論文 90-27-1-4

Context-Free 언어의 인식을 위한 일차원 시스토릭 어레이의 설계

(Design of One-Dimensional Systolic Array for Recognition of Context-Free Language)

禹 鍾 鎬*, 安 光 善**

(Chong Ho Woo and Gwang Sun Ahn)

要 約

Context-free 언어는 Cocke-Younger-Kasami 알고리듬에 의해서 인식될 수 있다. 이 알고리듬은 polyadic-nonserial 동적 프로그래밍 기법에 속하고 $O(n^3)$의 계산 복잡도를 갖는다.

본 논문에서는 이 알고리듬의 처리를 위한 일차원 시스토릭 어레이를 설계한다. 설계방법은 기존의 설계된 삼각형의 이차원 어레이를 일차원 어레이로 투영(projection)하여 변형한다. 설계한 선형 어레이는 문제의 크기가 n인 경우 n개의 처리요소로 구성되고, $\lfloor (n+1)/2 \rfloor (n-1)+3n-1$의 처리시간이 요구된다. 즉 $O(n^2)$의 계산 복잡도를 갖는다.

Abstract

Context-free language can be recognized by Cocke-Younger-Kasami algorithm. This algorithm is a class of polyadic-nonserial dynamic programming technique and has the $O(n^3)$ time complexity.

In this paper, a one-dimensional systolic array for recognition of context-free language is designed. The designed triangle type two-dimensional array is projected and transformed to an one-dimensional array. The designed one-dimensional array has n processing elements and $\lfloor (n+1)/2 \rfloor (n-1)+3n-1$ time units to process the algorithm (n is the length of input string). The time complexity is $O(n^2)$.

I. 서 론

Context-free 언어는 흔히 Cocke-Younger-Kasami 알고리듬에 의해서 인식된다. 이 알고리듬은 polyadic-nonserial 동적 프로그래밍 처리기법에 속하고 $O(n^3)$의 계산 복잡도를 갖는다.[1,2]

반도체기술의 발전과 더불어 알고리듬을 직접 VLSI에서 처리하여 계산시간을 크게 향상시키는 많은 연구가 있어왔다.[3] 특히 알고리듬과 관련된 처리식이 순환적(recursive)이고, 입출력에 비교해서 계산량이 절대적으로 많은(compute-bound) 알고리듬에 시스토릭 어레이 기법이 이용된다. 이 기법은 많은 수의 처리요소를 어레이로 구성하여 파이프라인과 병렬처리 기법을 이용하여 'Von Neumann의 병목현상'

*正會員, 釜山水産大學 電子工學科
(Dept. of Elec. Eng., Nat'l Fisheries Univ. of Pusan)

**正會員, 慶北大學校 電子計算機工學科
(Dept. of Computer Eng., Kyungbook Univ.)

接受日字 : 1989年 9月 18日

문제를 해결한다.[4] 알고리듬을 시스토릭 어레이로 변형하는 많은 체계적인 방법과 이것들을 이용한 설계를 자동으로 이행하는 소프트웨어 페케지들이 제안되었다.[5,6,7] 이차원 시스토릭 어레이는 문제의 크기가 증가함에따라 입출력 채널의 수가 증가하여 VLSI로 구현할 때 입출력 핀수가 증가된다. 따라서 문제의 크기에 관계없이 입출력 채널의 수를 고정시키고, 폴트 톨러런스에 적합한 일차원 선형 어레이에 관한 연구가 진행되어 왔다.[8,9,10]

polyadic-nonserial 동적 프로그래밍 처리를 위한 삼각형의 이차원 시스토릭 어레이들이 설계 제안되었고, 이 어레이들은 처리요소의 수는 $O(n^2)$, 처리시간은 $O(n)$이 요구된다.[11,12,13,15] Chu[14] 등은 context-free 언어의 인식을 위한 이차원 시스토릭 구조를 제안하였다.

