Design and Implementation of the Systolic Array for Dynamic Programming

Jae Jin Lee*, David Tien**, Gi Yong Song*

Abstract

We propose a systolic array for dynamic programming which is a technique for solving combinatorial optimization problems. We derive a systolic array for single source shortest path problem, SA SSSP, and then show that the systolic array serves as dynamic programming systolic array which is applicable to any dynamic programming problem by developing a systolic array for 0 1 knapsack problem, SA 01KS, with SA SSSP for a basis. In this paper, each of SA SSSP and SA 01KS is modeled and simulated in RT level using VHDL, then synthesized to a schematic and finally implemented to a layout using the cell library based on 0.35㎛ 1 poly 4 metal CMOS technology.

Key words : systolic array, dynamic programming

I. Introduction

Dynamic programming [1], a technique for solving combinatorial optimization problems, solves problems by combining the solutions to subproblems if an optimization problem has optimal substructure and overlapping subproblems property. We say that a problem exhibits optimal substructure if an optimal solution to the problem contains within it optimal solutions to subproblems. In order for dynamic programming to be applicable, a recursive algorithm for the problem solves the same subproblems over and over, rather than always generating new subproblems. When a recursive algorithm revisits the same problem over and over again, we say that the optimization problem has overlapping subproblems. Dynamic programming stated with recurrence relation exhibits regular and local data structure which is naturally compatible with systolic design.

A systolic array [2] [3] formed by interconnecting a set of identical data processing cells in a uniform manner is a combination of an algorithm and a circuit that implements it, and is closely related conceptually to arithmetic pipeline. In a systolic array, data words flow from external memory in a rhythmic fashion, passing through many cells before the results emerge from the array's boundary cell and return to external memory. Upon receiving data words, each cell performs same operation and transmits the intermediate results and data words to adjacent cells synchronously.

In this paper we propose a scalable, i.e., the size and near neighbor interconnections of the array may be either extended or reduced, systolic array which implements any instance of the dynamic programming problem by deriving a systolic array for a particular dynamic programming problem, and then showing that the systolic array derived for the particular problem can be scaled to a systolic array for another problem.

*Chungbuk National University, Research Institute for Computer and Information Communication
**Charles Sturt University Bathurst, NSW, 2975, Australia
Email : dtien@csu.edu.au

접수 일자 : 2003. 5. 15 수정 완료 : 2003. 7. 25
논문 번호 : 2003-2-13
※This research was supported in part by Information & Communication fundamental Technology Research Program supported by Ministry of Information & Communication in republic of Korea

II. Dynamic Programming Formulation

The dynamic programming problems can be stated in the following recurrence relation, where C(i, j) is an objective function which should be either minimized or

maximized subject to a set of constraints [4].

$$C(i,j)=\begin{cases} s>0 \rightarrow \min_{i<k<j}\{F(C(i,k), C(k,j))\} \\ \quad \text{where F is some function on the costs} \\ s=0 \rightarrow C_i \text{ for some individual cost} \end{cases}$$

where $s=j-1$ and i and j are integer in the range of $0<i<j\leq n$ for some constant n.

A recurrence relation for a particular problem is established by specifying the objective function and associated operations, and the recurrence relation once established can be transformed to another one by changing the objective function and associated operations. The recurrence relations and transformations among them can be modeled using complete graph where each vertex represents a recurrence relation describing a particular dynamic programming problem and each edge a transformation between recurrence relations. The transformation in the recurrence relation corresponds to the scalability, also called flexibility, in the systolic array such as programmability in processing elements and reconfigureability in the selection of local interconnection in view of implementation. If we choose a dynamic programming problem and derive a systolic array which implements the recurrence relation describing the problem, then it implementing another dynamic programming problem can be developed by applying the programmability and reconfigurability to the original systolic array.

We choose the single source shortest path problem, SSSP, as a particular dynamic programming problem and derive a systolic array for SSSP, SA SSSP, and then convert the SA SSSP to a systolic array for the 0 1 knapsack problem, SA 01KS to show that any instance of the dynamic programming problem can be solved with one systolic array in case of the scalability is allowed.

