MAPPING MULTIPLE ALGORITHMS INTO A RECONFIGURABLE SYSTOLIC ARRAY

[1]*Wei Jin,* [1]*Chang N. Zhang and* [2]*Hua Li*

[1]Department of Computer Science, University of Regina
Regina, SK S4S 0A2, Canada
[2]Department of Mathematics and Computer Science, University of Lethbridge
Lethbridge, AB T1K 3M4, Canada

ABSTRACT

Systolic array is a well known VLSI architecture to achieve extensive parallel and pipelining computing. Many systolic designs have been reported. All are algorithm based, that is one design is only for solving one specific problem. In this paper, the special purpose systolic architecture has been extended into a reconfigurable one and a systematic design approach to mapping two or more algorithms into a single reconfigurable systolic array is presented. First multiple algorithms are mapped into a reconfigurable systolic array that is able to compute one algorithm at a time with proper control settings. Second the reconfigurable systolic array is extended by using time or space redundancy so that it can compute multiple algorithms simultaneously. In addition, the optimal mapping, which minimizes the total hardware cost and computation time, is explored and the necessary condition of the transformation for computing multiple problem instances is also proposed. According to this condition, the search space of finding the optimal mapping can be significantly reduced.

Index Terms— systolic array, reconfigurable architecture, space-time mapping, systematic approach, optimal design

I. INTRODUCTION

The development of VLSI has a great impact on the design of application specific integration circuits. Now we can implement specific algorithms on a VLSI chip using multiple regularly connected processing elements (PE) to achieve parallel computing and pipelining. Due to the simple control and scheduling, local interconnection and modular designs, systolic arrays have received much attention and are preferred architectures for computationally extensive problems [1].

Many systolic array designs have been reported in the past several decades and some are equipped with fault tolerance [2-6]. However, all these designs are algorithm-based, which means one design is for solving one problem. Although the design in [8] could compute multiple problem instances, that design only works for problems with the same data dependency. Previous studies [4] developed a systematic approach to mapping one algorithm into a systolic array with fault tolerance, and the fault tolerance is achieved by time/space redundancy. Actually, the redundant version of the same algorithm instances for fault tolerance could be replaced by some new algorithm instances. In this paper, we propose a reconfigurable systolic array architecture for mapping multiple problems into a single reconfigurable systolic array according to the space-time mapping approach [4, 7]. The mapping procedure is based on a linear transformation of index space and data dependency. Our idea is that we first map different algorithms into different systolic arrays using the same transformation matrix; second, we extract the common part of the circuit design within PEs and then merge these systolic arrays into a reconfigurable one so that the common part can be shared as much as possible. Our design can compute different algorithms separately or simultaneously by reconfiguration and data input settings. Moreover, the optimal mapping is explored, which minimizes the total hardware cost and computation time.

The rest of this paper is organized as follows. Section 2 briefly reviews the space-time mapping approach and gives the performance evaluation rules. Section 3 introduces our design for mapping multiple problems into one reconfigurable systolic array. Section 4 gives the necessary condition of the valid transformation matrices, which are able to map multiple algorithms into a single systolic array. A detailed example is also presented. Section 5 concludes this paper.

II. SPACE TIME MAPPING

We usually describe the nested loop algorithms with regular data dependency and constant bounded index sets as follows: $for\ i_1\ :=\ v_1\ to\ l_1\ do$
$for\ i_2\ :=\ v_2\ to\ l_2\ do$
:
:
$for\ i_p\ :=\ v_p\ to\ l_p\ do$
$begin$

978-1-4244-1643-1/08/$25

$statement_1$;
$statement_2$;
:
:
$statement_n$;
end

where v_r and l_r, $1 \leq r \leq p$, are integers. The algorithm should be a single assignment algorithm [3], which means that in each statement a single value is computed for each occurrence of each indexed variable. Thus the data dependencies in the statements can be expressed by an integer matrix $D = (d_1, d_2, ..., d_m)$ [7], where d_i is a $p \times 1$ integer vector.

