論文 92-29B-2-1

MWLD 알고리듬을 이용한 문자열정합 1차원 Bit-Serial 어레이 프로세서의 설계

(A Study on 1-D Bit-Serial Array Processor Design for Code-String Matching Using a MWLD Algorithm)

朴 鍾 鎭* 金 銀 源* 趙 源 敬*

(Jong Jin Park, Eun Won Kim, and Won Kyung Cho)

要 約

본 논문에서는 문자열 정합 프로세서를 설계하기 위하여 기존의 WLD(weighted levenshtein distance) 알고리듬을 변형한 MWLD(modified weighted levenshtein distance) 알고리듬을 제안하였다. 제안된 MWLD 알고리듬은 문자사이의 거리를 Hamming Distance로 정의하는 방법에 의하여 용이하게 1차원 Bit-Serial 어레이 프로세서로 구성할 수 있다. 본 연구에서 설계된 프로세서는 실시간으로 입력되는 문자의 인식 실험에 적용하였다. 한글 기본획의 인식율은 98.65%로 좋은 결과를 얻었다.

Abstract

This paper is proposed a Modified WLD (Weighted Levenshtein Distance) algorithm for processor design of code-string matching. A proposed MWLD (Modified Weighted Levenshtein Distance) algorithm is consist of 1-dimension bit-serial array processor to pattern matching using a Hamming Distance. The proposed processor is applied to recognition of character with real time input. The recognition rate of Hangul strokes is resulted to 98.65%

I. 서 론

패턴 인식은 영상, 음성 및 문자열 정보 등을 일정한 패턴으로 표현하여 사전에 구성된 표준 패턴과 정합함으로 이루어진다.[1,2] 패턴 인식분야에 있어서, 패턴의 특징은 크게 수치적 표현과 비수치적 표현으로 나누어 기술할 수 있다. 수치적인 패턴의 표현으로서, n차원 공간에서의 좌표값, FFT 계수값, 히스토그램등에 의한 방법이 있으며, 비수치적 표현으로는 문자열에 의한 표현 방법이 있다.[3][4] 현재 WLD알고리듬은 문자열 정합을 위한 중요한 알고리듬으로 실시간 처리를 위한 어레이 프로세서 구조에 적합하기 때문에 이를 실현하기 위한 연구가 수행되고 있다.[5][6][7][8] 문자열 인식을 위한 문자열 정합은 대규모의 데이타로 구성된 데이타베이스에서 필요한 정보를 탐색할 경우 유용하게 이용된다.

본 논문에서는 문자열 정합을 위하여 기존의 WLD알고리듬을 변형한 MWLD 알고리듬을 제안하였다. 제안된 MWLD알고리듬은 병렬성과 동시성을 갖기 때문에, 시스토릭 어레이 프로세서의 실현에 적합하다.[9][10][11] MWLD 알고리듬은 문자사이의 유사도를 Hamming Distance로 정의하는 코딩 방식에 의하

*正會員, 慶熙大學校 電子工學科
(Dept. of Elec. Eng., Kyunghee Univ.)
接受日字 : 1991年 9月 4日

여 문자사이의 유사도를 계산하는 회로를 용이하게 설계할 수 있다. 또한 문자열 정합과정에서 문자사이의 유사도를 고려할 수 있기 때문에 인식율을 향상시킬 수 있었다.

Ⅱ. 文字列 정합 알고리듬

1. WLD(weighted levenshtein distance) 알고리듬

WLD알고리듬은 입력 문자열을 표준 문자열로 표환하는 과정에서 각 변환에 일정한 가중치를 부과함으로써 거리값을 구한다. W_i는 삽입 변환 $I(x_i, r_j)$에서 r_j를 삽입할 때 부과되는 가중치이고, W_d는 삭제 변환 $D(x_i)$에서 x_i를 삭제할 때 부과되는 가중치이다. 그리고 W_s는 대체 변환 $S(x_i, r_j)$에서 x_i를 r_j로 대체할 때 부과되는 가중치이다. 그림1은 각 경로를 지날 때, 가중치가 가산되는 과정이다. 그림1에서 (i, j)노드에 도달할 수 있는 가능한 경로는 (i-1, j)로부터의 삭제 경로, (i, j-1)로 부터의 삽입경로, (i-1, j-1)로 부터의 대체 경로가 있다. 이들 경로를 지날 때 계산되는 거리값중 가장 작은 값을 취함으로써 최적의 변환 경로에 따른 부분 거리값 P(i, j)를 구할 수 있다. 최적 변환에 의한 부분 거리값 P(i, j)의 계산은 식 (1)과 같다.

$$P(i,j)=Min\begin{vmatrix} P(i\text{-}1,j)+W_d \\ P(i,j\text{-}1)+W_i \\ P(i\text{-}1,j-1)+W_s \end{vmatrix} \qquad (1)$$

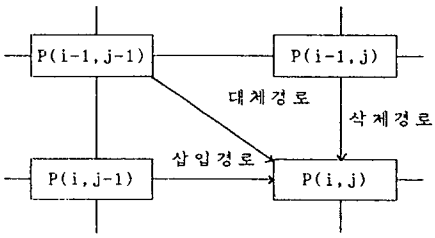

그림 1. WLD알고리듬의 각 변환 경로에 따른 가중치 가산 과정
Fig. 1. Weighted addition through the path with WLD algorithm.

