<table>
<thead>
<tr>
<th>Title</th>
<th>I/O Time Complexity and Synchronization in Iterative (or Systolic) Arrays</th>
</tr>
</thead>
<tbody>
<tr>
<td>Author(s)</td>
<td>Umeo, Hiroshi</td>
</tr>
<tr>
<td>Citation</td>
<td>数理解析研究所講究録 1984-05, 522: 212-224</td>
</tr>
<tr>
<td>Issue Date</td>
<td>1984-05</td>
</tr>
<tr>
<td>URL</td>
<td><a href="http://hdl.handle.net/2433/98465">http://hdl.handle.net/2433/98465</a></td>
</tr>
<tr>
<td>Type</td>
<td>Departmental Bulletin Paper</td>
</tr>
<tr>
<td>Textversion</td>
<td>publisher</td>
</tr>
</tbody>
</table>

Kyoto University
I/O Time Complexity and Synchronization
in Iterative(or Systolic) Arrays

Hiroshi Umeo
梅尾 博司

Dept. of Applied Electronic Engineering
Faculty of Engineering
Osaka Electro-Communication Univ.
18-8, Hatsu-cho, Neyagawa-shi, 572, Japan

Abstract In this paper we present the following two systolic simulation theorems.

(1) Let \( M_i \) \( (1 \leq i \leq k) \) be any simple SIMD machine with the same instruction set, each with time complexity \( T_i(n) \), where \( k \) is an even integer. Then there exists a systolic array \( A \) which simulates all \( M_i \)'s \( (1 \leq i \leq k) \) in \( 2 \sum_{i=1}^{k/2} T_{2i}(n) + kn + 2n + O(1) \) steps.

(2) For any one-way \( kn \) time-bounded cellular automaton \( M \) there exists a systolic array \( A \) which can simulate \( M \) in \( kn + n \) steps.

1. Introduction

There has been increasing interest in the study of systolic systems which overlap I/O operations and computations. In the design of systolic algorithms, speeding up the I/O operations, without sacrificing their total throughputs, is an important problem.
In [1] and [2] we have developed time-efficient conversion techniques from cellular algorithms and from SIMD algorithms, both of which separate computations from I/O operations, into systolic ones, respectively.

In this paper we will develop several more high-speed simulation techniques for the 1-dimensional (1-D) multiple SIMD machines with the same instruction set and for the 1-D cellular automata with restricted information-flows. It is shown that a more remarkable speed up is attained in their implementations than the former results in [1] and [2].

2. Systolic implementation theorems

A concept of systolic architecture was originally proposed for VLSI implementations of some matrix operations, after that, many works have been done for systolic arrays[6], [7], [11].

A linear systolic array, considered in this paper, consists of a single buffer(B) and a number of linearly-connected processors, called systolic cells(C_i), shown in Fig.1. Each processor C_i (i \geq 0) can communicate with its nearest neighbour processors C_{i-1} (or B) and C_{i+1}. We measure the time complexity of systolic arrays by the sum of steps required for loading inputs, processing, and retrieving outputs. Consult [7] for further details.

![Fig.1 A systolic array.](image)

2.1 Systolic implementation of simple SIMD algorithms

A simple SIMD (single instruction stream/multiple data stream) machine M is a restricted SIMD parallel computer model which consists of a control unit(CU), a linear array of n processing elements(PE's), each with its own finite number of registers, and a nearest-neighbour interconnection network, shown in Fig.2. The initial data a_0, a_1, ..., a_{n-1} are preloaded to each PE,
that is, \( a_i \) at PE\(_i\). We measure the parallel time complexity \( T(n) \) by the number of instructions broadcasted by the CU. The computational results are left on each PE in a distributive manner.

![Diagram of a simple SIMD machine](image)

Fig. 2 An illustration of a simple SIMD machine.

The simple SIMD machine is in a subclass of the conventional SIMD machines in the meaning that:

1. Each PE has no main memory.
2. The nearest-neighbour interconnection network is assumed.
3. The set of instructions broadcasted by the CU is restricted, that is, to the set of one-step instructions, described in [1].

Our SIMD model is restricted, however, there are many simple-type SIMD algorithms in the conventional SIMD parallel algorithms, such as sorting algorithms [5], [9], [10], image processing algorithms [8], [12], [13], graph algorithms [11], and so on. Thus our class of simple SIMD machines are thought as a wide class of SIMD machines. For further details the reader should refer to [1] and [3].

