ON THE DERIVATION OF PARALLEL FILTER STRUCTURES FOR ADAPTIVE EIGENVALUE AND SINGULAR VALUE DECOMPOSITIONS

Marc Moonen[†], *Ed Deprettere*[‡], *Ian K. Proudler*[††], *John G. McWhirter*[††]

[†] E.E. Dept., K.U. Leuven, 3001 Heverlee, Belgium, marc.moonen@esat.kuleuven.ac.be
[‡] E.E. Dept., T.U. Delft, 2628 CD Delft, The Netherlands, ed@dutentb.et.tudelft.nl
[††] Defence Research Agency, Malvern, Worcs WR14 3PS, UK, jgm@rsre.mod.uk

ABSTRACT

A graphical derivation is presented for a recently developed parallel filter structure (systolic array) for updating eigenvalue and singular value decompositions. The derivation of this array is non-trivial due to the presence of feedback loops and data contra-flow in the underlying signal flow graph (SFG). This would normally prohibit pipelined processing. However, it is shown that suitable delays may be introduced to the SFG by performing simple algorithmic transformations which compensate for the interference of crossing data flows and eliminate the critical feedback loops. The pipelined array is then obtained either by 2-slowing and retiming the SFG or by means of dependence graph scheduling and assignment, and turns out to be an improved version of the arrray presented in [6].

1. INTRODUCTION

The eigenvalue (EVD) and singular value decomposition (SVD) have by now become standard linear algebra tools for modern digital signal processing, which find applications in beamforming and direction finding, spectrum analysis, systems identification, *etc.*

For real-time applications, it is necessary to continuously update the EVD/SVD as new observations are continuously added to the problem. In [5, 6], a fast updating technique for adaptive SVD has been described, as well as its parallel implementation on a square processor array. The updating algorithm has a reduced computational complexity, $\mathcal{O}(n^2)$ instead of $\mathcal{O}(n^3)$ where n is the problem size, and on a square array with $\mathcal{O}(n^2)$ processors, the throughput is $\mathcal{O}(n^0)$. This means that new updates are processed, one at a time, at a rate which is independent of the problem size.

In this paper a graphical derivation of (an improved version of) this array is outlined, where use is made of standard SFG manipulations as well as non-standard algorithmic manipulations. The starting point is the original SVD-updating algorithm of [5], which has a cyclic-by-rows ordering. The corresponding SFG exhibits a long critical data path, and thus the clock rate is inversely proportional to the problem size n. This SFG is then worked into a new algorithm/SFG with odd-even ordering and a fully systolic organization. Here the clock rate is independent of the problem size.

2. SVD UPDATING

The singular value decomposition (SVD) of a given (real) $m \times n$ matrix A is given as $A = U \cdot \Sigma \cdot V^T$ where $U^T U = V^T V = I_{n \times n}$ and Σ is a diagonal $n \times n$matrix. The corresponding EVD of $A^T A$ is then given as $A^T A = V \cdot \Sigma^2 \cdot V^T$. Here we consider the updating problem, where the aim is to track the matrices Σ and V, when new observations are added in each time step, *i.e.* when A is defined in a recursive manner, $A_{[k]} = \left[\begin{array}{c} A_{[k-1]} \\ \hline a_{[k]}^T \end{array}\right]$. An algorithm for this is developed and analyzed in [5]. The idea is to combine QR updating [1] with a Jacobi-type SVD algorithm [3]. An orthogonal matrix V is stored and updated, together with a triangular matrix R, which is always close[1] to Σ. The algorithm is then given as follows, where initially, $R = 0$ and $V = I$:

for $k = 1, \ldots, \infty$

1. matrix-vector multiplication

$$\tilde{a}_{[k]}^T \Leftarrow a_{[k]}^T \cdot V$$

2. QR update

$$\left[\begin{array}{c} R \\ \hline 0 \end{array}\right] \Leftarrow Q_{[k]}^{n|n+1} \ldots Q_{[k]}^{1|n+1} \cdot \left[\begin{array}{c} R \\ \hline \tilde{a}_{[k]}^T \end{array}\right]$$

