고해상도 윤곽선 문자 발생가속기 설계에 대한 연구

서 주 하 · 안 태 영

A Study on The design of Accelerator of The Outlined Font Generation

Ju-Ha Seo, Tae-Young Ahn

ABSTRACT

This paper presents a design of the accelerate circuit for the conversion of the vector font data into the bit-mapped image. Among the Bezier curve algorithm, the subdivision algorithm gives the good performance and easy hardware implementation. The sequencer is realized by the proprammable gate array and the processing unit is composed of EPLDs and TTL ICs.

1. 서 론

문화수준이 향상되고, 과학기술이 발달함에 따라 사람들은 일상 생활에서 점점 더 많은 문자정보를 접하게 되고, 또 문화욕구가 확대됨에 따라 글자체에 있어서도 보다 아름답고 다양한 것을 요구하게 되었다. 이러한 요구에 부응하여 많은 종류의 출판 시스템이 출현하였으며, 그중에서도 컴퓨터를 이용한 DTP 시스템은 이런 요구에 유연하게 대처할 수 있다고 말할 수 있다.

DTP 시스템에서 사용되는 글자체를 표현하는 방법중에는 점행렬을 이용한 방법이 있는데, 문자 출력시 디코딩(Decoding)이 필요없으므로 고속의 출력이 가능하나 문자체의 종류가 많아짐에 따라 문자정보의 저장량이 증가하게 되며, 글자를 확대하면 그 모양이 거칠어 지는 문제가 발생한다. 또 이 점행렬 문자체 정보를 run-length code등 압축하여 문자체 정보량을 줄여서 표현하는 방법이 있으나 이도 확대시에는 같은 문제가 발생한다. 이와같은 문제를 해결할 수 있는 방법중의 하나가 Bezier 곡선과 같은 자유곡선을 이용하여 문자를 표현하는 방법이다. 이 방법은 확대 및 축소시 거칠어지는 문제는 발생하지 않으나 연산량이 많아 처리시간이 많이 걸린다. 이와같이 곡선을 사용하여 문자의 윤곽선을 표현하는 글자체를 윤곽선문자라 하며 이는 고해상도의 문자출력 시스템에는 꼭 필요한 것이다.

본 논문은 3차 Bezier 곡선 알고리즘으로부터 윤곽선 문자를 생성하는 시스템의 개발에 관한 연구이다. 윤곽선 문자생성시 가장 연산량이 많은 3차 다항식 연산을 부분분할(Subdivision) 알고리즘을 이용한 하드웨어로 구현하여 연산시간을 단축함으로써 수행능력을 향상시켰다.

2. Bezier 곡선 알고리즘

2-1. Bezier 곡선 다항식

Bezier 곡선은 다항식으로부터 자유곡선을 표현하기 쉽도록 곡선이 지나는 양 끝점과, 양 끝점을 지나는 점에서의 기울기와 곡선의 모양을 결정하는 제어점들로 구성되어 있다. 자유곡선을 표현하는데 가장 편리한 것은 곡선이 지나는 양 끝점과 각점에서의 기울기를 나타내는 두개의 제어점을 갖는 경우이다. 제어점이 하나이면 이웃한 다른 곡선과의 연결이 부드럽지 못하며 제어점이 세개 이상이면 곡선의 모양이 이웃한 제어점의 변화에 따라 함께 변하므로 모양을 예측하기가 어렵다.

n+1개의 Bezier점 $P_0, P_1, P_2, \cdots, P_{n-1}, P_n$를 갖는 n차 Bezier 곡선을 Bernstein 다항식(Bernstein polynomial)로 표현하면 다음과 같다.

