(19)대한민국특허청(KR)
(12) 등록특허공보(B1)

(51) 。Int. Cl. 6
G06F 15/18

(45) 공고일자 2003년05월17일
(11) 등록번호 10-0372968
(24) 등록일자 2003년02월07일

(21) 출원번호 10-1995-0054210
(22) 출원일자 1995년12월22일

(65) 공개번호 특1996-0025145
(43) 공개일자 1996년07월20일

(30) 우선권주장 363196 1994년12월22일 미국(US)

(73) 특허권자 모토로라 인코포레이티드
미국, 일리노이 60196, 샤움버그, 이스트 앨공퀸 로드 1303

(72) 발명자 제임스 그레그 바이옷
미국, 텍사스 78750, 오스틴 레익우드 드라이브 7816

(74) 대리인 이병호

심사관 : 마정윤

(54) 효율적인퍼지논리연산을수행하기위한데이터처리시스템및그방법

요약

데이터 처리 시스템(10)은 퍼지 논리 멤버쉽 함수에서 입력값의 멤버쉽 정도를 결정하는 명령 'MEM'으로 공지된 명령을 포함한다. 또한 MEM 명령에 응답하여, 데이터 처리 시스템(10)은 멤버쉽 함수의 경계를 한정하는 경계치가 어떤 선정된 값을 가지는지를 검사한다. 만약 가진다면, 상기 멤버쉽 함수는 멤버쉽 함수들 세트의 마지막 멤버쉽 함수이며 데이터 처리 시스템(10)은 종료 신호(151)를 발생시킨다. 한가지 구현된 예에서, 종료 신호(151)는 조건 코드 레지스터(69)에 비트를 설정하여 멤버쉽 함수들 세트의 연속적인 멤버쉽 함수들에서 입력값의 멤버쉽 정도를 결정해 주는 소프트웨어 프로그램 루프(1oop)가 있을 수 있도록 한다.

대표도

도 3

명세서

도면의 간단한 설명

제 1도는 퍼지 논리의 개념 및 기본 용어를 설명하기 위해 사용된 도면.
제 2도는 제 1도의 멤버쉽 세트를 보다 상세히 설명하기 위한 도면.
제 3도는 본 발명에 따른 데이터 처리 시스템의 블록도.
제 4도는 제 3도의 실행 유닛의 블륵도.
제 5도는 제 4도의 플래그 발생 논리 회로의 논리 회로도.
제 6도는 제 4도의 오퍼랜드 할당 논리 회로의 블록도.
제 7도는 제 4도의 산술 논리 유닛(ALU, arithmetic logic unit)의 블록도.
제 8도는 본 명세서에 설명된 본 발명에 따른 MEM 명령의 실행동안 수행되는 함수의 흐름을 나타내는 흐름도.
* 도면의 주요부분에 대한 부호의 설명

10 : 데이터 처리시스템 12 : CPU
14 : 실행 유닛 16 : 버스 제어 논리 회로
18 : 명령 복호화 논리 회로 20 : 제어 유닛
22 : 시퀀서 24 : 발진기
26 : 전력회로 28 : 타이밍 회로
30 : 외부 버스 인터페이스 32 : 내부 메모리
33 : 외부 데이터 버스 35 : 외부 어드레스 버스
34 : 내부 데이터 버스 36 : 내부 어드레스 버스
38 : 타이밍 제어 버스 40 : 외부정보 버스
42 : 데이터 버퍼 46 : 결과 선택 논리 회로
48 : 플래그 발생 논리 회로 50 : 오퍼랜드 할당 논리 회로
52 : A-입력 제어 논리 회로 54 : A-입력 멀티플렉서
56 : ALU 58 : B-입력 멀티플렉서
60 : 델타 레지스터 64 : 시스템 입력 레지스터
66 : 정보 버스 A 68 : 정보 버스 B
69 : 조건 코드 레지스터 70 : 플래그 버스
72, 74, 76, 78 : 멀티플렉서 90: 버퍼
80, 82, 84, 86, 88, 92, 94, 96, 98, 100 : 덧셈기
102 : 하이 캐리 룩어헤드 논리 회로(high carry lookahead logic)
104 : 로우 캐리 룩어헤드 논리 회로(low carry lookahead logic)
106 : 하이 상태 플래그 회로(high status flag circuitry)
108 : 로우 상태 플래그 회로(low status flag circuitry)
120 : 결과버스 130, 136 : 래치
110, 112, 116, 118, 140, 142, 146, 148 : NOR 게이트
114, 144, 150 : OR 게이트
132, 134, 138 : AND게이트

발명의 상세한 설명

발명의 목적

발명이 속하는 기술 및 그 분야의 종래기술

발명의 분야
본 발명은 일반적으로 데이터 처리 시스템에 관한 것으로, 특히, 퍼지 논리 연산을 수행하기 위한 데이터 처리 시스템에 관한 것이다.
발명의 배경
데이터 프로세서들은 그의 입력과 출력이 '0' 이나 '1'로서만 해석되고 다른 가능성은 존재하지 않는 2진 기계로서 기능하도록 발달되어 왔다. 이것은 대부분의 상황에서 적합할 수 있으나 어떤 때에는 응답이 'yes' 나 'no'가 아닌 그 사이의 어떤 것일 수 있다. 퍼지 논리로 불려진 이러한 개념이 2진 논리에 기초한 데이터 프로세서가 'yes' 및 'no' 사이의 응답을 제공하도록 발달되어 왔다.
퍼지 논리는 퍼지 경계(fuzzy boundary)를 갖는 멤버쉽 함수로 된 논리 시스템이다. 멤버쉽 함수는 '온도가 따뜻하다.'와 같은 주관적인 표현들을 특정 데이터 프로세서가 인식할 수 있는 값으로 변환시켜 준다. '따뜻함'과 같은 표시는 그 경계들이 한쪽에서는 진실이고 다른 한쪽에서는 거짓인 지점이 아닌 입력값의 범위를 나타내기 위해 사용된다. 퍼지 논리를 실행하는 시스템에서 멤버쉽 함수의 경계들은 점차적으로 변화하며, 인접한 멤버쉽 세트의 경계들을 중첩시킬 수 있다. 따라서 멤버쉽의 정도는 입력 값에 특정하게 정해진다. 예를 들어, 온도의 일정한 범위 에 걸친 주어진 2개의 멤버쉽 함수에서, 입력 온도는 '서늘함' 과 '따뜻함'으로 표시된 함수의 중첩 지역에 해당될 수 있다. 다음 처리가 각각의 멤버쉽 함수들에서 멤버쉽의 정도를 정하기 위해 필요하다(즉, 입력 온도가 멤버쉽 세트들, 즉, 서늘함과 따뜻함과 일치하는 경우에).
'퍼지화(fuzzification)'로서 표시된 단계가 퍼지 논리를 실행하는 시스템에서 멤버쉽 함수에 입력을 정하기 위해 사용된다. 퍼지화 처리는 '온도가 따뜻하다.'와 같은 주관적인 표현에 구체적인 숫자의 값을 정해준다. 이러한 숫자의 값은 절대적으로 진실인 환경에 일반적으로 국한되지 않는 인간 감각과 흡사한 근사치를 제공하고자 하는 것이다. 퍼지화 단계 이후에 규칙 평가 단계(rule evaluation step)가 실행된다. 규칙 평가 단계가 실행되는 동안 '최소-최대(min-max)' 추론으로 표시된 기술이 사용자에 의해 정의된 언어 규칙에 숫자적 결론을 계산하기 위해 사용된다. 규칙 평가 단계로부터의 결론들은 '퍼지 출력들(fuzzy outputs)'로 불려지며 다양한 정도에서 사실일 수 있다. 따라서, 경합하는 결과들이 얻어질 수 있다. 퍼지 논리 처리에서 마지막 단계는 '디퍼지피케이션'으로서 언급된다. 명칭이 의미하는 것과 같이, 디퍼지피케이션은 모든 퍼지 출력들을 일반적인 데이터 처리 시스템에 적용될 수 있는 합성 결과로 결합시키는

처리이다. 퍼지 논리에 대한 구체적인 정보는 제임스 엠. 시비트로스(James M. Sibigtroth)에 의한 '하드웨어에서 퍼지 전문가 규칙의 실행(Implementing Fuzzy Expert Rules in Hardware)'이라 표시된 기사에 제시되어 있다. 이 기사는 1992년 4월에 발간되었고 페이지 25에서 31 까지에 제시되어 있다. 또한 본 발명의 양수인에게 양수된 '퍼지 논리 연산 동안 세트에서 멤버쉽을 결정하기 위한 방법 및 회로(Circuit and Method for Determining Membership in a Set During a Fuzzy Logic Operation)'로 명명되고 1994년 3월 15일에 발간된 미국 특허 제 5,295,229 호에도 언급되어 있다.

일반적으로, 퍼지화 단계는 주변기기나 데이터 프로세서에 의해 실행되는 소프트웨어 프로그램과 함께 실행된다. 이와 같은 소프트웨어 프로그램은 상당히 많은 처리시간이 요구되는 수학 방정식을 사용하여 멤버쉽의 정도를 계산한다. 따라서 소프트웨어 프로그램이 정확한 결과를 제공할지라도, 일반적으로 처리 시간이 이러한 시스템의 수행 능력을 제한한다. 다른 일반적인 소프트웨어를 이용한 해결 방법들은 특정 입력값에 해당하는 값을 위해 메모리에 저장된 데이터 테이블에서 액세스 하는 '테이블 록-업(table look-up)' 루틴을 실행한다. 수학적 계산보다 빠를지라도, 테이블 록-업 루틴을 실행하기 위해 상당히 많은 메모리를 필요로 한다. 퍼지화를 위해 하드웨어를 사용한 해결책은 보다 빠른 결과를 제공한다. 그러나, 이러한 해결책은 어쩔 수 없이 상당히 많은 양의 회로를 필요로 한다. 퍼지화를 위한 하드웨어를 사용한 해결책이 소프트웨어 실행보다 상당히 빠를지라도, 하드웨어를 사용한 해결책은 상당한 크기의 회로를 위한 회로 면적을 필요로 하며 종종 비용이 많이 들어 데이터 처리 시스템 상에서 실행할 수 없는 문제점이 있다.

따라서, 이러한 문제점을 해결하기 위해 하드웨어적으로 비용이 적게들면서 퍼지화 단계가 빨리 실행되는 회로 및 방법이 필요하게 되었다. 일반적으로, 하드웨어를 이용한 해결책과 연관된 속도는 대개 소프트웨어를 이용한 해결책과 연관된 메모리를 사용하지 않는 것이 필요하다. 이러한 필요성에 의해 본 발명이 제시된 것이며, 이러한 특징 및 이점은 도면과 함께 설명된 하기에서 보다 분명하게 이해될 것이다. 그러나 본원에서 제시된 도면이 본 발명의 유일한 형태를 나타내는 것이 아니고 단지 한가지 실시예를 나타내는 것을 주목할 필요가 있다.

바람직한 실시예의 상세한 설명

본 발명은 데이터 처리 시스템에서 빠르고 더 적은 회로 규모를 가지면서 입력에 따라 일련의 퍼지 연산을 수행하는 회로 및 그 방법에 관한 것이다. 본원에서는 데이터 처리 시스템의 사용자가 길고 어려운 소프트웨어 프로그램을 사용하지 않고서도 보다 빠르게 퍼지화 단계를 수행하도록 해주는 명령(instruction)이 제공된다. 하나의 입력 라벨(label)의 퍼지화 단계는 데이터 처리 시스템에서 부가되는 과다한 양의 회로가 없이도 보다 빠르고 효율적으로 실행하는 명령에 인코딩된다. 특정 시스템 입력을 위한 것들과 같은 일련의 퍼지화 연산의 목적은 카운터나 그와 같은 것이 없이도 쉽게 검출되도록 하기 위한 것이다.

본 발명 구현의 상세한 설명에 있어서 '어서트(assert)' 및 '니게이트(negate)'와 다양한 형태의 용어들은 '액티브 하이(active high)' 나 '액티브 로우(active low)'와 같은 논리 신호의 혼합을 처리할 때 혼동을 피하기 위해 사용된다. '어서트'는 논리 신호나 레지스터 비트를 액티브 상태 즉 논리적으로 진실(true)인 상태로 만들기 위해 사용된다. '니게이트'는 논리 신호나 레지스터 비트를 인액티브 상태 즉 논리적으로 거짓인 상테로 만들기 위해 사용된다. 또한, 16진수 값은 값 앞에 '$' 표시를 하여 나타낸다.

제 1도는 퍼지 논리의 몇 가지 기본 개념과 용어들을 나타내기 위한 도면이다. 제 1도에서 '시스템 입력(system input)'은 온도를 나타낸다. 5개의 '멤버쉽 함수들(membership function)'이 온도의 범위를 나타내기 위하여 제공된다. 예를 들어, 0도 내지 30도의 온도는 '추움(cold)'으로 표시되어 있다. 유사하게, 20도 내지 50도의 온도는 '서늘함(cool)'으로 표시되어 있다. 각각의 멤버쉽 세트의 경계(boundary)는 인접한 세트들의 경계들과 중첩한다는 것을 주의해야 한다. 따라서, 하나의 온도는 하나 이상의 세트들에 포함될 수 있다. 예를 들어, 시스템 입력이 68도의 온도라면 제 1도에서 68도는 따뜻함(warm)과 매우 따뜻함(very warm) 멤버쉽 세트의 경계 내에 존재하게 되며 각각의 멤버쉽 세트에서 멤버쉽이 정도값은 0이 아닌 값이 얻어진다. 시스템 입력이 따뜻함 멤버쉽 세트에서는 $33의 멤버쉽 정도를 나타내며 매우따뜻함 멤버쉽 세트에서는 $CC의 멤버쉽 정도를 나타낸다. 시스템 입력이 75도의 온도를 나타내면 이 온도는 매우 따뜻함 멤버쉽 세트에서 $FF의 멤버쉽 정도를 나타내며, 더움(hot) 멤버쉽 세트에서는 $OO의 멤버쉽의 정도를 가진다. 제 1도에서 멤버쉽의 정도는 16진수값 $OO 내지 $FF의 범위이며 이것은 0.00 내지 0.996의 범위에서 일부에 해당될 수 있다.

제 2도는 4개의 값들(포인트 1, 포인트 2, 슬로프 1, 슬로프 2)에 의해 표현되는 하나의 멤버쉽 세트를 나타낸다. '델타 1(Delta 1)'로서 표시된 제 1값은 시스템 입력(포인트 'X')과 포인트 1 사이의 차이로서 계산된다. 유사하게 '델타 2(Delta 2)'로서 표시된 제 2입력은 포인트 2와 시스템 입력 사이의 차이로서 계산된다. 델타 1과 델타 2 값의 계산 결과로부터 멤버쉽의 정도는 본 발명의 방법에 따라 제공된다. 만약 델타 1이나 델타 2의 값이 0 보다 적다면 해당하는 멤버쉽 세트의 멤버쉽 정도는 0.00이다. 그렇지 않다면 멤버쉽의 정도는 델타 1과 슬로프 1의 프로덕트(product)나 델타 2와 슬로프 2의 프로덕트의 최소값 즉 포화치(saturation value)에 해당한다. 최소값은 검사되어지는 멤버쉽 세트에서 현재의 입력값의 멤버쉽의 정도일 것이다. 이러한 예에서 포화치는 16진수 값 $FF이다. 그러나 다른 실시예에서는 퍼지 논리 시스템의 사용자가 특정한 시스템의 규격에 따라 포화치를 선택하도록 허용 할 수 있다.

