

A Design of Time-Optimum and Register-Number-Minimum
Systolic Convolver

Hiroshi Umeo⁽⁺⁾

Osaka Electro-Communication University
Neyagawashi, Hatsucho, 18-8, Osaka, 572, Japan

Abstract

We present an optimum bit-parallel/word-sequential systolic convolver. Our design is the best one among the previous many convolvers in the sense that its optimality in time and space performances is simultaneously attained without augmenting any global control, broadcasting, preloading, and/or multi sequential or parallel I/O ports, which were allowed in most of the previous designs. As an application of our convolver we give a systolic polynomial divider which can compute the polynomial division in exactly $n + O(1)$ steps on $[\min(n-m, m)/2] + O(1)$ systolic cells, for the division of any degree n polynomial by any degree m polynomial ($n \geq m$).

1. Introduction

Convolution is probably one of the most important problems in the fields of signal and image processing. A large number of convolution algorithms have been proposed, since this type of computation is commonly used for many problems, such as pattern matching, digital filtering, discrete Fourier transforms, polynomial multiplication and division, and so on. Recently much attention has been paid to the study of systolic convolution algorithms[4-5], [7], [9-12], [14], [18-19], [21-24], [27-29], [33-36] and several practical constructions and implementations on VLSI are made[12], [18], [27], [35].

In this paper we will present a time-optimum and register-number-minimum systolic convolver. Our algorithm is based on Atrubin's binary parallel multiplier[2], Knuth's its revised version[20] and Cole's real-time iterative palindrome recognizer[8], which have been known as tricky cellular algorithms.

The systolic array that we assume is the most basic model which consists of a locally-connected semi-infinite array of identical systolic cells with a single I/O port being positioned at one end of the array. The data broadcasting, preloading, and/or multi sequential or parallel I/O ports are not allowed in our model.

(+) The author stays as an Alexander-von-Humboldt researcher at Institut für Theoret. Informatik, Techn. Universität Braunschweig, Gausstrasse 11, D-3300, Braunschweig, West Germany.

It is shown that there exists a systolic array M which can compute the convolution c_i ($i = 0, 1, 2, \dots, n + m - 1$) of any two finite sequences a_j ($j = 0, 1, 2, \dots, n - 1$) and b_l ($l = 0, 1, 2, \dots, m - 1$) in optimum real-time, exactly in $n + m + O(1)$ steps, using $[\min(m,n)/2] + O(1)$ systolic cells. The number of data registers in each systolic cell is minimum. A new data routing scheme developed for our convolver helps us to understand the correctness of the tricky algorithms above. As an application of our convolver, we give a systolic real-time pattern matcher and show that not only data preloading but also broadcasting are not necessarily essential operations in the design of systolic optimum-time pattern matcher. In addition we give a fastest systolic polynomial divider which can compute the polynomial division in exactly $n + O(1)$ steps on $[\min(n-m, m)/2] + O(1)$ systolic cells, for the division of any degree n polynomial by any degree m polynomial ($n \geq m$). The divider is time-optimum and register-number-minimum. Our design is superior to the previous ones [21], [33], [36] in both the time and space(register-number) complexities.

The organization of this paper is as follows: In section 2 an optimum real-time and register-number-minimum systolic convolver is presented. Its validity is also given. In section 3 several variations and applications of our convolver are presented. Lastly we give a conclusion in section 4.

2. Time-optimum and register-number-minimum
systolic convolver

2.1 Basic definitions

We first consider the convolution of two infinite sequences defined as follows:

Convolution: Given two infinite sequences A and B such that

$$A = \langle a_0, a_1, \dots, a_i, \dots \rangle \dots (1)$$

$$B = \langle b_0, b_1, \dots, b_j, \dots \rangle \dots (2)$$

Compute an infinite sequence C ,

$$C = \langle c_0, c_1, \dots, c_i, \dots \rangle \dots (3)$$

$$\text{defined by } c_i = \sum_{k=0}^i a_k b_{i-k}, i = 0, 1, 2, \dots (4)$$

A systolic convolver M that we design consists of a buffer and a semi-infinite array of identical processors, called systolic cells, C_i , $i = 1, 2, \dots$, that operate synchronously at discrete time steps driven by a common clock. See Fig. 1. The input (a_i, b_i) , $i = 0, 1, 2, \dots$, is fed serially to the leftmost cell via the buffer, and from which the serial output c_j , $j = 0, 1, 2, \dots$, is also obtained. The buffer receives a pair of (a_i, b_i) , $i = 0, 1, 2, \dots$, at time $t = i$ from the host computer at the rate of one pair / one step. The buffer also outputs c_j to the host computer at time $t = j + k$ at the rate of one symbol/ one step, where k is some fixed integer, for any nonnegative integer j . We establish the following main theorem. The design of the convolver and its correctness will be given below (Propositions 1 to 4).

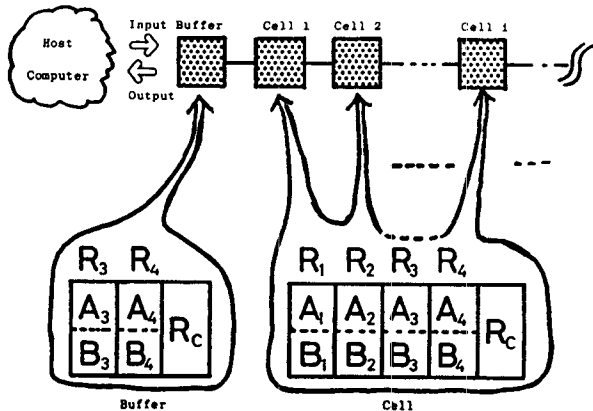


Fig. 1 An illustration of real-time systolic convolver.