본 논문에서는 context-free 언어 인식을 위한 일차원 시스토릭 어레이를 설계하였다. Chu등이 제안한 이차원 어레이를 일차원 어레이로 변환하여 공간 및 시간 스케줄링을 하였다. 설계된 어레이는 문제의 크기 n에 대해서 n개의 처리요소가 요구되고, 처리시간은 $\lfloor (n+1)/2 \rfloor (n-1)+3n-1$ 단위시간이 소요된다. 본 논문에서 설계한 일차원 어레이는 Varman 등[8]이 제안한 동적 프로그래밍 처리를 위한 일차원 어레이보다 처리속도가 빠르고, 처리요소에 요구되는 $O(n)$의 기억 공간을 요구하지 않는다. 또한 지연을 위해서 필요한 쉬프트 레지스터의 수를 줄였고, 어레이는 일방향(one-way)선형이므로 폴트 톨러런스가 용이하다.[10]

본 연구의 결과는 빠른 응답을 요구하는 음성인식, 패턴인식등의 실시간 처리에 이용될 수 있다.[16]

Ⅱ. Context-free 언어의 인식 알고리듬

Context-free 언어(CFL)의 인식에는 CKY알고리듬과 Early알고리듬이 널리 이용된다.[2,14] CKY 알고리듬은 Cocke, Kasami 및 Younger에 의해서 독자적으로 제안된 것으로, 문법이 Chomsky normal 형태를 갖는 것을 요구한다. 그러나 Early알고리듬은 문법의 형태에는 제한을 두지 않는다. 본 논문에서는 CKY 알고리듬을 이용한다.

〔정의〕 Context-free 문법은 다음 네가지 요소로 구성된다.

$G=(N, T, P, S)$

여기서 N은 nonterminal 기호의 유한집합, T는 terminal 기호의 유한집합, P는 production의 유한집합, S는 nonterminal 기호로 시작 기호를 나타낸다. 또한 $A \in N$, $\beta \in NUT$일 경우 각 production은 $A \rightarrow \beta$의 형태를 갖는다.

CKY알고리듬은 다음과 같다.

```
Cocke-Kasami-Younger Algorithm for recognition of CFL
  Input: String w of L(G) generated by context-free grammar G.
    w = a1, a2 ··· an, n≥1, and for 1≤k≤n ak∈T
  Output: S at the V1,n of recognition matrix V
  begin
    for i=1 to n do
      Vi,i-1 ← {A: A←ai∈P, ai is the i th symbol of w}
    for d=2 to n do
      for i=1 to n-d+1 do
        begin
          j ← d+i
          Vi,j ← φ
          for k=i+1 to j-1 do
            Vi,j ← Vi,j ∪ {A: A→BC∈P, B∈Vi,k, C∈Vk,j}
        end
  end
```

위의 알고리듬의 처리결과, 인식행렬 V의 $V_{1,n}$에 시작기호 S가 존재하면 입력스트링 $w \in L(G)$로서 수락(acception)되고, S가 존재하지 않으면 거부 (rejection)된다.

이 알고리듬은 polyadic-nonserial 동적 프로그래밍 기법에 근거를 두고 있고 $O(n^3)$의 시간 한계를 갖는다.

Ⅲ. 시스토릭 어레이의 설계

CFL의 인식을 위한 시스토릭 어레이의 구성은 polyadic-nonserial 동적 프로그래밍 처리를 위한 시스토릭 어레이[11,14]와 기본 구성이 같다. 설계에 앞서 처리의 효율과 하드웨어의 간략화를 위하여 문법의 production을 코우드화 한다.

1. 효율적인처리를 위한 Production의 코우드화

문법 G에서 사용한 시작기호와 nonterminal기호에 대해서 $A_i, 1 \le i \le m$로 둔다. 즉 $N=\{A_1, A_2 \cdots A_m\}$, 여기서 $m=|N|$이다. terminal기호에 대해서도 같은 방법을 적용한다. $T=\{a_1, a_2 \cdots a_t\}$, 여기서 $t=|T|$이다.

모든 A_i의 각 쌍에 대해서 문법의 production을 적용하여 m비트의 코우드를 정한다. 즉 $A_i \rightarrow A_j A_k \in P$이면, $A_j A_k$의 코우드는 m개의 비트에서 i번째 비트

가 '1'이 된다. 이렇게 구해진 모든 코우드를 production 코우드표라 하고, 이 표는 어레이의 각 처리요소에 기억된다.

Terminal 기호와 관련된 production도 같은 방법을 적용하여 코우드화 시킨다. 즉 $A_i \rightarrow a_j \in P$ 이면, a_j의 코우드에서 i번째 비트가 '1'이 된다. 이것은 어레이의 입력에서만 사용되고, 호스트 컴퓨터에서 처리되어 어레이로 전달될 수 있고, 직접 어레이의 입력단에서 하드웨어로 처리될 수도 있다.