의사 기호 라벨 'MEM'으로 표현된 본 발명의 명령은 여러 개의 멤버쉽 세트 중 하나에서 시스템 입력값의 멤버쉽의 정도를 결정하도록 하는 일련의 단계들을 수행하도록 한다. 또한 MEM 명령은 '종료(termination)' 신호가 보내지며 기록되게 할 수 있다. 각각의 멤버쉽 세트는 메모리로부터 복구되는 다수개의 경계치(포인트 1, 포인트 2, 슬로프 1, 슬로프 2)에 의해 특징지어 진다. 다수개의 경계치 각각은 메모리에 적절히 저장되고 시스템 입력값이 MEM 명령의 실행에 앞서 레지스터에 적절히 저장된다면, 여러 개의 멤버쉽 세트의 하나에서 시스템 입력의 멤버쉽의 정도는 결정될 수 있다. 그 후에 여려 개의 멤버쉽 세트의 각각에서 시스템 입력의 멤버쉽의 정도는 메모리 상에 미리 지정된 위치에 저장된다. MEM 명령의 실행은 데이터 처리 시스템의 사용자가 멤버쉽 평가 함수를 종래의 소프트웨어를 사용

한 실행보다 더 빠르게 그리고 하드웨어를 사용한 실행에 의해 얻어지는 것보다 더 적은 회로 면적을 가지면서 수행할 수 있도록 해준다. 예를 들어, 데이터 처리 시스템의 일반적인 연산에서 얻어지는 산술 논리 유닛(arithmetic logic unit)은 연산의 분 할 모드(split mode)에서 동작하도록 수정된다. 일반적으로, 산술 논리 유닛은 지정된 지점에서 시간 내에 하나의 16비트 연산을 수행할 수 있다. 그러나 본원에서 개시된 산술 논리 유닛은 하나의 16비트 연산이나 동시에 2개의 8비트 연산을 수행할 수 있도록 선택적으로 인에이블될 수 있다. 따라서 멤버쉽 세트의 멤버쉽의 정도를 계산하기 위해 필요한 산술 동작들은 종래의 방법보다 더 짧은 시간 내에 동시에 계산될 수 있다. 이를 위해 필요한 회로는 극히 적다. 산술 논리 유닛이 분할 모드에서 연산을 수행할 수 있게 하기 위해 하나의 인에이블 신호가 버퍼 회로에 제공된다. 부가적으로, 약간의 회로가 MEM 명령을 수행하기 위해 필요하다.

상기의 연산 방법과 명령의 실행이 제 3도에 나타나있다. 제 3도는 퍼지화 명령이 실행되는 데이터 처리 시스템(10)을 도시한다. 데이터 처리 시스템(10)은 일반적으로 중앙 처리 장치(CPU, central processing unit)(12), 발진기(24), 전력 회로(26), 타이밍 회로(28), 외부 버스 인터페이스(30), 및 내부 메모리(32)를 포함한다. CPU(12)는 일반적으로 실행 유닛(14), 버스 제어 논리 회로(16), 명령 디코딩 논리 회로(18), 제어 회로(20), 및 시퀀서(22)를 갖는다.

동작 동안 'Osc 1' 신호는 수정과 같은 외부 소스를 지나 발진기(24)에 제공된다. 수정은 발진할 수 있도록 Osc 1과 Osc 2 사이에 연결된다. Osc 1은 데이터 처리 시스템(10)의 나머지 부분에 '클락(Clock)'신호를 공급한다. 수정 발진기의 동작은 데이터 처리 시스템에서 공지 되어있으며 이 기술 분야에 숙련된 사람은 이러한 것을 잘 알고 있다.

유사하게, 전력 회로(26)는 외부의 전력원으로부터 'Vdd' 및 'Vss' 신호를 수신한다. Vdd 신호는 5볼트를 공급하고 Vss 신호는 기준 전압, 즉, 그라운드 전압을 공급하며, Vdd 및 Vss 신호는 데이터 처리 시스템(10)의 나머지 구성 요소들 각각에 제공된다. 상기 신호들의 라우팅은 데이터 처리 기술에서 잘 공지되어 있으며 이 기술 분야에 숙련된 사람에게는 이것은 분명하다.

타이밍 회로(28)는 클락 신호를 수신한 후에 타이밍 제어버스(38)를 경유하여 CPU(12), 외부 버스 인터페이스(30), 및 내부 메모리(32) 각각에 적당한 타이밍 신호를 공급한다.

다수의 어드레스 값들이 외부 어드레스 버스(35)를 경유하여 외부 버스 인터페이스(30)로부터 제공된다. 유사하게 다수의 데이터 값들이 외부 데이터 버스(33)를 경유하여 외부 버스 인터페이스(30)에 통신된다. 외부 버스 인터페이스(30)는 외부의 사용자와 데이터 처리 시스템(10) 사이에서 어드레스 및 데이터 값들을 통신하도록 해준다. 외부 버스 인터페이스(30)는 각기 내부 어드레스 버스(36)와 내부 데이터 버스(34)를 경유하여 데이터 처리 시스템(10)의 나머지 각 부분에 다수의 어드레스 및 데이터 값들을 제공한다. 내부 메모리(32)는 데이터 처리 시스템(10)의 적절한 동작을 위해 필요한 정보값을 저장하도록 해준다. 부가적으로 다른 데이터 값들은 내부 어드레스 버스(36)와 내부 데이터 버스(34)를 지나 제공되는 사용자 프로그램에서 정해지면 거기에 저장될 수 있다.

CPU(12)는 데이터 처리 시스템(10)의 동작 동안 필요한 각각의 명령들을 실행시킨다. 내부 어드레스 버스(36)와 내부 데이터 버스(34)는 실행 유닛(14)과 데이터 처리 시스템의 나머지 부분 사이에서 상호간에 정보를 교환시킨다. 버스 제어 논리 회로(16)는 명령들과 오퍼랜드들을 패치한다. 각각의 명령들은 명령 디코딩 논리 회로(18)에 의해 디코딩되며 제어 유닛(20)과 시퀀서(22)에 제공된다. 제어 유닛(20)과 시퀀서(22)는 데이터 처리 시스템(10)의 계산 능력을 가장 효율적으로 이용할 수 있도록 각각의 명령들을 연속적으로 실행할 수 있도록 한다. 부가적으로 제어 유닛(20)은 Micro-ROM 메모리(도시되지 않음)를 포함하며, 상기 Micro-ROM 메모리는 제어 버스(65)를 경유하여 실행 유닛(14), 버스 제어 논리(16), 및 명령 디코딩 논리(18) 각각에 다수의 제어 정보를 제공한다.

실행 유닛(14)은 제 4도에 보다 상세히 도시되어 있다. 실행 유닛(14)은 일반적으로 데이터 버퍼(42), 결과 선택 논리 회로(46), 플래그 발생 논리 회로(48), 오퍼랜드 할당 논리 회로(50), A-입력 제어 논리 회로(52), A-입력 멀티플렉서(54), 산술 논리 유닛(ALU)(56), B-입력 멀티플렉서(58), 델타 레지스터(60), 시스템 입력 레지스터(64), 및 조건 코드 레지스터(69)를 포함한다. 도면에는 표시되어 있지 않지만, 이 기술 분야에서 숙달된 사람이라면 실행 유닛(14)에 부가적인 레지스터를 첨가하는 것은 쉽게 가능할 것이다. 예를 들어, 실행 유닛(14)은 인덱스 레지스터 또는 누산기(accumulator)를 포함할 수도 있다. 부가적으로, 전형적인 데이터 처리 시스템에 있어서 시스템 입력레지스터(64)는 누산기로 이용될 수 있다.

외부 정보 버스(40)는 데이터 버퍼(42)에 어드레스 및 데이터 정보를 제공한다. 외부 정보 버스(40)는 내부 어드레스 버스(36)와 내부 데이터 버스(34)로부터 각기 어드레스 및 데이터 정보를 수신한다. 데이터 버퍼(42)는 정보 버스(A66, B68)를 경유하여 실행 유닛(14)의 나머지 부분에 외부 정보 버스(40)를 경유하여 전송되는 값을 제공한다. 본 명세서에서는 상세히 나타내지 않았지만, 정보 버스(A66, B68)는 16비트의 크기이며, 하이 사이드(8비트 내지 15비트)와 로우 사이드(0비트 내지 7비트)로 나누어져 있다.

정보 버스(A66)의 로우 사이드는 시스템 입력 레지스터(64)의 제 1 입력에 양방향으로 연결되어 있다. 유사하게 정보 버스(B68)의 하이 사이드는 시스템 입력 레지스터(64)의 제 2 입력에 양방향으로 연결되어다. '델타 1'로 표시된 제 1 델타값은 정보 버스(B68)의 하이 사이드를 경유하여 델타 레지스터(60)의 제 1 입력에 제공된다. 제 2 델타값 '델타 2'는 정보 버스(B68)의 로우 사이드를 경유하여 델타 레지스터(60)의 제 2입력에 제공된다. 델타 레지스터(60)는 또한 실행 유닛(14)의 다른 구성 요소에 정보를 제공하기 위하여 정보 버스(A66, B68)에 연결된다. 누산기와 인덱스 레지스터와 같은 부가적인 레지스터들은 유사하게 정보버스(A66, B68)에 연결될 수 있다.

정보 버스(A66)는 A-입력 멀티플렉서(54)의 제 1 입력에 연결된다. '제어(Control)'로 표시된 n-비트크기 신호가 A-입력 멀티플렉서(54)의 제 2 입력에 제공되며, 여기서 n은 정수이다. A-입력 멀티플렉서(54)는 ALU(56)의 제 1 입력에 'A-입력 로우 사이드(A-Input Low Side)'로 표시된 제 1 10비트 출력을 제공하며 ALU(56)의 제 2 입력에 'A-입력 하이 사이드(A-Input High Side)'로 표시된 제 2 10비트 출력을 제공한다.

정보 버스(B68)는 B-입력 멀티 플렉서(58)의 입력에 연결된다. B-입력 멀티 플렉서(58)가 바르게 동작하도록 하기 위해 필요한 제어 정보가 Micro-ROM 제어 버스(65)를 지나 전송되며 제어 유닛(20)에서 Micro-ROM에 의해 발생

된 다수의 신호를 경유하여 제어 입력에 제공된다. B-입력 멀티플렉서(58)는 ALU(56)의 제 3 입력에 'B-입력 로우 사이드'로 표시된 제 1 10비트 출력을 제공한다. 부가적으로 B-입력 멀티플렉서(58)는 ALU(56)의 제 4 입력에 'B-입력 하이 사이드'로 표시된 제 2 10비트 출력을 제공한다.

'분할 모드 제어(Split Mode Control)'로 표시된 신호가 ALU(56)의 제 5 입력에 제공된다. 분할 모드 제어 신호는 MEM 명령의 실행에 응답하여 제어 유닛(20)에서 Micro-ROM 메모리에 의해 발생된다. ALU(56)는 결과 버스(120)를 경유하여 전송되는 다수의 결과들을 제공하기 위해 각각의 입력을 처리한다. 결과 버스(120)는 각각의 결과 선택 논리 회로(46)와 플래그 발생 논리 회로(48)에 다수의 정보값들을 제공한다. 부가적으로 결과 버스(120)는 정보 버스(B68)에 연결된다.

결과 버스(120)에 더하여, 플래그 발생 논리 회로(48)는 델타 레지스터(60)와 데이터 버퍼(42)에 연결된다. 플래그 발생 논리 회로(48)는 델타 레지스터(60)로부터 '델타(Delta)' 신호와 데이터 버퍼(42)로부터 '버퍼링된 데이터(Buffered Data)' 신호를 수신한다. 플래그 발생 논리 회로(48)는 플래그 버스(70)를 지나 다수의 플래그 값들을 제공하기 위해 오퍼랜드 할당 논리 회로(50)와 결과 선택논리 회로(46)에 연결된다. 또한 플래그 발생 논리 회로(48)는 J_{high}와 J_{low} 신호를 ALU(56)의 제 6 및 제 7 입력에 각각 제공하기 위해 ALU(56)에 연결된다.

오퍼랜드 할당 논리 회로(50)는 또한 정보 버스(A66, B68) 및 플래그 버스(70)에 연결되며, '멀티플라이어 1'로 표시된 제 1 입력 신호 및 '멀티플라이어 2'로 표시된 제 2입력 신호를 제공하기 위해 A-입력제어 논리 회로(52)에 접속된다. 부가적으로 오퍼랜드 할당 논리 회로(50)는 델타 신호를 수신하기 위해 델타 레지스터(60)에 연결된다. A-입력 제어 논리(52)는 제어 신호를 공급하기 위해 A-입력 멀티플렉서(54)에 접속된다.

결과 선택 논리(46)는 플래그 버스(70) 및 결과 버스(120)를 경유하여 정보를 수신한다. 정보는 '셀렉트(select)'로 표시된 제 1 출력과 '포스 $FF (Force $FF)'로 표시된 제 2 출력을 공급하기 위해 처리된다.

플래그 발생 논리 회로(48)는 제 5도에 보다 상세히 나타나있다. 플래그 발생 논리 회로(48)는 일반적으로 NOR 게이트(110), NOR 게이트(112), OR 게이트(114), NOR 게이트(116), NOR 게이트(118), AND 게이트(132), AND 게이트(134), 래치(136), AND 게이트(138), NOR 게이트(140), NOR 게이트(142), OR 게이트(144), NOR 게이트(146), NOR 게이트(148), 및 OR 게이트(150)로 구성된다. 래치들(130, 136)은 데이터 입력(D), 클락 입력(C), 리셋 입력(R), 및 데이터 출력(Q)을 가지는 D 플립플롭으로 각기 사용될 수 있다. D 플립플롭은 데이터 처리 기술에서는 잘 공지된 기술이어서 더 상세히 설명하지 않을 것이다.

제 4도의 데이터 버퍼(42)는 NOR 게이트(112)에 제 1 정보값의 8내지 15 비트들을 제공한다. 본 명세서에서 제 1 정보값은 버퍼링된 데이터 1(Data 1) 신호를 지나 제공된다. 버퍼링된 데이터 1 신호는 포인트 1 값 또는 슬로프 1 값을 제공할 것이다. 'Z2'로 표시된 NOR 게이트(112)의 출력은 래치(136)의 데이터 입력(D)과 AND 게이트(134)의 제 1 입력에 제공된다. 타이밍 제어 신호는 타이밍 제어 버스(38)에 의해 래치(136)의 클락 입력(C)에 제공된다. 래치(136)의 데이터 출력(Q)은 'ZP2' 신호를 제공하기 위해 AND 게이트(134)의 제 2 입력에 접속된다. J_{Phigh}로 표시된 AND 게이트(134)의 출력은 OR 게이트(144)의 제 1 입력에 제공된다.

또한, 버퍼링된 데이터 1 신호의 12 내지 15 비트들은 NOR 게이트(140)에 제공된다. 'S2'로 표시된 NOR 게이트(140)의 출력은 NOR 게이트(142)의 제 1 입력에 제공된다.

데이터 버퍼(42)는 또한 NOR 게이트(118)와 AND 게이트(138)에 제 2 정보값의 0 내지 7비트들을 제공한다. 제 2 정보값은 제 2 버퍼링된 데이터 신호를 지나 제공된다. 버퍼링된 데이터 2 신호는 AND 게이트(138)에 포인트 2 값의 0 내지 7비트들을 제공하며, NOR 게이트(118)에 슬로프 2의 0 내지 7비트들을 제공한다. AND 게이트(138)는 래치(130)의 데이터 입력(D)에 출력을 제공한다. NOR 게이트(118)의 출력이 어서트될 때 '0'의 슬로프값이 나타난다. AND 게이트(138)의 출력이 어서트될 때 모든 입력들은 어서트되며 포인트 2의 BO 내지 B7은 모두 '1'을 나타낸다. BO 내지 B7이 모두 '1'일 때 포인트 2는 최대치에 이른다. NOR 게이트(118)는 출력 'Z4'를 제공한다, Z4 신호는 AND 게이트(132)의 제 1 입력으로 전송된다. 타이밍 제어 신호는 타이밍 제어 버스(38)에 의해 래치(130)의 클락 입력(C)에 제공된다. 래치(130)의 데이터 출력(D)은 'ZP4' 신호를 공급하기 위해서 AND 게이트(132)의 제 2 입력에 연결된다. 'J_{Plow}'로 표시된 AND 게이트(132)의 출력은 OR 게이트(150)의 제 1 입력에 제공된다. AND 게이트(132)의 출력이 어서트될 때 포인트 P2는 최대치를 나타내고 슬로프 2는 '0'의 값을 나타냄으로써, 소정의 한계에 도달했고 라인(151)상의 신호 J_{Plow}는 종료 신호라는 것을 나타낸다. 이와 같은 종료 신호가 어서트되지 않으면, 특정 시스템의 입력값의 퍼지화는 계속된다. 신호 J_{Plow}는 조건 코드 레지스터(69)의 입력에 제공된다. 조건 코드 레지스터(69)는 실행 유닛(14)에서 연산 결과를 나타내기 위해 여러 개의 종래의 비트들을 포함한다. 이러한 비트들은 정지 디스에이블 비트(S), 인터럽트 마스크 비트(X), 하프-캐리 비트(H), 1-인터럽트 마스크 비트(1), 니게이트 비트(N), 제로 비트(Z), 오버플로우 비트(V), 및 캐리/바로우 비트(C)를 포함한다. 상기 비트들은 이 기술 분야에서 잘 공지된 방법으로 클리어(clear)되고 설정(set)된다.