[Theorem 1] The systolic convolver M presented below can compute the convolution c_i ($i = 0, 1, 2, \dots$) of two infinite sequences a_j , b_j ($j = 0, 1, 2, \dots$) in real-time, that is, for any i , $i = 0, 1, 2, \dots$, M outputs c_i at time $t = i + k$, where k is some fixed integer. The number of data registers in each systolic cell is minimum. Further M outputs the convolution c_i ($i = 0, 1, 2, \dots, 2n - 1$) of a_j , b_j ($j = 0, 1, 2, \dots, n - 1$) in exactly $2n + O(1)$ steps, using $\lfloor n/2 \rfloor + O(1)$ systolic cells.

2.2 Design

First we design a systolic convolver with each cell having four pairs of data registers, since it helps us to understand the correctness of our convolver. Afterward we will show that three-pair is the necessary and sufficient data-register number for the real-time convolution computation.

Basic cell definition:

Each systolic cell has four pairs of data registers R_j ($1 \leq j \leq 4$) and R_c . Each R_j consists of two subregisters A_j and B_j , storing (a_k^j, b_k^j) for some k . The buffer has three registers acting similarly as R_3 , R_4 and R_c of the systolic cell. See Fig. 1. Each register is used for the following purposes:

- R_1, R_2 : Data holding register.
- R_3 : A pipeline register which transmits data

- to the right neighbour cell at unit speed.
- R_4 : Stores data temporary.
- R_c : Stores the partial sum of convolution and transmits it to the left neighbour R_c at unit speed.

In the description below we use the following symbolic conventions.

$R_j^t(i)$: Denotes the content of R_j in C_i at step t . Similar notations are used for other registers.
 $R = \phi$, $R \neq \phi$, $R \leftarrow \phi$: Each denotes that the register R is empty, is not empty, and is set empty, respectively.

Parallel Data Routing Scheme:

Initialization: At time $t = 0$ we assume that:

$$\begin{aligned} \text{Buffer: } R_3^0(\text{buffer}) &= (a_0, b_0), \quad R_4^0(\text{buffer}) = \phi \\ C_i (i \geq 1): R_j^0(i) &= \phi \quad (1 \leq j \leq 4) \end{aligned} \quad \dots\dots(5)$$

At time t ($t \geq 1$):

$$\text{Buffer: } R_3^t(\text{buffer}) = (a_t, b_t), \quad R_4^t(\text{buffer}) = R_3^{t-1}(\text{buffer}) \quad \dots\dots(6)$$

$$C_i (i \geq 1): \left. \begin{aligned} &\text{if } (R_3^{t-1}(i-1) = \phi) \text{ then } \{ C_i \text{ does nothing} \} \\ &\text{else if } (R_1^{t-1}(i) = \phi) \text{ then } \{ R_1^t(i) + R_3^{t-1}(i-1) \} \dots(A) \\ &\text{else if } (R_2^{t-1}(i) = \phi) \text{ then } \{ R_2^t(i) + R_3^{t-1}(i-1) \} \dots(B) \\ &\text{else if } (R_3^{t-1}(i) = \phi) \text{ then } \{ R_3^t(i) + R_3^{t-1}(i-1) \} \dots(C) \\ &\text{else } \{ R_3^t(i) + R_3^{t-1}(i-1), \quad R_4^t(i) + R_3^{t-1}(i) \} \dots(D) \end{aligned} \right\} \dots\dots(7)$$

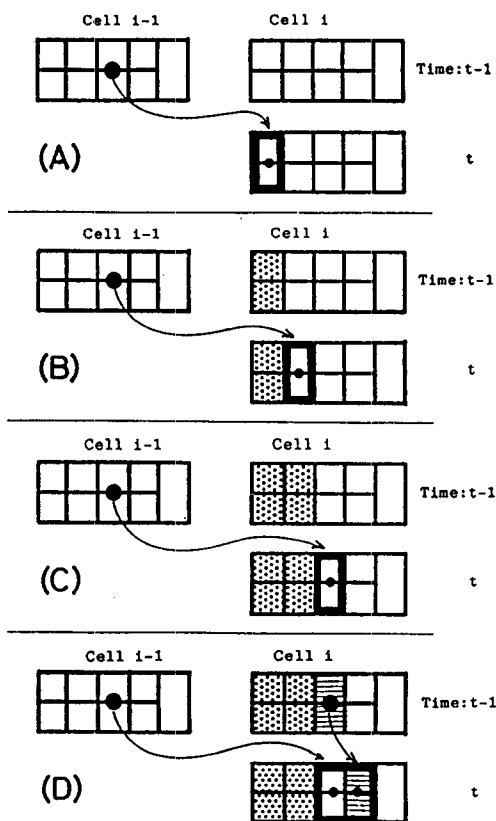


Fig. 2 Data routing scheme for real-time systolic convolver.

Basic operations:

The operation of the systolic cell consists of data routing which involves R_j ($1 \leq j \leq 4$) and convolution computation on R_j . Both of them are performed simultaneously, however, we will explain them separately for ease of description and understanding.

In the equation (7) we regard $R_3^{t-1}(i-1)$ as R_3^{t-1} (buffer) in the case $i = 1$. C_i does nothing at time t while $R_3^{t-1}(i-1)$ is empty. Let t be the time that $R_3(i-1)$ has its first data. Then, within the next three steps, that is, $t+1$, $t+2$, and $t+3$, C_i performs the operation (A), (B), and (C) in this order. After that C_i follows (D) at every step. Each data-pair is transmitted through the pipeline register R_3 in the right direction at unit speed until it will find an empty R_1 and/or R_2 . When they are found, R_1 has a priority for storing the data. The next pair will be loaded in R_2 in the same cell. Once a pair is stored in R_1 or R_2 , it will stay within that cell forever. The content of R_3 is also copied by R_4 in the same cell at the next step and is stored temporary in it. Fig. 2 shows our data routing scheme given above.

