Computers Math. Applic. Vol. 27, No. 1, pp. 1–10, 1994
Printed in Great Britain. All rights reserved

0898-1221/94 $6.00 + 0.00
Copyright© 1993 Pergamon Press Ltd

A Systolic Algorithm for Dynamic Programming

CHAU-JY LIN
Department of Applied Mathematics, National Chiao Tung University
Hsinchu, 30050, Taiwan, R.O.C.

(Received and accepted May 1993)

Abstract—We present a formal systolic algorithm to solve the dynamic programming problem for an optimal binary search tree. For a fixed integer j such that $2 \leq j \leq n$, first we derive a linear systolic array to evaluate the minimal cost $c_{i,j}$ for $1 \leq i < j$. Then we combine these $(n-1)$ linear systolic arrays to form a two-dimensional systolic array. The computational model consists of $\lceil (n^2 + 2n - 4)/4 \rceil$ processing elements. The algorithm requires $(2n-3)$ time steps to solve this problem. The elapsed time within a time step is independent of the problem size n. It is suitable for the VLSI implementation due to the identical and simple structure of processing elements. We also prove the correctness of this algorithm by induction.

1. INTRODUCTION

In recent years, computer science has devoted much attention to problems related to highly structured computer systems, and their potential applications. Current very large scale integrated (VLSI) technology requires uniformity and regularity in both the processing elements (PEs for short) and their integration on a given chip. These requirements naturally lead one to conceive of a multiprocessor system with a large number of relatively simple and uniform processors interconnected in a regular pattern. A simple computational model of such a system is the systolic array [1]. A systolic array is a model of parallel computer consisting of rudimentary PEs, each capable of performing some simple operation. Many systolic arrays have been designed to solve some problems [2–5]. A parallel algorithm which can be executed on a systolic array is called a **systolic algorithm**.

The dynamic programming for an optimal binary search tree can be solved sequentially in $O(n^3)$ running time steps for n its problem size [6]. In this paper, we present a formal parallel algorithm, which is executed on a computational model of two-dimensional systolic array, to solve this problem. The operations of each individual PE are designed explicitly. The design procedure to obtain the systolic algorithm is considered in detail. We hope that this concept can be applied to obtain systolic algorithms for solving some other problems.

A systolic array can be considered as a network which consists of a few types of computational PEs. Since there is no shared memory in our systolic array and the data broadcasting behavior is not allowed, the data transformation between PEs should be handled by some explicit communication links. A communication link with a name e-link which joins PE1 to PE2 is called an **input link** of PE2 and an **output link** of PE1. The data being sent out from PE1 via e-link is denoted by e_{out}. The data being received by PE2 via e-link is denoted by e_{in}. Each PE performs

This work was supported partially by the National Science Council in Taiwan, R.O.C. under the contract number NSC 82-0208-M009-22.

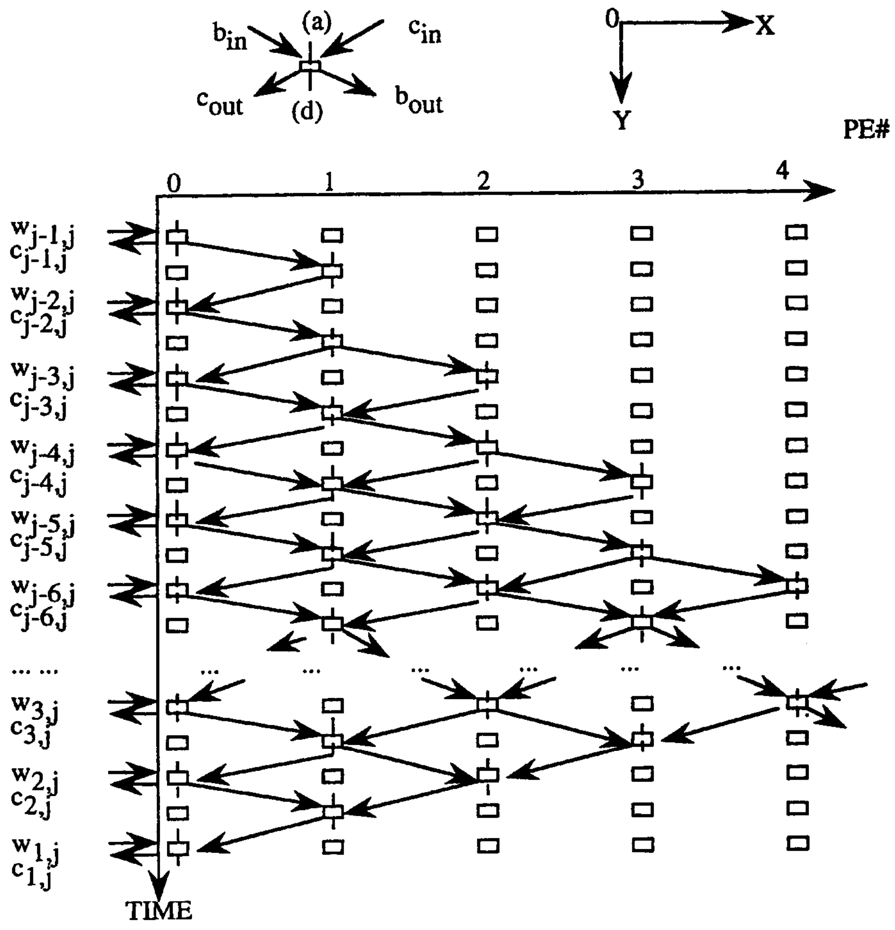

Figure 1. The graph for the dynamic programming problem.

the following three tasks:

(1) to receive data from its input links;
(2) to execute one loop of the designed systolic algorithm;
(3) to send data to its output links.

