(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2017년09월25일
(11) 등록번호 10-1781057
(24) 등록일자 2017년09월18일

(51) 국제특허분류(Int. Cl.)
G06F 9/30 (2017.01) *G06F 15/80* (2006.01)
G06F 9/38 (2006.01)
(52) CPC특허분류
G06F 9/30036 (2013.01)
G06F 15/8053 (2013.01)
(21) 출원번호 10-2016-7015679
(22) 출원일자(국제) 2014년11월14일
심사청구일자 2017년08월14일
(85) 번역문제출일자 2016년06월13일
(65) 공개번호 10-2016-0085873
(43) 공개일자 2016년07월18일
(86) 국제출원번호 PCT/US2014/065825
(87) 국제공개번호 WO 2015/073915
국제공개일자 2015년05월21일
(30) 우선권주장
14/082,073 2013년11월15일 미국(US)
(56) 선행기술조사문헌
WO2006106342 A1*
US20070061550 A1*
US20080140750 A1*
*는 심사관에 의하여 인용된 문헌

(73) 특허권자
퀄컴 인코포레이티드
미국 92121-1714 캘리포니아주 샌 디에고 모어하우스 드라이브 5775
(72) 발명자
칸, 라힐
미국 92121-1714 캘리포니아 샌 디에고 모어하우스 드라이브 5775
(74) 대리인
특허법인 남앤드남

전체 청구항 수 : 총 24 항 심사관 : 김경완

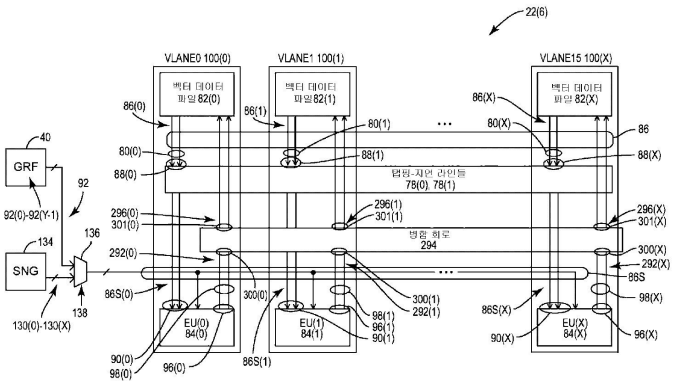
(54) 발명의 명칭 **실행 유닛들과 벡터 데이터 메모리 사이에 병합 회로를 갖는 벡터 프로세싱 엔진, 및 관련된 방법**

(57) 요 약

벡터 데이터 메모리에 저장되는 출력 벡터 데이터의 실시간적인 병합을 제공하기 위해서 실행 유닛들과 벡터 데이터 메모리 사이의 데이터 흐름 경로들에서 병합 회로를 이용하는 벡터 프로세싱 엔진(VPE)들이 개시된다. 관련된 벡터 프로세싱 명령들, 시스템들 및 방법들이 또한 개시된다. VPE에서 실행 유닛들과 벡터 데이터 메모리

(뒷면에 계속)

대 표 도



사이의 데이터 흐름 경로들에 병합 회로가 제공된다. 병합 회로는 출력 벡터 데이터 샘플 세트가 저장되기 위해서 실행 유닛들로부터 벡터 데이터 메모리로 출력 데이터 흐름 경로들을 통해 제공되고 있는 동안 실시간적으로 벡터 프로세싱 연산들을 수행한 결과로서 실행 유닛들로부터의 출력 벡터 데이터 샘플 세트를 병합하도록 구성된다. 병합된 출력 벡터 데이터 샘플 세트는 실행 유닛들에서 수행될 후속 벡터 프로세싱 연산들을 지연시킬 수 있는 추가적인 사후-프로세싱 단계들을 필요로 하지 않고도 벡터 데이터 메모리에 병합된 형태로 저장된다.

(52) CPC특허분류

G06F 9/30032 (2013.01)

G06F 9/3887 (2013.01)

G06F 9/3897 (2013.01)

명 세 서

청구범위

청구항 1

벡터 프로세싱 연산을 실행하는 적어도 하나의 실행 유닛에 의해 생성된 결과적 출력 벡터 데이터 샘플 세트를 실시간으로(in-flight) 병합하도록 구성되는 벡터 프로세싱 엔진(VPE)으로서,

적어도 하나의 벡터 데이터 파일;

적어도 하나의 실행 유닛; 및

적어도 하나의 병합 회로를 포함하고,

상기 적어도 하나의 벡터 데이터 파일은,

벡터 프로세싱 연산을 위해 적어도 하나의 입력 데이터 흐름 경로에서 입력 벡터 데이터 샘플 세트를 제공하고, 그리고

저장될 적어도 하나의 병합된 결과적 출력 벡터 데이터 샘플 세트를 적어도 하나의 출력 데이터 흐름 경로로부터 수신하도록 구성되고,

상기 적어도 하나의 실행 유닛은 상기 적어도 하나의 입력 데이터 흐름 경로에서 제공되고, 상기 적어도 하나의 실행 유닛은,

상기 적어도 하나의 입력 데이터 흐름 경로를 통해 상기 입력 벡터 데이터 샘플 세트를 수신하고, 그리고

상기 적어도 하나의 출력 데이터 흐름 경로를 통해 상기 결과적 출력 벡터 데이터 샘플 세트를 제공하기 위해 상기 입력 벡터 데이터 샘플 세트에 대해 상기 벡터 프로세싱 연산을 실행하도록 구성되고, 그리고

상기 적어도 하나의 병합 회로는,

상기 결과적 출력 벡터 데이터 샘플 세트를 수신하고,

상기 결과적 출력 벡터 데이터 샘플 세트가 상기 적어도 하나의 벡터 데이터 파일에 저장되는 것 없이 상기 적어도 하나의 병합된 결과적 출력 벡터 데이터 샘플 세트를 제공하기 위해 상기 수신된 결과적 출력 벡터 데이터 샘플 세트의 결과적 출력 벡터 데이터 샘플들을 병합하고 – 상기 적어도 하나의 병합 회로는, 상기 결과적 출력 벡터 데이터 샘플들을 가산(adding)하는 것, 상기 결과적 출력 벡터 데이터 샘플들 중에서 최대 벡터 데이터 샘플을 결정하는 것, 또는 상기 결과적 출력 벡터 데이터 샘플들 중에서 최소 벡터 데이터 샘플을 결정하는 것에 의해 상기 결과적 출력 벡터 데이터 샘플들을 병합하도록 구성됨 –,

복수의 래치들(latches) 중 선택된 하나 또는 그 초과의 래치들에 상기 적어도 하나의 병합된 결과적 출력 벡터 데이터 샘플 세트를 저장하고, 그리고

상기 적어도 하나의 출력 데이터 흐름 경로를 통해 상기 선택된 하나 또는 그 초과의 래치들에 저장된 상기 적어도 하나의 병합된 결과적 출력 벡터 데이터 샘플 세트를 제공하도록 구성되고, 그리고

상기 적어도 하나의 병합 회로는 상기 복수의 래치들에 대응하는 복수의 병렬 선택기들을 더 포함하고, 상기 적어도 하나의 병합 회로는 상기 복수의 래치들 중 선택된 하나 또는 그 초과의 래치들에 상기 적어도 하나의 병합된 결과적 출력 벡터 데이터 샘플 세트를 저장하도록 상기 복수의 선택기들을 제어하게 구성되는, 벡터 프로세싱 엔진(VPE).

청구항 2

제 1 항에 있어서,

상기 적어도 하나의 벡터 데이터 파일은,

상기 벡터 프로세싱 연산을 위해 상기 적어도 하나의 입력 데이터 흐름 경로에서 상기 적어도 하나의

벡터 데이터 파일의 폭의 입력 벡터 데이터 샘플 세트를 제공하고, 그리고

저장될 상기 적어도 하나의 벡터 데이터 파일의 폭의 상기 적어도 하나의 병합된 결과적 출력 벡터 데이터 샘플 세트를 상기 적어도 하나의 출력 데이터 흐름 경로로부터 수신하도록 구성되는, 벡터 프로세싱 엔진(VPE).

청구항 3

제 1 항에 있어서,

상기 적어도 하나의 벡터 데이터 파일은,

상기 적어도 하나의 입력 데이터 흐름 경로에서 적어도 하나의 벡터 데이터 파일 출력을 통해 상기 입력 벡터 데이터 샘플 세트를 제공하고, 그리고

상기 적어도 하나의 출력 데이터 흐름 경로에서 적어도 하나의 벡터 데이터 파일 입력을 통해 상기 적어도 하나의 병합된 결과적 출력 벡터 데이터 샘플 세트를 수신하도록 추가로 구성되고,

상기 적어도 하나의 실행 유닛은,

상기 적어도 하나의 입력 데이터 흐름 경로에서 적어도 하나의 실행 유닛 입력을 통해 상기 입력 벡터 데이터 샘플 세트를 수신하고, 그리고

상기 적어도 하나의 입력 데이터 흐름 경로에서 적어도 하나의 실행 유닛 출력을 통해 상기 결과적 출력 벡터 데이터 샘플 세트를 제공하기 위해 상기 입력 벡터 데이터 샘플 세트를 코드 시퀀스 벡터 데이터 샘플 세트와 곱하도록 구성되며, 그리고

상기 적어도 하나의 병합 회로는,

상기 적어도 하나의 실행 유닛으로부터 상기 적어도 하나의 입력 데이터 흐름 경로에서 적어도 하나의 병합 회로 입력을 통해 상기 결과적 출력 벡터 데이터 샘플 세트를 수신하고, 그리고

상기 적어도 하나의 출력 데이터 흐름 경로에서 적어도 하나의 병합 회로 출력을 통해 상기 적어도 하나의 병합된 결과적 출력 벡터 데이터 샘플 세트를 제공하도록 추가로 구성되는, 벡터 프로세싱 엔진(VPE).

청구항 4

제 3 항에 있어서,

상기 코드 시퀀스 벡터 데이터 샘플 세트는 적어도 하나의 CDMA 칩 코드 시퀀스를 포함하는, 벡터 프로세싱 엔진(VPE).

청구항 5

제 1 항에 있어서,

상기 적어도 하나의 병합 회로는 상기 적어도 하나의 병합된 결과적 출력 벡터 데이터 샘플 세트를 제공하기 위해 상기 결과적 출력 벡터 데이터 샘플 세트의 상기 결과적 출력 벡터 데이터 샘플들 중 적어도 두 개를 가산하도록 구성되는 적어도 하나의 가산기를 포함하는, 벡터 프로세싱 엔진(VPE).

청구항 6

제 5 항에 있어서,

상기 적어도 하나의 가산기는 가산기 트리(tree)에서 제공되는 복수의 가산기들을 포함하고, 상기 복수의 가산기들의 각각은 상이한 비트 폭을 각각 갖는 복수의 추가 병합된 결과적 출력 벡터 데이터 샘플 세트들을 제공하도록 구성되는, 벡터 프로세싱 엔진(VPE).

청구항 7

제 5 항에 있어서,

상기 적어도 하나의 병합 회로는 적어도 하나의 병합된 결과적 출력 벡터 데이터 샘플 세트들 중 하나를 선택하

도록 구성되는 병합 선택기를 더 포함하는, 벡터 프로세싱 엔진(VPE).

청구항 8

제 1 항에 있어서,

상기 적어도 하나의 병합 회로는 상기 적어도 하나의 병합된 결과적 출력 벡터 데이터 샘플 세트를 제공하기 위해, 두 개의 결과적 출력 벡터 데이터 샘플들 중에서 최대 벡터 데이터 값을 갖는 상기 두 개의 결과적 출력 벡터 데이터 샘플들 중의 하나를 선택하도록 구성되는 적어도 하나의 최대 벡터 데이터 샘플 선택기를 포함하는, 벡터 프로세싱 엔진(VPE).

청구항 9

제 8 항에 있어서,

상기 적어도 하나의 최대 벡터 데이터 샘플 선택기는 상이한 비트 폭을 각각 갖는 복수의 최대 병합된 결과적 출력 벡터 데이터 샘플 세트들을 제공하도록 각각 구성된 복수의 최대 값 데이터 샘플 선택기들을 포함하는, 벡터 프로세싱 엔진(VPE).

청구항 10

제 1 항에 있어서,

상기 적어도 하나의 병합 회로는 상기 적어도 하나의 병합된 결과적 출력 벡터 데이터 샘플 세트를 제공하기 위해, 두 개의 결과적 출력 벡터 데이터 샘플들 중에서 최소 벡터 데이터 값을 갖는 상기 두 개의 결과적 출력 벡터 데이터 샘플들 중의 하나를 선택하도록 구성되는 적어도 하나의 최소 벡터 데이터 샘플 선택기를 포함하는, 벡터 프로세싱 엔진(VPE).

청구항 11

제 10 항에 있어서,

상기 적어도 하나의 최소 벡터 데이터 샘플 선택기는 상이한 비트 폭을 각각 갖는 복수의 최소 병합된 결과적 출력 벡터 데이터 샘플 세트들을 제공하도록 각각 구성된 복수의 최소 값 데이터 샘플 선택기들을 포함하는, 벡터 프로세싱 엔진(VPE).

청구항 12

제 1 항에 있어서,

상기 적어도 하나의 병합 회로는 상기 결과적 출력 벡터 데이터 샘플들을 선택적으로 병합하기 위해서 프로그램가능 병합 데이터 경로 구성 입력에 기초하여 재구성되도록 구성가능한, 벡터 프로세싱 엔진(VPE).

청구항 13

제 12 항에 있어서,

상기 적어도 하나의 병합 회로는 상기 적어도 하나의 실행 유닛에 의해 실행될 상기 VPE의 각각의 클록 사이클에서 상기 결과적 출력 벡터 데이터 샘플들을 선택적으로 병합하기 위해서 상기 프로그램가능 병합 데이터 경로 구성 입력에 기초하여 재구성되도록 추가로 구성되는, 벡터 프로세싱 엔진(VPE).

청구항 14

제 12 항에 있어서,

상기 적어도 하나의 병합 회로는 상기 적어도 하나의 실행 유닛에 의해 실행될 다음 벡터 명령에 따라 상기 결과적 출력 벡터 데이터 샘플들을 선택적으로 병합하기 위해서 상기 프로그램가능 병합 데이터 경로 구성 입력에 기초하여 재구성되도록 추가로 구성되는, 벡터 프로세싱 엔진(VPE).

청구항 15

제 1 항에 있어서,

상기 적어도 하나의 실행 유닛은 상기 적어도 하나의 실행 유닛을 위한 프로그램가능 입력 데이터 흐름 경로 구성에 기초하여 상기 입력 벡터 데이터 샘플 세트로부터의 입력 벡터 데이터 샘플들의 상이한 비트 폭들을 프로세싱하도록 구성가능한, 벡터 프로세싱 엔진(VPE).

청구항 16

제 1 항에 있어서,

크로스바(crossbar) 회로를 더 포함하고,

상기 크로스바 회로는 상기 적어도 하나의 병합된 결과적 출력 벡터 데이터 샘플 세트를 상기 복수의 래치들 중 선택된 하나 또는 그 초과의 래치들에 라우팅하도록 구성되어서, 상기 크로스바 회로가, 상기 적어도 하나의 벡터 데이터 파일에서의 저장에 앞서 병합 벡터 프로세싱 연산들의 상이한 반복들 중에 상기 적어도 하나의 병합된 결과적 출력 벡터 데이터 샘플 세트가 상기 복수의 래치들에 스택(stack)되는 것을, 허용하도록 구성되게 하는, 벡터 프로세싱 엔진(VPE).

청구항 17

적어도 하나의 실행 유닛에 의해 생성된 결과적 출력 벡터 데이터 샘플 세트를 실시간으로 병합하기 위한 장치로서,

적어도 하나의 벡터 데이터 파일로부터 벡터 프로세싱 연산을 위한 적어도 하나의 입력 데이터 흐름 경로에서 입력 벡터 데이터 샘플 세트를 제공하기 위한 수단;

상기 적어도 하나의 입력 데이터 흐름 경로에 제공되는 적어도 하나의 실행 유닛에서 상기 적어도 하나의 입력 데이터 흐름 경로를 통해 상기 입력 벡터 데이터 샘플 세트를 수신하기 위한 수단;

상기 적어도 하나의 입력 데이터 흐름 경로를 통해 결과적 출력 벡터 데이터 샘플 세트를 제공하기 위해서 상기 입력 벡터 데이터 샘플 세트에 대해 상기 벡터 프로세싱 연산을 실행하기 위한 수단;

상기 결과적 출력 벡터 데이터 샘플 세트가 상기 적어도 하나의 벡터 데이터 파일에 저장되는 것 없이 적어도 하나의 병합된 결과적 출력 벡터 데이터 샘플 세트를 제공하기 위해 수신된 결과적 출력 벡터 데이터 샘플 세트의 결과적 출력 벡터 데이터 샘플들을 병합하기 위한 수단 - 상기 결과적 출력 벡터 데이터 샘플들을 병합하는 것은 상기 결과적 출력 벡터 데이터 샘플들을 가산하는 것, 상기 결과적 출력 벡터 데이터 샘플들 중에서 최대 벡터 데이터 샘플을 결정하는 것, 또는 상기 결과적 출력 벡터 데이터 샘플들 중에서 최소 벡터 데이터 샘플을 결정하는 것을 포함함 -;

복수의 래치들 중 선택된 하나 또는 그 초과의 래치들에 상기 적어도 하나의 병합된 결과적 출력 벡터 데이터 샘플 세트를 저장하기 위한 수단;

상기 적어도 하나의 출력 데이터 흐름 경로를 통해 상기 선택된 하나 또는 그 초과의 래치들에 저장된 상기 적어도 하나의 병합된 결과적 출력 벡터 데이터 샘플 세트를 제공하기 위한 수단; 및

상기 적어도 하나의 출력 데이터 흐름 경로로부터의 상기 적어도 하나의 병합된 결과적 출력 벡터 데이터 샘플 세트를 상기 적어도 하나의 벡터 데이터 파일에 저장하기 위한 수단을 포함하고,

상기 병합하기 위한 수단은 상기 복수의 래치들에 대응하는 복수의 병렬 선택기들을 포함하고, 상기 병합하기 위한 수단은 상기 복수의 래치들 중 선택된 하나 또는 그 초과의 래치들에 상기 적어도 하나의 병합된 결과적 출력 벡터 데이터 샘플 세트를 저장하도록 상기 복수의 선택기들을 제어하게 구성되는, 적어도 하나의 실행 유닛에 의해 생성된 결과적 출력 벡터 데이터 샘플 세트를 실시간으로 병합하기 위한 장치.

청구항 18

벡터 프로세싱을 실행하는 적어도 하나의 실행 유닛에 의해 생성된 결과적 출력 벡터 데이터 샘플 세트의 실시간 병합 방법으로서,

벡터 프로세싱 연산을 위해 적어도 하나의 입력 데이터 흐름 경로에서 적어도 하나의 벡터 데이터 파일로부터 입력 벡터 데이터 샘플 세트를 제공하는 단계;

상기 적어도 하나의 입력 데이터 흐름 경로에 제공되는 적어도 하나의 실행 유닛에서 상기 적어도 하나의 입력

데이터 흐름 경로를 통해 상기 입력 벡터 데이터 샘플 세트를 수신하는 단계;

상기 적어도 하나의 입력 데이터 흐름 경로를 통해 결과적 출력 벡터 데이터 샘플 세트를 제공하기 위해서 상기 입력 벡터 데이터 샘플 세트에 대해 상기 벡터 프로세싱 연산을 실행하는 단계;

상기 결과적 출력 벡터 데이터 샘플 세트가 상기 적어도 하나의 벡터 데이터 파일에 저장되는 것 없이 적어도 하나의 병합된 결과적 출력 벡터 데이터 샘플 세트를 제공하기 위해 상기 결과적 출력 벡터 데이터 샘플 세트의 결과적 출력 벡터 데이터 샘플들을 적어도 하나의 병합 회로에 의해 병합하는 단계 – 상기 결과적 출력 벡터 데이터 샘플들을 병합하는 것은 상기 결과적 출력 벡터 데이터 샘플들을 가산하는 것, 상기 결과적 출력 벡터 데이터 샘플들 중에서 최대 벡터 데이터 샘플을 결정하는 것, 또는 상기 결과적 출력 벡터 데이터 샘플들 중에서 최소 벡터 데이터 샘플을 결정하는 것을 포함함 –;

복수의 래치들 중 선택된 하나 또는 그 초과의 래치들에 상기 적어도 하나의 병합된 결과적 출력 벡터 데이터 샘플 세트를 저장하는 단계 – 상기 복수의 래치들 중 선택된 하나 또는 그 초과의 래치들에 상기 적어도 하나의 병합된 결과적 출력 벡터 데이터 샘플 세트를 저장하는 것은, 상기 벡터 프로세싱 연산의 제어하에 상기 복수의 래치들에 대응하는 복수의 병렬 선택기들을 사용하여 상기 선택된 하나 또는 그 초과의 래치들을 선택하는 것을 포함함 –;

상기 적어도 하나의 출력 데이터 흐름 경로를 통해 상기 선택된 하나 또는 그 초과의 래치들에 저장된 상기 적어도 하나의 병합된 결과적 출력 벡터 데이터 샘플 세트를 제공하는 단계; 및

상기 적어도 하나의 출력 데이터 흐름 경로로부터의 상기 적어도 하나의 병합된 결과적 출력 벡터 데이터 샘플 세트를 상기 적어도 하나의 벡터 데이터 파일에 저장하는 단계를 포함하는, 벡터 프로세싱을 실행하는 적어도 하나의 실행 유닛에 의해 생성된 결과적 출력 벡터 데이터 샘플 세트의 실시간 병합 방법.

청구항 19

제 18 항에 있어서,

상기 결과적 출력 벡터 데이터 샘플들을 병합하는 것은, 상기 적어도 하나의 병합된 결과적 출력 벡터 데이터 샘플 세트를 제공하기 위해 적어도 하나의 가산기에서 상기 결과적 출력 벡터 데이터 샘플들 중 두 개를 가산하는 것을 포함하는, 벡터 프로세싱을 실행하는 적어도 하나의 실행 유닛에 의해 생성된 결과적 출력 벡터 데이터 샘플 세트의 실시간 병합 방법.

청구항 20

제 19 항에 있어서,

상기 적어도 하나의 가산기는 가산기 트리에서 제공되는 복수의 가산기들을 포함하고, 상기 복수의 가산기들의 각각은 상이한 비트 폭을 각각 갖는 복수의 병합된 결과적 출력 벡터 데이터 샘플 세트들을 제공하도록 구성되는, 벡터 프로세싱을 실행하는 적어도 하나의 실행 유닛에 의해 생성된 결과적 출력 벡터 데이터 샘플 세트의 실시간 병합 방법.

청구항 21

제 20 항에 있어서,

상기 적어도 하나의 출력 데이터 흐름 경로에서 상기 적어도 하나의 병합된 결과적 출력 벡터 데이터 샘플 세트로서 제공하기 위해서 상기 복수의 병합된 결과적 출력 벡터 데이터 샘플 세트들 중 하나를 선택하는 단계를 더 포함하는, 벡터 프로세싱을 실행하는 적어도 하나의 실행 유닛에 의해 생성된 결과적 출력 벡터 데이터 샘플 세트의 실시간 병합 방법.

청구항 22

제 18 항에 있어서,

프로그램가능 병합 데이터 경로 구성 입력을 수신하는 단계; 및

상기 프로그램가능 병합 데이터 경로 구성 입력에 기초하여 상기 결과적 출력 벡터 데이터 샘플 세트를 선택적으로 병합하는 단계를 더 포함하는, 벡터 프로세싱을 실행하는 적어도 하나의 실행 유닛에 의해 생성된 결과적

출력 벡터 데이터 샘플 세트의 실시간 병합 방법.

청구항 23

제 22 항에 있어서,

상기 적어도 하나의 실행 유닛에 의해 실행될 VPE의 각각의 클록 사이클에서 상기 결과적 출력 벡터 데이터 샘플들을 선택적으로 병합하는 단계를 더 포함하는, 벡터 프로세싱을 실행하는 적어도 하나의 실행 유닛에 의해 생성된 결과적 출력 벡터 데이터 샘플 세트의 실시간 병합 방법.

청구항 24

제 22 항에 있어서,

상기 적어도 하나의 실행 유닛에 의해 실행될 다음 벡터 명령에 대해 상기 결과적 출력 벡터 데이터 샘플들을 선택적으로 병합하는 단계를 더 포함하는, 벡터 프로세싱을 실행하는 적어도 하나의 실행 유닛에 의해 생성된 결과적 출력 벡터 데이터 샘플 세트의 실시간 병합 방법.

청구항 25

삭제

청구항 26

삭제

청구항 27

삭제

발명의 설명

기 술 분 야

[0001] 본 출원은 "VECTOR PROCESSING ENGINES HAVING PROGRAMMABLE DATA PATH CONFIGURATIONS FOR PROVIDING MULTI-MODE VECTOR PROCESSING, AND RELATED VECTOR PROCESSORS, SYSTEMS, AND METHODS"라는 명칭으로 2013년 3월 13일에 출원된 미국 특허 출원 일련번호 제13/798,641호(123249)와 관련되며, 이 출원은 그 전체가 인용에 의해 본원에 통합된다.

