BIT 29 (1989), 23–36

A SYSTOLIC GENERATION OF COMBINATIONS

CHAU-JY LIN and JONG-CHUANG TSAY

Department of Applied Mathematics and Institute of Computer Engineering, National Chiao Tung University, Hsinchu, Taiwan, Republic of China

Abstract.

A parallel algorithm for generating all combinations of m (m fixed) items out of any n given items in lexicographic order is presented. The computational model is a linear systolic array consisting of m identical processing elements. This algorithm requires $\binom{n}{m}$ time-steps for the $\binom{n}{m}$ combinations, that is, one output at each time-step. Since all processing elements perform the same program, it is suitable for VLSI implementation. Based on mathematical induction, such an algorithm is proved to be correct.

CR categories: C.1, F.2.

Keywords and phrases: parallel algorithm, systolic array, combinations, algorithm verification.

1. Introduction.

Under the demand of faster and more powerful computers, there have been many attempts to develop electronic devices and processors operating at high speed. However, it is difficult to increase the speed of circuit components, because the laws of physics impose limits on the computational speed, so use of a parallel computer is a way to achieve higher computing speeds. The growing importance of parallel computers and parallel algorithms is highlighted in [3, 6, 7, 9, 10]. Systolic arrays represents one of the parallel computation models, and many examples of systolic array processors have been presented, e.g. in the fields of image processing, matrix arithmetic, digital signal processing etc. [6, 11, 12]. However, only a few systolic arrays are designed for combinatorial enumeration problems.

Generating combinations is an important combinatorial enumeration problem. It has received much attention, and various applications have been found. For example, by generating the combinations of at most m out of n items, Sahni [13] has presented an approximate algorithm for the 0/1-knapsack problem which guarantees a relative error bound. Chen et al. [14] have shown that a number of geometry problems can be solved by generating the combinations of two

Received December 1987. Revised May and August 1988.

out of n items. Several parallel algorithms [2, 4, 5] have been designed to generate the $\binom{n}{m}$ combinations. However, these algorithms do not generate the combinations in lexicographic order, or they are not systolic algorithms. In this paper we present a parallel algorithm to generate all combinations of m (m fixed) out of any n given items in lexicographic order. The used computational model is a linear systolic array consisting of m identical processing elements (PEs). Each PE performs the same program, and hence it is suitable for VLSI implementation. The problem when m is not fixed is discussed in Section 6.

The remainder of this paper is organized as follows. In Section 2 an overview of sequential and parallel algorithms for generating combinations is given. In Section 3 we desdribe the computation model of the linear systolic array. The parallel algorithm and the verification of the systolic array are presented in Sections 4 and 5, respectively. Two modifications of our algorithm are considered in Section 6. Some concluding remarks are offered in Section 7.

2. Some existing algorithms for generating combinations.

It is well known that the combinations in lexicographic order can be generated sequentially in a straightforward way, see [1]. In [8], Semba presented a sequential algorithm to generate all combinations of at most m out of n items in lexicographic order. In [2], Chan and Akl presented a parallel algorithm to generate the combinations in a single instruction multiple data (SIMD) machine which allows data read simultaneously from a shared memory. It can be seen that the assignment statements of their algorithm depend on the indexed position of the executing PEs. In [4], Chen and Chern presented a parallel algorithm to generate the permutations of at most m out of n items, but not in lexicographic order. Their architecture for the algorithm consists of a linear array with k PEs, say PE(i) for $1 \leq i \leq k$, and a selector which receives a value z from PE(k), then sends a value y to PE(1) where $y = z - k$ if $z > k$; otherwise $y = n + z - k$. Each PE has a stack of size m to store the necessary data during the execution of their algorithm. This algorithm can easily be modified to generate combinations.

Contrasting with [1, 8], our algorithm is parallel. All these algorithms generate the combinations in lexicographic order. Contrasting with [2, 4], our algorithm is a systolic algorithm which can be run on a linear array consisting of m PEs. In addition, these m PEs are not necessary to recognize their indexed positions during the execution of our algorithm.

3. The computational model.

Without loss of generality, let $1, 2, 3, \ldots, n$ denote the n given items under consideration. Based on the algorithm in [1], a computational model is designed

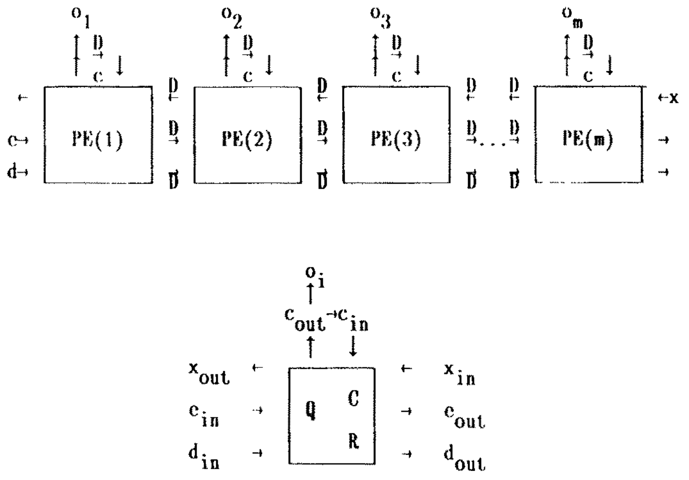

Fig. 1. The computational model and specification of PE(i).

consisting of m identical PEs to produce the $\binom{n}{m}$ combinations. Figure 1 indicates the layout of our computational model, where any individual PE is referred to as PE(i) for $1 \leq i \leq m$; c, e, d, x are four communication links; R is a register; C is a flag; Q is a queue and o_i is the output terminal. Each individual PE(i) is responsible for generating the ith component of any conbination. Each communication link has one delay (denoted by D).

For convenience, we consider the notations c_{in}, c_{out}, x_{in} etc. of Figure 1 as the names of variables within our algorithm. Each PE can perform the following tasks:

(1) receive input data from its input links,

(2) execute once the procedure that is defined by an algorithm,

(3) send output data to its output links.