WLD알고리듬에 의한 거리값 계산 과정의 예는 그림2에 나타내었다. 이때, 표준 문자열 R=ababb이며, 입력 문자열 X=aabaab이다. X와 R을 일치시키기 위한 최적 변환 과정은,

$$X=aabaab \xrightarrow{D(a)} abaab \xrightarrow{S(a,b)} ababb=R$$

과 같음을 알 수 있다. 이때, $d^w(X, R)=1*W_d+1*W_s$가 된다. 그림2의 예에서 부과되는 가중치는 $W_i=3$, $W_d=2$, $W_s=5$로 하였으며, 이때 두 패턴 X와 R의 최소 거리값 $d^w(X, R)=1*2+1*5=7$이다. 그림2의 예에서와 같이 WLD알고리듬은 문자사이의 유사도에 상관없이 각 변환의 가중치가 일정하기 때문에 문자사이의 방향차가 있는 경우, 방향차에 관계없이 유사도가 모두 같으므로 두 문자열의 구별이 어려워진다. 그러므로, 이 연구에서는 문자사이의 유사도에 따라 가중치를 변환할 수 있는 MWLD알고리듬을 제안하였다.

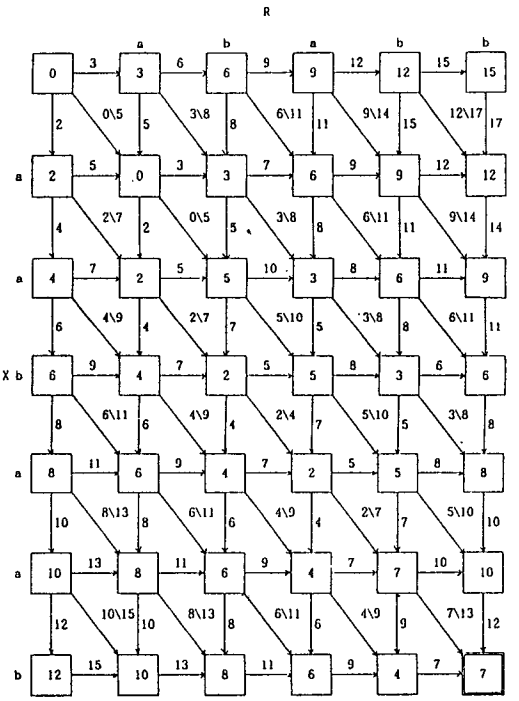

그림 2. WLD알고리듬의 계산예
Fig. 2. Example of WLD algorithm.

2. 변형된 WLD(modified weighted levenshtein distance) 알고리듬

기존 WLD알고리듬에서 문자열 정합시 코드열 데이타의 방향차에 관계없이 정합된 값과 정합되지 않은 값을 이용하였다.[5][8]

본 논문은 문자열 정합시 방향 코드의 차를 고려하기 위하여 Hamming distance를 적용하였다. 그림3은 8방향 코드를 나타낸다. 각각의 8방향 코드는 표

1에서와 같이 4개의 비트로 구성하였으며, 이웃되는 각각의 방향 코드의 차는 "1"이고, 방향 코드의 최대차는 "4"이다. 이 코드를 이용함으로서 입력 패턴 코드와 표준 패턴 코드사이의 방향 코드 차를 단순히 Hamming Distance로 구할 수 있다. 실제 두개의 방향 코드 d와 g를 계산한 예는 다음과 같다.

d	0	1	1	1
g	1	1	0	0
EX-OR	1	0	1	1

Hamming Distance(문자사이의 유사도)=3 코드 d와 g에서 다른 비트의 수가 3개이므로, 문자 사이의 유사도가 3임을 알 수 있다.

표 1. 8방향 코드의 코딩
Table 1. Coding of 8-directional code.

방향코드	4비트 코드			
a	0	0	0	0
b	0	0	0	1
c	0	0	1	1
d	0	1	1	1
e	1	1	1	1
f	1	1	1	0
g	1	1	0	0
h	1	0	0	0

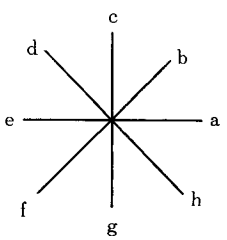

그림 3. 8방향 코드의 기술
Fig. 3. Representation of 8-directional code.

