(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2011년06월20일
(11) 등록번호 10-1042647
(24) 등록일자 2011년06월13일

(51) Int. Cl.
G06F 15/80 (2006.01)
(21) 출원번호 10-2005-7024017
(22) 출원일자(국제출원일자) 2003년12월18일
심사청구일자 2008년07월09일
(85) 번역문제출일자 2005년12월14일
(65) 공개번호 10-2006-0040597
(43) 공개일자 2006년05월10일
(86) 국제출원번호 PCT/GB2003/005548
(87) 국제공개번호 WO 2004/114127
국제공개일자 2004년12월29일
(30) 우선권주장
10/461,880 2003년06월16일 미국(US)
(56) 선행기술조사문헌
KR1019990061571 A
US05636351 A1
전체 청구항 수 : 총 32 항

(73) 특허권자
에이알엠 리미티드
영국 캠브리지 씨비1 9엔제이 체리힌톤 풀번로드 110
(72) 발명자
커셔 다니엘
미국 텍사스 78759 오스틴 #357 모라도 코브 10307
(74) 대리인
이화익

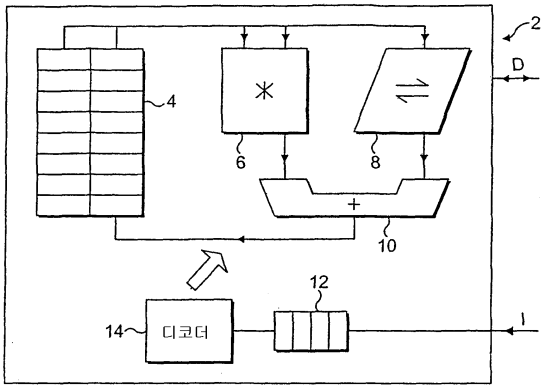
심사관 : 안철용

(54) 단일 명령 복수 데이터 처리 시스템 내의 결과 분할

(57) 요 약

단일 명령 복수 데이터 (SIMD) 방식의 연산을 제공하는 프로세서(2) 내에서, 단일 데이터 처리 명령은 복수 독립 입력 값에 대해 SIMD 방식 처리 연산을 수행하는 처리 로직(4, 6, 8, 10)을 제어하도록 기능하여, 대응하는 입력 값 보다 큰 데이터 폭을 갖는 복수 독립 결과 값을 생성한다. 적절히 제어된 멀티플렉서 형태의 재-분할기는 이들 결과 데이터 값을 별도 레지스터(38, 40) 내에 저장된 상위 비트부와 하위 비트부로 분할하도록 기능한다. 필요로 하는 SIMD 폭 보존 결과 값은 추가의 처리를 요하지 않고 소망의 상위 결과 레지스터(38) 또는 하위 결과 레지스터(40)로부터 독출될 수 있다. 또한, 모든 결과의 보존은 확장된 누적 연산 등에 대해 서와 같이 정확성 향상을 용이하게 한다.

대 표 도 - 도1



특허청구의 범위

청구항 1

단일 데이터 처리 명령에 응답하여 단일 명령 복수 데이터(SIMD) 데이터 처리 연산을 수행하는 장치로서,

승산 명령에 응답하여, 하나 이상의 입력 저장부 내에 저장된 복수의 독립 데이터 값을 입력하여, 각각의 복수의 결과 데이터 값을 출력하는 처리 로직과,

상기 복수의 결과 데이터 값에 응답하여, 상기 각각의 복수의 결과 데이터 값의 각 결과 데이터 값의 상위 비트부분을 상위 결과 저장부 내에 함께 저장하고, 상기 각각의 복수의 결과 데이터 값의 각 결과 데이터 값의 하위 비트부분을 하위 결과 저장부 내에 함께 저장하는 결과 분할기를 포함하고,

상기 출력의 데이터 폭은 상기 입력의 데이터 폭의 2배이고,

상기 결과 분할기는 상기 출력으로부터 상기 복수의 독립 데이터 값의 각각의 데이터 폭의 단위로 서로 다르게 선택하여 상기 상위 결과 저장부와 상기 하위 결과 저장부에 저장하는 것을 특징으로 하는 데이터 처리 연산수행장치.

청구항 2

제1항에 있어서,

상기 처리 로직은, 각 쌍의 독립 데이터 값인, 제1 입력 저장부로부터 취득한 한쌍의 제 1 독립 데이터 값과 제2 입력 저장부로부터 취득한 한쌍의 제2 독립 데이터 값을 함께 승산하도록 실행 가능한 것을 특징으로 하는 데이터 처리 연산 수행 장치.

청구항 3

제2항에 있어서,

상기 처리 로직은, 상기 복수의 결과 데이터 값을 생성하도록 상기 상위 결과 저장부 및 상기 하위 결과 저장부 내에 이미 저장된 값이 독립 데이터 값의 상기 각 쌍으로부터 발생된 값에 누산되도록 작동하는 것을 특징으로 하는 데이터 처리 연산 수행 장치.

청구항 4

제1항 내지 제3항 중 어느 한 항에 있어서,

상기 각 결과 데이터 값의 상기 상위 비트부분 및 하위 비트부분은 상기 결과 데이터 값의 중복되지 않는 연속 부분인 것을 특징으로 하는 데이터 처리 연산 수행 장치.

청구항 5

제2항 또는 제3항에 있어서,

상기 데이터 처리 명령이, 상기 독립 데이터 값이 부호가 첨부된 값인 것을나타낼 때, 상기 처리 로직은 제1 독립 데이터 값과 제2 독립 데이터 값을 승산하는 것에 의해 얻어지는 각 값을 2배화 하도록 실행 가능한 것을 특징으로 하는 데이터 처리 연산 수행 장치.

청구항 6

제1항 내지 제3항 중 어느 한 항에 있어서,

각 입력 저장부는 M개의 독립 N-비트 데이터 값을 저장하는 것을 특징으로 하는 데이터 처리 연산 수행 장치.

청구항 7

제6항에 있어서,

상기 데이터 처리 명령은 상기 독립 데이터 값의 데이터 폭을 지정하는 것을 특징으로 하는 데이터 처리 연산 수행 장치.

청구항 8

제2항 또는 제3항에 있어서,

상기 처리 로직은 독립 데이터 값의 상기 각 쌍을 함께 승산하도록 실행 가능한 정수 승산기를 포함하는 것을 특징으로 하는 데이터 처리 연산 수행 장치.

청구항 9

제1항 내지 제3항 중 어느 한 항에 있어서,

상기 처리 로직은 상기 독립 데이터 값에 대해 포화 데이터 처리 연산을 수행하도록 실행 가능한 것을 특징으로 하는 데이터 처리 연산 수행 장치.