$$O(t)=P_0B_0^n(t)+P_1B_1^n(t)+\cdots+P_nB_n^n(t) \qquad 0 \le t \le 1$$

$$B_k^n(t)={}_nC_k(1-t)^{n-k}\cdot t^k \qquad k=0, 1, 2, \cdots, n$$

$${}_nC_k=n!/(k!(n-k)!)$$

2-2. de Casteljau 알고리즘

Bernstein 다항식의 recursive한 성질을 이용하여 Bezier점 $P_0^0, P_1^0, P_2^0, P_3^0$인 3차 Bezier 곡선을 표현하면

$$P_0^1(t) = (1-t)P_0^0+tP_1^0$$
$$P_1^1(t) = (1-t)P_1^0+tP_2^0$$
$$P_2^1(t) = (1-t)P_2^0+tP_3^0$$

$$P_0^2(t) = (1-t)P_0^1+tP_1^1$$
$$P_1^2(t) = (1-t)P_1^1+tP_2^1$$

$$P_0^3(t) = (1-t)P_0^2+tP_1^2$$

이 되며 t값을 대입함으로써 곡선을 얻을 수 있다.

2-3. 부분 분할 알고리즘 (Subdivision algorithm)

자유곡선상의 점들을 구하는 기본적인 알고리즘으로 두점 사이에서 양 끝점과 일정한 비를 갖는 점들을 구해 나가면서 곡선를 그리는 방법이다. 즉 주어진 세 점 P_0, P_1, P_2가 있을 경우 P_0와 P_1 사이에서 구한 점을 P_0^1, P_1, P_2 사이에서 구한 점을 P_1^1, 그리고 P_0^1, P_1^1사이에서 구한 점을 P_0^2라 하면

$$P_0^1(t) = (1-t)P_0+tP_1$$
$$P_1^1(t) = (1-t)P_1+tP_2$$
$$P_0^2(t) = (1-t)P_0^1(t)+tP_1^1(t)$$
$$= (1-t)^2P_0+2t(1-t)^2P_1+t^2P_2$$

로 표현된다. 여기서 t는 0과 1사이의 값을 갖는다.

구해진 새로운 점(P_0^1, P_1^1, P_0^2)과 주어진 점(P_0)을 같은 방법으로 계속 분할해 나가면 구해진 세점에 의해 표현되는 또 다른 곡선의 제어점(control point)를 얻을 수 있다. 이를 n차 다항식으로 일반화한 것이 de Casteljau 알고리즘이다.

$$P_k^i(t)=(1-t)P_k^{i-1}+tP_{k+1}^{i-1}$$
$$i=1, 2, 3, \cdots, n$$
$$k=0, 1, 2, \cdots, n-i$$
$$P_k^0(t)=P_k$$
$$0 \le t \le 1$$

여기서 양 끝점 P_0와 P_{n-1}는 곡선이 반드시 지나는 점이 되고 다른 점들은 모두 P_0와 P_{n-1}점을 지나는 곡선의 모양을 결정하는 제어점이 된다.

n=3일때 de Casteljau 알고리즘을 적용하면

$$P_0^1(t) = (1-t)P_0 + tP_1$$
$$P_1^1(t) = (1-t)P_1 + tP_2$$
$$P_2^1(t) = (1-t)P_2 + tP_3$$
$$P_0^2(t) = (1-t)P_0^1(t) + tP_1^1(t)$$
$$P_1^2(t) = (1-t)P_1^1(t) + tP_2^1(t)$$
$$P_0^3(t) = (1-t)P_0^2(t) + tP_1^2(t)$$
$$= (1-t)^3P_0 + 3(1-t)^2tP_1 + 3(1-t)t^2P_2 + t^3P_3$$
$$0 \le t \le 1$$

여기서 t가 1/2이면

$$P_0^1(1/2) = (P_0+P_1)/2 \quad \cdots\cdots\cdots (2.1)$$
$$P_1^1(1/2) = (P_1+P_2)/2 \quad \cdots\cdots\cdots (2.2)$$
$$P_2^1(1/2) = (P_2+P_3)/2 \quad \cdots\cdots\cdots (2.3)$$
$$P_0^2(1/2) = (P_0^1+P_1^1)/2 = (P_0+2P_1+P_2)/4 \quad \cdots\cdots (2.4)$$
$$P_1^2(1/2) = (P_1^1+P_2^1)/2 = (P_1+2P_2+P_3)/4 \quad \cdots\cdots (2.5)$$
$$P_0^3(1/2) = (P_0^2+P_1^2)/2 = (P_0+3P_1+3P_2+P_3)/8 \quad (2.6)$$

와 같이 표현된다. 이를 그림으로 나타내면 그림 1과 같다. 아래에서는 $P_0^1, P_1^1, P_2^1, P_0^2, P_1^2, P_0^3$을 X축에서는 $XQ_0, XQ_1, XQ_2, XR_0, XR_1$, XS로, Y축에 대해서는 $YQ_0, YQ_1, YQ_2, YR_0, YR_1$,YS로 병행하여 표기한다.