MWLD알고리듬은 입력 문자열 패턴과 표준 문자열 패턴을 정합할 때 수평, 수직 가중치와 문자사이의 유사도(HD)에 의하여 계산이 이루어진다.[12] T는 수평, 수직 이동에 부과되는 가중치이다. 가중치 T의 범위는 식(2)와 같이 정의할 수 있다. 여기에서 N은 기술된 방향 코드의 갯수이다.

$$0<T\leqq N/4 \qquad (2)$$

또한 사선 이동에 부과되는 가중치 HD(i, j)는 두 문자사이의 Hamming Distance에 의하여 계산된 방향 코드의 차이값이다.

그림4는 각 경로에서 가중치가 누적되는 과정이다. (i, j)노드에 도달할 수 있는 가능한 경로는 (i−1, j), (i, j−1)로 부터 수직, 수평 이동 경로와 (i−1, j−1)로 부터의 사선 이동 경로가 있다. 이들 경로를 지날 때 계산되는 거리값 중 최소값을 취함으로써(i, j) 노드에 이르는 최적 이동 경로의 부분 거리값 P(i, j)를 구할 수 있으며, 식(3)과 같다.

$$P(i,j)=Min\begin{vmatrix}P(i-1,j)+T\\P(i,j-1)+T\\P(i-1,j-1)+HD(i,j)\end{vmatrix} \qquad (3)$$

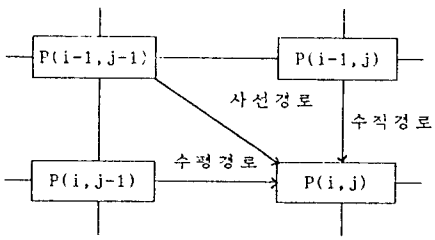

그림 4. MWLD알고리듬의 각 경로에 따른 가중치 가산 과정
Fig. 4. Weighted addition through the path with MWLD algorithm.

제안된 MWLD알고리듬에 의한 거리값 계산 과정의 예는 그림5에서 나타내었다. 이때 표준 문자열 R=ababb이며, 입력 문자열 X=aabaab이고, 그림 5의 예에서 수평, 수직 이동 경로 가중치 T=2로 하였을 경우, 두 패턴의 최소 거리값 $d^{sum}(X, R)=1*T+1*HD(x_4, r_5)=3$이 된다.

Ⅲ. 시스토릭 어레이 프로세서의 설계

1. 문자열 정합을 위한 Bit-serial 어레이 프로세서

제안된 MWLD알고리듬의 수행을 위한 1차원 Bit-serial 어레이 프로세서의 설계는 2차원 어레이 프로세서에 기초를 두며,[13,14] 그림6은 MWLD 알고리듬의 2차원 어레이를 1차원 어레이로 사상(mapping) 하는 과정이다. 시스토릭 어레이 프로세서의 PE의 갯수는 입력 문자열의 코드수에 비례하여 증가된다. 또한 각각의 문자열은 4비트로 구성하였으며, 모든 데

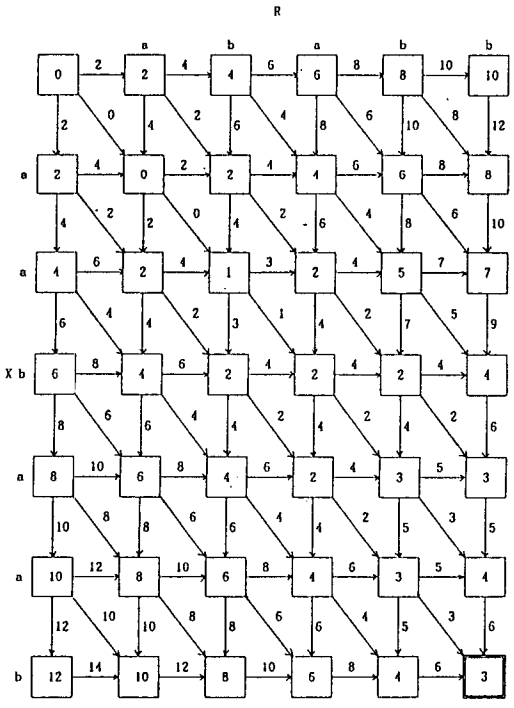

그림 5. MWLD알고리듬의 계산 예
Fig. 5. Example of MWLD algorithm.

