Low Power Algorithms for Signal Processing

Stephan Klauke and Jürgen Götze
Universität Dortmund, AG Datentechnik, Fakultät Elektrotechnik
Otto-Hahn-Strasse 4, 44221 Dortmund, Germany
http://www-dt.e-technik.uni-dortmund.de

Abstract

In this article we present techniques to reduce power consumption in parallel processor arrays. The reduction is substantially reached by avoiding non-essential operations, i. e. operations that hardly contribute to the convergence of the considered algorithm. The presented approach is clarified on the basis of a parallel implementation of a Jacobi-like real eigenvalue decomposition (EVD) but should be transferable to other algorithms that are suited to be implemented on a parallel processor array.

1 Introduction

A few years ago, the design constraints for an IC/ASIC were maximum circuit speed and minimum area. However, in recent years, power consumption has become a critical design concern not only as a result of the preceding success of portable consumer electronics but also because of the enormously increasing costs for packaging and cooling of high performance ICs with high power consumption. In a portable system, low power components are essential because power consumption has a direct impact on size and weight of batteries, i. e. on size and weight of a whole device.

If we address the area of mobile communication, the real-time requirements of many signal processing tasks are a very critical design concern. For this kind of applications, the DSPs/ASICs often did not deliver sufficient speed to process all data sequentially, so that concurrent processing techniques had to be introduced early to speed up the designs and to gain the desired data rates.

While such techniques like pipelining and parallel processing were at first used almost exclusively to fulfil the given timing constraints, some methodologies were developed later to exploit them for low power design issues.

A recurring theme in this context is "trading area/speed for power": The designer tries to speed up an architecture e. g. by inserting some extra arithmetic units in such a way that a higher level of concurrency is reached. Afterwards the supply voltage V_{dd} of the circuit can be lowered until the growing delay times reach the upper limit of timing constraints. Because of the quadratic influence of V_{dd} on power dissipation, the power consumption of the whole circuit is lowered in this way, although more area (and thus more load-capacity) is needed for the added logic [15, 2, 13].

About two decades ago, systolic architectures were first introduced as an efficient way for the implementation of many algorithms in different fields of applications [12, 5, 11]. 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 and especially for an automated VLSI implementation [4].

1.1 Power Reduction on Algorithm Level

As the systolic architecture itself is highly optimized in terms of parallelism and speed, the low power strategy mentioned above fails to gain further improvements in power consumption.

In this paper we concentrate on the algorithm design level and present some techniques to decrease power consumption in the systolic array implementation of a specific algorithm. The strategy is to ascertain parameters which allow to determine the contribution of an operation to the convergence of the whole algorithm. Of course, these 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 operation itself. 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 array was presented by Brent and Luk [1].

2 Eigenvalue Decomposition

An eigenvalue decomposition of the real symmetric $n \times n$ matrix A is its factorization into the product of three matrices $A = Q\Lambda Q^T$, where Q is an orthogonal matrix

($QQ^{\mathrm{T}} = I$) and Λ is diagonal and contains the eigenvalues of A.

C. G. J. Jacobi presented his iterative method for computing an EVD already in 1846 [10]. In each iteration he annihilated the off-diagonal element with the maximum absolute value 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.

About 100 years later Jacobi's method was used as a standard method for handling the symmetric eigenvalue problem but it was then almost completely replaced by the so-called QR-methods. The Jacobi-like methods were rediscovered again with the introduction of multiprocessor arrays for EVD and singular-value computations because of their regular structure with a high degree of inherent parallelism [7].

Because the maximum search performed in the Classic Jacobi algorithm is very inefficient to implement on a sequential computer, cyclic procedures for the pivot choice 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" until all off-diagonal elements are annihilated to a certain accuracy.

Brent and Luk [1] proposed another scheme for choosing the pivot element which optimized the EVD algorithm for the mapping on a parallel processor array.

In contrast to the Classic Jacobi algorithm the implementations of both variations mentioned are more regular and straightforward but hold the disadvantage that convergence slows down and thus the number of iterations goes up.

As depicted in Figure 1, the number of iterations is nearly doubled by using the cyclic-by-row instead of the Classic Jacobi algorithm.

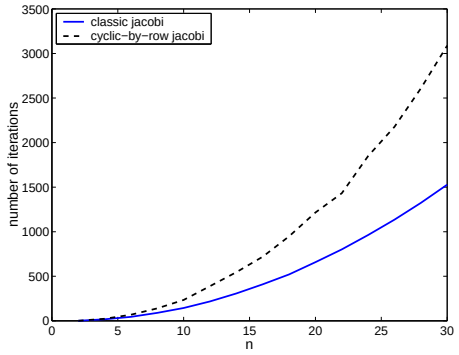

Figure 1 Average number of iterations needed to complete a classic and a cyclic-by-row Jacobi-EVD.