We call the maximal time units to perform the above three tasks a *time-step* in our algorithm. Moreover, since each communication link has a delay, it means that if PE(i) sends its c_{out}, e_{out}, d_{out}, x_{out}, to c, e, d, x links respectively at time-step $t = t_0$, then such a c_{out} is the c_{in} of PE(i), such e_{out}, d_{out} are the e_{in}, d_{in} of PE(i + 1) respectively, and such an x_{out} is the x_{in} of PE(i − 1) at the time-step $t = t_0 + 1$. (We consider PE(0) and PE(m + 1) being in the memory of the host computer.)

From the definition of lexicographic order, if $A = \{a_1, a_2, \ldots, a_m\}$ is any combination of m out of n, then we know that $a_i \leq n-m+i$ for all $1 \leq i \leq m$. Then $n-m+i$ is called the *limit value* of the ith component of any combination, and we denote it by R_i. We refer again to our computational model, and the usage of these four communication links, register R, flag C, and queue Q in each individual PE(i) for $1 \leq i \leq m$ is described as follows.

(1) The c-link transmits c_{out}, i.e. the ith component of any combination, to output terminal o_i, and such a c_{out} is the c_{in} of PE(i) at the next time-step in order to produce the ith component of next combination.

(2) The x-link transmits a message to PE($i-1$) such that the message indicates whether the c_{out} of PE(i) is its limit value, i.e. if $c_{out} = n-m+i$, then $x_{out} = 1$; otherwise $x_{out} = 0$.

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

(4) The e-link transmits elements to the Q of PE($i+1$). If PE(i) has $e_{in} \neq 0$, then PE(i) sends $e_{out} = e_{in}+1$. Or if the flag $C = 1$ in PE(i) is true, then PE(i) sends $e_{out} = d_{in}+3$ to PE($i+1$).

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

(6) The flag C indicates that if the condition "$c_{out} = R_i - 1$ and $x_{in} = 1$" is true, then PE(i) sets $C = 1$; otherwise $C = 0$.

(7) Q stores the data fetching from e_{in} if $e_{in} \neq 0$; if $C = 1$ then Q stores the element $d_{in} + 2$.

4. Parallel algorithm for generating combinations.

From the description of Section 3 we derive the parallel algorithm *COMGEN*(n, m) as shown in Algorithm 1, where *FRONT*(Q) means that it returns the front element of Q and removes it from Q; *ADDQ*(q) means that it adds the element q to the rear of Q. During the execution of *COMGEN* (n, m). there is a combination coming out at each time-step, hence only (n, m) time-steps are required tot generate all the $\binom{n}{m}$ combinations.

Since a systolic array processor is always attached to a host computer by an interface system, the signal to stop the execution of *COMGEN* (n, m) can be sent by the host computer when the message $x_{out} = 1$ of PE(1) is recognized. An other way to stop the execution of PEs is to detect whether the value of c_{in} possesses a special symbol, e.g. as if $c_{in} = -1$ then the algorithm stops its execution. This consideration will be discussed in Section 6.

```
ALGORITHM 1. COMGEN(n, m):
[initialize.]
L1:
        If $m < n-1$, let $c_{in} = i, x_{in} = 0$ in PE(i) for $1 \le i \le m-1$, and $c_{in} = m-1$,
        $x_{in} = 1$ in PE(m).
L2:     If $m = n-1$ or $m = n$, let $c_{in} = i-1$, $x_{in} = 1$ in PE(i) for $1 \le i \le m$.
L3:     Set $R = n-m+i$, $C = 0$, and let Q be empty in PE(i) for $1 \le i \le m$.
L4:     Set $d_{in} = 0$, $e_{in} = 0$ in PE(i) for $2 \le i \le m$.
L5:     Set $d_{in} = 0$, $e_{in} = 0$ in PE(1) and $x_{in} = 1$ in PE(m) at all time-steps.
[executive body.]
  begin
L6:     repeat /* do parallel for all PEs */
L7:     if $c_{in} \le R-1$ then $c_{out} := c_{in}+x_{in}$ else $c_{out} := FRONT(Q)$;
L8:     if $c_{out} = R$ then $x_{out} := 1$ else $x_{out} := 0$;
L9:     $d_{out} := c_{out}$;
```

```
  if $C = 1$ then /* beginning of propagating work. */
  begin
L10:  $ADDQ(d_{in}+2)$;
L11:  $e_{out} := d_{in}+3$;
L12:  $C := 0$
  end
  else /* $C = 0$, check whether $PE(i)$ is within a propagating work. */
  begin
    if $e_{in} \neq 0$ then /* within a propagating work. */
    begin
L13:  $ADDQ(e_{in})$;
L14:  $e_{our} := e_{in} + 1$
      end
L15:  else $e_{out} := 0$; /* not in propagating work. */
    end
L16:  if $x_{in} = 1$ and $c_{out} = R - 1$ then $C: = 1$ /* prepare for propagating work. */
      until host computer sends stop signal
   end.
```

Note that in Algorithm 1, after receiving input data x_{in}, c_{in}, the m PEs generate simultaneously the m components of a combination, then it detects whether c_{out} reaches its limit value to determine the value of x_{out}, and so on. In what follows, the symbol "L i" indicates that we are referring to the line number i of $COMGEN(n,m)$. First we observe the following four facts in $COMGEN(n,m)$.

(1) If $m < n-1$, "L 1, 5, 7" implies that the first combination coming out is $\{1, 2, \ldots, m\}$. If $m = n-1$ or $m = n$, "L 2, 5, 7" implies 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 a time-step t_0, and there exists an integer α such that PE(α) has $C = 1$ (this $C = 1$ is set via "L 16" at $t = t_0 - 1$), then at the following $(m - \alpha) + 1$ 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 $(m-\alpha)+1$ Q's of PE(i) for $\alpha \leq i \leq m$ respectively. This propagation works as follows.

(2a) At $t = t_0+1$: "L 10–12" implies that PE(α) receives $d_{in} = a_{\alpha-1}$, assigns $d_{in}+2 = a_{\alpha-1}+2$ to its Q, sends $d_{in}+3 = a_{\alpha-1}+3$ to e_{out}, resets $C = 0$.

(2b) At $t = t_0+2$: "L 13, 14" implies that PE($\alpha+1$) receives and assigns $e_{in} = a_{\alpha-1}+3$ to its Q, sends $a_{\alpha-1}+4$ to e_{out}.