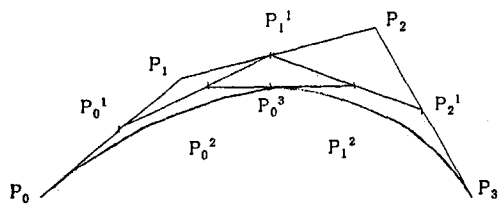

Fig. 1 Subdivision algorithm.

3. Hardware의 구성

3-1. 전체적인 구성

문자 발생 가속기 회로는, 부분 분할 알고리즘을 이용한 3차 Bezier 곡선을 그리는 방법에 기초하여 설계되었으며, 회로의 입력 데이타는 Bezier 곡선을 나타내기 위한 곡선의 시작점, 곡선의 끝점 그리고 이 두 점에서의 기울기를 나타내는 두개의 제어점의 x, y좌표값이 된다. 이들 입력 데이타로부터 얻을 수 있는 문자 발생 가속기의 출력은 메모리상의 점행렬 이미지가 된다. 하드웨어는 크게 4부분으로 구성된다. 평면상의 X축에 대한 좌표값을 계산하는 부분(X-axis calculation circuit), Y축에 대해 좌표값을 계산하는 부분(Y-axis calculation circuit), 그리고 이들 출력값으로부터 점행렬 이미지를 생성하기 위한 이미지 생성부(image generator) 또 이를 전체의 흐름을 제어하기 위한 제어부분(control circuit)으로 구성된다. X축 계산부분과 Y축 계산부분은 실제로 같은 기능을 수행하므로 하드웨어의 구성도 같다. 이 연산 장치(calculation circuit)는 4개의 데이타로부터 윤곽선을 형성할 모든 점을 계산한다.

전체적인 흐름을 살펴보면 그림 2와 같다. 네개의 Bezier 점의 X, Y좌표값이 주어지면, 먼저 첫 점(XP_0, YP_0)를 처리하게 된다. 이는 계산 과정에 얻어진 값에 의해서는 양 끝점이 빠지는 경우가 발생할 수 있으므로 이를 방지하기 위해서이다. X축 연산장치와 Y축 연산장치는 같은 방법에 의해 처리되므로 X축에 대해서만 설명한다. 주어진 네점 XP_0, XP_1, XP_2, XP_3을 레지스터 XP0, XP1, XP2, XP3에 저장하고 add와 shift연산으로 식(2.1), (2.2), (2.3)에서 XQ_0, XQ_1, XQ_2을 먼저 구한다. 다시 이 값으로 부터 식(2.4), (2.5)에 해당하는 값 XR_0, XR_1를 구하고, 이 두 값으로부터 식 (2.6)에서 XS을 구한다. Y축값에 대해서도 같은 방법으로 연산을 수행하여 YS를 구한다. 여기서 구해진 X, Y