청구항 10

제1항 내지 제3항 중 어느 한 항에 있어서,

상기 결과 분할기는 상기 데이터 처리 명령에 따라 제어되는 복수의 멀티플렉서를 포함하는 것을 특징으로 하는 데이터 처리 연산 수행 장치.

청구항 11

제1항 내지 제3항 중 어느 한 항에 있어서,

상기 장치는 프로세서 코어인 것을 특징으로 하는 데이터 처리 연산 수행 장치.

청구항 12

제1항 내지 제3항 중 어느 한 항에 있어서,

상기 하나 이상의 입력 저장부는,

레지스터 뱅크 레지스터;

전용 레지스터;

버퍼 메모리;

선입선출 버퍼; 및

메모리;

중 하나 이상인 것을 특징으로 하는 데이터 처리 연산 수행 장치.

청구항 13

제1항 내지 제3항 중 어느 한 항에 있어서,

상기 상위 결과 저장부는,

레지스터 뱅크 레지스터;

전용 레지스터;

버퍼 메모리;

선입선출 버퍼; 및

메모리;

중 하나인 것을 특징으로 하는 데이터 처리 연산 수행 장치.

청구항 14

제1항 내지 제3항 중 어느 한 항에 있어서,

상기 하위 결과 저장부는,

레지스터 뱅크 레지스터;

전용 레지스터;

버퍼 메모리;

선입선출 버퍼; 및

메모리;

중 하나인 것을 특징으로 하는 데이터 처리 연산 수행 장치.

청구항 15

제1항 내지 제3항 중 어느 한 항에 있어서,

상기 처리 로직은 각 결과 데이터에 대해 하나 이상의 상위 가드 비트를 생성하도록 실행 가능하며, 상기 결과 분할기는 상기 가드 비트를 가드 비트 저장부 내에 저장하도록 실행 가능한 것을 특징으로 하는 데이터 처리 연산 수행 장치.

청구항 16

제15항에 있어서,

상기 가드 비트 저장부는,

레지스터 뱅크 레지스터;

전용 레지스터;

버퍼 메모리;

선입선출 버퍼; 및

메모리;

중 하나인 것을 특징으로 하는 데이터 처리 연산 수행 장치.

청구항 17

단일 데이터 처리 명령에 응답하여 단일 명령 복수 데이터(SIMD)의 데이터 처리 연산을 수행하는 방법으로서,

승산 명령에 응답하여, 하나 이상의 입력 저장부 내에 저장된 복수의 독립 데이터 값을 입력하고, 각각의 복수의 결과 데이터 값을 출력하는 단계와,

상기 각각의 복수의 결과 데이터 값에 응답하여, 상기 각각의 복수의 결과 데이터 값의 각 결과 데이터 값의 상위 비트 부분을 상위 결과 저장부 내에 함께 저장하고, 상기 각각의 복수의 결과 데이터 값의 각 결과 데이터 값의 하위 비트부분을 하위 결과 저장부 내에 함께 저장하는 것에 의해 상기 결과 데이터 값을 분할하는 단계를 포함하고,

상기 출력의 데이터 폭은 상기 입력의 데이터 폭의 2배이고,

상기 분할하는 단계는 상기 출력으로부터 상기 복수의 독립 데이터 값의 각각의 데이터 폭의 단위로 서로 다르게 선택하여 상기 상위 결과 저장부와 상기 하위 결과 저장부에 저장하는 것을 특징으로 하는 데이터 처리 연산 수행 방법.

청구항 18

제17항에 있어서,

각 쌍의 독립 데이터 값인, 제1 입력 저장부로부터 취득한 한 쌍의 제 1 독립 데이터 값과 제2 입력 저장부로부터 취득한 한 쌍의 제2 독립 데이터 값은 서로 승산되는 것을 특징으로 하는 데이터 처리 연산 수행 방법.

청구항 19

제18항에 있어서,

상기 복수의 결과 데이터 값을 생성하도록, 상기 상위 결과 저장부 및 상기 하위 결과 저장부 내에 이미 저장된 값이 독립 데이터 값의 상기 각 쌍으로부터 발생된 값에 누산되는 것을 특징으로 하는 데이터 처리 연산 수행 방법.

청구항 20

제17항 내지 제19항 중 어느 한 항에 있어서,

상기 각 결과 데이터 값의 상기 상위 비트부분 및 하위 비트부분은 상기 결과 데이터 값의 중복되지 않는 연속 부분인 것을 특징으로 하는 데이터 처리 연산 수행 방법.

청구항 21

제18항 또는 제19항에 있어서,

상기 데이터 처리 명령이, 상기 독립 데이터 값이 부호가 첨부된 값인 것을 나타낼 때, 제1 독립 데이터 값과 제2 독립 데이터 값을 승산하는 것에 의해 얻어지는 각 값을 2배화 하도록 실행 가능한 것을 특징으로 하는 데이터 처리 연산 수행 방법.

청구항 22

제17항 내지 제19항 중 어느 한 항에 있어서,

각 입력 저장부는 M개의 독립 N-비트 데이터 값을 저장하는 것을 특징으로 하는 데이터 처리 연산 수행 방법.

청구항 23

제22항에 있어서,

상기 데이터 처리 명령은 상기 독립 데이터 값의 데이터 폭을 지정하는 것을 특징으로 하는 데이터 처리 연산 수행 방법.

청구항 24

제18항 또는 제19항에 있어서,

독립 데이터 값의 상기 각 쌍은 정수 승산기에 의해 함께 승산되도록 실행되는 것을 특징으로 하는 데이터 처리 연산 수행 방법.

청구항 25

제17항 내지 제19항 중 어느 한 항에 있어서,

상기 독립 데이터 값에 대해 포화 데이터 처리 연산이 수행되는 것을 특징으로 하는 데이터 처리 연산 수행 방법.

청구항 26

제17항 내지 제19항 중 어느 한 항에 있어서,

상기 데이터 처리 명령에 따라 제어되는 복수의 멀티플렉서에 의해 적어도 부분적으로 분할이 수행되는 것을 특징으로 하는 데이터 처리 연산 수행 방법.

청구항 27

제17항 내지 제19항 중 어느 한 항에 있어서,

상기 방법은 프로세서 코어 내에서 수행되는 것을 특징으로 하는 데이터 처리 연산 수행 방법.

청구항 28

제17항 내지 제19항 중 어느 한 항에 있어서,

상기 하나 이상의 입력 저장부는,

레지스터 뱅크 레지스터;

전용 레지스터;

버퍼 메모리;

선입선출 버퍼; 및

메모리;

중 하나 이상인 것을 특징으로 하는 데이터 처리 연산 수행 방법.