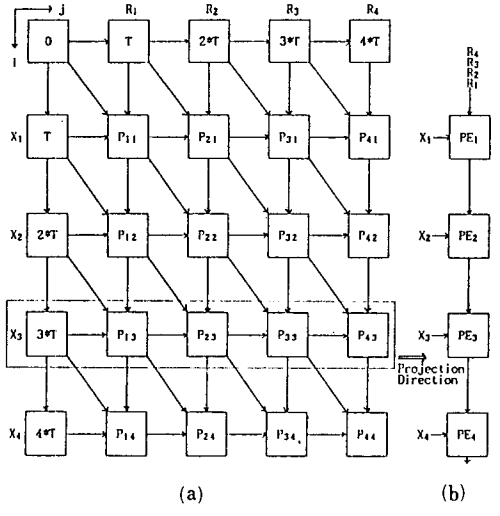

그림 6. 2차원을 1차원 시스토릭 어레이로의 사상
Fig. 6. Mapping of 2-D to 1-D systolic array.

이타는 한 비트씩, 입력되고, 출력된다. 그림6의 (a)와 (b)를 비교하여 보면, 최상위 PE, 즉 PE_1은 수평이동 경로에 대한 초기치를 T씩 증가시켜야 함을 알 수 있고, $PE_1 \rightarrow PE_2 \rightarrow PE_3 \rightarrow \cdots$와 같이 하위 PE로 데이타 의존 관계가 생성될 경우, 각 PE에 수직이동 경로의 부분 거리값에 대한 초기치가 설정되어야 함을 알 수 있다. 또한 동일한 k인덱스에 걸쳐있는 PE는 동시처리가 이루어지므로 1차원 Bit-serial 시스토릭 어레이의 경우 PE의 모든 계산은 k인덱스에 따라 $|X_n|+|R_m|-1$번의 반복으로 계산이 이루어진다. 그림6을 바탕으로 제안된 MWLD알고리듬 수행을 위한 1차원 Bit-serial 시스토릭 어레이 프로세서의 PE를 설계할 수 있다. PE는 부분합 처리부와 Hamming Distance 계산부로 나눌 수 있고, 부분합 처리부는 다음과 같은 순서로 계산한다.

1) $P_{out} \leftarrow S_2$ Register$+HD(X, R)$;
2) IF(X_{init}) THEN S_1 Register
   $\leftarrow S_2$ Register$+T$;
   ELSE S_1 Register $\leftarrow P_{in}$
3) IF(R_{init}) THEN S_2 Register $\leftarrow T_{in}$;
   ELSE S_2 Register $\leftarrow S_1$ Register;

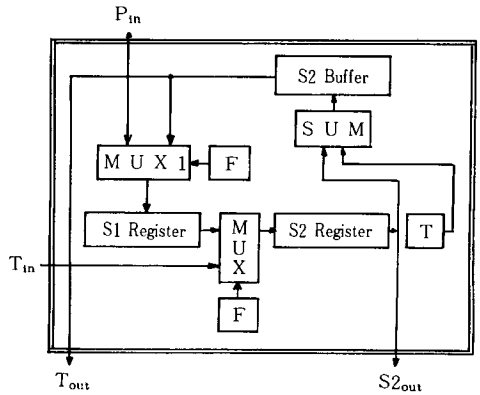

그림 7. 부분합 처리부의 블럭 다이어 그램
Fig. 7. Block diagram of partial sum processing unit.

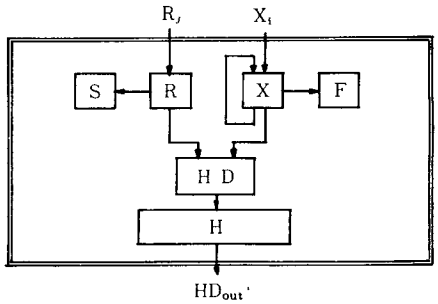

그림 8. Hamming distance 계산부의 블럭 다이어 그램
Fig. 8. Block diagram of hamming distance computation unit.

표 2. 부분합 처리부의 레지스터 기능 및 연산기의 기능

Table 2. Register and operator function of partial sum processing unit.

레지스터 및 연산기	기 능
S1 레지스터	-다음 단계에서 계산될 부분합을 저장 -부분합 계산후 S2 레지스터로 이동 -8비트 쉬프트 레지스터
S2 레지스터	-부분합 계산에 이용되는 레지스터 -8비트 쉬프트 레지스터
S2_Buffer	-부분합 계산후 S2의 내용과 T의 내용을 가산하여 저장 -8비트 쉬프트 레지스터
T	-수직, 수평이동시 부과되는 가중치 저장 -초기 Set를 할 수 있음 -8비트 쉬프트 레지스터
Tin	-상위 PE의 S2_Buffer의 내용을 저장 -8비트 쉬프트 레지스터
MUX1	-플래그 S에 의해 Pin의 내용과 S2_buffer의 내용을 선택하는 연산기
MUX2	-플래그 F에 의해 Tin의 내용과 S1의 내용을 선택하는 연산기
SUM	-가산기

Hamming Distance 계산부는 다음과 같이 동작한다.