III. Systolic Array for Single Source Shortest Path Problem

For a given graph G=(V, E), single source shortest path problem is to find a shortest path from a given source vertex $u\in V$ to every vertex $v\in V$ [1].

Let the number of vertices of G be denoted by n, and suppose we have a set of i vertices (where $1\leq i\leq n$), W_i, which satisfies the proprieties [5]:

1) $u\in W_i$

2) for any $w\in W_i$, the shortest u w path contains only Intermediate vertices in W_i.

For any vertex v of G, let d(i, v) denote the length of the shortest u v path containing only intermediate vertices from W_i, hence, if $v\in W_i$, then d (i, v) is the length of the shortest u v path. Let v(i) be the vertex not in W_i which has the smallest distance d(i, v) value. Then d(i, v(i)) is in fact the length of the shortest u v(i) path, since using other vertices not in W_i would lead to a longer path.

Hence, if we define $W_{i+1}=W_i\cup\{v(i)\}$, then W_{i+1} also satisfies proprieties 1) and 2) above.

We obtain the following recurrence relation:

$d(i+1,v)=d(i,v)$ if $v\in W_{i+1}$, $i\geq 1$.

$d(i+1,v)=\min\{d(i,v), d(i,v(i))+w(v(i),v)\}$ if $v\notin W_{i+1}$, $i\geq 1$.

The initial values are given by:

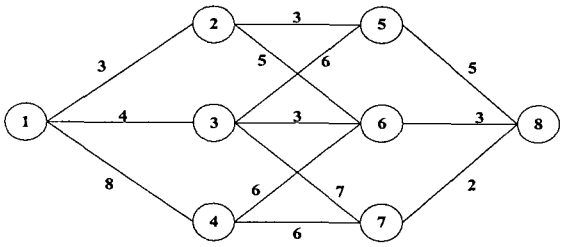
$d(i,v)=0$ if $v=u$

$d(i,v)=w(u,v)$ if $v\neq u$

For a sample graph, G, composed of eight nodes, Fig. 1(a), the elements of its distance matrix A are defined as follows:

1. aij = the distance from vertex i to vertex j.
2. aij = ∞ if there is no edge connecting i and j.
3. aij =0 if i = j.

The single source shortest path problem finding a shortest path from a given source vertex $u\in V$ to every vertex $v\in V$ can be solved by recurrence relation defined above. The values of $d(i,v)$ which is the solution of given sample problem are displayed in Table 1.



(a)

$$A=\begin{bmatrix} 0 & 3 & 4 & 8 & \infty & \infty & \infty & \infty \\ 3 & 0 & \infty & \infty & 3 & 5 & \infty & \infty \\ 4 & \infty & 0 & \infty & 6 & 3 & 7 & \infty \\ 8 & \infty & \infty & 0 & \infty & 6 & 6 & \infty \\ \infty & 3 & 6 & \infty & 0 & \infty & \infty & 5 \\ \infty & 5 & 3 & 6 & \infty & 0 & \infty & 3 \\ \infty & \infty & 7 & 6 & \infty & \infty & 0 & 2 \\ \infty & \infty & \infty & \infty & 5 & 3 & 2 & 0 \end{bmatrix}$$

(b)

Fig. 1. (a) A sample graph (b) Distance matrix A

Table 1. Calculation of d(i, v)

i/*v*	1	2	3	4	5	6	7	8	*v*(*i*)
1	0	3	4	8	∞	∞	∞	∞	2
2	0	3	4	8	6	8	∞	∞	3
3	0	3	4	8	6	7	11	∞	5
4	0	3	4	8	6	7	11	11	6
5	0	3	4	8	6	7	11	10	4
6	0	3	4	8	6	7	11	10	8
7	0	3	4	8	6	7	11	10	7
8	0	3	4	8	6	7	11	10	

In the following, we derive the dependence graph, DG, for a systolic array for the single source shortest path problem. The recurrence relation in single source shortest path problem can be represented as single assignment form by introducing an iteration dimension k:

For k from 1 to N 1, where N is the number of vertices
For i, j from 1 to N

$d(i+1,j,k)=d(i,j,k)$ if $j\in W_{i+1}$, $i\geq 1$

$d(i+1,j,k)=\min\{d(i,j,k), d(i,v(i),k)+w(v(i),j,k)\}$

if $j\notin W_{i+1}$, $i\geq 1$, where j denotes vertex in graph G

The cubic DG for the case N=8 is shown in Fig.2. In DG, we find that there exists a directed path of length 8N along the bold faced arcs. Dependence arcs on each k plane other than top plane are not shown for simplicity.