(예) Context-free 문법 G=(N, T, P, S)에서
N={S, A, B, C}, T={a, b}, P={S→AB BC, A→BA a, B→CC|b, C→AB|a}이다.
N에 대해서 $A_1=S$, $A_2=A$, $A_3=B$, $A_4=C$라 두고
T에 대해서 $a_1=a$, $a_2=b$라 둔다.
A_i의 각 쌍에 대한 코우드를 구하면 표1과 같다.

표 1. N에 관련된 production 코우드
Table 1. Production code to N.

pair of S, N		code	pair of S, N		code
SS	A_1A_1	0 0 0 0	BS	A_3A_1	0 0 0 0
SA	A_1A_2	0 0 0 0	BA	A_3A_2	0 1 0 0
SB	A_1A_3	0 0 0 0	BB	A_3A_3	0 0 0 0
SC	A_1A_4	0 0 0 0	BC	A_3A_4	1 0 0 0
AS	A_2A_1	0 0 0 0	CS	A_4A_1	0 0 0 0
AA	A_2A_2	0 0 0 0	CA	A_4A_2	0 0 0 0
AB	A_2A_3	1 0 0 1	CB	A_4A_3	0 0 0 0
AC	A_2A_4	0 0 0 0	CC	A_4A_4	0 0 1 0

Terminal기호 a_i에 대한 코우드를 구하면 표2와 같다.

표 2. T에 관련된 production 코우드
Table 2. Production code to T.

symbol of T		code
a	a_1	0 1 0 1
b	a_2	0 0 1 0

2. 이차원 시스토릭 어레이의 구성

CFL 인식을 위한 이차원 시스토릭 어레이는 Chu 등에 의해서 제안되었고, 이것은 Kung 등이 제안한 동적 프로그래밍을 위한 어레이와 그 구성이 같다. 이차원 어레이는 삼각형의 구성 형태를 가지고 처리요소는 그림 1과 같다.[15]

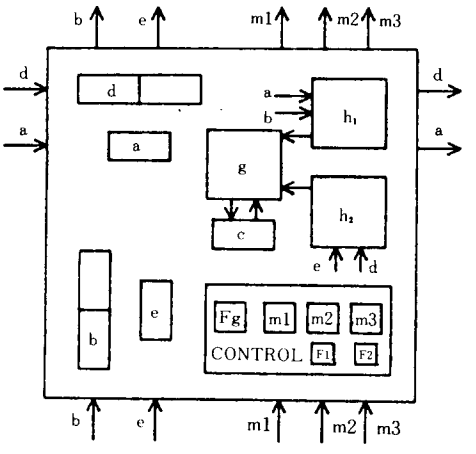


그림 1. CFL 인식을 위한 이차원 어레이의 처리요소
Fig. 1. The processing element of 2-dimensional array for recognition of CFL.

여기서 m2는 두 단위시간마다 [x, y]=[0, −1] 방향으로 이동되어 C레지스터의 내용이 a, e 레지스터로 옮겨지게하는 제어신호이고, m3는 셋 단위시간마다 두개의 처리요소로 이동되는 제어신호로 어레이에서 s=x−y이 짝수인 처리요소에서 a, e 레지스터의 내용이 d, b 레지스터로 각각 옮겨지게 한다. m1은 해당 처리요소의 위치를 나타내는 s의 값이 짝수인 여부를 알게하는 제어신호이다. Fg, F1, F2는 처리요소의 상태와 제어신호의 이동에 필요한 레지스터이다.

그림 1에서 h_1의 동작을 $R_{i,j}$로 표현하면 다음 식과 같다.

$$R_{i,j}=a_i \wedge b_j, \quad 1 \le i, \ j \le m \qquad (1)$$

여기서 ∧는 논리 AND, m=|N|이다.
h_2는 h_1의 동작과 같고 그 결과를 $Q_{i,j}$라 하면, $Q_{i,j}=e_i \wedge d_j$이다.

g는 production 코우드를 갖고있고, 다음 식의 동작을 먼저 행한다.

$$O_{i,j}=R_{i,j} \vee Q_{i,j}, \quad 1 \le i, \ j \le m \qquad (2)$$

여기서 $O_{i,j}$는 production 코우드 표의 위치를 나타내고, ∨는 논리 OR이다.