3. SVD steps (Jacobi)

$$V \Leftarrow V \cdot \Phi_{[k]}^{1|2} \ldots \Phi_{[k]}^{n-1|n}$$

$$R \Leftarrow \Theta_{[k]}^{n-1|n} \ldots \Theta_{[k]}^{1|2} \cdot R \cdot \Phi_{[k]}^{1|2} \ldots \Phi_{[k]}^{n-1|n}$$

end

The QR update in step 2 is carried out by means of a sequence of n plane transformations [1]. $Q_{[k]}^{1|n+1}$ combines

Marc Moonen is a research associate with the Belgian National Fund for Scientific Research (NFWO). This research was partially sponsored by ESPRIT Basic Research Action Nr. 6688. K.U.Leuven-ESAT is a member of the DSP Valley^{TM} Network.

[1] When R is close to Σ, the V is also close to the true V-matrix. With the tracking error and time variation defined in terms of the angles between true and estimated subspaces and subspaces at different time steps, respectively, it has been shown that the tracking error is always smaller than the time variation in $\mathcal{O}(n)$ time steps [5].

0-7803-2431-5/95 $4.00

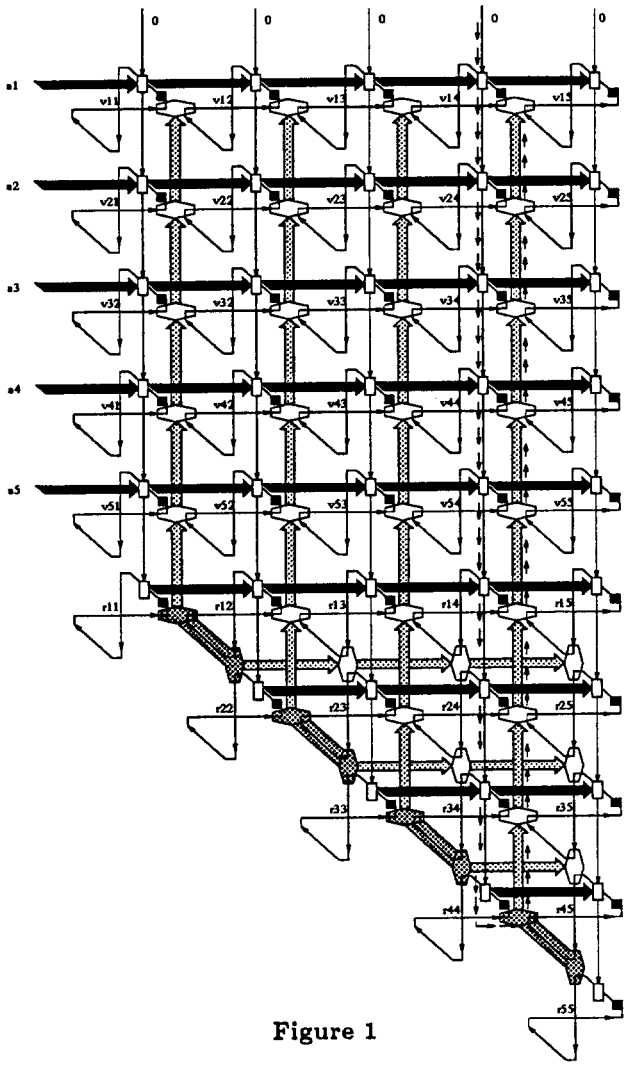

Figure 1

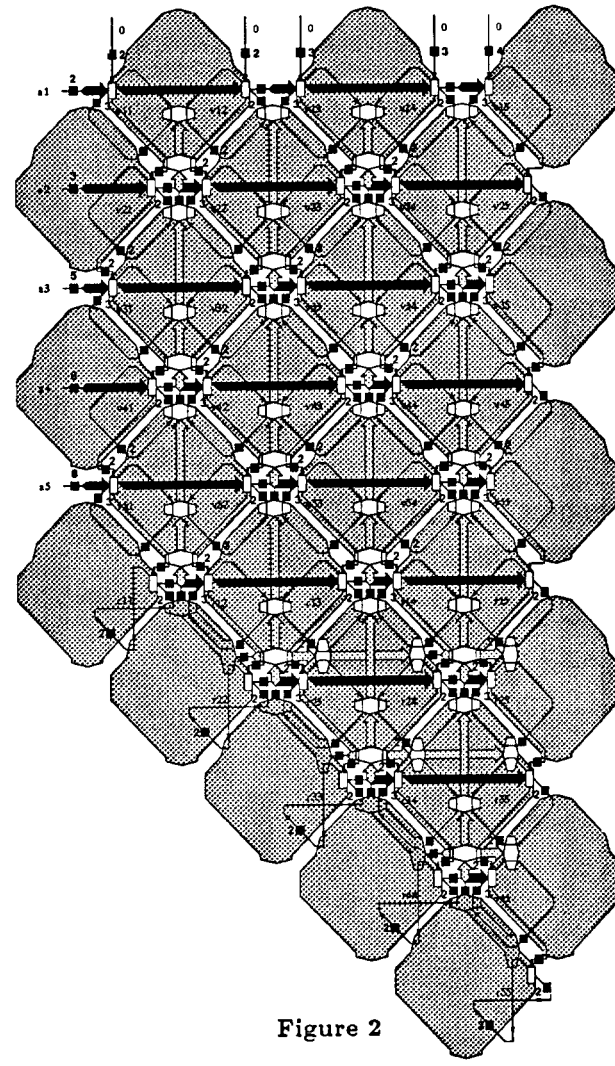

Figure 2

rows 1 and $n+1$ of the compound right-hand matrix, to zero its $(n+1,1)$ entry. $Q_{[k]}^{2|n+1}$ then combines rows 2 and $n+1$ to zero the $(n+1,2)$ entry, *etc.* The QR update moves the R matrix further from the diagonal form. The diagonal form is then (partly) restored in step 3, where a sequence of row and column transformations is applied (Jacobi-type diagonalization [3]). $\Theta_{[k]}^{1|2}$ and $\Phi_{[k]}^{1|2}$ combine rows 1 and 2 and columns 1 and 2 respectively, so that the (1,2) entry in R is zeroed. $\Theta_{[k]}^{2|3}$ and $\Phi_{[k]}^{2|3}$ combine rows 2 and 3 and columns 2 and 3 respectively, so that the (2,3) entry in R is zeroed., *etc.* A sequence of $n-1$ such row/column transformations is applied after each QR update. For more details, we refer to [3, 5, 6].

3. SIGNAL FLOW GRAPHS