The DG obtained can now be safely projected in the k, [0 0 1], direction. The signal flow graph, SFG, generated by the k projection is shown in Fig.3 for the case N=8. The array uses N2 PEs, and each PE performs the same operation. All dependence arcs except input arcs have unit delay each.

In general, it is possible to map an M dimensional SFG directly onto an (M r) dimensional SFG (r = 1, 2,…, M 1) by a scheme called multiprojection [6]. The spatial complexity of the initial SFG can be reduced by multiprojection. The systolic array obtained through the multiprojection in the i and k directions is shown in Fig.4. The array uses N PEs and the data pipeline period of the array is N unit delays.

The Fig.5 shows block diagram for SSSP that consists of two memories, SA SSSP and module for calculating v(i). Two memories is used to store the distance matrix and solution, i.e., d(i,v) table, respectively. Module for calculating v(i) selects the vertex having the smallest value and not used before. If v(i) is determined, the corresponding values of distance matrix is entered to SA SSSP proposed in this paper. SA SSSP performs SSSP algorithm and its results are stored d(i,v) table in memory.

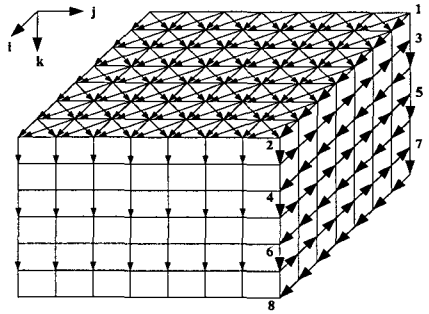


Fig. 2. DG for the case N=8

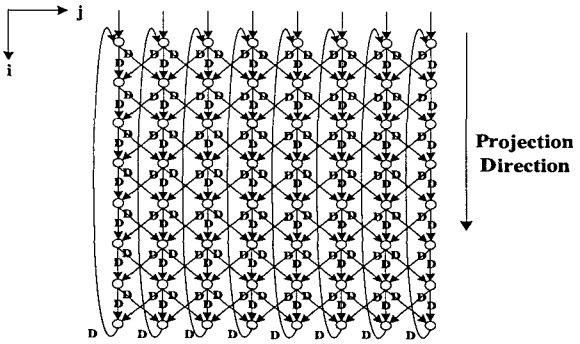


Fig. 3. SFG in ij plane from projection along k direction

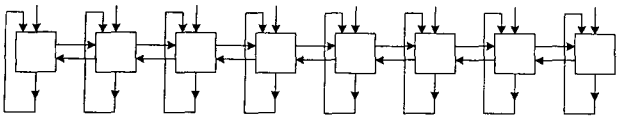


Fig. 4. Systolic array from multiprojection along i, k directions

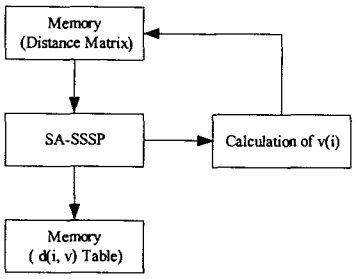


Fig. 5. Block diagram for SSSP

IV. Systolic Array for 0 1 Knapsack Problem

Suppose we are given profits p_i and weight w_i, for $1 \le i \le n$, and a knapsack capacity M. The problem is to maximize $\Sigma x_i w_i$, subject to $\Sigma x_i w_i \le M$ and $x_i = 0$ or 1, $1 \le i \le n$ [5].

If $x_i = 0$, then the best profit we can attain is whatever we can get from objects 1, 2,···, n 1. If $x_i = 1$, we have a profit of p_i, and the remaining knapsack capacity is $M - w_i$, with which we attain the optimal profit using object 1, 2,···, n 1. The optimal solution is the better of these two feasible solutions.