■ Computation rule for convolution:

● Initialization: $R_c^0(\text{buffer}) = R_c^0(i) = \phi$, for any $i \geq 1$ (8)

● At time $t (t \geq 1)$:
Buffer: $R_c^t(\text{buffer}) = R_c^{t-1}(1)$ (9)

$C_i (i \geq 1)$: Each cell C_i obeys the rule given below according to the number of pairs of data loaded in its $R_j (1 \leq j \leq 4)$ at time $t-1$. We will refer them as rule 1, 2, 3, and 4, respectively.

- No-pair: C_i does nothing for its R_c at time t .
- One-pair: $R_c^t(i) = A_1^{t-1}(i) B_1^{t-1}(i)$ Rule 1
- Two-pair: $R_c^t(i) = A_1^{t-1}(i) B_2^{t-1}(i) + A_2^{t-1}(i) B_1^{t-1}(i)$ Rule 2
- Three-pair: $R_c^t(i) = A_1^{t-1}(i) B_2^{t-1}(i) + A_2^{t-1}(i) B_2^{t-1}(i) + A_3^{t-1}(i) B_1^{t-1}(i)$ Rule 3 .. (10)
- Four-pair: $R_c^t(i) = R_c^{t-1}(i+1) + A_1^{t-1}(i) B_3^{t-1}(i) + A_3^{t-1}(i) B_1^{t-1}(i) + A_2^{t-1}(i) B_4^{t-1}(i) + A_4^{t-1}(i) B_2^{t-1}(i)$ Rule 4

Fig. 3 shows our computation rule for convolution. Exactly the following equations hold for R_c in the buffer and $C_i (i \geq 1)$. Consult Fig. 4 for the help of understanding of these equations. The mark @ in R_c denotes that its content is partial.

$$R_c^t(\text{buffer}) = \begin{cases} \phi & (t < 3) \\ c_{t-3} & (t \geq 3) \end{cases} \dots\dots\dots (11)$$

$$R_c^t(1) = \begin{cases} \phi & (t < 2) \\ c_{t-2} & (t \geq 2) \end{cases} \dots\dots\dots (12)$$

$$R_c^t(i) = \begin{cases} \phi & (t < 3i - 1, i \geq 2) \\ c_{t+i-3}^@ & (t \geq 3i - 1, i \geq 2) \end{cases} \dots\dots\dots (13)$$

2.3 Validity

We will show informally the correctness of our convolver. Let c_i be the term such that:

$$c_i = a_i b_0 + a_{i-1} b_1 + \dots\dots\dots + a_1 b_{i-1} + a_0 b_i$$

defined in (4). The index i can be represented as $i = 4k + \ell$, where k is any nonnegative integer and ℓ is any integer in $\{0, 1, 2, 3\}$. Then,

$$c_i = c_{4k+\ell} = w_1 + w_2 + \dots\dots + w_k + w_{k+1} \dots\dots (14),$$

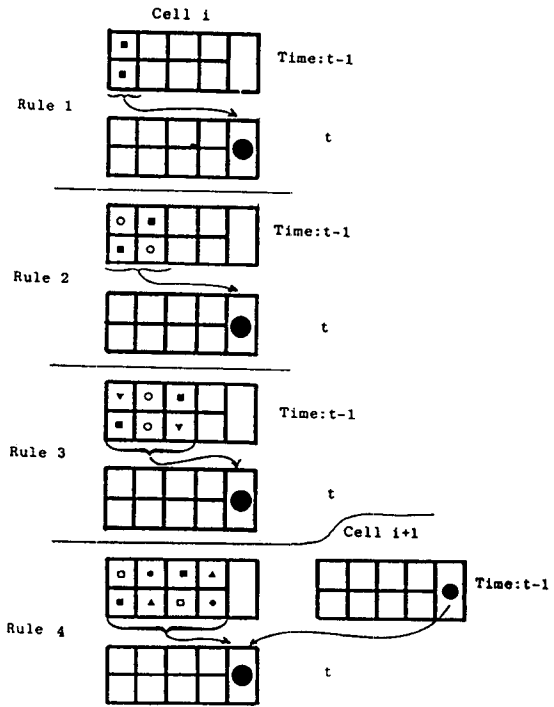


Fig.3 Computation rules for convolution.

where w_1, w_2, \dots, w_{k+1} are defined as follows:

$$w_j = a_{i-(2j-2)} b_{2j-2} + a_{i-(2j-1)} b_{2j-1} + a_{2j-2} b_{i-(2j-2)} + a_{2j-1} b_{i-(2j-1)}, 1 \leq j \leq k \text{ and}$$

$$w_{k+1} = \begin{cases} a_{2k} b_{2k} & , \text{when } \ell = 0 \\ a_{2k} b_{2k+1} + a_{2k+1} b_{2k} & \ell = 1 \\ a_{2k} b_{2k+2} + a_{2k+1} b_{2k+1} + a_{2k+2} b_{2k} & \ell = 2 \\ a_{2k} b_{2k+3} + a_{2k+1} b_{2k+2} + a_{2k+3} b_{2k} + a_{2k+2} b_{2k+1} & \ell = 3 \end{cases}$$

The computation of the value $c_{4k+\ell}$ is made on M as follows: The systolic cell C_{k+1}^{ℓ} firstly computes w_{k+1} , using rule 1, 2, 3, or 4 depending on whether $\ell^{k+1} = 0, 1, 2, \text{ or } 3$, respectively. At the next step this partial sum is transmitted to the left neighbour cell C_k and $w_k + w_{k+1}$ is computed on it, using rule 4. And at the next step, applying rule 4, C_{k-1} computes $w_{k-1} + w_k + w_{k+1}$, and so on. And k steps later, $w_1 + w_2 + \dots + w_k + w_{k+1}$ is obtained on C_1 . On the next step this value is

processed by the buffer. See Fig. 4. Exactly the following proposition holds. We omit its proof.