[0002] 본 출원은 "VECTOR PROCESSING CARRY-SAVE ACCUMULATORS EMPLOYING REDUNDANT CARRY-SAVE FORMAT TO REDUCE CARRY PROPAGATION, AND RELATED VECTOR PROCESSORS, SYSTEMS, AND METHODS"라는 명칭으로 2013년 3월 13일에 출원된 미국 특허 출원 일련번호 제13/798,618호(123248)와 관련되며, 이 출원은 그 전체가 인용에 의해 본원에 통합된다.

[0003] 본 출원은 또한 "VECTOR PROCESSING ENGINES (VPEs) EMPLOYING A TAPPED-DELAY LINE(S) FOR PROVIDING PRECISION FILTER VECTOR PROCESSING OPERATIONS WITH REDUCED SAMPLE RE-FETCHING AND POWER CONSUMPTION, AND RELATED VECTOR PROCESSOR SYSTEMS AND METHODS"라는 명칭으로 2013년 11월 15일에 출원된 미국 특허 출원 일련번호 제14/082,075호(124362)와 관련되며, 이 출원은 그 전체가 인용에 의해 본원에 통합된다.

[0004] 본 출원은 또한 "VECTOR PROCESSING ENGINES (VPEs) EMPLOYING A TAPPED-DELAY LINE(S) FOR PROVIDING PRECISION CORRELATION/COVARIANCE VECTOR PROCESSING OPERATIONS WITH REDUCED SAMPLE RE-FETCHING AND POWER CONSUMPTION, AND RELATED VECTOR PROCESSOR SYSTEMS AND METHODS"라는 명칭으로 2013년 11월 15일에 출원된 미국 특허 출원 일련번호 제14/082,079호(124364)와 관련되며, 이 출원은 그 전체가 인용에 의해 본원에 통합된다.

[0005] 본 출원은 또한 "VECTOR PROCESSING ENGINES (VPEs) EMPLOYING FORMAT CONVERSION CIRCUITRY IN DATA FLOW PATHS BETWEEN VECTOR DATA MEMORY AND EXECUTION UNITS TO PROVIDE IN-FLIGHT FORMAT-CONVERTING OF INPUT VECTOR DATA TO EXECUTION UNITS FOR VECTOR PROCESSING OPERATIONS, AND RELATED VECTOR PROCESSOR SYSTEMS AND METHODS"라는 명칭으로 2013년 11월 15일에 출원된 미국 특허 출원 일련번호 제14/082,088호(124365)와

관련되며, 이 출원은 그 전체가 인용에 의해 본원에 통합된다.

[0006] 본 출원은 또한 "VECTOR PROCESSING ENGINES (VPEs) EMPLOYING REORDERING CIRCUITRY IN DATA FLOW PATHS BETWEEN EXECUTION UNITS AND VECTOR DATA MEMORY TO PROVIDE IN-FLIGHT REORDERING OF OUTPUT VECTOR DATA STORED TO VECTOR DATA MEMORY, AND RELATED VECTOR PROCESSOR SYSTEMS AND METHODS"라는 명칭으로 2013년 11월 15일에 출원된 미국 특허 출원 일련번호 제14/082,081호(124450)와 관련되며, 이 출원은 그 전체가 인용에 의해 본원에 통합된다.

[0007] 본 출원은 또한 "VECTOR PROCESSING ENGINES (VPEs) EMPLOYING DESPREADING CIRCUITRY IN DATA FLOW PATHS BETWEEN EXECUTION UNITS AND VECTOR DATA MEMORY TO PROVIDE IN-FLIGHT DESPREADING OF SPREAD-SPECTRUM SEQUENCES, AND RELATED VECTOR PROCESSING INSTRUCTIONS, SYSTEMS, AND METHODS"라는 명칭으로 2013년 11월 15일에 출원된 미국 특허 출원 일련번호 제14/082,067호(124363U2)와 관련되며, 이 출원은 그 전체가 인용에 의해 본원에 통합된다.

[0008] 개시내용의 분야는 단일 명령 다중 데이터(SIMD) 프로세서들 및 다중 명령 다중 데이터(MIMD) 프로세서들을 포함하는, 벡터 및 스칼라 연산들을 프로세싱하기 위한 벡터 프로세서들 및 관련 시스템들에 관한 것이다.

배 경 기 술

[0009] 무선 통신 시스템들은 디지털 정보계에서 가장 유행하는 기술들 중 하나가 빠르게 되고 있는 중이다. 기술의 진보로 인해 더 작고 더 강력한 무선 통신 디바이스들이 만들어져 왔다. 예컨대, 무선 컴퓨팅 디바이스들은 보통 휴대용 무선 전화들, 개인휴대단말(PDA)들 및 페이징 디바이스들을 포함하는데, 이들은 작고 경량이어서 사용자가 휴대하기가 용이하다. 특히, 휴대용 무선 전화들, 예컨대 셀룰러 전화들 및 인터넷 프로토콜(IP) 전화들은 무선 네트워크들을 통해 음성 및 데이터 패킷들을 통신할 수 있다. 게다가, 이러한 많은 무선 통신 디바이스들은 다른 타입들의 디바이스들을 포함한다. 예컨대, 무선 전화는 디지털 스틸 카메라, 디지털 비디오 카메라, 디지털 레코더 및/또는 오디오 파일 플레이어를 포함할 수 있다. 또한, 무선 전화들은 인터넷에 액세스하기 위하여 사용될 수 있는 웹 인터페이스를 포함할 수 있다. 게다가, 무선 통신 디바이스들은 지정된 무선 통신 기술 표준들(예컨대, 코드 분할 다중 액세스(CDMA), 광대역 CDMA(WCDMA) 및 롱 텀 에벌루션(LTE))에 따라 고속 무선 통신 데이터를 프로세싱하기 위한 복합 프로세싱 자원들을 포함할 수 있다. 따라서, 이들 무선 통신 디바이스들은 중요한 컴퓨팅 능력들을 포함한다.

[0010] 무선 컴퓨팅 디바이스들이 더 작고 더 강력해짐에 따라, 무선 컴퓨팅 디바이스들은 점점 더 자원에 의해 제약을 받는다. 예컨대, 스크린 크기, 이용가능한 메모리 및 파일 시스템 공간량, 및 입출력량의 능력들이 디바이스의 작은 크기에 의해 제한받을 수 있다. 게다가, 배터리 크기, 배터리에 의해 공급되는 전력량 및 배터리 수명이 또한 제한받는다. 디바이스의 배터리 수명을 증가시키기 위한 하나의 방법은 전력을 덜 소비하는 프로세서들을 설계하는 것이다.

[0011] 이와 관련하여, 베이스밴드 프로세서들은 벡터 프로세서들을 포함하는 무선 통신 디바이스들을 위해 사용될 수 있다. 벡터 프로세서들은 벡터들, 즉 데이터의 어레이들에 대해 실행되는 고레벨 연산들을 제공하는 벡터 아키텍처를 가진다. 벡터 프로세싱은 하나의 데이터 세트에 대하여 벡터 명령을 실행하고 이후 벡터 내의 후속 엘리먼트들에 대해 벡터 명령을 재패치(re-fetching) 및 디코딩하는 것과 대조적으로, 벡터 명령을 한번 패치하고 이후 데이터 엘리먼트들의 전체 어레이에 걸쳐 벡터 명령을 여러 번 실행하는 것을 수반한다. 이러한 프로세스는 프로그램을 실행하는데 필요한 에너지를 감소시키는 것을 가능하게 하는데, 왜냐하면 다른 팩터들 중에서 각각의 벡터 명령은 보다 적은 횟수로 패치될 필요가 있기 때문이다. 벡터 명령들이 다수의 클록 사이클들 동안 긴 벡터들에 대하여 동시에 동작하기 때문에, 단순한 인-오더 벡터 명령 디스패치(in-order vector instruction dispatch)로 고도의 병렬화(parallelism)가 달성될 수 있다.

[0012] 도 1은 무선 컴퓨터 디바이스와 같은 컴퓨팅 디바이스에서 사용될 수 있는 예시적인 베이스밴드 프로세서(10)를 예시한다. 베이스밴드 프로세서(10)는 다수의 프로세싱 엔진들(PE들)(12)을 포함하며, 프로세싱 엔진들(PE들) 각각은 특정 애플리케이션들에 대한 기능-특정 벡터 프로세싱을 제공하는데 전용된다. 이러한 예에서, 6개의 개별 PE들(12(0)-12(5))이 베이스밴드 프로세서(10)에 제공된다. PE들(12(0)-12(5))은 공유 메모리(16)로부터 PE들(12(0)-12(5))로 제공되는 소정의 X 비트 폭 벡터 데이터(14)에 대한 벡터 프로세싱을 제공하도록 각각 구성된다. 예컨대, 벡터 데이터(14)는 폭이 512 비트일 수 있다. 벡터 데이터(14)는 작은 배수의 X 비트 폭 벡터 데이터 샘플 세트들(18(0)-18(Y))(예컨대, 16 비트 및 32 비트 샘플 세트들)로 정의될 수 있다. 이러한 방식으로, PE들(12(0)-12(5))은 고도의 병렬화를 달성하기 위하여 PE들(12(0)-12(5))에 병렬로 제공되는 다수의

벡터 데이터 샘플 세트들(18)에 대한 벡터 프로세싱을 제공할 수 있다. 각각의 PE(12(0)-12(5))는 벡터 데이터(14)에 대해 프로세싱되는 벡터 명령의 결과들을 저장하기 위한 벡터 레지스터 파일(VR)을 포함할 수 있다.

[0013] 도 1의 베이스밴드 프로세서(10)의 각각의 PE(12(0)-12(5))는 특정 타입들의 소정의 연산들을 효율적으로 수행하도록 특별히 설계된 특정 전용 회로소자 및 하드웨어를 포함한다. 예컨대, 도 1의 베이스밴드 프로세서(10)는 독립된 WCDMA PE들(12(0), 12(1)) 및 LTE PE들(12(4), 12(5))을 포함하는데, 왜냐하면 WCDMA 및 LTE가 상이한 타입들의 전문화된 연산들을 수반하기 때문이다. 따라서, 독립된 WCDMA-특정 PE들(12(0), 12(1)) 및 LTE-특정 PE들(12(4), 12(5))을 제공함으로써, PE들(12(0), 12(1), 12(4), 12(5)) 각각은 고효율 연산을 위하여 WCDMA 및 LTE에 대해 자주 수행되는 기능들에 특정한 전문화된 전용 회로소자를 포함하도록 설계될 수 있다. 이러한 설계는 상당히 많은 비관련 연산들을 지원하도록 융통성이 있으나 덜 효율적인 방식으로 설계된 더 일반적인 회로소자 및 하드웨어를 포함하는 스칼라 프로세싱 엔진들과 대조적이다.

[0014] 특정한 무선 베이스밴드 연산들은 이전의 프로세싱 연산들로부터 결정된 데이터 샘플들의 병합을 요구한다. 예컨대, 그것은 실행 유닛들의 데이터 경로들보다 더 넓은 가변 폭들의 벡터 데이터 샘플들을 누산시키기 위해 요구될 수 있다. 다른 예로서, 그것은 벡터 프로세싱 연산들에서 출력 벡터 데이터의 병합을 제공하기 위해, 상이한 실행 유닛들로부터의 출력 벡터 데이터 샘플들의 내적 곱셈을 제공하기 위해 요구될 수 있다. 이러한 벡터 프로세싱 연산들에서의 벡터 데이터 샘플들은 벡터 데이터 레인들을 크로스하는 데이터 경로들을 제공하는 복잡한 라우팅을 포함할 수 있다. 그러나, 이는 복잡성을 증가시키고, 그리고 상이한 벡터 데이터 레인들을 크로스 오버하여 병합될 출력 벡터 데이터의 병렬화 어려움들 때문에, 벡터 프로세싱 엔진(VPE)의 효율성을 감소시킬 수 있다. 벡터 프로세서들은, 실행 유닛들로부터 벡터 데이터 메모리로 저장되는 출력 벡터 데이터의 사후-프로세싱 병합을 수행하는 회로를 또한 포함할 수 있다. 벡터 데이터 메모리에 저장된, 사후-프로세싱된 출력 벡터 데이터 샘플들은 벡터 데이터 메모리로부터 패치되고, 원하는 대로 병합되며, 그리고 다시 벡터 데이터 메모리에 저장된다. 그러나, 이러한 사후-프로세싱은 VPE의 후속 벡터 프로세싱 연산들을 지연시킬 수 있고, 그리고 실행 유닛들의 계산 컴포넌트들이 충분히 이용되지 않게 할 수 있다.

발명의 내용

[0015] 본원에 개시된 실시예들은 벡터 데이터 메모리에 저장될 출력 벡터 데이터의 실시간 병합을 제공하기 위해서 실행 유닛들과 베터 데이터 메모리 사이의 데이터 흐름 경로들에서 병합 회로를 이용하는 벡터 프로세싱 엔진들(VPE들)을 포함한다. 관련된 벡터 프로세싱 명령들, 시스템들 및 방법들이 또한 개시된다. 병합 회로는 VPE의 벡터 데이터 메모리와 실행 유닛들 사이의 데이터 흐름 경로들에 제공된다. 병합 회로는, 출력 벡터 데이터 샘플 세트가 저장되기 위해 실행 유닛들로부터 출력 데이터 흐름 경로들을 통해 벡터 데이터 메모리로 제공되고 있는 동안에 실시간으로 벡터 프로세싱 연산들을 수행하는 결과로서 실행 유닛들로부터의 출력 벡터 데이터 샘플 세트를 병합하도록 구성된다. 출력 벡터 데이터 샘플 세트들의 실시간 병합은, 실행 유닛들에 의해 제공되는 출력 벡터 데이터 샘플 세트의 원하는 프로그래밍된 출력 벡터 데이터 샘플들이 벡터 데이터 메모리에 저장되기 이전에 병합될 수 있고, 따라서 출력 벡터 데이터 샘플 세트가 벡터 데이터 메모리에 병합된 포맷으로 저장됨을 의미한다. 비제한적인 예로서, 출력 벡터 데이터의 병합은 병합된 출력 벡터 데이터 샘플 세트들 및 출력 스칼라 데이터 샘플 세트를 제공하기 위해 출력 벡터 데이터 샘플 세트들을 추가하는 것을 포함할 수 있다. 또 다른 비제한적인 예로서, 출력 벡터 데이터 샘플 세트들의 병합은 실행 유닛들로부터의 비교되는 출력 벡터 데이터 샘플 세트들 간의 최대 및/또는 최소 출력 벡터 데이터를 생성하는 것을 포함할 수 있다. 병합된 출력 벡터 데이터 샘플 세트는 실행 유닛들에서 수행될 후속 벡터 프로세싱 연산들을 지연시킬 수 있는 추가적인 사후-프로세싱 단계들을 필요로 하지 않고 벡터 데이터 메모리에 병합된 형태로 저장된다.

[0016] 따라서, VPE에서 데이터 흐름 경로들의 효율성이 출력 벡터 데이터의 병합에 의해 제한되지 않는다. 실행 유닛들에서 후속 벡터 프로세싱은 출력 벡터 데이터 샘플 세트들이 벡터 데이터 메모리에 병합된 형태로 저장되어야 할 때 데이터 흐름 제한들에 의해서 보다는 계산 자원들에 의해서만 단지 제한된다. VPE는 실행 유닛들의 계산 엘리먼트들의 효율성에 영향을 주지 않으면서 벡터 데이터 메모리의 원하는 목적 위치에 병합된 인트라-벡터 출력 벡터 데이터 샘플 세트들을 제공하도록 또한 구성된다.

[0017] 이와 관련하여, 일 실시예에서, 벡터 프로세싱 연산을 실행하는 적어도 하나의 실행 유닛에 의해 생성된 결과적 출력 벡터 데이터 샘플 세트를 실시간으로 병합하도록 구성된 VPE가 제공된다. VPE는 적어도 하나의 벡터 데이터 파일을 포함한다. 벡터 데이터 파일(들)은 벡터 프로세싱 연산을 위해 적어도 하나의 입력 데이터 흐름 경로에서 패치된 입력 벡터 데이터 샘플 세트를 제공하도록 구성된다. 벡터 데이터 파일(들)은 또한 적어도 하나의 출력 데이터 흐름 경로로부터 저장될 적어도 하나의 병합된 결과적 출력 벡터 데이터 샘플 세트를 수신하도

록 구성된다. VPE는 또한 적어도 하나의 입력 데이터 흐름 경로에 제공되는 적어도 하나의 실행 유닛을 포함한다. 실행 유닛(들)은 적어도 하나의 입력 데이터 흐름 경로를 통해 입력 벡터 데이터 샘플 세트를 수신하도록 구성된다. 실행 유닛(들)은 또한 적어도 하나의 출력 데이터 흐름 경로를 통해 결과적 출력 벡터 데이터 샘플 세트를 제공하기 위해서 입력 벡터 데이터 샘플 세트에 대해 벡터 프로세싱 연산을 실행하도록 구성된다. VPE는 또한 적어도 하나의 병합 회로를 포함한다. 병합 회로는 또한 결과적 출력 벡터 데이터 샘플 세트가 적어도 하나의 벡터 데이터 파일에 저장되지 않고 적어도 하나의 병합된 결과적 출력 벡터 데이터 샘플 세트를 제공하기 위해 결과적 출력 벡터 데이터 샘플 세트를 병합하도록 구성된다. 병합 회로는 또한 적어도 하나의 출력 데이터 흐름 경로를 통해 적어도 하나의 병합된 결과적 출력 벡터 데이터 샘플 세트를 제공하도록 구성된다.

[0018] 다른 실시예에서, 벡터 프로세싱 연산을 실행하는 적어도 하나의 실행 유닛에 의해 생성된 결과적 출력 벡터 데이터 샘플 세트를 실시간으로 병합하도록 구성된 VPE가 제공된다. VPE는 적어도 하나의 벡터 데이터 파일 수단을 포함한다. 벡터 데이터 파일 수단은 벡터 프로세싱 연산을 위해 적어도 하나의 입력 데이터 흐름 경로 수단에서 패치된 입력 벡터 데이터 샘플 세트를 제공하기 위한 수단을 포함한다. 벡터 데이터 파일 수단은 또한 적어도 하나의 출력 데이터 흐름 경로 수단으로부터 저장될 적어도 하나의 병합된 결과적 출력 벡터 데이터 샘플 세트를 수신하기 위한 수단을 포함한다. VPE는 또한 적어도 하나의 입력 데이터 흐름 경로 수단에 제공되는 적어도 하나의 실행 유닛 수단을 포함한다. 실행 유닛 수단은 적어도 하나의 입력 데이터 흐름 경로 수단을 통해 입력 벡터 데이터 샘플 세트를 수신하기 위한 수단을 포함한다. 실행 유닛 수단은 또한 적어도 하나의 입력 데이터 흐름 경로 수단을 통해 결과적 출력 벡터 데이터 샘플 세트를 제공하기 위해서 입력 벡터 데이터 샘플 세트에 대해 벡터 프로세싱 연산을 실행하기 위한 수단을 포함한다.

[0019] 게다가, VPE는 또한 적어도 하나의 병합 회로 수단을 포함한다. 병합 회로 수단은 적어도 하나의 입력 데이터 흐름 경로 수단을 통해 결과적 출력 벡터 데이터 샘플 세트를 수신하기 위한 수단을 포함한다. 병합 회로 수단은 또한 결과적 출력 벡터 데이터 샘플 세트가 적어도 하나의 벡터 데이터 파일에 저장되지 않고 적어도 하나의 병합된 결과적 출력 벡터 데이터 샘플 세트를 제공하기 위해 결과적 출력 벡터 데이터 샘플 세트를 코드 시퀀스 벡터 데이터 샘플 세트와 병합하기 위한 수단을 포함한다. 병합 회로 수단은 또한 적어도 하나의 출력 데이터 흐름 경로를 통해 적어도 하나의 병합된 결과적 출력 벡터 데이터 샘플 세트를 제공하기 위한 수단을 포함한다.

[0020] 다른 실시예에서, 벡터 프로세싱 연산을 실행하는 적어도 하나의 실행 유닛에 의해 생성된 결과적 출력 벡터 데이터 샘플 세트를 실시간으로 병합하는 방법이 제공된다. 방법은 적어도 하나의 벡터 데이터 파일로부터 벡터 프로세싱 연산을 위해 적어도 하나의 입력 데이터 흐름 경로에서 패치된 입력 벡터 데이터 샘플 세트를 제공하는 단계를 포함한다. 방법은 또한 적어도 하나의 입력 데이터 흐름 경로에 제공되는 적어도 하나의 실행 유닛에서 적어도 하나의 입력 데이터 흐름 경로를 통해 패치된 입력 벡터 데이터 샘플 세트를 수신하는 단계를 포함한다. 방법은 또한 적어도 하나의 입력 데이터 흐름 경로를 통해 결과적 출력 벡터 데이터 샘플 세트를 제공하기 위해서 입력 벡터 데이터 샘플 세트에 대해 벡터 프로세싱 연산을 실행하는 단계를 포함한다. 방법은 또한 결과적 출력 벡터 데이터 샘플 세트가 적어도 하나의 벡터 데이터 파일에 저장되지 않고 적어도 하나의 병합된 결과적 출력 벡터 데이터 샘플 세트를 제공하기 위해 결과적 출력 벡터 데이터 샘플 세트를 병합하는 단계를 포함한다. 방법은 또한 적어도 하나의 출력 데이터 흐름 경로로부터의 적어도 하나의 병합된 결과적 출력 벡터 데이터 샘플 세트를 적어도 하나의 벡터 데이터 파일에 저장하는 단계를 포함한다.

도면의 간단한 설명

[0021] 도 1은 특정 애플리케이션들에 대한 기능-특정 벡터 프로세싱을 제공하는데 각각 전용되는 다수의 벡터 프로세싱 엔진(VPE)들을 포함하는 예시적인 벡터 프로세서의 개략도이다.

도 2는 VPE에 제공된 공통 회로 및 하드웨어가 별도의 VPE들을 제공할 필요성 없이, 다수의 애플리케이션들 또는 기술들에 대해 매우 효율적인 방식으로 특정 타입들의 벡터 연산들을 수행하기 위해 다수의 모드들에서 프로그래밍될 수 있도록, 프로그램가능 데이터 경로 구성들을 가지는 VPE를 포함하는 예시적인 기저대역 프로세서의 개략도이다.

도 3은 VPE에 의해 지원되는 필터 벡터 프로세싱 연산에 제공될 수 있는 이산 유한 임펄스 응답(FIR) 필터의 개략도이다.

도 4는 감소된 재패치 및 전력 소비를 가진 정밀한 필터 벡터 프로세싱 연산들을 제공하기 위하여 시프트된 입력 벡터 데이터 샘플 세트들을 수신하여 필터 계수 데이터로 프로세싱될 실행 유닛들에 제공하도록 탭핑-지연 라인들을 이용하는 예시적인 VPE의 개략도이다.

도 5는 예시적인 필터 벡터 명령에 따라 도 4의 VPE에서 수행될 수 있는 예시적인 필터 벡터 프로세싱 연산을 예시하는 흐름도이다.

도 6a는 도 4의 VPE의 레지스터 파일에 저장된 필터 탭 계수들의 개략도이다.

도 6b는 도 4의 VPE의 벡터 데이터 파일에 저장된 예시적인 입력 벡터 데이터 샘플 세트들의 개략도이다.

도 7은 도 4의 VPE에서 제공될 수 있는 예시적인 탭핑-지연 라인 및 선택적 섀도우 탭핑-지연 라인을 예시하는 개략도이고, 예시적인 탭핑-지연 라인들 각각은 VPE에 의해 수행되는 필터 벡터 프로세싱 연산들 동안 벡터 데이터 메모리로부터의 입력 벡터 데이터 샘플 세트 및 시프트된 입력 벡터 데이터 샘플 세트를 수신하여 실행 유닛들에 제공하기 위한 복수의 파이프라인 레지스터들을 포함한다.

도 8은 필터 벡터 프로세싱 연산 동안 입력 벡터 데이터 샘플 세트에서 입력 벡터 데이터 샘플들의 시프트를 위하여 파이프라인 레지스터들 사이에서 라우팅하는 인트라-레인(intra-lane) 및 인터(inter)-레인을 포함하는, 데이터 레인들의 파이프라인 레지스터들의 예시적인 상세를 예시하는, 도 7의 탭핑-지연 라인들의 보다 예시적인 상세를 예시하는 개략도이다.

도 9a는 예시적인 여덟(8) 탭 필터 벡터 프로세싱 연산의 제 1 필터 탭 실행의 부분으로서 도 4의 VPE에서 1차 탭핑-지연 라인에 처음에 저장된 입력 벡터 샘플 세트의 개략도이다.

도 9b는 도 9a에 예시된 예시적인 여덟(8) 탭 필터 벡터 프로세싱 연산 필터 벡터 프로세싱 연산의 제 1 필터 탭 실행의 부분으로서 도 4의 VPE에서, 섀도우 탭핑-지연 라인에 처음에 저장된 섀도우 입력 벡터 데이터 샘플 세트 및 레지스터 파일에 저장된 필터 탭 계수들의 개략도이다.

도 9c는 예시적인 여덟(8) 탭 필터 벡터 프로세싱 연산의 제 2 필터 탭 실행의 부분으로서 도 4의 VPE에서, 1차 탭핑-지연 라인 및 섀도우 탭핑-지연 라인에 저장된 시프트된 입력 벡터 데이터 샘플 세트들, 및 레지스터 파일에 저장된 필터 탭 계수들의 개략도이다.