그러나, 플래그 발생 논리(48)는 멤버쉽 함수가 마지막 세트에 있는지, 즉, 적당한 경계치가 $FF의 종점과 $OO의 슬로프를 가지는지의 여부를 검출하는 동안 신호 라인(151)의 신호 J_{Plow}는 C 비트를 설정한다. 그렇지 않으면, MEM 명령은 상기 비트를 클리어시킨다. 이러한 방법으로 조건 코드 레지스터(69)의 C 비트를 설정하는 것이 퍼지화 연산의 효율적으로 코딩할 수 있도록 하며 필요한 레지스터들의 수를 감소시킨다. 이러한 이점이 하기에서 보다 상세히 설명될 것이다.

또한 버퍼링된 데이터 2 신호의 4 내지 7비트들은 NOR 게이트(146)에 제공된 다. 'S4'로 표시된 NOR 게이트(146)의 출력은 NOR 게이트(148)의 제 1 입력에 제공된다.

델타 레지스터(60)는 NOR 게이트(110)에 제 3 정보값의 12 내지 15비트들을 제공한다. 상기 실행에서, 제 3 정보값은 델타 1 신호로 설명된 제 1델타 신호에 의해 제공된다. 'Z1'으로 표시된 NOR 게이트(110)의 출력은 또한 플래그 버스(70)를 경유하여 전송된다. Z1 신호는 또한 NOR 게이트(142)의 제 2 입력에 제공된다. 이와 유사하게 델타 레지

스터(60)는 NOR 게이트(116)에 제 4 정보값의 4 내지 7비트들을 제공한다. 'Z3'로 표시된 NOR 게이트(116)의 출력은 플래그 버스(70)를 경유하여 전송된다. 제 4정보값이 제 2 델타 신호를 거쳐 제공된다. Z3 신호는 또한 NOR 게이트(148)의 제 2 입력에 제공된다.

결과 버스(120)는 OR 게이트(114)의 제 1 및 제 2 입력에 각각 N_{low} ,와 N_{high} 신호를 제공한다. OR 게이트(114)의 출력은 N_{mem} 으로 표시되어 있다. N_{mem} 신호는 래치(130)와 래치(136)의 리셋 입력(R)에 제공된다. 또한 N_{mem} 신호는 NOR 게이트(142)와 NOR 게이트(148)의 각각의 제 3 입력에 제공된다.

'J_{Shigh} '로 표시된 NOR 게이트(142)의 출력은 OR 게이트(144)의 제 2 입력에 제공된다. OR 게이트(144)의 출력은 'J_{high} '로 표시된다. 또한 ,'J_{Slow} '로 표시된 NOR 게이트(148)의 출력은 OR 게이트(150)의 제 2 입력에 제공된다. OR 게이트(150)의 출력은 'J_{low} '로 표시된다. J_{low} 와 J_{high} 신호는 플래그 버스(70)를 경유하여 통신한 다.

오퍼랜드 할당 논리 회로(50)는 제 6도에 보다 상세히 도시되어 있다. 오퍼랜드 할당 논리 회로(50)는 일반적으로 제 1 멀티플렉서(72), 제 2 멀티플렉서(74), 제 3 멀티플렉서(76), 및 제 4 멀티플렉서(78)를 포함한다.

델타 레지스터(60)와 데이터 버퍼(42)는 멀티플렉서(72)의 제 1및 제 2 입력에 각각 델타 1과 슬로프 1의 낮은 4개의 비트를 제공한다. 또한, 델타 레지스터(60)와 데이터 버퍼(42)는 멀티플렉서(74)의 제 1및 제 2 입력에 각각 델타 1과 슬로프 1 값의 전체 8비트를 제공한다. 유사하게, 델타 레지스터(60)와 데이터 버퍼(42)는 각각 멀티플렉서(76)의 제 1 및 제 2 입력에 델타 2와 슬로프 2 값의 낮은 4비트를 제공한다. 델타 레지스터와 데이터 버퍼는 멀티플렉서(78)의 제 1 및 제 2 입력에 전체 8비트 델타 2와 슬로프 2 값을 제공한다.

'Z1' 신호는 플래그 버스(70)를 경유하여 멀티플렉서(72)의 제 3 입력에 제공되며, 또한 플래그 버스(70)를 경유하여 멀티플렉서(74)의 제 3입력에 제공된다. 유사하게 'Z3' 신호는 각각 멀티플렉서(76, 78)의 제 3 입력에 제공되며 플래그 버스(70)를 경유하여 전송된다.

멀티플렉서(72)는 4비트 멀티플라이어 1 신호를 출력하며 멀티플렉서(76)는 4비트 멀티플라이어 2 신호를 제공한다. 멀티플렉서(74)는 정보 버스(A66)의 하이 사이드를 거쳐 전송되는 8비트 신호 '멀티플리캔드 1'을 출력한다. 멀티플렉서(78)는 정보 버스(A66)의 로우 사이드를 거쳐 전송되는 8비트신호 '멀티플리캔드 2'를 출력한다.

ALU(56)는 제 7도에 보다 상세히 도시되어 있다. ALU(56)는 일반적으로 다수의 덧셈기 회로들(80, 82, 84, 86, 88, 92, 94, 96, 98, 100), 버퍼(90), 하이 캐리 룩어헤드 논리 회로(high carry lookahead logic circuit)(102), 로우 캐리 룩어헤드 논리 회로(low carry lookahead logic circuit)(104), 하이 상태 플래그 회로(high status flag circuit)(106), 및 로우 상태 플래그 회로(low status flag circuit)(108)를 포함한다.

A-입력 로우 사이드 신호는 'AO', 'A1', 'A2', 'A8','A9'로 각각 표시된 제 1 다수의 비트들을 제공한다. 비트 AO는 덧셈기(100)의 제 1 입력에 제공된다. 유사하게 비트 A1 내지 A9는 각각 덧셈기(98, 96, 94, 92)의 제 1 입력에 제공된다. B-입력 로우 사이드 신호는 'BO', 'B1','B2', 'B8', 'B9'로 각각 표시된 제 2 다수의 비트를 제공한다. 비트 BO은 덧셈기(100)의 제 2 입력에 제공된다. 유사하게 비트 B1 내지 B9는 각각 덧셈기(98, 96, 94, 92)의 제 2 입력에 제공된다. 제 7도에 상세히 도시되어 있지 않지만, 비트 A3 내지 A7, 비트 B3 내지 B7 각각은 제 1 다수의 덧셈기들(도시되지 않음) 중 각각의 하나에 제공된다. 제 1 다수의 덧셈기들(도시되지 않음) 각각은 덧셈기(92) 내지 덧셈기(100)와 같은 기능을 함으로써, 유사하게 구성된다.

'CO'로 표시된 신호가 로우 캐리 룩어헤드 논리(104)의 제 1 입력 및 덧셈기(100)의 제 3입력에 제공된다. 덧셈기(100)는 'SO'로 표시된 합계 신호를 출력한다. 또한 출력(100)은 로우 캐리 룩어헤드 논리 회로(104)에 'PO'로 표시된 전송 신호와 'GO'로 표시된 발생 신호를 제공한다.

'C1'으로 표시된 신호가 덧셈기(98)의 제 3 입력에 제공된다. 덧셈기(98)는 'S1'으로 표시된 합계 신호를 제공하며, 또한 전송 신호'P1'과 발생 신호'G1'을 발생시킨다. 유사하게 'C2'로 표시된 신호가 덧셈기(96)의 제 3 입력에 제공된다. 덧셈기(96)는 합계 출력'S2'와 전송 신호 'P2'와 발생 신호'G2'를 제공한다. 또한 'C8'로 표시된 신호가 덧셈기(94)의 제 3 입력에 제공된다. 덧셈기(94)는 합계 신호'S8', 전송 신호'P8', 및 발생 신호'G8'을 출력한다. 캐리 신호 'C9'는 덧셈기(92)에 제 3 입력을 제공한다. 덧셈기(92)는 합계 신호 'S9', 전송 신호'P9', 및 발생 신호'G9'를 출력한다. 덧셈기(92) 내지 덧셈기(100)에 의한 전송과 발생 신호들 각각은 로우 캐리 룩어헤드 논리 회로(104)에 제공된다.

로우 캐리 룩어헤드 논리 회로(104)는 버퍼(90)에 캐리 신호 'CO'를 제공한다. 분할 모드 제어 신호는 또한 버퍼(90)에 제공된다. 버퍼(90)는 'Cin'으로 표시된 신호를 제공한다.

A-입력 하이 사이드 신호는 'A10','A11','A12', 'A18', 'A19'로 각각 표시된 제 1 다수의 비트들을 제공한다. 비트 A10은 덧셈기(88)의 제 1 입력에 입력된다. 유사하게 비트 A11 내지 A19는 덧셈기(86, 84, 82, 80)의 제 1 입력에 각각 제공된다. B-입력 하이 사이드 신호는 'B10', 'B11', 'B12', 'B18', 'B19'로 각각 표시된 제 2 다수의 비트들을 제공한다, 비트 B10은 덧셈기(88)의 제 2 입력에 제공된다. 또한 비트 B11 내지 B19는 각각의 덧셈기(86, 84, 82, 80)의 제 2 입력에 각각 제공된다. 제 7도에 상세히 도시되어 있지 않지만 비트 A13 내지 A17 및 비트 B13 내지 B17의 각각의 비트들은 제 2 다수의 덧셈기들 중 각각의 하나에 제공된다. 제 2 다수의 덧셈기들 각각은 덧셈기(80) 내지 덧셈기(88)와 같이 기능함으로써, 유사하게 구성된다.

Cin 신호는 덧셈기(88)의 제 3 입력에 제공된다. 덧셈기(88)는 'S10'으로 표시된 합계 신호를 출력한다. 또한, 덧셈기(88)는 하이 캐리 룩어헤드 논리 회로(102)에 전송 신호'P10'과 발생 신호'G10'을 제공한다.

'C11'로 표시된 신호는 덧셈기(86)의 제 3입력에 제공된다. 덧셈기(86)는 합계 신호'S11'을 제공한다. 덧셈기(86)는 또한 전송 신호'P11'과 발생 신호 'G11'을 발생한다. 유사하게 'C12'로 표시된 신호는 덧셈기(84)의 제 3 입력에 제공된다. 덧셈기(84)는 합계 출력 'S12', 전송 신호 'P12', 및 발생 신호 'G12'를 제공한다. 또한 'C18'로 표시된 신호는 덧셈기(82)의 제 3입력에 제공된다. 덧셈기(82)는 합계 신호 'S18', 전송 신호 'P18', 및 발생 신호 'G18'을 출력한다. 'C19'로 표시된 캐리 신호는 덧셈기(80)에 제 3입력을 제공한다. 덧셈기(80)는 합계 신호 'S19', 전송 신호 'P19', 및 발생 신호'G19'를 출력한다. 덧셈기(80) 내지 덧셈기(88)에 의한 전송과 발생 신호 각각은 하이 캐리 룩어헤드 논리

회로(102)에 제공된다. 하이 캐리 룩어헤드 논리 회로(102)는 'C20'으로 표시된 캐리 신호를 출력한다.
제 7도에서는 도시되어 있지 않지만, 덧셈기(92) 내지 덧셈기(100) 및 로우 캐리 룩어헤드 논리 회로(104)에 의해 발생된 캐리 신호, 전송 신호, 합계 신호, 발생 신호 각각은 또한 로우 상태 플래그 회로(108)에 제공된다. J_{low} 신호는 또한 로우 상태 플래그 회로(108)에 제공된다. 유사하게 덧셈기(80) 내지 덧셈기(88) 및 하이 캐리 룩어헤드 논리 회로(102)에 의해 발생된 캐리 신호, 전송 신호, 합계 신호, 발생 신호 각각은 또한 하이 상태 플래그 회로(106)에 제공된다. J_{Phigh} 신호는 또한 하이 상태 플래그 회로(106)에 제공된다. 로우 상태 플래그 회로(108)는 'S_{low}', 'V_{low}', 'N_{low}', 'H_{low}', 및 'Z_{low}'로 표시된 다수의 신호들을 각각 제공한다. 하이 상태 플래그 회로(106)는 'S_{high}', 'V_{high}', 'N_{high}', 'H_{high}', 및 'Z_{high}'로 표시된 다수의 신호들을 각각 제공한다. 로우 상태 플래그 회로(108) 및 하이 상태 플래그 회로(106) 각각의 출력은 결과 버스(120)를 경유하여 전송된다.
퍼지 논리 동작을 실행하는 동안, 데이터 처리 시스템(10)의 사용자는 시스템 입력의 퍼지화를 수행하기 위해 의사 기호 형태인 MEM(Membership Evaluation)을 가지는 명령을 사용할 수 있다. 본 명세서에 설명된 본 발명의 실시예에 있어서, 사용자는 MEM 명령의 실행에 앞서 시스템 입력 레지스터(64)에 퍼지화된 입력을 저장해야만 한다.
또한, 사용자는 실행 유닛(14)에서 제 1 인덱스 레지스터(도시되지 않음)에 포인터를 저장해야 한다. 포인터는 멤버쉽 세트의 사다리꼴 모양을 확인하기 위해 필요한 네 지점의 출발 어드레스 위치를 가리킨다. 상기 제 2도에서 설명한 것과 같이, 네 지점은 포인트 1, 포인트 2, 슬로프 1, 슬로프 2로 각각 표시되어 있다. 다른 형태에서, 사용자는 상기 각각의 값들을 명령의 오퍼랜드로서 제공할 수 있다. 이러한 기술들은 데이터 처리 기술에서 공지된 기술이므로 본원에서는 자세히 언급하지 않는다. 사용자는 MEM 명령을 실행한 후에 퍼지화된 데이터의 저장위치를 명시해야 한다. 한가지 형태에서 제 2인덱스 레지스터는 퍼지화된 데이터의 저장 위치를 가리키기 위해 사용될 수 있다. 또한 시스템 입력값은 시스템 입력 레지스터(64)에 저장되어야 한다.
본 발명의 상기 실시예에서 MEM 명령의 실행이 제 8도의 흐름도에 따라 실행된다. 흐름도는 MEM 명령이 실행되는 동안에 실행되는 각각의 함수의 간단한 설명을 제공한다. 각 함수의 보다 상세한 설명은 MEM 명령이 실행되는 예를 따라 이루어질 것이다.
제 8도에서와 같이, MEM 명령 실행의 제 1 단계는 MEM 명령이 디코딩될 때 이루어진다. 인덱스 레지스터에 저장된 어드레스는 내부 메모리(32)의 어드레스 X로부터 포인트 1 값과 포인트 2 값을 동시에 액세스하기 위해 사용된다. 이때 어드레스는 다음 어드레스(X+2)를 가리키기 위해 증가된다.
그리고 나서, 델타 1 및 델타 2 값이 동시에 발생된다. 델타 1값은 포인트 1 값을 시스템 입력 레지스터(64)의 값에서 감산함으로써 얻어진다. 델타 2 값은 시스템 입력 레지스터(64)의 값을 포인트 2 값에서 감산함으로써 얻어진다. 델타 1과 델타 2 값은 델타 레지스터(60)에 저장된다.
다음으로, 포인트 1 값이 $OO에에 해당하면, ZP2 신호는 이진수 값 1로 래치되고 어서트된다. 유사하게 포인트 2 값이 $FF에 해당하면, ZP4 신호는 이진수 값 1로 래치되고 어서트된다.
슬로프 1 값 및 슬로프 2 값은 내부 메모리(32)의 어드레스 X+2에서 검색되고 동시에 데이터 버퍼(42)에 저장된다. 이때 어드레스는 다음 어드레스 X+4를 가 리키도록 증가된다.
델타 1 및 델타 2 값은 각각이 네거티브(negative)인지를 정하기 위해 검사된다. 델타 1 또는 델타 2 값이 네거티브이면 시스템 입력값은 현재 검사되고 있는 멤버쉽 세트의 외부에 있다. 따라서, 데이터 버퍼(42)의 내용은 $OO의 멤버쉽 정도를 강제하기 위해서 클리어된다. 또한, ZP2 및 ZP4 값들은 클리어된 래치들(136, 130)에 저장된다.
다음으로, 델타 1 및 델타 2 값들의 각각의 상위 4비트들은 16진수값 $O에 해당하는지의 여부를 결정하기 위해 검사된다. 델타 1의 상위 4비트 값들이 $O에 해당하면, Z1 플래그가 어서트된다. 유사하게 델타 2의 상위4 비트의 값들이 $0에 해당하면 Z3 플래그가 어서트된다.
만약 Z1 플래그가 어서트되면, 델타 1 값은 멀티플라이어 1 값이고 슬로프 1 값은 멀티플리캔드 1 값이다. 그렇지 않으면, 슬로프 1 값은 멀티플라이어 1 값이고 델타 1 값은 멀티플리캔드 1 값이다. Z3 플래그가 어서트되면 델타 2 값은 멀티플라이어 2 값이고 슬로프 2 값은 멀티플리캔드 2 값이다. 그렇지 않으면, 슬로프 2 값은 멀티플라이어 2 값이고 델타 2 값은 멀티플리캔드 1 값이다.
델타 1 값과 슬로프 1 값의 프로덕트가 다음에 발생된다. 또한, 델타 2 값과 슬로프 2 값의 프로덕트는 2개의 곱셈 연산에서 동시에 발생된다.
델타 1 값과 슬로프 1 값의 프로덕트가 $FF 보다 크면, 하이 포화 플래그(high saturation flag)(S_{high})는 어서트된다. 또한 J_{high} 신호가 어서트되면 S_{high} 신호가 어서트된다. J_{high} 는 슬로프 1 또는 델타 1 값이 상위 4비트에서 $O의 16진수값을 가지면 어서트되고, 또한 포인트 1 또는 슬로프 1 값이 16진수값 $OO을 가지는 특별한 경우에 어서트된다.
포인트 1 값이 $OO에 해당하면 ZP2 신호는 어서트되고, 슬로프 1 값이 $OO에 해당하면 Z2 신호는 어서트된다.
유사하게, 로우 포화 플래그(S_{low})는 델타 2 값과 슬로프 2 값이 $FF 보다 크면 어서트되고, 또한 J_{low} 신호가 어서트되면 어서트된다. J_{low} 는 슬로프 2 또는 델타 2 값들이 상위 4비트에 16진수값 $O을 가지지 않는다면 어서트된다. 또한, Z1 및 Z2 신호들은 반드시 니게이트 되어야 한다. J_{low} 는 또한 포인트 2 값이 16진수값 $FF를 가지며 슬로프 2 값이 16진수값 $OO을 가지는 특별한 경우에 어서트된다. 포인트 2 값이 $FF에 해당하면 ZP4 신호는 어서트되고, 슬로프 2 값이 $OO 에 해당하면 Z4 신호가 어서트된다. Z4 및 ZP4 신호들 모두가 어서트되면, J_{Plow} 신호는 종료 조건이 검출되었는지를 알리기 위해 어서트된다.
S_{high} 및 S_{low} 신호 모두가 어서트되면, 시스템 입력의 멤버쉽 정도는 $FF이다. S_{high} 가 니게이트되고 S_{low} 가 어서트되면 시스템 입력의 멤버쉽 정도는 델타 1 값과 슬로프 1 값의 프로덕트에 해당한다. 그렇지 않으면, 멤버쉽 정도는 델타 2와 슬로프 2 값의 프로덕트에 해당한다.
멤버쉽 정도는 사용자에 의해 명시된 내부 메모리(32)의 어드레스에 제공되 며, MEM 명령의 실행이 종료된다.