A p-nested loop structure with constant data dependencies can be represented by a pair (D, C_D), in which D is the data dependency matrix, and $C_D = \{(i_1, i_2, ..., i_p)^{tr} : 1 \leq i_1 \leq l_1, 1 \leq i_2 \leq l_2, ..., 1 \leq i_p \leq l_p\}$ is the index space (tr means transposes). A systolic array implementation of an algorithm can be obtained by a linear transformation (p $\times$ p matrix) T = $\begin{bmatrix} \pi \\ S \end{bmatrix}$, where π is a 1 $\times$ p vector (the first row of T) determining time scheduling, and S (the rest rows of T) is a (p - 1) $\times$ p submatrix, the space transformation, which maps C_D onto a (p-1)-dimensional systolic array.

Consider a valid mapping $T \times (D, C_D) = (\hat{D}, \hat{C}_D)$, which maps a 3-loop algorithm into a 2D systolic array (mapping a 2-loop algorithm into a 1D array is similar). T is a 3 $\times$ 3 transformation matrix; D is the data dependency matrix; C_D is the index space $\{(i, j, k) : 1 \leq i \leq l_1, 1 \leq j \leq l_2, 1 \leq k \leq l_p\}$; $\hat{C}_D = \{(t, x, y), (t, x, y)^{tr} = T(i, j, k)^{tr}\}$, where t is the time when a computation happens and (x, y) is the coordinate of the PE where the computation happens. In particular, $(t_1, x_1, y_1)^{tr} = T(1, 1, 1)^{tr}$ is the reference point. The first row of $\hat{C}_D$ indicates scheduling and the last two rows indicate the inter processor communication. A valid T must be nonsingular and all the elements of the first row of $\hat{C}_D$ must be positive.

We use matrix multiplication to illustrate this mapping approach.

Algorithm 1 (matrix multiplication: $C_{l_1 l_2}$ = $A_{l_1 l_3}$ $\times$ $B_{l_3 l_2}$)
$for\ i := 1\ to\ l_l\ do$
$for\ j := 1\ to\ l_2\ do$
$for\ k := 1\ to\ l_3\ do$
$begin$
$a(i, j, k) := a(i, j - 1, k)$;
$b(i, j, k) := b(i - 1, j, k)$;
$c(i, j, k) := a(i, j, k - 1) + a(i, j, k)b(i, j, k)$;
end

where a(i, 0, k) = a_{ik}, b(0, j, k) = b_{kj}, c(i, j, 0) = 0, and c(i, j, l_3) = c_{ij} for all i, j and k.

The data dependency in this algorithm is D = $\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ and C_D = (1, 1, 1)^{tr}, (1, 1, 2)^{tr}, ... ,(l_1, l_2, l_3)^{tr}. Let l_1 = l_2 = l_3 = 3; the resulting systolic arrays obtained by T_1 = $\begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \end{bmatrix}$ and T_2 = $\begin{bmatrix} 1 & 1 & 1 \\ -1 & 1 & 0 \\ 0 & -1 & 1 \end{bmatrix}$ are shown in Fig. 1 and Fig. 2.