The maximal time unit (all PEs are considered) to do these three tasks is called a **time step**. In our algorithm the elapsed time unit within a time step is constant. Moreover, if the e-link is labeled with γ delays (denoted by γD) for γ a positive integer, it means that when PE1 sends out its e_{out} at the time step t, then such e_{out} is the e_{in} of PE2 at the time step $t+\gamma$. Our systolic array requires five communication links. Only one of them has $\gamma = 2$; the remaining links have only one delay. The symbol γD on a link will be omitted when $\gamma = 1$.

2. THE DESIGN PROCEDURE FOR SOLVING DYNAMIC PROGRAMMING

Without loss of generality, the dynamic programming problem for an optimal binary search tree can be stated as follows.
"For $1 \le i < q < j \le n$, find $c_{i,j} = w_{i,j} + \min_{i<q<j} \{c_{i,q} + c_{q,j}\}$ where $w_{i,j}$ are the given values; the initial cost value $c_{i,i+1}$ are considered as the value of $w_{i,i+1}$ for $1 \le i \le n-1$."

For any fixed integers i, j with $1 \le i < j \le n$, the process to evaluate the value of the $c_{i,j}$ is considered as follows.

(1) Assume that $c_{i,q}$ and $c_{q,j}$ have been ready for all q with $i < q < j$.

(2) Group the above known cost values into the form $c_{i,q} + c_{q,j}$. Then two forms of them are set as a pair. These pairs are denoted by $P_k = \{c_{i,j-k} + c_{j-k,j},\ c_{i,i+k} + c_{i+k,j}\}$ for $1 \le k \le \lfloor (j-i)/2 \rfloor$.

(3) During the evaluation of $c_{i,j}$, the above pairs P_k will be referred in the decreasing order of k. That is, suppose that P_k is used at the time step t_k, then $k' < k''$ if and only if $t_{k'} > t_{k''}$.

(4) When P_1 is used to evaluate $c_{i,j}$ at a time step t_1, the given value $w_{i,j}$ is involved at the time step $t_1 + 1$. This $t_1 + 1$ is also the time step that $c_{i,j}$ has produced.

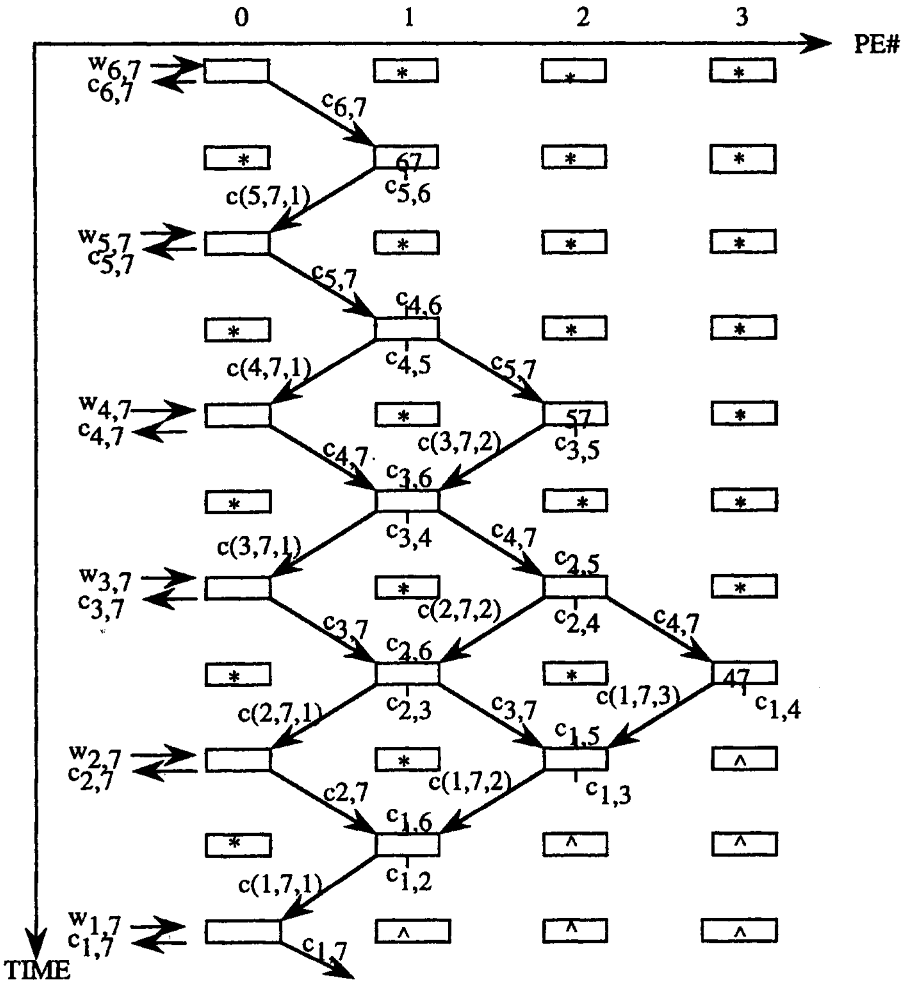

Figure 2. The graph for evaluating $c_{i,7}$.