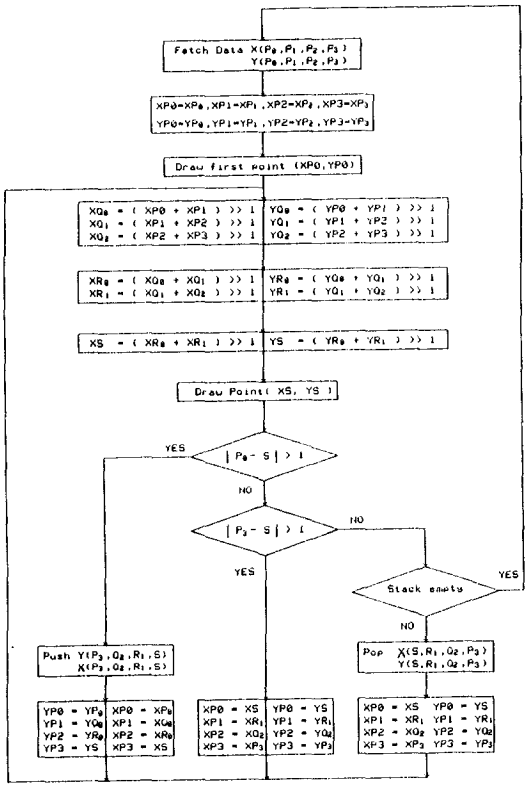

Fig. 2 System calculation flow diagram.

축의 두값, XS, YS가 바로 Bezier 곡선상의 한점이 된다. 이 점을 점행렬 데이타로 변환하여 이미지 메모리에 저장한다. 그런 다음 $|XP_0-XS|>1$이거나 $|YP_0-YS|>1$이면, X축에서 가지고 있는 일곱개의 데이타 XP_0, XQ_0, XR_0, XS, XR_1, XQ_2, XP_3와 Y축에서 가지고 있는 일곱개의 데이타 YP_0, YQ_0, YR_0, YS, YR_1, YQ_2, YP_3 중 여덟개의 데이타 YP_3, YQ_2, YR_1, YS, XP_3, XQ_2, XR_1, XS를 순서대로 스택 메모리에 넣는다. 여기서 나머지 네점 XP_0, XQ_0, XR_0, XS값을 XP0, XP1, XP2, XP3에 넣고 부분 분할을 계속한다. $|XP_0-XS|\leq 1$이고 $|YP_0-YS|\leq 1$이면, $|XP_3-XS|$와 $|YP_3-YS|$에 따라 분기하게 되는데, $|XP_3-XS|>1$이거나 $|XP_3-XS|>1$이면 XS, XP_1^2, XP_2^1, XP_3 값을 XP0, XP1, XP2, XP3에 넣고 (Y축에 대해서도 같은 처리) 부분분할을 계속한다. $|XP_3-XS|\leq 1$이고 $|YP_3-YS|\leq 1$이면 스택메모리가 비어있는지 여부를 검사하여 분기하게 된다. 이때 스택이 비어있지 않으면 연산이 끝나지 않은 상태이므로 스택에 있는 데이타를 XS, XR_1, XQ_2, XP_3, YS, YR_1, YQ_2, YP_3의 순서로 가져와서 변수 XP0. XP1, XP2, XP3, YP0, YP1, YP2, YP3에 넣고 부분분할을 계속하며, 스택이 비었으면 주어진 하나의 Bezier 곡선에 대해 모든 연산이 끝난 상태이므로 새로운 데이타를 데이타 메모리로부터 레지스터로 가져와 다음 연산을 계속하게 된다.

3-2. 연산 장치(calculation circuit)

초기화시 외부로부터 들어오는 Bezier점의 좌표값과 계산 도중에 얻어지는 내부 데이타를 피드백(feedback)시킨 값을 선택적으로 저장할 수 있도록 데이타 선택기가 있는 4개의 데이타 레지스터(Data Register)와 연산을 수행하는 덧셈기(Adder)들로 구성된다. 2로 나누는 Shift 동작은 앞단 덧셈기의 출력선을 다음단 덧셈기의 입력에 1 Bit씩 어긋나게 연결함으로 대신하였다. 연산 장치는 그림 3과 같다.

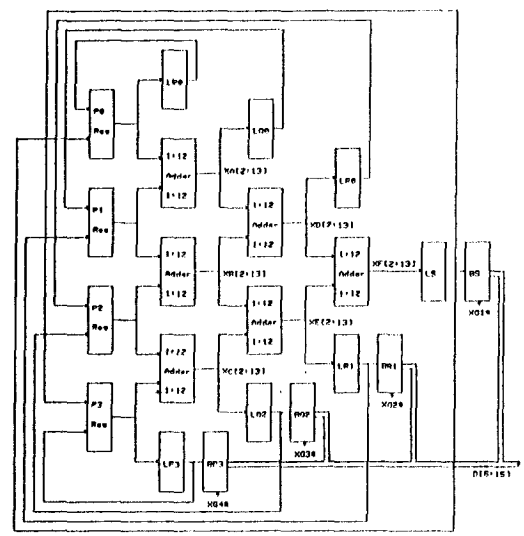

Fig. 3 Calculation part.