$T_{u,v,i}$를 기억된 코우드 표의 u, v위치의 i번째 비트로 두어 다음식의 동작을 수행한다.

$$G_i=T_{1,1,i} \vee \cdots \vee T_{m,m,i}, \quad 1 \le i \le m \qquad (3)$$

단, 여기서 $O_{u,v}$의 값이 '0'이면 그 위치의 $T_{u,v,i}$ 값은 기억된 코우드에 관계없이 '0'이 된다.
g의 결과 G는 C 레지스터에 기억된다.

처리결과(1, n)위치의 처리요소에서 a, e 레지스터의 S 비트값이 '1'이면 입력된 스트링은 수락(accept)되고 '0'이면 거부(reject)된다.

어레이에서 필요한 처리요소수는 $n(n+1)/2$이고 처리시간은 2n 단위시간이 소요된다. 처리할 스트링의 수가 많을 경우 연속으로 입력(block pipelining) 하여 처리하면 평균 $\lfloor (n+1)/2 \rfloor + 1$ 단위시간이 소요된다.[15]

3. 일차원 시스토릭 어레이의 설계

처리할 문제의 크기 n이 증가하면 이차원 어레이는 O(n)에 비례해서 입출력 채널의 수를 증가시키고 이것은 결국 VLSI의 핀수를 증가시킨다. 또한 입출력 채널이 증가하여 데이타전송율이 높더라도 호스트 컴퓨터의 입출력 데이타 전송율과 균형이 맞지 않으면 처리 속도가 향상되지 않는다. 이러한 문제를 해결하기 위하여 이차원 어레를 변형하여 일차원 어레이를 설계한다. 폴트 톨러런스(fault tolerance)를 용이하게 하기위하여 일방향(one-way) 선형 어레이가 되게한다.

그림 2는 이차원 어레이를 투영(projectoin)하여 일차원 어레를 구하는 과정을 나타낸다.

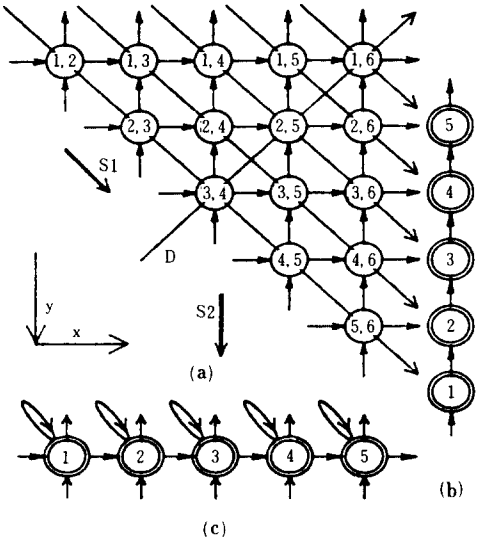


그림 2. 이차원 어레이를 일차원 어레이로 변형 (n=5)
Fig. 2. Transforming 2-dimensional array into 1-dimensional array (n=5).

투영 방향은 다양하게 주어질 수 있으나 결과의 일차원 어레이에서 처리요소의 수가 적고 기능이 간단한 방향, 요구되는 입출력 채널의 수가 문제의 크기에 관계없이 고정되는 방향 및 모든 통신 패스들이 한 방향이 되는 방향을 선택한다. 그림 2의 (b)는 [x, y]=[1, 1]방향으로 투영한 결과의 선형 어레이이고, 그림 2의 (c)는 [0, 1]방향으로 투영한 것이다.

CFL 인식을 위한 이차원 어레이는 [−1, 1]방향의 점선 D에 대해서 대칭이 되므로 (b)는 [−1, −1]방향의 투영 결과와 같고, (c)는 [0, −1]방향의 투영한 결과와 같다. 그림(c)는 각 처리요소에 셀프 루프가 존재하므로 이차원 어레이의 해당 처리요소의 결과를 기억해야 하므로 최대 n개의 기억공간이 필요하다. 또한 각 처리요소에 입출력 채널이 존재하므로 문제의 크기에 의존된다. 그래서 본 논문에서는 그림 2의 (b)를 선택한다.