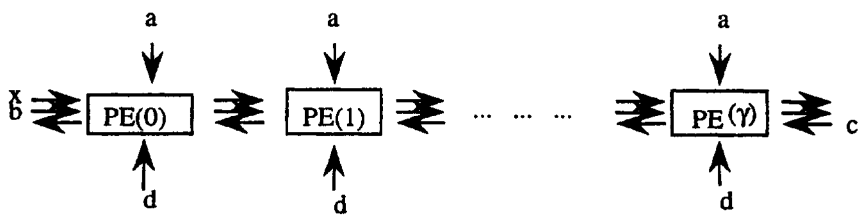

Figure 3. The linear systolic array with a fixed integer j.

Let $l = \lfloor (j-i)/2 \rfloor$. We know that P_l is the first pair to be used in order to evaluate the $c_{i,j}$. Suppose that t_0 is the time step to evaluate the minimal value in P_l. We define $c(i,j,l) = \min\ P_l$

and call this $c(i,j,l)$ the **P_l-partial result** of $c_{i,j}$. Following $(l-1)$ time steps, the pairs P_k for $1 \le k < l$ will be referred within the time interval $(t_0, t_0 + l - 1]$ respectively. That is, the P_k-partial result of $c_{i,j}$ which is defined as $c(i,j,k) = \min\{c(i,j,k+1),\, c_{i,j-k}+c_{j-k,j},\, c_{i,i+k}+c_{i+k,j}\}$ will be evaluated at the time step $t_0 + l - k$. If we let $c(i,j,l+1)$ be a large enough number "∞," then the $c(i,j,l)$ can be redefined as the $\min\{c(i,j,l+1),\, c_{i,j-l}+c_{j-l,j},\, c_{i,i+l}+c_{i+l,j}\}$. Since we have $c_{i,j} = c(i,j,1) + w_{i,j}$, this $c_{i,j}$ will be evaluated at the time step $t_0 + l$. We define $c(i,j,0) = c(i,j,1) + w_{i,j}$ as our desired result $c_{i,j}$.

Let j be a fixed integer such that $2 \le j \le n$. Under the preceding description to evaluate the $c_{i,j}$, we design a linear systolic array to produce the value of $c_{i,j}$ for $1 \le i < j$. First, an xy-plane with integer coordinates is chosen. The x-axis is the index of PE and the y-axis the time step (see Figure 1), where each PE will be referred to as PE(x,t) for $x \ge 0$ and $t \ge 1$. The solving strategy of evaluating $c_{i,j}$ is described as follows.

(1) From the description of the dynamic programming problem and the order of P_k to be referred, we know that the computation of $c_{i,j}$ requires the value of $c_{u,j}$ for $i < u$. Thus, when $c_{u,j}$ coming from a PE$(0,t)$, we need one communication link (say b-link) to transmit such $c_{u,j}$ passing through PE$(0,t)$, PE$(1,t+1), \ldots,$ to PE$(r,t+r)$ for $r = j - u$. Then this $c_{u,j}$ will be stored in a register, say E, of PE$(r,t+r)$ in order to evaluate the following $c_{v,j}$ for $1 \le v < u$.

(2) Since the value of $c_{u,j}$ is propagating on the b-link for $(j-u)$ time steps, and then this $c_{u,j}$ will be stored in a register of a PE, we need a control link (say x-link) to indicate that at what time step this $c_{u,j}$ will be stored in the register E.

(3) The P_k-partial results $c(i,j,k)$ of $c_{i,j}$ for $1 \le k \le \lfloor (j-i)/2 \rfloor$ are transmitted on a link (say c-link) from PE(k,t) to PE$(k-1,t+1)$, in order to be used to evaluate the following P_{k-1}-partial results $c(i,j,k-1)$ of the $c_{i,j}$.

The diagram of our designed consideration is shown in Figure 1, where b-link, x-link (not shown) and c-link have one delay of time step. The symbols (a) and (d) appearing in Figure 1 mean that these two corresponding input values are received from the output links of other linear arrays which are evaluating the $c_{\alpha,u}$ for $1 \le \alpha < u < j$. An illustrative example with $j = 7$ is shown in Figure 2, where the numbers 67, 57, 47 appearing respectively in three PEs mean that at the time steps 2, 5, 8, these PEs have stored $c_{6,7}$, $c_{5,7}$, $c_{4,7}$ in their registers, respectively. The symbols "*" and "^" mean that PEs are in waiting and stopping states, respectively.

Figure 1 is projected along the time axis (y-axis) to obtain a linear systolic array as shown in Figure 3, where the $c_{i,j}$ comes out from PE(0) at the time step $t = 2(j-i) - 1$ for $1 \le i < j$. There are $\gamma + 1$ PEs to be used with $\gamma = \lceil j/2 \rceil - 1$. In Figure 3, once the $c_{i,j}$ comes out, it is transmitted along the b-link for $(j-i)$ time steps and finally it is stored in the register E of PE$(j-i)$ provided that this PE exists. Since the control link x-link (having the same direction of b-link) is used to indicate that this $c_{i,j}$ will be stored in the register E of PE$(j-i)$ at the time step $t = 3(j-i) - 1$, we let the $x_{\text{in}} = (j-i)$ in PE(0) at the time step $t = 2(j-i) - 1$. When the $c_{i,j}$ is propagating on the b-link, the value on the x-link is decreased by one at each time step. This implies that if $x_{\text{in}} = 1$ is recognized by the PE$(j-i)$ then this PE stores its b_{in} into its register E.