제 8도의 흐름도는 데이터 처리 시스템(10)에서 MEM 명령이 실행되는 과정을 보여주며 보다 상세한 설명은 이하에서 이루어진다.
연산이 되는 동안, 사용자는 MEM 명령을 내부 또는 외부 메모리(32)에 저장된 소프트웨어 프로그램을 통해 데이터 처리 시스템(10)에 제공한다. 데이터 처리 시스템(10)에 외부 소스에 의해 제공된 MEM 명령은 외부 데이터 버스(33)를 경유하여 외부 버스 인터페이스(30)에 입력될 것이다. 외부 버스 인터페이스(3)는 계속적으로 내부 데이터 버스(34)를 경유하여 MEM 명령을 CPU(12)에 제공할 것이다. MEM 명령이 내부 메모리(32)의 소프트웨어 프로그램에 의해 제공되면, MEM 명령은 내부 데이터 버스(34)를 경유하여 CPU(12)에 제공될 것이다. CPU(12)에서, 실행 유닛(14)은 버스 제어 논리 회로(16), 제어 유닛(20), 및 시퀀서(22) 각각으로부터 수신된 제어 신호에 응답하여 MEM 명령을 명령 디코딩 논리 회로(18)에 제공할 것이다. 명령 디코딩 논리 회로(18)는 계속해서 MEM 명령의 적당한 실행을 위해 필요한 다수의 정보와 제어 신호를 제공하기 위해 MEM 명령을 디코딩한다.
MEM 명령을 수신하고 디코딩 하자마자 실행 유닛(14)은 시스템 입력을 퍼지화하기 위해 필요한 단계를 시작한다. 상기에서와 같이, 퍼지화된 값을 지정하는 시스템 입력은 MEM 명령의 실행에 앞서 데이터 처리 시스템(10)의 사용자에 의해 시스템 입력 레지스터(64)에 저장된다.
시스템 입력값에 더해, 데이터 처리 시스템의 사용자는 퍼지화 논리 연산의 특정 실행에서 정해진 각각의 멤버쉽을 정하는 4개의 값들을 입력할 필요가 있다. 예를 들어, 제 1도에서, 온도가 퍼지화되면, 5개의 멤버쉽 세트들이 정해질 필요가 있다. 따라서, 모두 20개의 값들이 전 시스템에서 각 멤버쉽 세트를 적절히 정의하기 위해 필요할 것이다. 제 2도에서 설명된 상기에서와 같이, 하나의 멤버쉽 세트를 명시하기 위해 필요한 값은 포인트 1, 포인트 2, 슬로프 1, 슬로프 2 이다. 상기 각각의 경계들은 2진수값 8비트에 의해 표시된다.
퍼지화 연산동안, 각각의 멤버쉽 세트들은 시스템 입력값을 기준으로 평가되어야 한다. 제 1도에서 보여진 퍼지화 논리 연산 시스템의 경우에, 5개의 멤버쉽 세트 평가들 시스템 입력 온도에서 나타난다. 따라서, 각 멤버쉽 세트에서 멤버쉽의 정도를 정의하기 위해서 필요한 시간을 최소화하는 것이 중요하다.
인덱스 레지스터의 포인터는 내부 메모리(32)에서 포인트 1 값의 어드레스를 가리킨다. 내부 메모리(32)로부터 액세스될 때, 포인트 1 값은 외부 정보 버스(40)를 경유하여 데이터 버퍼(42)에 제공된다. 데이터 버퍼(42)는 정보 버스(A66)를 경유하여 A-입력 멀티플렉서(54)에 포인트 1 값을 전달한다. A-입력 멀티플렉서(54)는 델타 1 값을 제공하기 위해 시스템 입력값으로부터 감산이 이루어지는 ALU에 포인트 1 값을 제공한다. 델타 1 값은 정보 버스(B68)를 경유하여 델타 레지스터(60)에 제공된다. 또한 데이터 버퍼(42)는 버퍼화된 데이터 1 신호를 경유하여 플래그 발생 논리 회로(48)에 포인트 1 값을 제공한다. 제 5도의 플래그 발생 논리 회로에서, 포인트 1 값의 각각의 비트들은 NOR 게이트(112)의 입력에 제공된다. NOR 연산의 결과인 Z2 신호는 타이밍 제어 신호에 따라 정해진 시간 내의 한 점에서 래치(136)에 저장된다.
본 발명의 상기 실행에서, 포인트 1 과 포인트 2 값은 동시에 내부 메모리(32)로부터 액세스된다. 2개의 값들은 외부 정보 버스(40)를 경유하여 데이터 버퍼(42)에 전송된다. 데이터 버퍼(42)는 정보 버스(B68)를 경유하여 B-입력 멀티플렉서(58)에 포인트 2 값을 전한다. B-입력 멀티플렉서(58)는 계속해서 시스템 입력값이 델타 2값을 제공하기 위해 감산되는 ALU에 포인트 2 값을 제공한다. 정보 버스(B68)는 델타 2 값을 델타 레지스터(60)에 제공한다. 또한 데이터 버퍼(42)는 버퍼링된 데이터 2 신호를 거쳐 플래그 발생 논리 회로(48)에 포인트 2 값을 제공한다. 제 5도의 플래그 발생 논리 회로(48)에서, 포인트 2 값의 각 비트는 AND 게이트(138)의 입력에 제공된다. AND 연산의 결과가 타이밍제어 신호에 따라 정해진 시간 내에 래치(130)에 저장된다. AND 연산은 포인트 2 값이 $FF와 같은지를 검출한다.
ALU(56)는 델타 1과 델타 2 값이 동시에 계산되도록 상기의 감산 연산들 각각을 수행하며, 연산이 정상(normal) 모드일 때 16비트 연산을 수행한다. 또한 ALU(56)는 분할 모드 연산일 때 2개의 독립된 8비트 계산들을 동시에 실행할 수 있다. 2개의 8비트 계산들이 동시에 수행되기 때문에 오직 16비트 연산이 실행되는 ALU의 경우에서 낭비되는 사이클은 없을 것이다. 따라서, ALU 는 8비트 연산을 수행할 때 실행 유닛(14)의 속도와 효율을 크게 증가시킬 것이다. ALU(56)의 연산은 후에 보다 상세히 설명될 것이다.
인덱스 레지스터의 포인터는 내부 메모리(32)에서 슬로프 1 값의 어드레스를 결과적으로 가리킨다. 내부 메모리(32)는 내부 정보 버스(40)를 경유하여 데이터 버퍼(42)에 8비트 슬로프 1 값과 슬로프 2 값을 제공한다. 데이터 버퍼(42)는 16비트 버퍼링된 데이터 신호를 경유하여 플래그 발생 논리 회로(48)에 슬로프 값 모두를 전달한다.
델타 1 및 델타 2 값의 계산과 슬로프 1 및 슬로프 2 값들의 수신과 동시에 플래그 발생 논리 회로(48)는 현재 검사되는 멤버쉽 세트에서 멤버쉽의 정도를 정하기 위해 필요한 다수의 플래그들을 발생시킬 수 있다. 제 5도에서 델타 1과 델타 2 및 슬로프 1과 슬로프 2 값들의 상위 4비트들은 각각 NOR 게이트(110, 116, 140, 146)에 제공된다.
NOR 게이트(140)는 S2신호를 발생시키며 NOR 게이트(146)는 S4신호를 발생시킨다. S2신호는 슬로프 1 값의 상위 4비트가 16진수값 $0을 가질 때 어서트된다. 유사하게, S4 신호는 슬로프 2 값의 상위 4비트가 16진수값 $0을 가질 때 어서트된다.
플래그 발생 논리 회로(48)에서, 슬로프 1과 슬로프 2 값들은 각각 버퍼링된 데이터 신호에 의해 NOR 게이트(112)와 NOR 게이트(118)에 제공된다. Z2 플래그는 슬로프 1 값이 16진수값 $OO에 해당할 때 어서트된다. NOR 게이트(112)에 의해 발생된 Z2 플래그는 AND 게이트(136)와 래치(136)의 데이터 입력에 제공된다. Z2 플래그는 타이밍 제어 신호가 제 1 소정의 논리값을 가질 때 래치(136)에 래치된다. 또한 타이밍 제어 신호가 제 2 소정의 논리값을 가질 때, 래치(136)는 AND 게이트(134)에 앞에서 래치된 ZP2 신호를 제공한다. AND게이트(134)는 J_{Phigh} 신호를 발생시키기 위해 사용된다.
Z4 플래그는 슬로프 2 값이 16진수값 $OO에 해당할 때 어서트된다. OR 게이트(118)에 의해 발생된 Z4 플래그는 AND 게이트(132)에 제공된다. AND게이트(138)의 출력은 래치(130)의 데이터 입력에 제공된다. AND게이트(138)의

출력은 타이밍 제어 신호가 제 1 소정의 논리값을 가질 때 래치(130)에서 래치된다. 또한 타이밍 제어 신호가 제 2 소정의 논리값을 가질 때 래치(130)는 AND 게이트(132)에 앞에서 래치된 ZP4 신호를 제공한다. AND 게이트(132)는 J_{Plow} 신호를 발생시키기 위해 사용된다 J_{Plow} 신호의 상태는 타이밍 조건이 검출되는 것을 기록하기 위해 제 4도의 조건 코드 레지스터(69)에 래치된다.

Z1 플래그 신호는 델타 1 값의 상위 4비트 각각이 니게이트 될 때에만 어서트된다. 유사하게 Z3 플래그 신호는 델타 2 값의 상위 4비트가 16진수값 $0을 가질 때에만 어서트된다. 플래그 Z1과 Z3 각각은 이하에서 보다 상세히 설명될 오퍼랜드 할당 단계 동안에 계속해서 사용된다.

델타 1과 델타 2값에 더하여, 결과 버스(120)는 N_{mem} 신호를 발생 시키기 위해 OR 게이트(114)에 N_{high} 및 N_{low} 신호를 전달한다. N_{low} 신호는 ALU(56)에 의해 제공된 델타 2 값이 0 보다 작으면 어서트된다. 유사하게 N_{high} 신호는 ALU(56)에 의해 제공된 델타 1 값이 0 보다 작으면 어서트된다. 상기에 설명된 것과 같이, 델타 1 또는 델타 2 값들이 0 보다 작으면, 시스템 입력은 검사되고 있는 멤버쉽 세트의 멤버가 아니다.

따라서, N_{mem} 신호가 어서트될 때, 시스템 입력은 현재 검사되고 있는 멤버쉽 세트에서 0의 멤버쉽 정도를 가진다. 계속해서, N_{mem} 신호는 A-입력 제어 논리(52)에 제공된다. 어서트될 때, N_{mem} 신호는 A-입력제어 논리 회로(52)가 A-제어 신호를 어서트하도록 하여 A-입력 멀티플렉서(54)가 ALU(56)에 16진수값 $OO을 제공하도록 한다. 계속해서, ALU(56)는 정보 버스(B68)를 경유하여 데이터 버스(42)에 $OO의 값을 제공한다.

또한, N_{mem} 신호가 어서트될 때 래치(130)는 리셋되어 ZP4 신호는 니게이트된다. 따라서, J_{Plow} 신호는 잘못해서 어서트되지 않을 수 있다. 유사하게 래치(136)는 리셋되어 ZP2신호가 니게이트된다. 따라서 J_{Phigh} 신호는 잘못해서 어서트되지 않을 수 있다.

N_{mem} 신호, Si 신호 및 Z1 신호는 J_{Shigh} 신호를 발생시키기 위해 NOR 게이트(142)에 각각 제공된다. J_{Shigh} 및 J_{Phigh} 신호는 J_{high} 신호를 제공하기 위해 'OR' 된다.

J_{Phigh} 신호는 포인트 1 값과 슬로프 1 값이 16진수값 $OO을 가지는 특별한 경우를 나타내기 위해 제공된다. 이러한 것은 멤버쉽 세트가 부분적으로 사다리꼴 내에 위치하여 멤버쉽의 정도가 경계값 포인트 1에서 $FF 값을 가지는 경우에 일어날 것이다. 제 1도의 추움 온도 멤버쉽 함수는 이러한 특별한 경우의 예이다. J_{high} 신호는 입력값이 평가되는 멤버쉽 세트내에 들어갈 때 포화 플래그(S_{high})가 어서트되도록 하기 위해 ALU(56)에 제공된다. J_{high} 신호는 또한 Z1, S2, 및 N_{mem} 신호가 모두 니게이트되면 어서트된다. 이것은 델타 1 또는 슬로프 1 값이 0 또는 그보다 작지 않고 현재의 입력이 멤버쉽 세트 내에 있는 경우에 해당한다.