도 9d는 예시적인 여덟(8) 탭 필터 벡터 프로세싱 연산의 제 8 필터 탭 실행의 부분으로서 도 4의 VPE에서, 1차 탭핑-지연 라인 및 섀도우 탭핑-지연 라인에 저장된 시프트된 입력 벡터 데이터 샘플 세트들, 및 레지스터 파일에 저장된 필터 탭 계수들의 개략도이다.

도 10은 예시적인 여덟(8) 탭 필터 벡터 프로세싱 연산이 완전히 실행된 후 도 4의 VPE에서 실행 유닛들의 누산기들의 컨텐츠의 개략도이다.

도 11은 감소된 재패치 및 전력 소비로 정확한 상관/공분산 벡터 프로세싱 연산들을 제공하기 위하여 시프트된 입력 벡터 데이터 샘플 세트들을 수신하여 시퀀스 번호 데이터를 가진 프로세싱될 실행 유닛들에 제공하도록 탭핑-지연 라인들을 이용하는 예시적인 VPE의 개략도이다.

도 12a 및 도 12b는 예시적인 상관/공분산 벡터 프로세싱 연산에 따라 패치되고 인터리빙된 정시(on-time) 및 늦은(late) 입력 벡터 데이터 샘플 세트들로 도 11의 VPE에서 병렬로 수행될 수 있는 예시적인 상관/공분산 벡터 프로세싱 연산들을 예시하는 흐름도들이다.

도 13은 도 11의 VPE에서 레지스터 파일에 저장된 상관/공분산 입력 벡터 데이터 샘플 세트의 개략도이다.

도 14는 도 11의 VPE에서 제공될 수 있는 예시적인 탭핑-지연 라인 및 선택적 섀도우 탭핑-지연 라인을 예시하는 개략도이고, 예시적인 탭핑-지연 라인들 각각은, VPE에 의해 수행된 상관/공분산 벡터 프로세싱 연산 동안, 벡터 데이터 메모리로부터의 입력 벡터 데이터 샘플 세트 및 시프트된 입력 벡터 데이터 샘플 세트를 수신하여 실행 유닛들에 제공하기 위한 복수의 파이프라인 레지스터들을 포함한다.

도 15a는 상관/공분산 벡터 프로세싱 연산의 제 1 프로세싱 스테이지의 부분으로서 도 11의 VPE에서 1차 탭핑-지연 라인에 처음에 제공된 벡터 데이터 파일로부터의 입력 벡터 데이터 샘플 세트의 개략도이다.

도 15b는 상관/공분산 벡터 프로세싱 연산의 제 1 프로세싱 스테이지의 부분으로서 도 11의 VPE에서 섀도우 탭핑-지연 라인에 처음에 저장된 벡터 데이터 파일로부터의 섀도우 입력 벡터 데이터 샘플 세트의 개략도이다.

도 15c는 상관/공분산 벡터 프로세싱 연산의 제 2 프로세싱 스테이지의 부분으로서 도 11의 VPE에서, 1차 탭핑-지연 라인 및 섀도우 탭핑-지연 라인에 저장된 시프트된 입력 벡터 데이터 샘플 세트들 및 레지스터 파일에 저장된 시프트된 입력 벡터 데이터 샘플 세트의 개략도이다.

도 15d는 상관/공분산 벡터 프로세싱 연산의 제 14 프로세싱 스테이지의 부분으로서 도 11의 VPE에서, 1차 탭핑-지연 라인 및 새도우 탭핑-지연 라인에 저장된 시프트된 입력 벡터 데이터 샘플 세트들, 및 레지스터 파일에 저장된 시프트된 입력 벡터 데이터 샘플 세트의 개략도이다.

도 16은 예시적인 상관/공분산 벡터 프로세싱 연산이 완전히 실행된 후 도 11의 VPE에서 실행 유닛들의 누산기들의 컨텐츠의 개략도이다.

도 17a는 별도로 저장된 결과적 필터 출력 벡터 데이터 샘플들의 실수 및 허수 컴포넌트들로 저장되는 저장된 결과적 필터 출력 벡터 데이터 샘플 세트를 도시하는 예시적인 벡터 데이터 파일들의 다이어그램이다.

도 17b는 별도로 저장된 그의 짝수 및 홀수 결과적 필터 출력 벡터 데이터 샘플들이 저장되어 있는 저장된 결과적 필터 출력 벡터 데이터 샘플 세트를 도시하는 예시적인 벡터 데이터 파일들의 다이어그램이다.

도 18a 및 도 18b는 각각 부호가 붙은 복소수 십육(16) 비트 포맷 및 복소수 여덟(8) 비트 포맷으로 VPE의 벡터 데이터 파일에 저장된 벡터 데이터 샘플 세트의 예시적인 인터리빙된 벡터 데이터 샘플들의 다이어그램들이다.

도 19는 벡터 프로세싱 연산을 실행하기 위하여 포맷-변환된 입력 벡터 데이터 샘플 세트를 적어도 하나의 실행 유닛에 제공하기 위하여, 입력 벡터 데이터 샘플 세트가 벡터 데이터 파일로부터 재패치되도록 요구받지 않고 벡터 데이터 파일과 적어도 하나의 실행 유닛 사이의 적어도 하나의 입력 데이터 흐름 경로에 입력 벡터 데이터 샘플 세트의 실시간(in-flight) 포맷-변환을 제공하도록 구성된 포맷 변환 회로를 이용하는 예시적인 VPE의 개략도이다.

도 20은 도 19의 VPE에서 수행될 수 있는 벡터 데이터 파일과 적어도 하나의 실행 유닛 사이의 적어도 하나의 입력 데이터 흐름 경로에서 입력 벡터 데이터 샘플 세트의 예시적인 실시간 포맷-변환을 예시하는 흐름도이다.

도 21은 도 19의 VPE에서 탭핑-지연 라인들과 실행 유닛들 사이에 제공된 예시적인 포맷 변환 회로의 개략도이고, 포맷 변환 회로는 입력 데이터 흐름 경로의 탭핑-지연 라인들에 의해 실행 유닛들에 제공되는 입력 벡터 데이터 샘플 세트의 실시간 포맷-변환을 제공하도록 구성된다.

도 22는 실행 유닛들에서의 수신 이전에 입력 데이터 흐름 경로에서 입력 벡터 데이터 샘플 세트의 실시간 포맷 변환을 제공하기 위하여 도 19의 VPE에 프로그래밍을 제공하기 위한 예시적인 벡터 명령 데이터 포맷을 예시한다.

도 23은 재정렬된 결과적 출력 데이터 샘플 세트를 제공 및 저장하기 위하여, 결과적 출력 벡터 데이터 샘플 세트가 적어도 하나의 벡터 데이터 파일에 저장되지 않고, 적어도 하나의 실행 유닛과 적어도 하나의 벡터 데이터 파일 사이의 적어도 하나의 출력 데이터 흐름 경로에서 결과적 출력 벡터 데이터 샘플 세트의 실시간 재정렬을 제공하도록 구성된 재정렬 회로를 이용하는 예시적인 VPE의 개략도이다.

도 24는 벡터 데이터 파일에 재정렬된 형태로 저장되도록 도 23의 VPE에서 벡터 데이터 파일과 적어도 하나의 실행 유닛 사이의 적어도 하나의 출력 데이터 흐름 경로에서 출력 벡터 데이터 샘플 세트의 예시적인 실시간 디인터리빙을 예시하는 흐름도이다.

도 25는 벡터 데이터 파일에 저장된 출력 벡터 데이터 샘플 세트들의 실시간 재정렬을 제공하기 위하여 실행 유닛들과 벡터 데이터 파일 사이의 출력 데이터 흐름 경로들에서 재정렬 회로를 이용하는 예시적인 VPE의 개략도이다.

도 26의 a는 통신 신호를 표현하는 예시적인 벡터 데이터 샘플 시퀀스의 다이어그램이다.

도 26의 b는 예시적인 코드 분할 다중 액세스(CDMA) 칩 시퀀스의 다이어그램이다.

도 26의 c는 도 26의 b의 CDMA 칩 시퀀스가 확산된 후 도 26의 a의 벡터 데이터 샘플 시퀀스의 다이어그램이다.

도 26의 d는 도 26의 a의 본래 벡터 데이터 샘플 시퀀스를 복원하기 위하여 도 26의 b의 CDMA 칩 시퀀스를 사용하여 도 26의 c의 확산된 벡터 데이터 샘플 시퀀스를 역확산하는 다이어그램이다.

도 27은 역확산된 결과적 출력 벡터 데이터 샘플 세트를 제공 및 저장하기 위하여, 결과적 출력 벡터 데이터 샘플 세트가 적어도 하나의 벡터 데이터 파일에 저장되지 않고, 적어도 하나의 실행 유닛과 적어도 하나의 벡터 데이터 파일 사이의 적어도 하나의 출력 데이터 흐름 경로에서 결과적 출력 벡터 데이터 샘플 세트의 역확산을 제공하도록 구성된 역확산 회로를 이용하는 예시적인 VPE의 개략도이다.

도 28은 적어도 하나의 벡터 데이터 파일에 역확산된 결과적 출력 벡터 데이터 샘플 세트를 제공 및 저장하기 위하여, 도 27의 VPE에서 적어도 하나의 벡터 데이터 파일과 적어도 하나의 실행 유닛 사이의 적어도 하나의 출력 데이터 흐름 경로에서 결과적 출력 벡터 데이터 샘플 세트의 예시적인 역확산을 예시하는 흐름도이다.

도 29는 적어도 하나의 벡터 데이터 파일에 역확산된 결과적 출력 벡터 데이터 샘플 세트들을 제공 및 저장하기 위하여 결과적 출력 벡터 데이터 샘플 세트들의 역확산을 제공하도록 도 27의 VPE에서 적어도 하나의 실행 유닛과 적어도 하나의 벡터 데이터 파일 사이의 출력 데이터 흐름 경로들에서의 예시적인 역확산 회로의 개략도이다.

도 30은 병합(merge)될 예시적인 벡터 데이터 샘플 및 병합된 결과적 벡터 데이터 샘플들을 예시하는 다이어그램이다.

도 31은 병합된 결과적 출력 벡터 데이터 샘플 세트를 제공 및 저장하기 위하여, 결과적 출력 벡터 데이터 샘플 세트가 적어도 하나의 벡터 데이터 파일에 저장되지 않고, 적어도 하나의 실행 유닛과 적어도 하나의 벡터 데이터 파일 사이의 적어도 하나의 출력 데이터 흐름 경로에서 결과적 출력 벡터 데이터 샘플 세트의 병합을 제공하도록 구성된 병합 회로를 이용하는 예시적인 VPE의 개략도이다.

도 32는 벡터 데이터 파일 내 가산-병합된 결과적 출력 벡터 데이터 샘플 세트를 제공 및 저장하기 위하여, 도 31의 VPE에서 벡터 데이터 파일과 적어도 하나의 실행 유닛 사이의 적어도 하나의 출력 데이터 흐름 경로에서 결과적 출력 벡터 데이터 샘플 세트의 예시적인 가산-병합을 예시하는 흐름도이다.

도 33은 결과적 출력 벡터 데이터 샘플 세트들의 가산-병합 및 벡터 데이터 파일로의 가산-병합된 결과적 출력 벡터 데이터 샘플 세트의 저장을 제공하기 위하여 도 31의 VPE에서 실행 유닛들과 벡터 데이터 파일 사이의 출력 데이터 흐름 경로들에서의 예시적인 병합 회로의 개략도이다.

도 34는 결과적 출력 벡터 데이터 샘플 세트들의 최대/최소 병합을 제공하고 그리고 벡터 데이터 파일 내의 최대/최소-병합 결과적 출력 벡터 데이터 샘플 세트들의 저장을 제공하기 위하여 도 31의 VPE에서 실행 유닛들과 벡터 데이터 파일 사이의 출력 데이터 흐름 경로들에서의 예시적인 병합 회로의 개략도이다.

도 35는 VPE에 제공될 수 있는 예시적인 벡터 프로세싱 스테이지들의 개략도이고, 특정 벡터 프로세싱 스테이지들은 프로그램 가능 데이터 경로 구성들을 가지는 예시적인 벡터 프로세싱 블록들을 포함한다.

도 36은 곱셈기 블록들과 누산기 블록들의 예시적인 벡터 프로세싱을 예시하는 흐름도이고, 상기 블록들 각각은 도 35의 예시적인 VPE에 프로그램가능 데이터 경로 구성들을 가지며 상이한 벡터 프로세싱 스테이지들에 제공된다.

도 37은 도 35의 VPE의 벡터 프로세싱 스테이지에 제공된 복수의 곱셈기 블록들의 더 상세한 개략도이고, 복수의 곱셈기 블록들 각각은 프로그램가능 데이터 경로 구성들을 가져서, 복수의 곱셈기 블록들은 특정 상이한 타입들의 벡터 곱셈 연산들을 수행하도록 다중 모드들로 프로그래밍될 수 있다.

도 38은 8 비트 × 8 비트 입력 데이터 샘플 세트들과 16 비트 × 16 비트 입력 벡터 데이터 샘플 세트들에 대한 곱셈 연산들을 제공하도록 프로그래밍될 수 있는 프로그램가능 데이터 경로 구성들을 가진 도 37의 복수의 곱셈기 블록들 중 일 곱셈기 블록의 내부 컴포넌트들의 개략도이다.

도 39는 도 38의 VPE에서 곱셈기 블록 및 누산기 블록의 일반화된 개략도이고, 누산기 블록은 캐리 전파를 감소시키기 위하여 리던던트 캐리-절약 포맷을 이용하는 캐리-절약 누산기 구조를 이용한다.

도 40은 도 35의 VPE에 제공된, 도 39의 누산기 블록의 예시적인 내부 컴포넌트들의 상세한 개략도이고, 누산기 블록은 프로그램가능 데이터 경로 구성들을 가져서, 누산기 블록은 리던던트 캐리-절약 포맷으로 특정의 상이한 타입들의 벡터 누산 연산들을 수행하도록 다수의 모드들로 프로그래밍될 수 있다.

도 41은 본원에 개시된 실시예들에 따른, 벡터 프로세싱 회로들 및 벡터 프로세싱 연산들을 제공하기 위하여 본원에 개시된 VPE들을 포함할 수 있는 벡터 프로세서를 포함할 수 있는 예시적인 프로세서-기반 시스템의 블록도이다.

발명을 실시하기 위한 구체적인 내용

[0022] 이제 도시된 도면들을 참조하여, 본 개시내용의 몇몇 예시적인 실시예들이 설명된다. 단어 "예시적"은 "예, 예증, 또는 예시로서 제공하는 것"을 의미하기 위하여 본원에서 사용된다. "예시적"으로서 본원에 설명된 임의의

실시예가 반드시 다른 실시예들에 비해 바람직하거나 유리한 것으로 이해될 필요는 없다.

[0023] 본원에 개시된 실시예들은 벡터 데이터 메모리에 저장될 출력 벡터 데이터의 실시간 병합을 제공하기 위해서 실행 유닛들과 베터 데이터 메모리 사이의 데이터 흐름 경로들에서 병합 회로를 이용하는 벡터 프로세싱 엔진(VPE)을 포함한다. 관련된 베터 프로세싱 명령들, 시스템들 및 방법들이 또한 개시된다. 병합 회로는 VPE의 벡터 데이터 메모리와 실행 유닛들 사이의 데이터 흐름 경로들에 제공된다. 병합 회로는, 출력 벡터 데이터 샘플 세트가 저장되기 위해 실행 유닛들로부터 출력 데이터 흐름 경로들을 통해 벡터 데이터 메모리로 제공되고 있는 동안에 실시간으로 벡터 프로세싱 연산들을 수행하는 결과로서 실행 유닛들로부터의 출력 벡터 데이터 샘플 세트를 병합하도록 구성된다. 출력 벡터 데이터 샘플들의 실시간 병합은, 실행 유닛들에 의해 제공되는 출력 벡터 데이터 샘플 세트의 원하는 프로그래밍된 출력 벡터 데이터 샘플들이 벡터 데이터 메모리에 저장되기 이전에 병합될 수 있고, 따라서 출력 벡터 데이터 샘플 세트가 벡터 데이터 메모리에 병합된 포맷으로 저장됨을 의미한다. 비제한적인 예로서, 출력 벡터 데이터의 병합은 병합된 출력 벡터 데이터 샘플 세트들 및 출력 스칼라 데이터 샘플 세트를 제공하기 위해 출력 벡터 데이터 샘프 세트들을 추가하는 것을 포함할 수 있다. 또 다른 비제한적인 예로서, 출력 벡터 데이터 샘플 세트들의 병합은 실행 유닛들로부터의 비교되는 출력 벡터 데이터 샘플 세트들 간의 최대 및/또는 최소 출력 벡터 데이터를 생성하는 것을 포함할 수 있다. 병합된 출력 벡터 데이터 샘플 세트는 실행 유닛들에서 수행될 후속 벡터 프로세싱 연산들을 지연시킬 수 있는 추가적인 사후-프로세싱 단계들을 필요로 하지 않고 벡터 데이터 메모리에 병합된 형태로 저장된다.

[0024] 따라서, VPE에서 데이터 흐름 경로들의 효율성이 출력 벡터 데이터의 병합에 의해 제한되지 않는다. 실행 유닛들에서 후속 벡터 프로세싱은 출력 벡터 데이터 샘플 세트들이 벡터 데이터 메모리에 병합된 형태로 저장되어야 할 때 데이터 흐름 제한들에 의해서 보다는 계산 자원들에 의해서만 단지 제한된다. VPE는 실행 유닛들의 계산 엘리먼트들의 효율성에 영향을 주지 않으면서 벡터 데이터 메모리의 원하는 목적 위치에 병합된 인트라-벡터 출력 벡터 데이터 샘플 세트들을 제공하도록 또한 구성된다.

[0025] 이와 관련하여, 도 2는 또한 벡터 프로세싱 엔진(VPE)(22)으로 지칭되는 예시적인 벡터 프로세싱 유닛(22)을 포함하는 기저대역 프로세서(20)의 개략도이다. 아래에 더 상세히 논의될 바와 같이, VPE(22)는 본원에 개시된 예시적인 벡터 프로세싱 연산들을 비롯하여 벡터 프로세싱 동작들을 제공하기 위해 실행 유닛들(84) 및 다른 특정 예시적인 회로 및 기능을 포함한다. 기저대역 프로세서(20) 및 그의 VPE(22)는 반도체 다이(24)에 제공될 수 있다. 이 실시예에서, 아래에 더 상세히 논의될 바와 같이, 기저대역 프로세서(20)는 상이한 프로그램가능 데이터 경로 구성들을 제공하도록 프로그램될 수 있는 프로그램가능 데이터 경로들(26)을 포함하는 공통의 VPE(22)를 포함한다. 이런 방식으로, VPE(22) 내의 벡터 데이터 파일들(82)과 실행 유닛들(84) 사이의 프로그램가능 데이터 경로들(26)은, 기저대역 프로세서(20)에 별도의 VPE들(22)을 제공할 필요성 없이 상이한 동작 모드들에서 상이한 특정 타입들의 벡터 프로세싱 연산들을 제공하기 위해 프로그램 및 재프로그램될 수 있다.

[0026] 도 3에서 시작하는 효율적인 프로세싱을 위해 본 개시내용에서 VPE(22)에 의해 제공되도록 구성된 특정 회로 및 벡터 프로세싱 연산들을 논의하기 전에, 도 2의 기저대역 프로세서(20)의 컴포넌트들이 먼저 설명된다. 이런 비제한적 예에서 기저대역 프로세서(20)는 512 비트 벡터 프로세서이다. 기저대역 프로세서(20)는 기저대역 프로세서(20)에서 벡터 프로세싱을 제공하는 VPE(22)를 지원하기 위하여 VPE(22) 이외에 컴포넌트들을 포함한다. 기저대역 프로세서(20)는 벡터 유닛 데이터 메모리(LMEM)(32)로부터의 벡터 데이터(30)를 수신 및 저장하기 위해 구성된 벡터 데이터 파일들(82)로 또한 알려진 벡터 레지스터들을 포함한다. 예컨대, 벡터 데이터(30)는 X 비트 폭이고, 'X'는 설계 선택(예컨대, 512 비트들)에 따라 정의된다. 벡터 데이터(30)는 벡터 데이터 샘플 세트들(34)로 나누어질 수 있다. 비제한적인 예로서, 벡터 데이터(30)는 256 비트 폭일 수 있고, 보다 작은 벡터 데이터 샘플 세트들(34(Y)-34(0))을 포함할 수 있다. 예로서, 일부 벡터 데이터 샘플 세트들(34(Y)-34(0))은 16 비트 폭일 수 있고, 벡터 데이터 샘플 세트들(34(Y)-34(0))의 다른 것들은 32 비트 폭일 수 있다. VPE(22)는 높은 병렬도를 달성하기 위하여 VPE(22)에 병렬로 제공된 특정 선택된 벡터 데이터 샘플 세트들(34(Y)-34(0))에 대해 벡터 프로세싱을 제공할 수 있다. 벡터 데이터 파일들(82)은 또한, VPE(22)가 벡터 데이터(30)를 프로세싱할 때 생성된 결과들을 저장하도록 구성된다. 특정 실시예들에서, VPE(22)는 보다 빠른 벡터 명령 실행 시간들을 제공하기 위하여 레지스터 기록들을 감소시키기 위해 중간 벡터 프로세싱 결과들을 벡터 데이터 파일들(82)에 저장하지 않도록 구성된다. 이 구성은 스칼라 프로세싱 디지털 신호 프로세서들(DSP들)과 같은 레지스터들에 중간 결과들을 저장하는 스칼라 프로세싱 엔진들에 의해 실행되는 스칼라 명령들과 반대이다.

[0027] 도 2의 기저대역 프로세서(20)는 또한 벡터 명령들의 조건적 실행에 사용하기 위해 조건들을 VPE(22)에 제공하고 업데이트된 조건들을 벡터 명령 실행의 결과로서 저장하도록 구성된 조건 레지스터들(36)을 포함한다. 기저대역 프로세서(20)는 또한 누산 레지스터들(38), 글로벌 레지스터들을 포함하는 글로벌 레지스터 파일(40), 및

어드레스 레지스터들(42)을 포함한다. 누산 레지스터들(38)은 벡터 데이터(30)에 대한 특정 특수화된 동작들을 실행한 결과로서 누산된 결과들을 저장하기 위해 VPE(22)에 의해 사용되도록 구성된다. 글로벌 레지스터 파일(40)은 VPE(22)에 의해 지원된 특정 벡터 명령들에 대한 스칼라 오퍼랜드(operand)들을 저장하도록 구성된다. 어드레스 레지스터들(42)은 벡터 로드에 의해 어드레스 가능한 어드레스들을 저장하고, 벡터 유닛 데이터 메모리(32)로부터 벡터 데이터(30)를 리트리브하기 위해 VPE(22)에 의해 지원되는 명령들을 저장하고, 벡터 유닛 데이터 메모리(32)에 벡터 프로세싱 결과들을 저장하도록 구성된다.

[0028] 도 2를 계속 참조하여, 이 실시예에서 기저대역 프로세서(20)는 또한 VPE(22)에 의해 제공된 벡터 프로세싱 이외에 기저대역 프로세서(20)에서 스칼라 프로세싱을 제공하기 위해 스칼라 프로세서(44)(또한 "정수 유닛"으로 지칭됨)를 포함한다. 매우 효율적인 동작을 위해 실행된 명령 타입에 기초하여 벡터 및 스칼라 명령 동작들 둘 다를 지원하도록 구성된 CPU(central processing unit)를 제공하는 것이 바람직할 수 있다. 이 실시예에서, 스칼라 프로세서(44)는 비제한적인 예로서 32 비트 축소 명령 집합 컴퓨팅(RISC: reduced instruction set computing) 스칼라 프로세서이다. 스칼라 프로세서(44)는 이 예에서 스칼라 명령 프로세싱을 지원하기 위해 산술 논리 유닛(ALU)(46)을 포함한다. 기저대역 프로세서(20)는 프로그램 메모리(50)로부터 명령들을 패치하고, 패치된 명령들을 디코딩하고, 그리고 명령 타입에 기초하여 패치된 명령들을 스칼라 프로세서(44)로 지향하거나 벡터 데이터 경로(53)를 통해 VPE(22)로 지향도록 구성된 명령 디스패치 회로(48)를 포함한다. 스칼라 프로세서(44)는 스칼라 명령들을 실행할 때 스칼라 프로세서(44)에 의해 사용하기 위한 범용 레지스터들(54)을 포함한다. 정수 유닛 데이터 메모리(DMEM: integer unit data memory)(56)는 스칼라 명령 실행을 위한 스칼라 프로세서(44)에 의해 액세스하기 위한 데이터를 메인 메모리로부터 범용 레지스터들(54)로 제공하기 위해 기저대역 프로세서(20)에 포함된다. DMEM(56)은 비제한적 예로서 캐시 메모리일 수 있다. 기저대역 프로세서(20)는 또한, 스칼라 프로세서(44)가 메모리 제어기 데이터 경로들(62)을 통해 메인 메모리에 액세스를 요구하는 벡터 명령들을 실행할 때 범용 레지스터들(54)로부터 메모리 어드레스들을 수신하도록 구성된 메모리 제어기 레지스터들(60)을 포함하는 메모리 제어기(58)를 포함한다.

