Computers Math. Applic. Vol. 17, No. 12, pp. 1523–1533, 1989
Printed in Great Britain.

0097-4943/89 $3.00 + 0.00

A PARALLEL ALGORITHM FOR GENERATING COMBINATIONS

C.-J. LIN
Department of Applied Mathematics and Institute of Computer Engineering, National Chiao Tung University, Hsinchu, Taiwan 30050, Republic of China

(*Received* 3 *February* 1988; *in revised form* 19 *July* 1988)

Abstract—A parallel algorithm for generating all combinations of m items out of n given items in lexicographic order is presented. The computational model is a linear systolic array consisting of m identical processing elements. It takes $\binom{n}{m}$ time-steps to generate all the $\binom{n}{m}$ combinations. Since any processing element is identical and executes the same procedure, it is suitable for VLSI implementation. Based on mathematical induction, such algorithm is proved to be correct.

1. INTRODUCTION

Because of the drastically lowered hardware cost and the advancement of hardware technology, parallel processing becomes more and more feasible in practice. Using a parallel computer is a way to achieve higher computing speeds, this appealing approach has promptly increased interest in the area of design and analysis of parallel algorithms. The growing importance of parallel computers and parallel algorithms is highlighted in [1–5]. Systolic array is one of the parallel computation models. A systolic array architecture is specified by the timing of data movement and interconnection of processing elements (PEs) such that the movement of data is simple, regular, and uniform. These array processors are typically made up of identical PEs that operate in synchronous. Many examples of systolic array processor have been presented, e.g. in the fields of image processing, matrix manipulation, digital signal processing and the solver of simultaneously linear equations etc. However, only a few systolic array are designed for combinatorial enumeration problems.

Generating combinations is important in combinatorics. Several parallel algorithms [6–8] have been designed to solve this problem. However, these algorithms do not generate the combinations in lexicographic order, or they are not systolic algorithms. In this paper we design a parallel algorithm in a linear systolic array to generate all combinations of m out of n items in lexicographic order.

It is known that the combinations in lexicographic order can be generated sequentially in a straightforward way [9]. In [6] Chan and Akl presented a parallel algorithm to generate the combinations on a single-instruction–multiple-data (SIMD) machine which allows data to be read simultaneously from a shared memory. The PE must know its indexed position and also has a mark to indicate whether the PE is active in order to perform its program. Moreover, in [7] a parallel algorithm for generating the permutations of at most m out of n but not in lexicographic order was presented. The architecture consists of a linear array and a selector. Each PE has a stack of size m to store the necessary data during the execution of algorithm. This algorithm can easily be modified to generate combinations. In [8] for k is a given ranking number they proposed an algorithm to evaluate the m components of the kth combination within a PE. If we have $\binom{n}{m}$ PEs to be used, then the kth PE will be assigned to generate the kth combination, hence all of the $\binom{n}{m}$ combinations can be produced simultaneously. Since a combination can be produced in $O(n)$ time units, their algorithm generates all the combinations in $O(n)$ time units provided that $\binom{n}{m}$ PEs are available in a SIMD computer. In this paper, we only use a linear array consisting of m PEs, and these PEs are not necessary to know their indexed positions during the execution of algorithm. We use four registers to replace the stack of size m as shown in [7]. In [10] Semba presented a sequential algorithm to generate all the combinations of at most m out of n items in lexicographic order. Such

result can also be produced by the use of our algorithm and then by the execution of two recursive procedures which are described in Section 4.

2. THE COMPUTATIONAL MODEL AND THE PARALLEL ALGORITHM

A systolic array processor can be viewed as a network composed of a few types of computational PEs, and an array processor is often used as an attached processor linked with a host computer through an interface system. Within a systolic array, if it is necessary to send data from PE1 to PE2 then there exists a communication link (say e-link) from PE1 to PE2. We call such e-link an input link of PE2 and an output link of PE1, we also write e_{in} and e_{out} to denote the input value of PE2 and the output value of PE1 via e-link, respectively. Moreover, if the e-link is labeled with an integer γ delays, it means that when PE1 sends e_{out} to e-link at time t_1, then such e_{out} is the e_{in} of PE2 at time $t_1 + \gamma$. For convenience, we use e_{in}, e_{out} as the names of variable in our parallel algorithm.

Our computational model for generating combinations is a linear systolic array consisting of m identical PEs provided that only the adjacent PEs can communicate their data via communication links. Figure 1 indicates the layout of our computational model, any individual PE is referred to as PE(i) for $1 \leqslant i \leqslant m$, and each individual PE is specified in Fig. 2, where c, d, x and y are four communication links, each link is labeled with one delay, R, C, T are registers, F is a flag, and o_i is the output terminal. We assume that each PE performs the following three tasks. (1) Receiving data from its input of communication links with names c_{in}, x_{in}, d_{in}, and y_{in}, respectively. (2) Executing the functions that are described by an existing algorithm. (3) Sending data to its output of communication links with name c_{out}, x_{out}, d_{out}, and y_{out}, respectively.

We call the needed time units to do the above three tasks as a *time-step*. Because the communication x-link has one delay in our computational model, so the output value x_{out} of PE(i) at time-step t is the input value x_{in} of PE($i-1$) at time-step $t+1$. Similarly, the c_{out} of PE(i) at time-step t is the c_{in} of PE(i) at time-step $t+1$, and the y_{out}, d_{out} of PE(i) at time-step t are the y_{in}, d_{in} of PE($i+1$) at time-step $t+1$, respectively.