청구항 29

제17항 내지 제19항 중 어느 한 항에 있어서,

상기 상위 결과 저장부는,

레지스터 뱅크 레지스터;

전용 레지스터;

버퍼 메모리;

선입선출 버퍼; 및

메모리;

중 하나인 것을 특징으로 하는 데이터 처리 연산 수행 방법.

청구항 30

제17항 내지 제19항 중 어느 한 항에 있어서,

상기 하위 결과 저장부는,

레지스터 뱅크 레지스터;

전용 레지스터;

버퍼 메모리;

선입선출 버퍼; 및

메모리;

중 하나인 것을 특징으로 하는 데이터 처리 연산 수행 방법.

청구항 31

제17항 내지 제19항 중 어느 한 항에 있어서,

처리 로직은 각 결과 데이터에 대해 하나 이상의 상위 가드 비트를 생성하도록 실행 가능하며, 결과 분할기는 상기 가드 비트를 가드 비트 저장부 내에 저장하도록 실행 가능한 것을 특징으로 하는 데이터 처리 연산 수행 방법.

청구항 32

제31항에 있어서,

상기 가드 비트 저장부는,

레지스터 뱅크 레지스터;

전용 레지스터;

버퍼 메모리;

선입선출 버퍼; 및

메모리;

중 하나인 것을 특징으로 하는 데이터 처리 연산 수행 방법.

명 세 서

[0001] 본 발명은 데이터 처리 시스템에 관한 것이다. 보다 상세하게, 본 발명은 단일 명령 복수 데이터(single instruction multiple data; SIMD)의 데이터 처리 시스템 내에 복수 결과 데이터 값을 포함하는 결과의 분할에 관한 것이다.

[0002] 단일 명령 복수 데이터 기능을 갖는 데이터 처리 시스템이 공지되어 있다. 이러한 시스템에서, 레지스터는 전형적으로 조작될 독립 데이터 값을 복수개 포함하고 있다. 예를 들면, 32-비트 레지스터는 2개의 독립 16-비트 데이터 값을 포함할 수 있으며, 이들 데이터 값은 예컨대, 다른 32-비트 레지스터 내에 저장된 2개의 다른 16-비트 데이터 값에 대해, 별도로 가산되거나, 승산되거나, 그렇지 않으면 조합된다. 이러한 SIMD 연산은 디지털 신호 처리 분야에서 보편적인 것이며, 처리 속도가 증가하고 코드 밀도가 감소되는 것을 포함하여 많은 장점을 갖는다.

[0003] 공지된 SIMD 기술의 예로는 Intel Corporation 제품인 Intel 프로세서의 MMX 명령이 있다. MMX 명령에는 각각 4개의 16-비트 데이터 값을 포함하는 레지스터 2개를 함께 승산하는 명령이 포함되어 있다. 16-비트 데이터 값을 다른 16-비트 데이터 값으로 승산하면, 그 결과는 32-비트 데이터 값이 된다. 따라서, MMX SIMD 명령에서 지정된 4쌍의 16-비트 데이터 값을 함께 승산하면, 그 결과는 4개의 32-비트 결과 데이터 값이 된다. 많은 경우, 이러한 연산 수행시 SIMD 포맷과 데이터 크기를 그대로 유지하는 것이 바람직하다. 이를 위해, MMX 명령은 상기의 경우 얻어진 결과가 각 32-비트 결과에서 16개의 최상위 비트가 되는 4개의 16-비트 결과 데이터 값의 형태를 갖는 방식의 명령을 포함하며, 이들 16-비트 값들은 단일 64-비트 레지스터 내에 조합되어, 즉 SIMD 방식의 결과를 얻는다. 다른 방식으로는, 64-비트 레지스터에 조합된 출력으로서 승산 결과에서 4개의 최하위 16-비트를 만들어 내는 별도의 명령들을 갖는 것도 가능하다.

[0004] 본 발명의 일 측면에 따라 제공된 장치로, 데이터 처리 명령에 응답하여 데이터 처리 연산을 수행하는 장치는:

[0005] 상기 데이터 처리 명령에 응답하여, 하나 이상의 입력 저장부 내에 저장된 복수의 독립 데이터 값으로부터 각각의 복수의 결과 데이터 값을 발생시키는 처리 로직과;

[0006] 상기 데이터 처리 명령에 응답하여, 상위 결과 저장부 내에 각 결과 데이터 값의 상위 비트부분을 저장하고, 하위 결과 저장부 내에 각 결과 데이터 값의 하위 비트부분을 저장하는 결과 분할기;

[0007] 를 포함한다.

[0008] 본 발명이 인정하는 바로서, 많은 경우 SIMD 형식의 결과를 생성하는 것이 바람직할 수 있지만, 몇몇 경우, 부적절한 라운딩 오차 등의 불리한 결과를 회피하기 위해서는 결과치의 정확성을 완전하게 유지하는 것이 중요하다는 것이다. 따라서, 본 기술은, 단일 데이터 처리 명령에 응답하여 고밀도의 코드를 형성하고, SIMD 형식의 연산을 복수의 독립 데이터 값에 대해 실행하여, 상위부가 일 저장부에 있고 하위부가 다른 저장부에 있는 SIMD 형식에 복수 결과 데이터 값이 저장되는 시스템을 제공한다. 따라서, SIMD 형식의 결과는 필요한 경우 추가의 처리 과정 없이 즉시 활용 가능하며, 결과의 정확성이 2개의 저장부의 조합에 온전히 유지되어 그로부터 조작될 수 있기 때문에, 정확성이 전적으로 유지되어 전방 자리올림될 수 있다.

[0009] 복수의 독립 입력 데이터 값으로부터 결과 데이터 값을 발생시키도록 처리 로직에 의해 수행된 데이터 처리 연산의 형식은 다양한 다른 형태를 취할 수 있음을 알 것이다. 처리 로직으로의 입력은, 소정의 계산 기법에 따

라 소정 정확도를 갖는 저장하고 있는 독립 데이터 값의 제곱, 또는 독립 데이터 값의 제곱근 등이 결과가 되는 단일 저장부의 컨텐츠일 수 있다. 그러나, 본 발명의 바람직한 실시예에서, 처리 로직은 각 쌍의 독립 데이터 값을, 제1 입력 저장부로부터 취한 쌍의 제1 독립 데이터 값과 제2 입력 저장부로부터 취한 쌍의 제2 독립 데이터 값에 승산하도록 실행 가능하다.

[0010] 이러한 SIMD 방식의 승산 연산은 보편적인 것으로, 정확성이 전적으로 유지되고 SIMD 방식의 결과가 여전히 직접 발생되는 경우 활용되는 본 기술을 필요로 하는 결과의 데이터 폭을 증가시킨다.