We can state this with the recurrence relation below. For $0 \le m \le M$, and $1 \le j \le n$, define p(j,m) to be the profit of the optimal solution to the 0 1 knapsack problem, using objects 1,2,···, j, and knapsack capacity m. Then, we have the recurrence relation for $2 \le j \le n$ and $0 \le m \le M$:

$$p(j,m) = p(j-1,m) \quad \text{if} \quad w_j > m$$

$$p(j,m) = \max\{p(j-1,m),\ p_j + p(j-1,m-w_j)\} \quad \text{if} \quad w_j \le m$$

The initial values are given by:

$$p(1,m) = 0 \quad \text{if} \quad w_1 > m$$

$$p(1,m) = p_1 \quad \text{if} \quad w_1 \le m$$

The optimal solution is given by $p(n,M)$. In the following, we derive the dependence graph, DG, for a systolic array for the 0 1 knapsack problem. The DG reflecting a specific recurrence relation is obtained from figuring out a pictorial form of node arrangement and data flow through arcs represented in the recurrence relation, and the DG for a recurrence relation can be developed on the basis of DG for another recurrence relation with different details such as objective function and local inter connections by either adding or deleting arcs and scaling the number of nodes according to the recurrence relation details. The recurrence relation in the 0 1 knapsack problem can be represented as single assignment form by introducing an iteration dimension k:

For k from 1 to N 1, where N is the number of objects
For i from 1 to M, where M is a knapsack capacity
For j from i to M

$$p(i,j,k) = p(i-1,j,k) \quad \text{if} \quad w_i > j$$

$$p(i,j,k) = \max\{p(i-1,j,k),\ p_i + p(i-1,j-w_i,k)\} \quad \text{if} \quad w_i \le j$$

The cubic DG obtained with DG of SSSP for a basis for the case N=8 and M=8 is shown in Fig.6. Dependence arcs on each k plane other than top plane are not shown for simplicity. The SFG generated by the k, [0 0 1], projection is shown in Fig.7 for the case N=8 and M=8. The array uses M(M+1)/2 PEs, and each PE performs the same operation. All dependence arcs except input arcs have unit delay each. The systolic array obtained through the multiprojection in the i and k directions is shown in Fig.8. The array uses M PEs.

The Fig.9 shows block diagram for 01KS that consists of memory and SA 01KS. The memory is used to store profit values generated by SA 01KS proposed in this paper.