[0029] VPE(22)에 의한 벡터 명령 프로세싱에 의해 지원되도록 요구될 수 있는 하나의 타입의 특수화된 벡터 프로세싱 연산은 필터링이다. 필터 연산은 샘플링된 입력 시간 함수의 컨볼루션의 양자화된 시간-도메인 표현 및 필터의 가중 함수의 표현을 계산한다. 시간 도메인에서 컨볼루션은 주파수 도메인에서 곱셈에 대응한다. 따라서, 디지털 필터들은 일정하게 이격된 샘플 간격에서 수행되는 곱셈들 및 가산들의 확장된 시퀀스에 의해 VPE(22)에서 실현될 수 있다. 예컨대, 이산 FIR(finite impulse response) 필터는 필터 함수를 계산하기 위해 "Y" 계산 필터 계수들을 갖는 지연 라인 상의 유한수(Y)의 지연 탭들을 사용하여 구현될 수 있다.

[0030] 이와 관련하여, 도 3은 도 2의 VPE(22) 내의 필터 벡터 프로세싱 연산을 통해 지원되도록 요구될 수 있는 예시적인 이산 FIR 필터(64)의 개략도이다. 디지털화된 입력 신호(66(x[n]))는 "필터 지연 탭들(68(1)-68(Y-1))"로 불리는 지연 구조들을 통해 디지털화된 입력 신호 샘플들(x[0], x[1],...x[n])을 통과시킴으로써 필터링될 수 있다. 필터 지연 탭들(68(1)-68(Y-1))은, 필터 샘플 피승수들(72(0)-72(Y-1))(즉, h(1)*x[n-1])을 제공하기 위해 필터 계수들(h[0]-h(Y-1))로 각각 곱셈될 모든 디지털화된 입력 신호 샘플들(즉, x[0], x[1],... x[n])에 대해 클록 디지털화된 입력 신호 샘플들(즉, x[0], x[1],... x[n])을 곱셈기들(70(0)-70(Y-1))로 시프트한다. 필터 샘플 피승수들(72(0)-72(Y-1))은 결과적 필터링된 출력 신호(76)(즉, y[n])를 제공하기 위해 합산기들(즉, 가산기들)(74(1)-74(Y-1))에 의해 함께 합산된다. 따라서, 도 3의 이산 FIR 필터(64)는 다음과 같이 요약될 수 있다.

$$y[n] = \sum_{l=0}^{l=Y-1} h(l) * x[n-l]$$

[0031]

[0032] 여기서, n은 입력 신호 샘플들의 수이고,

[0033] x[n]은 디지털화된 입력 신호(66)이고,

[0034] y[n]은 결과적 필터링된 출력 신호(76)이고,

[0035] h(l)은 필터 계수들이고,

[0036] Y는 필터 계수들의 수이다.

[0037] 필터 계수들(h(l))은 복소수일 수 있다. 일 양상에서, VPE(22)는 (예컨대, 글로벌 레지스터 파일(40)로부터)

필터 계수들을 수신할 수 있다. VPE(22)는 FIR 필터 기능을 수행하기 위해 수신된 필터 계수들을 직접적으로 사용할 수 있고, 이러한 경우에 위의 수학식에서 필터 계수들(h(l))은 수신된 필터 계수들을 나타낼 수 있다. 대안적으로, VPE(22)는 FIR 필터 기능을 수행하기 위해 수신된 필터 계수들을 사용하기 전에 그들의 복소 공액을 계산할 수 있고, 이러한 경우에, 위의 수학식에서 필터 계수들(h(l))은 수신된 필터 계수들의 공액들을 나타낼 수 있다.

[0038] 도 3에서 위의 이산 FIR 필터(64)는 다음과 같이 재구성될 수 있다.

[0039] y[n]=x[n]*h0 + x[n-1]*h1+ ... + x[n-7]*h7

[0040] 그러나, 도 3의 이산 FIR 필터(64)와 같은 필터링 연산은 벡터 프로세서에서 제공되는 특수화된 데이터 흐름 경로들로 인해 벡터 프로세서들에서 병렬화하기에 어려울 수 있다. 필터링될 입력 벡터 데이터 샘플 세트(예컨대, 벡터화된 디지털화된 입력 신호(66))가 필터 지연 탭들(예컨대, 68(1)-68(Y-1)) 사이에서 시프트될 때, 입력 벡터 데이터 샘플 세트는 벡터 데이터 파일로부터 재패치되고, 따라서 전력 소비를 증가시키고 스루풋을 감소시킨다. 벡터 데이터 파일로부터 입력 벡터 데이터 샘플 세트들의 재패치를 최소화하기 위해 벡터 프로세서 내의 데이터 흐름 경로는 효율적인 병렬화된 프로세싱을 위해 필터 지연 탭들(예컨대, 68(1)-68(Y-1))과 동일한 수의 곱셈기들(예컨대, 70(0)-70(Y-1))을 제공하도록 구성될 수 있다. 그러나, 다른 벡터 프로세싱 연산들은 더 적은 곱셈기들을 요구할 수 있고, 이로써 데이터 흐름 경로에서 곱셈기들의 비효율적인 스케일링 및 낮은 활용을 제공한다. 곱셈기들의 수가 확장성을 제공하기 위한 필터 지연 탭들의 수 미만으로 감소되면, 병렬화는 필터 프로세싱의 상이한 단계들에 대한 동일한 입력 벡터 데이터 샘플 세트를 획득하기 위해 메모리에 요구되는 더 많은 재패치들에 의해 제한된다.

[0041] 이와 관련하여, 도 4는 도 2의 VPE(22)로서 제공될 수 있는 예시적인 VPE(22(1))의 개략도이다. 아래에 더 상세히 설명될 바와 같이, 도 4의 VPE(22(1))는 벡터 데이터 샘플 재패치가 제거 또는 감소되고 전력 소비가 감소된, VPE(22(1)) 내의 정밀 필터 벡터 프로세싱 연산들을 제공한다. 벡터 데이터 샘플 재패치를 요구하는 중간 결과들의 저장을 요구하고 이로써 결과적으로 전력 소비를 증가시키는 필터 벡터 프로세싱 연산들과 비교하여 VPE(22(1))에서 정밀 필터 벡터 프로세싱 연산들이 제공될 수 있다. 전력 소비를 감소시키고 프로세싱 효율을 개선하기 위해 벡터 데이터 파일로부터 입력 벡터 데이터 샘플들의 재패치를 제거 또는 최소화하기 위해, 탭핑-지연 라인들(78)이 VPE(22(1)) 내의 실행 유닛들(84(0)-84(X))(또한 "EU"로 라벨링됨)과 벡터 데이터 파일들(82(0)-82(X)) 사이의 입력 데이터 흐름 경로들(80(0)-80(X))에 포함된다. 'X'+1은 이러한 예에서 벡터 데이터 샘플들의 프로세싱을 위해 VPE(22(1))에 제공되는 병렬 입력 데이터 레인들의 최대수이다. 탭핑-지연 라인들(78)은 입력 벡터 데이터 샘플 세트(86(0)-86(X))의 입력 벡터 데이터 샘플들(86)의 서브세트 또는 전부로서 탭핑-지연 라인 입력들(88(0)-88(X)) 상에서 입력 벡터 데이터 샘플 세트(86(0)-86(X))를 벡터 데이터 파일들(82(0)-82(X))의 대응하는 서브세트 또는 전부로부터 수신하도록 구성된다. 입력 벡터 데이터 샘플 세트(86(0)-86(X))는, 이러한 예에서 86(0), 86(1),..., 및 86(X)인 'X+1' 입력 벡터 데이터 샘플들(86)에 포함된다.

[0042] 도 4를 계속해서 참조하면, 탭핑-지연 라인들(78)은 필터 벡터 프로세싱 연산을 위해 실행 유닛들(84(0)-84(X))에 의해 프로세싱될 벡터 데이터 파일들(82(0)-82(X))로부터 패치된 입력 벡터 데이터 샘플 세트들(86(0)-86(X))을 저장한다. 아래의 도 6 및 7에 관련하여 아래에 더 상세히 논의될 바와 같이, 탭핑-지연 라인들(78)은 시프트된 입력 벡터 데이터 샘플 세트(86S(0)-86S(X))를 실행 유닛들(84(0)-84(X))에 제공하기 위해 VPE(22(1))에 의해 실행될 필터 벡터 명령에 따라 필터 벡터 프로세싱 동작의 각각의 필터 지연 탭(즉, 필터 프로세싱 스테이지)에 대한 입력 벡터 데이터 샘플 세트들(86(0)-86(X))을 시프트하도록 구성된다. 시프트된 입력 벡터 데이터 샘플들(86S) 모두는 시프트된 입력 벡터 데이터 샘플 세트(86S(0)-86S(X))를 포함한다. 탭핑-지연 라인들(78)은 필터 벡터 프로세싱 연산 동안에 실행 유닛들(84(0)-84(X))의 실행 유닛 입력들(90(0)-90(X))에 시프트된 입력 벡터 데이터 샘플(86S(0)-86S(X))을 제공한다. 이러한 방식으로, 필터 벡터 프로세싱 연산의 필터 탭들에 대한 시프트된 입력 벡터 데이터 샘플 세트(86S(0)-86S(X))에 대해 수행되는 연산들에 기초한 중간 필터 결과들은 VPE(22(1))에 의해 수행되는 필터 벡터 프로세싱 연산의 각각의 프로세싱 스테이지 동안에 저장, 시프트되고 그리고 벡터 데이터 파일들(82(0)-82(X))로부터 재패치되지 않아도 된다. 따라서, 탭핑-지연 라인들(78)은 전력 소비를 감소시키고, VPE(22(1))에 의해 수행되는 필터 벡터 프로세싱 연산들에 대한 프로세싱 효율을 증가시킬 수 있다.

[0043] "벡터 프로세싱 스테이지"로 또한 지칭되는, VPE(22(1)) 내의 프로세싱 스테이지는 특정 작업 또는 동작을 수행하도록 설계된 회로 및 연관된 벡터 데이터 경로들을 포함한다. 벡터 프로세싱 연산은 몇몇의 상이한 프로세싱

스테이지들에서 VPE(22(1))에 의해 실행될 수 있다. 각각의 프로세싱 스테이지는 VPE(22(1))의 하나 또는 다수의 클록 사이클들에 걸쳐 수행될 수 있다. 결과적으로, VPE(22(1)) 내의 벡터 프로세싱 연산의 실행은 완료하는데 많은 클록 사이클들이 걸릴 수 있는데, 왜냐하면 벡터 프로세싱 연산의 각각의 프로세싱 스테이지가 하나 또는 그 초과의 클록 사이클들 각각을 소비할 수 있기 때문이다. 예컨대, 프로세싱 스테이지는 도 4의 VPE(22(1))의 탭핑-지연 라인들(78)로의 입력 벡터 데이터 샘플 세트(86(0)-86(X))의 패치를 포함할 수 있다. VPE(22(1)) 내의 벡터 프로세싱 스테이지들이 파이프라이닝될 수 있다.

[0044] 실행 유닛들(84(0)-84(X))은 패치된 입력 벡터 데이터 샘플 세트(86(0)-86(X))를 프로세싱하는 하나 또는 그 초과의 파이프라인 스테이지들을 포함할 수 있다. 예컨대, 실행 유닛들(84(0)-84(X)) 내의 하나의 파이프라인 스테이지는 누산 연산들을 수행하도록 구성된 누산기들을 포함하는 누산 스테이지를 포함할 수 있다. 다른 예로서, 실행 유닛들(84(0)-84(X)) 내의 다른 파이프라인 스테이지는 곱셈 연산들을 수행하도록 구성된 곱셈기들을 포함하는 곱셈 스테이지를 포함할 수 있다.

[0045] 계속해서 도 4를 참조하면, 실행 유닛들(84(0)-84(X))은 필터 벡터 프로세싱 연산을 위해 도 2의 글로벌 레지스터 파일(40)에 저장된 필터 계수들(92(0)-92(Y-1)) 사이로부터 필터 계수(92)를 수신하고, 여기서 'Y'는 필터 벡터 프로세싱 연산을 위한 필터 계수들의 수와 동일할 수 있다. 실행 유닛들(84(0)-84(X)) 각각은 실행 유닛들(84(0)-84(X))에서 중간 필터 벡터 데이터 출력 샘플들을 제공하기 위해 벡터 필터 프로세싱 연산의 각각의 프로세싱 스테이지 동안에 수신된 필터 계수(92(0), 90(1),...90(Y-1)) 중 하나와 시프트된 입력 벡터 데이터 샘플 세트(86S(0)-86S(X))의 시프트된 입력 벡터 데이터 샘플(86S(0), 86S(1),...86S(X))를 곱셈하도록 구성된다. 중간 필터 벡터 데이터 출력 샘플 세트들은 실행 유닛들(84(0)-84(X)) 각각에서 누산된다(즉, 이전의 누산된 필터 출력 벡터 데이터 샘플이 현재 누산된 필터 출력 벡터 데이터 샘플에 가산됨). 이것은 시프트된 입력 벡터 데이터 샘플 세트(86S(0)-86S(X)) 내의 각각의 시프트된 입력 벡터 데이터 샘플 세트(86S(0), 86S(1),...86S(X))에 대해 출력 데이터 흐름 경로들(98(0)-98(X)) 상의 실행 유닛 출력들(96(0)-96(X)) 상에서 실행 유닛들(84(0)-84(X))에 의해 각각 제공되는 마지막, 결과적 필터 출력 벡터 데이터 샘플 세트(94(0)-94(X))를 제공한다. 결과적 필터 출력 벡터 데이터 샘플 세트(94(0)-94(X))는, 이러한 예에서, 94(0), 94(1),..., 및 94(X)인 'X+1' 결과적 필터 출력 벡터 데이터 샘플들(94)을 포함한다. 결과적 필터 출력 벡터 데이터 샘플 세트(94(0)-94(X))는, 실행 유닛들(84(0)-84(X))에 의해 생성된 중간 필터 벡터 데이터 출력 샘플 세트들을 저장 및 시프트하지 않고서, VPE(22(1))에 의한 추가의 사용 및/또는 프로세싱을 위해 각각의 벡터 데이터 파일들(82(0)-82(X))에 다시 저장된다.

[0046] 도 4를 계속 참조하고, 아래에 더 상세히 논의될 바와 같이, 탭핑-지연 라인들(78)은 프로세싱된 벡터 명령에 따라 제어되도록 프로그램가능하다. 필터 벡터 명령이 프로세싱되지 않는다면, 탭핑-지연 라인들(78)은 벡터 데이터 파일들(82(0)-82(X))과 실행 유닛들(84(0)-84(X)) 사이의 입력 데이터 흐름 경로들(80(0)-80(X))에 포함되지 않도록 프로그램될 수 있다. 이러한 실시예에서, 탭핑-지연 라인들(78)은 필터 벡터 프로세싱 연산의 각각의 필터 탭에 대한 시프트된 입력 벡터 데이터 샘플 세트(86S(0)-86S(X))를 제공하기 위해 벡터 데이터 파일들(82(0)-82(X))로부터 수신된 입력 벡터 데이터 샘플 세트(86(0)-86(X))를 로딩 및 시프트하도록 구성된다. 따라서, 시프트된 입력 벡터 데이터 샘플 세트(86S(0)-86S(X))는 필터 벡터 프로세싱 연산의 필터 탭의 실행을 위해 실행 유닛들(84(0)-84(X))에 제공될 수 있다. 탭핑-지연 라인들(78) 없이, 필터 벡터 프로세싱 연산의 후속 필터 탭들에 대해 시프트된 중간 입력 벡터 데이터 샘플 세트를 다시 실행 유닛들(84(0)-84(X))에 제공하기 위해 별개의 시프팅 프로세스가 수행되어야 할 것이고, 이로써 레이턴시를 증가시키고 부가적인 전력을 소비한다. 또한, VPE(22(1)) 내의 입력 및 출력 데이터 흐름 경로들(80(0)-80(X), 98(0)-98(X))의 효율은 필터 벡터 프로세싱 연산 동안에 벡터 데이터 파일들(82(0)-82(X))로부터 시프트된 입력 벡터 데이터 샘플 세트(86S(0)-86S(X))의 재패치 지연에 의해 제한되지 않는다.

[0047] 시프트된 입력 벡터 데이터 샘플 세트(86S(0)-86S(X))는 실행 유닛들(84(0)-84(X))에 로컬화된 탭핑-지연 라인들(78)에 의해 제공된다. 실행 유닛들(84(0)-84(X)) 내의 벡터 프로세싱은 데이터 흐름 제한들보다는 계산 자원들에 의해서만 제한된다. 이것은, 실행 유닛들(84(0)-84(X))이, 벡터 데이터 파일들(82(0)-82(X))로부터 패치될 시프트된 입력 벡터 데이터 샘플 세트(86S(0)-86S(X))를 대기해야 하지 않고서 벡터 프로세싱 연산들을 수행하기 위해 시프트된 입력 벡터 데이터 샘플 세트(86S(0)-86S(X))를 계속해서 또는 실질적으로 계속해서 수신하기에 바쁘다는 것을 의미한다.

[0048] 또한, 도 4의 VPE(22(1))에 의해 수행되는 필터 벡터 프로세싱 연산들은 탭핑-지연 라인들(78)을 사용함으로써 더 정밀할 수 있는데, 왜냐하면 실행 유닛들(84(0)-84(X)) 내의 중간 필터 프로세싱 스테이지들에 대한 출력 누산들이 벡터 데이터 파일들(82(0)-82(X))에 저장될 필요가 없기 때문이다. 실행 유닛들(84(0)-84(X))로부터의

중간 출력 벡터 데이터 샘플 세트들의 벡터 데이터 파일들(82(0)-82(X)) 내의 저장은 라운딩을 발생시킬 수 있다. 따라서, 다음 중간 출력 벡터 데이터 샘플 세트가 벡터 프로세싱 연산을 위해 실행 유닛들(84(0)-84(X))에 제공될 때, 임의의 라운딩 에러가 벡터 프로세싱 연산의 각각의 곱셈 단계 동안에 전파 및 부가될 것이다. 이와 반대로, 도 4의 VPE(22(1))의 예에서, 실행 유닛들(84(0)-84(X))에 의해 계산된 중간 출력 벡터 데이터 샘플 세트들은 벡터 데이터 파일들(82(0)-82(X))에 저장될 필요가 없다. 실행 유닛들(84(0)-84(X))은 다음 필터 지연 탭들에 대해 이전의 중간 출력 벡터 데이터 샘플 세트들과 중간 출력 벡터 데이터 샘플 세트들을 누산할 수 있는데, 왜냐하면 탭핑-지연 라인들(78)이 프로세싱될 벡터 프로세싱 연산 동안에 시프트된 입력 벡터 데이터 샘플 세트(86S(0)-86S(X))를 실행 유닛들(84(0)-84(X))에 제공하기 때문이고, 결과들은 이전의 필터 지연 탭들에 대해 이전의 벡터 데이터 샘플 세트들과 누산된다.

[0049] 도 4를 계속 참조하면, VPE(22(1))는 이러한 실시예에서 병렬화된 프로세싱을 위해 복수의 벡터 데이터 레인들(VLANE0-VLANEX)로 라벨링됨)(100(0)-100(X))을 포함한다. 각각의 벡터 데이터 레인(100(0)-100(X))은 이러한 실시예에서 벡터 데이터 파일(82) 및 실행 유닛(84)을 포함한다. 예로서 벡터 데이터 레인(100(0))을 취하면, 그 안의 벡터 데이터 파일(82(0))은 필터 벡터 프로세싱을 위해 실행 유닛(84(0))에 의해 수신될 입력 벡터 데이터 샘플(86(0))을 입력 데이터 흐름 경로(80(0)) 상에서 제공하도록 구성된다. 위에서 논의된 바와 같이, 입력 벡터 데이터 샘플(86(0))을 시프트하고 필터 벡터 프로세싱을 위해 시프트된 입력 벡터 데이터 샘플 세트(86S(0))를 실행 유닛(84(0))에 제공하기 위해, 탭핑-지연 라인들(78)이 입력 데이터 흐름 경로(80(0))에 제공된다. 벡터 데이터 파일(82(0))은 또한, VPE(22(1))에 의해 프로세싱될 현재 또는 다음 벡터 명령에 따라 필요로 되거나 요구될 때, 후속 벡터 프로세싱 연산을 위해 벡터 데이터 파일(82(0))에 다시 저장될 출력 데이터 흐름 경로(98(0))로부터의 필터 벡터 프로세싱의 결과로서 실행 유닛(84(0))에 의해 제공되는 결과적 필터 출력 벡터 데이터 샘플(94(0))을 수신하도록 구성된다.

[0050] 요구되는 바에 따라, 임의의 수의 벡터 데이터 레인들(100(0)-100(X))이 VPE(22(1))에 제공될 수 있다. VPE(22(1))에 제공되는 벡터 데이터 레인들(100(0)-100(X))의 수는, 부가적인 벡터 데이터 레인들(100(0)-100(X))을 제공하는 데에 수반되는 부가적인 회로, 공간, 및 전력 소비에 대한(versus), 효율성 목적들을 위한 병렬화된 벡터 프로세싱을 위한 트레이드오프들에 기초할 수 있다. 하나의 비제한적인 예로서, 16개의 벡터 데이터 레인들(100)이 VPE(22(1))에 제공될 수 있으며, 각각의 벡터 데이터 레인(100)은, VPE(22(1)) 내에 벡터 데이터의 최대 512 비트의 병렬화된 프로세싱을 제공하기 위해, 삼십이(32) 비트들의 데이터 폭 능력을 갖는다.

[0051] 도 4를 계속해서 참조하면, 일 예로서, 하지만 모든 벡터 데이터 파일들(82(0)-82(X))에 대해 적용가능한 예로서, 벡터 데이터 레인(100(0)) 내의 벡터 데이터 파일(82(0))을 사용하여, 벡터 데이터 파일(82(0))은 입력 벡터 데이터 샘플(86(0)) 중 하나 또는 다수의 샘플들이 벡터 프로세싱을 위해 저장되도록 허용한다. 입력 벡터 데이터 샘플(86(0))의 폭은, VPE(22(1))에 의해 실행되는 특정 벡터 명령에 따른 입력 벡터 데이터 샘플(86(0))의 프로그래밍에 따라 제공된다. 입력 데이터 흐름 경로(80(0))의 폭은, 탭핑-지연 라인들(78) 및 실행 유닛(84(0))에 입력 벡터 데이터 샘플(86(0))의 상이한 폭들을 제공하기 위해, 벡터 명령 단위(vector-instruction-by-vector-instruction basis)(주어진 벡터 명령에 대한 클록 사이클 단위(clock-cycle-by-clock-cycle basis)를 포함함)로 프로그램가능하고 재프로그램가능하다. 이러한 방식으로, 벡터 데이터 레인(100(0))은, 실행되는 벡터 명령의 타입에 따라, 입력 벡터 데이터 샘플(86(0))의 상이한 폭들의 프로세싱을 제공하기 위해 프로그램되고 재프로그램될 수 있다.

[0052] 예컨대, 벡터 데이터 파일(82(0))은 삼십이(32) 비트 폭(wide)일 수 있고, 또한 최대 삼십이(32) 비트 폭일 수 있는 입력 벡터 데이터 샘플들(86)을 저장할 수 있다. 입력 벡터 데이터 샘플(86(0))은 벡터 데이터 파일(82(0))의 전체 폭(예컨대, 32 비트들)을 소비할 수 있거나, 또는 벡터 데이터 파일(82(0)) 폭의 더 작은 샘플 크기들로 제공될 수 있다. 입력 벡터 데이터 샘플(86(0)) 크기는, VPE(22(1))에 의해 실행되는 벡터 명령에 기초하는 입력 벡터 데이터 샘플(86(0))의 크기에 대한 입력 데이터 흐름 경로(80(0)) 구성의 프로그래밍에 기초하여 구성될 수 있다. 예컨대, 입력 벡터 데이터 샘플(86(0))은 하나의 벡터 명령에 대해 두개(2)의 개별적인 16 비트 벡터 데이터 샘플들을 포함할 수 있다. 다른 예로서, 입력 벡터 데이터 샘플(86(0))은, 한개(1)의 32 비트 벡터 데이터 샘플과 대조적으로, 다른 벡터 명령에 대해 벡터 데이터 파일(82(0))에 네개(4)의 8 비트 벡터 데이터 샘플들을 포함할 수 있다. 다른 예에서, 입력 벡터 데이터 샘플(86(0))은 한개(1)의 32 비트 벡터 데이터 샘플을 포함할 수 있다. VPE(22(1))는 또한, 각각의 벡터 명령 및/또는 주어진 벡터 명령의 각각의 클록 사이클에 대해 벡터 데이터 파일(82(0))에 실행 유닛(84(0))에 의해 제공되는 결과적 필터 출력 벡터 데이터 샘플들(94(0))의 상이한 크기들을 수신하기 위해, 벡터 데이터 파일(82(0))에 대한 출력 데이터 흐름 경로(98(0))를 프로그램 및 재프로그램할 수 있다.

[0053] 이제, 본 실시예에서 입력 데이터 흐름 경로들(80(0)-80(X)) 내의 실행 유닛들(84(0)-84(X))에 시프트된 입력 벡터 데이터 샘플 세트(86S(0)-86S(X))를 제공하기 위한 탭핑-지연 라인들(78) 및 도 4의 VPE(22(1))의 부가적인 세부사항들 및 피쳐들의 추가의 설명이 설명될 것이다. 이와 관련하여, 도 5는, 예시적인 필터 벡터 명령에 따라, 탭핑-지연 라인들(78)을 이용하여 도 4의 VPE(22(1))에서 수행될 수 있는 예시적인 필터 벡터 프로세싱 연산(102)을 예시하는 흐름도이다. 도 5의 필터 벡터 프로세싱 연산(102)에서 수행되는 예시적인 태스크들이 도 6a 내지 도 10에 제공되는 예들과 관련하여 설명될 것이다.