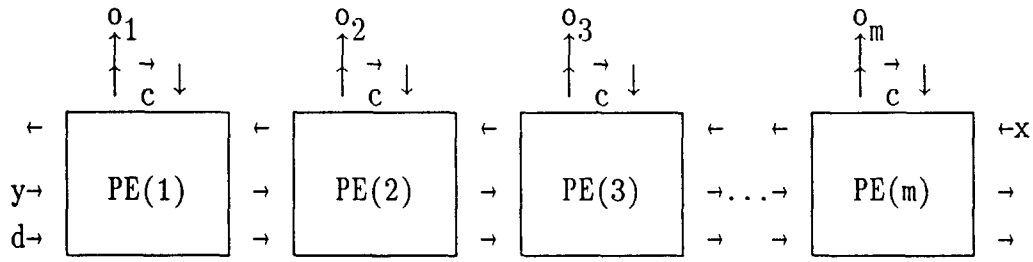

Fig. 1. The computational model.

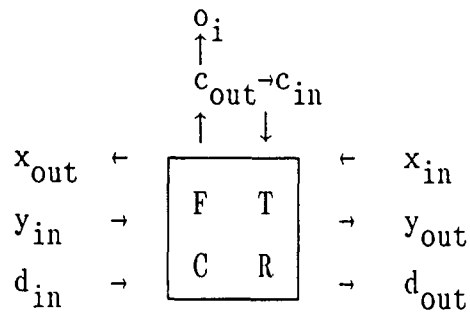

Fig. 2. The specification of PE(i).

Without loss of generality, it is assumed that the n given items are $1, 2, 3, \ldots, n$. By the definition of lexicographic order, if $A = \{a_1, a_2, \ldots, a_m\}$ is a combination, then $a_i \leqslant n - m + i$ for all $1 \leqslant i \leqslant m$. We call $n - m + i$ the *limit value* of the ith component of A, and denote it by R_i. The use of communication links and registers in each individual PE(i) ($1 \leqslant i \leqslant m$) are described as follows.

(1) The communication c-link transmits the input/output (c_{in}/c_{out}) of the ith component of any combination.

(2) The communication x-link indicates that whether the current output of c-link (i.e. c_{out}) is equal to its limit value.

(3) The communication y-link transmits data to the register T of PE($i + 1$), if PE($i + 1$) exists.

(4) The communication d-link transmits the same data as c_{out}, i.e. $d_{out} = c_{out}$ at all time-steps.

(5) Register R contains the limit value $R_i = n - m + i$.

(6) Register T stores a temporary element when the algorithm is working. T receives element from y-link.

(7) Register C contains a counter indicating at what time-step the element in T will be retrieved and assigned to c_{out}.

(8) Flag F indicates that if the condition "$c_{out} = R_i - 1$ and $x_{in} = 1$" is true then PE(i) is ready to transmit elements into the Ts in PE(k) for $i \leqslant k \leqslant m$.

Our parallel algorithm is shown in Algorithm 1. It produces a combination within a time-step. Since the elapsed time units within a loop is constant, its time complexity is $O(\binom{n}{m})$.

Algorithm 1

Initial state:

L1: Set $c_{in} = i$ in PE(i) for $1 \leqslant i \leqslant m - 1$ and $c_{in} = m - 1$ in PE(m).
L2: Set $R = n - m + i$, $T = 0$, $C = 0$ and $F = 0$ in PE(i) for all $1 \leqslant i \leqslant m$.
L3: Set $x_{in} = 0$ in PE(i) for $1 \leqslant i \leqslant m - 1$, and $d_{in} = 0$, $y_{in} = 0$ in PE(i) for $2 \leqslant i \leqslant m$.
L4: Set $x_{in} = 1$ in PE(m) and $d_{in} = 0$, $y_{in} = 0$ in PE(1) at all time-steps.

Executive state:

```
    begin
L5: repeat/* do simultaneously for all PEs. */
L6: if c_in < R then c_out:=c_in + x_in
          else
          begin
L7:         if C = 1 then c_out:=T else c_out:=R;
L8:       C:=C - 1
          end;
L9: if c_out = R then x_out:=1 else x_out:=0;
L10: d_out:=c_out;
L11: if F = 1 then
          begin
L12:      T:=d_in + 2;
L13:      y_out:=d_in + 3;
L14:      F:=0;
L15:      C:=C + 1
          end
        else
          begin
L16:      if y_in > 0 then
                begin
L17:              T:=y_in;
L18:              y_out:=y_in + 1;
L19:              C:=C + 1
                end
```

L20: **else** $y_{out}:=0$
end;
L21: **if** $x_{in}=1$ **and** $c_{out}=R-1$ **then** $F:=1$
until the $x_{out}=1$ of PE(1) is recognized by host computer
end.

Note that in Algorithm 1, after receiving the input data x_{in}, c_{in}, the m PEs generate simultaneously the m components of a combination. For $1 \leqslant i \leqslant m$ PE(i) produces the ith component. At the same time-step the combination comes out from the terminals o_i as shown in Fig. 1, and then we detect whether c_{out} reaches its limit value in order to determine the value of x_{out}, and so on. In what follows, the symbol "Li indicates that we are referring to the line number i of Alorithm 1. First we observe the following four facts:

(1) If $m \leqslant n-1$, L1, 3, 4, 6 imply that the first combination coming out is $\{1, 2, \ldots, m\}$. If $m=n$, L1, 2,4, 6, 7 imply that the first combination is also $\{1, 2, \ldots, m\}$. That is, at time-step $t=1$, the first combination comes out in lexicographic order.

(2) Suppose that the combination $A=\{a_1, a_2, \ldots, a_m\}$ comes out at time-step t_0, and there exists an integer α such that $F=1$ in PE(α) (this $F=1$ is set via L21), then at the following time-steps (from t_0+1 to $t_0+(m-\alpha)+1$) PE(α) propagates the $(m-\alpha)+1$ values (say $S_\alpha=\{a_{\alpha-1}+2, a_{\alpha-1}+3, \ldots, a_{\alpha-1}+(m-\alpha)+2\}$) to the $(m-\alpha)+1$ Ts of PE(i) for $\alpha \leqslant i \leqslant m$ respectively. This propagation works as follows.