[Proposition 1] For any nonnegative integer x in $\{0, 1, 2, \dots, k\}$,

$$R_c^{3k+l+2+x}(k+1-x) = w_{k+1} + w_k + \dots + w_{k-x+1}$$

By letting $x = k$ in [Proposition 1], we get the following equation:

$$R_c^{4k+l+2}(1) = w_{k+1} + w_k + \dots + w_1 = w_{4k+l}$$

Thus, for any $i, i = 0, 1, 2, \dots, c_i$ is obtained at the buffer at time $t = i + \frac{1}{2}$ in real-time. The computation of c_{i+k+l} is started on C_{k+1} and uses C_k, C_{k-1}, \dots, C_1 (k cells). So the systolic real-time convolver M^1 uses $\lfloor n/2 \rfloor + O(1)$ systolic cells for the computation of convolution $c_i (i = 0, 1, 2, \dots, 2n - 1)$ of $a_j, b_j (j = 0, 1, 2, \dots, n - 1)$.

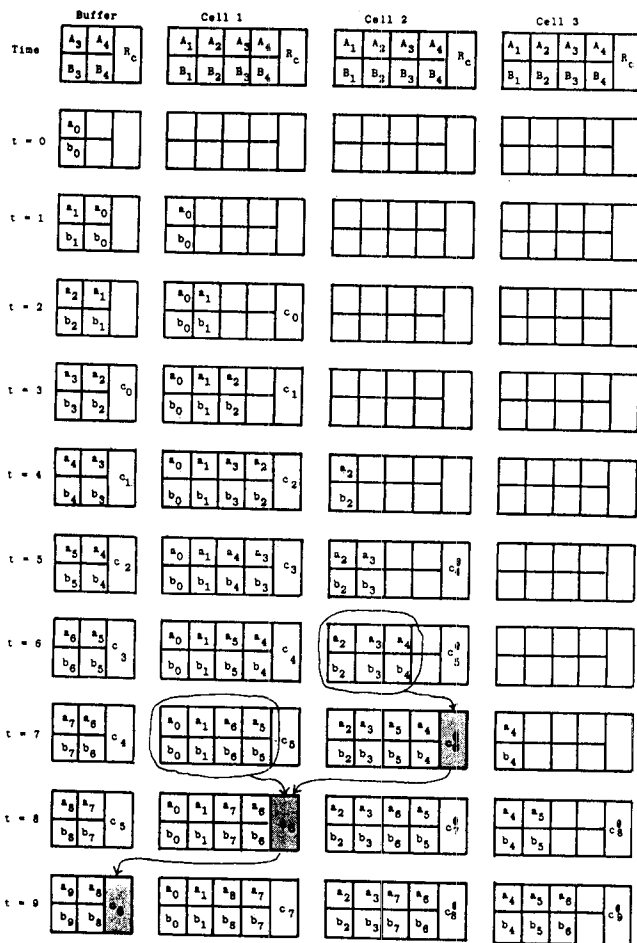


Fig.4 Snapshots of our real-time systolic convolver of two infinite sequences ($t=0$ to 9).

2.4 Optimality

In this section it is shown that our systolic convolver M is optimum not only in the parallel time complexity but also in the number of registers

of each systolic cell. From the definition of I/O rates the time optimality in [Theorem 1] is easily seen. Next we show that three pairs of data registers in each cell l , excluding R_c , are minimum for the real-time systolic computation of convolution.

[Proposition 2] The three pairs of data registers $R_1, R_2,$ and R_3 in each cell are necessary and sufficient for any real-time systolic convolver.

(Proof) Let $c_i = a_i b_0 + a_{i-1} b_1 + \dots + a_1 b_{i-1} + a_0 b_i$. We refer the set of cells $C_1, C_2, \dots, C_{i/4}$ of M as region I. See Fig. 5. The following observations are obtained.

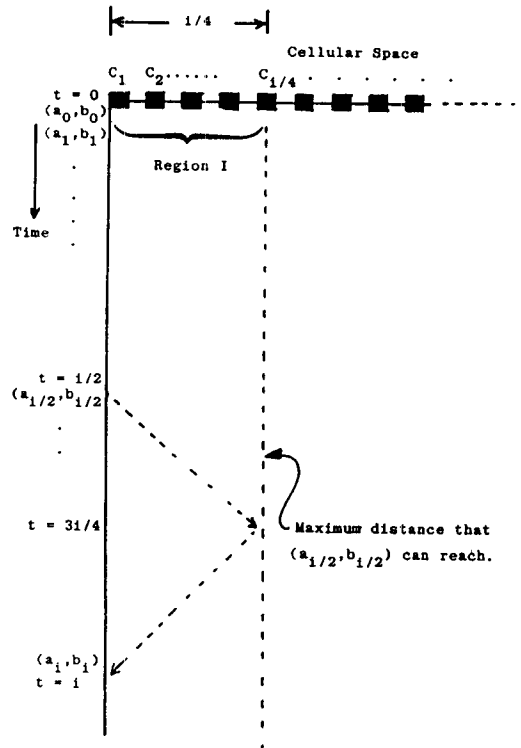


Fig.5 A region of systolic cells which involve the computation of $c_i = a_i b_0 + a_{i-1} b_1 + \dots + a_0 b_i$.

(1) It is the subterm $a_{i/2} b_{i/2}$ that can be computed firstly on M within $i/2 + 1$ subterms which constitute c_i , since it is the first subterm that both of the input data are ready for the computation. Moreover the multiplication of $a_{i/2} b_{i/2}$ must be made on some cell in the region I , since the pair is fed at time $t = i/2$ and the result must be output at time $t = i$ in real-time. From the same reason $(a_i, b_i), i = 0, 1, 2, \dots, i/2 - 1$ must stay at somewhere in the region I .