[0054] 도 5를 참조하여, 필터 벡터 프로세싱 연산(102)을 위해, 필터 벡터 명령에 따라 필터 벡터 프로세싱 연산(102)에서 프로세싱될 입력 벡터 데이터 샘플 세트(86(0)-86(X))가 벡터 데이터 파일들(82(0)-82(X))로부터 입력 데이터 흐름 경로들(80(0)-80(X))로 패치된다(블록(104)). 도 4에서 VPE(22(1))와 관련하여 상기 논의된 바와 같이, 입력 벡터 데이터 샘플 세트(86(0)-86(X))에는, 실행 유닛들(84(0)-84(X)) 내의 글로벌 레지스터 파일(40)로부터 수신된 필터 계수들(92(0)-92(Y-1))이 곱해진다. 예컨대, 도 6a는 글로벌 레지스터 파일(40) 내의 필터 계수들(92(0)-92(Y-1))(즉, h7-h0)을 예시한다. 이러한 예에서, 수행될 필터 벡터 프로세싱 연산(102)에 여덟개(8)의 필터 탭들을 제공하는 글로벌 레지스터 파일(40)에 저장된 여덟개(8)의 필터 계수들(92)이 존재한다. 이러한 예에서, 상기 논의된 도 3의 이산 FIR 필터(64) 방정식으로부터의 필터 벡터 프로세싱 연산(102)은 y[n] = x[n]*h0 + x[n-1]*h1+ ... + x[n-7]*h7 임을 주목한다.

[0055] 도 6b는 필터 벡터 프로세싱 연산(102)에 의해 필터링될 입력 신호를 나타내는, 도 4의 VPE(22(1)) 내의 벡터 데이터 파일들(82(0)-82(X)) 내에 저장되는 예시적인 입력 벡터 데이터 샘플 세트(86(0)-86(X))를 예시한다. 이러한 예에서, 샘플(X0)은 가장 오래된 샘플이고, 샘플(X63)은 가장 최근의 샘플이다. 다시 말해, 이러한 예에서, 샘플(X63)은 시간적으로 샘플(X0) 이후에 발생한다. 도 6b에 도시된 바와 같이, 벡터 데이터 파일들(82(0)-82(X))의 각각의 어드레스는 16 비트 폭이기 때문에, 벡터 데이터 파일들(82(0)-82(X))에 저장된 제 1 입력 벡터 데이터 샘플 세트(86(0)-86(X))는 ADDRESS 0 및 ADDRESS 1에 걸친다(span). 이는, 도 4의 VPE(22(1)) 예에서의 실행 유닛들(84(0)-84(X))의 32 비트 폭 능력을 지원하기 위해, 벡터 데이터 파일들(82(0)-82(X))이 32 비트 폭의 입력 벡터 데이터 샘플들(86)을 제공하도록 허용한다. 이와 관련하여, 제 1 입력 벡터 데이터 샘플 세트(86(0)-86(X))를 포함하는, 육십네개(64)의 총 입력 벡터 데이터 샘플 서브세트들(즉, X0-X63)이 존재하며, 각각의 8 비트 폭은 총 512 비트들이다. 유사하게, ADDRESS 2 및 ADDRESS 3은, 벡터 데이터 파일들(82(0)-82(X)) 내에 저장된 다른 제 2 입력 벡터 데이터 샘플 세트(86(0)-86(X))를 저장한다. 도 6b의 이러한 예에서는, 256개의 총 입력 벡터 데이터 샘플들(86)(즉, X0-X255)을 예시하는, 각각의 벡터 데이터 파일(82(0)-82(X))의 여덟개(8)의 어드레스들(ADDRESS 0-7)이 도시되어 있지만, 이는 제한적인 것이 아님을 주목한다.

[0056] 도 4의 VPE(22(1)) 내의 벡터 데이터 레인들(100(0)-100(X)) 중에서 어느 하나, 일부, 또는 전부는, 필터 벡터 프로세싱 연산(102)에 수반되는 입력 벡터 데이터 샘플 세트(86(0)-86(X))의 폭에 따라서 벡터 명령의 프로그래밍에 따라 필터 벡터 프로세싱 연산(102)을 제공하기 위해 이용될 수 있다. 벡터 데이터 파일들(82(0)-82(X))의 전체 폭이 요구되는 경우, 모든 벡터 데이터 레인들(100(0)-100(X))이 필터 벡터 프로세싱 연산(102)을 위해 이용될 수 있다. 필터 벡터 프로세싱 연산(102)은, 필터 벡터 프로세싱 연산(102)을 위해 이용될 수 있는 벡터 데이터 레인들(100(0)-100(X))의 서브세트만을 요구할 수 있음을 주목한다. 이는, 입력 벡터 데이터 샘플 세트(86(0)-86(X))의 폭이 모든 벡터 데이터 파일들(82(0)-82(X))의 폭 미만이기 때문일 수 있는데, 이 경우, 필터 벡터 프로세싱 연산(102)과 동시에 수행될 다른 벡터 프로세싱 연산들을 위해 부가적인 벡터 데이터 레인들(100)을 이용할 것이 요구된다. 본 예를 논의하는 목적들로, 필터 벡터 프로세싱 연산(102)에서 이용되는 입력 벡터 데이터 샘플 세트(86(0)-86(X))는 모든 벡터 데이터 레인들(100(0)-100(X))을 수반하는 것으로 가정한다.

[0057] 도 5를 다시 참조하면, 벡터 데이터 파일들(82(0)-82(X))로부터의 패치된 입력 벡터 데이터 샘플 세트(86(0)-86(X))가 입력 데이터 흐름 경로들(80(0)-80(X))에 제공되어, 탭핑-지연 라인들(78) 내로 현재 입력 벡터 데이터 샘플 세트(86(0)-86(X))로서 로딩된다(블록(106)). 입력 벡터 데이터 샘플 세트(86(0)-86(X))는, 필터 벡터 프로세싱 연산(102)을 위해 실행 유닛들(84(0)-84(X))에 의해 프로세싱될 입력 벡터 데이터 샘플 세트(86(0)-86(X))로서 1차 탭핑-지연 라인(78(0)) 내로 로딩된다. 1차 탭핑-지연 라인(78(0)) 내로 로딩된 입력 벡터 데이터 샘플 세트(86(0)-86(X))는 필터 벡터 프로세싱 연산(102)의 제 1 필터 탭 연산을 위해 시프트되지 않는다. 하지만, 상기 논의된 바와 같이 그리고 도 7과 관련하여 하기에서 더 상세히 논의되는 바와 같이, 탭핑-지연 라인들(78)의 목적은, 필터 벡터 프로세싱 연산(102)의 이후의 필터 탭 연산들을 위해, 시프트된 입력 벡터 데이터 샘플 세트(86S(0)-86S(X))를 실행 유닛들(84(0)-84(X))에 제공하기 위하여 입력 벡터 데이터 샘플

세트(86(0)-86(X))의 시프트를 제공하는 것이다. 실행 유닛들(84(0)-84(X))에 의해 실행되는 필터 벡터 프로세싱 연산(102)의 각각의 프로세싱 스테이지 동안, 시프트된 입력 벡터 데이터 샘플 세트(86S(0)-86S(X))를 실행 유닛들(84(0)-84(X))에 제공하기 위해, 입력 벡터 데이터 샘플들(86)은 1차 탭핑-지연 라인(78(0))에서 시프트된다. 이러한 방식으로, 입력 벡터 데이터 샘플 세트(86(0)-86(X))는, 필터 벡터 프로세싱 연산(102)의 각각의 필터 탭 연산을 위해, 저장되고, 벡터 데이터 파일들(82(0)-82(X)) 내에서 시프트되고, 재패치될 필요가 없다.

[0058] 선택적인 섀도우 탭핑-지연 라인(78(1))이 VPE(22(1))에 제공되는 경우, 다음 입력 벡터 데이터 샘플 세트(86N(0)-86N(X))가 또한, 벡터 데이터 파일들(82(0)-82(X))로부터 섀도우 탭핑-지연 라인(78(1)) 내로 로딩될 수 있다. 도 7과 관련하여 하기에서 더 상세히 논의될 바와 같이, 다음 입력 벡터 데이터 샘플 세트(86N(0)-86N(X))는, 시프트된 입력 벡터 데이터 샘플 세트(86S(0)-86S(X))의 적어도 일부가 되도록, 필터 벡터 프로세싱 연산(102) 동안 1차 탭핑-지연 라인(78(0)) 내로 시프트된다. 따라서, 필터 벡터 프로세싱 연산(102)을 위해 실행될 다음 입력 벡터 데이터 샘플 세트(86N(0)-86N(X))가 벡터 데이터 파일들(82(0)-82(X))로부터 1차 탭핑-지연 라인(78(0))으로 패치될 때까지, 실행 유닛들(84(0)-84(X))이 대기할 것을 요구하였을 경우에 초래되었을 패치 지연없이, 1차 탭핑-지연 라인(78(0))이, 필터 벡터 프로세싱 연산(102) 동안 이용가능한 시프트된 입력 벡터 데이터 샘플 세트(86S(0)-86S(X))를 가질 수 있다.

[0059] 이와 관련하여, 도 7은 도 4에서 VPE(22(1)) 내에 제공될 수 있는 예시적인 탭핑-지연 라인들(78)을 예시한다. 이러한 실시예에서, 탭핑-지연 라인들(78)은 섀도우 탭핑-지연 라인(78(1)) 및 1차 탭핑-지연 라인(78(0))을 포함한다. 이러한 예에서 1차-탭핑 지연 라인(78(0))은, 길이에 있어서 8 비트들까지로의 입력 벡터 데이터 샘플들(86)의 분해능을 허용하기 위해, 복수의 8 비트 1차 파이프라인 레지스터들(120)로 구성된다. 하기에서 도 9a와 관련하여 논의될 바와 같이, 본 예에서, 실행 유닛들(84(0)-84(X))에 의해 프로세싱되는 제 1 입력 벡터 데이터 샘플 세트(86(0)-86(X))는 필터 벡터 프로세싱 연산(102)의 제 1 필터 탭을 위해 시프트되지 않을 것이다. 실행 유닛들(84(0)-84(X))은 필터 벡터 프로세싱 연산(102)을 위해 이후의 필터 탭들을 프로세싱하기 때문에, 1차 탭핑-지연 라인(78(0))에 저장된 입력 벡터 샘플 세트(86(0)-86(X)) 내의 입력 벡터 데이터 샘플들(86)은, 도 7에서 화살표들에 의해 나타낸 바와 같이, 시프트된 입력 벡터 데이터 샘플 세트(86S(0)-86S(X))가 되도록, 1차 파이프라인 레지스터들(120(0)-120(4X+3))에서 시프트된다. 이러한 방식으로, 입력 벡터 데이터 샘플 세트(86S(0)-86S(X))를 저장 및 시프트하고, 벡터 데이터 파일들(82(0)-82(X))로부터, 시프트된 입력 벡터 데이터 샘플 세트(86S(0)-86S(X))를 재패치할 필요없이, 시프트된 입력 벡터 데이터 샘플 세트(86S(0)-86S(X))의 필터 벡터 프로세싱 연산 (102)을 수신하고 수행함으로써, 실행 유닛들(84(0)-84(X))이 완전히 활용된다.

[0060] 이러한 실시예에서, 1차 파이프라인 레지스터들(120(0)-120(4X+3))이 집합적으로, 도 4에서의 벡터 데이터 파일들(82(0)-82(X))의 폭이다. 폭이 512 비트인 벡터 데이터 파일들(82(0)-82(X))("X"는 십오(15)임)의 예에서는, 512 비트들의 총 폭을 제공하기 위해 육십네개(64)의 총 1차 파이프라인 레지스터들(120(0)-120(63))(각각 팔(8) 비트 폭임)이 존재할 것이다(즉, 각각 64 레지스터들 x 8 비트들). 따라서, 이러한 예에서, 1차 탭핑-지연 라인(78(0))은 한개(1)의 입력 벡터 데이터 샘플 세트(86(0)-86(X))의 전체 폭을 저장할 수 있다. 본 예에서는 팔(8) 비트 폭의 1차 파이프라인 레지스터들(120(0)-120(4X+3))을 제공함으로써, 입력 벡터 데이터 샘플 세트(86(0)-86(X))는 8 비트 필터 벡터 프로세싱 연산들에 대해 팔(8) 비트들의 벡터 데이터 샘플 크기까지 아래로 1차 파이프라인 레지스터들(120(0)-120(4X+3)) 내에서 시프트될 수 있다. 필터 벡터 프로세싱 연산을 위해, 예컨대 16 비트 또는 32 비트 샘플들과 같은, 더 큰 크기의 입력 벡터 데이터 샘플(86) 크기들이 요구되는 경우, 입력 벡터 데이터 샘플 세트(86(0)-86(X))는 한 번에 두개(2)의 1차 파이프라인 레지스터들(120) 만큼 1차 파이프라인 레지스터들(120(0)-120(4X+3)) 내에서 시프트될 수 있다.

[0061] 도 7을 계속해서 참조하면, 섀도우 탭핑-지연 라인(78(1))이 또한, 탭핑-지연 라인(78) 내에 제공된다. 섀도우 탭핑-지연 라인(78(1))은, 이후의 벡터 프로세싱 연산을 위해 벡터 데이터 파일들(82(0)-82(X))로부터 다음 입력 벡터 데이터 샘플 세트(86N(0)-86N(X))를 래칭 또는 파이프라이닝하는 데에 사용될 수 있다. 필터 벡터 프로세싱 연산(102)에 대한 각각의 필터 탭이 실행 유닛들(84(0)-84(X))에 의해 실행될 때, 다음 입력 벡터 데이터 샘플 세트(86N(0)-86N(X))로부터의 다음 입력 벡터 데이터 샘플들(86N)은, 섀도우 탭핑-지연 라인(78(1))으로부터 1차 탭핑-지연 라인(78(0))으로 시프트된다. 섀도우 탭핑-지연 라인(78(1)) 또한, 1차 탭핑-지연 라인(78(0))과 유사하게, 길이에 있어서 8 비트들까지로의 입력 벡터 데이터 샘플들(86)의 분해능을 허용하기 위해, 복수의 8 비트 섀도우 파이프라인 레지스터들(122)로 구성된다. 1차 파이프라인 레지스터들(120(0)-120(4X+3))처럼, 섀도우 탭핑-지연 라인(78(1)) 내에 제공되는 섀도우 파이프라인 레지스터들(122(0)-122(4X+3))은 집합적으로 벡터 데이터 파일들(82(0)-82(X))의 폭인데, 이는 본 예에서 512 비트들이다. 따라서, 섀도우 탭핑-지연

라인(78(1))의 섀도우 파이프라인 레지스터들(122(0)-122(4X+3)) 또한, 한개(1)의 입력 벡터 데이터 샘플 세트(86(0)-86(X))의 전체 폭을 저장할 수 있다. 따라서, 이러한 실시예에서, 1차 탭핑-지연 라인(78(0)) 내에 포함되는 섀도우 파이프라인 레지스터들(122(0)-122(4X+3))의 수는 벡터 데이터 레인들(100(0)-100(X))의 수의 4배이며, 이는 본 예에서 총 열여섯개(16)이다(즉, X=15). 따라서, 섀도우 파이프라인 레지스터들(122)의 수 또한, 총 512 비트들에 대해 본 예에서 총 육십네개(64) 이다(즉, 각각 64 레지스터들 x 8 비트들). 1차 탭핑-지연 라인(78(0))과 관련하여 상기 논의된 바와 같이, 본 예에서 팔(8) 비트 폭의 섀도우 파이프라인 레지스터들(122(0)-122(4X+3))을 제공함으로써, 다음 입력 벡터 데이터 샘플 세트(86N(0)-86N(X))는 8 비트 필터 벡터 프로세싱 연산들에 대해 팔(8) 비트들의 벡터 데이터 샘플 크기까지 아래로 시프트될 수 있다.

[0062] 도 8은 도 7에서의 1차 및 섀도우 탭핑-지연 라인들(78(0), 78(1)) 내에 존재하는 선택된 1차 파이프라인 및 섀도우 파이프라인 레지스터들(120, 122)을 예시하는 개략도이다. 도 8은 1차 및 섀도우 파이프라인 레지스터들(120, 122) 간에 입력 벡터 데이터 샘플들(86)을 시프트시키는 예를 논의하는 것을 용이하게 하기 위해 제공된다. 상기 논의된 바와 같이, 입력 벡터 데이터 샘플들(86)은 또한, 1차 및 섀도우 탭핑-지연 라인들(78(0), 78(1)) 내에서 뿐만 아니라, 섀도우 탭핑-지연 라인(78(1))으로부터 1차 탭핑-지연 라인(78(0))으로 시프트될 수 있다. 파이프라인 레지스터들(120, 122)은, 요구되는 경우 8 비트의 분해능으로 시프트하는 입력 벡터 데이터 샘플(86)을 허용하기 위해, 본 예에서 각각 8 비트 폭이다. 이는 하기에서 보다 상세히 논의될 것이다. 1차 및 섀도우 탭핑-지연 라인들(78(0), 78(1))은 또한, 하기에서 보다 상세히 또한 논의될 바와 같이, 입력 벡터 데이터 샘플들(86)의 16 비트 및 32 비트 시프팅 분해능을 수행할 수 있다.

[0063] 이와 관련하여, 도 8은, 도 7에서의 1차 탭핑-지연 라인(78(0)) 내의 입력 벡터 데이터 샘플(86S(X))에 대한 저장 레지스터들을 형성하는 1차 파이프라인 레지스터들(120(4X+3), 120(2X+1), 120(4X+2), 및 120(2X))로의 입력 벡터 데이터 샘플들(86)의 시프팅을 예시한다. 1차 파이프라인 레지스터들(120(4X+3) 및 120(4X+2))은 각각, 도 7에서의 1차 탭핑-지연 라인(78(0)) 내의 레지스터들(B_{31} 및 B_{30})이다. 1차 파이프라인 레지스터들(120(2X+1) 및 120(2X))은 각각, 도 7에서의 1차 탭핑-지연 라인(78(0)) 내의 레지스터들(A_{31} 및 A_{30})이다. 도 7에 예시된 바와 같이, 레지스터들(B_{31} 및 B_{30})에 대한 1차 파이프라인 레지스터들(120(4X+3) 및 120(4X+2))은, 섀도우 탭핑-지연 라인(78(1)) 내의 인접하는 섀도우 파이프라인 레지스터들(122)로부터, 시프트된 입력 벡터 데이터 샘플들(86)을 수신하도록 구성된다. 따라서, 도 8의 예에서, 레지스터들(A'_0 및 A'_1)에 대한 섀도우 파이프라인 레지스터들(122(0), 122(1))은 각각, 입력 벡터 데이터 샘플들(86)을 B_{31} 및 B_{30}에 대한 1차 파이프라인 레지스터들(120(4X+3) 및 120(4X+2))로 시프트시키도록 구성된 것으로 예시된다. 유사하게, 도 8의 예에서, 1차 탭핑-지연 라인(78(0)) 내의 레지스터들(B_1 및 B_0)에 대한 1차 파이프라인 레지스터들(120(2X+3) 및 120(2X+2))은 각각, 입력 벡터 데이터 샘플들(86)을 레지스터들(A_{31} 및 A_{30})에 대한 인접하는 1차 파이프라인 레지스터들(120(2X+1) 및 120(2X))로 시프트시키도록 구성된 것으로 예시된다. 이제, 이들 레지스터들 간의 입력 벡터 데이터 샘플들(86)의 예시적인 시프팅이 논의될 것이다.

[0064] 도 8을 계속해서 참조하면, 입력 벡터 데이터 샘플들(86)의 시프팅 뿐만 아니라, 도 4에서의 벡터 데이터 파일들(82(0)-82(X))로부터 새로운 입력 벡터 데이터 샘플 세트들(86(0)-86(X))을 로딩하기 위해 1차 및 섀도우 파이프라인 레지스터들(120, 122)을 구성하는 유연성을 제공하기 위하여, 입력 벡터 데이터 샘플 선택기가 1차 및 섀도우 파이프라인 레지스터들(120, 122) 각각과 연관된다. 이와 관련하여, 입력 벡터 데이터 샘플 선택기들(124(0)-124(4X+3))이 각각, 1차 탭핑-지연 라인(78(0)) 내의 1차 파이프라인 레지스터들(120(0)-120(4X+3))로 시프팅 또는 로딩된 벡터 데이터에 대해 제공된다. 입력 벡터 데이터 샘플 선택기들(126(0)-126(4X+3))이 각각, 섀도우 탭핑-지연 라인(78(1)) 내의 섀도우 파이프라인 레지스터들(122(0)-122(4X+3))로 시프팅 또는 로딩된 벡터 데이터에 대해 제공된다. 입력 벡터 데이터 샘플 선택기들(124(0)-124(4X+3)) 및 입력 벡터 데이터 샘플 선택기들(126(0)-126(4X+3))은 본 예에서 각각 멀티플렉서이다. 하기에서 보다 상세히 논의될 바와 같이, 입력 벡터 데이터 샘플 선택기들(124(0)-124(4X+3), 126(0)-126(4X+3))은, 1차 및 섀도우 파이프라인 레지스터들(120(0)-120(4X+3), 122(0)-122(4X+3))로 시프팅 또는 로딩될 입력 벡터 데이터를 선택하기 위해 데이터 폭 시프트 제어 입력들(125)에 의해 각각 제어될 수 있다.

[0065] 도 8에서는, 레지스터들(B_{31}, B_{30}, A_{31} 및 A_{30}) 각각에 대응하는 1차 파이프라인 레지스터들(120(4X+3), 120(4X+2), 120(2X+1), 120(2X)) 각각에 대한 입력 벡터 데이터 샘플 선택기들(124(4X+3), 124(4X+2), 124(2X+1), 124(2X)) 만이 도시되어 있음을 주목한다. 도 8에는, 레지스터들(A'_1, A'_0, B_1, 및 B_0) 각각에 대응하는 파이프라인 레지스터들(122(1), 122(0), 120(2X+3), 120(2X+2)) 각각에 대한 입력 벡터 데이터 샘플 선택

기들(126(1), 126(0), 124(2X+3), 124(2X+2)) 만이 도시된다.

[0066] 도 8을 계속해서 참조하면, 벡터 프로세싱 연산을 위해 1차 및 섀도우 탭핑-지연 라인들(78(0), 78(1)) 내로 새로운 입력 벡터 데이터가 로딩될 예정인 경우, 입력 벡터 데이터 샘플 선택기들(124(4X+3), 124(4X+2), 124(2X+1), 124(2X))로 하여금 로드 데이터 흐름 경로들(133(4X+3), 133(4X+2), 133(2X+1), 133(2X))을 선택하도록 하기 위해, 도 4에서의 VPE(22(1))에 의해 데이터 폭 시프트 제어 입력들(125)이 구성될 수 있다. 로드 데이터 흐름 경로들(133(4X+3), 133(4X+2), 133(2X+1), 133(2X))을 선택하는 것은, 벡터 데이터 파일들(82(0)-82(X))로부터의 입력 벡터 데이터가 1차 파이프라인 레지스터들(120(4X+3), 120(4X+2), 120(2X+1), 120(2X)) 내에 저장되도록 허용한다. 벡터 데이터 파일들(82(0)-82(X))로부터의 입력 벡터 데이터를 로딩하는 것은, 예로서 VPE(22(1))에 의해 프로세싱될 새로운 또는 다음 벡터 명령에 대해 수행될 수 있다. 유사하게, 데이터 폭 시프트 제어 입력들(125)은 또한, 입력 벡터 데이터 샘플 선택기들(126(1), 124(2X+3), 126(0), 124(2X+2))로 하여금 입력 데이터 흐름 경로들(135(1), 133(2X+3), 135(0), 133(2X+2))을 선택하도록 하기 위해, 도 4에서의 VPE(22(1))에 의해 구성될 수 있다. 로드 데이터 흐름 경로들(135(1), 133(2X+3), 135(0), 133(2X+2))을 선택하는 것은 벡터 데이터 파일들(82(0)-82(X))로부터의 입력 벡터 데이터가 파이프라인 레지스터들(122(1), 120(2X+3), 124(0), 120(2X+2)) 내에 저장되도록 허용한다.

[0067] 도 8을 계속해서 참조하면, 1차 탭핑-지연 라인(78(0)) 및 섀도우 탭핑-지연 라인(78(1)) 내에 저장된 벡터 데이터가 벡터 프로세싱 연산을 위해 시프트될 것이 요구되는 경우, 데이터 폭 시프트 제어 입력들(125)은, 입력 벡터 데이터 샘플 선택기들(124(4X+3), 124(4X+2), 124(2X+1), 124(2X))로 하여금 벡터 데이터 샘플 시프팅을 위해 입력 데이터 흐름 경로들(137(4X+3), 137(4X+2), 137(2X+1), 137(2X))을 선택하도록 하기 위해, 도 4의 VPE(22(1))에 의해 구성될 수 있다. 데이터 폭 시프트 제어 입력들(125)은 또한, 입력 벡터 데이터 샘플 선택기들(126(1), 124(2X+3), 126(0), 124(2X+2))이 벡터 데이터 샘플 시프팅을 위해 입력 데이터 흐름 경로들(139(1), 137(2X+3), 139(0), 137(2X+2))을 선택하게 한다. 그 내에서 예시되는 바와 같이, 입력 벡터 데이터 샘플 선택기들(124(4X+3), 124(4X+2), 124(2X+1), 124(2X)) 및 입력 벡터 데이터 샘플 선택기들(126(1), 124(2X+3), 126(0), 124(2X+2)) 각각은, 벡터 데이터가 다른 레지스터들로 시프트될 수 있게 허용하는, 출력 데이터 흐름 경로들(141(4X+3), 141(4X+2), 141(2X+1), 141(2X) 및 143(1), 141(2X+3), 143(0), 124(2X+2))을 각각 포함한다. 도 8에 도시된 출력 데이터 흐름 경로들은, 이제 전체로서 도시되지만, 1차 탭핑-지연 라인(78(0)) 내의 입력 벡터 데이터 샘플 선택기들(124(0)-124(4X+3)) 및 섀도우 탭핑-지연 라인(78(1)) 내의 입력 벡터 데이터 샘플 선택기들(126(0)-126(4X+3)) 각각에 대해 포함되는 출력 데이터 흐름 경로들(141(0)-141(4X+3) 및 143(0)-143(4X+3))의 일부이다.