(2a) At t_0+1: L12–15 imply that PE(α) receives $d_{in}=a_{\alpha-1}$, assigns $d_{in}+2=a_{\alpha-1}+2$ to its T, sends $d_{in}+3=a_{\alpha-1}+3$ to y_{out}, resets $F=0$, and increases C by one.

(2b) At t_0+2: L17–19 imply that PE($\alpha+1$) receives and assigns $y_{in}=a_{\alpha-1}+3$ to its T, sends $a_{\alpha-1}+4$ to y_{out}, and increases C by one.

(2c) In general at t_0+j: L17–19 imply that PE($\alpha+j-1$) receives and assigns $y_{in}=a_{\alpha-1}+j+1$ to its T, sends $a_{\alpha-1}+j+2$ to y_{out}, and increases C by one.

(2d) And so on up to PE(m) receives and assigns $y_{in}=a_{\alpha-1}+(m-\alpha)+2$ into its T, sends $a_{\alpha-1}+(m-\alpha)+3$ to y_{out}, and increases C by one at $t_0+(m-\alpha)+1$.

We define PE(α) to be the *leader* of a *propagating work* within the *propagating time interval* $I=[t_0+1, t_0+(m-\alpha)+1]$, and the $(m-\alpha)+1$ values in S_α are called the *propagating values* of PE(α) within time interval I.

(3) From L7 if $c_{in}=R_i$ then c_{out} is assigned a value from T or R according as the content of C is 1 or not.

(4) By L5 the Algorithm 1 repeats its execution until the $x_{out}=1$ in PE(1) is recognized by a host computer.

We assume that there exists a simple control circuit which can stop the linear array at the time-step such that $x_{out}=1$ of PE(1) is recognized.

3. THE PROOF OF CORRECTNESS

When the systolic array begins its operation, all PEs have $F=0$ and $C=0$ by L2. If $m \neq n$, Algorithm 1 increases the mth component by one at each time-step for generating a new combination. After $(n-m)$ time-steps, the combination $A=\{1, 2, 3, \ldots, m-1, n-1\}$ comes out. By L21 PE(m) sets $F=1$ because PE(m) has $x_{in}=1$ and $c_{out}=R_m-1=n-1$. It means that the assumption of the fact (2) in Section 2 is satisfied for $\alpha=m$ at time-step $t=n-m$. We will discuss the behaviors of propagating works of some PEs under the above assumption. That is, "*There exists an integer α such that the $(m-\alpha+1)$ PEs $(\alpha \leqslant i \leqslant m)$ set all $F=1$ and all the m PEs have $C=0$ at some time-step t_0*" is satisfied. For simplicity, we use the notation $\|\,\mathrm{PE}(i)$; $c_{out}=j$, $x_{in}=k, \ldots; t=t_0\|$ to denote the statement that PE(i) has $c_{out}=j$, $x_{in}=k$ and so on at the time-step $t=t_0$. And the symbol "$A \Rightarrow B$" means that statement A implies statement B.

Lemma 1

Suppose at time-step t_0, there exists an integer α such that PE(i) ($\alpha \leqslant i \leqslant m$) sets $F = 1$ and all $C = 0$, then for integer i such that $\alpha \leqslant i \leqslant m$ we have (a): $\|\,\mathrm{PE}(i);\ x_{\mathrm{in}} = 1,\ c_{\mathrm{out}} = R_i - 1,\ x_{\mathrm{out}} = 0;\ t = t_0\|$; (b): PE($i$) begins its propagating work at time-step $t_0 + 1$.

Proof. (a) By L21,9. (b) By L12–15.□

Lemma 2

Under the assumption of Lemma 1. If PE($\alpha - 1$) has $c_{\mathrm{out}} = \beta$ such that $\beta \leqslant R_{\alpha-1} - 2$. (When $\alpha = 1$, we let $\beta = 0$, $R_0 = 2$.) Then for integer j such that $1 \leqslant j \leqslant (m - \alpha + 1)$, PE($\alpha$) propagates $\beta + j + 1$ into the T of PE($\alpha + j - 1$) at $t_0 + j$. In other words, PE(α) performs its propagating work within the time interval $I = [t_0 + 1, t_0 + (m - \alpha) + 1]$, and PE($\alpha + j - 1$) receives and assigns the propagating value $\beta + j + 1$ of leader PE(α) to the T of PE($\alpha + j - 1$) at time-step $t_0 + j$.

Proof. By (b) of Lemma 1 PE(α) begins its propagating work at time-step $t_0 + 1$. This lemma is proved by the descriptions (2a)–(2d) in Section 2.□

Lemma 3

Under the assumption of Lemma 2. For integers k, j such that $1 \leqslant k \leqslant m - \alpha$ and $1 \leqslant j \leqslant m - \alpha - k + 1$, PE($\alpha + k$) propagates $R_{\alpha+k+j-1}$ to the T of PE($\alpha + k + j - 1$) at $t_0 + j$. That is, PE($\alpha + k + j - 1$) receives and assigns the propagating value $R_{\alpha+k+j-1}$ of leader PE($\alpha + k$) to the T of PE($\alpha + k + j - 1$) at time-step $t_0 + j$.

Proof. By (b) of Lemma 1 PE($\alpha + k$) begins its propagating work at time-step $t_0 + 1$. This lemma is also shown by (2)of Section 2.□