(2) The $i/4$ pairs of data fed from time $t = 3i/4$ to i cannot go out of the region I , even if they march in the right direction at its maximum speed.

Therefore M must hold $3i/4 (= i/2 + i/4)$ pairs of input data on the region I consisting of $i/4$

systolic cells. So three pairs of data registers are necessary for our computation.

By a slight observation we show that the register R_4 of each systolic cell can be removed from M . The R_4 is used only for the rule 4. When we apply the rule 4, the following register R is used instead of $R_4^{t-1}(i)$.

$$R = \begin{cases} R_1^{t-1}(i+1), & \text{when } R_2^{t-1}(i+1) = \phi, \\ R_2^{t-1}(i+1), & \text{when } R_2^{t-1}(i+1) \neq \phi \text{ and } R_3^{t-1}(i+1) = \phi, \\ R_3^{t-1}(i+1), & \text{otherwise.} \end{cases}$$

It is easily seen from (7) that the content of $R_4^{t-1}(i)$ is the same as that of R above. ■

3. Other convolvers and their applications

We will develop several variations and applications of our systolic real-time convolver designed in the preceding section. A slight modification enables us to use M for the computation of the convolution of one finite and one infinite sequences, which is an important problem in the fields of digital filtering.

[Theorem 2] The systolic convolver M can compute the convolution $c_i (i = 0, 1, 2, \dots)$ of an infinite sequence $a_j (j = 0, 1, 2, \dots)$ and a finite sequence $b_\ell (\ell = 0, 1, 2, \dots, k-1)$ in real-time, using only $\lfloor k/2 \rfloor + O(1)$ systolic cells.

(Proof sketch) We use $*$ as an end mark of the finite sequence $b_\ell, \ell = 0, 1, 2, \dots, k-1$. The input to M is $(a_j, *)$ for any $j \geq k$. The number of systolic cells necessary for the computation is $\lfloor k/2 \rfloor + O(1)$, since the product of $*$ and a_j is zero. ■

If multiplication and addition operations are interpreted as character-comparison and boolean AND, respectively, then the convolution considered

above becomes the pattern matching problem [13], [15], [22]. In [22] and [29], were presented systolic pattern matching algorithms with global data broadcasting and/or preloading. In the next theorem it is shown that the both broadcasting and preloading are not necessarily essential operations in the design of real-time systolic pattern matcher.

[Theorem 3] For any infinite long "pattern" and any "text" of length k , the systolic convolver M can detect and output the positions of all occurrences of the text in the pattern in real-time, using $\lfloor k/2 \rfloor + O(1)$ systolic cells.

The following theorem for open convolutions of two finite sequences is easily obtained from Theorems 1 and 2.

[Theorem 4] The systolic convolver M can compute the convolution $c_i (i = 0, 1, 2, \dots, n+m-1)$ of any two finite sequences $a_j (j = 0, 1, 2, \dots, n-1)$ and $b_\ell (\ell = 0, 1, 2, \dots, m-1)$ in exactly $n+m+O(1)$ steps, using $\lfloor \min(m,n)/2 \rfloor + O(1)$ systolic cells.

Lastly we apply our convolver to the design of systolic real-time polynomial divider. Our design is superior to the previous ones [21], [33], and [36] in both the time and space complexities. In Table 1 we give a summary of the designs. Let's begin with the definitions. Let $A(x)$ and $B(x)$ be any polynomials of degree n and $m (n \geq m)$, respectively, such that:

$$A(x) = a_0 x^n + a_1 x^{n-1} + \dots + a_{n-1} x + a_n \dots (15),$$

$$B(x) = b_0 x^m + b_1 x^{m-1} + \dots + b_{m-1} x + b_m \dots (16).$$

Then there exist unique polynomials $C(x)$ and $D(x)$ that satisfy the following division property:

$$A(x) = B(x) C(x) + D(x), \text{ where } \dots (17)$$




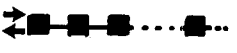
Design	Number of Systolic Cells	Time Complexity (cycles)	I/O Rates	I/O Architecture
Kung[21] (1982)	m	$2n + m$	1-data/2-step	Sequential I/O at opposite ends 
Wada, Mizuno and Kawaguchi [33] (1985)	$n + 1$	$2n + 2$	1-data/1-step	Sequential I/O at opposite ends 
Zak and Hwang [36] (1985)	$n - m + 1$	$2n - m + 2$	1-data/1-step	Parallel I/O 
this paper	$\lfloor \min(n-m, m)/2 \rfloor$	n (optimum)	1-data/1-step	Sequential I/O at one end 

Table 1 Performance comparison of systolic polynomial divider in the case where degree n polynomial is divided by degree m polynomial.

$$c_0 x^{n-m} + c_1 x^{n-m-1} + \dots + c_{n-m-1} x \quad \dots\dots(18)$$

$$D(x) = d_0 x^{m-1} + d_1 x^{m-2} + \dots\dots + d_{m-2} x + d_{m-1} \quad \dots\dots(19)$$

Let $z_i (i = 0, 1, 2, \dots, n+1)$ be the convolution of (b_0, b_1, \dots, b_m) and $(c_0, c_1, \dots, c_{n-m})$. Then the following equation holds.

$$a_i = \begin{matrix} z_i & 0 \leq i \leq n-m \\ z_i + d_{i-n+m-1} & n-m+1 \leq i \leq n \end{matrix} \quad \dots\dots(20)$$

Therefore coefficients of $C(x)$ and $D(x)$ are determined by the following equations:

$$c_i = \{ a_i - (b_i c_0 + b_{i-1} c_1 + \dots\dots + b_1 c_{i-1}) \} / b_0, \quad 0 \leq i \leq n-m. \quad \dots\dots(21)$$

$$d_j = a_{i+n-m+1} - \sum_{j=0}^{n-m} b_{i+n-m+1-j} c_j, \quad 0 \leq i \leq m-1. \quad \dots\dots(22)$$

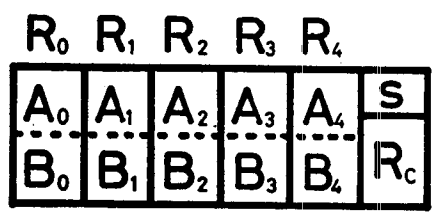


Fig.6 Systolic cell C_1 for the polynomial divider.