[0068] 예들로서, 8 비트 벡터 데이터 시프트 동안, 입력 벡터 데이터 샘플 선택기들(124(4X+3), 124(4X+2), 124(2X+1), 124(2X)) 및 입력 벡터 데이터 샘플 선택기들(126(1), 124(2X+3), 126(0), 124(2X+2))은 입력 데이터 흐름 경로들(137(4X+3), 137(4X+2), 137(2X+1), 137(2X), 139(1), 137(2X+3), 139(0), 137(2X+2))을 각각 선택하도록 구성된다. 이와 관련하여, 일례로, 1차 파이프라인 레지스터(120(2X+1))(즉, A_{31})에서의 벡터 데이터는 도 8에 예시된 바와 같이, 출력 데이터 흐름 경로(141(2X+1)) 상에서 1차 파이프라인 레지스터(120(2X))(즉, A_{30})로 시프트된다. 1차 파이프라인 레지스터(120(4X+3))(즉, B_{31})에서의 벡터 데이터는 도 8에 예시된 바와 같이, 출력 데이터 흐름 경로(141(4X+3)) 상에서 1차 파이프라인 레지스터(120(4X+2))(즉, B_{30})로 시프트된다. 섀도우 파이프라인 레지스터(122(0))(즉, A'_0)에서의 벡터 데이터는 도 8에 예시된 바와 같이, 출력 데이터 흐름 경로(143(0)) 상에서 1차 파이프라인 레지스터(120(4X+3))(즉, B_{31})로 시프트된다. 1차 파이프라인 레지스터(120(2X+3))(즉, B_1)에서의 벡터 데이터는 도 8에 예시된 바와 같이, 출력 데이터 흐름 경로(141(2X+3)) 상에서 1차 파이프라인 레지스터(120(4X+2))(즉, B_{30})로 시프트된다. 섀도우 파이프라인 레지스터(122(1))(즉, A'_1)에서의 벡터 데이터는 도 8에 예시된 바와 같이, 출력 데이터 흐름 경로(143(1)) 상에서 섀도우 파이프라인 레지스터(122(0))(즉, A'_0)로 시프트된다. 1차 파이프라인 레지스터(120(2X+2))(즉, B_0)에서의 벡터 데이터는 도 8에 예시된 바와 같이, 출력 데이터 흐름 경로(141(2X+2)) 상에서 1차 파이프라인 레지스터(120(2X+1))(즉, A_{31})로 시프트된다.

[0069] 도 8을 계속해서 참조하면, 16 비트 벡터 데이터 시프트 동안, 입력 벡터 데이터 샘플 선택기들(124(4X+3), 124(4X+2), 124(2X+1), 124(2X)) 및 입력 벡터 데이터 샘플 선택기들(126(1), 124(2X+3), 126(0), 124(2X+2))은 입력 데이터 흐름 경로들(145(4X+3), 145(4X+2), 145(2X+1), 145(2X), 147(1), 145(2X+3), 147(0),

145(2X+2))을 각각 선택하도록 구성된다. 이와 관련하여, 일례로, 1차 파이프라인 레지스터(120(2X+2))(즉, B_0)에서의 벡터 데이터는 도 8에 예시된 바와 같이, 출력 데이터 흐름 경로(141(2X+2)) 상에서 1차 파이프라인 레지스터(120(2X))(즉, A_{30})로 시프트된다. 섀도우 파이프라인 레지스터(122(0))(즉, A'_0)에서의 벡터 데이터는 도 8에 예시된 바와 같이, 출력 데이터 흐름 경로(143(0)) 상에서 1차 파이프라인 레지스터(120(4X+2))(즉, B_{30})로 시프트된다. 1차 파이프라인 레지스터(120(2X+3))(즉, B_1)에서의 벡터 데이터는 도 8에 예시된 바와 같이, 출력 데이터 흐름 경로(141(2X+3)) 상에서 1차 파이프라인 레지스터(120(2X+1))(즉, A_{31})로 시프트된다. 섀도우 파이프라인 레지스터(122(1))(즉, A'_1)에서의 벡터 데이터는 도 8에 예시된 바와 같이, 출력 데이터 흐름 경로(143(1)) 상에서 1차 파이프라인 레지스터(120(4X+3))(즉, B_{31})로 시프트된다.

[0070] 1차 및 섀도우 탭핑-지연 라인들(78(0), 78(1))에서 32 비트 벡터 데이터 시프트가 요구된다면, 1차 파이프라인 레지스터들(120(0)-120(4X+3)) 및 섀도우 파이프라인 레지스터들(122(0)-122(4X+3))에 저장된 벡터 데이터는 2개의 16 비트 벡터 데이터 시프트 연산들에서 원하는 경우, 시프트될 수 있다.

[0071] 도 7에서, 레지스터들(B_{31}, B_{30})에 대한 1차 파이프라인 레지스터들(120(4X+3), 120(4X+2)) 및 레지스터들(A_{31}, A_{30})에 대한 1차 파이프라인 레지스터들(120(2X+1), 120(2X))은 논리적으로는 시프트된 입력 벡터 데이터 샘플들(86S(X))로 서로 연관되지만, 물리적으로는 도 8에 예시된 바와 같이 서로 인접하지 않는다는 점에 주목한다. 도 6b에 예시된 바와 같이, 벡터 데이터 파일들(82(0)-82(X))에서 입력 벡터 데이터 샘플 세트(86(0)-86(X))의 저장 패턴으로 인해 이 예에서는 이러한 배열이 제공된다. 도 6b에 또한 예시된 바와 같이, 벡터 데이터 파일들(82(0)-82(X))에 저장된 입력 벡터 데이터 샘플 세트(86(0)-86(X))는 어드레스 0과 어드레스 1에 이른다. 그러나 본 명세서의 개시내용은 벡터 데이터 파일들(82(0)-82(X))에서 입력 벡터 데이터 샘플 세트(86(0)-86(X))의 이러한 저장 패턴으로 한정되는 것은 아니라는 점에 주목한다.

[0072] 추가로, 도 8과 관련하여, 탭핑-지연 라인들(78(0), 78(1))은 실행될 벡터 명령에 따라 탭핑-지연 라인들(78(0), 78(1))에 대한 프로그램가능 입력 데이터 경로 구성에 기초하여 벡터 데이터 파일들(82(0)-82(X))과 실행 유닛들(84(0)-84(X)) 간의 입력 데이터 흐름 경로들(80(0)-80(X))에서 선택적으로 제공되거나 제공되지 않도록 구성가능하다. 예컨대, 벡터 명령이 필터 벡터 프로세싱 명령이 아니고 그리고/또는 다르게는 탭핑-지연 라인들(78(0), 78(1))이 입력 벡터 데이터 샘플 세트들(86(0)-86(X))을 시프트할 것을 요구한다면, 탭핑-지연 라인들(78(0), 78(1))은 입력 벡터 데이터 샘플 세트들(86(0)-86(X))을 래치하지 않도록 구성될 수 있다. 1차 및 섀도우 탭핑-지연 라인들(78(0), 78(1))을 바이패스함으로써 벡터 데이터 파일들(82(0)-82(X))로부터 각각의 실행 유닛들(84(0)-84(X))로 입력 벡터 데이터 샘플 세트들(86(0)-86(X))이 제공될 수 있다. 이러한 프로그램가능 데이터 경로 구성은 추가로, 입력 데이터 흐름 경로들(80(0)-80(X))에서 1차 및 섀도우 탭핑-지연 라인들(78(0), 78(1))이 제공되거나 제공되지 않게 한다. 1차 및 섀도우 탭핑-지연 라인들(78(0), 78(1))은 원하는 경우, 각각의 벡터 명령에 대해 입력 데이터 흐름 경로들(80(0)-80(X))에서 제공되거나 제공되지 않도록 프로그램될 수 있다.

[0073] 도 9a는 필터 벡터 프로세싱 명령의 제 1 클록 사이클(CYCLE0) 동안 벡터 데이터 파일들(82(0)-82(X))로부터 1차 탭핑-지연 라인(78(0))으로 로딩되는 입력 벡터 데이터 샘플 세트(86(0)-86(X))를 예시한다. 1차 탭핑-지연 라인(78(0)) 및 섀도우 탭핑-지연 라인(78(1))은 도 7로부터 단순화된 형태로 도시된다. 글로벌 레지스터 파일(40)이 또한 도시된다. 제 1 입력 벡터 데이터 샘플 세트(86(0)-86(X))는 입력 벡터 데이터 샘플들(X0-X63)로서 1차 탭핑-지연 라인(78(0))에 로딩된다. 예컨대, 제 1 입력 벡터 데이터 샘플 세트(86(0)-86(X))를 1차 탭핑-지연 라인(78(0))(그리고 또한 아래 더 상세히 논의되는 바와 같이, 섀도우-탭핑 지연 라인(78(1)))으로 로딩하도록 특수 벡터 명령이 지원될 수도 있다. 이 제 1 입력 벡터 데이터 샘플 세트(86(0)-86(X))는 도 6b에 도시된 벡터 데이터 파일들(82(0)-82(X))의 어드레스들 0 및 1에 저장되었다. 이 예에서, X0, X1, X32 및 X33은 제 1 입력 벡터 데이터 샘플(86(0))을 형성하는데, 이는 단지 이 예의 경우 도 4의 VPE(22(1))에서의 벡터 데이터 파일들(82(0)-82(X))의 저장 패턴 때문이라는 점에 주목한다. 다른 입력 벡터 데이터 샘플들(86)은 도 9a에 도시된 바와 같이 비슷하게 형성된다(예컨대, 86(1), 86(2), … 86(X)). 입력 벡터 데이터 샘플들(86)을 함께 그룹화하여 입력 벡터 데이터 샘플 세트(86(0)-86(X))를 형성하도록 다른 패턴들이 제공될 수 있다.

[0074] 도 9b는 필터 벡터 프로세싱 명령의 제 2 클록 사이클(CYCLE1) 동안 섀도우 탭핑-지연 라인(78(1))으로 로딩되는 다음 입력 벡터 데이터 샘플 세트(86N(0)-86N(X))를 예시한다. 다음 입력 벡터 데이터 샘플 세트(86N(0)-86N(1))는 벡터 데이터 파일들(82(0)-82(X))로부터의 제 1 입력 벡터 데이터 샘플 세트(86(0)-86(X))가 필터 벡터 프로세싱 연산의 실행을 셋업하도록 1차 탭핑-지연 라인(78(0))으로 로딩된 후 섀도우 탭핑-지연 라인

(78(1))으로 로딩된다. 이러한 다음 입력 벡터 데이터 샘플 세트(86N(0)-86N(X))는 입력 벡터 데이터 샘플들(X(64)-X(127))로서 섀도우 탭핑-지연 라인(78(1))으로 로딩된다. 이러한 다음 입력 벡터 데이터 샘플 세트(86N(0)-86N(X))는 도 6b에 도시된 벡터 데이터 파일들(82(0)-82(X))의 어드레스들 2 및 3에 저장되었다. 이 예에서, X64, X65, X96 및 X97은 제 1 입력 벡터 데이터 샘플(86(0))을 형성하는데, 이는 단지 이 예의 경우 도 4의 VPE(22(1))에서의 벡터 데이터 파일들(82(0)-82(X))의 저장 패턴 때문이라는 점에 주목한다. 입력 벡터 데이터 샘플들(86)을 함께 그룹화하여 입력 벡터 데이터 샘플 세트(86(0)-86(X))를 형성하도록 다른 패턴들이 제공될 수 있다. 글로벌 레지스터 파일(40)로부터의 제 1 필터 계수들(92(0))은 또한 필터 벡터 프로세싱 연산(102)에서 사용하기 위해 도 9b의 실행 유닛들(84(0)-84(X))로 레지스터("C")에서 제공되는 것으로 도시된다.

[0075] 도 7을 다시 참조하면, 입력 벡터 데이터 샘플들(86)이 필터 벡터 프로세싱 연산(102)의 각각의 프로세싱 스테이지 동안 1차 탭핑-지연 라인(78(0))에서 시프트될 때, 섀도우 파이프라인 레지스터들(122)에 저장된 다음 입력 벡터 데이터 샘플들(86N)이 또한 섀도우 탭핑-지연 라인(78(1))의 섀도우 파이프라인 레지스터들(122)에서 시프트된다. 도 7에서 제 1 섀도우 파이프라인 레지스터(122(0))에 저장된 입력 벡터 데이터 샘플(86)이 각각의 시프트 동안 1차 탭핑-지연 라인(78(0))의 마지막 1차 파이프라인 레지스터(120(4X+3))로 시프트된다. 따라서 이런 식으로, 필터 벡터 프로세싱 연산(102) 프로세싱 스테이지들이 실행 유닛들(84(0)-84(X))에서 진행할 때, 섀도우 탭핑-지연 라인(78(1))에 초기에 저장된 다음 입력 벡터 데이터 샘플 세트(86N(0)-86N(X))의 적어도 일부가 1차 탭핑-지연 라인(78(0))으로 시프트되어 프로세싱을 위해 실행 유닛들(84(0)-84(X))에 제공된다. 이 예에서 시프트들의 수는 필터 벡터 프로세싱 연산(102)에서 제공되는 필터 탭들의 수에 좌우될 것이다. 벡터 데이터 파일들(82(0)-82(X))로부터의 1차 탭핑-지연 라인(78(0)) 및 섀도우 탭핑-지연 라인(78(1))으로 패치되는 입력 벡터 데이터 샘플 세트(86(0)-86(X))에서 입력 벡터 데이터 샘플들(86)의 수가 필터 벡터 프로세싱 연산(102)에서 필터 탭들의 수보다 더 많다면, 실행 유닛들(84(0)-84(X))은 어떠한 추가 입력 벡터 데이터 샘플 세트들(86(0)-86(X))도 벡터 데이터 파일들(82(0)-82(X))로부터 재패치되지 않고 필터 벡터 프로세싱 연산(102)을 수행할 수 있다. 그러나 필터 벡터 프로세싱 연산(102)에서 필터 탭들의 수가 벡터 데이터 파일들(82(0)-82(X))로부터의 1차 탭핑-지연 라인(78(0)) 및 섀도우 탭핑-지연 라인(78(1))으로 패치되는 입력 벡터 데이터 샘플 세트(86(0)-86(X))에서 입력 벡터 데이터 샘플들(86)의 수보다 더 많다면, 필터 벡터 프로세싱 연산(102)의 일부로서 벡터 데이터 파일들(82(0)-82(X))로부터 추가 입력 벡터 데이터 샘플 세트들(86(0)-86(X))이 패치될 수 있다. 시프트된 입력 벡터 데이터 샘플 세트(86S(0)-86S(X))에 대해 필터 벡터 프로세싱 연산(102)이 완료된 후, 다음에는 탭핑-지연 라인들(78(0), 78(1))에 프로세싱되지 않은 입력 벡터 데이터 샘플(86S)이 존재한다면 다음 상관 벡터 프로세싱 연산을 위한 시프트된 입력 벡터 데이터 샘플 세트(86S(0)-86S(X))로서 1차 탭핑-지연 라인(78(0))에 저장된 이전의 다음 입력 벡터 샘플 데이터 세트(86N(0)-86N(X))가 실행 유닛들(84(0)-84(X))에 제공될 수 있다.

[0076] 섀도우 탭핑-지연 라인(78(1))을 제공하기 위한 다른 예시적인 근거는 다음과 같다. 현재 필터 벡터 프로세싱 연산(102)이 벡터 데이터 레인들(100(0)-100(X))의 폭으로 제공될 수 있는 것보다 더 많은 입력 벡터 데이터 샘플들(86)을 수반한다면, 섀도우 탭핑-지연 라인(78(1))으로 로딩되는 추가 입력 벡터 데이터 샘플 세트(86(0)-86(X))가 필터 벡터 프로세싱 연산(102) 동안 지연 없이 실행 유닛들(84(0)-84(X))에 이용가능할 것이다. 앞서 논의된 바와 같이, 필터 벡터 프로세싱 연산(102)이 실행 동안 시프트된 입력 벡터 데이터 샘플 세트들(86S(0)-86S(X))에 걸쳐 진행할 때, 섀도우 탭핑-지연 라인(78(1))으로 로딩되는 추가적인 다음 입력 벡터 데이터 샘플 세트들(86N(0)-86N(X))이 1차 탭핑-지연 라인(78(0))으로 시프트된다. 따라서 이런 식으로, 실행 유닛들(84(0)-84(X))에 의한 벡터 프로세싱에 사용하기 위한 다음 입력 벡터 데이터 샘플 세트(86N(0)-86N(X))가 지연 없이 이용가능하다. 실행 유닛들(84(0)-84(X))은 벡터 데이터 파일들(82(0)-82(X))의 폭의 단일 패치된 입력 벡터 데이터 샘플 세트(86(0)-86(X))가 전체 필터 벡터 프로세싱 연산(102)을 수행하기에 충분한지 여부에 관계없이 필터 벡터 프로세싱 연산(102) 동안 계속해서 충분히 이용될 수 있다.

[0077] 제 1 입력 벡터 데이터 샘플 세트(86(0)-86(X)) 및 다음 입력 벡터 데이터 샘플 세트(86N(0)-86N(X))가 각각 1차 탭핑-지연 라인(78(0)) 및 섀도우 탭핑-지연 라인(78(1))으로 로딩된 후, 1차 탭핑-지연 라인(78(0))에 제공된 제 1 입력 벡터 데이터 샘플 세트(86(0)-86(X))가 필터 벡터 프로세싱 연산(102)의 제 1 프로세싱 스테이지에서 프로세싱되도록 각각의 실행 유닛들(84(0)-84(X))에 제공된다(도 5의 블록(108)). 제 1 입력 벡터 데이터 샘플 세트(86(0)-86(X))가 실행 유닛들(84(0)-84(X))에 의해 프로세싱된 후 제 1 입력 벡터 데이터 샘플 세트(86(0)-86(X))는 실행 유닛들(84(0)-84(X))에 의해 프로세싱될 시프트된 입력 벡터 데이터 샘플 세트(86S(0)-86S(X))가 되도록 1차 탭핑-지연 라인(78(0))에서 시프트된다. 도 4의 VPE(22(1))에서 예시된 것과 같이, 시프트된 입력 벡터 데이터 샘플(86S(0))은 실행 유닛(84(0))에 제공되고, 시프트된 입력 벡터 데이터 샘플(86S

(1))은 실행 유닛(84(1))에 제공되는 식이다.

[0078] 다음에, 실행 유닛들(84(0)-84(X))이 필터 벡터 프로세싱 연산(102)을 수행한다(도 5의 블록(110)). 보다 구체적으로, 실행 유닛들(84(0)-84(X))은 이 예에서는 연산: y[n] = x[n-7]*h7에 따라 제 1 반복에서 제 1 입력 벡터 데이터 샘플 세트(86(0)-86(X))를 현재 필터 계수(92(0))와 곱하며, 여기서 x[n-7]은 결과적 필터 출력 벡터 데이터 샘플 세트(94(0)-94(X))를 제공할 제 1 입력 벡터 데이터 샘플 세트(86(0)-86(X))이다. 필터 벡터 프로세싱 연산(102)의 다음 반복(도 5의 블록(110))에서, 필터 벡터 프로세싱 연산(102)에 대한 다음 시프트된 입력 벡터 데이터 샘플 세트들(86S(0)-86S(X))이 현재 필터 계수(92(1)-92(Y-1))와 곱해진다. 실행 유닛들(84(0)-84(X))은 새로운 이전의 결과적 필터 출력 벡터 데이터 샘플 세트(94(0)-94(X))를 제공하도록 결과적 필터 벡터 출력 벡터 데이터 샘플 세트(94(0)-94(X))를 실행 유닛들(84(0)-84(X))에 의해 계산된 이전의 결과적 필터 출력 벡터 데이터 샘플 세트(94(0)-94(X))와 누산한다(도 5의 블록(112)). 필터 벡터 프로세싱 연산(102)의 제 1 프로세싱 스테이지에는, 이전의 결과적 필터 출력 벡터 데이터 샘플 세트가 없다.

[0079] 필터 벡터 프로세싱 연산(102)의 모든 프로세싱 스테이지들이 완료되었다면(도 5의 블록(114)), 누산된 이전의 결과적 필터 출력 벡터 데이터 샘플 세트(94(0)-94(X))는 벡터 데이터 파일들(82(0)-82(X))에 제공되어 저장되도록 출력 데이터 흐름 경로들(98(0)-98(X))에서 결과적 필터 출력 벡터 데이터 샘플 세트(94(0)-94(X))로서 제공된다(도 5의 블록(116)). 필터 벡터 프로세싱 연산(102)의 모든 프로세싱 스테이지들이 완료되지 않았다면(도 5의 블록(114)), 탭핑-지연 라인들(78(0), 78(1))에 저장된 샘플들이 필터 벡터 프로세싱 연산(102)에 대한 다음 시트프된 입력 벡터 데이터 샘플 세트(86S(0)-86S(X))를 제공하도록 탭핑-지연 라인들(78(0), 78(1)) 내에서 시프트된다(도 5의 블록(118)). 필터 벡터 프로세싱 연산(102)이 완료할 때까지 이전의 결과적 필터 출력 벡터 데이터 샘플 세트와 누산될 중간 결과로서 다음 결과적 필터 출력 벡터 데이터 샘플 세트를 계산하기 위해 시프트된 입력 벡터 데이터 샘플 세트(86S(0)-86S(X))가 제공된다. 탭핑-지연 라인들(78(0), 78(1))에서 시프트된 입력 벡터 데이터 샘플 세트(86S(0)-86S(X))를 제공하기 위한 입력 벡터 데이터 샘플들(86)의 시프트는 도 7에 관하여 앞서 상세히 미리 설명되었다. 필터 벡터 프로세싱 연산(102)을 위해 실행 유닛들(84(0)-84(X))에 의해 제공되는 중간 결과들의 최종 누산은 도 4에 예시된 바와 같이, 실행 유닛들(84(0)-84(X))로부터의 결과적 필터 출력 벡터 데이터 샘플 세트(94(0)-94(X))로서 제공된다.

[0080] 도 9c는 입력 벡터 데이터 샘플 세트(86(0)-86(X))가 다음 필터 프로세싱 연산(y[n] = x[n-6] * h6)을 위한 다음 시프트된 입력 벡터 데이터 샘플 세트(86S(0)-86S(X))가 되도록 필터 벡터 프로세싱 연산(102)의 제 2 프로세싱 스테이지에서 시프트될 때 탭핑-지연 라인들(78)의 컨텐츠를 예시한다. 1차 탭핑-지연 라인(78(0))에서 시프트된 입력 벡터 데이터 샘플 세트(86S(0)-86S(X))는 실행되는 벡터 명령에 의해 규정된 입력 벡터 데이터 샘플 시프트의 폭에 따라 1차 파이프라인 레지스터들(120(0)-120(4X+3))에서 시프트된다. 예컨대, 도 9c에 예시된 바와 같이, 시프트된 입력 벡터 데이터 샘플(86S(0))에서 샘플(X2)이 시프트된다. 필터 벡터 프로세싱 연산(102)의 다음 필터 탭에 대한 실행을 위해 새로운 시프트된 입력 벡터 데이터 샘플 세트(86S(0)-86S(X))가 실행 유닛들(84(0)-84(X))에 제공된다. 실행 유닛들(84(0)-84(X))에 제공되는 필터 계수(92)는 또한 다음 필터 계수(92)이기도 하며, 이는 이 예에서 "h6"이다.

[0081] 도 5를 계속 참조하면, 다음 필터 계수(92)와 곱해지도록(도 5의 블록(110)) 시프트된 입력 벡터 데이터 샘플 세트(86S(0)-86S(X))를 1차 탭핑-지연 라인(78(0))으로부터 실행 유닛들(84(0)-84(X))로 제공(도 5의 블록(108))함으로써 프로세스가 반복된다. 결과적 필터 벡터 출력 벡터 데이터 샘플 세트(94(0)-94(X))가 이전의 결과적 필터 출력 벡터 데이터 샘플 세트(94(0)-94(X))와 누산된다(도 5의 블록(112)). 도 9d는 예시적인 필터 벡터 프로세싱 연산(102)의 마지막 프로세싱 스테이지 동안 탭핑-지연 라인들(78(0), 78(1))에 존재하는 입력 벡터 데이터 샘플들(86)의 상태를 예시한다. 도 9d에 도시된 바와 같이 이 예에서는, 필터 계수들(92)("h7" - "h0")(즉, 92(0)-92(Y-1)) 때문에 필터 벡터 프로세싱 연산(102)에 8개의 필터 탭들(Y)이 있었다. "h0"은 도 9d에 도시된 바와 같이, 필터 벡터 프로세싱 연산(102)의 마지막 필터 계수(92)이다. 시프트된 입력 벡터 데이터 샘플 세트(86S(0)-86S(X))는, 필터 벡터 프로세싱 연산(102)에 대한 최종 8번째 프로세싱 스테이지에서 입력 벡터 데이터 샘플(X(39))이 1차 탭핑-지연 라인(78(0))의 시프트된 입력 벡터 데이터 샘플(86S(0))에 저장되도록 (필터 탭들의 수보다 1회 더 적은) 7회 시프트된다.