From Lemmas 2 and 3, there exist $(m - \alpha) + 1$ PEs (PE(i) for $\alpha \leqslant i \leqslant m$) such that they begin concurrently their propagating works at $t_0 + 1$, respectively. We call such PE(α) the *leftmost-leader* among these $(m - \alpha) + 1$ leaders PE(i), and notice that the propagating time interval with leader PE($i + 1$) is a subset of the propagating time interval with leader PE(i). The behaviors of Lemmas 2 and 3 are illustrated by the paths with arrows in Fig. 3, where the nodes are located in a x–y-plane coordinate system with x the variable of PE's index, and y the variable of time-step. Any path in Fig. 3 means that a propagating work of leader PE($\alpha + k$) for $0 \leqslant k \leqslant m - \alpha$. Note that during time interval $[t_0 + 1, t_0 + (m - \alpha) + 1]$ the last value being assigned to T of PE(i) ($\alpha \leqslant i \leqslant m$) is the propagating value with the leftmost-leader PE(α). Under the assumption of Lemma 2, since $\|\,\mathrm{PE}(i);\ c_{\mathrm{out}} = R_i - 1,\ x_{\mathrm{out}} = 0;\ t = t_0\|$ for all $\alpha \leqslant i \leqslant m$, by L6,9 we have

$$
\begin{aligned}
&[\mathrm{PE}(m);\quad c_{\mathrm{out}} = R_m - 1;\quad t = t_0]\\
\Rightarrow&[\mathrm{PE}(m);\quad c_{\mathrm{in}} = R_m - 1,\ x_{\mathrm{in}} = 1,\ c_{\mathrm{out}} = R_m,\ x_{\mathrm{out}} = 1;\quad t = t_0 + 1]\\
\Rightarrow&[\mathrm{PE}(m-1);\quad c_{\mathrm{in}} = R_{m-1} - 1,\ x_{\mathrm{in}} = 1,\ c_{\mathrm{out}} = R_{m-1},\ x_{\mathrm{out}} = 1;\quad t = t_0 + 2]\\
\Rightarrow&[\mathrm{PE}(m-2);\quad c_{\mathrm{in}} = R_{m-2} - 1,\ x_{\mathrm{in}} = 1,\ c_{\mathrm{out}} = R_{m-2},\ x_{\mathrm{out}} = 1;\quad t = t_0 + 3]\\
&\ \ \vdots\\
\Rightarrow&[\mathrm{PE}(\alpha);\quad c_{\mathrm{in}} = R_\alpha - 1,\ x_{\mathrm{in}} = 1,\ c_{\mathrm{out}} = R_\alpha,\ x_{\mathrm{out}} = 1;\quad t = t_0 + (m - \alpha) + 1].
\end{aligned}
$$

And during $I = [t_0 + 1, t_0 + (m - \alpha) + 1]$ by L8,15,19 the Cs of PE(α) and PE(m) are always 1, while the Cs of PE(k) for $\alpha + 1 \leqslant k \leqslant m - 1$ are greater than 1. Hence Lemmas 2 and 3 and L7 imply that PE(ρ) has $c_{\mathrm{out}} = R_\rho$ in the time interval $[t_0 + (m - \rho) + 1,\ t_0 + (m - \alpha) + 1]$ for $\alpha \leqslant \rho \leqslant m$. In fact, if we again refer to Fig. 3, where the four vertices A, B, C and D have coordinates $(\alpha, t_0 + 1)$, $(m, t_0 + 1)$, $(m, t_0 + (m - \alpha) + 1)$, $(\alpha, t_0 + (m - \alpha) + 1)$ respectively, and E is the intersection of line-segments $\overline{AC}$ and $\overline{BD}$. For any fixed PE(i) ($\alpha \leqslant i \leqslant m$) the contents of its register C are increased by 1 within or on the triangle $\triangle ABC$, and decreased by 1 within the triangle $\triangle BCD$ or on the segments $\overline{BC}$, $\overline{DC}$. Therefore the value of C in any fixed PE(i) is increased by 1 within and on $\triangle ABE$, kept the same value within $\triangle AED$, $\triangle BCE$ or on the segments $\overline{ED}$, $\overline{EC}$, and decreased by 1 within $\triangle CDE$ or on $\overline{CD}$. That is, we obtain the values of Cs of PE($\alpha + \rho$) for $0 \leqslant \rho \leqslant m - \alpha$ during the time interval $I = [t_0 + 1, t_0 + (m - \alpha) + 1]$ as follows.

If

$$0 \leqslant \rho \leqslant \left\lfloor \frac{m - \alpha}{2} \right\rfloor$$

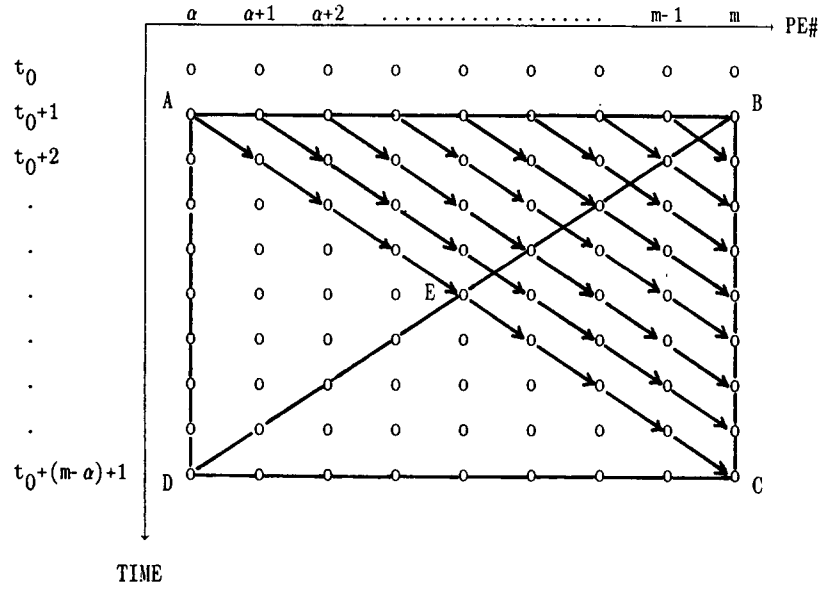

Fig. 3. The illustration of propagating work.

then (1) $C = j$ for $1 \leqslant j \leqslant \rho + 1$; (2) $C = \rho + 1$ for $\rho + 2 \leqslant j \leqslant m - \alpha - \rho + 1$; and (3) $C = m - \alpha - j + 2$ for $m - \alpha - \rho + 2 \leqslant j \leqslant m - \alpha + 1$.