(2c) In general at $t = t_0+j$: "L 13, 14" implies that PE($\alpha+j-1$) receives and assigns $e_{in} = a_{\alpha-1}+j+1$ to its Q, sends $a_{\alpha-1}+j+2$ to e_{out}.

(2d) This goes on up to PE(m) which receives and assigns $e_{in} = a_{\alpha-m} + (m-\alpha) + 2$ into its Q, and sends $a_{\alpha-1} + (m-\alpha) + 3$ to e_{out} at $t = t_0 + (m-\alpha) + 1$.

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

(3) From "L 7" if $c_{in} = R_i$ in PE(i) the front element of Q in PE(i) is retrieved and assigned to the c_{out} of PE(i).

(4) "L 6" implies that $COMGEN(n,m)$ repeats its execution until $x_{out} = 1$ in PE(1) is recognized by the host computer, and then the host computer sends a signal to stop the execution of $COMGEN(n,m)$.

EXAMPLE 1: Table 1 is an example with $n = 5$, $m = 3$ for illustrating the results of execution of $COMGEN(n,m)$. The values of the variables have the corresponding locations as shown in Figure 1, where the arrows are omitted and the symbol ϕ denotes an empty set. Note that the maximal size of Q is 2 which appears in PE(2) at time-step 9.

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

T \ P	PE(1)	PE(2)	PE(3)	OUT
0	$\begin{matrix} & 1 & 0 \\ 0 & \boxed{\begin{matrix} \phi & 0 \\ & 3 \end{matrix}} & \\ 0 & & \end{matrix}$	$\begin{matrix} & 2 & 0 \\ 0 & \boxed{\begin{matrix} \phi & 0 \\ & 4 \end{matrix}} & \\ 0 & & \end{matrix}$	$\begin{matrix} & 2 & 1 \\ 0 & \boxed{\begin{matrix} \phi & 0 \\ & 5 \end{matrix}} & \\ 0 & & \end{matrix}$	
1	$\begin{matrix} 0 & 1\;1 & 0 \\ 0 & \boxed{\begin{matrix} \phi & 0 \\ & 3 \end{matrix}} & 0 \\ 0 & & 1 \end{matrix}$	$\begin{matrix} 0 & 2\;2 & 0 \\ 0 & \boxed{\begin{matrix} \phi & 0 \\ & 4 \end{matrix}} & 0 \\ 0 & & 2 \end{matrix}$	$\begin{matrix} 0 & 3\;2 & 1 \\ 0 & \boxed{\begin{matrix} \phi & 0 \\ & 5 \end{matrix}} & 0 \\ 0 & & 3 \end{matrix}$	123
2	$\begin{matrix} 0 & 1\;1 & 0 \\ 0 & \boxed{\begin{matrix} \phi & 0 \\ & 3 \end{matrix}} & 0 \\ 0 & & 1 \end{matrix}$	$\begin{matrix} 0 & 2\;2 & 0 \\ 0 & \boxed{\begin{matrix} \phi & 0 \\ & 4 \end{matrix}} & 0 \\ 1 & & 2 \end{matrix}$	$\begin{matrix} 0 & 4\;3 & 1 \\ 0 & \boxed{\begin{matrix} \phi & 1 \\ & 5 \end{matrix}} & 0 \\ 2 & & 4 \end{matrix}$	124
3	$\begin{matrix} 0 & 1\;1 & 0 \\ 0 & \boxed{\begin{matrix} \phi & 0 \\ & 3 \end{matrix}} & 0 \\ 0 & & 1 \end{matrix}$	$\begin{matrix} 0 & 2\;2 & 0 \\ 0 & \boxed{\begin{matrix} \phi & 0 \\ & 4 \end{matrix}} & 0 \\ 1 & & 2 \end{matrix}$	$\begin{matrix} 1 & 5\;4 & 1 \\ 0 & \boxed{\begin{matrix} 4 & 0 \\ & 5 \end{matrix}} & 5 \\ 2 & & 5 \end{matrix}$	125
4	$\begin{matrix} 0 & 1\;1 & 0 \\ 0 & \boxed{\begin{matrix} \phi & 0 \\ & 3 \end{matrix}} & 0 \\ 0 & & 1 \end{matrix}$	$\begin{matrix} 0 & 3\;2 & 1 \\ 0 & \boxed{\begin{matrix} \phi & 1 \\ & 4 \end{matrix}} & 0 \\ 1 & & 3 \end{matrix}$	$\begin{matrix} 0 & 4\;5 & 1 \\ 0 & \boxed{\begin{matrix} \phi & 1 \\ & 5 \end{matrix}} & 0 \\ 2 & & 4 \end{matrix}$	134
5	$\begin{matrix} 0 & 1\;1 & 0 \\ 0 & \boxed{\begin{matrix} \phi & 0 \\ & 3 \end{matrix}} & 0 \\ 0 & & 1 \end{matrix}$	$\begin{matrix} 0 & 3\;3 & 0 \\ 0 & \boxed{\begin{matrix} 3 & 0 \\ & 4 \end{matrix}} & 4 \\ 1 & & 3 \end{matrix}$	$\begin{matrix} 1 & 5\;4 & 1 \\ 0 & \boxed{\begin{matrix} 5 & 0 \\ & 5 \end{matrix}} & 6 \\ 3 & & 5 \end{matrix}$	135
6	$\begin{matrix} 0 & 1\;1 & 0 \\ 0 & \boxed{\begin{matrix} \phi & 0 \\ & 3 \end{matrix}} & 0 \\ 0 & & 1 \end{matrix}$	$\begin{matrix} 1 & 4\;3 & 1 \\ 0 & \boxed{\begin{matrix} 3 & 0 \\ & 4 \end{matrix}} & 0 \\ 0 & & 4 \end{matrix}$	$\begin{matrix} 1 & 5\;5 & 1 \\ 4 & \boxed{\begin{matrix} 4 & 0 \\ & 5 \end{matrix}} & 6 \\ 3 & & 5 \end{matrix}$	145
7	$\begin{matrix} 0 & 2\;1 & 1 \\ 0 & \boxed{\begin{matrix} \phi & 1 \\ & 3 \end{matrix}} & 0 \\ 0 & & 2 \end{matrix}$	$\begin{matrix} 0 & 3\;4 & 1 \\ 0 & \boxed{\begin{matrix} \phi & 1 \\ & 4 \end{matrix}} & 0 \\ 1 & & 3 \end{matrix}$	$\begin{matrix} 0 & 4\;5 & 1 \\ 0 & \boxed{\begin{matrix} \phi & 1 \\ & 5 \end{matrix}} & 0 \\ 4 & & 4 \end{matrix}$	234
8	$\begin{matrix} 0 & 2\;2 & 0 \\ 0 & \boxed{\begin{matrix} 2 & 0 \\ & 3 \end{matrix}} & 3 \\ 0 & & 2 \end{matrix}$	$\begin{matrix} 0 & 3\;3 & 0 \\ 0 & \boxed{\begin{matrix} 4 & 0 \\ & 4 \end{matrix}} & 5 \\ 2 & & 3 \end{matrix}$	$\begin{matrix} 1 & 5\;4 & 1 \\ 0 & \boxed{\begin{matrix} 5 & 0 \\ & 5 \end{matrix}} & 6 \\ 3 & & 5 \end{matrix}$	235
9	$\begin{matrix} 0 & 2\;2 & 0 \\ 0 & \boxed{\begin{matrix} 2 & 0 \\ & 3 \end{matrix}} & 0 \\ 0 & & 2 \end{matrix}$	$\begin{matrix} 1 & 4\;3 & 1 \\ 3 & \boxed{\begin{matrix} 4 & 0 \\ 3 & 4 \end{matrix}} & 4 \\ 2 & & 4 \end{matrix}$	$\begin{matrix} 1 & 5\;5 & 1 \\ 5 & \boxed{\begin{matrix} 5 & 0 \\ & 5 \end{matrix}} & 6 \\ 3 & & 5 \end{matrix}$	245
10	$\begin{matrix} 1 & 3\;2 & 1 \\ 0 & \boxed{\begin{matrix} 2 & 0 \\ & 3 \end{matrix}} & 0 \\ 0 & & 3 \end{matrix}$	$\begin{matrix} 1 & 4\;4 & 1 \\ 0 & \boxed{\begin{matrix} 3 & 0 \\ & 4 \end{matrix}} & 0 \\ 2 & & 4 \end{matrix}$	$\begin{matrix} 1 & 5\;5 & 1 \\ 4 & \boxed{\begin{matrix} 4 & 0 \\ & 5 \end{matrix}} & 5 \\ 4 & & 5 \end{matrix}$	345

