A REALIZABLE PARALLEL RLS PARAMETER ESTIMATOR

F.M.F. Gaston and D.W. Brown

Digital Systems and Vision Processing, University of Birmingham,
Edgbaston, Birmingham, B15 2TT, UK
Tel: +44 (0)121 414 4283; Fax: +44 (0)121 414 4291
e-mail: f.m.gaston@bham.ac.uk

ABSTRACT

In this paper we derive, from a dependence graph, a rectangular parallel architecture for RLS parameter estimation. It has a number of advantages over the traditional triangular structure whilst maintaining the same throughput. These advantages are identical cells, easy expandability for increases in the number of parameters, reduced data flow, useful data is easily extracted and all these properties together make it more attractive for VLSI implementation.

1 INTRODUCTION

Adaptive parameter estimation is a fundamental tool in both control and signal processing. Many applications require not only fast computation but also algorithms which are highly accurate and robust. Algorithms can lose numerical accuracy, and even fail altogether if the data is insufficiently exciting or there are significant, cumulative numerical errors due to limited arithmetic. Robust algorithms are usually more complex and therefore require more computations.

For these reasons, fast implementations have been sought using modern technology with faster processors and also parallel processing. Recursive least squares (RLS) parameter estimation was shown to be parallelizable in [7]. A triangular array of simple processing cells was proposed, where the cells are connected to their nearest neighbours and the data is clocked through. The processing cells use floating point arithmetic and theoretically data can be accepted on every clock cycle. Even though these architectures were proposed over a decade ago they have not yet been implemented on VLSI hardware, primarily because of the complexity of the floating point arithmetic required. Kadlec, in [5] proposed a fixed point architecture based on the triangular floating point array making VLSI implementation more feasible. However, other disadvantages of the triangular array then become significant. The triangular array has two types of processing cell with different functionality and different data flow, and the size of the array required is dependent on the order of the parameter vector. This structure makes adaption to the problem order difficult and VLSI implementation more inefficient.

In this paper we will develop a dependence graph [4] for the RLS parameter estimation algorithm and from this generate a rectangular architecture. This will have the following advantages over the more traditional triangular structure:-

- All the cells are identical.
- All the interesting data is available, none is locked in the memory of the cells.
- The rectangular architecture is attractive for VLSI implementation.
- Because it is rectangular it is also easily expandible for larger numbers of parameters.
- The rotation parameters are stored in the memory of the cells and so the amount of data passed between cells is reduced. Instead of 2 rotation parameters being passed, only the updated elements of *R* are passed.

This architecture can be readily extended to implement the regularised parameter estimator [6] and other algorithms based on recursive least squares such as the Kalman filter.

2 RLS PARAMETER ESTIMATION

In this paper, we shall consider a model:

$$y(k) = \underline{\theta}^T(k)\,\underline{\varphi}(k) + e(k) \qquad 2.1$$

where k is discrete time, the n-vector $\underline{\varphi}(k)$ is the data regressor, $y(k)$ is the output signal, $\underline{\theta}(k)$ is the n-parameter vector to be estimated and $e(k)$ is assumed to be white noise.

The parameter estimate $\hat{\theta}(k)$ can be estimated using the RLS 'square-root' algorithm which is defined in equations 2.2 and 2.3.

$$Q^T(k)\begin{bmatrix}\beta(k)R(k-1) & \beta(k)\underline{r}(k-1)\\ \underline{\varphi}^T(k) & y(k)\end{bmatrix} = \begin{bmatrix}R(k) & \underline{r}(k)\\ \underline{0}^T & \alpha(k)\end{bmatrix} \qquad 2.2$$

where $\beta(k)$ is the square-root of the forgetting factor, $\alpha(k)$ is related to the *a posteriori* least squares residual at time k [7], which in many applications such as adaptive beamforming may be more important than the actual parameter estimates. $R(k)$ is referred to as the Cholesky factor of the information matrix and $r(k)$ is defined as:

$$r(k) = R(k)\hat{\theta}(k) \qquad 2.3$$

3 RLS DEPENDENCE GRAPH REPRESENTATION

A dependence graph (DG) may be considered as a directed graph (digraph), where the ordering implies direction and is denoted by an arrow [4]. A graph is built up using two basic elements:

1. *Nodes* - Instantaneous cells for evaluating simple computations.
2. *Arcs* - Links that connect nodes for data communication.