주어진 데이타에 대해 한번의 연산이 끝난 후 얻어진 결과 데이타 LP0, LQ0, LR0, LS, LR1, LQ2, LP3에 동시에 래치(latch)되어 다음 연산시 사용할 수 있게 하였다. 각각의 래치(latch)에는 다음과 같은 값이 저장된다.

$$LP0=P0$$
$$LQ0=(P0+P1)\gg 1$$
$$LR0=((P0+P1)\gg 1+(P1+P2)\gg 1)\gg 1=(LQ0+(P1+P2)\gg 1)\gg 1$$
$$LS=(LR0+LR1)\gg 1$$
$$LR1=((P1+P2)\gg 1+(P2+P3)\gg 1)\gg 1=((P1+P2)\gg 1+LQ2)\gg 1$$
$$LQ2=(P2+P3)\gg 1$$
$$LP3=P3$$

여기서 ≫는 Shift right를 나타낸다.

LP0. LQ0, LR0, LS에 대해 다시 연산을 하는 동안 LS, LR1, LQ2, LP3는 다음에 사용할 수 있도록 스택(stack)에 저장할 필요가 있다. 그러려면 외부 메모리와 입출력이 필요하게 되며 여기서 데이타의 충돌을 방지하기 위해 버퍼(buffer)를 삽입하였다.

연산후 비교기(Comparator)에서 발생하는 신호 LG가 high면 스택 어드레스를 감소시키면서 래치된 데이타 LP3, LQ2, LR1, LS를 외부 스택에 저장하고, LP0, LQ0, LR0, LS는 P0, P1, P2, P3로 들어가 다음 연산을 계속하게 된다. LG는 low이고 RG가 high면 LS, LR1, LQ2. LP3의 값이 각각 P0, P1, P2, P3로 들어가 연산이 이루어지며, LG와 RG 그리고 SE 모두 low이면 스택 어드레스를 증가시키면서 스택으로부터 LS, LR1, LQ2, LP3를 각각 P0, P1, P2, P3에 넣고 연산을 계속한다. LG와 RG가 low이고 SE가 high면 하나의 곡선에 대한 연산이 끝나게 된다.

비교기는 그림 4에서 나타나 있듯이 곡선을 나타내는 양 끝점의 값과 연산후 얻은 값을 비교하여 다음의 동작을 결정하는데 사용

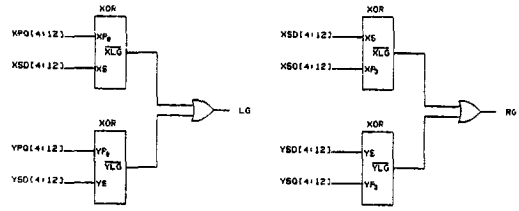

Fig. 4 Comparator.

된다. 비교기는 데이타의 정수 부분중 최하위 bit를 제외한 나머지 부분을 배타적 논리합(exclusive OR)하여 두값의 차가 1이상인 경우를 검출해 낸다. 비교기에서 발생하는 신호를 보면 다음과 같다.

LQ=1 : $|XP_0-XS|>1$ or $|YP_0-YS|>1$
0 : otherwise

RG=1 : $|XP_3-XS|>1$ or $|YP_3-YS|>1$
0 : otherwise

SE=1 : 스택이 비었을 경우(스택의 최상위 어드레스일때 발생)
0 : 스택이 안 비었을 경우

LD=1 : Data pointer의 값 = End Data pointer의 값
0 : otherwise

3-3. 점행렬 이미지 생성기의 하드웨어

두개의 연산 장치로부터 계산된 XS, YS는 각각 평면상의 한점의 X, Y좌표값을 나타낸다. 연산에서 얻어진 결과 XS, YS에서 소수점 이하 부분을 제외한 나머지 부분은 이미지 메모리의 어드레스와 데이타로 분리할 수 있다. 여기에서 사용되는 XS, YS는 최소한 한

