reach this goal one has to inspect the specific algorithm for sources of avoidable operations.

3.1. EVD - Inspecting the different internal parameters

Figure 2 shows the development of the ODN plotted during the run of two variants of EVD algorithms ($n = 10$). The parameters also shown are the absolute value of σ and of the element to be annihilated $\beta = a_{12}$. The computation stops if the ODN is reduced by a certain factor. In the case of the cyclic algorithm this criterion is checked each time a whole sweep of rotations has finished.

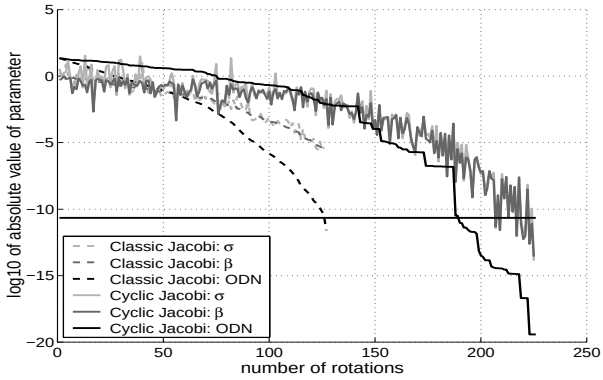

Figure 2: Development of parameters during the run of a Classic Jacobi and a cyclic Jacobi algorithm with parallel ordering.

3.2. EVD - Using a threshold method

As we inspect the decrease of the ODN during the cyclic Jacobi method we recognize several plateaus in the curves in figure 2. It is obvious that the rotations along these plateaus do not contribute very much to the overall result. Note, that we become aware of the plateaus by plotting the ODN over the single rotations of the algorithm (not over whole sweeps as usual).

A way to reduce such ”inefficient” operations is to apply a threshold strategy. That means in each sweep a threshold value is used in order to decide if a transformation is computed or not. The threshold method was already mentioned by Wilkinson in [16]. He worked with a fixed sequence of threshold values, lowering from sweep to sweep to reduce computation time. Every transformation regarding an off-diagonal element that is below the threshold value

n	$S_{\text{aver.}}$	$S_{\max}$	S_{wc}	**average overhead**
8	4.07	5.04	6	47.4 %
10	4.39	5.56	6	36.7 %
20	5.23	5.93	6	14.7 %
30	5.67	6.62	7	23.5 %
50	6.17	7.13	8	29.6 %

Table 1: Simulation results from [1].

is omitted. This means in the case of the parallel EVD array, that in the appropriate diagonal cell the cosine and sine value are set to 1 and 0 without further activity. The computation of equations (3) to (6) is skipped. In addition the computational complexity of the $(n-2)$ cells in the corresponding row and colum is reduced to a one sided rotation as $\mathbf{Q}(\theta_{\text{r}})$ resp. $\mathbf{Q}(\theta_{\text{c}})$ becomes the identity matrix.

Results from simulations using σ as threshold criterion showed that there is no considerable further reduction of the number of rotations. As the effort to compute σ is not neglectable we went back on a_{12} for the threshold decision.

3.3. EVD - Monitoring the stage of diagonalization

As mentioned before, in general the ODN is used to determine the progress of diagonalization. As for the computation of this parameters access to all data elements of a matrix is required, this approach is not amenable for a systolic array with its data distributed over several processing elements. Thus it is often proposed to execute a predetermined number of sweeps (rotations).

To ensure the desired accuracy, some kind of worst case assumption must be made. This will cause additional iterations in the average case, as in general the execution is stopped after a whole sweep only. Table 1 shows some simulation results taken from [1] concerning the average and maximum number of sweeps for different matrix sizes ($n \times n$). The table also reveals the overhead between the worst and the average case in percent.

Our second strategy to reduce switching activity is to monitor the stage of diagonalization as proposed in [6] to optimize the number of sweeps. It was shown that for a parallel Jacobi algorithm the stage of quadratic convergence is reached if the maximum absolute value of σ does not exceed $1/2$ during sweep number l:

$$\left|\sigma^{(l)}\right|_{\max} < 1/2 \tag{9}$$

While it is almost impossible to sum all off-diagonal elements after a sweep in a hardware implementation it is easy to check if condition (9) is met. As the value of σ is computed in the diagonal PEs only, each diagonal PE has to set a binary flag if the condition is not met for any of the rotations evaluated in the respective sweep. If the condition is met a fixed number of additional rotations follow which depends on the desired accuracy.