2.1 A parallel EVD array

The parallel 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 2 shows an array for a 6×6 matrix with 9 processing elements.

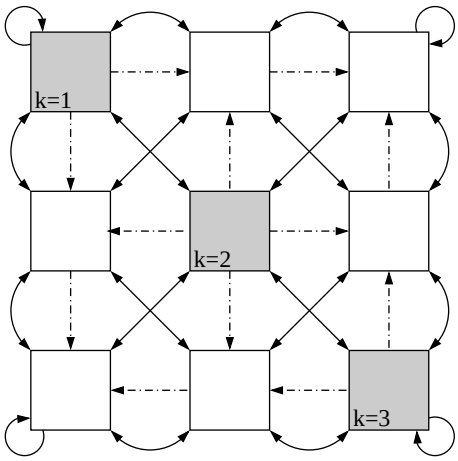

Figure 2 SVD/EVD array proposed by Brent and Luk.

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 A_{k} given by

$$A_{\mathrm{k}} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} . \tag{1}$$

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:

$$\sigma_{\mathrm{k}} = \frac{2a_{12}}{a_{22} - a_{11}} \tag{2}$$

$$\tan\theta_{\mathrm{k}} = \frac{\operatorname{sign}\sigma_{\mathrm{k}}}{|\sigma_{\mathrm{k}}| + \sqrt{1+\sigma_{\mathrm{k}}^2}} \tag{3}$$

$$\cos\theta_{\mathrm{k}} = \frac{1}{\sqrt{1+\tan\theta_{\mathrm{k}}}} \tag{4}$$

$$\sin\theta_{\mathrm{k}} = \tan\theta_{\mathrm{k}}\cos\theta_{\mathrm{k}} \tag{5}$$

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 2.

In a second step all PEs perform a two sided plain rotation with the appropriate angles described by

$$B = Q(\theta_{\mathrm{r}})^{\mathrm{T}} A Q(\theta_{\mathrm{c}}) \tag{6}$$

with

$$Q(\theta) = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} . \tag{7}$$

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

3 The Algorithm Level Approach

Our general approach to lower the power consumption of the systolic array implementation is to reduce average switching activity by avoiding operations that do not contribute in a certain way to the convergence of the whole algorithm. In order to reach this goal we had to inspect the algorithm for sources of avoidable switching activity.

3.1 Inspecting the different internal parameters

Figure 3 shows the development of the ODN plotted during the run of two different EVD algorithms. Additionally the absolute value of σ and of the element to be annihilated $\beta = a_{12}$ are shown. 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.

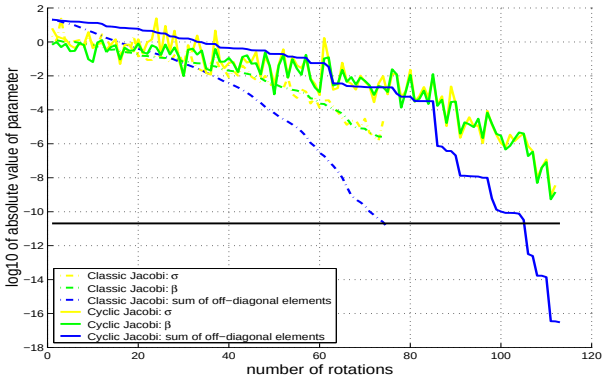

Figure 3 Development of parameters during the run of a Classic Jacobi and a cyclic Jacobi algorithm with parallel ordering.

3.2 Using a threshold method

As we inspect the decrease of the ODN during the cyclic EVD we recognize several plateaus in the curves in figure 3. It is obvious that the rotations along these plateaus do not contribute very much to the overall convergence of the algorithm. Note, that we become aware of the plateaus by plotting the ODN over the single rotations of the algorithm (not over whole sweeps as shown in many other cases).

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 rotation is computed or not. The threshold method was already mentioned by Wilkinson in [17]. He worked with a fixed sequence of threshold values, lowering from sweep to sweep to reduce computation time. Every rotation regarding an off-diagonal element that is below the threshold value is omitted. This means in the case of the parallel 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 (2) to (5) is skipped. In addition the computational complexity of the $2(n-1)$ cells in the corresponding row and column is reduced to a one sided rotation as $Q(\theta_r)$ resp. $Q(\theta_c)$ becomes the identity matrix.

We also considered to use σ as a threshold criterion as we use this parameter for monitoring the stage of diagonalization, i. e. the convergence of the algorithm (described below). We went back to a_{12} as threshold criterion as simulation results showed that with the use of σ the number of rotations and thus switching activity is not considerably reduced, especially as we cannot neglect the effort to compute σ (subtraction and division).

n	$S_{aver.}$	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 parallel [1].

