4.1 Matrix Manipulations

Matrix manipulations belong to the most important basic operations of digital signal processing. Filtering, computation of convolution, linear and nonlinar transformations all use some matrix operations, like multiplications of matrices, matrix-vector multiplications, matrix additions, or more complex operations as eigenvalue decomposition, singular value decomposition, etc. The purpose of this section is only to show that all these operations are inherently parallel ones, they can be easily paralleised and realised by some parallel architecture. Instead of extensive discussion two examples will be presented only.

Matrix-vector Multiplication

Matrix-vector multiplication is defined by

$$\underline{c} = \underline{\underline{A}}\ \underline{b} \quad \text{or} \quad c_i = \sum_{j=1}^{q} a_{ij} b_j \tag{4.1}$$

The locally recursive counterpart of this equation is:

$$c_i^{(j+1)} = c_i^{(j)} + a_i^{(j)} * b_i^{(j)} \tag{4.2}$$

where:
$c_i^{(1)} = 0$
$a_i^{(j)} = a_{ij}$
$b_i^{(j)} = b_j$

The DG created from this equation is shown on figure 4.1.

Note that only dependences between nodes are shown in the figure. The operations inside each node are deliberately ignored in the DG, since they will be assigned to the same processing element when the DG is used to map an algorithm to an array processor. However it is easy to extend the DG concept to include the operations inside each node. In figure 4.1 only local communications are used, where the value of b(j)– felso index zarojelben– is propagated step by step, without being modified, to all the nodes with the same j-index. From DG we can get a one-dimensional SFG as it is shown on figure –ide egy abra kell es az arra valo hivatkozas.–and its corresponding systolic array, wherethe pipelining period is one.

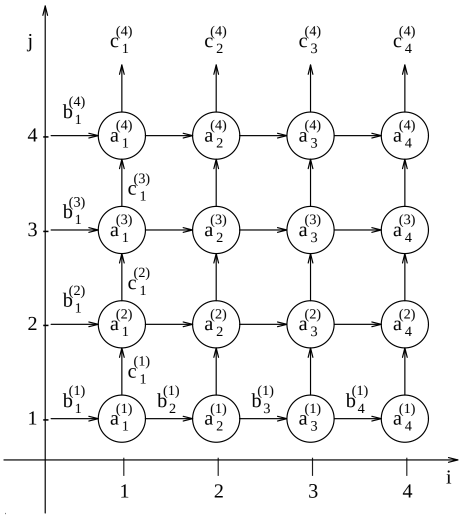

Figure 4.1: DG of matrix-vector multiplication with local communication

Squared Matrix Multiplication

Matrix multiplication for squered martices is given by:

$$\underline{\underline{C}} = \underline{\underline{AB}} \tag{4.3}$$

where

$$\underline{\underline{C}} : c_{ij} = \sum_{k=1}^{N} a_{ik} b_{kj} \;; \quad \underline{\underline{A}} : a_{ij} \;; \quad \underline{\underline{B}} : b_{ij}; i, j = 1..N \tag{4.4}$$

For band matrices (when there is an upper and a lower triangle filled with zeros) the multiplication results also band matrix, and the summation limits in equation 4.4 are smaller, usually much smaller than N. Let's analyse the systolic realization [Kung88] on figure 4.2.

The question is how such a scheme actually produces the desired multiplication results. The easiest way to answer this question is to display the space-time activities in the first few consecutive beats, and then derive the remaining activities by induction and inspection. In this example a two-dimensional hexagonal array is seen to be a natural topology for the band matrix multiplication problem. As shown in figure 4.2.

The question is how such a scheme actually produces the desired multiplication results. The easiest way to answer this question is to display the space-time activities in the first few consecutive beats, and then derive

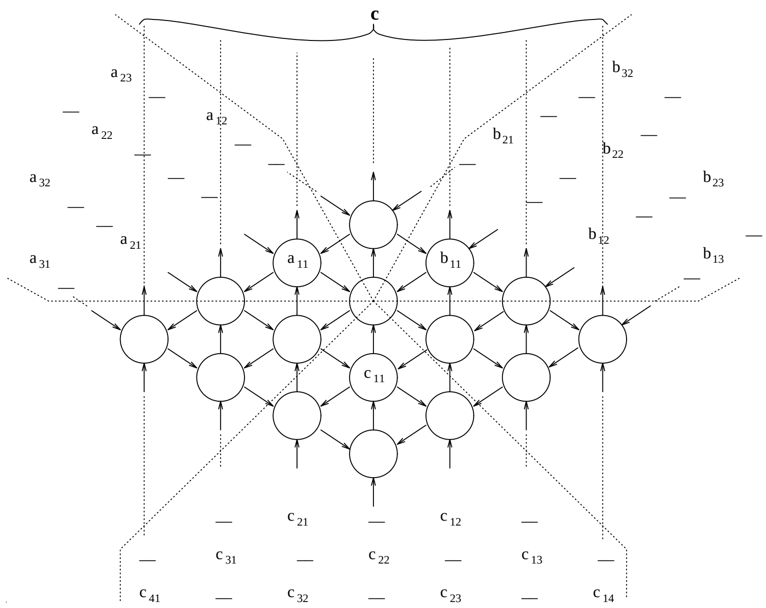

Figure 4.2: Band matrix multiplication with systolic array

the remaining activities by induction and inspection. In this example a two-dimensional hexagonal array is seen to be a natural topology for the band matrix multiplication problem. As shown in figure 4.2 the input data (from matrices $\underline{\underline{A}}$ and $\underline{\underline{B}}$)are prearranged in an olderly squence. The output data (of the matrix $\underline{\underline{C}}$) will be pumped from the other side of the array, meeting the right data and collecting all the desired products. The detailed description of data movements and computations is often furnished in terms of the snapshots of the activities.