A SFG for one complete update in the above algorithm is shown in **Figure 1**. Note that such a SFG may be viewed as a graphical representation of the algorithm, but also as a (admittedly fairly complicated) piece of hardware or filter structure. Black squares represent memory cells (delay elements), storing matrix elements (V-matrix in the square part, R-matrix in the triangular part). The components of the new data vector $a_{[k]}$ are fed in from the left and propagated to the right (black arrows). The white rectangles correspond to multiply-add cells in the square part (*cfr.* matrix-vector product, accumulated from top to bottom) and orthogonal rotation cells in the triangular part (*cfr.* QR updating). The white hexagons represent column and row transformations in the diagonalization procedure, which are initiated on the main diagonal and then propagated upwards and to the right.

It is seen that an element of the triangular matrix, *e.g.* $R_{[k-1]}(i,j)$, is fed out from the corresponding memory cell, first transformed through the QR updating process, and subsequently rotated with its upper, lower, left and right neighbour respectively. The resulting $R_{[k]}(i,j)$ is then again clocked into the memory cell. The SFG exhibits a long data path of length $4n$, without delay elements (as indicated in Figure 1), such that the clock rate is inversely proportional to n. This SFG will therefore be worked into a new algorithm/SFG with odd-even ordering, which is depicted in **Figure 2**. Here the shaded areas can all operate in parallel, because of the memory cells on all the connections in

between every two such areas. This means that the clock rate is independent of n. The SFG is 4-slowed (see below), which means that a new data vector $a_{[k]}$ may be fed in after each four clock cycles. This SFG corresponds to the SVD updating systolic array of [6] (up to a few modifications). Due to space limitations, only a brief outline of its derivation will be given. A formal description of the derivation in an automatic synthesis framework is given in [2].