It follows that a data dependence graph gives the designer a global view of the computation. However, there may be more than one possible DG for an algorithm and therefore if the DG originally derived is unsatisfactory then further modifications should be applied in order to achieve a better one since the structure of a DG greatly affects the final array design. In large order systems, the number of interconnections can become unmanageable so it is more convenient to use a hierarchical dependence graph (HDG) in place of the original DG. This is simply a more compact representation of the same structure. For example, the HDG representation of equation 2.2 is given in Figure 1 where the inputs and outputs are denoted as complete matrices and all detail relating to the underlying nodal dependence structure has been suppressed. In this context, Figure 1 must be regarded as a convenient hierarchical representation of Figure 2 and not just as an arbitrary description of RLS. Using standard Givens rotations to generate the orthogonal matrix results in the node descriptions defined in Figure 3.

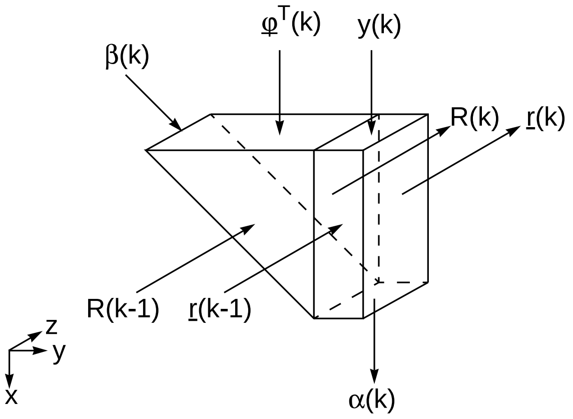


Figure 1 HDG for RLS update

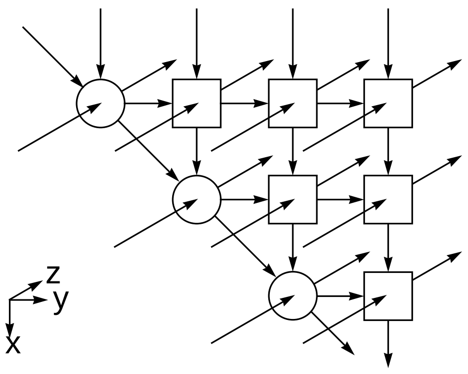
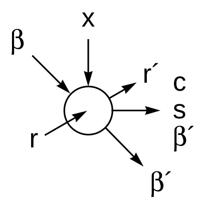


Figure 2 Dependence graph nodal structure

Having performed the update step detailed in equation 2.2, it only remains to calculate the RLS parameter estimate, $\hat{\underline{\theta}}(k)$, using equation 2.3. This weight flushing calculation shall not be considered in the paper due to page restrictions but can be calculated using the Faddeev algorithm [2].

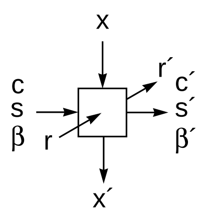
In reality the RLS parameter estimate may not be calculated every timestep, rather it is much more computationally attractive to update the upper triangular information matrix R, and corresponding vector $\underline{r}$, with successive data inputs and only calculate the parameter estimate when required, e.g. once every N iterations. This type of approach reduces the latency introduced by weight flushing every iteration. The HDG representation of N successive updates is shown in Figure 4.

BOUNDARY



$r' = \sqrt{(\beta^2 . r^2 + x^2)}$
$c = \beta . r / r'$
$s = x / r'$
$\beta' = \beta$

INTERNAL



$r' = \beta . c . r + s . x$
$x' = c . x - \beta' . s . r$
$c' = c$
$s' = s$
$\beta' = \beta$