N_{mem} 신호, S4 신호, 및 Z3 신호는 J_{Slow} 신호를 발생시키기 위해 각각 NOR 게이트 (148)에 제공된다. J_{Slow} 와 J_{Plow} 신호는 J_{low} 신호를 제공하기 위해 'OR'된다.

J_{low} 신호는 포인트 2 값이 16진수값 $FF를 가지며 슬로프 2 값이 16진수값 $OO을 가지는 특별한 경우를 나타내기 위해 제공된다. 이와 같은 경우는 멤버쉽 세트가 부분적으로 사다리꼴 내에 위치하여 멤버쉽의 정도가 경계값 포인트 2에서 $FF로 끝난다면 나타날 것이다. 물론 J_{low} 신호는 델타 2와 슬로프 2 값의 적이 $FF 보다 클 때 또한 어서트된다. 그러나, 어서트될 때 신호 J_{Plow} 는 $FF에서 끝나는 퍼지 세트만을 나타내며 $OO의 슬로프 2 값은 마지막 퍼지세트를 나타내어 현재의 시스템 입력을 위한 퍼지화의 종료가 나타난다. 제 1도의 더움 온도 멤버쉽 함수는 이러한 특별한 경우의 예이다. J_{low} 신호는 입력값이 현재 평가 되는 멤버쉽 세트 내에 해당될 때 포화 플래그(S_{low})가 어서트되도록 한다. J_{low} 신호는 Z3, S4, 및 N_{mem} 신호가 모두 니게이트되면 또한 어서트된다. 이것은 델타 2 또는 슬로프 2 값이 $OF 또는 그보다 작지 않은 경우에 해당하며 현재의 입력이 멤버쉽 세트 내에 있는 경우이다.

그러나, 시스템 입력값이 현재 검사되고 있는 멤버쉽 세트 내에 있다면 N_{mem} 신호는 니게이트되며 Z3 플래그 신호는 멤버쉽 세트에서 멤버쉽의 정도를 정하기 위해 오퍼랜드 할당 논리 회로(50)와 결과 선택 논리 회로(46)에 제공된다.

멤버쉽 세트에서 멤버쉽의 정도는 멤버쉽 세트를 정하는 사다리꼴에서 포화 포인트인 델타 1과 슬로프 1의 프로덕트나 델타 2와 슬로프 2의 프로덕트의 최소값을 얻음으로써 결정된다. 제 2도의 예에서 포화 지점은 16진수값 $FF를 가지며 0.996으로 해석된다.

델타 1과 슬로프 1의 프로덕트와 델타 2와 슬로프 2의 프로덕트를 계산하기 위해 빠른 곱셈 연산을 가능하게 하는 방법이 발달해왔다. 이러한 기술 분야에서 잘 공지된 것과 같이 멀티플라이 연산은 멀티플라이어가 멀티플리캔드 보다 작을 때 보다 빨리 이루어진다. 이러한 예에서 멤버쉽 세트의 포화 값이 $FF 이라면 수학적 해석은 프로덕트를 이루는 델타나 슬로프 중 적어도 하나는 $OF 보다 작거나 같아야만 한다는 것을 나타낸다. 그러나 이러한 경우의 한가지 예외가 현재의 시스템 입력값이 멤버쉽 세트 외에 있거나 멤버쉽 세트의 포화 영역 내에 있을 때 일어난다.

각각의 델타 및 슬로프 값들의 4개의 상위 비트들을 검사함으로써 16진수값 $OF 또는 그보다 작은 값을 가지는 첫 번째 조건이 멀티플라이어로서 사용될 수 있 다. 따라서, 곱셈 연산을 완성하기 위해 필요한 시간은 더 짧아진다. 델타 1 또는 슬로프 1 값이 $OF 보다 더 큰 16진수 값을 가지면 J_{high} 신호는 S_{high} 포화 플래그가 어서트되게 한다. 유사하게 델타 2와 슬로프 2 값이 $OF 보다 큰 16진수값을 가지면 J_{low} 신호는 S_{low} 포화 플래그가 어서트되도록 한다. Z1 플래그 신호는 델타 1 이나 슬로프 1 값이 4비트 멀티플라이어 1 신호로서 제공되는지를 나타내기 위해 사용된다. 유사하게 Z3 플래그 신호는 델타 2 또는 슬로프 2 값이 4비트 멀티플라이어 2 신호로서 제공 되는지를 나타내기 위해 사용된다. 제 6도의 오퍼랜드 할당 논리 회로(50)는 수행되는 2개의 곱셈 연산 각각의 멀티플라이어 값을 결정하기 위한 한가지 실시예를 제공한다.

제 6도에서와 같이 멀티플렉서(72)는 델타 1 또는 슬로프 1 값이 4-비트 멀티플라이어 1 신호에 할당되는지를 결정한다. Z1 플래그 신호가 어서트되면 델타 1 값은 4비트 멀티플라이어 1 신호를 지나 전해진다. Z1 신호가 어서트되지 않으면 4비트 멀티플라이어 1 신호는 슬로프 1 값에 할당될 것이다. 또한, Z1 신호가 어서트되면 멀티플렉서(74)는 8비트 멀티플리캔드 1 신호로서 슬로프 1 값을 A_{high} 에 제공한다. 만일 Z1 신호가 니게이트되면, 델타 1값은 8비트

멀티플리캔드 1 신호에 할당된다.
유사하게 멀티플렉서(76)는 델타 2 또는 슬로프 2 값이 Z3 플래그신호의 값에 기초한 멀티플라이어 2 신호에 할당되는지를 결정한다. Z3 플래그 신호가 어서트되면, 멀티플렉서(76)는 멀티플라이어 2 신호로서 델타 2신호를 제공한다. Z3 신 호가 어서트되지 않으면, 4비트 멀티플라이어 2 신호는 슬로프 2 값에 할당될 수 있다. 또한 Z3 플래그 신호가 어서트되면 멀티플렉서(78)는 멀티플리캔드 2 신호로서 슬로프 2 값을 제공한다. 만일 Z3 신호가 니게이트되면 델타 2 값은 8-비트 멀티플리캔드 1 신호에 할당된다.
멀티플라이어 1 및 멀티플라이어 2 신호들은 오퍼랜드 할당 논리회로(50)로부터 A-입력 제어 논리 회로(52)에 전해진다. A-입력 제어 논리 회로(52)는 A-입력 멀티플렉서(54)의 특정 쉬프트 동작을 결정하는 A-제어 신호를 제공하기 위해 각각 멀티플라이어 1과 멀티플라이어 2 신호를 가진다.
연산 동안, 델타 1 또는 슬로프 1 값인 멀티플라이어 1 신호는, 제 1 곱셈 연산을 수행하기 위해 멀티플리캔드 1 신호가 0, 1, 또는 2에 의해 쉬프트되어야 하는지를 결정하기 위해, A-입력 제어 논리 회로(52)에 제공된다. 또한 A-입력 제어 논리 회로(52)는 멀티플리캔드 1 신호를 니게이트하거나 0으로 할 수 있다. A-제어 신호는 실행되고 있는 연산 형태를 나타내기 위해 A-입력멀티플렉서(54)에 제공된다. 멀티플리캔드 1 신호는 정보 버스(A66)의 하이 사이드를 경유하여 A-입력 멀티프렉서(54)에 전해진다. A-입력 제어 논리 회로(52)는 A-제어 신호에 따라 멀티플리캔드 1 신호를 수정하며 수정된 멀티플리캔드 1 신호를 ALU(56)에 제공한다. 유사하게 델타 2 또는 슬로프 2 값인 멀티플라이어 2 신호는 A-입력 제어 논리 회로(52)에 제공된다. 멀티플라이어 2 신호는 제 2 멀티플리케이션 연산을 동시에 실행하기 위해 0, 1, 또는 2에 의해 멀티플리캔드 2신호를 쉬프트하기 위해 사용된다. 또한 A-입력 제어 논리 회로(52)는 멀티플리캔드 2 신호를 0 또는 니게이트 할 수 있다. 멀티플라이어 2 신호는 정보 버스(A66)의 로우 사이드를 경유하여 A-입력멀티플렉서(54)에 전해진다. A-입력 제어 논리 회로(52)는 A-제어 신호에 따라 멀티플리캔드 2신호를 수정하며 이 수정된 멀티플리캔드 2 신호를 ALU(56)에 제공한다.
B-입력 멀티 플렉서(58)의 제어는 Micro-ROM 제어 버스(65)를 경유하여 전송되며 MEM 명령을 디코딩 하는 동안에 발생된 외부 Micro-ROM 제어 신호에 의해 제공된다. Micro-ROM 제어 신호의 발생, 라우팅, 및 사용은 데이터 처리기술에서 잘 공지된 기술로서 본원에서는 상세히 논의되지 않을 것이다. 본원에서 설명된 예에서 $OO의 초기화된 부분 프로덕트(partial product)는 B-입력 멀티플렉서(58)를 경유하여 ALU(56)에 제공된다.
A-입력 제어 논리 회로(52)로부터 적절한 A-제어 신호를 수신하자 마자 A-입력 멀티플렉서(54)는 수정된 멀티플리캔드 1및 수정된 멀티플리캔드 2 신호를 각각 ALU(56)의 로우 사이드와 하이 사이드에 제공한다. B-입력 멀티플렉서(58)는 동시에 초기화된 부분 프로덕트를 ALU(56)의 하이 사이드와 로우 사이드에 각각 제공한다.
ALU(56)는 제 7도에 보다 상세히 도시되어 있다. 연산되는 동안 수정된 멀티플리캔드 2 신호는 덧셈기(92)에서 덧셈기(100)의 각 하나에 제공된다. 'a9'로 표시된 수정된 멀티플리캔드 2 신호의 비트 9는 덧셈기(92)의 제 1 입력에 접속된다. 유사하게 수정된 멀티플리캔드 2 신호의 0 내지 8비트들은 덧셈기(94, 96, 98, 100)의 제 1 입력에 각각 접속된다. 'b9'로 표시된 초기화된 부분 프로덕트의 비트 9는 덧셈기(92)의 제 2 입력에 접속된다. 초기화된 부분 프로덕트의 비트 0 내지 비트 8은 또한 덧셈기(94, 96, 98, 100)의 제 2 입력에 각각 접속된다.
'CO'로 표시된 캐리 신호는 덧셈기(100)의 제 3입력에 제공된다. CO 신호는 멀티플라이어 2 신호의 값과 실행되는 연산에 응답하여, 외부 소스에 의해 ALU(56)에 제공된다. 3개의 입력 신호 각각에 기초하여 덧셈기(100)는 3개의 출력 신호를 제공한다. 제 1 출력신호인 SO신호는 다음과 같다.
(1) SO = aO = bO = cO
제 2 출력 신호는 발생 신호, 즉, GO 신호이다. 이것은 다음과 같다:
(2) GO = aO ·bO
제 3 출력 신호는 전달 신호, 즉, PO 신호이다. 이것은 다음과 같다:
(3) PO = aO + bO
GO 및 PO 신호들에 응답하여 로우 캐리 룩어헤드 논리 회로(104)는 덧셈기(98)의 제 3 입력에 제공되는 C1 신호를 발생시킨다. C1 신호는 다음과 같이 표현된다.
(4) C1 = GO + PO ·CO
a1, b1, 및 c1 신호들 각각에 응답하여, 덧셈기(98)는 발생 신호 'G1'과 전달 신호 'P1'을 로우 캐리 룩어헤드 논리 회로(104)에 제공한다.
G1, P1, C1 신호들에 응답하여, 로우 캐리 룩어헤드 논리 회로(104)는 덧셈기(96)의 제 3 입력에 제공되는 C2 신호를 발생시킨다. C2 신호는 다음과 같은 방정식에 의해 계산된다.
(5) C2 = G1 + P1(GO+PO)CO
덧셈기(96)는 이때 발생 신호 'G2'와 전달 신호 'P2'를 로우 캐리 룩어 헤드 논리 회로(104)에 제공한다.
이후에, 로우 캐리 룩어헤드 논리 회로(104)는 ALU(56)의 로우 사이드 위에 나머지 덧셈기들(94, 92)의 각각 제 3 입력에 제공되는 캐리 신호를 발생시킨다. 나머지 캐리 신호들 각각은 상기의 발생 조건과 상기의 전달과 캐리 조건의 프로덕트를 더함으로 계산된다. 본 명세서에는 나타나있지 않지만 부가적인 덧셈기가 A-입력 로우 사이드와 B-입력 로우 사이드에 의해 제공되는 비트 3 내지 비트 7 값의 합계를 계산하기 위해 사용된다.
마지막 덧셈기인 덧셈기(92)로부터 전달과 발생 조건을 수신하자 마자 로우 캐리 룩어헤드 논리 회로(104)는 캐리 신호'C10'을 제공한다. C10 신호는 버퍼(90)에 제공된다. 분할 모드 제어 신호에 응답하여 버퍼(90)는 캐리 신호'C10'를 ALU(56)의 하이 사이드에서 다음 덧셈기로 나아가게 하거나 여기서 발생된 다른 캐리값을 제공한다. 분할 모드 제어 신호는 외부 소스에 의해 실행 유닛(14)에 제공된다. 예를 들어, MEM 명령을 실행하는 동안에 분할 모드 제어 신호는 명령을 디코딩하는 동안 발생된다. 또한 분할 모드 제어 신호는 데이터 처리 시스템(10)의 사용자에 의해 규정된 명령 이외의 다른 소스에 의해 제공될 수 도 있다.

C10 캐리 신호가 Cin 캐리 신호로서 덧셈기(88)에 공급되면, ALU(56)는 이 기술 분야에서 공지되어 있는 전형적인 16비트 연산 논리로 작동한다. 그러나 Cin 캐리 신호가 로우 캐리 룩어헤드 논리 회로(104) 이외의 다른 소스에 의해 제공되 면, ALU(56)는 분할 모드 동작에서 동작한다. 분할 모드 연산에서, ALU(56)는 동시에 2개의 8비트 결과를 제공할 수 있다. MEM 명령을 실행하는 동안에, ALU(56)는 델타 1과 슬로프 1 신호의 프로덕트와 델타 2와 슬로프 2 신호의 프로덕트를 동시에 제공하도록 분할 모드 연산에서 동작한다. 또한 ALU(56)는 분할 모드에서 연산하는 동안에 델타 1과 델타 2 값을 동시에 계산한다. 따라서 분할 모드 연산을 행함으로 ALU(56)는 보다 빠르고 효율적으로 8비트 연산결과를 제공할 수 있다.

또한, ALU(56)에서 로우 상태 플래그 회로(108)는 ALU(56)의 로우 사이드의 연산에 대한 상태 정보를 제공한다. 로우 상태 플래그 회로(108)는 결과 버스(120)를 경유해 전송되는 다수의 상태 신호 S_{low}, V_{low}, H_{low}, Z_{low}를 제공한다. 본 명세서에 도시되어 있지는 않지만, 덧셈기(92) 내지 덧셈기(100)의 각 출력은 로우 상태 플래그 회로(108)에 제공된다. 이와 같은 라우팅은 데이터 처리 기술에서는 잘 공지되어 있고 이 기술 분야에서 숙련된 사람은 쉽게 실행할 수 있는 것이다. 각 상태 신호는 ALU(56)의 로우 사이드에 의해 발생된 다수의 캐리 신호와 합계 신호를 결합함으로써 결정된다.

상기의 발명을 실시함에 있어, S_{low} 신호는 다음과 같은 방정식으로 표현된다.

(6) $S_{low} = S9 + S8 + J_{low}$

V_{low}, N_{low}, H_{low}, Z_{low} 신호는 다음과 같이 표현될 수 있다.

(7) $V_{low} = C6 = C7$

(8) $N_{low} = S8$

(9) $H_{low} = C3$

(10) $Z_{low} = \overline{S_7 + S_6 + S_5 + \ldots\ldots + S_0}$

또한, 덧셈기(92) 내지 덧셈기(100)에 의한 합계 신호 출력은 결과 버스(120)를 경유하여 ALU(56)로부터 전해진다. 덧셈기(92) 내지 덧셈기(100)에 의한 합계 신호 출력 각각은 멀티플라이어 2와 멀티플리캔드 2 신호의 프로덕트의 각각의 비트를 나타낸다. 멀티플라이어 2와 멀티플리캔드 2 신호의 프로덕트는 델타 2와 슬로프 2 신호의 프로덕트와 같다.