1) Flag Set ← X Register, R Register;
2) H Register ←
   X Register EX-OR Register;
3) P_{out} ← H Register;

그림8에서 X와 R은 입력 문자열과 표준 문자열을 저장하는 레지스터이다. 입력 문자열 x_i와 표준 문자열 r_j가 한 비트씩 입력되면 HD에서 x_i와 r_j의 Hamming Distance를 계산하여 H 레지스터에 저장된다. 플래그 F와 플래그 S는 각각 입력 문자열의 시작 코드와 표준 문자열의 시작 코드를 판독하여 저장한다. 입력 문자열의 코드 "0100"이 입력되면 플래그 S는 1로 세트되고, 플래그 F는 표준 문자열의 "0100"코드가 입력되면 1로 세트된다.

2. PE(processing element)의 세부설계

1절에서 설계한 부분합 처리부와 Hamming Distance 계산부는 실제적으로 Binary Counter, MUX, 가산기, 레지스터, 기본적인 Logic Gate로 구성되어진다. 각 연산기는 모두 Bit-serial 방식으로 입출력과 연산이 이루어진다.

본 절에서 구성된 모든 연산기는 플립 플롭과 게이트 차원(Gate level)에서 설계하였다. 그림 9에서 Hamming Distance 계산부는 2개의 4bit 레지스터, 4개의 EX-OR 게이트와 2개의 8비트 레지스터로 구성되어 있다. X레지스터는 입력 문자열을 저장하고, Y 레지스터는 표준 문자열을 저장한다. EX-OR 게이트는 입력 코드와 표준 코드가 일치할 경우, 1을 출력하고, 일치하지 않을 경우 0을 출력하여 H 레지스터에 저장한다. 그리고 X 레지스터와 Y 레지스터를 조사하여 S플래그와 F플래그를 세트시킨다. 가산기는 2개가 사용되며, SUM_1은 S_2 레지스터와 H레지스터를 가산하여 P_{out} 레지스터에 저장하고, SUM_2는 S_2레지스터와 T레지스터를 가산하여 S_2 버퍼에 저장한다. 그림9에서 구성된 모듈을 이용하여 제안된 MWLD 알고리듬을 위한 1차원 Bit-serial 시스토릭 어레이의 PE를 설계하였다. 전체 PE의 세부 설계는 그림10에서 보여준다. 레지스터와 레지스터사이, 레지스터와 연산기 사이의 모든 데이타 라인들이 단일 라인으로 구성되어 있으며, 연산기의 구조가 매우 단순화되어 있음을 알 수 있다.

표 3. Hamming Distance 계산부의 레지스터 기능 및 연산기의 기능

Table 3. Register and operator function of hamming distance computation unit.

레지스터 및 연산기	기 능
X	-입력 코드를 저장 -순환하는 4비트 쉬프트 레지스터
R	-표준 코드를 저장 -1비트 저장하는 플립플롭
H	-입력 코드와 표준 코드의 EX-OR된 결과를 저장 -4비트 쉬프트 레지스터
S Flag	-표준 코드가 시작코드일 때 "1"로 세트
F Flag	-표준 코드가 끝 코드일 때 "1"로 세트
HD	-두 코드의 비트 차를 계수 -EX-OR과 Binary Counter로 구성

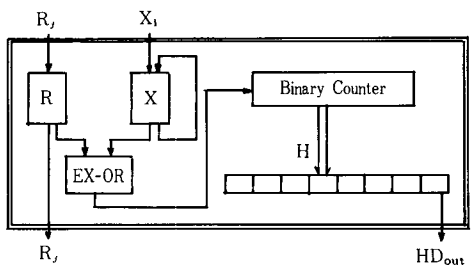

그림 9. HD의 세부 설계

Fig. 9. Internal block diagram of HD.

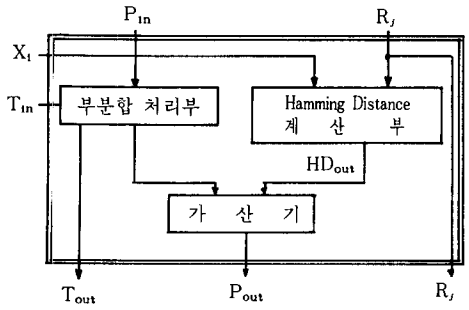

그림10. 1차원 Bit-serial 어레이의 PE의 설계
Fig. 10. Design of PE with 1-D Bit-serial array.