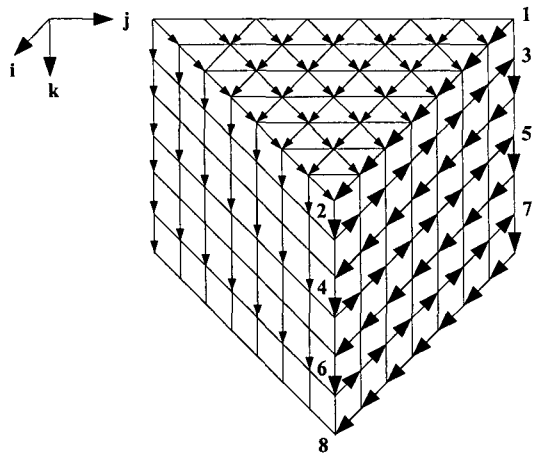


Fig. 6. DG for the case N=8 and M=8

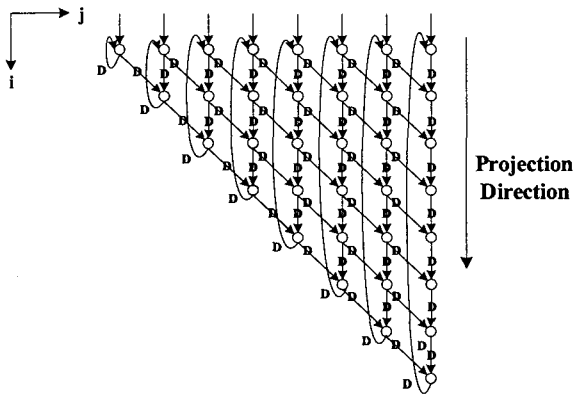


Fig. 7. SFG in ij plane from projection along k direction

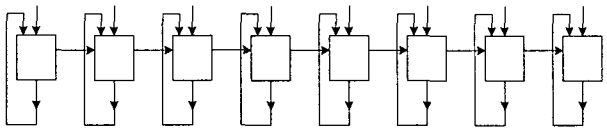


Fig. 8. Systolic array from multiprojection along i, k directions

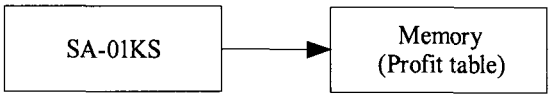


Fig. 9. Block diagram for 01KS

V. Simulation, Synthesis and Implementation

Each of the systolic arrays derived for the single source shortest path problem and 0 1 knapsack problem was modeled and simulated in RT level using VHDL [7], and synthesized to a schematic using Synopsys design compiler [8][9]. The simulation result, i.e., the solution of given instance for SSSP, is shown in Fig. 10 in which the rows and columns in Table 1 are interchanged. The infinity, ∞, is assigned a value of 20 and signals, COLUMN1~8, traced in simulation make up the distance table showing the value of each d(i, v).

For an instance of 0 1 knapsack with weights 2, 4, 6, 9, 14, and 17, profits 1, 3, 4, 6, 8, 11 respectively, and capacity 30, the simulation result is shown in Fig. 11for COLUMN0~12 and in Fig. 12 for COLUMN13~30. Signals, COLUMN0~30, traced in simulation make up the profit table showing the value of each p(j, m).

A synthesized schematic using Synopsys design compiler for SA SSSP is shown in Fig. 13(a) and for SA 01KS in Fig. 13(b). Table 2 show synthesis results for SA SSSP and SA 01KS.

After being synthesized by Synopsys design compiler, each of SA SSSP and SA 01KS was automatically implemented to a layout using Apollo tool provided by AVANT!. Used cell library is based on 0.35㎛ CMOS 1 poly 4 metal CMOS technology. Each of the layout and information of core area for SA SSSP and SA 01KS is shown in Fig. 14, Table3, Fig. 15, and Table 4, respectively.

This paper focuses on design methodology, which derives 1 dimensional systolic array for a dynamic programming, not implementation, so implementation details such as clock, performance, constraint and etc. are not carefully considered.