In [1] we have shown the following theorem.

[Theorem 1][1] (Single-Task Systolic Simulation Theorem)
For any simple SIMD machine \( M \) with time complexity \( T(n) \), there
exists a systolic array $A$ which simulates $M$ in $2T(n) + 3n + O(1)$ steps.

(Proof sketch) Without loss of generality we assume that $M$ has $n$
processing elements $PE_i (0 \leq i \leq n-1)$, each with a single data register and an
address register, where $n = 2^m$ for some integer $m$. A similar method
presented below can be applied to the simulation of $M$ with $k (\geq 2)$ data
registers. Let $a_i (0 \leq i \leq n - 1)$ be the data preloaded to $PE_i$ and $I_t (1 \leq t \leq T(n))$ be the instruction broadcasted to each PE by the CU.

The organization of the systolic array $A$ which simulates $M$ is as
follows:

$A$ consists of a buffer $B$ and $(n + 1)$ systolic cells $C_i (0 \leq i \leq n)$. The
buffer $B$ contains an input and output buffer registers, denoted by $B_{in}$ and
$B_{out}$, respectively. The contents of $B_{in}$ and $B_{out}$ are updated by the host
computer and by $C_0$, respectively. Each systolic cell contains a $[\log_2 n]$-bit
address register $R_a$ and five auxiliary registers, $R_0, R_1, R_2, R_3$, and $R_s$. Let
$L_j$ denote all $R_j$'s on the array, referred to as the $j$-th layer ($0 \leq j \leq 3$).

In the simulation of $M$, for each $i, 0 \leq i \leq n - 1$, $C_i$ simulates $PE_i$
individually. $C_n$ acts as a boundary cell. Auxiliary registers in the
systolic cell are used for the following purposes:

$R_0$ : used as an auxiliary register for the address setting.
$L_1$ : used as a pipeline which transmits initial data and instructions
in the right direction.
$L_2, L_3$ : used as book-keeping registers which store the contents of the
data register in PE. $L_3$ is also used as a pipeline which transmits
outputs in the left direction.

The host computer prepares the date for $A$ in the following initial
data/instruction stream format such that:

$$a_0 \ a_1 \ \ldots \ a_{n-1} \ I_1 \ * \ I_2 \ * \ \ldots \ I_{T(n)-1} \ * \ I_{T(n)} \ * \ \Box \ \ldots \ (1)$$

The data stream (1) is supplied to $A$ through $B_{in}$ according to the order $a_0,$
$a_1, \ldots, I_1 \ldots$ at the rate of 1 symbol/1 step. We assume that $B_{in}^0 = a_0$.
The symbol "*" and "\Box" represents a blank and a terminal symbol,
respectively. Thus instructions are supplied at the rate of 1 instruction/2 steps.

![Figure 3](image)

**Fig. 3** Time-space diagram for systolic simulation of simple SIMD machine.

We assume that each cell can distinguish $a_i$, $I_t$, "*", and "■" from each other and can interpret and execute all instructions broadcasted by M. The simulation in each cell consists of three phases, that is, an initial data loading(I), an instruction-execution(II), and an output phases(III). The register $R_s$ is used to indicate the current phase-state that the cell is assuming. On the whole array the simulation proceeds pipelinedly by overlapping these phases. See Fig. 3.
Fig. 4 Configurations of the systolic array which simulates a simple SIMD machine (n = 5 and T(n) = n).
An output is obtained in $B_{\text{out}}$ in every two steps and it is taken into the host computer at once.

The array requires $2T(n) + n$ steps for loading the data/instruction stream and $2n + 1$ steps for outputting. Thus $2T(n) + 3n + O(1)$ steps are required for the simulation.

In Fig. 4 we illustrate the configurations (on $R_1$, $R_2$ and $R_3$ only) of the systolic array $A$ which simulates $M$. In that figure $s^t_i$ denotes the content of the data register of $P_{E_1}$ of $M$ at time $t$.

Next we consider the multiple-use of the single systolic array for the simulation of many simple SIMD machines with the same instruction set. By a slight modification of the systolic cells we can make the symbol "#" shown in Fig. 4 (at time $t = 21, \ldots, 25$) reset the entire systolic cells successively. This modification enables us to use the systolic array repeatedly.