4. SFG MANIPULATIONS

The *first step* in going from Figure 1 to Figure 2 is a simple **retiming** of all the column transformations (hexagons), starting with the one in the top right corner and moving on towards the diagonal. This means that the delay elements on the output arrows of each column transformation are removed, while at the same time delays are added on its input arrows. The result is given in **Figure 3.**

The *second step* in more significant, and involves a sequence of **algorithmic transformations.** The shaded area in Figure 3 has two multiply-add operations, involving elements $V(1,4)$ and $V(1,5)$ which are first rotated (column transformation). It is easy to show (matrix associativity) that these elements can be used in the multiply-add operations *before* being rotated, on condition that the multiply-add results are corrected afterwards by means of the same rotation. This is indicated in **Figure 4.**

The same algorithmic transformation may then be applied to two similar areas, as indicated in **Figure 5**, then to three areas, **Figure 6**, *etc.*. The end result for a complete 'wave' of transformations, from the top right corner towards the diagonal, is indicated in **Figure 7.**

Note that Figure 7 differs from Figure 1 only in the bottom layer. Because of that, a second 'wave' of transformations may be applied, starting again from the top right corner. Obviously, the second wave should end one layer earlier. Then a third wave is applied, *etc.*. The end result is given in **Figure 8.**

The *third step* is again a (standard) SFG retiming step. It is easy to show that by first **2-slowing and retiming** along SW-NE cuts end then 2-slowing and retiming along SE-NW cuts, Figure 8 may be transformed into Figure 2, which is our end result.

5. REFERENCES

[1] W.M. Gentleman, H.T. Kung, 'Matrix triangularization by systolic arrays'. *Real-Time Signal Processing IV*, Proc. SPIE, Vol. **298**, 1982, pp 19-26.

[2] P. Kapteijn, H.W. van Dijk, E.F. Deprettere, 'Controlling the critical path in time adaptive QR^{-}1 recursions. *Proceedings VLSI Signal Processing Workshop, VII*, pp 326-335 (1994).

[3] F.T. Luk, 'A triangular processor array for computing singular values'. *Lin. Alg. Appl.*, 1986, pp 259-273.

[4] I K Proudler, J G McWhirter, 'Algorithmic Engineering in Adaptive Signal Processing - Worked Examples'. *IEE Proceedings VIS*, 1994, pp 19-26.

[5] M. Moonen, P. Van Dooren, J. Vandewalle, 'An SVD updating algorithm for subspace tracking'. *SIAM J. Matrix Anal. Appl.*, 1992, pp 1015-1038.

[6] M. Moonen, P. Van Dooren, J. Vandewalle, 'A systolic array for SVD updating'. *SIAM J. Matrix Anal. Appl.*, 1993, pp 353-371.