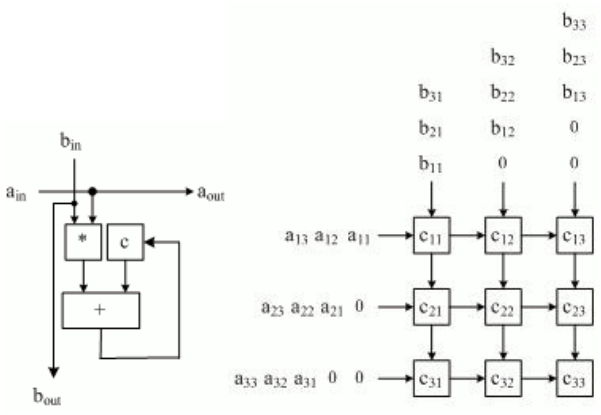

Fig. 1. Systolic array obtained by T_1

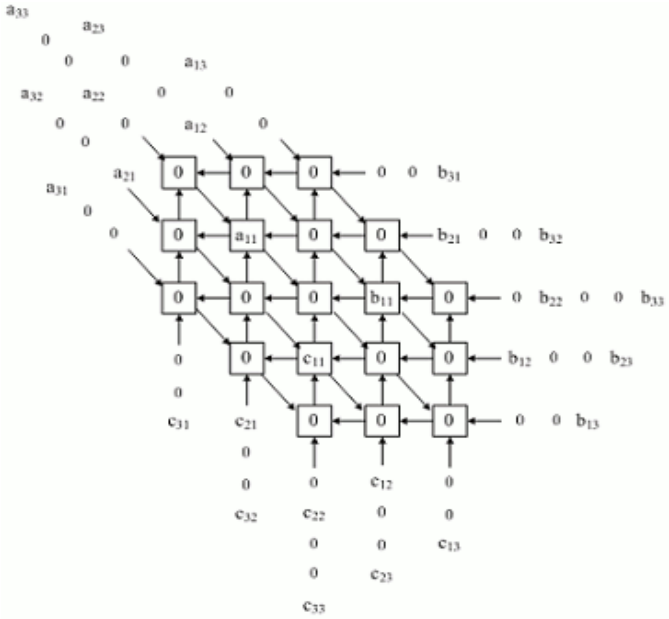

Fig. 2. Systolic array obtained by T_2

We see from this example that different transformation matrices result in different systolic arrays. Generally, for a given algorithm A, there may exist an infinite number of valid transformation matrices. Two factors that determine the performance of a systolic array are the total number of PEs and the total computation time. Studies in [4][9] found that these two factors depend only on the transformation matrix and the length of the loop. The results are showed below.

Definition 1 [4]: The number of PEs, n_p, is the number of

different elements in the set {(x, y)}, which can be computed by the following theorem.

Theorem 1 [4]: The number of PEs is $l_1 l_2 l_3$ if $|a_j| \geq l_j$ for some j = 1, 2, 3, and $l_1 l_2 l_3 - (l_1 - |a_1|)(l_2 - |a_2|)(l_3 - |a_3|)$ otherwise, where $a_j = \frac{T_{1,j}}{GCD(T_{1,1},\ T_{1,2},\ T_{1,3})}$ and $T_{1,j}$ is the (1,j)-cofactor of matrix T, $1 \leq j \leq 3$.

Definition 2 [4]: The computation time, t_c, is the number of clock cycles required to compute one problem instance in a systolic array $t_c = max\{t,\ (t,x,y) \in C_D\} - min\{t,\ (t,x,y) \in C_D\} + 1$ that can be calculated by $t_c = (l_1 - 1)|t_{11}| + (l_2 - 1)|t_{12}| + (l_3 - 1)|t_{13}| + 1$

Theorem 2 [9]: If $T(D, C_D) = (\hat{D}, \hat{C}_D)$ and $0 \leq l_{j_2}, l_{j_3} \leq l_{j_1} (j_1, j_2, j_3\ is\ 1,\ 2,\ 3\ in\ some\ order)$, then the minimum number of PEs is $l_{j_2} l_{j_3}$ and is realized by a space-time transformation $T = \begin{bmatrix} t_{11} & t_{12} & t_{13} \\ t_{21} & t_{22} & t_{23} \\ t_{31} & t_{32} & t_{33} \end{bmatrix}$ if and only if $t_{2j_1} = t_{3j_1} = 0$.

The optimal transformation matrix for a given algorithm should first be nonsingular for conflict free, otherwise, two index points may be mapped into one PE at the same time; second, to reduce computation time, the absolute values of the numbers in the first line of T that determine schedule should be as small as possible; third, to maintain causal time, all the numbers of the first line of TD must be greater than 0; fourth, the transformation matrix should be in the form of $\begin{bmatrix} t_{11} & t_{12} & t_{13} \\ 0 & X & X \\ 0 & X & X \end{bmatrix}$ ((t ,0 ,0)^{tr} could be in any column) according to theorem 2.