If

$$\lfloor \frac{m-\alpha}{2} \rfloor + 1 \leqslant \rho \leqslant m - \alpha$$

then (1) $C = j$ for $1 \leqslant j \leqslant m - \alpha - \rho + 1$; (2) $C = m - \alpha - \rho + 1$ for $m - \alpha - \rho + 2 \leqslant j \leqslant \rho + 1$; and (3) $C = m - \alpha - j + 2$ for $\rho + 2 \leqslant j \leqslant m - \alpha + 1$.

Notice that all PE(i) have $C = 1$ at $t = t_0 + (m - \alpha) + 1$ for $\alpha \leqslant i \leqslant m$. We also note that all combinations during this time interval $I = [t_0 + 1,\ t_0 + (m - \alpha) + 1]$ come out in lexicographic order. Therefore, we have the following lemma.

Lemma 4

Under the assumption of Lemma 2, let $\{a_1, a_2, \ldots, a_{\alpha-2}, \beta, R_\alpha - 1, \ldots, R_m - 1\}$ be the combination coming out at t_0, then we have the following six results.

(a) For any integer j such that $1 \leqslant j \leqslant m - \alpha + 1$ we have $\|\,\text{PE}(\rho);\ c_{\text{out}} = R_\rho;\ t = t_0 + j\,\|$ for all $m - j + 1 \leqslant \rho \leqslant m$.

(b) If there exists an integer ρ such that $\|\,\text{PE}(\rho);\ c_{\text{out}} = R_\rho;\ t = t_1\,\|$ then we have $\|\,\text{PE}(i);\ c_{\text{out}} = R_i;\ t = t_1\,\|$ for all $\rho \leqslant i \leqslant m$.

(c) Within the propagating time interval $I = [t_0 + 1,\ t_0 + (m - \alpha) + 1]$ of the leftmost-leader PE(α) all the combinations come out in lexicographic order.

(d) $\|\,\text{PE}(i);\ C = 1,\ T = \beta + (i - \alpha) + 2;\ t = t_0 + (m - \alpha) + 1\,\|$ for all $\alpha \leqslant i \leqslant m$.

(e) The maximal value of C is

$$\lfloor \frac{m-\alpha}{2} \rfloor + 1$$

and it appears at PE(k) for

$$k = \alpha + \lfloor \frac{m-\alpha}{2} \rfloor.$$

(f) The combination $A = \{a_1, a_2, \ldots, a_{\alpha-2}, \beta + 1, \beta + 2, \ldots, \beta + (m - \alpha) + 2\}$ comes out and all $C = 0$ at time-step $t_0 + (m - \alpha) + 2$.

Proof. The parts of (a)–(e) are the results of the aforementioned discussions. So we only prove

the result of (f). By parts (a), (d) of this lemma with $j = (m-\alpha)+1$ and L7, for all $\alpha \leqslant i \leqslant m$ we have

$$\| \mathrm{PE}(i);\quad C=1,\ c_{\mathrm{out}}=R_i,\ T=\beta+(i-\alpha)+2;\quad t=t_0+(m-\alpha)+1 \|$$
$$\Rightarrow \| \mathrm{PE}(i);\quad c_{\mathrm{in}}=R_i,\ c_{\mathrm{out}}=\beta+(i-\alpha)+2,\ C=0;\quad t=t_0+(m-\alpha)+2 \|.$$

Since PE($\alpha-1$) has $c_{\mathrm{out}}=\beta$ and $C=0$ during $[t_0, t_0+(m-\alpha)+1]$, and

$$\| \mathrm{PE}(\alpha);\quad c_{\mathrm{out}}=R_\alpha,\ x_{\mathrm{out}}=1;\quad t=t_0+(m-\alpha)+1 \|$$
$$\Rightarrow \| \mathrm{PE}(\alpha-1);\quad c_{\mathrm{in}}=\beta,\ x_{\mathrm{in}}=1,\ c_{\mathrm{out}}=\beta+1,\ C=0;\quad t=t_0+(m-\alpha)+2 \|.$$

Therefore the combination A comes out and all $C=0$ at time-step $t_0+(m-\alpha)+2$.□

From (f) of Lemma 4, if $\beta < R_{\alpha-1}-2$, then any ith component of A is not the value R_i-1. This means that the assumption of Lemma 2 is not satisfied for any integer α such that $\alpha \leqslant m$. Algorithm 1 increases the mth component by one to generate new combination at the following time-step. But if $\beta = R_{\alpha-1}-2$, then for all $\alpha-1 \leqslant i \leqslant m$ we have

$$\| \mathrm{PE}(i);\quad c_{\mathrm{out}}=R_i-1,\ x_{\mathrm{in}}=1,\ F=1,\ C=0;\quad t=t_0+(m-\alpha)+2 \|.$$

This implies that the assumption of Lemma 2 holds for decreasing α by one. And these $(m-\alpha+2)$ leaders (PE(i) for a $\alpha-1 \leqslant i \leqslant m$) begin simultaneously their propagating works at time-step $t_0+(m-\alpha)+3$. Therefore we have the following lemmas.

Lemma 5

There exist exactly $(m-\alpha+2)$ PEs (PE(i) for $\alpha-1 \leqslant i \leqslant m$) such that PE($i$) sets $F=1$ and all $C=0$ at time-step $t_0+(m-\alpha)+2$ if and only if there exist exactly $(m-\alpha+1)$ PEs (PE(k) for $\alpha \leqslant k \leqslant m$) such that PE($k$) sets $F=1$, all $C=0$, and PE($\alpha-1$) has $c_{\mathrm{out}}=R_{\alpha-1}-2$ at time-step t_0.

Lemma 6

If there exist exactly $(m-\alpha+1)$ components of $A=\{a_1, a_2, \ldots, a_{\alpha-1}, R_\alpha, \ldots, R_m\}$ arriving at their limit values at $t_0+(m-\alpha)+1$ respectively, and $a_{\alpha-1} \leqslant R_{\alpha-1}-2$. Then at time-step t_0 the combination $D=\{a_1, a_2, \ldots, a_{\alpha-1}, R_\alpha-1, R_{\alpha+1}-1, \ldots, R_m-1\}$ comes out and PE(i) ($\alpha \leqslant i \leqslant m$) sets $F=1$ and all $C=0$.