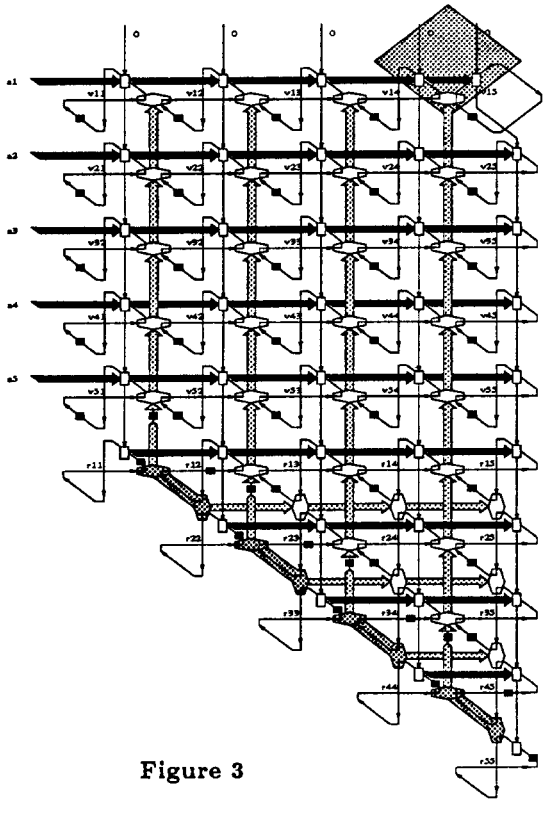

Figure 3

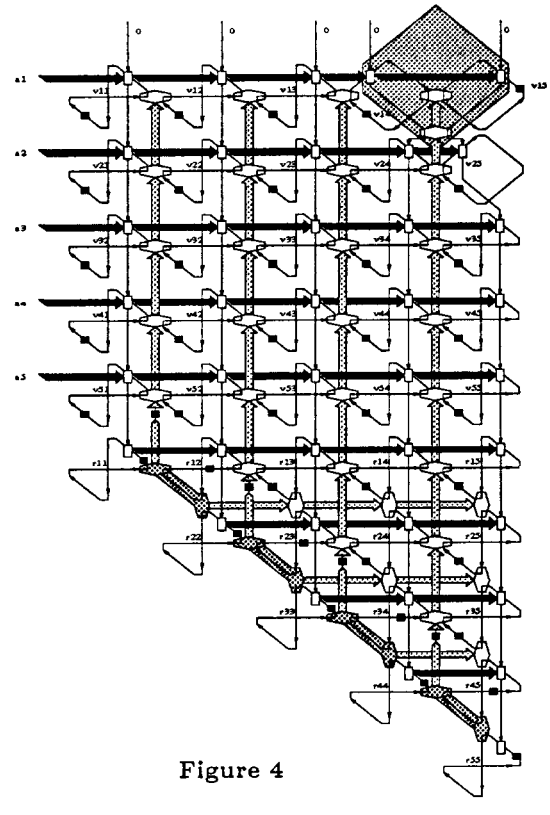

Figure 4

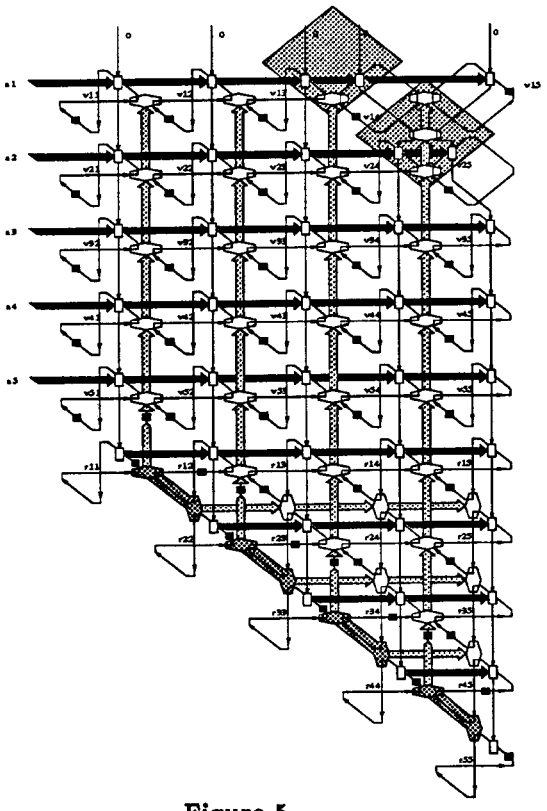

Figure 5