From the above design with $2 \le j \le n$, we obtain $(n-1)$ linear systolic arrays each consisting of $\lceil j/2 \rceil$ PEs. We combine these $(n-1)$ linear systolic arrays to form a two-dimensional array as shown in Figure 4, where $c_{i,j}$ comes out from the c-link of PE$(j,0)$ at the time step $t = 2(j-i)-1$. In order to let these data $c_{i,j}$ in time arrive at the PEs which require such data $c_{i,j}$, we introduce two communication links (the a-link and the d-link) between the PEs in Figure 4. The d-link goes from PE(j,k) to PE$(j+1,k+1)$ and the a-link from PE(j,k) to PE$(j+1,k)$. Moreover, the d-link has one time step delay but the a-link has two delays. The $c_{i,j}$ is first transmitted along the d-link for $(j-i)$ time steps then it is transmitted along the a-link after this time step. That is, once the $c_{i,j}$ has been computed at PE$(j,0)$, it passes through PE$(j+1,1)$, PE$(j+2,2), \ldots,$

PE$(2j-i, j-i)$ along the d-link, then passing through PE$(2j-i, j-i+1)$, PE$(2j-i, j-i+2)$, $\ldots$, PE$(2j-i, n)$ along the a-link. The choice of d-link or a-link for propagating the $c_{i,j}$ is controlled by a new control link (say y-link). Having a similar consideration on the x-link, the y-link indicates whether the $c_{i,j}$ being on the d-link should be redirected to the a-link or not. Note that the work of the y-link can be replaced by that of the x-link, because any $c_{i,j}$ is propagating on d-link and on b-link for the same $(j-i)$ time steps. Thus, the x-link and y-link have the same indicator in a PE at any time step. So we do not show the y-link in our systolic array.

Note that there are two types of PEs in our computational model. One type contains the PE$(j,0)$. The other type contains the PE(j,k) for $k > 0$. We use an index I in each PE to distinguish these two types. When we consider the behavior of each PE as a finite state machine, the diagram of its states is shown in Figure 5, where s_0, s_1, s_2, and s_3 are the initial, executing, stopping and waiting states, respectively; "^, *" are two special symbols; "~^, ~*" denote the logical complements of "^, *" respectively; $D = \{d_{\text{in}}\}$ denotes that the input data d_{in} of a PE is used as a signal to control the states of PEs. When $d_{\text{in}} = \hat{}$ is recognized by a PE, then this PE stops its execution after this time step.

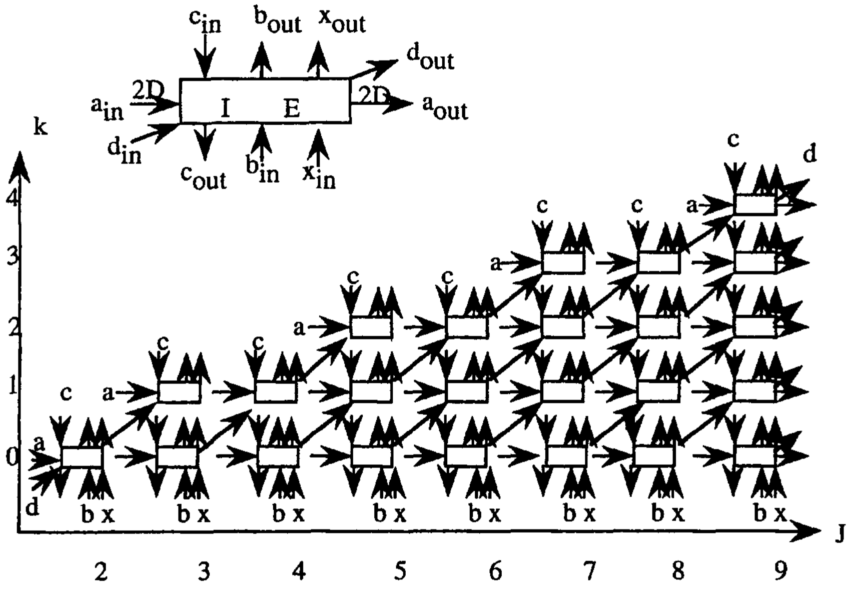

Figure 4. The systolic array for the dynamic programming.

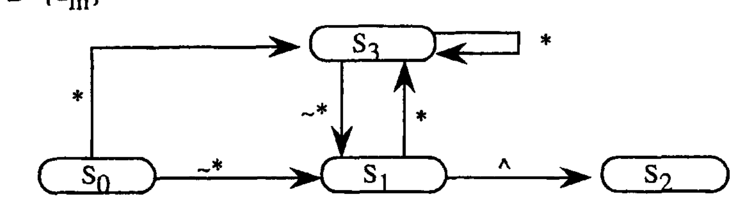

Figure 5. The states of PEs.