이차원 어레이의 각 처리요소가 일차원 어레이에서 처리되는 시간을 스케줄하면 그림 3과 같다.

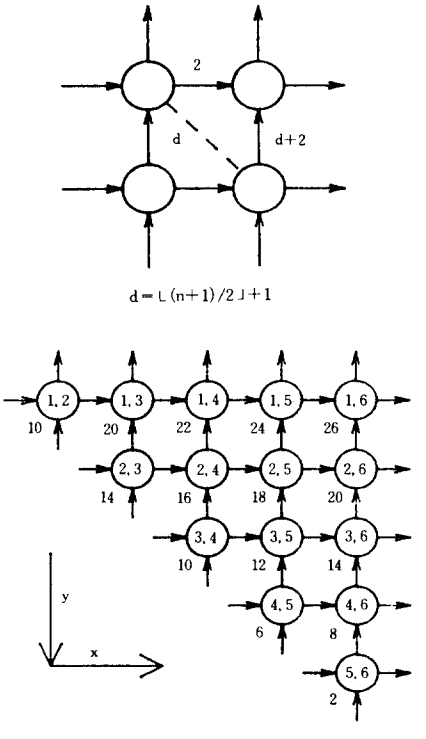


그림 3. 일차원 어레이를 위한 시간 스케줄링 (n=5)
Fig. 3. Time scheduling for 1-dimensional array (n=5).

이차원 어레이에서 [x, y]=[1, 0]방향의 패스(path)는 변환없이 일차원 어레이에 mapping 시켜서 이웃 처리요소의 처리 결과는 2 단위시간의 차이가 난다. [0, −1] 방향의 패스는 그림 4 (a)에서 d 단위시간을 지연시켜서 일차원 어레이에 mapping한다. 지연시간 d는 데이타의 흐름에서 유효한 데이타를 보호하기 위한 최소한의 시간이다. 그래서 d는 이차원 어레이에서 블럭 파이프라인 시간과 같다. 즉 d=⌊(n+1)/2⌋+1이다. 그림 3은 n=5가 되므로 d=4이다. (x, y) 위치의 처리요소에서 처리결과가 결정되는 시각은 식(4)와 같다.

$$t_{x,y}=d(n-y)+2(x-y) \qquad (4)$$

이차원 어레이에서 [0, −1]방향의 데이타 흐름을 조정하기 위한 레지스터의 크기 Df, Ds는 다음식과 같다.

$$Df=d+1,\quad Ds=d+2 \qquad (5)$$

여기서 Df, Ds는 이차원 어레이에서 e, b를 각각 나타낸다.

일차원 어레이를 위한 처리요소의 구성은 그림4와 같다.

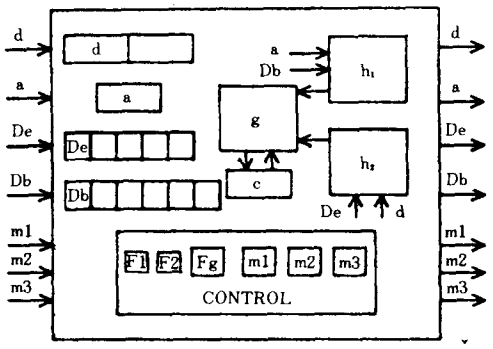


그림 4. CFL 인식을 위한 일차원 어레이의 처리요소 (n=5)
Fig. 4. The processing element of 1-dimensional array for recognition of CFL (n=5).

여기서 h1, h2, g의 동작, m1, m2, m3 등의 제어기능 및 a, d, c 레지스터는 이차원 어레이와 동일하다. 그림 4 에서 n=5이므로 식(5)에 의해서 Df=5, Ds=6이고 지연을 위한 쉬프트 레지스터의 갯수가 된다. 설계된 처리요소들로 구성된 CFL인식을 위한 일차원 어레이는 그림 5 과 같다.

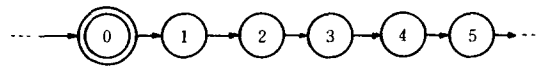


그림 5. CFL 인식을 위한 일차원 어레이 (n=5)
Fig. 5. The 1-dimensional array for recognition of CFL (n=5).