[0082] 앞서 설명한 필터 벡터 프로세싱 연산(102)의 예는 VPE(22(1))의 벡터 데이터 레인들(100(0)-100(X)) 각각을 이용하여 필터 벡터 프로세싱 연산(102)을 제공하지만, 이것이 필수적이진 않다는 점에 주목한다. 필터 벡터 프로세싱 연산(102)은 단지 벡터 데이터 레인들(100(0)-100(X))의 서브세트만이 필터 벡터 프로세싱 연산(102)에 이용될 것을 요구할 수도 있다. 예컨대, 입력 벡터 데이터 샘플 세트(86(0)-86(X))의 폭은 모든 벡터 데이터 파일들(82(0)-82(X))의 폭 미만일 수도 있으며, 여기서는 필터 벡터 프로세싱 연산(102)과 동시에 수행될 다른

벡터 프로세싱 연산들을 위한 추가 벡터 데이터 레인들(100)을 이용하는 것이 바람직하다. 이 시나리오에서, 도 7의 탭핑-지연 라인들(78(0), 78(1))은 최종 벡터 데이터 레인(100(X))에 도달하기 전에 벡터 데이터 레인(100)에서 시프트된 입력 벡터 데이터 샘플 세트(86S(0)-86S(X))로서 다음 입력 벡터 데이터 샘플 세트(86N(0)-86N(X))를 섀도우 탭핑-지연 라인(78(1))에서 1차 탭핑-지연 라인(78(0))으로 시프트하도록 수정될 필요가 있을 수도 있다.

[0083] 도 10은 위의 예에서 예시적인 8개의 탭 필터 벡터 프로세싱 스테이지들이 y[n] = x[n]*h0 +x[n-1]*h1 + ... x[n-7]*h7에 따라 완전히 실행된 후, 도 4의 VPE(22(1))에서 실행 유닛들(84(0)-84(X))의 누산기들의 컨텐츠(즉, 결과적 필터 출력 벡터 데이터 샘플들(94))의 개략도이다 누산기들(Acc0-Acc3)은 도 10에 도시되는데, 이 예에서 각각의 실행 유닛(84(0)-84(X))이 각각의 벡터 데이터 레인(100(0)-100(X))에 대해 병렬로 배치된 4개의 누산기들을 갖기 때문이다. 누산된 결과적 출력 벡터 데이터 샘플들은 추가 분석 및/또는 프로세싱을 위해 그 내부에 저장될 집합적인 결과적 필터 출력 벡터 데이터 샘플 세트(94(0)-94(X))로서 출력 데이터 흐름 경로들(98(0)-98(X)) 상에서 벡터 데이터 파일들(82(0)-82(X))로 제공될 수 있다. 특수 벡터 명령은 원하는 경우, 벡터 데이터 파일들(82(0)-82(X))로부터 도 2의 벡터 유닛 데이터 메모리(32)로 결과적 필터 출력 벡터 데이터 샘플 세트(94(0)-94(X))의 행들을 이동시키도록 VPE(22(1))에 의해 지원될 수 있다.

[0084] 필터 벡터 프로세싱 연산(102) 이외의 다른 타입들의 벡터 프로세싱 연산들이 또한 위에서 논의된 도 4의 VPE(22(1))에서 제공된 것과 같은 또는 비슷한 탭핑-지연 라인들(78)의 사용에 의해 VPE에서 프로세싱 효율들을 누릴 수 있다. 예컨대, VPE에서 입력 벡터 데이터 샘플 세트들(86)의 시프트를 수반하는 다른 특수 벡터 프로세싱 연산은 (본 명세서에서는 "상관 벡터 프로세싱 연산"으로 지칭되는) 상관/공분산 벡터 프로세싱 연산이다. 일례로, CDMA 시스템에서 사용자 신호와 다른 사용자들의 신호들 간의 양호한 분리를 제공하기 위해 CDMA 시스템에서 사용자 신호를 복조하기 위한 DSSC(direct spread-spectrum code)(즉, 칩 시퀀스)를 선택하기 위한 상관 연산들을 제공하기 위해 벡터 프로세싱을 이용하는 것이 바람직할 수도 있다. 신호들의 분리는 원하는 사용자의 국소적으로 생성된 칩 시퀀스와 수신된 신호를 상관시킴으로써 이루어진다. 신호가 원하는 사용자의 칩 시퀀스와 매칭한다면, 상관 함수는 높을 것이고 CDMA 시스템은 그 신호를 추출할 수 있다. 원하는 사용자의 칩 시퀀스가 그 신호와 거의 또는 전혀 공통점이 없다면, 상관은 가능한 한 0에 가까워야 하는데(따라서 신호를 제거해야 하는데), 이는 상호-상관으로 지칭된다. 칩 시퀀스가 0이 아닌 임의의 시간 오프셋으로 신호와 상관된다면, 상관은 가능한 0에 가까워야 한다. 이는 자기-상관으로 지칭되며, 다중-경로 간섭을 거부하는 데 사용된다.

[0085] 그러나, 상관 연산들은 벡터 프로세서들에서 제공되는 특수화된 데이터 흐름 경로들로 인해 벡터 프로세서들에서 병렬화하기에 어려울 수 있다. 상관될 신호를 나타내는 입력 벡터 데이터 샘플 세트가 지연 탭들 사이에서 시프트될 때, 입력 벡터 데이터 샘플 세트는 벡터 데이터 파일로부터 재패치되고, 따라서 전력 소비를 증가시키고 스루풋을 감소시킨다. 메모리로부터 입력 벡터 데이터 샘플 세트의 재패치를 최소화하기 위해, 데이터 흐름 경로는 효율적인 병렬화된 프로세싱을 위해 지연 탭들과 동일한 수의 곱셈기들을 제공하도록 구성될 수 있다. 그러나, 다른 벡터 프로세싱 연산들은 더 적은 곱셈기들을 요구할 수 있고, 이로써 데이터 흐름 경로에서 곱셈기들의 비효율적인 스케일링 및 낮은 활용을 제공한다. 곱셈기들의 수가 확장성을 제공하기 위한 지연 탭들의 수 미만으로 감소되면, 병렬화는 상관 프로세싱의 상이한 단계들에 대한 동일한 입력 벡터 데이터 샘플 세트를 획득하기 위해 메모리에 요구되는 더 많은 재패치들에 의해 제한된다.

[0086] 이와 관련하여, 도 11은 도 2의 VPE(22)로서 제공될 수 있는 다른 예시적인 VPE(22(2))의 개략도이다. 아래에 더 상세히 설명될 바와 같이, 도 11의 VPE(22(2))는 벡터 데이터 샘플 재패치가 제거 또는 감소되고 전력 소비가 감소된, VPE(22(2)) 내의 정밀 상관 벡터 프로세싱 연산들을 제공하도록 구성된다. 벡터 데이터 샘플 재패치를 요구하는 중간 결과들의 저장을 요구하고 이로써 결과적으로 전력 소비를 증가시키는 상관 벡터 프로세싱 연산들과 비교하여 VPE(22(2))에서 정밀 상관 벡터 프로세싱 연산들이 제공될 수 있다. 전력 소비를 감소시키고 프로세싱 효율을 개선하기 위해 벡터 데이터 파일로부터 입력 벡터 데이터 샘플들의 재패치를 제거 또는 최소화하기 위해, 도 4의 VPE(22(1)에 포함된 탭핑-지연 라인들(78)이 또한 VPE(22(2)) 내의 실행 유닛들(84(0)-84(X))("EU"로 또한 라벨링됨)과 벡터 데이터 파일들(82(0)-82(X)) 사이의 입력 데이터 흐름 경로들(80(0)-80(X))에 포함된다. 'X'+1은 이러한 예에서 벡터 데이터 샘플들의 프로세싱을 위해 VPE(22(2))에 제공되는 병렬 입력 데이터 레인들의 최대 수이다. 위에서 이전에 논의된 바와 같이, 탭핑-지연 라인들(78)은 입력 벡터 데이터 샘플 세트(86(0)-86(X))의 입력 벡터 데이터 샘플들(86)의 서브세트 또는 전부로서 탭핑-지연 라인 입력들(88(0)-88(X)) 상에서 입력 벡터 데이터 샘플 세트(86(0)-86(X))를 벡터 데이터 파일들(82(0)-82(X))의 대응하는 서브세트 또는 전부로부터 수신하도록 구성된다. 입력 벡터 데이터 샘플들(86) 모두는 입력 벡터 데이터

샘플 세트(86(0)-86(X))를 포함한다. 아래에 더 상세히 논의될 바와 같이, 벡터 데이터 파일들(82(0)-82(X))로부터의 입력 벡터 데이터 샘플 세트(86(0)-86(X))는 결과적 상관된 출력 벡터 데이터 샘플 세트(132(0)-132(X))를 제공하기 위해 VPE(22(2))에서 기준 벡터 데이터 샘플 세트(130(0)-130(X))와 상관된다. 기준 벡터 데이터 샘플 세트(130(0)-130(X))는, 이러한 예에서 130(0), 130(1),..., 및 130(X)인 'X+1' 기준 벡터 데이터 샘플들(130)에 포함된다. 결과적 상관된 출력 벡터 데이터 샘플 세트(132(0)-132(X))는 이러한 예에서 132(0), 132(1),..., 및 132(X)인 'X+1' 결과적 상관된 출력 벡터 데이터 샘플(132)에 포함된다.

[0087] 도 11을 계속해서 참조하면, 탭핑-지연 라인들(78)은 시프트된 입력 벡터 데이터 샘플 세트(86S(0)-86S(X)를 제공하기 위해 VPE(22(2))에 의해 실행될 상관 벡터 명령에 따라 상관 벡터 프로세싱 연산의 각각의 상관 지연 탭(즉, 상관 프로세싱 스테이지)에 대한 입력 벡터 데이터 샘플 세트(86(0)-86(X))를 시프트한다. 시프트된 입력 벡터 데이터 샘플들(86S) 전부는 시프트된 입력 벡터 데이터 샘플 세트(86S(0)-86S(X))를 포함한다. 탭핑-지연 라인들(78)은 상관 벡터 프로세싱 동작 동안에 시프트된 입력 벡터 데이터 샘플 세트(86S(0)-86S(X))를 실행 유닛들(84(0)-84(X))의 실행 유닛 입력들(90(0)-90(X))에 제공하기 위해 입력 벡터 데이터 샘플 세트(86(0)-86(X))를 시프트한다. 이러한 방식으로, 시프트된 입력 벡터 데이터 샘플 세트(86S(0)-86S(X))에 대해 수행되는 연산들에 기초한 중간 상관 결과들은 VPE(22(2))에 의해 수행되는 상관 벡터 프로세싱 연산의 각각의 프로세싱 스테이지 동안에 저장, 시프트되고, 벡터 데이터 파일들(82(0)-82(X))로부터 재패치되지 않아도 된다. 따라서, 탭핑-지연 라인들(78)은 전력 소비를 감소시키고, VPE(22(2))에 의해 수행되는 상관 벡터 프로세싱 연산들에 대한 프로세싱 효율을 증가시킬 수 있다.

[0088] 도 11을 계속 참조하면, 실행 유닛들(84(0)-84(X))은 또한 상관 벡터 프로세싱 동작을 위해 SNG(sequence number generator)(134)에 저장된 기준 벡터 데이터 샘플 세트(130(0)-130(X)) 사이로부터 기준 벡터 데이터 샘플(130)을 수신한다. 실행 유닛들(84(0)-84(X))은 상관 벡터 프로세싱 동작의 부분으로서 입력 벡터 데이터 샘플 세트(86(0)-86(X))와 기준 벡터 데이터 샘플 세트(130(0)-130(X))를 상관시키도록 구성된다. 그러나, 시퀀스 번호 생성기(134)는 또한 레지스터 또는 다른 파일일 수 있다는 것을 주목하라. 기준 벡터 데이터 샘플 세트(130(0)-130(X))를 제공하기 위해 시퀀스 번호 생성기(134)가 이러한 실시예에 제공되는데, 왜냐하면 상관 벡터 프로세싱 동작이 이러한 예에서 CDMA 상관 벡터 명령에 대한 것이기 때문이다. 기준 벡터 데이터 샘플 세트(130(0)-130(X))는, 기준 벡터 데이터 샘플 세트(130(0)-130(X))와 입력 벡터 데이터 샘플 세트(86(0)-86(X)) 사이의 상관이 높은 경우에, 입력 벡터 데이터 샘플 세트(86(0)-86(X))로부터의 신호 추출에서 사용하기 위한 생성된 칩 시퀀스로서 제공된다.

[0089] 예컨대, CDMA 벡터 상관 명령에 대한 상관 벡터 프로세싱 동작은 입력 벡터 데이터 샘플 세트(86(0)-86(X)) 내의 정시 입력 벡터 데이터 샘플들(86)과 입력 벡터 데이터 샘플 세트(86(0)-86(X)) 내의 늦은 입력 벡터 데이터 샘플들 사이의 상관을 제공할 수 있다. 예컨대, 입력 벡터 데이터 샘플 세트(86(0)-86(X)) 내의 정시 입력 벡터 데이터 샘플들(86)은 입력 벡터 데이터 샘플 세트(86(0)-86(X)) 내의 짝수 입력 벡터 데이터 샘플들(86)(예컨대, 86(0), 86(2), 86(4),...86(X-1))일 수 있다. 입력 벡터 데이터 샘플 세트(86(0)-86(X)) 내의 늦은 입력 벡터 데이터 샘플들(86)은 입력 벡터 데이터 샘플 세트(86(0)-86(X)) 내의 홀수 입력 벡터 데이터 샘플들(86)(예컨대, 86(1), 86(3), 86(5),...86(X))일 수 있다. 대안적으로, 정시 입력 벡터 데이터 샘플들(86)은 홀수 입력 벡터 데이터 샘플들(86)일 수 있고, 늦은 입력 벡터 데이터 샘플들(86)은 짝수 입력 벡터 데이터 샘플들(86)일 수 있다. 상관 벡터 프로세싱 동작의 결과들, 정시 입력 벡터 데이터 샘플들(86) 및 늦은 입력 벡터 데이터 샘플들(86)에 대한 결과적 상관된 출력 벡터 데이터 샘플 세트(132(0)-132(X))는 신호 추출을 위해 입력 벡터 데이터 샘플 세트(86(0)-86(X))로부터 정시 또는 늦은 입력 벡터 데이터 샘플들을 사용할지를 결정하는데 사용될 수 있다. 예컨대, 정시 상관 벡터 프로세싱 연산은 다음에 따라 제공될 수 있다.

[0090] $$R_{xy}^{OT}[n] = \sum_{l=0}^{l=511} y[2l]^* x[2l+n]$$, 여기서

[0091] n은 입력 신호 샘플들의 수이고,

[0092] x[n]은 디지털화된 입력 신호(66)이고,

[0093] y[n]은 기준 신호이고,

[0094] l은 샘플 수이다.

[0095] 늦은 상관 벡터 프로세싱 동작은 다음에 따라 제공될 수 있다.

$$R_{xy}^{LT}[n] = \sum_{l=0}^{l=511} y[2l+1]^* x[2l+1+n]$$

[0096]

[0097] 여기서, n은 입력 신호 샘플들의 수이고,

[0098] x[n]은 디지털화된 입력 신호(66)이고,

[0099] y[n]은 기준 신호이고,

[0100] l은 샘플 수이다.

[0101] 기준 신호 y[n](즉, 기준 벡터 데이터 샘플들)은 복소수일 수 있다. 일 양상에서, VPE(22(2))는 (예컨대, 시퀀스 번호 생성기(134)로부터) 기준 신호를 수신할 수 있다. VPE(22(2))는 정시 및 늦은 상관 연산들을 수행하기 위해 수신된 기준 신호를 직접적으로 사용할 수 있고, 이러한 경우에 위의 수학식들에서 기준 신호 y[n]은 수신된 기준 신호를 나타낼 수 있다. 대안적으로, VPE(22(2))는 정시 및 늦은 상관 연산들을 수행하기 위해 기준 신호를 사용하기 전에 수신된 기준 신호의 복소 공액을 계산할 수 있고, 이러한 경우에 위의 수학식들 내의 기준 신호 y[n]는 수신된 기준 신호의 공액들을 나타낼 수 있다.

[0102] 도 11을 계속 참조하면, 실행 유닛들(84(0)-84(X)) 각각은, 실행 유닛들(84(0)-84(X))에서 중간 상관 출력 벡터 데이터 샘플들을 제공하기 위해 상관 벡터 프로세싱 연산의 각각의 프로세싱 스테이지 동안에 기준 벡터 데이터 샘플 세트(130(0)-130(X))와 시프트된 입력 벡터 데이터 샘플 세트(86S(0)-86S(X))의 시프트된 입력 벡터 데이터 샘플들(86S(0), 86S(1),...86S(X))을 곱셈하도록 구성된다. 중간 상관 출력 벡터 데이터 샘플 세트들은 실행 유닛들(84(0)-84(X)) 각각에서 누산된다(즉, 이전의 누산된 상관 출력 벡터 데이터 샘플이 현재 상관 출력 벡터 데이터 샘플에 부가됨). 이것은, 실행 유닛들(84(0)-84(X))에 의해 생성된 중간 상관 벡터 데이터 출력 샘플 세트들을 저장 및 시프트하지 않고서, VPE(22(2))에 의한 추가의 사용 및/또는 프로세싱을 위해 각각의 벡터 데이터 파일들(82(0)-82(X))에 다시 저장될 각각의 입력 벡터 데이터 샘플 세트(86(0), 86(1),...86(X))에 대해, 출력 데이터 흐름 경로들(98(0)-98(X)) 상의 실행 유닛 출력들(96(0)-96(X)) 상에서 실행 유닛들(84(0)-84(X))에 의해 각각 제공되는 마지막, 결과적 상관된 출력 벡터 데이터 샘플 세트(132(0)-132(X))를 제공한다.

[0103] 또한, 도 11의 VPE(22(2))에 제공된 동일한 컴포넌트들 및 아키텍처가 도 4의 VPE(22(1))에 제공된다는 것을 주목하라. 시퀀스 번호 생성기(134)가 부가되어, 기준 벡터 데이터 샘플 세트(130(0)-130(X))와 프로세싱될 필터 계수들(92(0)-92(Y-1)) 또는 다른 데이터를 제공할 수 있는 글로벌 레지스터 파일(40)과 멀티플렉서(136)에 의해 멀티플렉싱된다. 따라서, 도 11의 VPE(22(2))는 멀티플렉서(136)의 제어에 의해 여기서 그리고 아래에 더 상세히 논의되는 상술된 필터 벡터 프로세싱 연산들 및 상관 벡터 프로세싱 연산들 둘 모두를 제공할 수 있다. 멀티플렉서(136)는 VPE(22(2))에 의해 실행될 벡터 명령에 기초하여 제어되는 선택기 신호(138)에 의해 제어될 수 있다. 필터 벡터 명령에 대해, 선택기 신호(138)는 실행 유닛들(84(0)-84(X))에 제공될 글로벌 레지스터 파일(40)로부터 필터 계수들(92(0)-92(Y-1))을 제공하도록 구성될 수 있다. 상관 벡터 명령에 대해, 선택기 신호(138)는 실행 유닛들(84(0)-84(X))에 제공될 시퀀스 번호 생성기(134)로부터 기준 벡터 데이터 샘플 세트(130(0)-130(X))를 선택하도록 구성될 수 있다.

[0104] 도 11을 계속 참조하여 그리고 아래에 더 상세히 논의될 바와 같이, 탭핑-지연 라인들(78(0), 78(1))은 프로세싱되는 벡터 명령에 따라 제어되도록 프로그램 가능하다. 탭핑-지연 라인들(78)을 사용하지 않는 상관 벡터 명령 또는 다른 명령이 프로세싱되지 않는다면, 탭핑-지연 라인들(78)은 벡터 데이터 파일들(82(0)-82(X))과 실행 유닛들(84(0)-84(X)) 사이의 입력 데이터 흐름 경로들(80(0)-80(X))에 포함되지 않도록 프로그램될 수 있다. 이러한 실시예에서, 이전에 논의된 바와 같이, 2 개의 탭핑-지연 라인들(78), 1차 탭핑-지연 라인들(78(0)) 및 섀도우 탭핑-지연 라인들(78(1))이 제공되고, 섀도우 탭핑-지연 라인들(78(1))은 이러한 실시예에서 선택적이다. 이전에 논의된 바와 같이, 탭핑-지연 라인들(78) 없이, 시프트된 중간 입력 벡터 데이터 샘플 세트를 다시 실행 유닛들(84(0)-84(X))에 제공하기 위해 별개의 시프팅 프로세스가 수행되어야 할 것이고, 이로써 레이턴시를 증가시키고, 부가적인 전력을 소비한다. 또한, VPE(22(2)) 내의 입력 및 출력 데이터 흐름 경로들(80(0)-80(X), 98(0)-98(X))의 효율은 상관 벡터 프로세싱 연산 동안에 벡터 데이터 파일들(82(0)-82(X))로부터의 시프트된 입력 벡터 데이터 샘플 세트(86S(0)-86S(X))의 재패치 지연에 의해 제한되지 않는다. 시프트된 입력 벡터 데이터 샘플 세트(86S(0)-86S(X))는 실행 유닛들(84(0)-84(X))에 로컬화된 탭핑-지연 라인들(78)에

의해 제공된다. 실행 유닛들(84(0)-84(X)) 내의 벡터 프로세싱은 데이터 흐름 제한들보다는 계산적인 자원들에 의해서만 제한된다.

[0105] 또한, 도 11의 VPE(22(2))에 의해 수행되는 상관 벡터 프로세싱 연산들은 탭핑-지연 라인들(78)을 사용함으로써 더 정밀하게 이루어질 수 있는데, 왜냐하면 실행 유닛들(84(0)-84(X))에서 중간 상관 프로세싱 스테이지들에 대한 출력 누산들이 벡터 데이터 파일들(82(0)-82(X))에 저장될 필요가 없기 때문이다. 실행 유닛들(84(0)-84(X))로부터의 중간 벡터 데이터 샘플 세트들의 벡터 데이터 파일들(82(0)-82(X)) 내의 저장은 라운딩을 발생시킬 수 있다. 따라서, 다음 중간 출력 벡터 데이터 샘플 세트가 벡터 프로세싱 연산을 위해 실행 유닛들(84(0)-84(X))에 제공될 때, 임의의 라운딩 에러가 벡터 프로세싱 연산의 각각의 곱셈 단계 동안에 전파 및 부가될 것이다. 이와 반대로, 도 11의 VPE(22(2))의 예에서, 실행 유닛들(84(0)-84(X))에 의해 계산된 중간 상관 출력 벡터 데이터 샘플 세트들은 벡터 데이터 파일들(82(0)-82(X))에 저장될 필요가 없다. 이전의 중간 상관 출력 벡터 데이터 샘플 세트들은 다음 상관 출력 벡터 데이터 샘플 세트들에 대해 중간 상관 출력 벡터 데이터 샘플 세트들과 누산될 수 있는데, 왜냐하면 탭핑-지연 라인들(78)이 프로세싱될 벡터 프로세싱 연산 동안에 시프트된 입력 벡터 데이터 샘플 세트들(86S(0)-86S(X))을 실행 유닛들(84(0)-84(X))에 제공하기 때문이고, 결과들은 이전의 상관 출력 벡터 데이터 샘플 세트들에 대해 이전의 벡터 데이터 샘플 세트들과 누산된다.

[0106] 위의 도 4의 VPE(22(1))에 제공된 컴포넌트들의 이전 논의는 도 11의 VPE(22(2))에 대해 동일하게 적용 가능하고, 따라서 다시 설명되지 않을 것이다.

[0107] 이러한 실시예에서 입력 데이터 흐름 경로들(80(0)-80(X))에서 시프트된 입력 벡터 데이터 샘플 세트(86S(0)-86S(X))를 실행 유닛들(84(0)-84(X))에 제공하기 위한 탭핑-지연 라인들(78) 및 도 11의 VPE(22(2))의 부가적인 세부사항들 및 특징들의 추가의 설명이 이제 설명될 것이다. 이와 관련하여, 도 12a 및 12b는 예시적인 상관 벡터 명령에 따라 탭핑-지연 라인들(78)을 사용하는 도 11의 VPE(22(2))에서 수행될 수 있는 예시적인 상관 벡터 프로세싱 동작(140)을 예시한 흐름도들이다. 도 12a 및 12b는 예시적인 상관/공분산 벡터 프로세싱 연산에 따라 패치된 인터리브된 정시 및 늦은 입력 벡터 데이터 샘플 세트들에 대해 도 11의 VPE(22(2))에서 동시에 수행될 수 있는 예시적인 상관/공분산 벡터 프로세싱 동작들을 예시한 흐름도들이다.