From the assumption of the previous fact (2) we have a modified assumption as follows. "*There exist two integers α, β and a time-step t_0 such that the combination $\{a_1, a_2, \ldots, a_{\alpha-2}, \beta, R_\alpha - 1, R_{\alpha+1} - 1, \ldots, R_m - 1\}$ comes out at $t = t_0$, since β is the c_{out} of $PE(\alpha - 1)$ and $\beta \leq R_{\alpha-1} - 2$*". This assumption will be satisfied many times during the execution $COMGEN(n,m)$. (When $\alpha = 1$, let $\beta = 0$,

$R_0 = 2$.) Recall that R_i is the limit value of PE(i). Let $ACOM(n,m)$ be the set of the combinations satisfying this modified assumption. The elements in $ACOM(n,m)$ are indexed by their emerging order during the execution of $COMGEN(n,m)$.

EXAMPLE 2: We refer to Example 1 again. There are three combinations in its set ACOM(5,3), namely $\{1,2,4\}$, $\{1,3,4\}$, $\{2,3,4\}$ appearing at $t = 2,4,7$, respectively. The corresponding values of their indexes, time-steps, combinations, α, β, and R_α are listed in Table 2.

Table 2. *The related values of $ACOM(5,3)$.*

N	t_0	combination	α	β	$R_\alpha - 1$
1	2	{1,2,4}	3	2	4
2	4	{1,3,4}	2	1	3
3	7	{2,3,4}	1	0	2

5. The correctness proof.

At the beginning of the execution of $COMGEN(n,m)$, all PEs have empty queues and $C = 0$ by "L 3". If $m < n-1$, Algorithm 1 increases the mth component by one at each time-step in order to generate a new combination. After $n-m$ time-steps, the combination A $= \{1,2,\ldots,m-1,n-1\}$ comes out. "L 16" implies that PE(m) sets its $C = 1$ because PE(m) has $x_{in} = 1$ and $c_{out} = R_m - 1 = n-1$. It means that the assumption in Section 4 is satisfied for $\alpha = m$, $\beta = m-1$ at time-step $t_0 = n-m$. We will discuss the behaviors of propagating work of PE(i), $\alpha \le i \le m$ when this assumption is satisfied.

First we show that the algorithm $COMGEN(n,m)$ generates the $\binom{n}{m}$ combinations in lexicographic order. We give a sketch of proof. From Example 2 in Section 4, we know that there repeatedly exist two integers α, β and a time-step t_0 such that a combination of the form $\{a_1, a_2, \ldots, a_{\alpha-2}, R_\alpha - 1, R_{\alpha+1} - 1, \ldots, R_m - 1\}$ comes out at $t = t_0$, where β is the c_{out} of PE($\alpha - 1$) such that $\beta \le R_{\alpha-1} - 2$, all m queues are empty, and the C's of PE(γ). $\alpha \le \gamma \le m$, are set to 1 at $t = t_0$. Then after giving several Lemmas. Theorem 1 shows that if X, Y are any two consecutive combinations belonging to $ACOM(n,m)$, then all the combinations between X and Y are generated by $COMGEN(n,m)$ in lexicographical order. Using this result and mathematical induction we show in Theorem 2 that $COMGEN(n,m)$ is correct for generating all $\binom{n}{m}$ combinations in lexicographic order.