번의 연산을 거친 후의 값이 된다. 이로 인해 곡선의 시작점과 끝점을 빠뜨리는 경우가 생길 수 있으므로 이를 방지하기 위해 curve의 시작점이나 끝점을 연산(operation)이 시작하거나 끝날 때 찍어 주어야 한다. 이 회로는 곡선의 시작점을 찍게 했으며, 이를 위해 곡선의 시작점 P0와 연산의 결과 S를 multiplexing하여 이미지 생성기에 연결하였다. 계산된 XS와 YS값은 기존 메모리의 데이타를 읽은 후 이 값과 논리합(OR)하여 메모리에 써넣음으로써 원래의 데이타를 보존한다.
(그림 5참조)

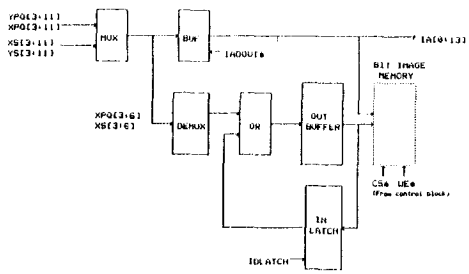

Fig. 5 Image generator.

3-4. 제어부(Control block)

Host Processor가 한 글자분의 Bezier 점들의 데이타를 메모리에 써 넣고, 그 데이타의 마지막 어드레스를 Controller의 EDP (End Data pointer)에 써넣은후 연산 시작(Operation Start)명령을 내리면 이를 받아 연산을 수행하고, 계산이 끝나면 호스트(Host)에 인터럽트를 걸어 알려준다.

제어 장치의 구성은 다음과 같다.

3-4-1. 레지스터 블록(Register block)

• 마지막 데이타 포인터(End Data pointer)

연산시 사용할 데이타의 마지막 어드레스를 보관하여 데이타 포인터(Data pointer)의 값과 비교하는데 사용된다. 두 값이 같으면 LD 신호가 high가 되어 한 글자에 대한 처리가 끝났음을 알려준다.

• 상태 레지스터(Status Register)

연산이 끝난후 문자 발생 가속기 내부의 상태(status)를 보관한다.

• 명령 레지스터(Command Register)

호스트로 부터 명령(command)을 받아 저장한다.

3-4-2. 주소발생기(Address generator)

• 데이타 포인터(Data pointer)

연산이 시작할때 클리어(clear)되며 데이타를 데이타 메모리로부터 가지고 올때 마다 하나씩 증가하며 다음에 가지고 올 데이타의 위치를 가르킨다. Data end register와 비교하여 LD 신호를 만드는데에도 사용된다.

• 스택 포인터(Stack pointer)

계산 도중에 생기는 데이타를 스택 메모리에 보관할때 저장될 위치를 나타낸다. Push할 경우 데이타를 저장한후 어드레스를 감소시키며, Pop할 경우 어드레스를 증가시킨후 데이타를 가지고 온다.

스택 포인터가 스택의 최상위 어드레스를 가리킬때 Stack Empty 신호 SE가 high가 되어 스택이 비었음을 알린다.

• 어드레스 선택기(Address selector)

데이타 어드레스와 스택 어드레스를 상태에 따라 선택적으로 외부 메모리의 어드레스 선(Address line)으로 보내준다. Push나 pop을 할 경우 스택 어드레스가 어드레스선상에 나타나게 되고 데이타 호출시는 어드레스 선상에 Bezier 곡선의 데이타 어드레스가 나타나게 된다.

3-4-3. 순차기

X, Y 연산 장치로 부터 온 LG, RG 신호와 호스트로부터의 명령 그리고 상태 플래그(status flag)의 값을 입력으로 하여 회로의 동작에 필요한 제어신호를 생성하여 시스템이 효율적으로 연산을 수행할 수 있게 해 준다.