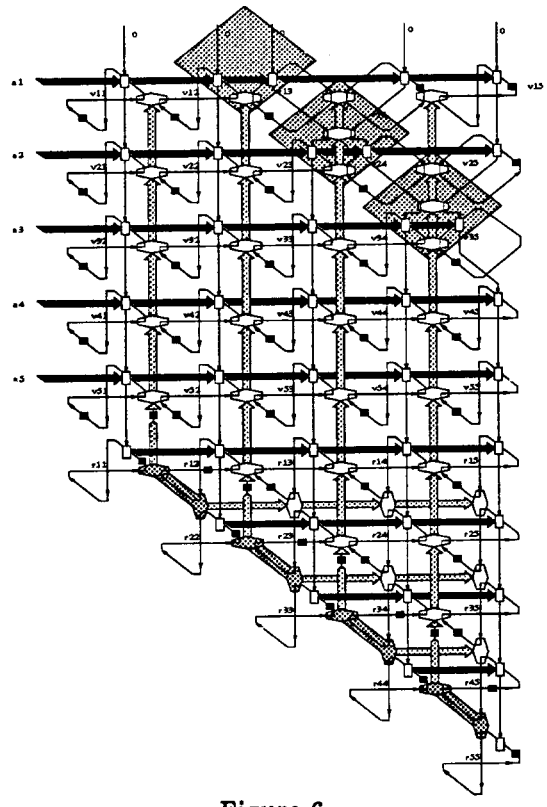

Figure 6

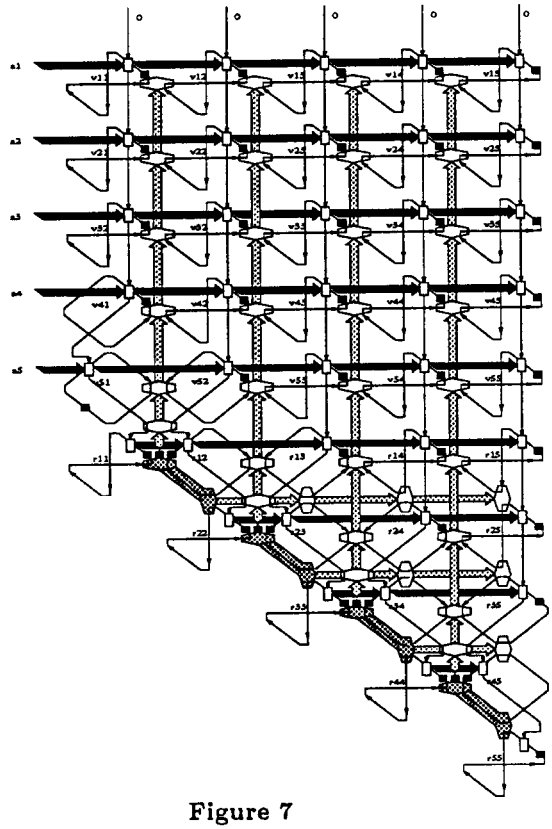

Figure 7

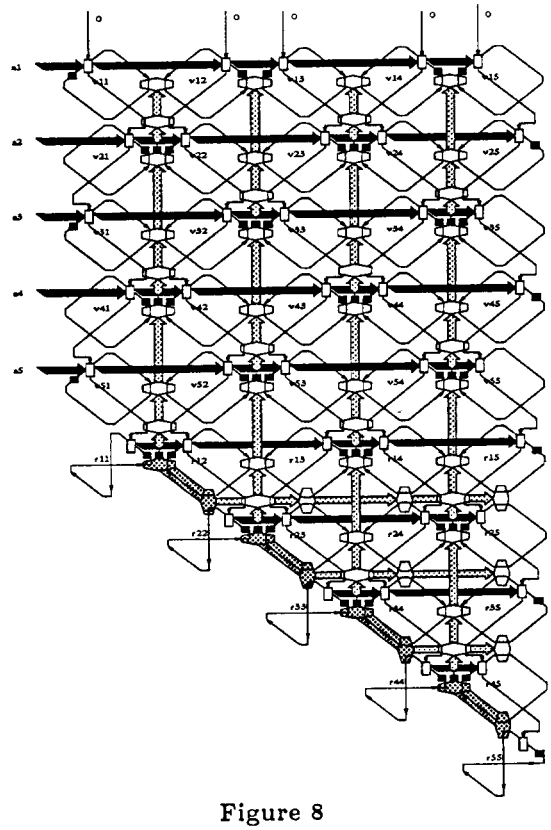

Figure 8