[Theorem 2] (Multi-Task Systolic Simulation Theorem (ver. 1))

Let $M_i$ ($1 \leq i \leq k$) be any simple SIMD machine, each with time complexity $T_i(n)$, which has the same instruction set. Then there exists a systolic array $A$ which simulates $M_i$'s ($1 \leq i \leq k$) in $2 \sum_{i=1}^{k} T_i(n) + 3kn + O(1)$ steps.

(Proof sketch) The host computer supplies the systolic array $A$ with $k$ initial data/instruction streams, each separated by $2n$ blank symbols, prepared in the following form:

\[
\begin{align*}
\text{Data/Instruction Stream} & \quad \text{For } M_1 \\
\text{For } M_2 & \\
\text{For } M_k \\
\end{align*}
\]

(2)

In the proof of Theorem 1, $R_3$ is used both for the temporary data register and for the output pipeline. If we furnish an another register for the output pipeline and give the symbol "#" a reset function, then the host
computer can overlap the I/O operations. This speedups the simulation in Theorem 2 by $2(k-1)n$ steps.

[Theorem 3] (Multi-Task Systolic Simulation Theorem (ver.2))

Let $M_i$ ($1 \leq i \leq k$) be any simple SIMD machine, each with time complexity $T_i(n)$, which has the same instruction set. Then there exists a systolic array $A$ which simulates $M_i$'s ($1 \leq i \leq k$) in $2 \sum_{i=1}^{k} I_i + kn + 2n + O(1)$ steps.

In Theorem 3 we obtained a high-speed version by overlapping the I/O operations. In the next theorem we develop another high-speed version by interleaving instructions.

[Theorem 4] (Two-Task Systolic Simulation Theorem (ver.3))

Let $M_1$ and $M_2$ be any simple SIMD machine, each with the same instruction set and with time complexity $T_1(n)$ and $T_2(n)$, respectively. Then there exists a systolic array $A$ which simulates both $M_1$ and $M_2$ in $\max(2T_1(n), 2T_2(n)) + 4n + O(1)$ steps.

Proof sketch) Suppose that $T_1(n) \leq T_2(n)$. Let $a_i(b_i)$ be an initial data preloaded in $P_{E_i}$ of $M_1(M_2)$, and $I^a_j(i^b_k)$ be an instruction broadcasted by $M_1$ at time $t = j$ (by $M_2$ at time $t = k$), where $0 \leq i \leq n-1$, $1 \leq j \leq T_1(n)$ and $1 \leq k \leq T_2(n)$. The host computer supplies $A$ with an initial data/instruction stream given below:

$$a_0a_1\ldots a_{n-1}b_1b_2\ldots b_{n-1} I^a_1 I^b_1 I^a_2 I^b_2 \ldots I^a_{T_1(n)} I^b_{T_1(n)} I^a_{T_2(n)} I^b_{T_2(n)} \ldots$$

$$\ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots$$

(3)

A modification for this simulation is to add only one more register to each systolic cell given in proof of Theorem 1. The register $R_2$ is shared as the temporary data register for $M_1$ and $M_2$. Note that it is not necessary to broaden the I/O bus which connects the systolic array to the host computer.

By combining Theorems 3 and 4 we get the following theorem. We omit the proof.
[Theorem 5] (Multi-Task Systolic Simulation Theorem (ver.4))

Let \( M_i (1 \leq i \leq k) \) be any simple SIMD machine with the same instruction set, each with time complexity \( T_i(n) \) such that \( T_1(n) \leq T_2(n) \leq \ldots \leq T_k(n) \), where \( k \) is an even integer. Then there exists a systolic array \( A \) which simulates \( M_i \)'s (1 \( \leq i \leq k \)) in \( 2 \sum_{i=1}^{k/2} T_{2i}(n) + kn + 2n + O(1) \) steps.

In Table 1 we summarize our systolic simulation techniques developed in this paper.

<table>
<thead>
<tr>
<th>Theorem</th>
<th>Number of Registers</th>
<th>Time Complexity of ( T_i(n) )</th>
<th>Time Complexity of Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>( T(n) )</td>
<td>( 2T(n) + 3n )</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>( \sum_{i=1}^{k} T_i(n) )</td>
<td>( 2 \sum_{i=1}^{k} T_i(n) + 3kn )</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>( \max(2T_1(n), 2T_2(n)) )</td>
<td>( \max(2T_1(n), 2T_2(n)) + 4n )</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>( T_1(n) ) and ( T_2(n) )</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>( \sum_{i=1}^{k/2} T_i(n) )</td>
<td>( 2 \sum_{i=1}^{k/2} T_i(n) + kn + 2n )</td>
</tr>
</tbody>
</table>

Table 1. The number of auxiliary data registers and parallel steps required for our simulations.