시스템에 필요한 기본적인 제어신호을 발생시키는데 사용하기 위해 5bits의 ring counter를 사용하였다. 이 ring counter는 기본적인 contol signal생성과 전체 회로의 동기를 맞추는데 사용된다. basic CSG(control signal generator)는 회로의 동작에 가장 기본이 되는 OPLTCH, IDLTCH, ADOUT*, RD*, DATAOUT, WR*, LXYP*등과 같은 신호를 만들며, 이 신호들은 state register의 출력 신호와 함께 CSG로 들어가 SADOUT, SRD*, SWR*, IDLATCH, IDOUT*등과 같은 신호를 생성한다.

그림 6와 같이 COMMAND READ 상태 S0에서 MEMORY CLEAR 명령인 CM이 들어오면 S12로 가서 이미지 메모리(image memory)의 내용을 모두 clear한다. 메모리가 모두 clear된 후 8KCZ신호가 high가 되면 INTERRUPT 상태인 S13에 들어가서 INT 신호를 내보낸 후 다시 COMMAND READ state로 간다. COMMAND READ 상태 S0에서 OPERATION START 명령인 OS신호가 들어오면 S1 상태로 넘어가 데이타 포인터의 값을 증가시키면서 4개의 Bezier점 데이타를 가지고 온다. 이미 그려진 이미지가 있을 경우 그 데이타를 보존하면서, 곡선의 시작점인 P_0를 이미지 메모리에 그리기 위해 이미지 메모리의 내용을 읽어온 후(S2 state) 다시 논리합(OR)한 값을 써 넣는다(S3 state). 이어서 CALCULATION 상태로 들어가 연산을 하게 되며, 여기에서 나온 값 S를 다시 이미지 메모리에 써넣게 된다.(S5, S6 state)

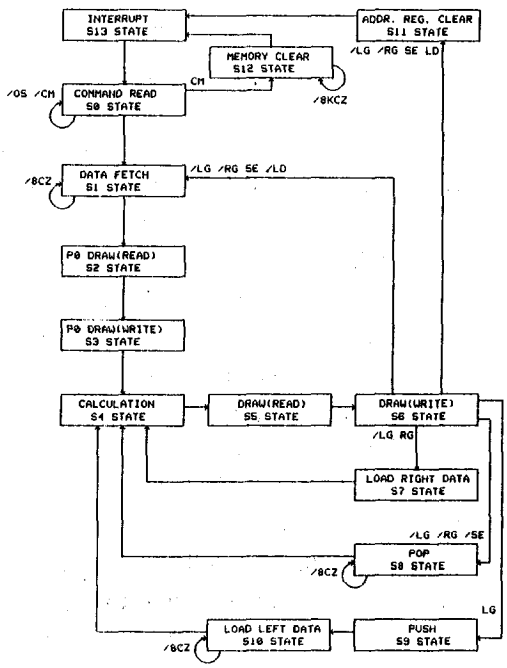

Fig. 6 State flow diagram.

S6 상태에서는 S4 상태에서 발생한 LG, RG신호와 스택 포인터에서 발생하는 SE (stack empty), 데이타 포인터와 EDP(End data pointer)를 비교하여 생성되는 LD(last data) 신호에 따라 다음 상태로 분기하게 된다. 입력신호에 따른 분기와 동작은 아래 표에 나타내었다. 이곳에서 사용된 8KCZ신호는 이미지 메모리 clear시 메모리가 모두 clear되었을 때 발생하는 신호이며, 8CZ는 DATA FETCH, PUSH, POP 상태와 같이 8번의 연속동작이 행해지는 상태에서 8진 카운터(counter)를 사용하여 동작할 때 발생하는 신호다.

제어부(control block)는 Xilink사의 PGA (Programmable gate array)인 XC3030 (3000 gate)과 ALTERA사의 EPLD(Erasable Programmable Logic Device)소자인 EP1810과 TTL로 구성되었으며, 다른 block들은 TTL을 사용하여 구현하였다.