III. MAPPING MULTIPLE ALGORITHMS INTO ONE RECONFIGURABLE SYSTOLIC ARRAY

Reconfigurable computing has become a popular research area in the past years because it combines the performance advantage of static hardware functionality and the flexibility of software programmable substrates [10]. The term reconfigurable refers to the computing method characterized by the ability to change the hardware architecture, including logic functionality and interconnect of a computing system, during algorithm execution. There are several reconfigurable architecture designs concerning systolic array in the literature. All of these designs are algorithm-based. [11] proposed a reconfigurable design approach for computing a specific problem, either combinatorial optimization or image filtering, featured a dynamic scheduling for nested loops. [12] discussed a reconfigurable systolic architecture based on BP networks. Both of their reconfigurations are about the specific problem's scale. In this paper, the proposed reconfiguration is for computing multiple problems

Definition 3: A reconfigurable systolic array is a systolic array that can be adapted to computing multiple problems through either software or hardware configuration, or both.

The reconfiguration of our design is achieved by adding some switches, multiplexers or/and demultiplexers within PEs or/and along the data path between PEs. We use two 2-loop algorithms and two 3-loop algorithms to demonstrate our design approaches. Suppose we are given a bunch of 3-loop or 2-loop algorithms $\{D_1,\ D_2,\ ...\ ,\ D_n\}$. We can choose one transformation matrix T to map these algorithms into different systolic arrays $\{\hat{D}_1,\ \hat{D}_2,\ ...\ ,\ \hat{D}_n\}$, and then merge those arrays together with some control settings to obtain a general one. Besides the individual hardware cost n_p and computation time cost t_c mentioned in the last section, we also need to consider the circuit sharing within a PE and the inter communication between PEs. The circuit design within a PE is based on the real computation statements of the algorithm, usually including addition, multiplication, logic, and shift operations. The inter communication pathes between PEs is the data transmission path. So the last 2 rows of matrices $\{\hat{D}_1,\ \hat{D}_2,\ ...\ ,\ \hat{D}_n\}$ should have the same vectors as much as possible. Hence the data path circuit could be shared as much as possible.

Consider two 3-loop algorithms: algorithm 1 and the following algorithm.

Algorithm 2 $for\ i := 1\ to\ l_1\ do$
$for\ j := 1\ to\ l_2\ do$
$for\ k := 1\ to\ l_3\ do$
$begin$
$x(i,j,k) := x(i,j-1,k);$
$y(i,j,k) := y(i-1,j,k) + x(i,j,k);$
$z(i,j,k) := y(i,j,k) * z(i-1,j-1,k-1);$
end

where inputs are $x(i,0,k) = x_{ik}$, $y(0,j,k) = y_{jk}$, $and\ z(i,j,k)$ if one of i, j, k is 0, and outputs are $y(i,j,l_3)$ for all i, j and $z(i,j,k)$ if one of i, j, k reaches the index bound.

The data dependency matrix D is $D = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}$. Suppose we use the same transformation matrix $T_1 = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \end{bmatrix}$. $\hat{D} = T_1 D = \begin{bmatrix} 1 & 1 & 3 \\ 1 & 0 & 1 \\ 0 & -1 & -1 \end{bmatrix}$. Let $l_1 = l_2 = l_3 = 3$. The resulting systolic array is shown in Fig. 3.