We want to design a real-time systolic polynomial divider A which outputs both $c_i (0 \leq i \leq n-m)$ and $d_j (0 \leq j \leq m-1)$ in exactly $n+O(1)$ steps, when coefficients of $A(x)$ and $B(x)$ are given as inputs of A. The operation of A is based on M which computes the open convolution of (b_0, b_1, \dots, b_m) and $(c_0, c_1, \dots, c_{n-m})$ in [Theorem 4]. Therefore the number of systolic cells needed for the division is $[\min(n-m+1, m+1)/2] + O(1) = [\min(n-m, m)/2] + O(1)$. All of the systolic cells of A, except C_1 , are the same as these of M. So in the below, we will only give the description of C_1 .

The cell C_1 consists of several registers shown in Fig. 6. The R_0 and S registers are added to the previous design. The R_0 is the data register storing $a_i (0 \leq i \leq n-1)$, and b_0 , respectively. The S register assumes either "q" or "r" state, which notifies C_1 that the current computation is for the quotient or the remainder, respectively. From time $t = 0$ to $n-m+1$, S assumes "q" and after that, from $t = n-m+2$ to the end, S assumes "r". The change of the state in S is caused by the signal fed by the host computer at time $t = n-m+1$. C_1 performs the operation given in Table 2 depending on whether S is "q" or "r", respectively.

In Fig. 7 we show an example of the systolic polynomial division on A in the case where $n = 7$ and $m = 3$. We give the following theorem without proof.

[Theorem 5] For the division of any degree n polynomial by any degree m polynomial ($n \geq m$), there exists a systolic array A which can compute the polynomial division in exactly $n + O(1)$ steps on $[\min(n-m, m)/2] + O(1)$ systolic cells. The divider A is time-optimum and register-number-minimum.

4. Conclusion

In this work we have presented a best systolic convolver which can compute the convolution $c_i (i = 0, 1, 2, \dots, n+m-1)$ of any two finite sequences $a_j (j = 0, 1, 2, \dots, n-1)$ and $b_\ell (\ell = 0, 1, 2, \dots, m-1)$ in optimum real-time, exactly in $n+m+O(1)$ steps, using $[\min(m,n)/2] + O(1)$ systolic cells. The number of data registers in each systolic cell is minimum. Several variations and applications of our convolver, such as real-time pattern matcher and time-optimum polynomial divider, are also developed. Our convolver design is the best one as far as the design is within the systolic architectures.

Acknowledgement

The author is pleased to acknowledge the considerable assistance of Prof. R. Vollmar. This research is supported in part by an Alexander-von-Humboldt Foundation Fellowship and by the Grant-in-Aid(No.61750346) for Scientific Research of the Ministry of Education, Culture and Science of Japan.

References

- [1] Apostolico, A. and Negro, A.; "Systolic algorithms for string manipulations", IEEE Trans. on Computers, Vol. C-33, No. 4, pp.361-364, (1984).
- [2] Atrubin, A.J.; "A one-dimensional real-time iterative multiplier", IEEE Trans. on EC., EC-14, pp.394-399, (1965).
- [3] Baudet, G.M., Preparata, F.P., and Vuillemin, J.E.; "Area-time optimal VLSI circuits for convolution", IEEE Trans. on Computers, Vol. C-32, No. 7, pp.684-688, (1983).
- [4] Cappello, P.R. and Steiglitz, K.; "Unifying VLSI array designs with geometric transformations", Proc. of the IEEE Intern. Conf. on Parallel Processing, pp.448-457, (1983).
- [5] Cappello, P.R. and Steiglitz, K.; "Unifying VLSI array design with linear transformations of space-time", in "Advances in Computing Research", Vol. 2, pp.23-65, JAI Press, (1984).
- [6] Chen, I-Ngo and Willoner, R.; "An $O(n)$ parallel multiplier with bit-sequential input and output", IEEE Trans. on Computers, Vol. C-28, No.10, pp.721-727, (1979).
- [7] Cohen, D.; "Mathematical approach to iterative computation network", Proc. of The Computer Architecture, pp.226-239, (1978).
- [8] Cole, S.N.; "Real-time computation by n-dimensional iterative arrays of finite state machines", IEEE Trans. on Computers, Vol.C-18, No.4, pp.349-365, (1969).
- [9] Culik II, K. and Fris, I.; "Topological transformations as a tool in the design of systolic networks", Tech. Rep of Waterloo Univ., CS-84-11, (1984).

[10] Danielsson, P.E.; "Serial/parallel convolvers", IEEE Trans. on Computers, Vol. C-33, No. 7, pp.652-667, (1984).
 [11] Ersoy O.; "Semisystolic array implementation of circular, skew circular, and linear convolutions", IEEE Trans. on Computers, Vol. C-34, No. 2, pp.190-196, (1985).