연산을 하는 동안 ALU(56)의 하이 오더 사이드는 로우 오더 사이드와 유사하게 작용한다. 'Cin' 캐리 신호는 덧셈기(88)의 제 3 입력에 제공된다. 상기에서와 같이, Cin 신호는 로우 캐리 룩어헤드 논리 회로(104)로부터 C10 캐리 신호에 의해서나 ALU(56)에 외부 소스에 의해 제공되어 질 수 있다. 3개의 입력 신호 각각에 기초하여, 덧셈기(88)는 3개의 출력 신호를 제공한다. 제 1 출력 신호인 S10 신호는 다음과 같다.

(11) $S10 = a10 = b10 = Cin$

제 2 출력 신호는 발생 신호, 즉, G10이고 이것은 다음과 같다.

(12) $G10 = a10 \cdot b10$

제 3 출력 신호는 전달 신호, 즉, P10이고 이것은 다음과 같다.

(13) $P10 = a10 + b10$

G10과 P10 신호에 응답하여, 하이 캐리 룩어헤드 논리 회로(102)는 덧셈기(86)의 제 3 입력에 제공되는 C11 신호를 발생시킨다. C11 신호는 다음과 같이 표현된다.

(14) $C11 = G10 + P10 \cdot C10$

a11, b11, c11 신호들 각각에 응답하여 , 덧셈기(86)는 하이 캐리 룩어헤드 논리 회로(102)에 'G11' 발생 신호와 'P11' 전달 신호를 제공한다.

G11, P11, C11 신호에 응답하여 하이 캐리 룩어헤드 논리 회로(102)는 덧셈기(84)의 제 3 입력에 제공되는 C12 신호를 발생시킨다. C12 신호는 다음과 같은 방정식에 의해 계산된다.

(15) $C12 = G11 + P11(G10+P10)C10$

덧셈기(84)는 하이 캐리 룩어헤드 논리 회로(102)에 'P12' 전달 신호와 'G12' 발생 신호를 제공한다.

계속해서, 하이 캐리 룩어헤드 논리 회로(102)는 ALU(56)의 하이 사이드상의 나머지 덧셈기들(80, 82)의 제 3입력에 제공되는 캐리 신호를 발생시킨다. 나머지 캐리 신호들 각각은 상기의 전달과 캐리의 프로덕트와 상기 발생의 합계로서 계산된다. 본 명세서에는 도시되어 있지 않지만, 부가적인 덧셈기가 A-입력 하이 사이드와 B-입력 하이 사이드에 의해 제공되는 비트 13 내지 비트 17의 합계를 계산하기 위해 사용된다.

마지막 덧셈기인 덧셈기(80)로부터 전달 조건과 발생 조건을 수신 하자마자, 하이 캐리 룩어헤드 논리 회로(102)는 캐리 신호 'C20'을 제공한다. C20 신호는 다음 동작을 위해 결과 버스(120)를 경유하는 ALU(56)로부터의 출력이다.

또한 ALU(56)에서 하이 상태 플래그 회로(106)는 ALU(56)의 하이 사이드의 동작에 대한 상태 정보를 제공한다. 하이 상태 플래그 회로(106)는 각각 결과 버스(120)를 거쳐 전송되는 다수의 상태 신호 S_{high}, V_{high}, N_{high}, H_{high}, Z_{high}를 제공한다. 덧셈기(80) 내지 덧셈기(88)의 출력 각각은 하이 상태 플래그 회로(106)에 제공된다. 이와 같은 라우팅은 데이터 처리 기술에서는 잘 알려진 기술이고 이 기술 분야에 숙련된 사람이라면 쉽게 실행할 수 있는 것이다. 각각의 상태 신호는 ALU(56)의 하이 사이드에 의해 발생되는 다수의 캐리 신호와 합계 신호를 결합함에 의해 결정된다.

본 발명의 이러한 실시예에서 S_{high} 신호는 다음과 같은 방정식에 의해 표현된다.

(16) $S_{high} = S19 + S18 + J_{high}$

V_{high}, N_{high}, H_{high} 신호는 각각 다음과 같이 표현된다.

(17) $V_{high} = C16 = C17$

(18) $N_{high} = S18$
(19) $H_{high} = C13$
(20) $Zhigh = \overline{S_{19} + S_{18} + S_{17} + \ldots\ldots + S_{10}}$

또한, 덧셈기(80) 내지 덧셈기(88)의 각각에 의한 합계 신호 출력은 결과 버스(120)를 경유하여 ALU(56)에 전송되며, 멀티플라이어 1과 멀티플리캔드 1 신호의 프로덕트의 각각의 비트를 나타낸다. 멀티플라이어 1과 멀티플리캔드 1 신호의 프로덕트는 델타 1과 슬로프 1신호의 프로덕트와 같다.

ALU(56)의 하이 사이드와 로우 사이드로부터의 각각의 상태 비트들은 계속해서 결과 선택 논리 회로(46)에 제공된다. 결과 선택 논리 회로(46)는, 델타 1과 슬로프 1 신호의 프로덕트나 델타 2와 슬로프 2 신호의 프로덕트나 포화값 $FF 가 검사되고 있는 멤버쉽 세트에서 멤버쉽의 정도로서 제공되고 있는지를 결정하기 위해 각각 S_{low} 및 S_{high} 신호를 계속해서 사용한다.

시스템 입력값이 멤버쉽 세트의 사다리꼴 내에 해당하는 모든 경우에 있어서, S_{high} 가 어서트되거나 S_{low} 가 어서트되며 또는 S_{high} 와 S_{low} 모두가 어서트될 것이다. S_{high} 와 S_{low} 신호 모두가 어서트되면, 시스템 입력은 포화 포인트($FF)의 멤버쉽의 정도로 검사되는 멤버쉽 세트의 멤버 쉽이다. 따라서 S_{low} 와 S_{high} 신호가 어서트될 때, 결과 선택 논리 회로(46)는 데이터 버퍼(42)의 내용인 시스템 입력이 멤버쉽 세트의 멤버이고 멤버쉽의 정도 $FF를 가지는 것을 나타내는 $FF가 되도록 하기 위해 포스 $FF신호를 어서트한다.

S_{low} 신호가 어서트되고 S_{high} 신호가 니게이트되면, 결과 선택 논리 회로(46)는 데이터 버퍼(42)에 델타 1과 슬로프 1 신호의 프로덕트를 저장하기 위해 선택 신호를 어서트한다. 유사하게 S_{low} 신호가 니게이트되고 S_{high} 신호가 어서트되면, 결과 선택 논리 회로(46)는 데이터 버퍼(42)에 델타 2와 슬로프 2 신호의 프로덕트를 저장하기 위해 선택 신호를 니게이트한다.

S_{high} 및 A_{low} 가 모두 니게이트되면, 시스템 입력값은 경계치에 의해 정의되는 사다리꼴의 멤버쉽 세트 내에 있지 않다. 시스템 입력값이 사다리꼴의 멤버쉽 세트 내에 있지 않는 모든 경우에, N_{mem} 신호가 어서트될 것이다. N_{mem} 신호는 멀티플라이 연산 동안, A-입력 제어 논리 회로(52)가 ALU(56)의 입력에 0이 선택되도록 한다. 결과 선택 논리 회로(46)는 포스 $FF 신호와 선택 신호를 니게이트시키며, 16진수값 $OO은 데이터 버퍼(42)에 저장된다. 따라서 결과 선택 논리 회로(46)는 멤버쉽 계산의 멤버쉽 정도 결과값을 결정하며, 데이터 버퍼(42)에 이 결과를 제공한다. 외부 소스에 의해 실행 유닛(14)에 요구될 때, 데이터 버퍼(42)는 외부 정보 버스(40)를 경유하여 결과를 제공한다.

지금까지는 하나의 소프트웨어 명령을 사용하여 멤버쉽 세트에서 멤버쉽의 정도를 정하기 위한 회로 및 방법이 제시되었다. 상기의 예에서 소프트웨어 명령은 MEM 명령이다. MEM 명령으로 프로그램될 때 데이터 처리 시스템(10)은 매우 빨리 실행될 수 있는 하나의 소프트웨어 명령을 가진 멤버쉽 세트에서 멤버쉽의 정도를 결정할 수 있다. 상기에 제시된 실행 방법은 실행하기 위해 많은 시간이 필요한 상당한 양의 소프트웨어 코드를 필요로 했다. 멤버쉽의 정도를 계산하는 전형적인 소 프트웨어 프로그램에서, MEM 명령은 상당한 수의 소프트웨어 명령을 줄일 수 있다. 예를 들어, MC68HC11 명령 세트를 사용하여 실행되는 소프트웨어 프로그램은 멤버쉽의 정도를 계산하기 위해 31개의 명령들이 필요하다. MEM 명령은 MC68HC11 프로그램의 31개의 명령들을 대치한다. 또한 MEM 명령은 멤버쉽의 정도를 계산하는 상기의 소프트웨어를 이용한 실행보다 더 빨리 71 싸이클들에 결과를 제공한다.

MEM 명령은 이것이 데이터 처리 시스템(10)의 명령 세트의 멤버이기 때문에 보다 빠르고 효율적으로 멤버쉽 정도의 계산을 수행할 수 있다. 명령 세트의 멤버인 명령은 명령을 실행하기 위해 데이터 처리 시스템(10)을 프로그램하는 외부 루틴 보다 더 빨리 실행할 수 있다는 것은 잘 공지되어 있다. 또한, 오퍼랜드 할당 논리 회로(50)에 의해 수행되는 함수는 또한 MEM 명령을 실행하는 데이터 처리 시스템(10)의 수행 능력을 증가시키기 위해 사용된다. 오퍼랜드 할당 논리 회로(50)는 멤버쉽의 정도를 계산하는 동안 사용된 멀티플라이어나 멀티플리캔드 중 적어도 하나는 상위 4비트에서 16진수값 $0을 가져야만 하는 것을 인식한다. 따라서 상위 4비트에서 $O을 가지는 값을 만듦으로써, 멀티프라이어나 곱셈 연산은 상당히 시간적으로 단축되며 더 적절한 방법으로 이루어진다.

또한 8비트 결과가 동시에 얻어지는 분할 모드에서 ALU(56)를 연산함으로써, 더 많은 시간이 단축될 수 있다. ALU(56)는 분할 모드 연산에서는 2개의 8비트 연산 결과를 발생시키고 분할 모드 연산이 아닐때에는 16비트 연산결과를 발생시키는 독특한 회로를 제공한다. 여기서 설명된 예의 속성상, 오직 8비트 결과는 슬로프값과 델타값의 곱셈과 감산하는 동안에 얻어진다. 따라서, ALU(56)는 MEM 명령을 실 행하는 동안에 분할 모드에서 연산될 수 있으며 2개의 감산과 곱셈 연산은 동시에 수행될 수 있다. 따라서 이러한 연산을 수행하기 위해 필요한 시간은 분할 모드 연산에서 더 효율적이다.

카운터는 사용자가 평가되는 퍼지 세트의 수를 셀 수 있도록 순서적으로 실행된다. 본 발명은 최대 포인트값 $FF 슬로프값 $OO에 의해 트리거되는 종료 신호의 사용을 통해 마지막 세트를 나타냄으로써 카운터의 필요성을 제거한다. 이러한 이점은 일련의 멤버쉽 함수에서 입력 값의 멤버쉽의 정도를 계산하기 위해 미국 특허 제 5,295,229 호에 따라 씌어질 수 있는 어셈블리 언어 프로그램을 비교함으로 증명될 수 있다.

```
LDAA SYSTEM_INPUT
LDAB #5 ; 멤버쉽을 위한 루프 카운터
function
L1 MEM ; 하나의 멤버쉽
calculation
DBNZ B,L1 ; 제로(0)가 아니면 카운터와
브랜치를 감소시킴
```

본 발명에 따른 데이터 처리 시스템에서 이러한 어셈블리 언어 프로그램은 다음과 같이 다시 씌어질 수 있다.

LDAA SYSTEM_INPUT
L1 MEM ; 하나의 멤버쉽
calculation
BCC L1 ; 캐리가 클리어하면 브랜치하라
데이터 처리 시스템(10)에서, BCC 명령은 이것이 루프 카운터를 계속해서 감소시킬 필요가 없기 때문에 DBNZ 명령보다 빨리 실행시킬 수 있다. 또한 이러한 시퀀스는 하나의 레지스터도 사용하지 않는다.
이러한 기술은 마지막 멤버쉽 함수가 슬로프값 $OO과 $FF를 가지는 것을 보장할 수 있는 퍼지 시스템에서 유용하다. 그러나, 마지막 멤버쉽 함수를 위해 이러한 제약을 가지지 않는 시스템에서, 제 1 명령 시퀀스는 여전히 사용될 수 있다.
지금까지 설명된 본 발명의 실시는 오직 한가지 실시예에 의해서만 제시되었다. 그러나 다른 많은 실시예가 본원의 함수를 실시하기 위해 존재할 수 있다. 예를 들어, 멤버쉽 세트를 정의하기 위해 필요한 포인트는 상기의 메모리에 저장된 데이터값으로 보다는 MEM 명령의 오퍼랜드로서 데이터 처리 시스템(10)의 사용자에 의해 제공될 수 있다. 또한 ALU(56)는 32비트 결과가 얻어질 수 있도록 실행될 수 있다. 곱셈 연산이나 감산 연산의 경우에 각각의 로우 사이드와 하이 사이드는 분할 모드 연산에서 16비트 결과를 발생시킬 수 있다.
지금까지 본 발명의 원리들에 대해 설명했지만, 이것이 본 발명이 속하는 기술 분야에서 숙련된 사람들이 이해할 수 있도록 한가지 실시예에 의해 자세히 설명된 것이지 본 발명의 범위를 국한시키는 것은 아니다. 다른 실시예에서, MEM 명령은 퍼지 논리 세트가 종점이라기 보다는 피지 논리의 다른 특성을 검출하기 위해 설정된 마지막 퍼지 논리라는 것을 결정할 수 있다. 예를 들어, 다른 시스템은 포 인트 2 값이 포인트 1 값 보다 작도록 정함으로써 퍼지 논리 세트의 출현을 나타낼 수 있다. 이러한 경우에서, MEM 명령은 이러한 다른 특성을 찾을 수 있다. 더구나 MEM 명령은 CPU 내에서나 CPU와 코프로세서(co-processor)에 의해 완전히 수행될 수 있다. 후자의 경우에서 CPU는 명령 디코더를 포함한다고 간주될 수 있고, 코프로세서는 실행 유닛을 포함한다고 간주될 수 있다. 따라서 본원은 청구된 특허청구범위에 의해 본 발명의 정신과 범위를 벗어나지 않는 모든 수정도 포함하고 있는 것이다.

(57) 청구의 범위

청구항 1.
효율적인 퍼지 논리 연산들을 수행하기 위한 데이터 처리 시스템 (10)에 있어서,
(a) 멤버쉽 함수에서 멤버쉽의 정도를 결정하고, 다수의 명령들을 포함하는 명령 세트로부터 처리될 명령(instruction)을 수신하는 입력및 출력을 갖는 명령 디코더(18), 및
(b) 상기 명령 디코더(18) 출력으로부터의 다수의 입력 신호에 응답하여 다수의 산술 및 논리 함수들을 실행하고, 상기 멤버쉽 함수에서 멤버쉽의 정도를 결정하기 위한 상기 명령에 응답하여 적어도 하나의 입력 신호를 포함하는 실행 유닛(14)으로서, 상기 실행 유닛(14)은 또한 상기 멤버쉽 함수가 멤버쉽 함수의 소정 특징을 검사함으로써 다수의 멤버쉽 함수들 중 최종 멤버쉽 함수인지의 여부를 결정하는, 실행 유닛(14)을 구비하며,
(c) 상기 실행 유닛(14)은 조건 코드 레지스터(69)를 포함하고, 상기 멤버쉽 함수가 상기 다수의 멤버쉽 함수들 중 상기 최종 멤버쉽 함수라는 결정에 응답하여 소정의 논리 상태로 상기 조건 코드 레지스터(69)의 선택된 비트를 위치시키는, 효율적인 퍼지 논리 연산들을 수행하기 위한 데이터 처리 시스템.