2.2 Systolic implementation of one-way cellular algorithms

A 1-dimensional one-way cellular automaton (CA) \( M \) consists of an array of finite state automata, called cells \( C_i (1 \leq i \leq n) \), which are uniformly interconnected. See Fig.5. \( M \) is a pair \( M = (Q, \delta) \), where \( Q \) is the set of
cell states and $\delta: Q^2 \rightarrow Q$ is the one-way local transition function. We denote the state of $C_i$ at time $t$ by $s_t^i$. At time $t=0$, CA receives an spatial input in the way such that $s_0^i = a_i (1 \leq i \leq n)$. A step of computation of M consists of a state transformation of each cell, that is, the simultaneous applications of $\delta$ at all cells in such a way that $s_{t+1}^i = \delta(s_t^i, s_{i+1}^t)$. The configuration $s_1^t s_2^t \ldots s_n^t$ is considered as the output of $T(n)$ time-bounded CA, M, for an input $a_1, a_2, \ldots, a_n$. Following the convention in the cellular automata theory, we measure the time complexity for the CA's by the parallel steps required only for the computations.

![Cellular automaton diagram]

Fig.5 Cellular automaton.

In [2] we get the following result for the two-way CA's.

[Theorem 6][2] For any two-way $kn$ time-bounded CA, M, there exists a systolic array A which can simulate M in $kn + 3n$ steps.

If the direction of the information-flow of the CA's is restricted to one-way, the following fast systolic simulation is possible.

[Theorem 7] For any one-way $kn$ time-bounded CA, M, there exists an SA, A, which can simulate M in $kn + n$ steps.

(Proof sketch) We construct an SA, A which simulates M in $kn + n$ steps. First we prove the case $k \geq 2$. A consists of $n$ systolic cells, each contains four data registers $R_1, R_2, R_3$, and $R_4$. We refer to $R_1$ and $R_2$ in each cell as the first layer and $R_3$ and $R_4$ as the second layer. Initial data is loaded through the buffer B according to the order $a_n, a_{n-1}, \ldots, a_2, a_1$ at the rate of 1 data/ 1 step. The data movement on the array is as follows: Each data continues to advance in the right direction at a unit speed on the first layer, searching for empty $R_3$ and $R_4$ registers. When they are found, the data stays at $R_3$ and $R_4$ until output signal is transmitted to that cell.

Due to the one-way information-flow of $M$, at every step each data on both layers can substantially simulate a 1-step state transition of the
corresponding cell, since the information necessary for the simulation is found in either the right or the left neighbour cell. From time $t = 1$ A begins to prepare the firing squad synchronization which will fire at time $t = kn$. The firing tells each cell to begin output operations. Each cell begins to shift its data on the second layer in the left direction at a unit speed. Note that at time $t = kn$ the second layer contains the configuration $s_n s_{n-1} s_{n-2} \ldots s_1 s_1$. Additional $n$ steps are required for the output operations. In Fig.6 we illustrate the configurations of A in the case $k = 2$ and $n = 5$.

In the case $k = 1$, $n/2$ cells are sufficient. At time $t = n$ the firing occurs and $n$ data are output after $n$ steps. Thus A requires $kn + n$ steps for the simulation of M.

### 3. Concluding Remarks

In this paper we have developed algorithmic conversion techniques which simulate a certain class of SIMD machines and one-way cellular automata on systolic arrays. Several time-efficient systolic simulation theorems are established (Theorems 2, 3, 4, 5, and 7). The systolic simulation theorem given in this paper presents a uniform method which...
implements any simple SIMD algorithm and cellular algorithms onto VLSI systolic arrays. The method enables us to use SIMD algorithms, which have been developed and accumulated for conventional SIMD machines such as ILLIAC-IV, on the VLSI systolic arrays almost without loss of time efficiency. Similar discussions for the two-dimensional arrays can be easily obtained.

Acknowledgements
This work has been supported by Grant of the Ministry of Education, Science and Culture, No.58780050.

References