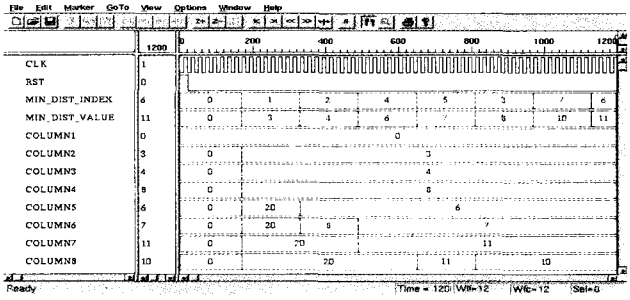


Fig. 10. Simulation showing COLUMN1~8 of $d(i,v)$ table

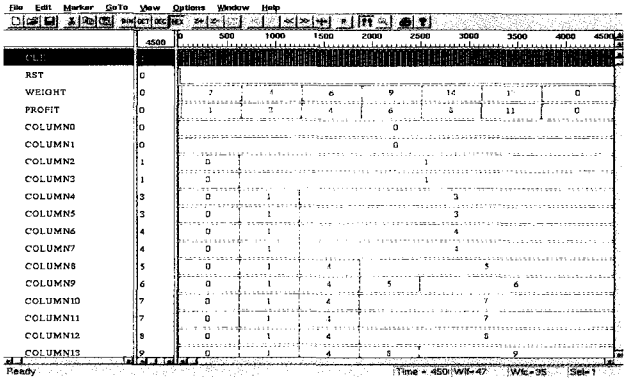


Fig. 11. Simulation showing COLUMN0~12 of $p(j,m)$ table

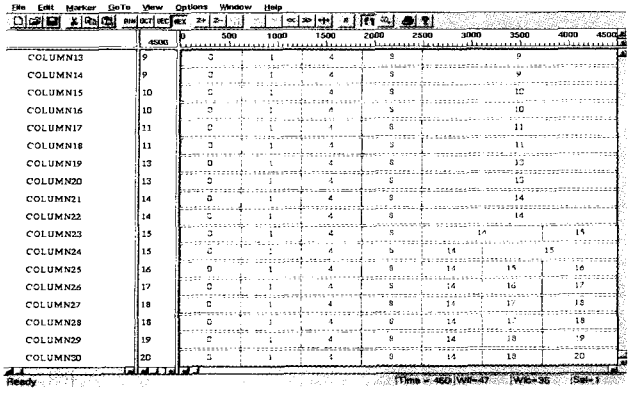
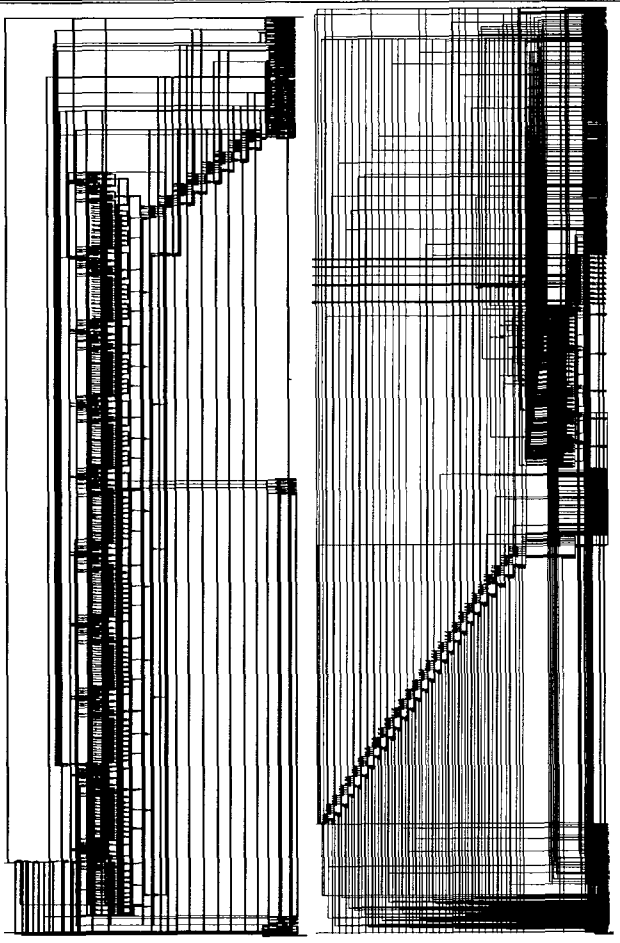


Fig. 12. Simulation showing COLUMN13~30 of $p(j,m)$ table

Table 2. Synthesis results

Design	Total area (2 input NAND)	Number of netlists
SA SSSP	761	1362
SA 01KS	1002	2518



(a) (b)

Fig. 13. (a) Synthesized schematic for the SA SSSP
(b) Synthesized schematic for the SA 01KS