[0011] 본 기술은 본 기술에 의해 유지되는 부가 정확성이 누적 방식의 연산에 발생할 수 있는 복수의 라운딩 에러의 누적 효과를 없애는데 도움이 되기 때문에 누적 연산이 승산과 연관되는 상황에 특히 적합하다.

[0012] 상위 비트부분과 하위 비트부분이 다양한 상이한 관계를 가질 수 있음을 알겠지만, 이들이 관련 결과 데이터 값에서 중복되지 않고 연속되는 경우 가장 효율적이고 바람직하다.

[0013] 데이터 처리 명령은 정수 승산 또는 부호가 첨부된 값의 승산 등의 여러가지 다른 형태의 승산 연산을 지정할 수 있다. 그러나, 본 발명은, 지정된 승산이 부호가 첨부된 값의 승산이며, 처리 로직이 각 입력 데이터 값에 부호 표시 비트의 존재를 고려하도록 각 결과 데이터 값을 2배화하도록 실행 가능한 상황에 특히 적합하다. 상기 2배화는 부가 오버헤드(overhead)가 적은 그외의 연산으로 효과적으로 포함될 수 있다.

[0014] 독립 SIMD 데이터 값의 데이터 폭은 변화될 수 있으며, 바람직한 실시예에서, 데이터 처리 명령은 관련 데이터 폭을 지정한다.

[0015] 승산기는 요구되는 특정 상황에 따라 여러가지 형태를 취할 수 있지만, 특히 바람직한 형태는 비교적 간단하고 속도가 빠르면서도 적절한 구성의 다양한 상이한 형태의 연산을 구성할 수 있는 정수 승산기이다.

[0016] 데이터 처리 명령에 의해 지정될 수 있는 처리 연산의 형태의 예로서, 처리는 포화 연산을 수행하도록 하는 것과 같이 선택적일 수 있다.

[0017] 결과 분할기는 결과 데이터 값을 다른 저장부 사이로 분할하도록 기능하며, 바람직한 실시예에서, 이를 위해 복수의 멀티플렉서가 사용된다. 본 기술은 DSP 등의 많은 다른 형태의 데이터 처리 시스템에 적용될 수 있지만, 특히 프로세서 코어에 사용되기에 적합하다.

[0018] 시스템 내의 입력 저장부, 상위 결과 저장부, 하위 결과 저장부 및 저장부는 다양한 다른 형태를 가질 수 있지만, 레지스터 뱅크 레지스터, 전용 레지스터, 버퍼 메모리, 선입선출(first-in-first-out) 버퍼 및 메모리 일부(예, 캐시, 메인, 벌크 등) 중의 하나 이상임을 알 것이다. 이들 상이한 형태의 저장부들은 상이한 저장부가 다른 형태를 갖는 복합 상황에 사용될 수 있다. 저장부를 위한 레지스터 대신에 메모리 또는 버퍼를 사용하는 경우, 조작될 일련의 데이터 값의 스트리밍이 편리하게 제공될 수 있다.

[0019] 본 기술에 쉽게 적용되는 방식으로 계산될 결과의 범위를 증가시키는 방법으로서, 바람직한 실시예에서는 포화 연산의 콘텍스트에 이용될 수 있는 것과 같은 하나 이상의 상위 가드 비트를 발생시키기도 한다. 이들 가드 비트는 결과 분할이 가드 비트를 저장하는 자체 저장부를 가질 수 있다.

[0020] 본 발명의 다른 측면에 따라 제공된 방법으로서, 데이터 처리 명령에 응답하여 데이터 처리 연산을 수행하는 방법은:

[0021] 상기 데이터 처리 명령에 응답하여, 하나 이상의 입력 저장부 내에 저장된 복수의 독립 데이터 값으로부터 각각의 복수의 결과 데이터 값을 발생시키는 단계와;

[0022] 상기 데이터 처리 명령에 응답하여, 상위 결과 저장부 내에 각 결과 데이터 값의 상위 비트부분을 저장하고, 하위 결과 저장부 내에 각 결과 데이터 값의 하위 비트부분을 저장하는 것에 의해 상기 결과 데이터 값을 분할하는 단계;

[0023] 를 포함한다.

[0024] 이하, 하기의 첨부 도면을 참조로 본 발명의 실시예를 단지 예시의 목적으로 설명한다. 도면에서,

[0025] 도 1은 본 기술을 활용할 수 있는 방식의 프로세서 코어를 개략적으로 나타내며;

[0026] 도 2는 다양한 SIMD 데이터 포맷을 개략적으로 나타내며;

[0027] 도 3은 다양한 데이터 폭에 대해 본 발명에 따른 입력 데이터 값과 출력 데이터 값 사이의 관계를 개략적으로

나타내며;

[0028] 도 4는 도 1의 프로세서 코어 내의 데이터 처리 경로의 일부를 개략적으로 나타내며;

[0029] 도 5는 본 기술에 따라 결과 데이터 값을 분할하기 위한 멀티플렉스 장치를 나타내며;

[0030] 도 6은 본 기술에 따른 승산 누적 연산의 다른 형태를 개략적으로 나타낸다.

[0031] 도 1은 영국, 캠브리지의 ARM Limited의 제품과 같은 프로세서 코어(2)를 도시한다. 프로세서 코어(2)는 레지스터 뱅크(4), 승산기(6), 시프터(8), 및 데이터 처리 데이터 경로의 일부를 형성하는 가산기(10)를 포함한다. 데이터 처리 명령은 명령 파이프라인(12)으로 받아들여져서, 이로부터 명령 디코더(14)에 의해 디코딩되어, 프로세서(2) 내의 다른 회로 소자의 동작을 제어하는 제어 신호가 발생된다. 프로세서(2)는 전형적으로 많은 추가의 회로 소자를 포함하지만, 이들 소자는 간결성을 위해 도시되지 않음을 알 것이다. 도 1의 예에서, 입력 데이터 값은 레지스터 뱅크(4) 내의 레지스터로부터 읽혀지고, 레지스터 뱅크(4)의 레지스터로 다시 기록되어진다. 다른 실시예에서, 입력 값과 결과 값은 전용 레지스터, 버퍼 메모리, 선입선출 버퍼 및 범용 메모리 등의 상이한 방식의 저장부를 대상으로 읽혀지고 기록된다. 이들은 다른 선택예로서 이용될 수 있고, 다양한 혼합 조합으로 이용될 수 있다. 이들 상이한 선택예는 도 1에 도시되지 않는다.