LEMMA 1: *Suppose that there exists an integer α and a time-step t_0 such that at $t = t_0$ all m queues are empty and the C's of PE(i) for $\alpha \leq i \leq m$ are set to 1. Then for $\alpha \leq i \leq m$ we have*

(*a*) *PE(i) has $c_{out} = R_i - 1$, and $x_{out} = 0$ at $t = t_0$.*
(*b*) *PE(i) begins its propagating work at time-step $t_0 + t$.*

PROOF. (*a*) By "L 16, 8". (*b*) By "L 10–12". ■

LEMMA 2: *Under the assumption of Lemma 1, suppose PE($\alpha-1$) has $c_{out} = \beta$ so that $\beta \leq R_{\alpha-1} - 2$ (if $\alpha = 1$ let $\beta = 0$, $R_0 = 2$). Then for integer j such that $1 \leq j \leq m-\alpha+1$, PE(α) propagates $\beta+j+1$ into the Q of PE($\alpha+j-1$) at time-step t_0+j, and PE(α) resets $C = 0$ at $t = t_0 + 1$.*

PROOF. By (*b*) of Lemma 1 PE(α) begins its propagating work at t_0+1. This lemma is proved by the descriptions (2a)-(2d) in Section 4. ■

LEMMA 3: *Under the assumption of Lemma 1, for integer k, j such that $1 \leq k \leq m - \alpha$ and $1 \leq j \leq m-\alpha-k+1$, PE($\alpha+k$) propagates $R_{\alpha+k+j-1}$ into the Q of PE($\alpha+k+j-1$) at time-step t_0+j, and PE($\alpha+k$) resets $C = 0$ at t_0+1.*

PROOF. By Lemma 1 and the fact (2) of Section 4. ■

From Lemmas 2, 3, there exist $(m-\alpha)+1$ PEs (PE(i) for $\alpha \leq i \leq m$) such that they begin concurrently their propagating work at t_0+1. 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 behavior of Lemmas 2, 3 is illustrated by the paths with arrows in Figure 2, where the nodes are located in a xy-plane coordinate system with x the variable of PE's index, and y the time-step. Any path in Figure 2 means a propagating work of the leader PE($\alpha+k$) for $0 \leq k \leq m-\alpha$. Note that during the time interval $[t_0+1,\ t_0+(m-\alpha)+1]$, $\alpha \leq i \leq m$, the last value to be inserted into the Q of PE(i) is the propagating value with the leftmost leader PE(α).

For simplicity, we write $\|\text{PE}(i);\ c_{in} = 2, x_{out} = 1, \ldots; t = t_0\|$ to denote the statement that PE(i) has $c_{in} = 2$, $x_{out} = 1$ and so on at time-step $t = t_0$. The symbol "S1 $\Rightarrow$ S2" means that statement S1 implies statement S2.

Under the assumption of Lemma 2, since $\|\text{PE}(i): c_{out} = R_i - 1, x_{out} = 0; t = t_0\|$ for all $\alpha \leq i \leq m$, by "L 7, 8" we have

$$
\begin{aligned}
&\|\text{PE}(m);\ c_{out} = R_m - 1;\ t = t_0\| \\
\Rightarrow &\|\text{PE}(m);\ c_{in} = R_m - 1;\ x_{in} = 1,\ c_{out} = R_m,\ x_{out} = 1;\ t = t_0 + 1\| \\
\Rightarrow &\|\text{PE}(m-1);\ c_{in} = R_{m-1} - 1,\ x_{in} = 1,\ c_{out} = R_{m-1},\ x_{out} = 1;\ t = t_0 + 2\| \\
\Rightarrow &\|\text{PE}(m-2); c_{in} = R_{m-2} - 1, x_{in} = 1, c_{out} = R_{m-2}, x_{out} = 1; t = t_0 + 3\| \\
&\cdots\cdots\cdots\cdots\cdots\cdots \\
\Rightarrow &\|\text{PE}(\alpha); c_{in} = R_\alpha - 1, x_{in} = 1, c_{out} = R_\alpha, x_{out} = 1; t = t_0 + (m-\alpha)+1\|.
\end{aligned}
$$

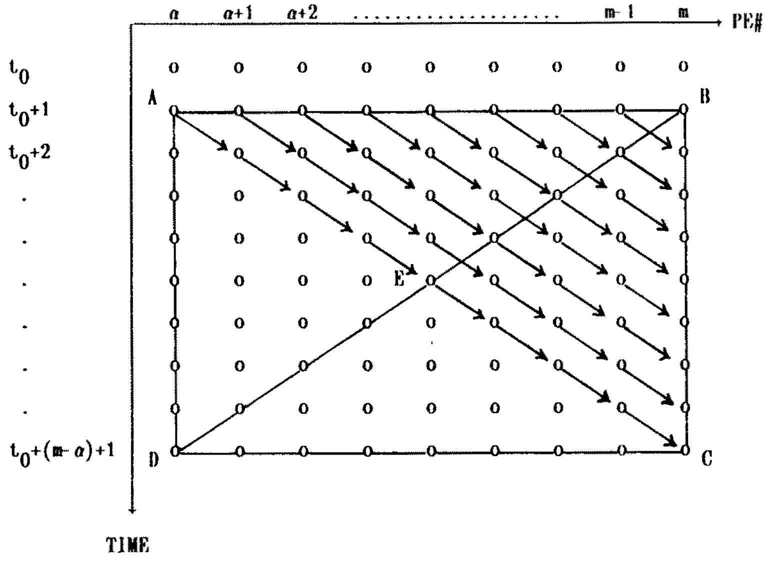

Fig. 2. The illustration of propagating work.

During $\mathrm{I} = [t_0 + 1, t_0 + (m - \alpha) + 1]$ "L 7, 10–14" implies that PE(ϱ) has $c_{out} = R_\varrho$ in the time interval $[t_0 + (m - \varrho) + 1,\ t_0 + (m - \alpha) + 1]$ for $\alpha \le \varrho \le m$. In fact, if again we refer to Figure 2, 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 $\overline{BD}$, then for any fixed PE(i), $\alpha \le i \le m$, the number of elements in its Q is increased by 1 within or on the triangle ABC, and decreased by 1 within the triangle BCD or on the segments $\overline{BC}$, $\overline{DC}$ excluding B. That is, the number of elements in the Q of a fixed PE(i) is increased by 1 within and on ABE, the same within AED, BCE or on the segments $\overline{ED}$, $\overline{EC}$ excluding E, and decreased by 1 within CDE or on $\overline{CD}$. We also notice that: (1) If the leader is not the leftmost leader, then its propagating value being inserted into the Q of a related PE(i) is the limit value R_i. (2) For a fixed i such that $\alpha \le i \le m$, the propagating values of the leader PE(i) are propagated to PE($i+k$), $0 \le k \le m-i$ at the time-step $t = t_0 + k + 1$ respectively, and these $m-i+1$ propagating values of PE(i) are used (retrieved from queues) simultaneously at time-step $t_0 + (m-i) + 2$. (3) The combinations coming out within the time interval $[t_0 + 1, t_0 + (m-\alpha) + 1]$ are 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 five results.