그림 5 에서 가장 왼편의 처리요소는 제3장의 표 2와 같은 terminal 기호를 처리하는 요소이다. 처리결과는 우측 끝의 처리요소로 식(4)에 의해서 t=d(n-1)+2n 단위시간에 출력된다. 이 때 a의 워드(word)에서 시작 기호 S에 대응되는 비트가 '1'이면 입력 스트링 w는 정의된 CFG에 의해서 생성된 것으로 수락된다.

Ⅳ. 평가 및 고찰

설계된 일차원 어레이는 문제의 크기가 n일 경우 n개의 처리요소를 요구하고, 식(4)에 의해서 ⌊(n+1)/2⌋(n−1)+3n−1의 처리시간을 갖는다. 즉 O(n)의 처리요소와 $O(n^2)$의 계산시간을 갖는다.

처리기의 이용율은 처리시간동안 전체 처리요소에 대한 사용된 처리요소의 비율로서 정의되고, 처리시간 T, 처리요소의 수 N, 처리시간 동안 이용된 처리요소의 수를 P라 하면 $\mu=P/(N, T)$이다. 일차원 어레이에서 P는 이차원 어레이에서와 같다. 일차원 어레이의 이용률은 μ_1, 이차원 어레이의 이용율을 μ_2라 하여 비교하면 식(6)과 같다.

$$\mu_2/\mu_1=[P/(n(n+1)/2)(2n)]/[P/(n(\lfloor(n+1)/2\rfloor (n-1)+3n-1))]\cong(n^2+5n-2)/(2n^2+2n) \qquad (6)$$

일차원 어레이의 이용율은 이차원보다 높고 n이 충분히 크면 2배가 된다.

그림 5에서 어레이는 일방향(one-directional) 선형이므로 폴트 톨러런스의 구현이 용이하고[8] 또한 I/O 채널의 수가 고정되므로 집적회로의 핀수가 n의 크기에 관계없이 고정된다.

본 논문에서 제안한 일차원 어레이는 Varman등이 제안한 동적프로그래밍 처리를 위한 선형 어레이[8]와 비교하면 표 2와 같다.

설계한 일차원 어레이의 각 처리요소에서 정확한 데이타 지연을 위한 추가되는 쉬프트 레지스터의 수는 2(⌊(n+1)/2⌋+1)이고, 이것은 문제의 크기 n에 의존되므로 어레이의 확장성이 결여된다. 그러나 Varman 방법 보다는 문제의 크기가 증가함에 따른 추가되는 쉬프터 레지스터의 수는 감소되었다. 이경

표 3. 본 논문에서 설계한 일차원 시스토릭 어레이와 Varman의 일차원 시스토릭 어레이와의 비교(문제의 크기=n)

Table 3. Comparison of the designed 1-dimensional systolic array in this paper and Varman's 1-dimensional systolic array (problem size = n).

	This paper	Method of Varman
Number of PEs	n	n
Processing Time	$\lfloor (n+1)/2 \rfloor (n-1)+3n-1$	$2(n^2+n-1)$
Shift-registers for delay/PE	data : $2\lceil (n+1)/2 \rceil +8$ control : none	data : $4n+12$ control : $2n+7$
Local memory & Address/PE	1 none	n $2(n+2)$

우 문제의 크기에 관계없는 일차원 시스토릭 어레이는 일반적으로 $O(n^2)$개의 처리요소를 요구한다.[10]

V. 결 론

본 논문에서는 context-free 언어의 인식을 위한 일차원 어레이를 제안하였다. 이차원 어레이를 최적의 방향으로 투영(projection)하여 일차원 어레이로 변형하고 시간 스케줄링을 했다.

제안한 어레이는 입력 스트링의 길이가 n일 경우 처리요소의 수는 n이고 처리시간은 $\lfloor (n+1)/2 \rfloor +3n-1$ 단위시간이 요구된다. 어레이가 동형이고, 일방향으므로 폴트 톨러런스가 용이하다. 처리기의 이용율은 이차원 어레이보다 높다. 입력 스트링인 non-terminal 기호의 입력 자연시간 d는 $\lfloor (n+1)/2 \rfloor +1$ 이다. 정확한 지연을 위한 변형된 두 데이타 레지스터들의 크기는 d+1, d+2 이고, 다른 동작과 제어는 이차원 어레이와 동일하다.

Varman 등이 제안한 동적 프로그래밍 처리를 위한 선형어레이와 비교하면 제안한 어레이는 하드웨어가 적게 요구되고 처리시간은 빠르다.