3.4. EVD - Convergence of the modified algorithm

Global and ultimate quadratic convergence of the cyclic-by-row method with and without threshold strategy have been proved [3, 15]. Luk and Park [13] have shown that the parallel ordering is essentially equivalent to the cyclic ordering and thus holds the same convergence properties. The proof of global and ultimate quadratic convergence for the Jacobi algorithm using diagonalization monitoring is given in [6].

3.5. EVD - Impact on power consumption

Until now all savings were expressed as the "number of reduced operations". To get a better feeling about what saving one "operation" means in terms of switching activity, let us consider the following: The implementation of equation (3) to (6) needs four additions, three multiplications, two square-roots and three divisions in each diagonal cell. The double-sided transformations in the diagonal cells need sixteen multiplications and eight additions.

Regarding an EVD array, saving one "operation" means skipping all these computations in the diagonal cell plus a one-sided transformations (eight multiplications and four additions) in the corresponding $(\frac{n}{2}-1)$ off-diagonal cells.

4. SIMULATION RESULTS

We compared three different variants of the EVD algorithm:

- **A1** parallel ordering without further enhancements with fixed number of sweeps
- **A2** parallel ordering and threshold strategy with fixed number of sweeps
- **A3** parallel ordering with threshold strategy and diagonalization monitoring

The different algorithms were applied to random symmetric $n \times n$ matrices containing uniformly and independently distributed elements in the range $[-1,1]$. The number of fixed sweeps was derived from the simulation results given in table 1, which were produced using the stopping criterion that the ODN of the output matrix is at least 10^{12} times smaller than the ODN of the input matrix.

At the moment our simulations are idealized in the way that they are performed sequentially using Matlab. Thus hardware overhead produced by the implementation of the control circuitry that is needed for our enhancements is neglected.

For compensation we only consider the number of multiplications as the main source of switching activity, neglecting all other arithmetic operations that are also needed. With this estimation the energy saved per skipped operation is the energy produced by the $19+8(\frac{n}{2}-1)$ multiplications required to perform one rotation.

Our simulation results show that a considerable reduction of multiplications can be achieved by applying our strategies. Figure 3 shows the average number of multiplications needed to perform the EVD using the algorithms A1 to A3. The average numbers shown were computed on a basis of 1000 simulation passes.

In figure 4 the average reduction in percent is plotted assuming that switching activity depends linearly on the number of multiplications performed. As the figure reveals a reduction between 20% and 30% can be reached comparing our method (A3) with the standard parallel ordering (A1). A reduction of 15% to 25% is reached if our method is compared with the parallel ordering using the threshold strategy (A2).

As depicted in figure 5 the accuracy constraint is not violated using our algorithm. The average factor by which the ODN is reduced always stays well above 10^{12}. As the figure also shows, in the average case the other algorithms reduce the ODN far beyond the required accuracy.

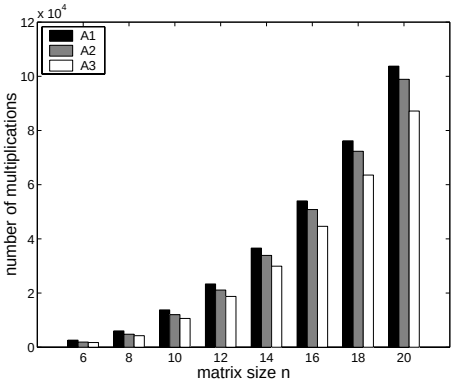

Figure 3: Average number of multiplications performed to compute an EVD with different variants of the parallel ordering.

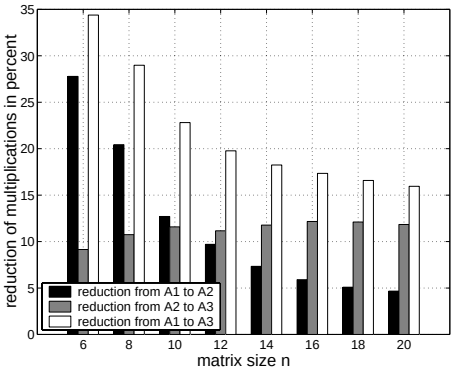

Figure 4: Average reduction of multiplications in percent.