Lemma 7

If there exists an integer α such that PE(α) sets $F=1$, so do all PE(i), $\alpha \leqslant i \leqslant m$.

Lemma 8

If there exist exactly $(m-\alpha+2)$ PEs (PE(k) for $\alpha-1 \leqslant k \leqslant m$) such that PE($k$) has $c_{\mathrm{out}}=R_k-1$ at $t_0+(m-\alpha)+2$, then PE(k) has $x_{\mathrm{in}}=1$, for all $\alpha-1 \leqslant k \leqslant m$.

Following the previous Lemmas, we should prove that Algorithm 1 in our linear systolic array is correct.

Theorem 1

Algorithm 1 for generating the combinations in lexicographic order is correct.

Proof. The proof is by induction on the index N of the combinations in lexicographic order. Note that N is also the time-step t of Algorithm 1.

(1) $N=1$. From the description of fact (1) in Section 2, we know that the first combination is $\{1, 2, \ldots, m\}$.

When $N=2$. (i) If $m=n$, then we have $\| \mathrm{PE}(1);\ c_{\mathrm{out}}=1,\ x_{\mathrm{out}}=1;\ t=1 \|$, hence $x_{\mathrm{out}}=1$ of PE(1) is recognized at $t=2$ and Algorithm 1 stops by L5. (ii) Suppose $m \neq n$ then all PEs have $c_{\mathrm{out}} \neq R_i$ and send $x_{\mathrm{out}}=0$ at $t=1$. By L6 for $1 \leqslant i \leqslant m-1$ we have $\| \mathrm{PE}(i);\ c_{\mathrm{in}}=i,\ x_{\mathrm{in}}=0,\ c_{\mathrm{out}}=i;\ t=2 \|$, and $\| \mathrm{PE}(m);\ c_{\mathrm{in}}=m,\ x_{\mathrm{in}}=1,\ c_{\mathrm{out}}=m+1;\ t=2 \|$. That is, the second combination $\{1, 2, \ldots, m-1, m+1\}$ is generated at $t=2$.

(2) Suppose that the theorem is true for all $N \leqslant k$, i.e. all combinations with indexes $N \leqslant k$ of

lexicographic order are generated correctly. Let $A = \{a_1, a_2, \ldots, a_m\}$ be the kth combination that comes out at time-step $t = k$.

(3) For $N = k + 1$. Let $B = \{b_1, b_2, \ldots, b_m\}$ be the next combination coming out after A. We want to show that B has index $k + 1$ under the lexicographic order. We classify the proof according to whether there exists a component of A arriving at limit value.

(a) If $a_m \neq n$, we must show that $b_m = a_m + 1$ and $b_i = a_i$ for all $1 \leqslant i \leqslant m - 1$. Since at $t + 1$ all PEs have $x_{in} = 0$ except that Pe(m) has $x_{in} = 1$. By L6 the combination $B = \{a_1, a_2, \ldots, a_{m-1}, a_m + 1\}$ comes out at $t = k + 1$. Hence $N = k + 1$ is true.

(b) If $a_m = n = R_m$. Then part (b) of Lemma 4 implies that there exists a minimal positive integer δ such that $A = \{a_1, a_2, \ldots, a_{\delta-1}, R_\delta, R_{\delta+1}, \ldots, R_m\}$, where $a_{\delta-1} < R_{\delta-1}$. Following the part (a) of Lemma 4 and Lemma 5 the combination A must be within a propagating time interval (say $I = [t_0 + 1, t_0 + (m - \alpha) + 1]$) of a leftmost-leader PE(α) for $\alpha \leqslant \delta$.

(b-i) If $a_{\delta-1} = R_{\delta-1} - 1$ i.e. $\alpha < \delta$, then A comes out within the time interval I. Hence the parts (a), (c) of Lemma 4 imply that $B = \{a_1, a_2, \ldots, a_{\delta-2}, R_{\delta-1}, R_\delta, \ldots, R_m\}$ comes out at $t = k + 1$ and B has index $k + 1$ in lexicographic order. This proves the theorem for $N = k + 1$.

(b-ii) If $a_{\delta-1} < R_{\delta-1} - 1$ i.e. $\alpha = \delta$, then by Lemma 5 the combination A comes out at $t = t_0 + (m - \alpha) + 1$. By Lemma 6, the combination $D = \{a_1, a_2, \ldots, a_{\alpha-1}, R_\alpha - 1, R_{\alpha+1} - 1, \ldots, R_m - 1\}$ was generated at the time-step $t_0 = t - (m - \alpha) - 1$. By induction hypothesis, D comes out in lexicographic order at t_0 because of $t_0 \leqslant k$. By L27 and Lemma 8 PE(i) sets $F = 1$ at t_0 for $\alpha \leqslant i \leqslant m$, and all $C = 0$, thus these $(m - \alpha) + 1$ PEs (PE(i) for $\alpha \leqslant i \leqslant m$) begin simultaneously their propagating works within the time interval $[t_0 + 1, t_0 + (m - i) + 1]$ respectively. By (f) of Lemma 4 the combination $B = \{b_1, b_2, \ldots, b_m\} = \{a_1, a_2, \ldots, a_{\alpha-2}, a_{\alpha-1} + 1, a_{\alpha-1} + 2, \ldots, a_{\alpha-1} + (m - \alpha) + 2\}$ comes out at time step $t_0 + (m - \alpha) + 2 = t + 1$ because of $t_0 = t - (m - \alpha) - 1$. Hence $N = k + 1$ is true.