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

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

(*c*) *For all $\alpha \le i \le m$, $PE(i)$ has exactly one element $\beta + (i - \alpha) + 2$ in its Q at $t = t_0 + (m - \alpha) + 1$.*

(*d*) *The maximum value of Q is $\lfloor (m - \alpha)/2 \rfloor + 1$ and it appears at $PE(k)$ for $k = \alpha + \lfloor (m - \alpha)/2 \rfloor$.*

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

EXAMPLE 3: We illustrate the usage of the queue Q in each PE. Suppose that at time step t_0, we have a combination $\{*, 6, 10, 11, 12, 13, 14\}$ and all PE(γ), $3 \le \gamma \le 7$, have $c_{out} = R_\gamma - 1$, $C = 1$, $x_{out} = 0$ and the m queues are empty. In other words, the assumption of Lemma 2 is satisfied with $n = 15$, $m = 7$, $\alpha = 3$, and $\beta = 6$. Table 3 shows such a situation, where * indicates a symbol that we are not interested in. The contents of the queue are put into the first column of each PE. The flag C and the PE's c_{out} are located in the second column. The five paths in Table 3 denote the propagating work of the leaders PE(i) for $3 \le i \le 7$. Note that the combinations coming out during time-steps $[t_0, t_0 + 6]$ are in lexicographic order. The maximal size of Q is 3, appearing in PE(5) at

Table 3. *The behavior of propagating work with $n = 15$, $m = 7$, $\alpha = 3$, $\beta = 6$.*

P# / Time ↓	PE(1)	PE(2)	PE(3)	PE(4)	PE(5)	PE(6)	PE(7)
t_0	*	$\phi\ {}^{0}_{6}$	$\phi\ {}^{1}_{10}$	$\phi\ {}^{1}_{11}$	$\phi\ {}^{1}_{12}$	$\phi\ {}^{1}_{13}$	$\phi\ {}^{1}_{14}$
$t_0{+}1$	*	$\phi\ {}^{0}_{6}$	$8\ {}^{0}_{10}$	$12\ {}^{0}_{11}$	$13\ {}^{0}_{12}$	$14\ {}^{0}_{13}$	$15\ {}^{0}_{15}$
$t_0{+}2$	*	$\phi\ {}^{0}_{6}$	$8\ {}^{0}_{10}$	$\substack{12\\9}\ {}^{0}_{11}$	$\substack{13\\13}\ {}^{0}_{12}$	$\substack{14\\14}\ {}^{0}_{14}$	$15\ {}^{0}_{15}$
$t_0{+}3$	*	$\phi\ {}^{0}_{6}$	$8\ {}^{0}_{10}$	$\substack{12\\9}\ {}^{0}_{11}$	$\substack{13\\13\\10}\ {}^{0}_{13}$	$\substack{14\\14}\ {}^{0}_{14}$	$15\ {}^{0}_{15}$
$t_0{+}4$	*	$\phi\ {}^{0}_{6}$	$8\ {}^{0}_{10}$	$\substack{12\\9}\ {}^{0}_{12}$	$\substack{13\\10}\ {}^{0}_{13}$	$\substack{14\\11}\ {}^{0}_{14}$	$15\ {}^{0}_{15}$
$t_0{+}5$	*	$\phi\ {}^{0}_{6}$	$8\ {}^{0}_{11}$	$9\ {}^{0}_{12}$	$10\ {}^{0}_{13}$	$11\ {}^{0}_{14}$	$12\ {}^{0}_{15}$
$t_0{+}6$	*	$\phi\ {}^{0}_{7}$	$\phi\ {}^{0}_{8}$	$\phi\ {}^{0}_{9}$	$\phi\ {}^{0}_{10}$	$\phi\ {}^{0}_{11}$	$\phi\ {}^{0}_{12}$
$t_0{+}7$	*	$\phi\ {}^{0}_{7}$	$\phi\ {}^{0}_{8}$	$\phi\ {}^{0}_{9}$	$\phi\ {}^{0}_{10}$	$\phi\ {}^{0}_{11}$	$\phi\ {}^{0}_{13}$
$t_0{+}8$	*	$\phi\ {}^{0}_{7}$	$\phi\ {}^{0}_{8}$	$\phi\ {}^{0}_{9}$	$\phi\ {}^{0}_{10}$	$\phi\ {}^{0}_{11}$	$\phi\ {}^{1}_{14}$

$t = t_0 + 3$. Also notice that at time-step $t_0 + (n-1) - \beta = t_0 + 8$, the assumption of Lemma 2 is satisfied again with $\alpha = 7$ and $\beta = 11$.

From the result (*e*) of Lemma 4, there are two cases to be considered according to the value of β.

(1) If $\beta < R_{\alpha-1} - 2$, then we have $\|\mathrm{PE}(\gamma);\ c_{out} = \beta + \gamma - \alpha + 2 < R_\gamma - 1;\ t = t_0 + (m-\alpha) + 2\|$ for $\alpha - 1 \leq \gamma \leq m$, so that the assumption of Lemma 2 is not satisfied for any integer α such that $\alpha \leq m$ at this time-step. This implies that for all $1 \leq i \leq m-1$ we have

$$\|\mathrm{PE}(m); c_{in} < R_m, x_{in} = 1, c_{out} = c_{in}+1; t = t_0+(m-\alpha)+3\| \quad \text{and}$$

$$\|\mathrm{PE}(i);\ x_{in} = 0, c_{out} = c_{in};\ t = t_0 + (m-\alpha) + 3\|.$$

That is, the c_{out} of PE(i) preserves its previous value, i.e. $c_{out} = c_{in}$, and the c_{out} of PE(m) increases c_{in} by one. Continue this process up to the case that we have $\|\mathrm{PE}(m); c_{out} = R_m - 1 = n-1; t = t_0+(n-1)-\beta\|$. At this moment, the assumption of Lemma 2 is again satisfied for the old values of t_0, α, β being replaced by the new values $t_0+n-1-\beta$, m, and $\beta+(m-\alpha)+1$ respectively.