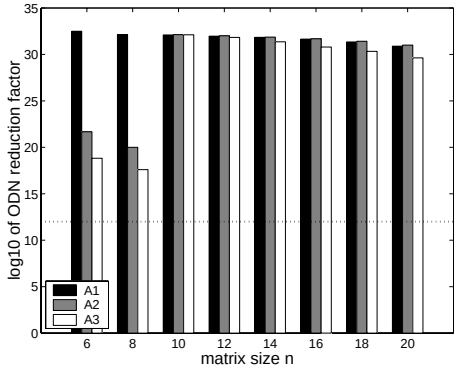

Figure 5: Average reduction of ODN.

5. CONCLUSIONS

The general approach presented can be applied to various iterative algorithms, provided that a simple criterion for skipping unnecessary operations can be established. It should even be extendible to direct algorithms that can be formulated as iterative methods (so called semi-iterative algorithms) [5].

At the current state of our work we just consider to switch off whole basic operations. At a next state a further reduction of power consumption may be reached by optimizing the basic operations themselves. One approach will be to replace the 2×2 matrix transformations with CORDIC-like elements that perform approximated rotations as presented in [7]. Using CORDIC elements, the application becomes much more suitable for a VLSI implementation and hardware complexity is reduced significantly. Of course there is also a disadvantage. The speed of convergence is slowed down so that more iteration steps are needed. Further simulations will show the contribution of such techniques to an additional reduction of switching activity.

6. REFERENCES

[1] R.P. Brent and F.T. Luk. The Solution of Singular-Value and Symmetric Eigenvalue Problems on Multiprocessor Arrays. *SIAM J. Sci. Stat. Computing*, 6(1):69–84, Jan. 1985.

[2] A.P. Chandrakasan and R.W. Brodersen. Minimizing Power Consumption in Digital CMOS Circuits. *Proceedings of the IEEE*, 83(4), April 1995.

[3] G. Forsythe and P. Henrici. The Cyclic Jacobi Method for Computing the Principal Values of a Complex Matrix. In *Trans. Americ. Math. Soc.*, 94, pages 1–23, 1960.

[4] J.A.B. Fortes and B.W. Wah. Special Issue: Systolic Arrays – From Concept to Implementation. *Computer*, 20(7):12–17, July 1987.

[5] J. Götze. An Iterative Version of the QRD for Adaptive RLS Filtering. *SPIE Conference on "Advanced Signal Processing: Algorithms, Architectures and Implementations". (San Diego, U.S.A)*, pages 438–450, 1994.

[6] Jürgen Götze. Monitoring the Stage of Diagonalization in Jacobi-Type Methods. In *Int. Conf. on Acoust., Speech and Signal Processing*, pages 441–447. IEEE, 1994.

[7] Jürgen Götze and Gerben J. Hekstra. An algorithm and architecture based on orthonormal μ-rotations for computing the symmetric EVD. *INTEGRATION, the VLSI journal*, 20:21–29, 1995.

[8] C.G.J. Jacobi. Über ein leichtes Verfahren, die in der Theorie der Säkulärstörungen vorkommenden Gleichungen numerisch aufzulösen. *Crelle J. reine angew. Mathematik*, 30:51–94, 1846.

[9] H.T. Kung. Why Systolic Architectures? *Computer*, 15(1):37–46, Jan. 1982.

[10] H.T. Kung and C.E. Leiserson. Systolic Arrays (for VLSI). In *Sparse Matrix Proc. 1978*, pages 256–282. Academic Press, Orlando, Florida, 1979.

[11] Paul Eric Landman. *Low-Power Architectural Design Methodologies*. PhD thesis, University of California at Berkeley, 1994.

[12] K.J.R. Liu, A.-Y. Wu, A. Raghupathy, and J. Chen. Algorithm-Based Low-Power and High-Performance Multimedia Signal Processing. In *Proceedings of the IEEE*, volume 86, pages 1155–1202, 1998.

[13] F. T. Luk and H. Park. On the equivalence and convergence of parallel Jacobi SVD algorithms. In *Proc. SPIE Advanced Algorithms and Architectures for Signal Processing II*, volume 826, pages 152–159. Society of Photooptical Instrumentation Engineers, 1987.

[14] R. Mehra, D.B. Lidsky, A. Abnous, P.E. Landman, and J. Rabaey. *Low Pwer Design Methodologies*, chapter Algorithm and Architectural Level Methodologies for Low Power. Kluwer Academic Publishers, 1996.

[15] J.H. Wilkinson. Note on the Quadratic Convergence of the Cyclic Jacobi Process. *Numer. Math.*, 4:296–300, 1962.

[16] J.H. Wilkinson. *The Algebraic Eigenvalue Problem*, pages 277–278. Clarendon Press, Oxford, 1995.