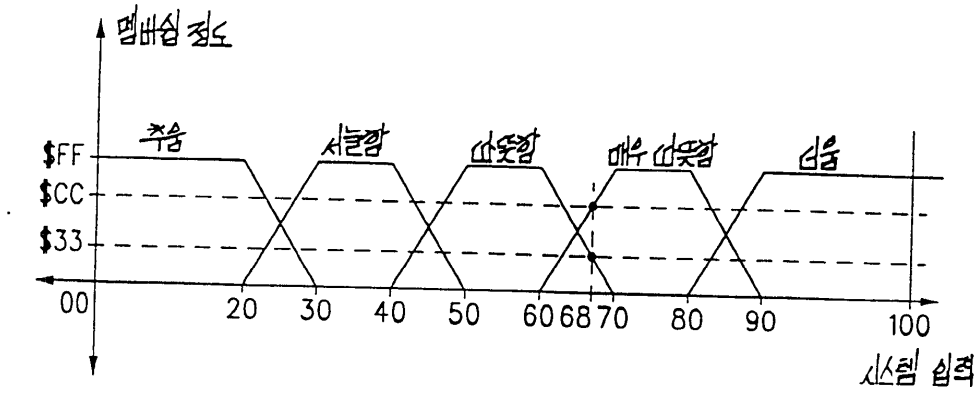
청구항 2.
효율적인 퍼지 논리 연산을 수행하기 위한 데이터 처리 시스템 (10)에 있어서,
명령을 수신하기 위한 입력, 및 다수의 제어 신호들을 제공하기 위한 출력을 가지는 명령 디코더(18), 및
상기 명령 디코더(18)의 출력에 연결된 제어 입력, 및 정보 버스(40)에 연결된 데이터 단자를 가지는 실행 유닛(14)을 구비하며,
상기 명령 디코더(18)는 상기 실행 유닛(14)이 상기 데이터 단자를 경유하여 입력값을 수신하고, 퍼지 멤버쉽 명령에 응답하여 멤버쉽 함수에서 상기 입력값의 멤버쉽의 정도를 결정하도록 상기 실행 유닛(14)을 제어하고,
상기 명령 디코더(18)는 또한 상기 멤버쉽 함수가 다수의 멤버쉽 함수들 중 최종 멤버쉽 함수인지의 여부를 결정하도록 상기 실행 유닛(14)을 제어하고,
상기 실행 유닛(14)은,
캐리 비트를 포함하는 다수의 조건 코드 비트들을 갖고, 종료 신호(151)를 수신하기 위한 제어 입력 단자를 갖는 조건 코드 레지스터(69)로서, 상기 종료 신호(151)의 활성화에 응답하여 상기 캐리 비트를 설정하는, 상기 조건 코드 레지스터(69), 및
상기 명령 디코더(18)에 연결된 제어 입력, 및 상기 종료 신호(151)를 제공하기 위한 출력 단자를 갖는 플래그 발생 논리 회로(48)로서, 상기 멤버쉽 함수가 상기 다수의 멤버쉽 함수들 중 상기 최종 멤버쉽 함수인 것을 결정하는 상기 명령 디코더(18)에 응답하여 상기 종료 신호(151)를 활성화하는 상기 플래그 발생 논리 회로(48)를 더 구비하는, 효율적인 퍼지 논리 연산들을 수행하기 위한 데이터 처리 시스템.

청구항 3.
데이터 처리 시스템(10)에서, 관찰된 가변 입력점의 다수의 멤버쉽 함수들 중 적어도 하나의 멩버쉽 함수에서 멤버쉽의 정도를 결정하는 동안 멤버쉽 함수가 다수의 멤버쉽 함수들 중 최종 멩버쉽 함수인지의 여부를 결정하기 위한방법에 있어서,
(a) 상기 관찰된 가변 입력점의 멤버쉽의 정도를 결정하기 위한 다수의 명령들을 포함하는 명령 세트로부터 명령을

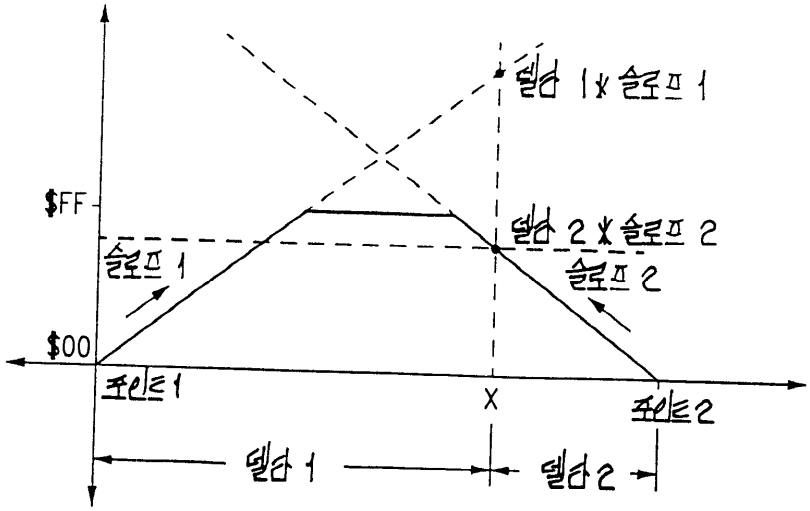
수신하는 단계,
(b) 상기 다수의 멤버쉽 함수들 각각의 경계를 한정하는 제 1 및 제 2 경계치들을 검출하는 단계,
(c) 상기 경계가 상기 다수의 멤버쉽 함수들의 종점(endpoint)을 포함하는지의 여부를 결정하기 위해 상기 다수의 멤버쉽 함수들 각각의 상기 제 1 및 제 2 경계치들 각각을 연속적으로 검사하는 단계,
(d) 상기 관찰된 가변 입력점에 대해 연속적으로 검사하는 상기 단계의 종료를 나타내는 종료 신호를 제공함으로써 검사되고 있는 멤버쉽 함수가 상기 다수의 멤버쉽 함수들 중 상기 최종 멤버쉽 함수인 것을 나타내는 단계, 및
(e) 소정의 논리 상태로 조건 코드 레지스터(69)의 선택된 비트를 위치시키는 단계를 포함하는, 데이터 처리 시스템의 방법.

도면

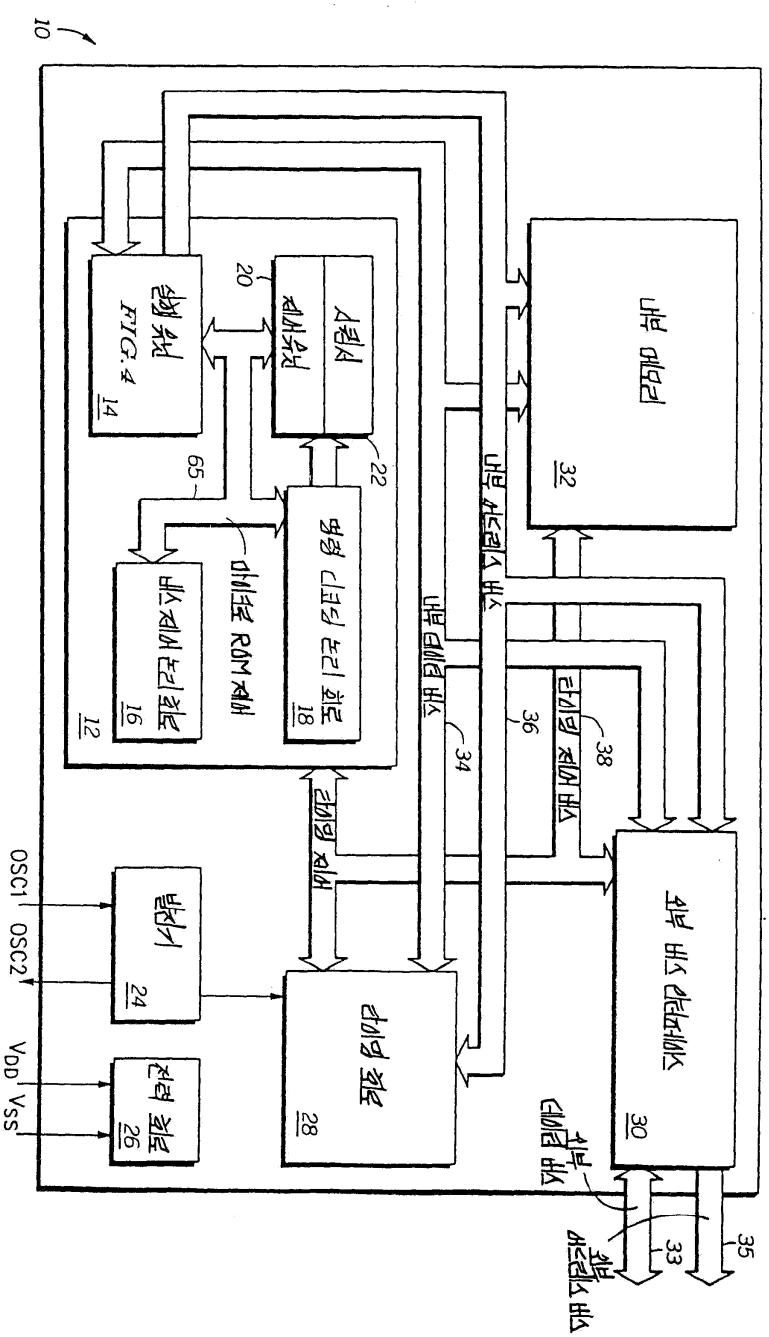
도면1



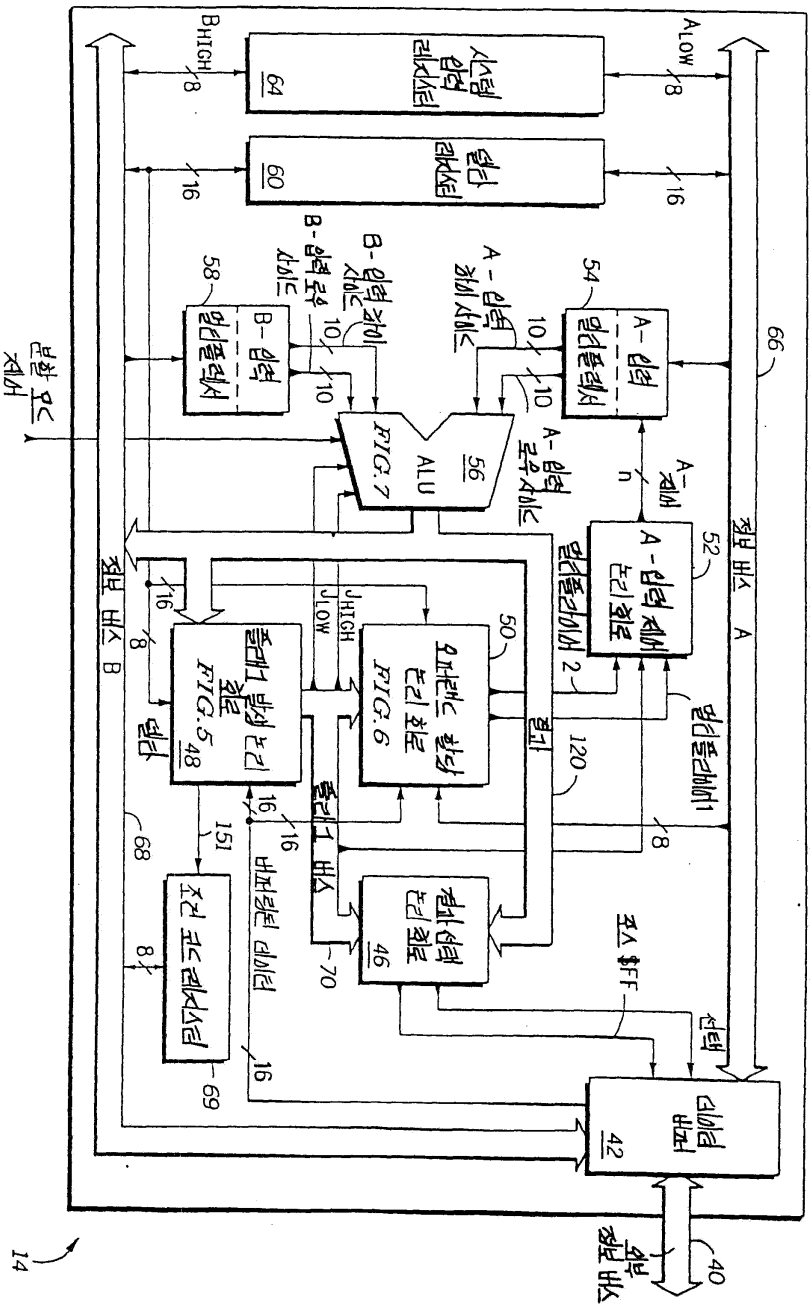
도면2



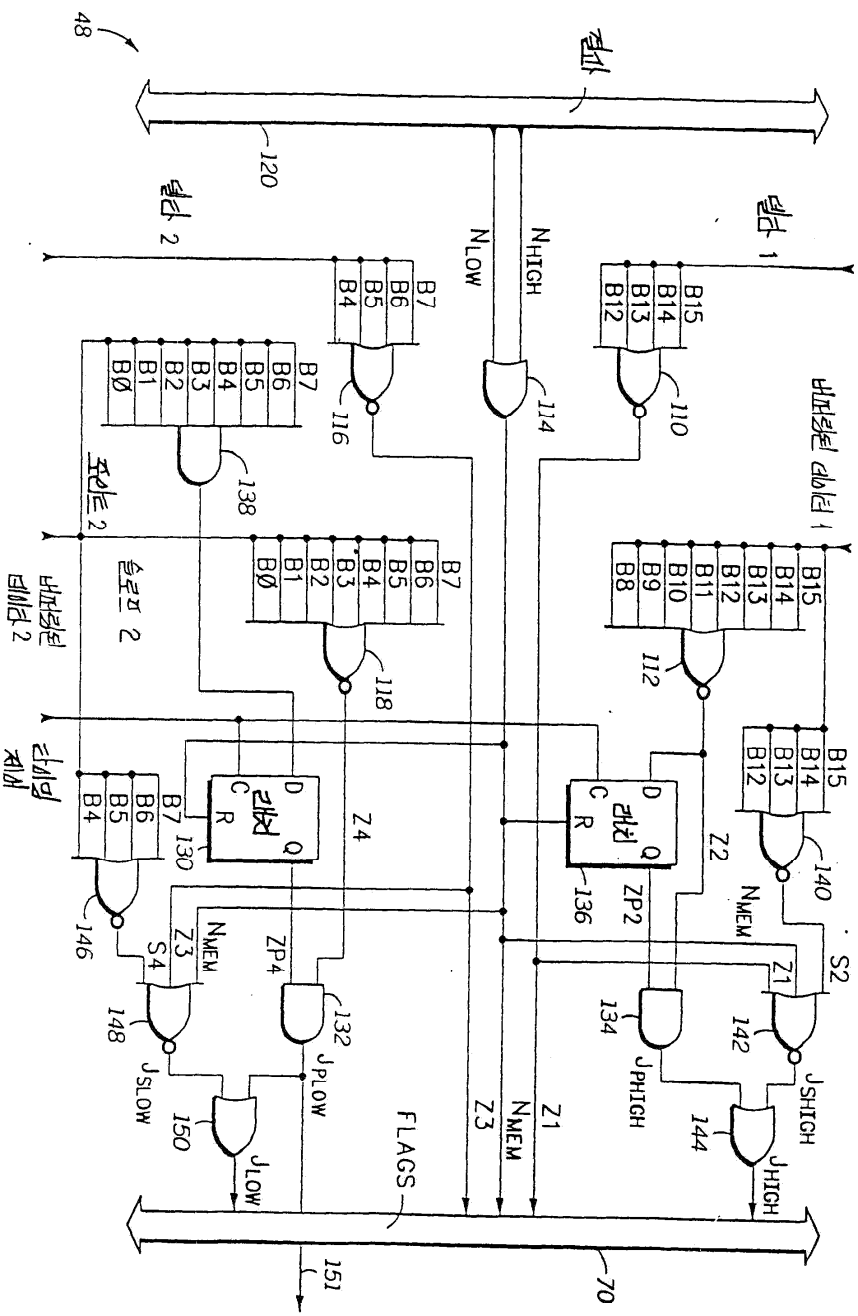
도면3


10
실행 유닛
FIG.4
14
시퀀서
제어유닛
20
22
65
명령 디코딩 논리 회로
18
마이크로 ROM 제어
버스 제어 논리 회로
16
12
내부 메모리
32
내부 어드레스 버스
내부 데이터 버스
34
36
타이밍 제어 버스
38
타이밍 제어
OSC1
OSC2
발진기
24
VDD
VSS
전력 회로
26
타이밍 회로
28
외부 버스 인터페이스
30
외부 데이터 버스
외부 어드레스 버스
33
35