3. THE SYSTOLIC ALGORITHM

To design the systolic algorithm, we define the corresponding five procedures as follows:

signal-stop $\equiv$ **begin** $b_{\text{out}} = \hat{}$; $a_{\text{out}} = \hat{}$; $c_{\text{out}} = \hat{}$; $d_{\text{out}} = \hat{}$; $x_{\text{out}} := \hat{}$ **end**.

waiting $\equiv$ **begin** $b_{\text{out}} = *$; $a_{\text{out}} = \infty$; $c_{\text{out}} = \infty$; $d_{\text{out}} = *$; $x_{\text{out}} := *$ **end**.

adding–w-value $\equiv$

begin $x_{\text{out}} := x_{\text{in}}$; $a_{\text{out}} := \infty$; $c_{\text{out}} := b_{\text{in}} + c_{\text{in}}$; $b_{\text{out}} := b_{\text{in}} + c_{\text{in}}$; $d_{\text{out}} := b_{\text{in}} + c_{\text{in}}$ **end**.

storing–c-value ≡ **begin** $E := b_{\text{in}};\ a_{\text{out}} := d_{\text{in}};\ b_{\text{out}} := *;\ d_{\text{out}} := *;\ x_{\text{out}} := *$ **end.**
decreasing–x-value ≡ **begin** $b_{\text{out}} := b_{\text{in}};\ x_{\text{out}} := x_{\text{in}} - 1;\ a_{\text{out}} := a_{\text{in}};\ d_{\text{out}} := d_{\text{in}}$ **end.**

Algorithm: **DYNAMIC_PROGRAM** ≡

[Initial state].
Set $c_{\text{in}} = 0$ in PE$(j,0)$ for $2 \le j \le n$. Set $c_{\text{in}} = \infty$ for PE(j,k) with $k > 0$. Set $E = *$ in all PEs. Set $a_{\text{in}} = \infty$ in PE$(2m+1,m)$ for m positive integer. Set $I = 0$ in PE$(j,0)$ and $I = 1$ in PE(j,k) for $k > 0$. The input data for the d-link in PE$(j,0)$ are $\{0,*,0,*,\dots,0,\hat{}\}$, where the 0 appears exactly $(j-1)$ times. The input data for the b-link and the x-link in PE$(j,0)$ are the sequences $\{w_{j-1,j},*,w_{j-2,j},*,\dots,w_{1,j},*\}$ and $\{1,*,2,*,\dots,j-1,*\}$, respectively. That is, the components of links are interspaced within stars which provide the delay needed so the components can meet PE$(j,0)$ correctly.
[Execution state].

```
repeat /* do parallel for all PEs */
    if $d_{\text{in}} = \hat{}$ then signal-stop;
    if $d_{\text{in}} = *$ then waiting;
    if $d_{\text{in}} \ne \hat{}$ and $d_{\text{in}} \ne *$ then
      begin
        case I of
        0: adding–w-value;
        1: begin
               if $x_{\text{in}} = 1$ then storing–c-value else decreasing–x-value;
               $c_{\text{out}} := \min\{c_{\text{in}}, a_{\text{in}} + b_{\text{in}}, d_{\text{in}} + E\}$
           end
      end
until $d_{\text{in}} = \hat{}$.
```

4. THE CORRECTNESS PROOF

We use the notation PE$(j,k)[d_{\text{in}} = 1, c_{\text{out}} = 2, \dots]t = t_0$ to denote the statement that PE(j,k) has $d_{\text{in}} = 1$, $c_{\text{out}} = 2$, and so on at the time step $t = t_0$. The symbol "A ⇒ B" means that the statement A implies the statement B. From the initial state of our systolic algorithm, two lemmas are obtained.

LEMMA 1. *For* $0 \le k < \lceil j/2 \rceil < j \le n$, *we have* PE$(j,k)[d_{\text{in}} = \hat{}, d_{\text{out}} = \hat{}]t = 2j - k - 2$.

PROOF. The input data for the d-link of PE$(j-k,0)$ are $\{0,*,0,*,\dots,0,\hat{}\}$ with zeros appearing $(j-k-1)$ times. The procedure of *signal-stop* implies that

$$\begin{aligned}
&\text{PE}(j-k,0)[d_{\text{in}} = \hat{}, d_{\text{out}} = \hat{}]t = 2(j-k-1).\\
\Rightarrow &\text{PE}(j-k+1,1)[d_{\text{in}} = \hat{}, d_{\text{out}} = \hat{}]t = 2(j-k-1)+1.\\
\Rightarrow &\text{PE}(j,k)[d_{\text{in}} = \hat{}, d_{\text{out}} = \hat{}]t = 2(j-k-1)+k = 2j-k-2.
\end{aligned}$$ ∎

LEMMA 2. *For* $1 \le k < j \le n$, *we have* PE$(j,k)[x_{\text{in}} = 1, x_{\text{out}} = *]t = 3k - 1$.

PROOF. From the initial values on the x-link, we have PE$(j,0)[x_{\text{in}} = \delta, x_{\text{out}} = \delta]t = 2\delta - 1$ for $1 \le \delta \le j-1$. The procedure *decreasing–x-value* executes the statement: "if $x_{\text{in}} > 1$ then $x_{\text{out}} = x_{\text{in}} - 1$ else $x_{\text{out}} = *$" in all the PE(j,k) with $k \ne 0$. Thus, we have

$$\begin{aligned}
&\text{PE}(j,0)[x_{\text{in}} = k, x_{\text{out}} = k]t = 2k-1.\\
\Rightarrow &\text{PE}(j,1)[x_{\text{in}} = k, x_{\text{out}} = k-1]t = 2k.\\
\Rightarrow &\text{PE}(j,k)[x_{\text{in}} = 1, x_{\text{out}} = *]t = 3k-1 \quad \text{for } 1 \le k < \lceil j/2 \rceil.
\end{aligned}$$ ∎

LEMMA 3. *For $1 \le k < \lceil j/2 \rceil < j \le n$, PE$(j,k)$ assigns its b_{in} into its register E and PE(j,k) assigns its d_{in} to a_{out} at the time step $t = 3k - 1$.*

PROOF. By Lemma 2 and the execution of procedure *storing–c-value* which performs the assignments $E := b_{\text{in}}$ and $a_{\text{out}} := d_{\text{in}}$. ▮

The major part of the correctness verification in our algorithm is the following proposition. (**) "The value of $c_{i,j}$ is produced in PE$(j,0)$ at the time step $t = 2(j-i)-1$ for $1 \le i < j \le n$."