(2) If $\beta = R_{\alpha-1}-2$, then for $\alpha-1 \leq i \leq m$ we have $\|\mathrm{PE}(i); c_{out} = \beta+i-\alpha+2 = R_i - 1; t = t_0 + (m-\alpha) + 2\|$. Note that for $\alpha \leq \gamma \leq m$ we have $\|\mathrm{PE}(\gamma); c_{out} = R_\gamma;\ x_{out} = 1;\ t = t_0 + (m-\alpha) + 1\|$. Hence for $\alpha - 1 \leq i \leq m$ we have $\|\mathrm{PE}(i); c_{out} = R_i - 1, x_{in} = 1,\ C = 1;\ t = t_0 + (m-\alpha) + 2\|$, implying that the assumption of Lemma 2 is satisfied when the old values of t_0, α, β are replaced by the new values $t_0 + (m-\alpha) + 2$, $\alpha - 1$, and $a_{\alpha-2}$ respectively.

From the above discussion and the result (*b*) of Lemma 4, we have the following theorem.

THEOREM 1: *Under the assumption of Lemma* 2, *we have*

(1): *If $\beta < R_{\alpha-1}-2$, then all queues are empty and PE(m) sets its $C = 1$ at time-step $t = t_0+n-1-\beta$.*

(2): *If $\beta = R_{\alpha-1}-2$, then all PEs have empty queue and PE(i), $\alpha-1 \leq i \leq m$, sets its $C = 1$ at $t = t_0+(m-\alpha)+2$.*

(3): *All combinations coming out between $[t_0, t_0+n-1-\beta]$ or $[t_0, t_0+(m-\alpha)+2]$ are in lexicographic order.*

Following the previous Lemmas and Theorem 1, we verify that *COMGEN*(n,m) generates $\binom{n}{m}$ combinations in lexicographic order.

THEOREM 2: *The algorithm COMGEN(n,m) generating combinations in lexicographic order is correct.*

PROOF. Let N be an ordinal number and A_N the corresponding combination of the set $ACOM(n,m)$. We shall prove that, for any N not larger than the maximal ordinal number of $ACOM(n,m)$, all combinations generated up to A_N are in lexicographic order. This can be done by mathematical induction on N.

(1): For $N = 1$.

(1a): If $m < n-1$, by "L 1", the initial values of c_{in} are $\{1,2,\ldots,m-2,m-1,m-1\}$ and $x_{in} = 0$ in PE(i) for all $1 \leq i \leq m-1$ and $x_{in} = 1$ in PE(m). Therefore, "L 7" implies that the first generated combination is $\{1,2,\ldots,m\}$. "L 8" implies that $x_{out} = 0$ for all PEs. At the next time-step, since PE(i), $1 \leq i \leq m-1$, has $x_{in} = 0$ and PE(m) has $x_{in} = 1$, "L 7" implies that c_{out} of PE(m) is incremented by one. If c_{out} of PE(m) equals $R_m - 1$, then the assumption of Lemma 2 is satisfied with $\alpha = m$, $\beta = m-1$, $t_0 = 2$ because of $m = n-2$. Otherwise, since $x_{in} = 1$ in PE(m) for all time-steps, its c_{out} is incremented up to $R_m - 1$ at $t = n-m$, so that the assumption of Lemma 2 is also satisfied with $\alpha = m$, $\beta = m-1$ and $t_0 = n-m$. This shows that $N = 1$ is true for $m < n-1$.

(1b): If $m = n-1$, by "L 2, 7", the first combination is also $\{1,2,\ldots,m\}$ and the assumption of Lemma 2 is satisfied with $\alpha = 1$, $\beta = 0$, $t_0 = 1$. Hence $N = 1$ is true for $m = n-1$.

(1c): If $m = n$, "L 2, 7" implies that the first combination is also $\{1,2,\ldots,m\}$ and because PE(1) has arrived at its limit value $R_1 = 1$, "L 8" implies that PE(1) sends $x_{out} = 1$ to the host computer. Hence $COMGEN(n,m)$ stops its execution and the theorem is proved.

(2): Suppose the theorem true for all $N \leq k$. Let $A = \{a_1, a_2, \ldots, a_m\}$ be the combination with the ordinal number k which satisfies the assumption of Lemma 2 and A comes out at time-step t_0. Then all combinations are generated correctly in lexicographic order at all time-steps t such that $t \leq t_0$.

(3): For $N = k+1$, from the result (b) or (a) of Theorem 1 the assumption of Lemma 2 is again satisfied at either time-step $t_1 = t_0 + (m-\alpha)+2$ or $t_1 = t_0 + (n-1) - \beta$. In either case all combinations generated during time interval $[t_0, t_1]$ are in lexicographic order by (c) of Theorem 1. Further, at time-step t_1, the assumption of Lemma 2 is satisfied again. This shows the truth of the theorem for $N = k+1$.