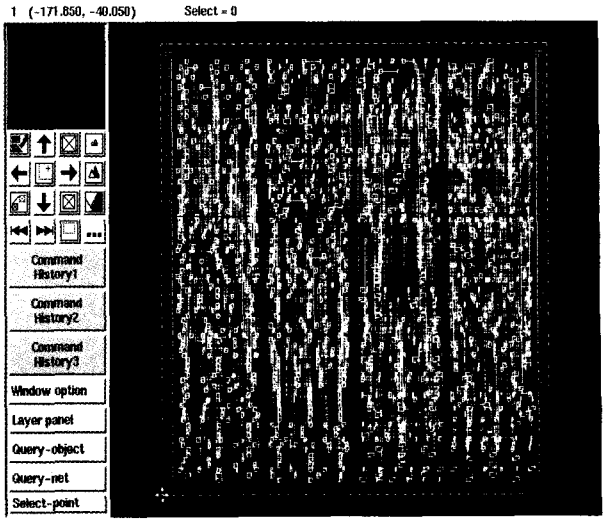


Fig. 14. Layout of SA SSSP

Table 3. Area information of SA SSSP

Core width (㎛)	Core height (㎛)
1200	1200

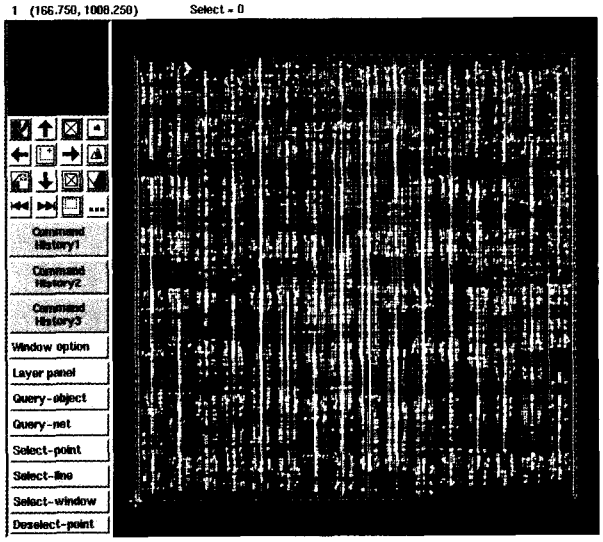


Fig. 15. Layout of SA 01KS

Table 4. Area information of SA 01KS

Core width (㎛)	Core height (㎛)
2000	2000

VI. Conclusions

We derived a systolic array for a particular instance of dynamic programming problem, single source shortest path problem, and then developed a systolic array for 0

1 knapsack problem with SA SSSP for a basis to show that the systolic array for a dynamic programming problem has the generality which makes the systolic array for one dynamic programming problem applicable to another dynamic programming problem.

Initial SFGs generated by projection in the k direction required O(n2) PEs. To reduce the spatial complexity of the systolic array, multiporjection method was used. By this scheme, we could construct a efficient systolic array having n PEs. Each of the systolic array derived through the multiprojection was modeled and simulated in RT level using VHDL, then synthesized and implemented to a layout.

References

Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Intorduction to Algorithms, McGraw Hill, 1991.

H.T.Kung, "Why Systolic Architectures?," Computer Vol.15, No.1, pp.37 46, January 1982.

V.Kumar, A.Grama, A.Gupta, and G.Karypis, Int. to Parallel Computing Design and Analysis of Algorithms, The Benjamin/Cummings Publishing Company, Inc., 1994.

M. C. Chen, Synthesizing VLSI architectures : dynamic programming solver, In international Conference on Parallel Processing, pp. 776 784, Chicago, IL, August 1986.

D.R.Stinson, An Intorduction to the Design and Analysis of Algorithms, The Charles Babbage Research Centre, 1987.

S.Y.Kung, VLSI Array Processors, Prentice Hall, 1988.

Y.C.Hsu, K.F.Tsai, J.T.Liu and E.S.Lin, VHDL Modeling for Digital Design Synthesis, Kluwer Academic Publishers, 1995.

K.C. Chang, Digital Systems Design with VHDL and Synthesis, IEEE Computer Society Press, 1999.

Weng Fook Lee, VHDL Coding and Logic Synthesis with Synopsys, Academic Press, 2000.

Jae Jin Lee
Member

He received the B.S. and M.S. degrees in computer engineering from Chungbuk National University in 2000 and 2003, respectively.

He is currently working towards Ph.D. degree on computer engineering.

His research interests include the areas of VLSI design, design automation and computer architecture.

David Tien

He received his undergraduate, master's and Ph.D. degrees in Computer Science, Pure Mathematics and Electrical Engineering from Harbin, Chinese Science Academy, the Ohio State University, USA and the University of Sydney, Australia, respectively.

Prior to joining the School of Information Technology, Charles Sturt University in 2000, he had 20 years' experience in research and teaching at the University of Sydney, Ohio State University and Singapore.

Currently, his major research interests are artificial intelligence, image and signal processing, telecommunication, education theory, and biomedical engineering. During the past 15 years, David served as the Secretary, Treasurer and Chairman of the IEEE Singapore Section, MDC Chairman of Tele Communication Society, Region 10, and is currently the Treasurer of NSW Section. David also serves as a member of the Charles Sturt University Senate.

Gi Yong Song
Member

He received the B.S. and M.S. degrees in electronic engineering from Seoul National University in 1978 and 1980, respectively and the Ph.D. degree from University of Louisiana in 1995.

He is currently a professor in the School of Electrical and Computer Engineering.

His research interests include the areas of VLSI design, design automation, and computer architecture.