본 논문의 연구결과는 빠른 응답을 요구하는 음성인식,[16] 패턴인식의 실시간 처리에 응용될 수 있고, 일차원 어레이이므로 VLSI의 구현이 용이하다.

參 考 文 獻

[1] J.E. Hopcroft and J.D. Ullman, *Introduction to automata theory, languages and computation*, Addison-Wesley Pub.l pp. 125-141, 1979.

[2] K.Q. Brown, "Dynamic programming in computer science," Tech. Report CMU-CS-79-106, Carnegie-Mellon Univ., 1979.

[3] L. Snyder, L.H. Jamieson, D.B. Gannon and H.J. Siegel, *Algorithmically specialized parallel computers*, Academic Press, 1985.

[4] H.T. Kung, "Why systolic architectures?", *IEEE Computer*, vol. 15., no. 1, pp. 37-46, Jan. 1982.

[5] J.A. Fortes, K.S. Fu and B.W. Wah, "Systematic approaches to the design of algorithmically specified systolic arrays," Proc. of int'l Conf. on ASSP, pp. 300-303, Mar. 1985.

[6] S.Y. Kung, *VLSI array processors*, Prentice Hall, pp. 110-294, 1988.

[7] D.I. Moldovan, "ADVIS: a software package for the design of systolic arrays," *IEEE trans. on computer-Aided Design*, vol. Cad-6, no. 1, pp. 33-40, Jan. 1987.

[8] P.J. Varman and I.V. Ramakrishnan, "Dynamic programming and transitive closure on linear pipelines," Proc. of Int'l Conf. on parallel Processing, pp. 359-364, 1984.

[9] I.V. Ramakrishnan, D.S. Fussel and A. Silberschatz, "Mapping homogeneous graphs on linear arrays," *IEEE Trans. on Computers*, vol. C-35, no. 3, pp. 189-209, Mar. 1986.

[10] P. Lee and Z.M. Kedem, "Synthesizing linear array algorithms from nested for loop algorithms," *IEEE Trans. on Computers*, vol. 37, no. 12, pp. 1578-1598, Dec. 1988.

[11] L.J. Guibas, H.T. Kung and C.D. Tompson, "Direct VLSI implementation of combinatorial algorithms," CALTECH Conf on VLSI, pp. 509-525, Jan. 1979.

[12] M.C. Chen, "A design methodology for synthesizing parallel algorithms and architectures," *Journal of Parallel Distributed Computing*, vol. 2, pp. 461-491, 1986.

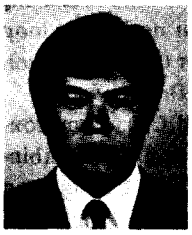
[13] G. Li and B.W. Wah, "Systolic processing for dynamic programming problems," Proc. of Int'l Conf. on Parallel Processing, pp. 434-441. 1985.

[14] K. Chu and K. Fu, "VLSI architectures for high speed recognition of context-free languages and finite-state Languages," Proc. of the 9th Annual Symposium on C.A., pp. 43-49, Apr. 1982.

[15] 우종호, 안광선, "Polyadic-nonserial 동적 프로그래밍 처리를 위한 시스토릭 어레이의 설계 및 효율적인 운영," 대한전자공학회 논문지, 제26권, 제 8 호, pp. 1178－1186, 1989년 8월.

[16] H. Ney, "Dynamic programming speech recognition using a context-free grammer," Proc. of Conf. on ASSP, vol. 1, pp. 69-72, Apr. 1987.

著 者 紹 介



禹 鍾 鎬 (正會員)

1954年 7月 9日生. 1978年 2月 경북대학교 전자공학과(전자계산기공학)졸업. 1981年 2月 경북대학교 대학원 전자공학과 (전산공학전공)석사학위 취득. 1986年경북대학교 대학원 전자공학과 박사과정 수료. 1978年～1981年 2月 경남공업전문대학 전자계산과 전임강사. 1981年 3月～현재 부산수산대학 전자공학과 부교수. 1987年 8月～1988年 8月 미국 랜슬러공대(RPI)컴퓨터학과 객원연구원. 주관심분야는 병열처리, VLSI 계산, cellular 오토마타 등임.

安 光 善 (正會員) 第25卷 第 7 號 參照
현재 경북대학교 전자계산기공학과 교수