[0032] 도 2는 여러가지 상이한 SIMD 데이터 포맷을 나타낸다. 도 1에 도시된 데이터 경로의 데이터 폭은 이러한 데이터 폭을 지원하도록 변형된 버전의 ARM 프로세서에서는 64-비트일 수 있다. 이 데이터 경로는 64-비트 워드의 전체 길이를 비-SIMD 모드로 조작할 수 있다. 이 예에서, 다양한 SIMD 모드는 2개의 32-비트 데이터 값, 4개의 16-비트 데이터 값, 또는 8개의 8-비트 데이터 값을 조작한다. SIMD 모드에서, 데이터 값은 서로 독립적이며, 도 1의 프로세서(2)의 데이터 경로는 SIMD 데이터 값의 크기에 따라 구성되어, 예컨대, 적절한 위치 등에서 자리 올림 연쇄(carry chain)를 차단하는 것에 의해, 이들 데이터 값을 별도로 처리한다. SIMD 방식의 연산을 수행하는 데이터 경로의 정합(adaption)은 그 자체가 공지된 것으로 여기에 추가 설명하지 않는다.

[0033] 도 3은 본 기술에 따른 상이한 SIMD 데이터 폭 모드의 입력 데이터 값과 결과 데이터 값 사이의 관계를 나타낸다. 실시예 (i)에서, 입력 데이터 값은 제1 64-비트 레지스터에 저장된 2개의 32-비트 입력 값(A0, A1)과, 제2 레지스터에 저장된 2개의 32-비트 입력 값(B0, B1)으로 이루어진다. 이 실시예에서, 처리 명령에 의해 지정된 데이터 처리 연산은 SIMD 승산이므로, 32-비트 값(A0)은 32-비트 값(B0)에 의해 승산되고, 32-비트 값(A1)은 32-비트 값(B1)에 의해 승산된다. 이들 승산은 각각 A0B0 및 A1B1의 64-비트 결과를 산출한다. 이들 2개의 결과의 최상위 32 비트는 상위 결과 레지스터(17)에 기록된다. 이들 2개의 결과의 최하위 32 비트는 하위 결과 레지스터(18)에 기록된다. 상이한 레지스터(17, 18)에 기록된 2개의 부분은 중복되지 않는 연속부이다.

[0034] 실시예 (ii), (iii)은 유사한 것으로, 각각은 SIMD 승산 명령에 의해 승산의 적용을 받고, 총 결과의 절반 상위이거나 총 결과의 절반 하위인 상이한 레지스터에 각각 결과 데이터 값을 생성시키는 16-비트 입력 값과 8-비트 입력 값에 관한 것이다.

[0035] 동일한 데이터 폭의 추가의 SIMD 방식의 연산의 승산에 의해 생성된 결과를 이용하여 추가의 처리를 계속하는 것을 원한다면, 보다 상위 결과 레지스터(17)를 직접 독출하여 상기 추가 연산을 위한 입력으로서 이용할 수 있다. 코드 밀도, 속도, 전력 소비 등의 향상을 위해 시프팅(shifting) 또는 재-배열은 필요하지 않다. 특별히 바람직한 경우는 상위 결과 레지스터(17)와 하위 결과 레지스터(18)가 누적 연산을 위한 수신지로서 이용되어, 연속적인 승산의 결과가 이들 레지스터에 축적되고, 하위 결과 레지스터(18)에 보존된 하위 결과 값이 보다 정확한 결과를 산출하고 라운딩 오차를 방지하도록 연속적으로 업데이트될 수 있다. 따라서, 본 기술은 단일 명령을 이용하여 올바른 데이터 폭 값으로 직접 액세스되도록 하며, 결과값의 전 데이터 폭의 유지를 통해 여전히 정확성을 보유하도록 한다.

[0036] 도 4는 도 1의 데이터 경로의 일부를 보다 상세히 나타낸 개략도이다. SIMD 정수 승산기(20)는 레지스터 뱅크(4)의 각 레지스터로부터 얻은 2개의 64-비트 입력 값을 공급받는다. 이들 입력 값들은 단일의 64-비트*64-비트 비-SIMD 연산, 또는 전술한 3가지 SIMD 방식 연산중 하나를 나타낼 수 있다. SIMD 승산기(20)는 자리올림 연쇄 등에 적절한 차단을 제공하여 독립 입력값과 결과 출력 값을 적절히 분할한다. SIMD 승산기(20)로부터의 출력은 자리올림 보존 포맷이다. 시스템이 부호 표시 소수 모드(signed fractional mode)로 동작시, 멀티플렉서(22,24)로 공급된 신호를 나타내는 소수 모드는 최상위 위치에 추가 부호 비트를 보상하는 방식으로 수치값을 이배화하는 것과 같이 자리올림 보존 출력을 1 비트 위치 만큼 시프팅하도록 기능한다. 가산기(26)는 SIMD 승산기(20)로부터의 자리올림 보존 출력을, 보존 및 자리올림 레지스터(28, 30)로부터 재순환된 부분 누적된 값

또는 멀티플렉서(32, 34)에 의해 선택된 레지스터 뱅크(40)의 레지스터(D, C)로부터의 128-비트 값에 가산하는 기능을 한다. 멀티플렉서(32, 34)는 그 다양한 값이 도 4의 하부의 테이블에 보여지는 누산 제어 신호에 의해 제어된다. 시스템은 소스 레지스터 파일로부터 누산하고, 누적없이 승산하거나, 벡터형 연산 도중과 같이 이전의 부분 연산된 결과에 누산하거나, 누산 값을 위한 소스로서 레지스터 뱅크를 바이패스하거나 하도록 배열될 수 있다.

[0037] 주어진 처리 동작에서 승산 및 가산 연산을 완료하면, 레지스터(28, 30)로부터의 최종 128-비트 보존 및 자리올림 값이 가산기(36)로 전달되어, 그곳에서 함께 가산되어 결과의 통상적인 128-비트 대표값을 얻는다. 승산 및 가산은 파이프라인 방식의 연산일 수 있다. 가산기(36)의 출력은 레지스터(A, B)로 부터의 64-비트 입력값에 비해 비트폭이 2배이다. 따라서, SIMD 결과값은 독립 SIMD 입력 값의 폭의 2배값을 갖는다. 가산기(36)의 출력은 본 실시예에서 도 5에 도시된 다양한 멀티플렉서의 형태를 갖는 결과 분할기로 공급된다.

[0038] 도 5에서, 상위 결과 레지스터(38)는 상위부가 되는 각 결과값의 선택된 부분을 수용한다. 하위 결과 레지스터(40)는 결과값의 대응하는 하위부를 수용한다. 제어 신호(B, H, W 및 L)는 사용되는 SIMD 데이터 폭(바이트, 반워드, 워드 및 긴 워드)을 나타낸다. 이들 값 중 하나는 어느 때고 한 번 "1"로, 나머지는 "0"으로 확정된다. 이들 폭 지정 신호는 관련 멀티플렉서의 다양한 입력 사이를 선택하도록, 인접한 각 멀티플렉서에 주어진 논리적 표현에 따라 도 5에 도시된 멀티플렉서를 제어한다. 그 제어 신호에 의해 제어되는 도 5의 멀티플렉서의 전체 동작은 가산기(36)에 의한 128 비트 출력 사이에서부터 선택/재분할하여, 도 3의 다른 실시예에서 나타낸 바와 같은 상위 결과 레지스터(38)와 하위 결과 레지스터(40)의 컨텐츠를 형성한다.