We need to merge the systolic arrays in Fig. 1 and Fig. 3 to obtain a single reconfigurable one. Since the circuit connections of the two designs within a PE are quite different, we need several multiplexers and demultiplexers to control the data flow, and thus to achieve reconfigurable ability. Both $\hat{D}$ of the two algorithms have the vectors (1, 0) and (0, -1), the horizontal and the vertical data path between PEs can be perfectly shared. The data along diagonal path can be controlled by the demultiplexer. The array structure is as same as the one in Fig. 3. The detailed design within a PE is shown in Fig. 4. If the control signals e=0, then the multiplexers select the left pin as the input and the demultiplexers forward the input to the left output pin, then the PE is used for computing algorithm 2 and e=1 controls

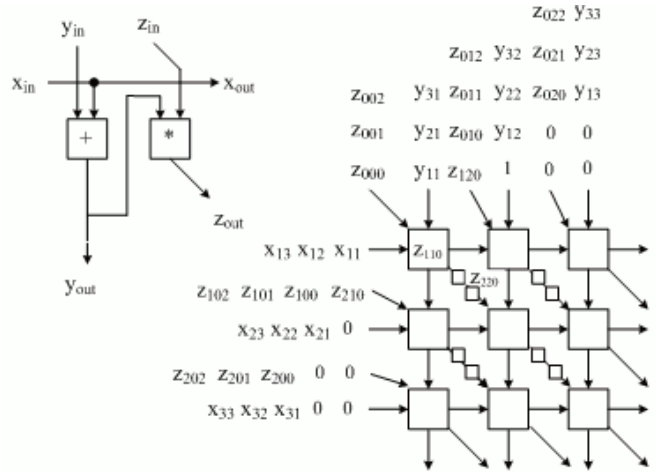

Fig. 3. Systolic array for Algorithm 2

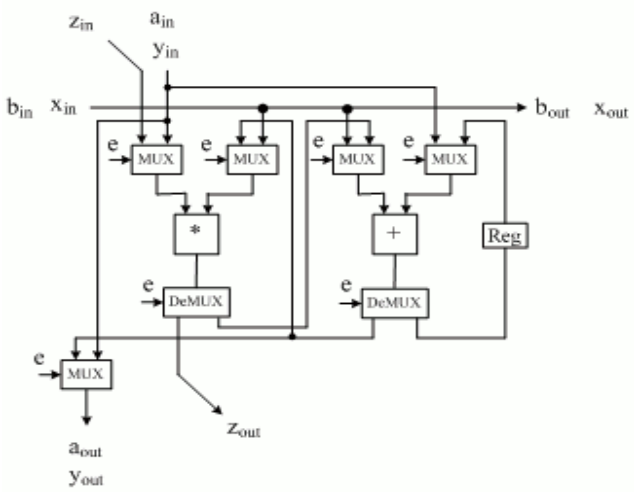

Fig. 4. The inner PE design of merging Algorithm 1 and Algorithm 2

the PE to compute algorithm 1.

From the above example, we can have the following observations and facts: 1) Multiple algorithms: there should be some common arithmetic operations among the target algorithms so that the circuits can be shared in the resulting systolic array; 2) Optimal reconfigurable systolic array: the different transformation matrices will result different reconfigurable systolic arrays in terms of the number of PEs, total computation time, connections between PEs and a number of control signals. In order to minimize the total computation time and hardware cost, a general search process of finding an optimal transformation matrix should be conducted. One guide for the search, as mentioned in the last section, is that the matrix should be in the form of $\begin{bmatrix} t_{11} & t_{12} & t_{13} \\ 0 & X & X \\ 0 & X & X \end{bmatrix}$ ((t ,0 ,0)^{tr} could be in any column).

IV. COMPUTING MULTIPLE ALGORITHMS SIMULTANEOUSLY ON A SINGLE SYSTOLIC ARRAY

The above example shows that the reconfigurable systolic array can be used for computing different algorithms with proper inputs and control signal settings once at a time. Now we want to compute two or more problems simultaneously. This can be realized by selecting a transformation matrix which leads to double or multiply the pipelining time period or space utilization. Then we can insert another computing instance at every second idle time or at the neighbor idle PE.

Definition 4 [4]: The processor pipelining time period, t_p, indicates the pipeline rate of the data paths and the throughput of the array. $t_p = 1$ indicates the full computation rate. If $t_p > 1$, then every PE in the array is active at one of t_p cycles. It can be determined by the following formula:$t_p = \frac{DET(T)}{GCD(T_{1,1},\ T_{1,2},\ T_{1,3})}$, where $T_{i,j}$ is the (i, j)-cofactor of matrix T.