By mathematical induction principle, the combinations coming out in lexicographic order is proved. The last combination $\{n - m + 1, n - m + 2, \ldots, n\}$ reaches terminals at time-step $\binom{n}{m}$. Hence PE(1) sends $x_{out} = 1$ at $t = \binom{n}{m}$ and thus $x_{out} = 1$ of PE(1) is recognized at the time-step $\binom{n}{m} + 1$. Therefore, Algorithm 1 stops at that right time-step.

This completes the proof of this theorem.□

4. EXAMPLES

Example 1

Table 1 is an example of $n = 5$, $m = 3$ for illustrating the results of operations in Algorithm 1. The values of $x_{in}, x_{out}, d_{in}, d_{out}, y_{in}, y_{out}$ and T, C, F, R of PEs are located at their corresponding positions of Fig. 2. The limit values of PEs are fixed by 3, 4, 5 in PE(1), PE(2), PE(3), respectively. The maximal value of C is 2 which appears in PE(2) at time-step $t = 9$.

Example 2

In Table 2 we give an example to illustrate the behaviors of Lemmas 2 and 3 with the contents of C, T, F in some PEs during the execution of Algorithm 1. Let $n = 15$, $m = 7$, $\alpha = 3$ and $\beta = 6$ be in Lemma 2 of Section 3. Suppose we have a combination (@, 6, 10, 11, 12, 13, 14) at time-step t_0, where @ indicates any number belonging to $\{1, 2, 3, 4, 5\}$. Then at time $t = t_0 - 1$ the combination is (@, 6, 9, 12, 13, 14, 15), hence we have $x_{in} = 1$ and $c_{out} = R_i - 1$ in PE(i) at the time t_0 for $3 \leqslant i \leqslant 7$. In Algorithm 1 these PEs start simultaneously their propagating works in order to assign values to their corresponding Ts. The values of T, C, F and the components of each related combinations are shown in Table 2, where * indicates any number in which we are not interested. The values of F, C are put in the first column. The values of T and the c_{out} are located in the second column. That is, F, C, and T have the positions as shown in Fig. 2 but the position of R is replaced by c_{out}. Note that the maximal value of C is 3, as appeared in PE(5) at time step $t = t_0 + 3$.

With the aid of two recursive procedures (Algorithms 2 and 3) we can modify the output stream of Algorithm 1 to produce all γ-subsets of the set $\{1, 2, \ldots, n\}$ for $1 \leqslant \gamma \leqslant m$. Let B be a

Table 1. An example with $n = 5$, $m = 3$

T	PE(1)	PE(2)	PE(3)	OUT
0	0 0 1 0 0 0 0 0 0 0 3 0	0 0 2 0 0 0 0 0 0 0 4 0	0 0 2 1 0 0 0 0 0 0 5 0	000
1	0 1 1 0 0 0 0 0 0 0 3 1	0 2 2 0 0 0 0 0 0 0 4 2	0 3 2 1 0 0 0 0 0 0 5 3	123
2	0 1 1 0 0 0 0 0 0 0 3 1	0 2 2 0 0 0 0 0 1 0 4 2	0 4 3 1 0 1 0 0 2 0 5 4	124
3	0 1 1 0 0 0 0 0 0 0 3 1	0 2 2 0 0 0 0 0 1 0 4 2	1 5 4 1 0 0 4 5 2 1 5 5	125
4	0 1 1 0 0 0 0 0 0 0 3 1	0 3 2 1 0 1 0 0 1 0 4 3	0 4 5 1 0 1 4 0 2 0 5 4	134
5	0 1 1 0 0 0 0 0 0 0 3 1	0 3 3 0 0 0 3 4 1 1 4 3	1 5 4 1 0 0 5 6 3 1 5 5	135
6	0 1 1 0 0 0 0 0 0 0 3 1	1 4 3 1 0 0 3 0 0 1 4 4	1 5 5 1 4 0 4 6 3 1 5 5	145
7	0 2 1 1 0 1 0 0 0 0 3 2	0 3 4 1 0 1 3 0 1 0 4 3	0 4 5 1 0 1 4 0 4 0 5 4	234
8	0 2 2 0 0 0 2 3 0 1 3 2	0 3 3 0 0 0 4 5 2 1 4 3	1 5 4 1 0 0 5 6 3 1 5 5	235
9	0 2 2 0 0 0 2 0 0 1 3 2	1 4 3 1 3 0 3 4 2 2 4 4	1 5 5 1 5 0 5 6 3 1 5 5	245
10	1 3 2 1 0 0 2 0 0 1 3 3	1 4 4 1 0 0 3 0 2 1 4 4	1 5 5 1 4 0 4 5 4 1 5 5	345

first-in-first-out buffer containing all the output of Algorithm 1, the modified algorithm can be designed by applying the following steps.

(1) Read a combination $A = \{a_1, a_2, \ldots, a_m\}$ from B until B is empty.

(2) For each combination $A = \{a_1, a_2, \ldots, a_m\}$ of B, suppose that $A' = \{a'_1, a'_2, \ldots, a'_m\}$ is the preceding combination of A (initial $A' = \{0, 0, \ldots, 0\}$), find the smallest index r such that $a_r = a'_r + 1$.

(3-1) If $r = m$ or $a_r \neq n - m + r$ then we produce the combinations $\{a_1, a_2, \ldots, a_i\}$ one by one for $r \leqslant i \leqslant m$ under the recursive procedure extension(r, m) as shown in Algorithm 2, where "parallel-output $a_1, a_2, \ldots, a_r$" means that PE(ρ) sends $c_{\text{out}} = a_\rho$ for $1 \leqslant \rho \leqslant r$ and PE(β) sends the blank signal to c_{out} for $r + 1 \leqslant \beta \leqslant m$ at a same time-step.