[0039] 도 1의 디코더(14)로 공급되어 도 4 및 도 5의 회로를 예시된 방법으로 제어하는 프로그램 명령은 사용되는 데이터 폭이 비-SIMD 전 데이터 폭이거나 다양한 SIMD 데이터 폭 중 하나인지 여부를 지정하는 파라미터들을 포함하는 구문을 갖는다. 프로그램 명령은 또한 누산의 수행 여부 및 이 누산이 추가의 레지스터 값 또는 "내부" 부분 결과를 이용하여 수행되는지 여부를 지정한다.

[0040] 도 5의 2개의 결과 레지스터(38, 40) 이외에, 가드 레지스터가 또한 제공될 수 있다. 이 가드 레지스터 내로는 확장된 버전의 누산 결과로부터 연산된 가드 비트가 공급된다. 예컨대, 16-비트 SIMD 데이터 값이 승산 누적 연산에 사용되면, 누산기는 누산 값의 오버플로가 가드 비트 내에 수용되도록, 2개 또는 4개의 가드 비트 제공 여부에 따라 예컨대, 34 또는 36 비트로서, 32 비트 보다 클 수 있다. 이 실시예에서, 가드 비트는 별도의 가드 비트 레지스터로 분할될 수 있으며, 이 형태에서 가드 비트 레지스터는, 하위 결과 레지스터가 결과값의 하부에 가드 비트를 제공하고 상위 결과 레지스터가 정상적으로 필요한 SIMD 폭 보존 데이터 값을 제공하도록, 결과의 최상단 위치에 가드 비트를 제공하는 것으로 간주될 수 있다.

[0041] 도 6은 다중 데이터 포맷에 의한 다중 누적 연산으로 누적 레지스터 결과(stacked register result)를 제공하는 것을 개략적으로 도시한다.

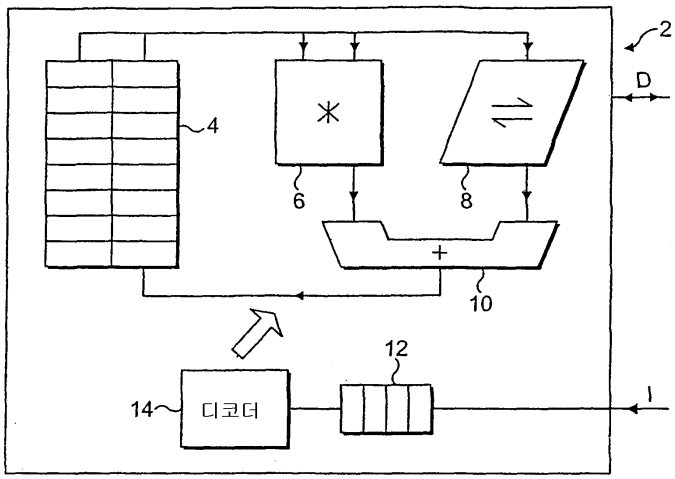
[0042] 레지스터(A, B)는 이 경우 4개의 16-비트 량(A0-A3 및 B0-B3)을 보유하는 64 비트 SIMD 레지스터이다. 이들 레지스터를 함께 승산한 결과는 각각 32 비트 폭 까지일 수 있는 4개의 결과의 벡터이다.

[0043] 4개의 32 비트 승산 결과는 각각 2개의 32 비트량을 보유하는 다른 2개의 레지스터(C, D)에 보유된 4개의 32 비트값에 누산될 수 있다.

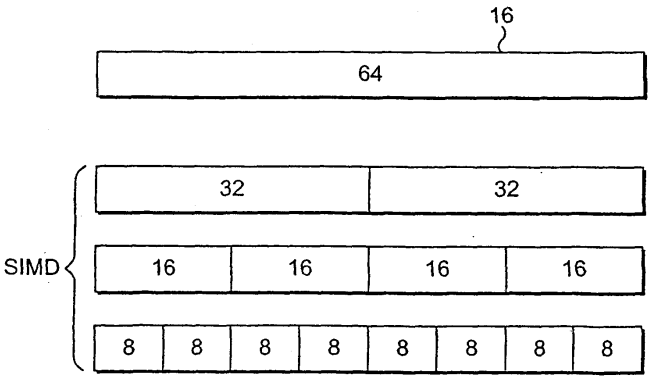
[0044] 가산의 결과는 누적 포맷(stacked format)의 레지스터(RL, RH)에 저장될 수 있다.

도면

도면1

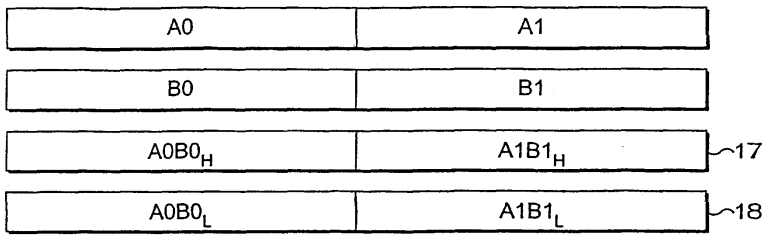


도면2

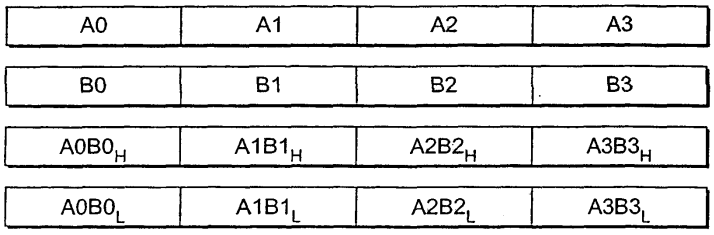


도면3

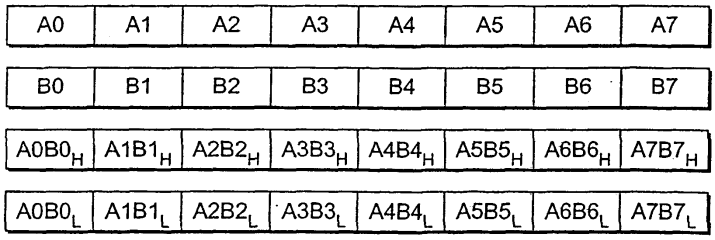
(i)