Ⅳ. 실험 및 고찰

1. 한글 자소 인식에 대한 실험

본 논문에서 제안된 MWLD알고리듬의 유용성을 고찰하기 위하여 한글 자소의 기본 획 인식에 적용하였다. 문자열 인식 시스템 구성도는 그림11과 같이 크게 3단계로 나눌 수 있다. 첫번째 단계는 입력 단계로서 테블렛상에 펜이 접촉되었을 때부터 입력이 발생되다. 펜의 위치가 이동함에 따라 코드는 벡터적으로 발생하게 된다. 두번째 단계는 전처리 단계에서 발생된 문자열을 8방향 코드로 변환시킨다. 또한, 발생한 8방향 코드는 표1에서 구성된 4 비트의 코드로 변환된다. 세번째 단계는 4비트로 구성된 코드는 1차원 Bit-serial 어레이 프로세서에 입력되어 사전에 구성된 표준 문자열과의 거리값을 계산하여 최소값을 갖는 표준 패턴으로 인식한다.

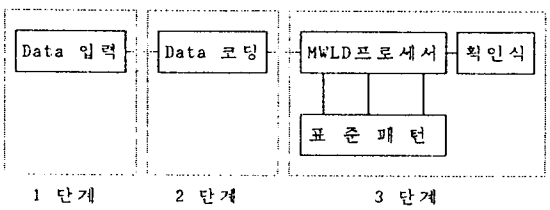

그림11. 문자열 인식 시스템
Fig. 11. Code-string recognition system.

표4는 표준 문자열 패턴으로 사용된 한글 자소의 표준획을 나타낸다. 한글 자소를 구성하는 표준 획은 20개로 하였고, 8방향 코드 벡터를 이용하였다. 모든 표준 패턴은 획의 길이를 12개로 정규화 하였다. 표5는 한글 자소를 구성하여 각 획의 인식율을 WLD알고리듬과 MWLD알고리듬으로 구분하여 나

표 4. 한글 자소의 표준획의 구성
Table 4. Construction of reference stroke with hanguel phoneme.

NO	표준 획	표준획 코드열											획 길이
1	ㄷ	0	0	0	4	4	5	6	6	7	0	0	12
2	ㄹ	0	0	7	6	5	4	0	0	4	6	0	12
3	ㄹ	0	0	0	6	4	4	4	6	6	0	0	12
4	ㅈ	0	0	0	7	6	5	4	4	4	0	0	12
5	ㅀ	6	6	2	1	0	0	7	6	5	4	0	12
6	ㅇ	6	6	6	6	5	5	4	3	2	1	0	12
7	ㅓ	6	6	6	7	7	0	0	1	1	2	2	12
8	ㅋ	7	6	5	4	4	0	0	0	6	5	5	12
9	ㅎ	0	0	0	0	5	5	6	7	0	1	2	12
10	ㅅ	6	6	6	5	5	4	4	0	0	0	0	12
11	ㅎ	6	5	4	0	0	5	6	7	0	1	2	12
12	ㅣ	6	6	6	6	6	6	6	6	6	6	6	12
13	ㅡ	0	0	0	0	0	0	0	0	0	0	0	12
14	ㅅ	5	5	5	5	5	5	5	5	5	5	5	12
15	ㅅ	7	7	7	7	7	7	7	7	7	7	7	12
16	ㄱ	0	0	0	0	0	5	5	5	5	5	5	12
17	ㄱ	0	0	0	0	0	0	6	6	6	6	6	12
18	ㅇ	4	5	5	6	7	0	0	1	2	3	4	12
19	ㅇ	4	3	2	1	1	0	0	7	6	5	5	12
20	ㄴ	6	6	6	6	6	0	0	0	0	0	0	12

표 5. 한글 자소의 획 인식율
Table 5. Stroke recognition rates of hanguel phoneme.

Stroke NO.	실험횟수	WLD	MWLD	
			T=1	T=2
		인식율	인식율	인식율
1	100	100%	98%	95%
2	100	84%	100%	99%
3	100	86%	95%	99%
4	100	97%	99%	98%
5	100	98%	100%	100%
6	100	100%	99%	98%
7	100	100%	100%	100%
8	100	96%	99%	99%
9	100	93%	95%	97%
10	100	97%	100%	98%
11	100	84%	94%	96%
12	100	100%	100%	100%
13	100	100%	100%	100%
14	100	100%	100%	100%
15	100	100%	100%	100%
16	100	100%	100%	99%
17	100	100%	100%	100%
18	100	96%	99%	100%
19	100	84%	93%	95%
20	100	100%	100%	100%

타낸다. 표준획 1과 6의 경우는 MWLD 알고리듬의 획 인식율이 WLD알고리듬 보다 더 낮음을 볼 수 있다. MWLD알고리듬에서 표준획 1(ㄷ)은 표준획 13(一)으로 인식하는 경우가 있었고, 표준획 6(ㅓ)은 표준획 19(ㅇ)로 인식하는 경우가 있었다. 발생된 코드열의 분석에 의하여 오인식 원인은 필기자의 필기 습관에 기인함을 알 수 있었다. MWLD알고리듬의 한글 자소의 기본 획 인식률은 98.65%였다.