That is, we will show PE$(j,0)[c_{\text{out}} = c_{i,j}]t = 2(j-i)-1$. This proposition (**) will be proved by mathematical induction on the variable $N = (j-i)$.

LEMMA 4. *The proposition (**) is true for $N = 1$.*

PROOF. When $N = 1$, this proposition (**) is shown by the initial state and the execution of *adding–w-value*. That is, we have PE$(j,0)[b_{\text{in}} = w_{j-1,j}, c_{\text{in}} = 0, c_{\text{out}} = b_{\text{in}} + c_{\text{in}}]t = 1$. Thus, $c_{j-1,j} = w_{j-1,j}$ is produced in PE$(j,0)$ at the time step $t = 2(j-j+1)-1 = 1$. ▮

Now we assume that the proposition (**) is true for all $N \le m$ with m a given positive integer. We want to show that the proposition (**) is true for $N = m+1$. The following three lemmas are considered under this induction hypothesis.

LEMMA 5. *For $1 \le i < j \le n$, $N = m+1$ and $1 \le k \le \lfloor (j-i)/2 \rfloor$, we have*

(a) PE$(j,k)[E = c_{j-k,j}]t = 3k - 1$.
(b) PE$(j,k)[d_{\text{in}} = c_{i,j-k}]t = 2j - k - 2i - 1$.
(c) PE$(j,k)[b_{\text{in}} = c_{i+k,j}]t = 2j - k - 2i - 1$.
(d) PE$(j,k)[a_{\text{in}} = c_{i,i+k}]t = 2j - k - 2i - 1$.

PROOF.

(a) Following Lemma 3, we have PE$(j,k)[x_{\text{in}} = 1]t = 3k - 1$. By the induction assumption with $j - (j-k) = k \le m$, we have

$$\text{PE}(j,0)[b_{\text{out}} = c_{\text{out}} = c_{j-k,j}]t = 2k - 1.$$
$$\Rightarrow \text{PE}(j,k)[b_{\text{in}} = c_{j-k,j}]t = 3k - 1.$$

Hence, we obtain PE$(j,k)[E = c_{j-k,j}]t = 3k - 1$ by the procedure *storing-c-value*.

(b) By $(j-k) - i \le m$, PE$(j-k,0)[d_{\text{out}} = c_{\text{out}} = c_{i,j-k}]t = 2(j-k-i)-1$ and this $c_{i,j-k}$ is transferring on the d-link for k time steps. We obtain

$$\text{PE}(j,k)[d_{\text{in}} = c_{i,j-k}]t = 2(j-k-i) - 1 + k = 2j - k - 2i - 1.$$

(c) By $j - (i+k) \le m$, PE$(j,0)[b_{\text{out}} = c_{\text{out}} = c_{i+k,j}]t = 2(j-i-k) - 1$ and this $c_{i+k,j}$ is transferring on the b-link for k time steps. We have

$$\text{PE}(j,k)[b_{\text{in}} = c_{i+k,j}]t = 2j - k - 2i - 1.$$

(d) By $(i+k)-i \le m$, PE$(i+k,0)[d_{\text{out}} = c_{\text{out}} = c_{i,i+k}]t = 2k-1$ and this $c_{i,i+k}$ is propagating on the d-link for k time steps. We have

$$\text{PE}(i+2k,k)[d_{\text{in}} = c_{i,i+k}]t = 3k - 1.$$

From Lemma 2 together with $k < \lceil (i+2k)/2 \rceil$, we have

$$\text{PE}(i+2k,k)[x_{\text{in}} = 1]t = 3k - 1.$$
$$\Rightarrow \text{PE}(i+2k,k)[a_{\text{out}} = d_{\text{in}} = c_{i,i+k}]t = 3k - 1.$$

Then this $c_{i,i+k}$ is transferring on the a-link for $2(j-i-2k)$ time steps. Therefore, we have

$$\text{PE}(j,k)[a_{\text{in}} = c_{i,i+k}]t = 2j - k - 2i - 1.$$ ▮

From the value of k with the constraint $1 \leq k \leq \lfloor (j-i)/2 \rfloor$, we obtain $2j - k - 2i - 1 \geq 3k - 1$. Thus, the two values of $c_{i,j-k} + c_{j-k,j}$ and $c_{i,i+k} + c_{i+k,j}$ for evaluating the $c(i,j,k)$ will be retrieved and performed by PE(j,k) in time. The evaluation of the P_k-partial result of $c_{i,j}$ is at the time step $t = 2j - k - 2i - 1$ for $1 \leq k \leq \lfloor (j-i)/2 \rfloor$. For an illustrative example of the evaluation of $c_{1,7}$, we have $i = 1$, $j = 7$ and $k \leq 3$. The evaluations of $c(1,7,3)$, $c(1,7,2)$ and $c(1,7,1)$ are at the time step 8, 9, 10 respectively. The value $c_{1,7} = w_{1,7} + c(1,7,1)$ is obtained at $t = 11$ (see Figure 2).