By mathematical induction and Lemma 4, the execution of $COMGEN(n,m)$ reaches a state where all PEs have $c_{out} = R_i$ and PE(1) sends $x_{out} = 1$. In fact, the last combination $\{R_1, R_2, \ldots, R_m\} = \{n-m+1, n-m+2, \ldots, n\}$ comes out at time step $\binom{n}{m}$, and $x_{out} = 1$ in PE(1) is recognized by the host computer at the next time-step, so the algorithm $COMGEN(n,m)$ stops its execution at the right time-step. ■

6. Discussion.

In this section, we discuss two modifications of $COMGEN(n,m)$ in order to satisfy different requirements. First if m is not a fixed number, from the results of Lemmas 2, 3, we know that within a propagating time interval $[t_0+1, t_0+(m-\alpha)+1]$, for an integer i such that $\alpha \leq i \leq m$ the last element being inserted into the Q of PE(i) is the propagating value of the leftmost leader PE(α). These $(m-\alpha)+1$ propagating values of PE(α) are used simultaneously at $t = t_0 + (m-\alpha) + 2$. The other propagating values inserted into the Q of PE(i) are always the limit value R_i. This observation implies that we can only store the propagating values of the leftmost header PE(α) and disregard the propagating values of the leader PE(k) for $\alpha+1 \leq k \leq m$, but we need a counter to indicate the number of elements reserved for the Q of PE(i). Therefore, the queue Q in each individual PE can be replaced by two registers, one a temporary storage T, the other a counter K. Two operations of queue ($ADDQ(q)$ and $c_{out} := FRONT(Q)$) will be replaced by some assignment statements under the initial values of T, K being zero. There are three lines of $COMGEN(n,m)$ to be replaced (denoted by $\cong$), namely:

```
(1) "L7"  ≅ if $c_{in} \leq R-1$ then $c_{out} := c_{in}+x_{in}$ else
              begin if $K = 1$ then $c_{out} := T$ else $c_{out} := R$; $K := K-1$ end.
(2) "L10" ≅ $T := d_{in}+2$; $K := K+1$.
(3) "L13" ≅ $T := e_{in}$; $K := K+1$.
```

The second modification is that we can use the value of c_{in} to stop the execution of $COMGEN(n,m)$. When the last combination of ACOM(n,m), i.e. $\{R_1-1,\ldots,R_m-1\}$, comes out at $t = \binom{n}{m} - m + 1$. PE(1) is the leftmost leader of propagating work at this time-step. (PE(1) has one and only one time to become a leader of propagating work.) Its propagating values $\{2,3,4,\ldots,m+1\}$ are not used because PE(1) has $c_{out} = R_1$ at $t = \binom{n}{m}$ and then $COMGEN(n,m)$ stops its execution. Therefore, instead of the set $\{2,3,\ldots,m+1\}$ PE(1) sends a set of signals, say $\{-1,-1,\ldots,-1\}$, to all m PEs, and then the condition $c_{in} = -1$ will stop the execution of $COMGEN(n,m)$. For this purpose, there are some lines of $COMGEN(n,m)$ to be replaced, namely:

```
(1) "L6" ≅ while $c_{in} \geq 0$ do /* parallel for all PEs. */
(2) "L10, 11" ≅ if $d_{in} > 0$ then begin $ADDQ(d_{in}+2)$; $e_{out} := d_{in}+3$ end
(3) "L13, 14" ≅ if $e_{in} > 0$ then begin $ADDQ(e_{in})$; $e_{out} := e_{in}+1$ end
                else begin $ADDQ(-1)$; $e_{out} := -1$ end.
```

This modified algorithm will work as the $COMGEN(n,m)$ except that it produces the set $\{-1,-1,\ldots,-1\}$ at $t = \binom{n}{m} + 1$, and then its execution stops at the next time-step.

7. Conclusion.

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 consisting of m PEs. The algorithm is to be contrasted to [2, 4, 5], where they are not systolic algorithms or they do not generate the combinations in lexicographic order. Since the systolic array consists of identical PEs, if m is not too large or the queue is replaced by two registers, then it is suitable for VLSI implementation. Under some modifications of $COMGEN(n, m)$ a new parallel algorithm can be designed to generate all combinations of at most m items out of n given items in lexicographic order, as the sequential algorithm shown in [8]. Finally, there exist many other important combinatorial enumeration problems for which efficient parallel algorithms are yet to be developed. A representative one is to generate the permutations of m out of n items. If the $m!$ permutations can be generated in a linear systolic array, then this can be used in our combinations generating algorithm to produce all the permutations of m out of n given items.

Acknowledgement.

The authors wish to thank an anonymous referee for a number of instructive comments.

REFERENCES

1. E. M. Reingold, J. Nievergelt, and N. Deo, *Combinatorial Algorithms: Theory and Practice*, Prentice-Hall INC. Englewood Cliffs, New Jersey, 1977.
2. B. Chan and S. G. Akl, *Generating combinations in parallel*, BIT 26 (1986), pp. 2–6.
3. S. G. Akl, *Parallel Sorting Algorithms*, Academic Press, Orlando, Florida, 1985.
4. G. H. Chen and M. S. Chern, *Parallel generating of permutations and combinations*, BIT 26 (1986), pp. 277–283.
5. C. Y. Tag, M. W. Du, and R. C. T. Lee, *Parallel generation of combinations*, in Proc. Int'l. Comput. Symp., Taipei, Taiwan, 1984, pp. 1006–1010.
6. H. T. Kung, *The structure of parallel algorithms*, in *Advances in Computers*, M. C. Yovits, Ed. Academic Press, New York, 1980, pp. 65–112.
7. H. S. Stone, *Parallel computers*, in *Introduction to Computer Architectures*, 1980.
8. I. Semba, *An efficient algorithm for generating all k-subsets* ($1 \leq k \leq m \leq n$) *of the set* $\{1, 2, \ldots, n\}$ *in lexicographic order*, Journal of Algorithms 5, 1984, pp. 281–283.
9. R. Sedgewick, *Permutation generation methods*, Computing Surveys, Vol. 9, No. 2. 1977, pp. 137–164.
10. V. Zakharov, *Parallelism and array processing* IEEE Trans. on Computers, Vol. C-33, No. 1, 1984, pp. 45–78.
11. D. I. Moldovan, *On the design of algorithms for VLSI systolic arrays*, Proc. IEEE, Vol. 71, No. 1, 1983, pp. 113–120.
12. D. I. Moldovan and J. A. B. Fortes, *Partitioning and mapping algorithms into fixed size systolic arrays*, IEEE Trans. on Computers, Vol. C-35, No. 12.
13. S. Sahni, *Approximate algorithms for the 0/1 knapsack problem*, J. ACM, Vol. 22, No. 1, 1975, pp. 115–124.
14. G. H. Chen, M. S. Chern, and R. C. T. Lee, *A new systolic architecture for convex hull and half-plane intersection problems*, BIT 27, 1987, pp. 141–147.