A0
A1
B0
B1
A0B0H
A1B1H
17
A0B0L
A1B1L
18

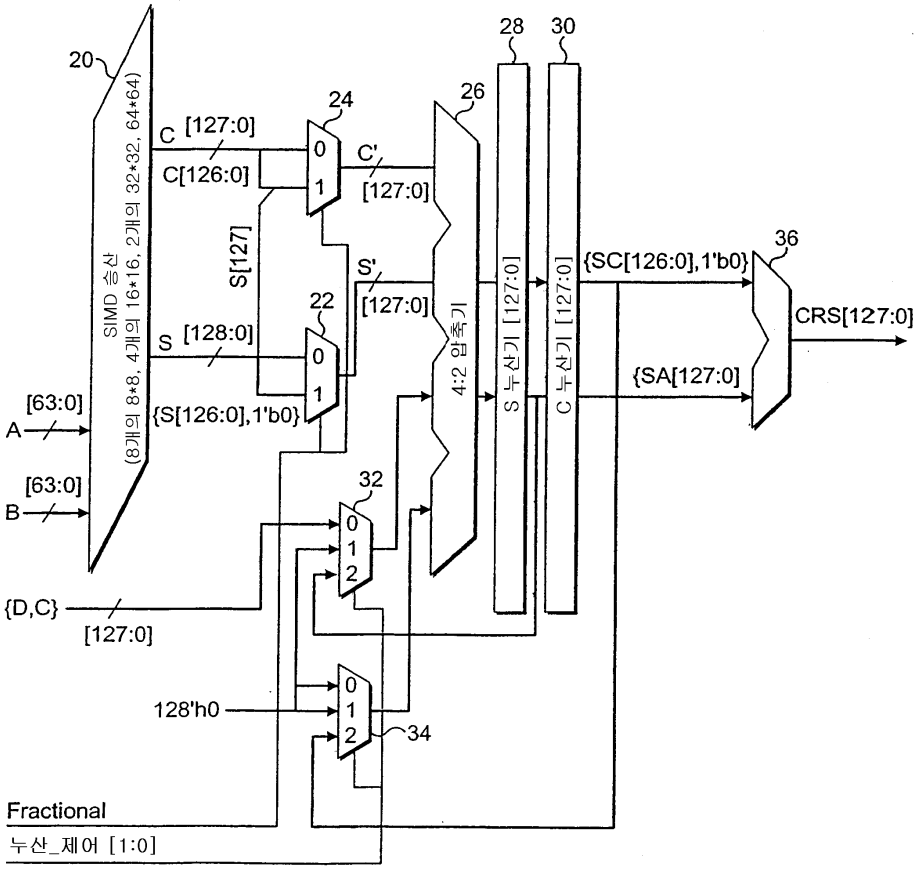
(ii)


A0
A1
A2
A3
B0
B1
B2
B3
A0B0H
A1B1H
A2B2H
A3B3H
A0B0L
A1B1L
A2B2L
A3B3L

(iii)


A0
A1
A2
A3
A4
A5
A6
A7
B0
B1
B2
B3
B4
B5
B6
B7
A0B0H
A1B1H
A2B2H
A3B3H
A4B4H
A5B5H
A6B6H
A7B7H
A0B0L
A1B1L
A2B2L
A3B3L
A4B4L
A5B5L
A6B6L
A7B7L