Definition 5 [4]: Space utilization u_x and u_y: The definition of space utilization is similar to that of the processor pipelining time period. The space utilization of a systolic array with respect to x is u_x if and only if there are $u_x - 1$ idle PEs between any two nearest active PEs in the x direction for every clock time. And u_y is defined similarly. They can be calculated by the following formulas: $u_x = \frac{DET(T)}{GCD(T_{2,1},\ T_{2,2},\ T_{2,3})}$, $u_y = \frac{DET(T)}{GCD(T_{3,1},\ T_{3,2},\ T_{3,3})}$, where $T_{i,j}$ is the (i, j)-cofactor of matrix T. Theorem 3: Suppose $T(D_1, C_{D_1}) = (\hat{D_1}, \hat{C_{D_1}})$, $T(D_2, C_{D_2}) = (\hat{D_2}, \hat{C_{D_2}})$, ..., $T(D_L, C_{D_L}) = (\hat{D_L}, \hat{C_{D_L}})$, then T is a valid transformation for computing L algorithms simultaneously if $max\{t_p, u_x, u_y\} \geq L$.

Proof: Suppose $max\{t_p, u_x, u_y\} = L$ (the other cases are similar). Since $t_p = L$, according to the definition of t_p, every PE in the array is active at one of t_p cycles for computing each algorithm. Then we can construct L systolic arrays as follows: let $\hat{C_{D_1}} = \{(t, x, y)\}$, $\hat{C_{D_2}} = \{(t+1, x, y)\}$, ..., $\hat{C_{D_L}} = \{(t+L-1, x, y)\}$. These L systolic arrays differ from the computation time, which means the same index point of the original index space, say (i, j, k) can be mapped into $(t, x, y), (t+1, x, y), ..., (t+L-1, x, y)$ respectively by T. Therefore if we merge $\hat{C_{D_1}}$, $\hat{C_{D_2}}$, ..., $\hat{C_{D_L}}$ together and arrange the inputs for each index point of each algorithm one after another, the merging systolic array can compute L algorithms simultaneously.

Theorem 3 gives the necessary condition of the transformation matrices which are able to map L algorithms into a single reconfigurable systolic array. This result can be used to reduce the search space of transformation matrices significantly.

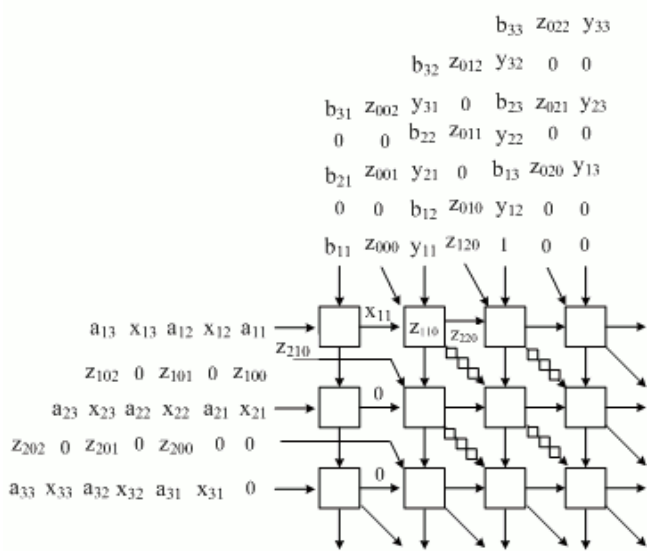

Fig. 5. Computing algorithm 1 and 2 simultaneously using redundant space utilization in x direction

We can choose $T_3 = \begin{bmatrix} 1 & 1 & 2 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \end{bmatrix}$, then $t_p = u_x = 2$; The systolic array using space redundancy is shown in Fig. 5, and the inner PE circuit is as same as Fig. 4. Note that T_1 is different from T_3 by changing t_{13} from 1 to 2; so the delay along data pathes is increased correspondingly. It takes 7 time units to compute either algorithm 1 or algorithm 2 once at a time. But the array in Fig. 5 just takes 9 time units to finish the computations simultaneously at the cost of adding one more column PEs. Generally, given L algorithms, each needs t_{ci} time units; then it takes $Max\{t_{ci}, 1 \leq i \leq L\} + t_p$ time units to do the computations simultaneously using space redundancy.