[12] Evans, R.A., Wood, D., Wood, K., McCanny, J.V., McWhirter, J.G., and McCabe, A.P.H.; "A CMOS implementation of a systolic multi-bit convolver chip", in "VLSI 83(eds. F. Anceau and E.J. Aus)", pp.227-235, NorthHolland, (1983).
 [13] Fisher, M.J. and Paterson, M.S.; "String-matching and other products",

$t = 0$: $R_j^0(1) = R_c^0(1) = \phi$, ($0 \leq j \leq 4$), and $s^0 = q$
 $t = 1$: $R_0^1(1) = (a_0, b_0)$, $R_j^1(1) = R_c^1(1) = \phi$, ($1 \leq j \leq 4$), and $s^0 = q$
 $t \geq 2$: Given below.

```

while (St-1 = "q") do { A0t(1) + A3t-1(buffer); B0t(1) + B0t-1(1);
  if(R1t-1(1) = φ) then { A1t(1) + Z ; B1t(1) + B3t-1(buffer); Rct(1) + Z }
  else if(R2t-1(1) = φ) then { A2t(1) + ZX ; B2t(1) + B3t-1(buffer); Rct(1) + ZX }
  else if(R3t-1(1) = φ) then { A3t(1) + ZXX ; B3t(1) + B3t-1(buffer); Rct(1) + ZXX }
  else if(R4t-1(1) = φ) then { A3t(1) + ZXXX ; B3t(1) + B3t-1(buffer); Rct(1) + ZXXX;
    A4t(1) + A3t-1(1); B4t(1) + B3t-1(1) }
  else { A3t(1) + ZXXXX ; B3t(1) + B3t-1(buffer); Rct(1) + ZXXXX ; A4t(1) + A3t-1(1);
    B4t(1) + B3t-1(1) }
},
  
```

where Z, ZX, ZXX, ZXXX, and ZXXXX are defined as follows:

$$Z: A_0^{t-1}(1)/B_0^{t-1}(1),$$

$$ZX: (A_0^{t-1}(1) - A_1^{t-1}(1) B_1^{t-1}(1))/B_0^{t-1}(1),$$

$$ZXX: (A_0^{t-1}(1) - A_1^{t-1}(1) B_2^{t-1}(1) - A_2^{t-1}(1) B_1^{t-1}(1))/B_0^{t-1}(1),$$

$$ZXXX: (A_0^{t-1}(1) - A_1^{t-1}(1) B_3^{t-1}(1) - A_2^{t-1}(1) B_2^{t-1}(1) - A_3^{t-1}(1) B_1^{t-1}(1))/B_0^{t-1}(1),$$

$$ZXXXX: (A_0^{t-1}(1) - A_1^{t-1}(1) B_3^{t-1}(1) - A_2^{t-1}(1) B_4^{t-1}(1) - A_3^{t-1}(1) B_1^{t-1}(1) - A_4^{t-1}(1) B_2^{t-1}(1) - R_c^{t-1}(2))/B_0^{t-1}(1).$$

```

while (St-1 = "r") do { A0t(1) + A3t-1(buffer); B0t(1) + B0t-1(1);
  if(R2t-1(1) = φ) then { A2t(1) + "*" ; B2t(1) + B3t-1(buffer); Rct(1) + $ }
  else if(R3t-1(1) = φ) then { A3t(1) + "*" ; B2t(1) + B3t-1(buffer); Rct(1) + $$ }
  else if(R4t-1(1) = φ) then { A3t(1) + "*" ; B3t(1) + B3t-1(buffer); Rct(1) + $$$ ;
    A4t(1) + A3t-1(1); B4t(1) + B3t-1(1) }
  else { A3t(1) + "*" ; B3t(1) + B3t-1(buffer); Rct(1) + $$$$ ; A4t(1) + A3t-1(1);
    B4t(1) + B3t-1(1) }
},
  
```

where \$, \$\$, \$\$\$, and \$\$\$\$ are defined as follows:

$$$: A_0^{t-1}(1) - A_1^{t-1}(1) B_1^{t-1}(1),$$

$$$$: A_0^{t-1}(1) - A_1^{t-1}(1) B_2^{t-1}(1) - A_2^{t-1}(1) B_1^{t-1}(1),$$

$$$$$: A_0^{t-1}(1) - A_1^{t-1}(1) B_3^{t-1}(1) - A_2^{t-1}(1) B_2^{t-1}(1) - A_3^{t-1}(1) B_1^{t-1}(1),$$

$$$$$$: A_0^{t-1}(1) - A_1^{t-1}(1) B_3^{t-1}(1) - A_2^{t-1}(1) B_4^{t-1}(1) - A_3^{t-1}(1) B_1^{t-1}(1) - A_4^{t-1}(1) B_2^{t-1}(1) - R_c^{t-1}(2)$$

Table 2 Operation of C_1 of the real-time systolic polynomial divider.

SIAM-AMS, Proceedings, Vol.7, pp.113-125, (1974)

[14] Foster, M.J. and Kung, H.T.; "The design of special purpose VLSI chips", IEEE Computer, Vol. 13, No. 1, pp.26-40, (1980).

[15] Galil, Z.; "Real-time algorithms for string-matching and palindrome recognition", Proc. of the 8th ACM Symp. on Theory of Computing, pp.161-173, (1976).

[16] Gnanasekaran, R.; "On a bit-serial input and bit-serial output multiplier", IEEE Trans. on Computers, Vol. C-32, No. 9, pp.878-880, (1983).

[17] Gnanasekaran, R.; "A fast serial-parallel binary multiplier", IEEE Trans. on Computers, Vol. C-34, No. 8, pp.741-744, (1985).

[18] Hurson, A.R. and Shirazi, B.; "A systolic multiplier unit and its VLSI design", Proc. of Computer Architecture, pp.302-309, (1985).

[19] Ibarra, O.H., Kim, S.M., and Palis, M.A.; "Designing systolic algorithms using sequential machines", IEEE Trans. on Computers, Vol. C-35, No. 6, pp.531-542, (1986).

[20] Knuth, D.E.; "The Art of Computer Programming", Vol.2, pp297-299, Second Edition, (1980).

[21] Kung, H.T.; "Use of VLSI in algebraic computation: Some suggestions", Proc. of 1981 ACM Symp. on Symbolic and Algebraic Computation, pp.218-222, (1981).