Table 2. An illustrative example with $n = 15$, $m = 7$, $\alpha = 3$, $\beta = 6$

T	PE(1)	PE(2)	PE(3)	PE(4)	PE(5)	PE(6)	PE(7)
$t_0 - 1$	*	0 * 0 * 0 11 0 12 0 13 0 14 0 6 0 9 1 12 1 13 1 14 1 15
t_0	*	0 * 1 * 1 11 1 12 1 13 1 14 0 6 0 10 0 11 0 12 0 13 0 14
$t_0 + 1$	*	0 * 0 8 0 12 0 13 0 14 0 15 0 6 1 10 1 11 1 12 1 13 1 15
$t_0 + 2$	*	0 * 0 8 0 9 0 13 0 14 0 15 0 6 1 10 2 11 2 12 2 14 1 15
$t_0 + 3$	*	0 * 0 8 0 9 0 10 0 14 0 15 0 6 1 10 2 11 3 13 2 14 1 15
$t_0 + 4$	*	0 * 0 8 0 9 0 10 0 11 0 15 0 6 1 10 2 12 2 13 2 14 1 15
$t_0 + 5$	*	0 * 0 8 0 9 0 10 0 11 0 12 0 6 1 11 1 12 1 13 1 14 1 15
$t_0 + 6$	*	0 * 0 8 0 9 0 10 0 11 0 12 0 7 0 8 0 9 0 10 0 11 0 12

Table 3. The γ-subsets of $\{1, 2, 3, 4, 5\}$ for $\leqslant \gamma \leqslant 3$

B	γ-subsets	Called subroutine
123	1, 12, 123.	extension(1, 3).
124	124.	extension(3, 3).
125	125.	extension(3, 3).
134	13, 134.	extension(2, 3).
135	135.	extension(3, 3).
145	14, 145, 15.	subset(2, 3, 5).
234	2, 23, 234.	extension(1, 3).
235	235.	extension(3, 3).
245	24, 245, 25.	subset(2, 3, 5).
345	3, 34, 345, 35, 4, 45, 5.	subset(1, 3, 5).

(3-2) Otherwise, i.e. if $r < m$ and $a_r = n - m + r$ then we call the procedure subset(r, m, n) as shown in Algorithm 3 which can be considered as to generate all ρ-subsets of the set $\{n - m + r, n - m + r + 1, \ldots, n - 1, n\}$ for $1 \leqslant \rho \leqslant m - r + 1$, and then add $\{a_1, a_2, \ldots, a_{r-1}\}$ to those ρ-subsets to get the γ-subsets of the set $\{1, 2, \ldots, n\}$ for $1 \leqslant \gamma \leqslant m$.

(4) Go to step (1).

Example 3

Following Example 1 the lexicographic enumeration of all γ-subsets ($1 \leqslant \gamma \leqslant 3$) corresponding to the combinations in B are shown in Table 3.

Algorithm 2

```
extension(r, m) ≡
  begin
  parallel-output a_1, a_2, ..., a_r;
  if r < m then extension(r + 1, m)
  end.
```

Algorithm 3

```
subset(r, m, n) ≡
  begin
  a_r:=n − m + r; /* assign n − m + r to the c_out of PE(r) */
  parallel-output a_1, a_2, ..., a_r;
  if a_r < n then begin
                  subset(r + 1, m, n);
                  subset(r, m − 1, n)
                  end
  end.
```

5. CONCLUSIONS

In this paper we present a parallel algorithm to generate all combinations of m items out of n given items in lexicographic order. The computational model is a linear systolic array processor. The algorithm is contrasted with that of [6–8], where they are either not in systolic array or they are not in lexicographic order. We also present two recursive procedures in order to modify the output of Algorithm 1 for generating all combinations of at most m items out of n given items in lexicographic order as shown in [10, 11]. Since all PEs are identical and execute the same program, it is suitable for VLSI implementation. If the number of PEs has a limitation, say only β PEs can be used, the technique in [12] for partitioning and mapping algorithms into the β PEs may be applied. Finally there are many other important combinatorial enumeration problems existed for which parallel algorithms are yet to be developed. For example, can we design a systolic algorithm to generate the $m!$ permutations? If it can be solved then we can generate all the permutations of m items out of n given items in a computational model of linear systolic array.

REFERENCES

1. H. T. Kung, Why systolic architectures? *IEEE Trans. Comput.* **15,** 37–46 (1982).
2. H. T. Kung, The structure of parallel algorithms. In *Advances in Computers*, (Ed. M. C. Yovits), pp. 65–112. Academic Press, New York (1980).
3. H. S. Stone, Parallel computers. In *Introduction to Computer Architectures*, pp. 363–425. Science Research Associates, Chicago, Ill. (1980).
4. D. S. Hirschberg, Fast parallel sorting algorithm. *Commun. ACM* **21,** 657–661 (1978).
5. V. Zakharov, Parallelism and array processing. *IEEE Trans. Comput.* **C33,** 45–78 (1984).
6. B. Chan and S. G. Akl, Generating combinations in parallel. *Bit* **26,** 2–6 (1986).
7. G. H. Chen and M. S. Chern, Parallel generating of permutations and combinations. *Bit* **26,** 277–283 (1986).
8. C. Y. Tag, M. W. Du and R. C. T. Lee, Parallel generation of combinations. *Proc. Int. Computer Symp.*, Taipei, Taiwan, pp. 1006–1010 (1984).
9. E. M. Reingold, J. Nievergelt and N. Deo, *Combinatorial Algorithms: Theory and Practice*. Prentice-Hall, Englewood Cliffs, N.J. (1977).
10. I. Semba, An efficient algorithm for generating all k-subsets ($1 \leqslant k \leqslant m \leqslant n$) of the set $\{1, 2, \ldots, n\}$ in lexicographic order. *J. Algorithm* **5,** 281–283 (1984).
11. I. Stojmenovic and M. Miyakawa, Applications of a subset-generating algorithm to base enumeration, knapsack and minimal covering problems. *Comput. J.* **31,** 65–70 (1988).
12. D. I. Moldovan and J. A. B. Fortes, Partitioning and mapping algorithms into fixed size systolic arrays. *IEEE Trans. Comput.* **C35,** 1–12 (1986).