Figure 3 Node descriptions for RLS update

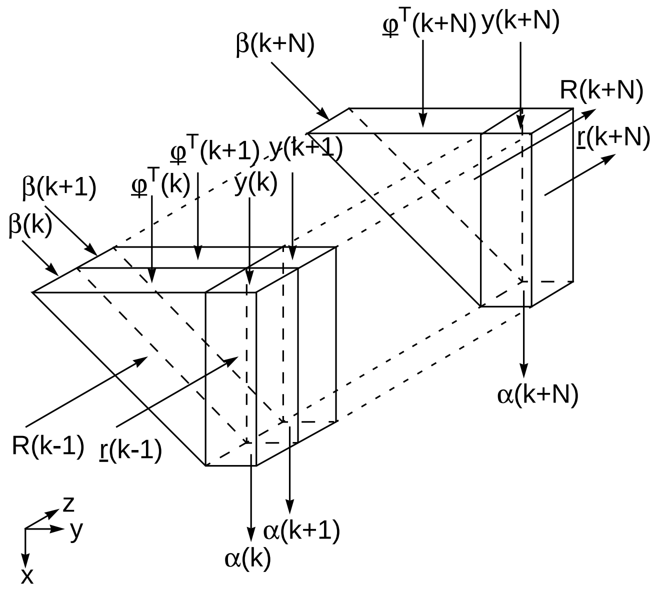


Figure 4 *N* iterations of the RLS update algorithm

4 GENERATING A REALIZABLE SYSTOLIC ARCHITECTURE

Fast implementations have been sought for the RLS algorithm using modern technology with faster processors and also parallel processing. RLS parameter estimation was shown to be parallelizable by McWhirter [7] using the famous triangular array of Gentleman and Kung [3]. This triangular array, given in Figure 5, can easily be derived using the HDG of Figure 4 by scheduling, projecting along the z-axis, and pipelining [1]. The triangular array has two types of processing cells with different functionality and different data flow. The systolic cell descriptions can be obtained directly from Figure 3. The size of the array required is known to depend on the order of the parameter vector. This structure makes adaption to the problem order difficult and VLSI implementation more inefficient.

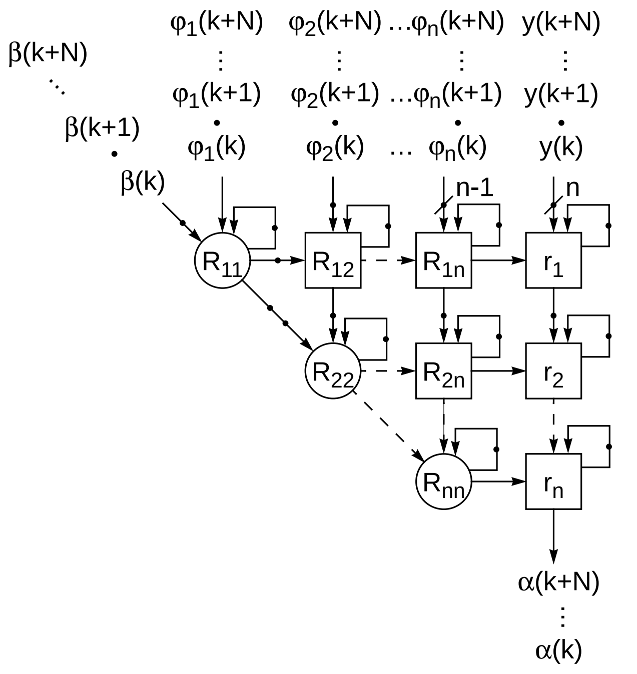


Figure 5 Existing triangular architecture

From the HDG in Figure 4, a rectangular array for RLS parameter estimation can be developed by scheduling and projecting along the y-axis, as illustrated in Figure 6. It has all the advantages of the triangular array, i.e. same processing speed and fixed point implementation properties, but it also has the additional advantage that every cell is identical, as defined in Figure 7. It is therefore easily expandible for any problem order and its rectangular shape is much more attractive for VLSI implementation, since it is more likely to lead to an efficient use of silicon area. Only the rotation parameters are stored in the memory of the cells and all the interesting data is available, none is locked in the memory of the cells. Lastly, the amount of data passed between cells is reduced; instead of 2 rotation parameters being passed, only the updated elements of *R* are passed.

When the *R* matrix reaches the final processing cells of the array (the *N*th column) it is fed back to the first column of cells. Data can be controlled by either latches, or hand-shaking. The feedback loops can be implemented in VLSI hardware by placing them in a separate layer to the computation elements thus ensuring equal lengths to avoid clock skew problems. If $N = n + 1$, then the top row of cells are in use all the time, however the bottom row are only in use 2 cycles out of $n + 1$. If n is large then the efficiency will become 50%. With this structure we could go into three dimensions by folding the columns around so that there were no feedback loops and all the connections were the same length, thus avoiding the need for extra latches, or buffers. In this case the R matrix rotates continuously around and data pours in at the top of the array and any interesting data can be extracted at any point.