3-4-4. 메모리의 구성

메모리는 사용 용도에 따라 3가지로 분류된다. 문자 이미지 생성에 필요한 곡선 데이타를 저장하는 데이타 메모리와 연산중에 발생하는 데이타를 일시적으로 보관하기 위한 스택 메모리, 그리고 연산에 의해 얻어진 문자의 bit-image를 저장하는 이미지 메모리가 있다. 이 시스템에서 이미지 메모리, 스택 메

모리, 데이타 메모리의 크기는 각각 16K Words, 8K Words, 8K Words씩 할당하였다.

4. 결 론

본 연구에서는 Bezier 곡선 알고리즘에서 부분분할 방법을 이용하여 윤곽선문자를 그리는 시스템을 설계하였다. 설계한 시스템은 TTL을 이용하여 연산부를 구성하였고, 제어회로는 PGA(Programmable Gate Array)인 XC3030(3000 gate, 100 CLBs, 80 IOBs, 360 Flip-flops, 84 Pin PLCC type)과 EPLD(Erasable Programmable Logic Device)인 EP1810J(48 Macrocells, 48 Registers, 16 Buried Registers, 16 Inputs, 48 I/O Pin, 68 Pin JLCC type)을 이용하여 구현하였다.

이 회로는 ORCAD/VST를 사용하여 시뮬레이션(simulation)하였으며, 결과로 원하는 제어 신호들을 얻을 수 있었다. 이 윤곽선 문자생성 시스템은 고속으로 Bezier 곡선의 점행렬 이미지를 생성해 낼 수 있으므로 고해상도의 DTP(Desk Top Publishing) 시스템이나 전자출판 시스템에 적합하리라 생각된다. Bezier 곡선 알고리즘중 본 논문에서 사용한 부분 분할은 부가적인 스택 메모리가 필요하다는 단점이 있지만 특성상 빠른 연산을 할 수 있다는 장점을 가지고 있다. 그림 7은 윤곽선 문자 발생기에 의해 생성된 글자를 보여준다.

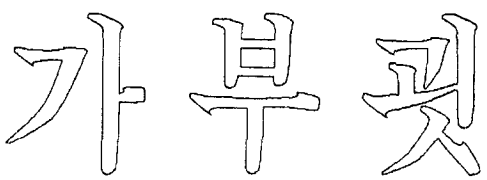

Fig. 7 Generated outline font characters.

참 고 문 헌

1. J.D.Foley, and A.VAN DAM, "Fundamentals of Interactive Computer Graphics", Addison Wesley, July 1984.
2. Cornel K.Pokorny, Curtis F.Gerald, "COMPUTER GRAPHICS : THE PRINCIPLES BEHIND THE ART AND SCIENCE", Franklin, Bredle & Associates, 1989.
3. 황규철, "전자출판 시스템에서 사용되는 고해상도 문자의 발생을 가속시키기 위한 하드웨어의 설계 및 제작", 한국과학기술원 전기 및 전자공학과 석사논문, 1989.
4. 이태형, "외곽선 문자의 발생을 가속시키기 위한 VLSI칩의 설계 및 구현", 한국과학기술원 전기 및 전자공학과 석사논문, 1990.
5. Ph. Coueignoux, "Character Generation by Computer", Computer Graphics and Image Processing, Vol. 16, 1981, pp. 240~269.
6. George Merrill Chaikin, "An Algorithm for High-Speed Curve Genertion", Computer Graphics and Image Processing, Vol. 3, 1974, pp. 346~349.
7. R.A. Ernshaw, "A Review of Curve Drawing Algorithm", NATO ASI Series, Vol. 17, 1983.
8. T.N.T goodman and K.Unsworth, "Generation of spline curves using a recurrence relation", NATO ASI Series, Vol. 17, 1983.
9. Texas Instruments inc, "TMS34010 Application Guide", Texas Instruments.
10. Texas Instruments inc, "TMS34010 User's Guide", Texas Instruments.
11. Xilinx inc, "The Programmable Gate Array Data Book", Xilink.
12. ALTERA co., "USER-CONFIGURA-

BLE LOGIC Application Handbook", ALTERA.
13. ALTERA co., "A+PLUS User Guide", ALTERA.
14. ALTERA co., "Logi Caps", ALTERA
15. ALTERA co., "ADLIB", ALTERA.