2. MWLD알고리듬의 고찰

1) 방향 벡터에 따른 인식율 비교

본 절은 8방향, 16방향의 방향 코드에서 입력 패턴과 표준 패턴사이의 인식율을 비교하였으며, T의 가중치를 유동적으로 변화시킴으로서 제안된 MWLD 알고리듬의 유용성을 고찰하였다.

표 6. 방향 코드벡터에 의한 획 인식율
Table 6. Stroke recognition rates by directional code.

알 고 리 듬			8방향 코드	16 방향 코드
WLD		실험횟수	2000	2000
		인 식 율	96.10%	94.00%
MWLD	T=1	실험횟수	2000	2000
		인 식 율	98.55%	98.60%
	T=2	실험횟수	2000	2000
		인 식 율	98.65%	98.80%
	T=3	실험횟수	X	2000
		인 식 율	X	98.54%
	T=4	실험횟수	X	2000
		인 식 율	X	98.82%

표6에서와 같이 WLD알고리듬에 의한 획 인식율은 8방향 코드보다 16방향으로 구성된 코드의 인식율이 낮았다. 반면에 MWLD알고리듬은 8방향과 16방향에 의한 인식율이 거의 유사하게 나타났다. 따라서 방향 코드 벡터의 수를 크게 하여 패턴을 세부적으로 기술하였을 경우, MWLD알고리듬은 WLD알고리듬에 비하여 인식율이 더욱 높아진다. 한글 자소 인식에서 8방향 코드 벡터를 이용하였을 경우, 가중치(T)를 2로 하였을 때 인식율이 가장 높았고, 16방향 코드 벡터를 이용하였을 경우, 가중치(T)를 4로 하였을 때 인식율이 가장 높았다.

2) MWLD와 WLD알고리듬의 PE 비교

본 논문에서 제안된 MWLD알고리듬으로 구성된 1차원 Bit-Serial 시스토릭 어레이의 PE의 기능을 평가하기 위하여 [8]에서 제안한 WLD 알고리듬으로 구성된 Bit-serial 시스토릭 어레이의 PE와 비교, 분석하였다. 표7과 같이 비교의 대상은 외부 데이타 라인의 수, 내부 데이타 라인의 수, 연산기의 수로 하였다. MWLD알고리듬에 의하여 구성된 Bit-serial 어레이 프로세서는 WLD알고리듬에 비하여, 외부 데이타 라인 수는 약간 증가했지만 내부 데이타 라인 수는 많은 감소를 가져왔다. 따라서 여러개의 PE로 구성된 어레이 프로세서 설계시 MWLD 알고리듬은 WLD알고리듬 보다 더욱 단순하게 설계할 수 있다.

표 7. WLD알고리듬의 PE와 MWLD알고리듬의 PE 비교
Table 7. Compare with PE of WLD algorithm and MWLD algorithm.

		WLD알고리듬	MWLD알고리듬
외부 데이타 라인의 수		5	7
내부 데이타 라인의 수		21	13
연산기의 수	EX-OR	4	1
	AND	10	0
	OR	2	0
	전가산기	3	2
	MUX	1	3
	Register	11	6
	Flip Flop	9	7

V. 결 론

본 논문에서는 문자열사이의 유사도를 이용하여 문자열을 정합하는 MWLD알고리듬을 제안하였다. 제안된 MWLD알고리듬은 문자사이의 유사도가 Hamming Distance로 표시되는 문자의 코딩 방식을 이용하였다. 그 결과 문자사이의 유사도를 단지 EX-OR만을 이용하여 쉽게 계산할 수 있기 때문에 문자열 인식에 문자 사이의 유사도를 이용할 수 있었다. 제안된 MWLD알고리듬을 대규모 데이타의 실시간 처리에 이용하기 위하여 Bit-serial 어레이 프로세서를 프로세서를 설계하였다. 설계된 1차원 Bit-serial 시스토릭 어레이는 데이타의 전송 라인이 단순하기 때문에, VLSI 회로 설계시 칩 내부의 데이타 전송 버스의 수를 크게 줄일 수 있다.