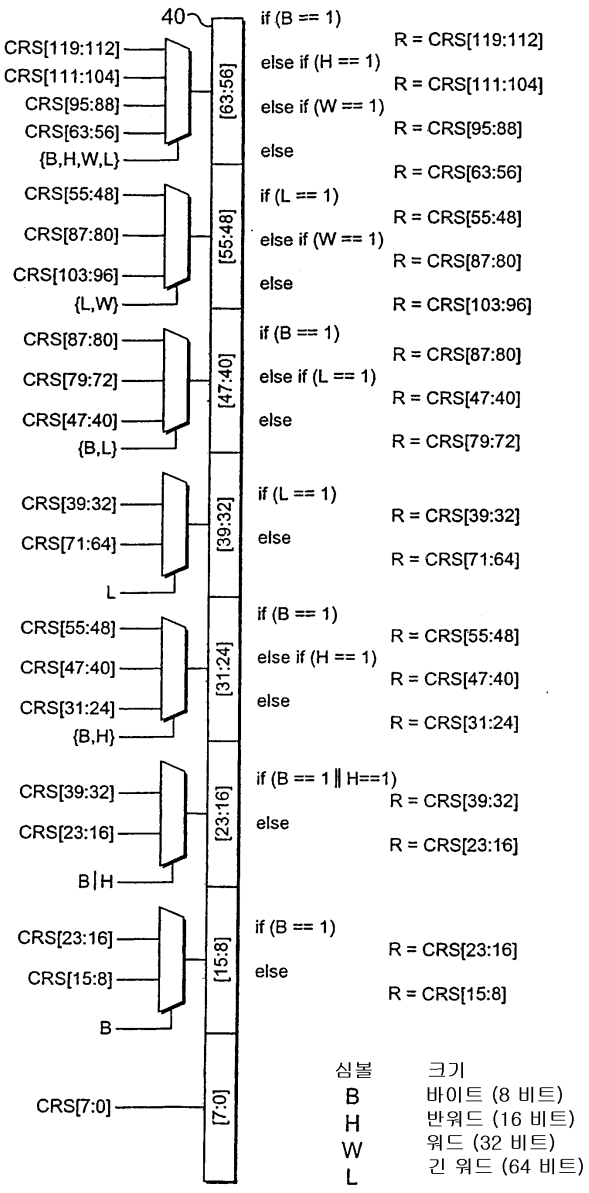
도면4



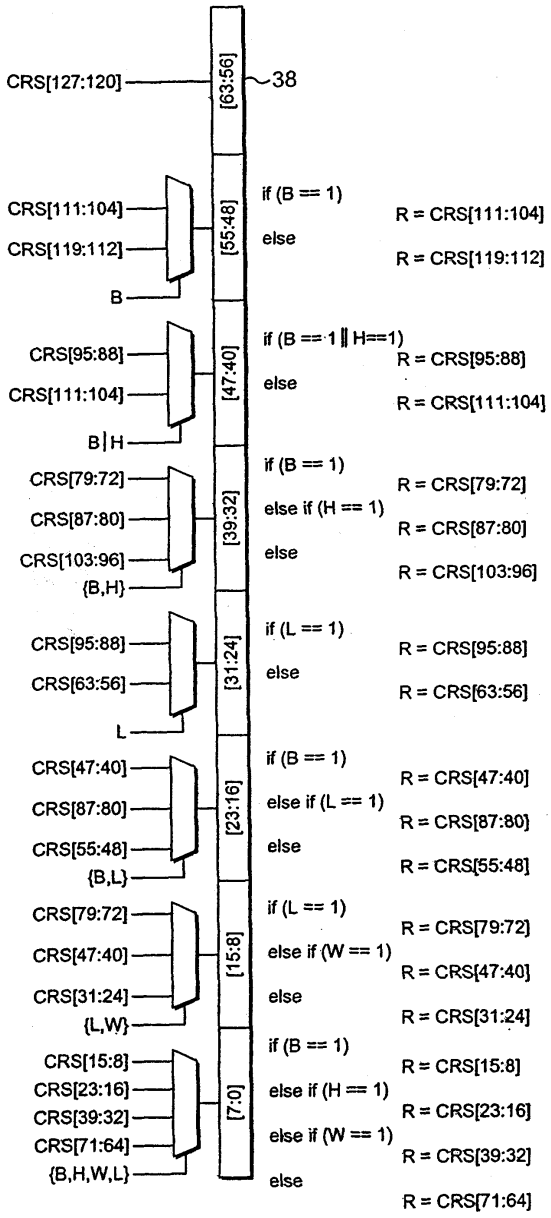
누산_제어

[0]	[1]	동작
0	0	레지스터 파일로부터 소스 누산
1	0	승산(비-누산)
0	1	이전 결과에 누산- 레지스터 뱅크 소스 바이패스
1	1	위법

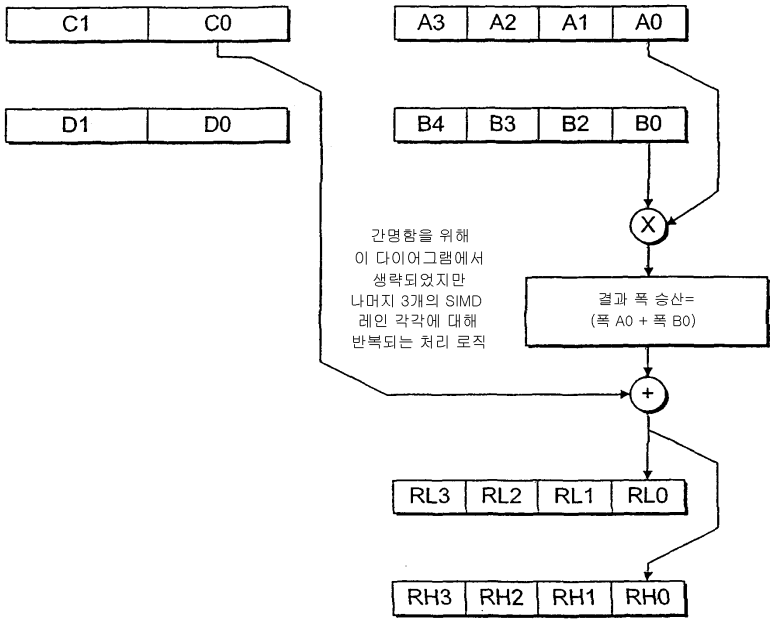
도면5


40
CRS[119:112]
CRS[111:104]
CRS[95:88]
CRS[63:56]
{B,H,W,L}
[63:56]
if (B == 1)
R = CRS[119:112]
else if (H == 1)
R = CRS[111:104]
else if (W == 1)
R = CRS[95:88]
else
R = CRS[63:56]
CRS[55:48]
CRS[87:80]
CRS[103:96]
{L,W}
[55:48]
if (L == 1)
R = CRS[55:48]
else if (W == 1)
R = CRS[87:80]
else
R = CRS[103:96]
CRS[87:80]
CRS[79:72]
CRS[47:40]
{B,L}
[47:40]
if (B == 1)
R = CRS[87:80]
else if (L == 1)
R = CRS[47:40]
else
R = CRS[79:72]
CRS[39:32]
CRS[71:64]
L
[39:32]
if (L == 1)
R = CRS[39:32]
else
R = CRS[71:64]
CRS[55:48]
CRS[47:40]
CRS[31:24]
{B,H}
[31:24]
if (B == 1)
R = CRS[55:48]
else if (H == 1)
R = CRS[47:40]
else
R = CRS[31:24]
CRS[39:32]
CRS[23:16]
B|H
[23:16]
if (B == 1 || H==1)
R = CRS[39:32]
else
R = CRS[23:16]
CRS[23:16]
CRS[15:8]
B
[15:8]
if (B == 1)
R = CRS[23:16]
else
R = CRS[15:8]
CRS[7:0]
[7:0]
심볼
크기
B
바이트 (8 비트)
H
반워드 (16 비트)
W
워드 (32 비트)
L
긴 워드 (64 비트)

도면5a


CRS[127:120]
[63:56]
38
CRS[111:104]
CRS[119:112]
B
[55:48]
if (B == 1)
R = CRS[111:104]
else
R = CRS[119:112]
CRS[95:88]
CRS[111:104]
B|H
[47:40]
if (B == 1 || H==1)
R = CRS[95:88]
else
R = CRS[111:104]
CRS[79:72]
CRS[87:80]
CRS[103:96]
{B,H}
[39:32]
if (B == 1)
R = CRS[79:72]
else if (H == 1)
R = CRS[87:80]
else
R = CRS[103:96]
CRS[95:88]
CRS[63:56]
L
[31:24]
if (L == 1)
R = CRS[95:88]
else
R = CRS[63:56]
CRS[47:40]
CRS[87:80]
CRS[55:48]
{B,L}
[23:16]
if (B == 1)
R = CRS[47:40]
else if (L == 1)
R = CRS[87:80]
else
R = CRS[55:48]
CRS[79:72]
CRS[47:40]
CRS[31:24]
{L,W}
[15:8]
if (L == 1)
R = CRS[79:72]
else if (W == 1)
R = CRS[47:40]
else
R = CRS[31:24]
CRS[15:8]
CRS[23:16]
CRS[39:32]
CRS[71:64]
{B,H,W,L}
[7:0]
if (B == 1)
R = CRS[15:8]
else if (H == 1)
R = CRS[23:16]
else if (W == 1)
R = CRS[39:32]
else
R = CRS[71:64]

도면6



RL0 = 하부 (C0 + A0*B0)
RH0 = 상부 (C0 + A0*B0)

RL1 = 하부 (C1 + A1*B1)
RH1 = 상부 (C1 + A1*B1)

RL2 = 하부 (D0 + A2*B2)
RH2 = 상부 (D0 + A2*B2)

RL3 = 하부 (D1 + A3*B3)
RH3 = 상부 (D1 + A3*B3)