Let $l = \lfloor (j-i)/2 \rfloor$. The following two lemmas show that the first P_l-partial result $c(i,j,l)$ of $c_{i,j}$ is evaluated at the time steps $t = 3l - 1$ or $t = 3l + 1$ depending on whether $(j-i)$ is an even or an odd integer.

LEMMA 6. *For $1 \leq i < j \leq n$, $N = m+1$, let $l = \lfloor (j-i)/2 \rfloor$. If $m+1$ is an even integer, then we have*

(a) *The P_l-partial result $c(i,j,l)$ of $c_{i,j}$ is evaluated in* PE(j,l) *at the time step $t = 3l - 1$.*
(b) *The value of $c_{i,j}$ is coming out from* PE$(j,0)$ *at the time step $t = 2(j-i) + 1$.*

PROOF.

(a) Following the description in Section 2, we have $P_l = \{c_{i,j-l} + c_{j-l,j},\ c_{i,i+l} + c_{i+l,j}\}$ with $l = \lfloor (j-i)/2 \rfloor$ and $c(i,j,l) = \min P_l$. By $(j-i) = N = m+1$ is even, we have $l = (j-i)/2$ and $j - l = i + l$. Hence, the two terms appearing in P_l are identical. That is, we have $c(i,j,l) = c_{i,j-l} + c_{j-l,j}$. By the induction assumption with $(j-l) - i = 2l - l = l \leq m$, we have

$$\text{PE}(j-l,0)[c_{\text{out}} = c_{i,j-l},\ d_{\text{out}} = c_{i,j-l}]t = 2(j-l-i) - 1 = 2(i+l-i) - 1 = 2l - 1.$$
$$\Rightarrow \text{PE}(j,l)[d_{\text{in}} = c_{i,j-l}]t = 3l - 1.$$

Similarly, we have

$$\text{PE}(j,0)[c_{\text{out}} = c_{j-l,j}, b_{\text{out}} = c_{j-l,j}]t = 2l - 1.$$
$$\Rightarrow \text{PE}(j,l)[b_{\text{in}} = c_{j-l,j}]t = 3l - 1.$$

From Lemma 2 we have

$$\text{PE}(j,l)[x_{\text{in}} = 1]t = 3l - 1.$$
$$\Rightarrow \text{PE}(j,l)[E = b_{\text{in}} = c_{j-l,j}]t = 3l - 1.$$

Note that PE$(j,l)[a_{\text{in}} = \infty, c_{\text{in}} = \infty]t = 3l - 1$ from the initial state. Thus, we obtain

$$\text{PE}(j,l)[c_{\text{out}} = c(i,j,l) = d_{\text{in}} + E = c_{i,j-l} + c_{j-l,j}]t = 3l - 1.$$

(b) From Lemma 5, the remaining P_k-partial results $c(i,j,k)$ of $c_{i,j}$ with $1 \leq k < l$ are evaluated in the time interval $[3l, 4l-2]$. That is, we have

$$\text{PE}(j,k)[c_{\text{in}} = c(i,j,k+1), c_{\text{out}} = c(i,j,k)]t = 3l - 1 + (l-k) \qquad \text{for } 1 \leq k \leq l.$$
$$\Rightarrow \text{PE}(j,0)[c_{\text{in}} = c(i,j,1), b_{\text{in}} = w_{i,j}, c_{\text{out}} = c_{\text{in}} + b_{\text{in}} = c_{i,j}]t = 4l - 1.$$

Since $j - i = m + 1 = 2l$, we have $4l - 1 = 2(m+1) - 1 = 2(j-i) - 1$. Hence, $c_{i,j}$ is coming out from PE$(j,0)$ at the time step $t = 2(j-i) - 1$ for $j - i = m + 1$. Therefore, the proposition $(**)$ is true for $N = m + 1$ which is an even integer. ∎

LEMMA 7. *For $1 \leq i < j \leq n$, $N = m+1$, let $l = \lfloor (j-i)/2 \rfloor$. If $m+1$ is an odd integer, then we have*

(a) *The P_l-partial result $c(i,j,l)$ of $c_{i,j}$ is evaluated in* PE(j,l) *at the time step $t = 3l + 1$.*
(b) *The $c_{i,j}$ is coming out from* PE$(j,0)$ *at the time step $t = 2(j-i) - 1$.*

PROOF.

(a) Since $j-i=m+1$ is an odd integer, we have $l=m/2$ and $j-i=2l+1$. As we have mentioned, the first P_l-partial result of $c_{i,j}$ is $c(i,j,l)=\min\{c_{i,j-l}+c_{j-l,j},\ c_{i,i+l}+c_{i+l,j}\}$; here we assume that $j-l>i+l$. We claim that these four cost values will appear in PE(j,l) at $t=3l+1$.

(1) By $(j-l)-i\le m$, PE$(j-l,0)[c_{\text{out}}=c_{i,j-l},\ d_{\text{out}}=c_{i,j-l}]t=2(j-l-i)-1=2l+1$.

$$\Rightarrow \text{PE}(j,l)[d_{\text{in}}=c_{i,j-l}]t=3l+1.$$

(2) By $j-(j-l)\le m$, PE$(j,0)[c_{\text{out}}=c_{j-l,j},\ b_{\text{out}}=c_{j-l,j}]t=2(j-j+l)-1=2l-1$.

$$\Rightarrow \text{PE}(j,l)[b_{\text{in}}=c_{j-l,j}]t=3l-1.$$

By Lemma 2 we have PE$(j,l)[x_{\text{in}}=1]t=3l-1$. Hence, PE$(j,l)[E=c_{j-l,j}]t=3l-1$. This implies PE$(j,l)[E=c_{j-l,j}]t=3l+1$.