본 논문에서 제안된 MWLD알고리듬을 이용하여 설계된 1차원 Bit-serial 시스토릭 어레이 프로세서의 유용성을 고찰하기 위하여, 제안된 프로세서와 [8]에서 구성된 WLD 알고리듬을 이용한 1차원 Bit-serial 시스토릭 어레이 프로세서의 인식율과 하드웨

어의 복잡도를 비교하였다. 또한, 인식율을 높이기 위하여 문자열 정합 계산에서 부과되는 가중치 T를 인식 대상에 따라 변화시켜, 가장 적당한 T값을 구할 수 있게 하였다.

본 논문에서는 실시간 한글 필기시 발생하는 20개의 표준 획을 인식하는 실험을 하였다. 한글 자소 인식 실험에서 MWLD알고리듬의 가중치가 2인 경우 가장 높은 인식율을 얻었다. 제안된 MWLD알고리듬에 의한 한글 자소의 획 인식율은 98.65% 이었고 WLD알고리듬에 의한 획 인식율은 96.10% 이었다. 따라서 한글 자소의 획 인식율에서 WLD알고리듬의 우수함을 알 수 있었다. 제안된 MWLD알고리듬을 이용한 문자열 정합 프로세서는 고속성과 하드웨어의 간결성으로 인하여 대규모의 문자열을 실시간으로 정합하여야 하는 응용 분야(문자 인식, 대규모 데이타베이스의 탐색, 물체 인식등)에 효과적으로 이용될 수 있을 것으로 기대된다.

參 考 文 獻

[1] King-Sun Fu, Y.T. Chien and Gerald P. Cardillo, "A dynamic programming approach to sequential pattern recognition," *IEEE Trans. PAMI* , vol. 8, no. 3, May, 1986.

[2] King Sun Fu, "VLSI for pattern recognition and image processing," Springer-Verlag, 1984.

[3] 박종진외 4명, "필기체 한글, 영문자 및 숫자 패턴의 온라인 인식," 대한전자공학회 추계종합학술대회 논문집, vol, 12, no, 2, pp. 651-620, 1989.

[4] Lionel M. Ni and Anil K. Jain "A VLSI systolic architecture for pattern clustering," *IEEE Trans. PAMI,* vol. 7, no. 1, Jan., 1985.

[5] 김은원, "병렬성을 갖는 WLD알고리듬을 이용한 온라인 필기체 문자 인식에 관한 연구," 경희대학교 대학원 석사학위 논문, 1990.

[6] H.D. Cheng and K.S. Fu, "VLSI architecture for dynamic time-warp recognition of handwritten symbols," *IEEE Trans. ASSP,* vol. 34, no. 3, Jun., 1986.

[7] Mehdi Hatamian and Glenn L. Cash, "Parallel bit-level pipelined VLSI designs for high-speed signal processing," *Proceedings of The IEEE* vol. 75, no. 9, Sep., 1987.

[8] 김윤제, "WLD알고리듬을 위한 1차원 Bit-serial 시스토릭 어레이 프로세서의 설계에 관한 연구," 경희대학교 대학원 석사학위 논문, 1991.

[9] Neil Weste, David J. Burr and Bryan D. Ackland, "Dynamic time warp pattern matching using and integrated multiprocessing array," *IEEE Trans. Computers,* vol. 32, no. 8, Aug., 1983.

[10] David J. Burr, Bryan D. Ackland, Neil Weste, "Array configuration for dynamic time warping," *IEEE Trans.* vol. ASSP-32, no. 1, Feb. 1984.

[11] S.Y. Kung, "VLSI Array Processors," Prentice Hall, 1988.

[12] 박종진외 2명, "변형된 DTW 알고리듬을 이용한 문자열 정합," 대한전자공학회 추계종합학술대회 논문집, vol. 13, no. 2, pp. 223-226, 1990.

[13] Mary Jane Irwin, "A digit pipelined dynamic time warp processor," *IEEE Transn. ASSP,* vol. 36, no. 9, Sep., 1988.

[14] John V. McCanny, "The use of data dependence graphs in the design of bit-level systolic arrays," *IEEE Trans.*, vol. ASSP-38, no. 5, May, 1990.

著 者 紹 介

朴 鍾 鎭 (正會員)

1966年 12月 17日生. 1989年 경희대학교 전자공학과 졸업(공학 학사). 1991年 경희대학교 대학원 전자공학과 졸업(공학석사). 1991年~현재 경희대학교 대학원 전자공학과 박사과정 재학중. 주관심 분야는 패턴인식, VLSI설계등으로서 실시간 인식을 위한 알고리듬 연구 및 고속 처리용 프로세서 설계 분야임.

金 銀 源 (正會員) 第28卷 B編 第10號 參照
현재 경희대학교 대학원 전자공학과 박사과정 재학중.

趙 源 敬 (正會員) 第28卷 B編 第10號 參照
현재 경희대학교 전자공학과 교수