V. CONCLUSION

In this paper, we proposed a systematic design approach for mapping multiple problems into one reconfigurable systolic array. To demonstrate our approach, we map two 3-loop algorithms to two 2D systolic arrays and merge these two arrays to a reconfigurable one. With some control settings, our design can compute more than two problems once at a time or simultaneously. The optimal mapping which minimizes the total hardware cost and computation time is explored. The necessary condition of the valid transformation matrices that can map multiple algorithms into a single systolic array is also presented. The main advantage of our approach is that because of the merging step, some circuits can be shared by multiple algorithms within or between PEs; hence the complexity of circuit is reduced. In addition, although the reconfiguration in our examples is realized by hardware, if we give more powerful computation ability to the PEs and use some software reconfigurations, our approach can be used to design systolic arrays for computing more complicated algorithms. The limitation is that it required all the algorithms have somewhat similarity, such as similar logic/arithmetic operations (the statements within loops for real computation) and/or similar data flows (the data dependencies). Otherwise there is no much benefit for the merging. Also we only discussed 3-loop in this paper. How to map multiple algorithms with higher dimensional index space into a two dimensional reconfigurable systolic array is to be studied in the future.

VI. REFERENCES

[1] H.T. Kung, Why Systolic Architectures, IEEE Computer, Vol. C-31, pp.37-46, 1982

[2] S.Y. Kung, S.C. Lo and P.S. Lewis, Optimal systolic design for the transitive closure and the shortest path problems, IEEE transactions on computer, Vol. C-36, No.6, May 1987

[3] S.Y. Kung, VLSI Array Processors, Prentice Hall, 1988

[4] C.N.Zhang, T.M. Bachtiar, and W.K. Chou, Optimal fault-tolerant design approach for VLSI array processors, IEE Proc. Comput. Digit. Tech., Vol.144, pp15-21, Jan. 1997

[5] S. Peng and S.G. Sedukhin, Design of optimal array processors for two-step division-free Gaussian elimination, IEICE Trans. Inf. & Syst., VOL.E82-D, No.12, December 1999

[6] N.M. Stojanovic, E.I. Milovanovic, I. Stojmenovic, I.Z. Milovanovic and T.I. Tokic, Mapping Matrix Multiplication Algorithm onto Fault-Tolerant Systolic Array, Computers and Mathematics with Applications 48, pp.275-289, 2004

[7] Moldovan. D.I., On the design of algorithms for VLSI systolic arrays, Proc. IEEE,71:113-120, 1983.

[8] C.N. Zhang Systematic design of systolic arrays for computing multiple problem instances, Microelectronics Journal, 23, pp543-553, 1992.

[9] C.N. Zhang, J.H. Weston and H.Li, Some space considerations of VLSI systolic array mappings, Transactions on Circuits and Systems-II: Analog and Digital Signal Processing, Vol. 48, No. 4, pp.419-424, Apr. 2001

[10] Russell Tessier and Wayne Berleson, Reconfigurable Computing for Digital Signal Processing: A Survey, Journal of VLSI Signal Processing 28, 7-27, Kluwer Academic Publishers 2001

[11] Ioannis Panagopoulos, Christos Pavlatos, George Manis, George K. Papakonstantinou, A Flexible General-Purpose Parallelizing Architecture for Nested Loops in Reconfigurable Platforms. PATMOS 2007: 20-30

[12] Qin Wang, Ang Li, Zhancai Li, Yong Wan: A Design and Implementation of Reconfigurable Architecture for Neural Networks Based on Systolic Arrays. ISNN (2) 2006: 1328-1333

Intentional Blank Page