[22] Kung, H.T.; "Why systolic architecture", IEEE Computer, Jan., pp.37-46, (1982).

[23] Leiserson, C.E. and Saxe, J.E.; "Optimizing synchronous systems", J. of VLSI and Computer Systems, Vol.1, No.1, pp.41-67, (1983).

[24] Li, Guo-Jie and Wah, B.W.; "The design of optimal systolic arrays", IEEE Trans. on Computers,

Vol. C-34, No. 1, pp.66-77, (1985).

[25] Luk, W.K. and Vuillemin J.E.; "Recursive implementation of optimal time VLSI integer multipliers", Tech. Rep. of Carnegie-Mellon Univ., CMU-CS-84-149, (1984)

[26] Lyon, R.F.; "Two's complement pipeline multiplier", IEEE Trans. on Communications, Comm-24, pp.418-425, (1976).

[27] McWhirter, J.G. and McCanny, J.V.; "Novel multibit convolver/correlator chip design based on systolic array principles", Proc. of SPIE, Vol. 341, Real Time Processing V(1982), pp.66-73, (1982).

[28] McWhirter, J.G., Wood, D., Wood, K.W., Evans, R.A., McCanny, J.V., and McCabe A.P.H.; "Multibit convolution using a bit level systolic array", IEEE Trans. on Circuits and Systems, Vol. CAS-32, No. 1, pp.95-99, (1985).

[29] Mukhopadhyay, A.; "Hardware algorithms for nonnumeric computation", IEEE Trans. on Computers, Vol. C-28, No. 6, pp.384-394, (1979).

[30] Preparata, F.P.; "A mesh-connected area-time optimal VLSI multiplier of large integers", IEEE Trans. on Computers, Vol. C-32, No. 2, pp.194-198, (1983).

[31] Ullman, J.D.; "Computational Aspects of VLSI", Computer Science Press, p.495, (1984).

[32] Umeo, H.; "A class of SIMD algorithms implemented on systolic VLSI arrays", IEEE Proc. of the Inter. Conf. on Parallel Processing, pp.374-376, (1984).

[33] Wada, k., Mizuno, M., and Kawaguchi, K.; "Systolic algorithms for polynomial multiplication and division", Tech. Rep of IECE(Japan), AL84-60, pp.97-106, (in Japanese), (1985).

[34] Wang, Chin-Liang, Wei, Che-Ho, and Chen Sin-Hong; "Improved systolic array for linear discriminant function classifier", Electronics Letters, 16th Jan., Vol. 22, No. 2, pp.85-86, (1986).

[35] Wood, D., Evans, R.A., and Wood, K.W.; "An 8 bit serial convolver chip on a bit level systolic array", Proc. of the Custom Integrated Circuits Conf., pp.256-261, (1983).

[36] Zak, S.H. and Hwang, K.; "Polynomial division on systolic arrays", IEEE Trans on Computers, Vol. C-34, No. 6, pp.577-578, (1985).

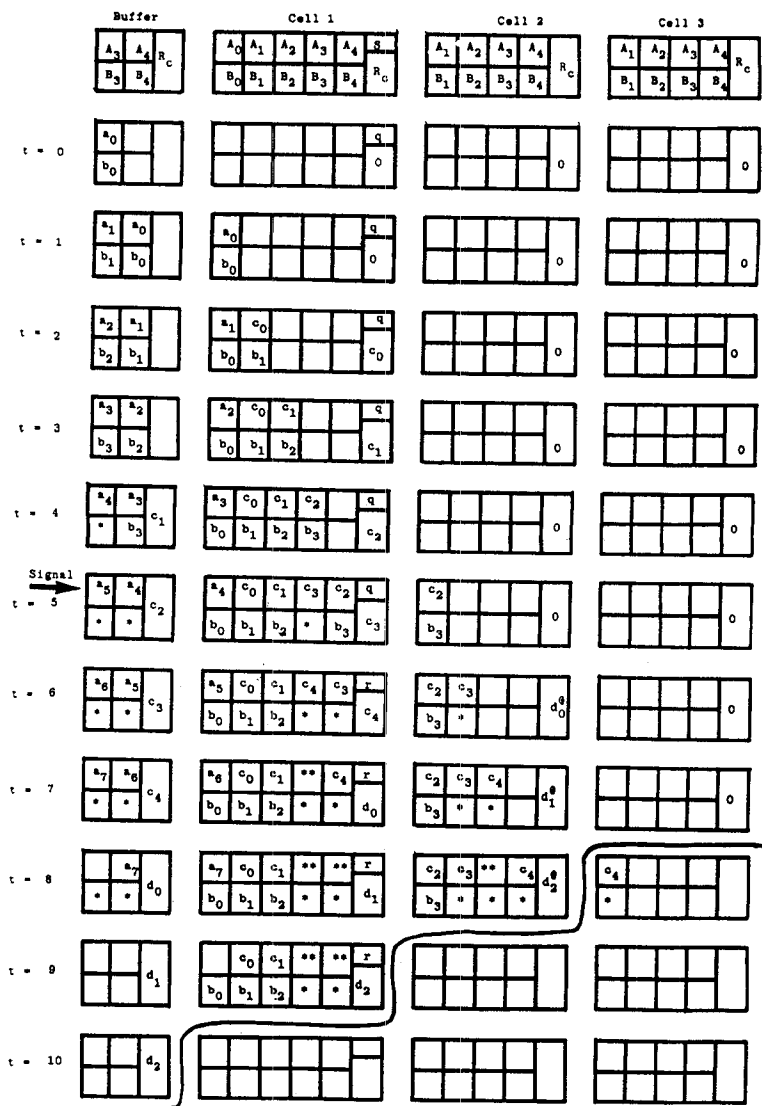


Fig. 7 Snapshots of the real-time systolic polynomial divider($t = 0$ to 10).