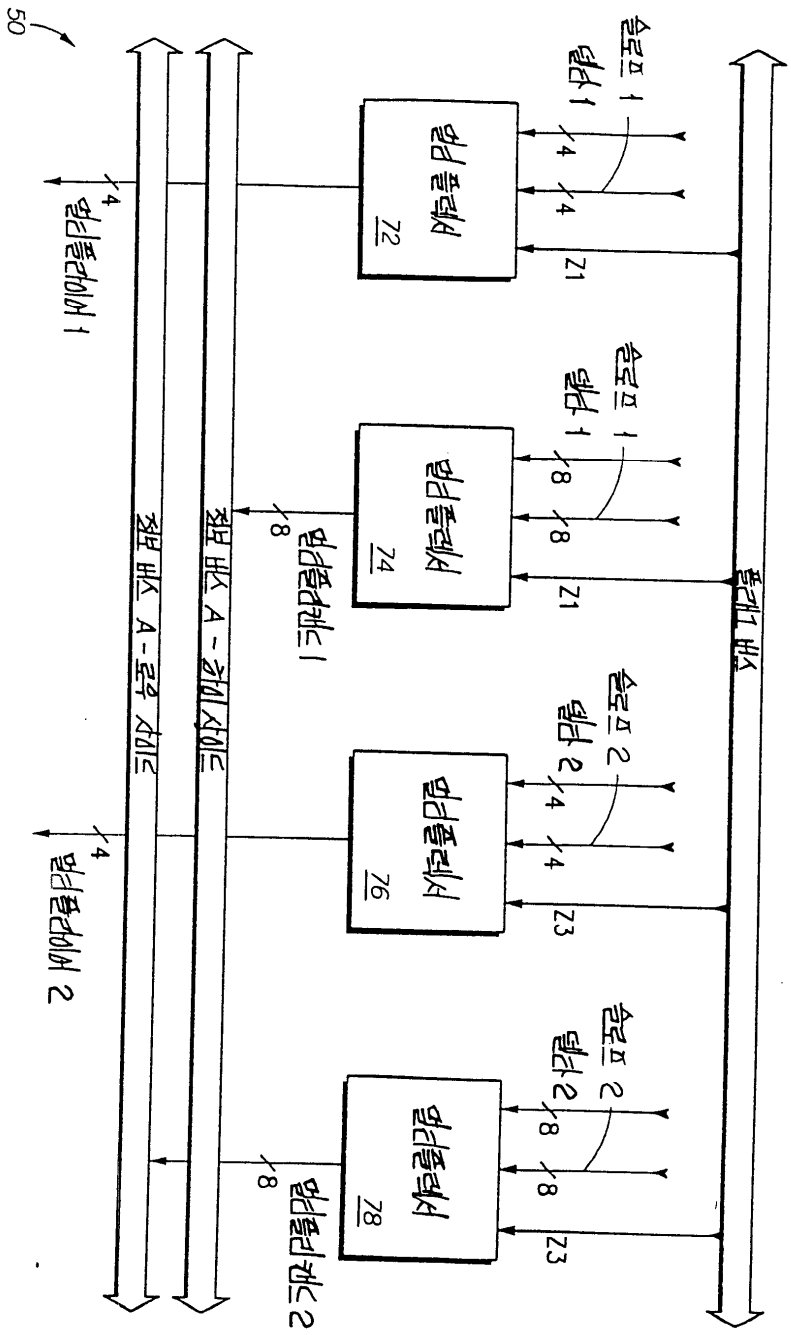
도면4


시스템 입력 레지스터
64
입력 레지스터
60
A-입력 멀티플렉서
54
B-입력 멀티플렉서
58
ALU
56
FIG.7
A-입력 제어 논리 회로
52
멀티플라이어 2
멀티플라이어 1
오퍼랜드 할당 논리 회로
50
FIG.6
플래그 발생 논리 회로
48
FIG.5
결과 생성 논리 회로
46
조건 코드 레지스터
69
마이크로 머신
42
정보 버스 A
정보 버스 B
정보 버스 외부
분할 모드 제어
A-입력 레지스터 출력 버스
B-입력 레지스터 출력 버스
A-입력 레지스터 출력 버스
A-입력 로우 사이드
B-입력 로우 사이드
A-제어
플래그
플래그 버스
버퍼링된 데이터
포스 $FF
선택
14

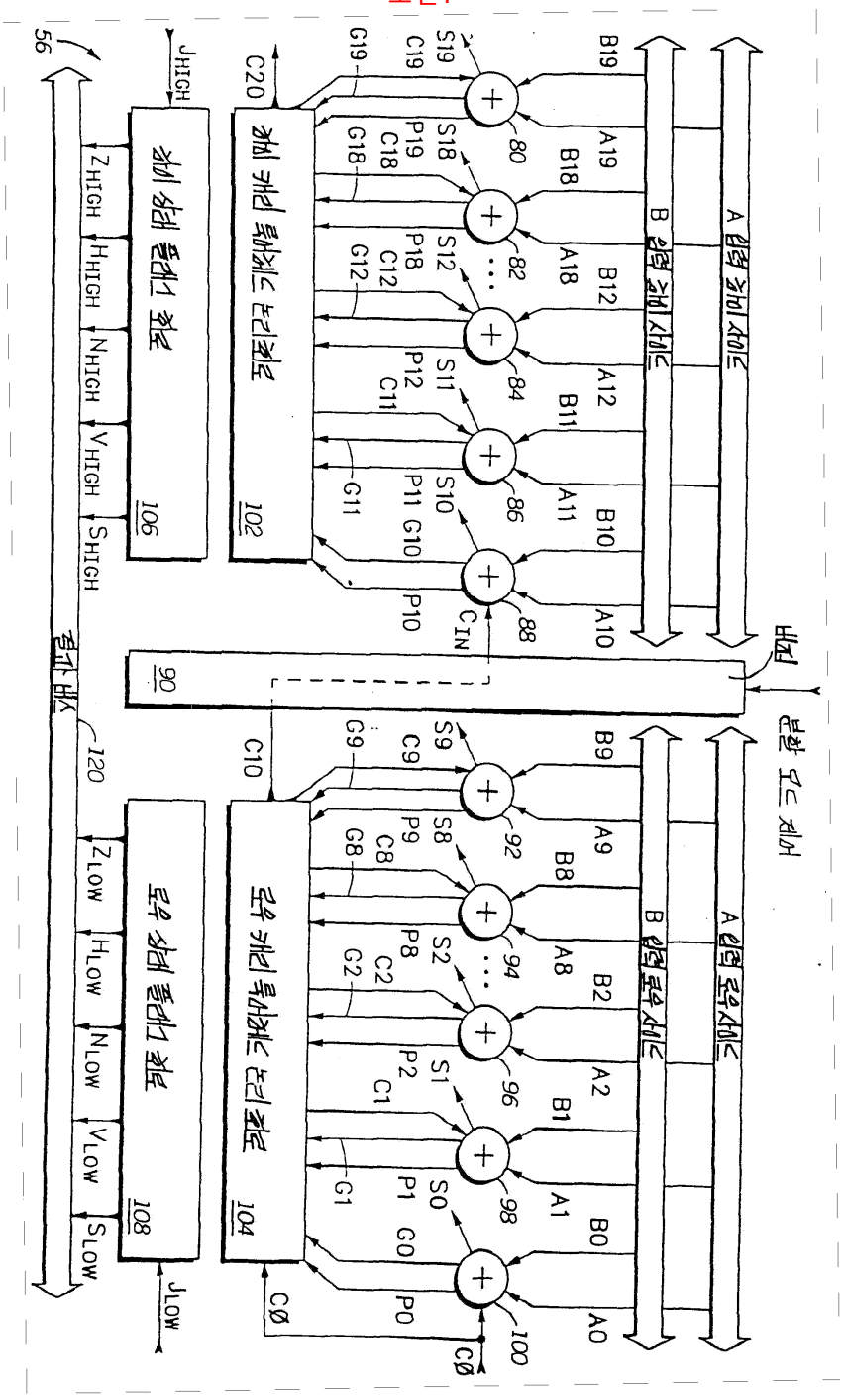
도면5


48
120
110
112
114
116
118
130
132
134
136
138
140
142
144
146
148
150
151
70
B15
B14
B13
B12
B11
B10
B9
B8
B7
B6
B5
B4
B3
B2
B1
B0
NHIGH
NLOW
Z1
Z2
Z3
Z4
ZP2
ZP4
S2
S4
NMEM
JSHIGH
JHIGH
JPHIGH
JPLOW
JLOW
JSLOW
FLAGS

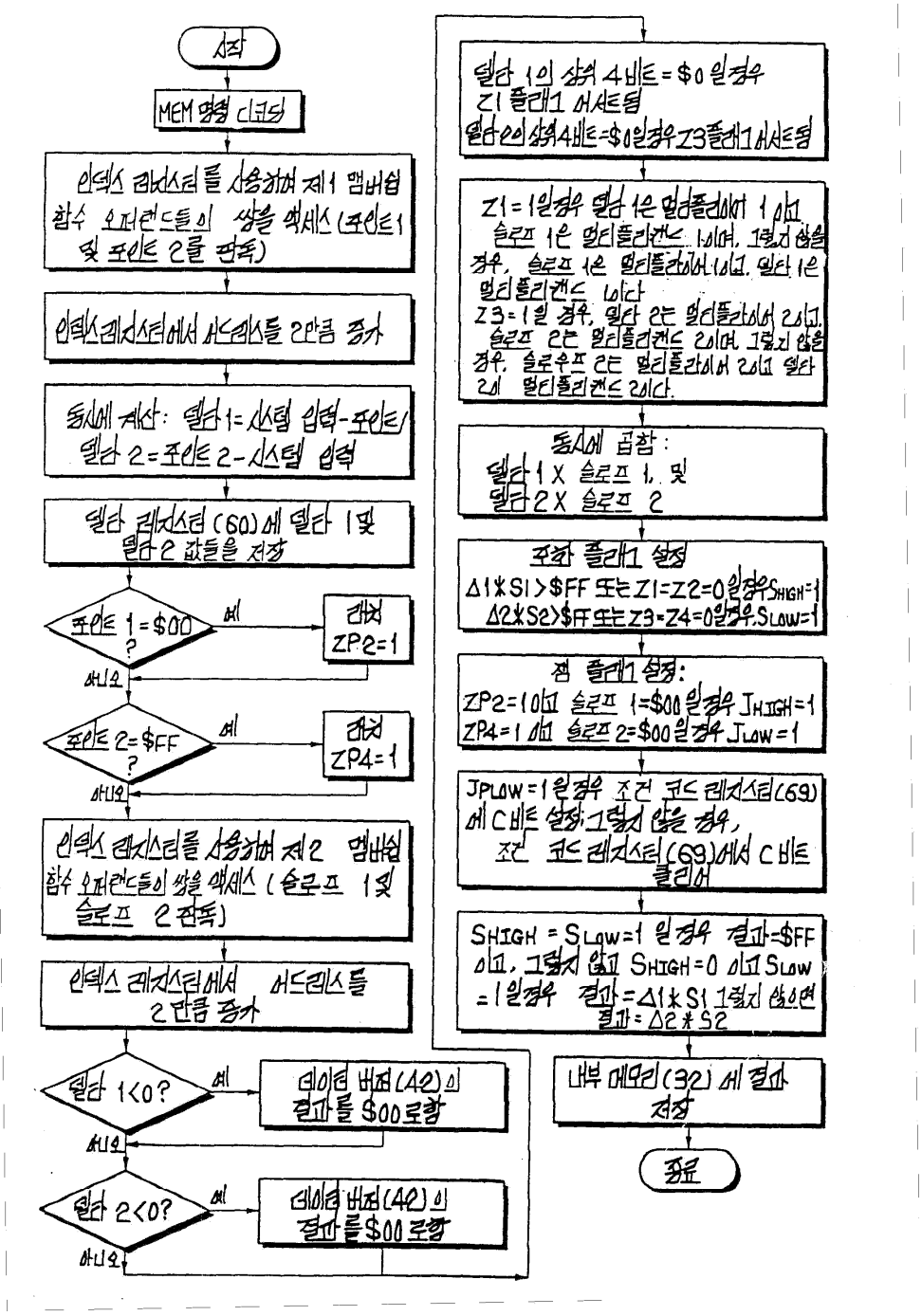
도면6


50
슬로프 1
밸타 1
밸 플렉서
72
4
4
Z1
슬로프 1
밸타 1
밸 플렉서
74
8
8
Z1
슬로프 2
밸타 2
밸 플렉서
76
4
4
Z3
슬로프 2
밸타 2
밸 플렉서
78
8
8
Z3
플래그 버스
밸 플렉서 1
밸 플렉서 2
밸 버스 A-로우 사이드
밸 버스 A-하이 사이드

도면7


A 입력 레이스
B 입력 레이스
A 입력 콕수 시스
B 입력 콕수 시스
비교
분할 모드 제어
90
102
104
106
108
120
56
80
82
84
86
88
92
94
96
98
100
C20
C10
CØ
JHIGH
JLOW
ZHIGH
HHIGH
NHIGH
VHIGH
SHIGH
ZLOW
HLOW
NLOW
VLOW
SLOW
CIN

도면8


시작
MEM 명령 디코딩
인덱스 레지스터를 사용하여 제1 멤버쉽 함수 오퍼랜드들의 쌍을 액세스 (포인트1 및 포인트 2를 판독)
인덱스레지스터에서 서브레스를 2만큼 증가
동시에 계산: 델타1=시스템 입력-포인트1 델타2=포인트2-시스템 입력
델타 레지스터(60)에 델타 1및 델타 2 값들을 저장
포인트 1=$00 ?
예
레지 ZP2=1
아니오
포인트 2=$FF ?
예
레지 ZP4=1
아니오
인덱스 레지스터를 사용하여 제2 멤버쉽 함수 오퍼랜드들의 쌍을 액세스 (슬로프 1및 슬로프 2 판독)
인덱스 레지스터에서 서브레스를 2만큼 증가
델타 1<0?
예
데이터 버퍼(42)의 결과를 $00로함
아니오
델타 2<0?
예
데이터 버퍼(42)의 결과를 $00로함
아니오
델타 1의 상위 4비트=$0 일경우 Z1 플래그 세트됨
델타2의 상위4비트=$0일경우 Z3플래그 세트됨
Z1=1일경우 델타 1은 멀티플리케이터 이고 슬로프 1은 멀티플리컨드 이며, 그렇지 않을 경우, 슬로프 1은 멀티플라이어 이고, 델타 1은 멀티플리컨드 이다
Z3=1일 경우, 델타 2는 멀티플라이어 이고, 슬로프 2는 멀티플리컨드 이며, 그렇지 않은 경우, 슬로우프 2는 멀티플라이어 이고 델타 2는 멀티플리컨드 이다.
동시에 곱함:
델타1 X 슬로프 1, 및
델타2 X 슬로프 2
조합 플래그 설정
Δ1*S1>$FF 또는 Z1=Z2=0일경우 SHIGH=1
Δ2*S2>$FF 또는 Z3=Z4=0일경우 SLOW=1
점 플래그 설정:
ZP2=1이고 슬로프 1=$00 일경우 JHIGH=1
ZP4=1 이고 슬로프 2=$00일경우 JLOW=1
JPLOW=1일경우 조건 코드 레지스터(69)에 C비트 설정 그렇지 않을 경우, 조건 코드 레지스터(69)에서 C비트 클리어
SHIGH=SLOW=1 일 경우 결과=$FF 이고, 그렇지 않고 SHIGH=0 이고 SLOW=1일경우 결과=Δ1*S1 그렇지 않으면 결과=Δ2*S2
내부 메모리(32)에 결과 저장
종료