(3) By $(i+l)-i\le m$, we have

$$\text{PE}(i+l,0)[c_{\text{out}}=c_{i,i+l},\ d_{\text{out}}=c_{i,i+l}]t=2(i+l-i)-1=2l-1.$$

By $l<\lceil (i+2l)/2\rceil$ and Lemma 2, we have

$$\text{PE}(i+2l,l)[x_{\text{in}}=1]t=3l-1.$$
$$\Rightarrow \text{PE}(i+2l,l)[d_{\text{in}}=c_{i,i+l},\ x_{\text{in}}=1,\ a_{\text{out}}=c_{i,i+l}]t=3l-1.$$
$$\Rightarrow \text{PE}(j,l)[a_{\text{in}}=c_{i,i+l}]t=3l-1+2(j-i-2l)=3l+1.$$

(4) By $j-(i+l)\le m$, PE$(j,0)[c_{\text{out}}=c_{i+l,j},\ b_{\text{out}}=c_{i+l,j}]t=2(j-i-l)-1=2l+1$.

$$\Rightarrow \text{PE}(j,l)[b_{\text{in}}=c_{i+l,j}]t=3l+1.$$

Therefore, these four cost values are already in PE(j,l) at $t=3l+1$. Hence, the P_l-partial result $c(i,j,l)$ of $c_{i,j}$ is evaluated in PE(j,l) at $t=3l+1$.

(b) From PE$(j,l)[c_{\text{out}}=c(i,j,l)]t=3l+1$ and the result of Lemma 5, the following P_k-partial results $c(i,j,k)$ with $1\le k<l$ are evaluated in the following $(l-1)$ time steps. That is, we have

$$\text{PE}(j,l)[c_{\text{out}}=c(i,j,l)]t=3l+1.$$
$$\Rightarrow \text{PE}(j,k)[c_{\text{in}}=c(i,j,k+1),\ c_{\text{out}}=c(i,j,k)]t=3l+1+l-k=4l-k+1 \quad \text{for } 1\le k<l.$$
$$\Rightarrow \text{PE}(j,1)[c_{\text{in}}=c(i,j,2),\ c_{\text{out}}=c(i,j,1)]t=4l.$$
$$\Rightarrow \text{PE}(j,0)[c_{\text{in}}=c(i,j,1),\ b_{\text{in}}=w_{i,j}, c_{\text{out}}=c_{\text{in}}+b_{\text{in}}=c_{i,j}]t=4l+1.$$
$$\Rightarrow \text{PE}(j,0)[c_{\text{out}}=c_{i,j}]t=4l+1.$$

Since $l=m/2$ and $m+1=j-i$, we have $4l+1=2(m+1)-1=2(j-i)-1$. Therefore, the proposition (**) is true when $N=m+1$ is an odd integer. ∎

THEOREM. *For $1\le i<j\le n$, the systolic algorithm DYNAMIC_PROGRAM is correct to produce $c_{i,j}$ in* PE$(j,0)$ *at the time step $t=2(j-i)-1$.*

PROOF. Under the mathematical induction, this theorem is proved by the results of Lemmas 4, 6 and 7. ∎

5. CONCLUSION

In this article, we present a formal systolic algorithm to solve the dynamic programming problem of an optimal binary search tree. For a fixed integer j such that $2 \leq j \leq n$, first we derive a linear systolic array with $\lceil j/2 \rceil$ PEs to evaluate the minimal cost $c_{i,j}$ for $1 \leq i < j$. Then we combine these $(n-1)$ linear systolic arrays to form a two-dimensional systolic array. Hence, this computational model consists of $\lceil (n^2 + 2n - 4)/4 \rceil$ PEs. The systolic algorithm requires $(2n-3)$ time steps to solve this problem. The elapsed time within a time step is independent of the problem size n. It is very suitable for the VLSI implementation due to the identical and simple structure of PEs. We also prove the correctness of this algorithm. In general, the verification of a parallel algorithm is difficult for the concurrent executions of many PEs. However, the mathematical induction is a suitable method to be used for verifying the correctness of a systolic algorithm. This designed consideration of our systolic algorithm can be applied to solve other problems, such as the combinatorial enumeration, the computational geometry and the graph theory problems. Furthermore, for developing the systolic algorithms, we are interested in the study of systolic algorithms such that the storage in any PE and the elapsed time of a time step are considered to be independent of the problem size.

REFERENCES

1. H.T. Kung, Why systolic architecture?, *IEEE Trans. Computers* **15**, 37–46 (1982).
2. C.Y. Chang and K. Yao, Systolic array processing of the sequential decoding algorithms, *Intern. J. it High Speed Computing* **1**, 465–480 (1989).
3. F.C. Lin and I.C. Wu, Broadcast normalization in systolic design, *IEEE Trans. on Comput.* **C-37**, 1121–1126 (1988).
4. J.A. Mchugh, *Algorithmic Graph Theory*, Prentic-Hall, Inc. Englewood Cliffs, NJ, (1990).
5. S.Y. Kung, On supercomputing with systolic/wavefront array processors, *Proc. of IEEE* **72**, 867–884 (1984).
6. E. Horowitz and S. Sahni, *Fundamentals of Computer Algorithms*, Computer Science Press, Inc., (1978).
7. C.A. Mead and L.A. Conway, *Introduction to VLSI Systems*, Addison-Wesley, Reading, MA, (1980).