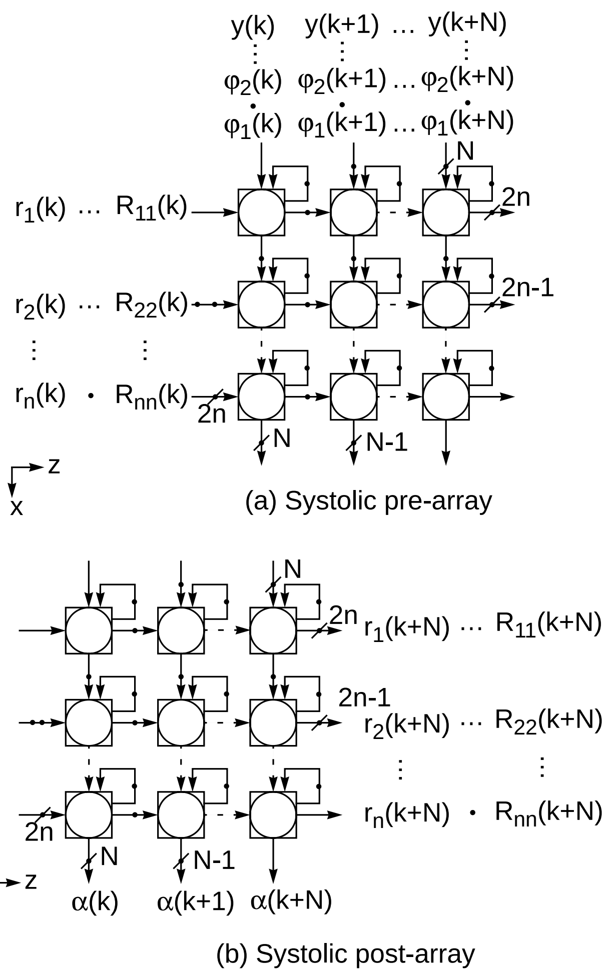
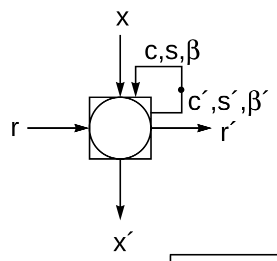


Figure 6 Rectangular architecture

5 CONCLUSIONS

In this paper we have derived, from a DG, a rectangular parallel architecture for RLS parameter estimation. It has a number of advantages over the traditional triangular structure while maintaining the same throughput. These advantages are identical cells, easy expandability for an increase in the number of parameters, reduced data flow, useful data is easily extracted and all these properties together make it more attractive for VLSI implementation.

We gratefully acknowledge the support of the Defence Research Agency, Malvern.



Mode 1 (boundary)
$r' = \sqrt{(\beta^2 . r^2 + x^2)}$
$c' = \beta . r / r'$
$s' = x / r'$
$\beta' = \beta$

Mode 2 (internal)
$r' = \beta . c . r + s . x$
$x' = c . x - \beta . s . r$
$c' = c$
$s' = s$
$\beta' = \beta$

Figure 7 Systolic cell descriptions for rectangular RLS

REFERENCES

[1] Brown, D.W. and Gaston, F.M.F. "The design of parallel square-root Kalman filters using algorithm engineering", Integration, the VLSI Journal, Elsevier Science Publishers, vol. 20, pp. 101-119, 1995.

[2] Faddeev, D.K. and Faddeeva, V.N. "Computational methods of linear algebra", New York: W.H.Freeman and Co., 1963.

[3] Gentleman, W.M. and Kung, H.T. "Matrix triangularisation by systolic arrays", Proc. SPIE, vol. 298, Real Time Signal Processing IV, pp. 19-26, 1981.

[4] Jean, J.S.N. and Kung, S.Y. "Array compiler design for VLSI/WSI systems", Transformational Approaches to Systolic Design, ed. G.M. Megson, Chapman & Hall, London, pp. 133-157, 1994.

[5] Kadlec, J., Gaston, F.M.F. and Irwin, G.W. "Implementation of the regularised parameter estimator", Proc. IEEE Workshop on VLSI and Signal Processing, Kung Yao et al. (eds.), IEEE Special Publications, NY., pp. 520, 1992.

[6] Kadlec, J., Gaston, F.M.F. and Irwin, G.W. "The block regularised parameter estimator and its parallelisation.", Automatica, vol. 31, no. 8, pp. 1125-1136, 1994.

[7] McWhirter, J.G. "Recursive least squares minimisation using a systolic array", Proc. SPIE (Real time and Signal Processing IV), vol. 431, pp. 105-112, 1983.