3.3 Monitoring the stage of diagonalization

Another problem already mentioned, especially for parallel structures, is the fact that Jacobi used the ODN to evaluate the progress of convergence. In a parallel array the addition of all off-diagonal elements distributed over the processing cells is not amenable. 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 rotations in the average case, as in general the execution is stopped after a whole sweep only. Table 1 shows some figures 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 [8] 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:

$$\left|\sigma^{(l)}\right|_{\max} < 1/2 \tag{8}$$

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 (8) 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 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, 16]. Luk and Park [14] 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 [8].

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 our enhancements is neglected and only the reduction of operations is considered. This assumption should not be too far from reality because as described above the basic operations to be omitted are made up of the computation of equations (2) to (5) and a two-sided 2×2 matrix transformation in the diagonal cell and additionally $2(n-1)$ one sided transformations in the corresponding off-diagonal cells.

Our simulation results show that a considerable reduction of operations can be achieved by applying our strategies. Figure 4 shows the average number of rotations

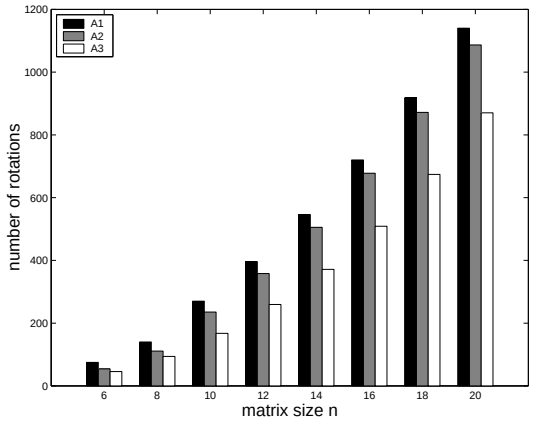

Figure 4 Average number of rotations performed to compute an EVD with different variants of the parallel ordering.

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 5 the average reduction is plotted in percent. As the figure reveals a reduction between 20% and 40%

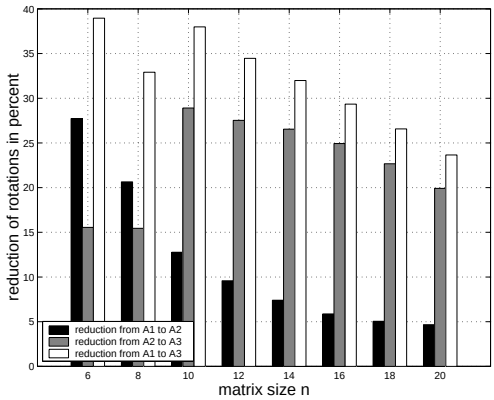

Figure 5 Average reduction of rotations in percent.

can be reached comparing our method (A3) with the standard parallel ordering (A1). A reduction of 15% to 30% is still reached if our method is compared with the parallel ordering using the threshold strategy (A2).

As depicted in figure 6 the accuracy constraint is not violated using our algorithm. The average factor by which the ODN is reduced is always above 10^{12}.

5 Conclusions

The presented approach can be applied to various iterative algorithms, provided that a simple criterion for skipping unnecessary operations can be established. It is even extendible to non-iterative, i. e. direct algorithms that can

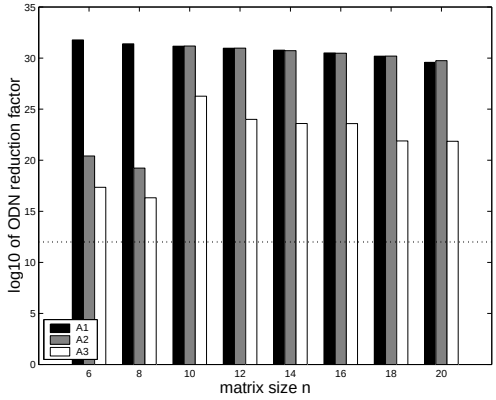

Figure 6 Average reduction of ODN.

be formulated as iterative method (so called semi-iterative algorithms) [6].

Another advantage not further discussed regards the diagonalization monitoring. It is possible to regulate the power consumption depending on the desired accuracy. To implement this feature, only the number of rotations that are computed after quadratic convergence has been reached has to be programmable. Of course there still is the problem of determining the desired accuracy level and to relate the number of rotations with the accuracy reached.

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 [9].

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] W.M. Gentleman and H.T. Kung. Matrix Triangularization by Systolic Arrays. In *SPIE Real-Time Signal Processing IV 298*, pages 19–26, 1981.

[6] 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.

[7] J. Götze. *Orthogonale Matrixtransformationen.* Oldenbourg, 1995.

[8] 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.

[9] 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.

[10] 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.

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

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

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

[14] 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.

[15] Jan M. Rabaey and Massoud Pedram, editors. *Low Power Design Methodologies*. Kluwer Academic Publishers, 1996.

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

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