[0108] 도 12a 및 12b에서 상관 벡터 프로세싱 연산(140)에서 수행되는 예시적인 작업들은 도 13-17b에 제공된 예들을 참조하여 설명될 것이다. 도 12a를 참조하면, 상관 벡터 명령에 따라 상관 벡터 프로세싱 연산(140)에서 프로세싱될 입력 벡터 데이터 샘플 세트(86(0)-86(X))는 상관 벡터 프로세싱 연산(140)을 위해 벡터 데이터 파일들(82(0)-82(X))로부터 입력 데이터 흐름 경로들(80(0)-80(X))로 패치된다(블록 142). 도 11의 VPE(22(2))에 관련하여 앞서 논의된 바와 같이, 입력 벡터 데이터 샘플 세트(86(0)-86(X))는 실행 유닛들(84(0)-84(X)) 내의 시퀀스 번호 생성기(134)로부터 수신된 기준 벡터 데이터 샘플 세트(130(0)-130(X))에 의해 곱셈된다. 예컨대, 도 13은 시퀀스 번호 생성기(134) 내의 기준 벡터 데이터 샘플 세트(130(0)-130(X))를 예시한다. 이러한 예에서, 입력 벡터 데이터 샘플 세트(86(0)-86(X)) 내의 열여섯개(16)의 입력 벡터 데이터 샘플들(86(0), 86(1),...86(15))과 상관될 글로벌 레지스터 파일(40)에 저장된 열 여섯(16) 개의 기준 벡터 데이터 샘플들(130(0), 130(1),...130(15))이 존재한다. 위에서 이전에 논의된 도 6b는 벡터 데이터 파일들(82(0)-82(X))에 저장된 예시적인 입력 벡터 데이터 샘플 세트(86(0)-86(X))를 예시하였고, 이것은 또한 이러한 예에서 적용 가능하고, 따라서 여기서 다시 설명되지 않을 것이다.

[0109] 도 11의 VPE(22(2)) 내의 벡터 데이터 레인들(100(0)-100(X)) 중 하나, 일부 또는 전부는 상관 벡터 프로세싱 연산(140)에서 상관될 기준 벡터 데이터 샘플 세트(130(0)-130(X)) 및 입력 벡터 데이터 샘플 세트(86(0)-86(X))의 폭에 의존하는 벡터 명령의 프로그래밍에 따라 상관 벡터 프로세싱 연산(140)을 제공하도록 사용될 수 있다. 벡터 데이터 파일들(82(0)-82(X))의 전체 폭이 요구되면, 모든 벡터 데이터 레인들(100(0)-100(X))은 상관 벡터 프로세싱 연산(140)을 위해 사용될 수 있다. 상관 벡터 프로세싱 연산(140)이 상관 벡터 프로세싱 연산(140)을 위해 사용될 수 있는 벡터 데이터 레인들(100(0)-100(X))의 서브세트만을 요구할 수 있다는 것을 주목하라. 이것은, 입력 벡터 데이터 샘플 세트(86(0)-86(X))의 폭이 모든 벡터 데이터 파일들(82(0)-82(X))의 폭 미만이기 때문일 수 있고, 여기서 다른 벡터 프로세싱 연산들이 상관 벡터 프로세싱 연산(140)과 동시에 수행되도록 부가적인 벡터 데이터 레인들(100)을 사용하는 것이 바람직하다. 현재 예를 논의할 목적으로, 상관 벡터 프로세싱 연산(140)에서 사용되는 입력 벡터 데이터 샘플 세트(86(0)-86(X)) 및 기준 벡터 데이터 샘플 세트(130(0)-130(X))가 VPE(22(2))에서 모든 벡터 데이터 레인들(100(0)-100(X))을 수반한다고 가정된다.

[0110] 도 12a를 다시 참조하면, 패치된 입력 벡터 데이터 샘플 세트(86(0)-86(X))는 상관 벡터 프로세싱 연산(140)을 위한 제 1 입력 벡터 데이터 샘플 세트(86S(0)-86(X))로서 탭핑 지연 라인들(78)로 로딩되도록 벡터 데이터 파

일들(82(0)-82(X))로부터의 입력 데이터 흐름 경로들(80(0)-80(X))에 제공된다(블록 144). 입력 벡터 데이터 샘플 세트(86(0)-86(X))는 상관 벡터 프로세싱 연산(140)을 위해 실행 유닛들(84(0)-84(X))에 의해 프로세싱될 입력 벡터 데이터 샘플 세트(86(0)-86(X))로서 1차 탭핑 지연 라인(78(0))으로 로딩된다. 1차 탭핑 지연 라인(78(0))으로 로딩되는 입력 벡터 데이터 샘플 세트(86(0)-86(X))는 상관 벡터 프로세싱 연산(140)의 제 1 연산을 위해 시프트되지 않는다. 다음 입력 벡터 데이터 샘플 세트(86N(0)-86N(X))는 또한 실행 유닛들(84(0)-84(X))에 의해 프로세싱될 다음 입력 벡터 데이터 샘플 세트(86N(0)-86N(X))로서 섀도우 탭핑 지연 라인(78(1))으로 로딩될 수 있다. 위에서 이전에 논의되고 아래에 더 상세히 논의되는 바와 같이, 탭핑 지연 라인들(78)의 목적은 상관 벡터 프로세싱 연산(140)의 연산 동안에 후속 상관 연산들을 위해 시프트된 입력 벡터 데이터 샘플 세트(86S(0)-86S(X))를 실행 유닛들(84(0)-84(X))에 제공하기 위해 입력 벡터 데이터 샘플 세트(86(0)-86(X))의 시프팅을 제공하는 것이다. 실행 유닛들(84(0)-84(X))에 의해 실행되는 상관 벡터 프로세싱 연산(140)의 각각의 프로세싱 스테이지 동안에, 입력 벡터 데이터 샘플들(86)은 시프트된 입력 벡터 데이터 샘플 세트(86S(0)-86S(X))를 실행 유닛들(84(0)-84(X))에 제공하기 위해 1차 탭핑 지연 라인(78(0))에서 시프트된다. 이러한 방식으로, 입력 벡터 데이터 샘플 세트(86(0)-86(X))는 벡터 데이터 파일들(82(0)-82(X))에서 저장, 시프트되고 상관 벡터 프로세싱 연산(140)의 각각의 상관 연산을 위해 재패치되지 않아도 된다.

[0111] 이와 관련하여, 도 14는 도 11의 VPE(22(2))에 제공될 수 있는 예시적인 탭핑 지연 라인들(78)을 예시한다. 이러한 실시예에서, 탭핑 지연 라인들(78)은 섀도우 탭핑 지연 라인(78(1)) 및 1차 탭핑 지연 라인(78(0))을 포함한다. 위에서 이전에 논의된 바와 같이, 1차 탭핑 지연 라인(78(0))은 이러한 예에서 입력 벡터 데이터 샘플들(86)의 분해능을 8 비트 길이로 낮추도록 허용하기 위해 복수의 8 비트 1차 파이프라인 레지스터들(120)을 포함한다. 실행 유닛들(84(0)-84(X))에 의해 프로세싱되는 제 1 입력 벡터 데이터 샘플 세트(86(0)-86(X))는 이러한 예에서 상관 벡터 프로세싱 연산(140)의 제 1 상관 연산을 위해 시프트되지 않을 것이다. 실행 유닛들(84(0)-84(X))이 상관 벡터 프로세싱 연산(140)을 위해 후속 상관 연산들을 프로세싱할 때, 1차 탭핑 지연 라인(78(0))에 저장된 입력 벡터 데이터 샘플 세트(86(0)-86(X)) 내의 입력 벡터 데이터 샘플들(86)은, 도 14에서 화살표들로 표시된 바와 같이, 시프트된 입력 벡터 데이터 샘플 세트(86S(0)-86S(X))가 되도록 1차 파이프라인 레지스터들(120(0)-120(4X+3))에서 시프트된다. 이러한 방식으로, 실행 유닛들(84(0)-84(X))은, 벡터 데이터 파일들(82(0)-82(X))로부터 입력 벡터 데이터 샘플 세트(86(0)-86(X))를 저장, 시프트하고, 그리고 재패치하지 않고서도, 시프트된 입력 벡터 데이터 샘플 세트(86S(0)-86S(X))의 상관 벡터 프로세싱 연산(140)을 수신 및 수행함으로써 완전히 사용된다.

[0112] 상관 벡터 프로세싱 연산(140)에 대해 1차 및 섀도우 탭핑-지연 라인들(78(0), 78(1))에서 수행되는 시프트들의 수는 상관될 샘플들의 수에 의존할 것이다. 벡터 데이터 파일들(82(0)-82(X))로부터 1차 탭핑-지연 라인(78(0)) 및 섀도우 탭핑-지연 라인(78(1))으로 패치된 입력 벡터 데이터 샘플 세트(86(0)-86(X))의 입력 벡터 데이터 샘플들(86)의 수가 상관 벡터 프로세싱 연산(140)에서 상관 연산들의 수보다 더 큰 경우, 실행 유닛들(84(0)-84(X))은, 어떠한 추가의 입력 벡터 데이터 샘플 세트들(86(0)-86(X))도 벡터 데이터 파일들(82(0)-82(X))로부터 재패치됨 없이 상관 벡터 프로세싱 연산(140)을 수행할 수 있다. 그러나 상관 벡터 프로세싱 연산(140)에서 상관 연산들의 수가 벡터 데이터 파일들(82(0)-82(X))로부터 1차 탭핑-지연 라인(78(0)) 및 섀도우 탭핑-지연 라인(78(1))으로 패치된 입력 벡터 데이터 샘플 세트(86(0)-86(X))의 입력 벡터 데이터 샘플들(86)의 수보다 큰 경우, 부가적인 입력 벡터 데이터 샘플 세트들(86(0)-86(X))은 상관 벡터 프로세싱 연산(140)의 부분으로서 벡터 데이터 파일들(82(0)-82(X))로부터 패치될 수 있다.

[0113] 이 실시예에서, 1차 파이프라인 레지스터들(120(0)-120(4X+3))은 집합적으로 벡터 데이터 파일들(82(0)-82(X))의 폭이다. 벡터 데이터 파일들(82(0)-82(X))이 폭 측면("X"가 십오(15)임)에서 512 비트들인 예에서, 육십사개(64)의 총 1차 파이프라인 레지스터들(120(0)-120(63))이 있을 것이고, 각각의 팔(8) 비트들 폭은 총 512 비트들의 폭을 제공하기 위한 것이다(즉, 64 레지스터들 x 8 비트들 각각). 따라서, 이 예에서, 1차 탭핑-지연 라인(78(0))은 하나(1)의 입력 벡터 데이터 샘플 세트(86(0)-86(X))의 전체 폭을 저장할 수 있다. 이 예에서 팔(8) 비트 폭들의 1차 파이프라인 레지스터들(120(0)-120(4X+3))을 제공함으로써, 입력 벡터 데이터 샘플 세트(86(0)-86(X))는 8 비트 상관 벡터 프로세싱 연산들에 대해 팔(8) 비트들의 벡터 데이터 샘플 크기로 아래로 시프트될 수 있다. 예를 들어, 16 비트 또는 32 비트 샘플들과 같이 더 큰 입력 벡터 데이터 샘플(86) 크기들이 상관 벡터 프로세싱 연산(140)에 대해 바람직한 경우, 입력 벡터 데이터 샘플 세트(86(0)-86(X))는 한 번에 두개(2)의 1차 파이프라인 레지스터들(120)에 의해 1차 파이프라인 레지스터들(120(0)-120(4X+3))에서 시프트될 수 있다.

[0114] 도 15a는 상관 벡터 프로세싱 명령(140)의 제 1 클록 사이클(CYCLE0) 동안 벡터 데이터 파일들(82(0)-82(X))로

부터 1차 탭핑-지연 라인(78(0))으로 로딩되는 입력 벡터 데이터 샘플 세트(86(0)-86(X))를 예시한다. 제 1 입력 벡터 데이터 샘플 세트(86(0)-86(X))는 입력 벡터 데이터 샘플들(X1-X32)로서 1차 탭핑-지연 라인(78(0))에 로딩되지만, 육십사(64) 입력 벡터 데이터 샘플들이 제공된다. 1차 파이프라인 레지스터들(120(0)-120(2X+1))(도 14를 또한 참조)에는 입력 벡터 데이터 샘플 세트(86(0)-86(X))로부터 정시 및 늦은 입력 벡터 데이터 샘플들(86)이 로딩된다. 예를 들어, 특수 벡터 명령은 입력 벡터 데이터 샘플 세트(86(0)-86(X))의 정시 및 늦은 입력 벡터 데이터 샘플들을 1차 탭핑-지연 라인(78(0))(그리고 아래에서 보다 상세히 논의되는 바와 같이, 또한 섀도우-탭핑 지연 라인(78(1)))으로 로딩하도록 지원될 수 있다. 예를 들어, 1차 파이프라인 레지스터들(122(0), 122(1), 122(2X+2), 및 122(2X+3))은 집합적으로 입력 벡터 데이터 샘플(86(0))을 포함한다. 1차 파이프라인 레지스터들(122(0), 122(1))은 X(0) 및 X(1)인 정시 입력 벡터 데이터 샘플(86OT(0))을 포함하고, 여기서 "OT"는 "정시"를 의미한다. 1차 파이프라인 레지스터들(122(2X+2), 122(2X+3))은 X(1) 및 X(2)인 늦은 입력 벡터 데이터 샘플들(86L(0))을 포함하며, 여기서 "L"은 "늦음"을 의미한다. 1차 탭핑-지연 라인(78(0))에서 입력 벡터 데이터 샘플(86) 저장 패턴은 다른 1차 파이프라인 레지스터들(122(2)-122(2X+1) 및 122(2X+4)-122(4X+3))에 대해 반복된다(도 14 참조).

[0115] 도 14를 다시 참조하면, 섀도우 탭핑-지연 라인(78(1))은 또한 탭핑-지연 라인(78)에서 제공된다. 섀도우 탭핑-지연 라인(78(1))은 후속 벡터 프로세싱 연산을 위해 벡터 데이터 파일들(82(0)-82(X))로부터 다음 입력 벡터 데이터 샘플 세트(86N(0)-86N(X))를 래치 또는 파이프라인하는데 이용될 수 있다. 섀도우 탭핑-지연 라인(78(1))은 또한, 입력 벡터 데이터 샘플들의 해상도를 1차 탭핑-지연 라인(78(0))과 유사한 길이의 8 비트들로 낮추도록 허용하기 위해 복수의 8 비트 섀도우 파이프라인 레지스터들(122)로 구성된다. 섀도우 파이프라인 레지스터들(122)은 집합적으로 이 예에서 512 비트들인 벡터 데이터 파일들(82(0)-82(X))의 폭이어서, 섀도우 탭핑-지연 라인(78(1))은 또한 1차 탭핑-지연 라인(78(0))처럼 하나(1)의 입력 벡터 데이터 샘플 세트(86(0)-86(X))의 전체 폭을 저장할 수 있다. 따라서, 이 실시예에서, 1차 탭핑-지연 라인(78(0))에 포함되는 섀도우 파이프라인 레지스터들(122(0)-122(4X+3))의 수는 총 열여섯개(16)인 벡터 데이터 레인들(100(0)-100(X))의 수의 4배이며, 각각의 벡터 데이터 레인(100(0)-100(X))은 이 예에서 각각 32 비트를 지원할 수 있다. 따라서, 1차 파이프라인 레지스터들(120)의 수는 또한, 이 예에서 총 512 비트들(즉, 64 레지스터들 x 8 비트들 각각)에 대해 총 육십네개(64)이다.

[0116] 도 15b는 상관 벡터 프로세싱 연산(140)의 제 2 클록 사이클(CYCLE1) 동안 섀도우 탭핑-지연 라인(78(1))으로 로딩되는 다음 입력 벡터 데이터 샘플 세트(86N(0)-86N(X))를 예시한다. 다음 입력 벡터 데이터 샘플 세트(86N(0)-86N(1))는, 벡터 데이터 파일들(82(0)-82(X))로부터의 제 1 입력 벡터 데이터 샘플 세트(86(0)-86(X))가 상관 벡터 프로세싱 연산(140)의 실행을 셋업하도록 1차 탭핑-지연 라인(78(0))으로 로딩된 이후 섀도우 탭핑-지연 라인(78(1))으로 로딩된다. 이러한 다음 입력 벡터 데이터 샘플 세트(86N(0)-86N(X))는 정시 및 늦은 입력 벡터 데이터 샘플들(86OT, 86L) 둘 다를 갖는 입력 벡터 데이터 샘플들(X(32)-X(63))로서 섀도우 탭핑-지연 라인(78(1))으로 로딩된다. 이 예에서, 위에서 논의된 1차 탭핑-지연 라인(78(0))에서 제공되는 저장 패턴과 같이 X(32) 및 X(33)는 입력 벡터 데이터 샘플(86(0))의 정시 입력 벡터 데이터 샘플들(86OT)을 형성하고, X(33) 및 X(34)는 입력 벡터 데이터 샘플(86(0))의 늦은 입력 벡터 데이터 샘플들(86L)을 형성한다. 다른 패턴들은 입력 벡터 데이터 샘플 세트(86(0)-86(X))를 형성하도록 입력 벡터 데이터 샘플들(86)을 함께 그룹핑하기 위해 제공될 수 있다. 시퀀스 번호 생성기(134)로부터 기준 벡터 데이터 샘플 세트(130(0)-130(X))로부터의 상관 벡터 프로세싱 연산(140)의 제 1 프로세싱 스테이지 동안 상관되는 기준 벡터 데이터 샘플들(130)(즉, Y(0) 및 Y(1))은 또한 상관 벡터 프로세싱 연산(140)에서 이용하기 위해 도 15b의 실행 유닛들(84(0)-84(X))로 레지스터("C")에서 제공되는 것으로서 도시된다.

[0117] 도 14를 다시 참조하면, 입력 벡터 데이터 샘플 세트(86(0)-86(X))의 입력 벡터 데이터 샘플들(86)이 상관 벡터 프로세싱 연산(140)의 각각의 프로세싱 스테이지 동안 1차 탭핑-지연 라인(78(0))에서 시프트되기 때문에, 섀도우 파이프라인 레지스터들(122)에 저장된 다음 입력 벡터 데이터 샘플들(86N)은 또한 섀도우 탭핑-지연 라인(78(1))의 섀도우 파이프라인 레지스터들(122)에서 시프트된다. 이 예에서, 입력 벡터 데이터 샘플 세트(86(0)-86(X))의 입력 벡터 데이터 샘플들(86)이 정시 및 늦은 버전들로서 저장되기 때문에, 도 14의 탭핑-지연 라인들(78(0) 및 78(1)) 간에 제공된 시프트 패턴은 도 7의 탭핑-지연 라인들(78(0) 및 78(1)) 간에 제공된 시프트 패턴과 상이하다. 도 14에서 도시된 바와 같이, 정시 입력 벡터 데이터 샘플들(86OT)은 섀도우 탭핑-지연 라인(78(1))의 섀도우 파이프라인 레지스터들(122(0))로부터 1차 탭핑-지연 라인(78(0))의 1차 파이프라인 레지스터(120(2X+1))로 시프트된다. 마찬가지로, 늦은 입력 벡터 데이터 샘플들(86L)은 섀도우 탭핑-지연 라인(78(1))의 섀도우 파이프라인 레지스터(122(2X+2))로부터 1차 탭핑-지연 라인(78(0))의 1차 파이프라인 레지스터(120(4X+3)))로 시프트된다. 이러한 방식으로, 정시 입력 벡터 데이터 샘플들(86OT) 및 늦은 입력 벡터 데이

터 샘플들(86OT)은, 입력 벡터 데이터 샘플들(86)의 시프트가 상관 벡터 프로세싱 연산(140) 동안 발생하기 때문에 탭핑-지연 라인들(78(0), 78(1))에서 서로 분리된 채로 유지된다.

[0118] 상관 벡터 프로세싱 연산(140) 프로세싱 스테이지들은 실행 유닛(84(0)-84(X))에서 진행되고, 결국, 섀도우 탭핑-지연 라인(78(1))에 초기에 저장된 전체 다음 입력 벡터 데이터 샘플 세트(86N(0)-86N(X))는 프로세싱을 위해 실행 유닛들(84(0)-84(X))에 제공되도록 1차 탭핑-지연 라인(78(0))에서 완전히 시프트된다. 이러한 방식으로, 상관 벡터 프로세싱 연산(140)이 현재 입력 벡터 데이터 샘플 세트(86(0)-86(X)) 상에서 완료된 이후, 실행 유닛들(84(0)-84(X))에는 그 후, 원하는 경우, 지연 없이 다음 상관 벡터 프로세싱 연산(140)을 위한 현재 입력 벡터 데이터 샘플 세트(86(0)-86(X))로서 1차 탭핑-지연 라인(78(0))에 저장된 이전의 다음 입력 벡터 샘플 데이터 세트(86N(0)-86N(X))가 제공될 수 있다.

[0119] 제 1 입력 벡터 데이터 샘플 세트(86(0)-86(X)) 및 다음 입력 벡터 데이터 샘플 세트(86N(0)-86N(X))가 도 15b에서 도시된 바와 같이 각각 1차 탭핑-지연 라인(78(0)) 및 섀도우 탭핑-지연 라인(78(1))으로 로딩된 이후, 1차 탭핑-지연 라인(78(0))에 제공된 제 1 입력 벡터 데이터 샘플 세트(86(0)-86(X))는 상관 벡터 프로세싱 연산(140)의 제 1 프로세싱 스테이지에서 프로세싱되도록 각각의 실행 유닛들(84(0)-84(X))에 제공된다(도 12a의 블록(146)). 제 1 입력 벡터 데이터 샘플 세트(86(0)-86(X))가 실행 유닛들(84(0)-84(X))에 의해 프로세싱되는 현재 입력 벡터 데이터 샘플 세트(86(0)-86(X))가 된다. 도 11의 VPE(22(2))에서 예시된 바와 같이, 현재 입력 벡터 데이터 샘플(86(0))은 실행 유닛(84(0))에 제공되고, 현재 입력 벡터 데이터 샘플(86(1))은 실행 유닛(84(1))에 제공되는 식이다. 입력 벡터 데이터 샘플 세트(86(0)-86(X))와 상관될 기준 벡터 데이터 입력 샘플들(130(0)-130(X))은 상관 벡터 프로세싱 연산(140)의 현재 프로세싱 스테이지의 실행 유닛들(84(0)-84(X))에 제공된다(도 12a의 블록(148)).

[0120] 다음으로, 실행 유닛들(84(0)-84(X))은 상관 벡터 프로세싱 연산(140)을 수행한다(도 12a의 블록(150). 보다 구체적으로, 실행 유닛들(84(0)-84(X))은, 연산(정시 입력 벡터 데이터 샘플들(86OT)에 대해 R(OT)[n] = y[0] * x[n], 그리고 늦은 입력 벡터 데이터 샘플들(86L)에 대해 R(L)[n] = y[1] * x[1+n], 여기서 y[]는 지정된 기준 벡터 데이터 샘플(130)이고 x[n]은 현재 입력 벡터 데이터 샘플 세트(86(0)-86(X)임)에 따라, 제 1 프로세싱 스테이지 동안 기준 벡터 데이터 샘플들(130)로 현재 입력 벡터 데이터 샘플 세트(86(0)-86(X))를 곱셈한다. 상관의 결과는 현재 정시 상관 출력 벡터 데이터 샘플 세트(R(OT)[n]) 및 현재 늦은 상관 출력 벡터 데이터 샘플 세트(R(L)[n])이다. 실행 유닛들(84(0)-84(X))은 그 후, 새로운 이전의 입력 벡터 데이터 샘플 세트들(86(0)-86(X))을 제공하도록 각각의 현재 결과적 상관 벡터 데이터 샘플 세트를 실행 유닛들(84(0)-84(X))에 의해 계산된 그의 대응하는 이전의 결과적 상관 벡터 데이터 샘플 세트와 누산한다(도 12b의 블록(152)). 상관 벡터 프로세싱 연산(140)의 제 1 프로세싱 스테이지에서, 이전의 결과적 상관된 출력 벡터 데이터 샘플 세트(132(0)-132(X))가 없다. 따라서, 제 1/현재 결과적 상관된 출력 벡터 데이터 샘플 세트(132(0)-132(X))는 단순히 상관 벡터 프로세싱 연산(140)의 다음 제 2 프로세싱 스테이지에 대해 이전의 입력 벡터 데이터 샘플 세트(86(0)-86(X))가 될 것이다.

[0121] 상관 벡터 프로세싱 연산(140)의 모든 프로세싱 스테이지들이 완료된 경우(도 12b의 블록(154)), 누산된 이전의 결과적 상관된 출력 벡터 데이터 샘플 세트(132(0)-132(X))는 벡터 데이터 파일들(82(0)-82(X))에 제공되고 저장되도록 출력 데이터 흐름 경로들(98(0)-98(X))에서 결과적 상관된 출력 벡터 데이터 샘플 세트(132(0)-132(X))로서 제공된다(도 12b의 블록(157)). 상관 벡터 프로세싱 연산(140)의 모든 프로세싱 스테이지들이 완료되지 않은 경우(도 12a의 블록(154)), 시트프된 입력 벡터 데이터 샘플 세트(86S(0)-86S(X))는 시프트된 입력 벡터 데이터 샘플 세트(86S(0)-86S(X))를 제공하도록 상관 벡터 프로세싱 연산(140)에 대한 다음 포지션으로 탭핑-지연 라인들(78(0), 78(1))에서 시프트된다(도 12b의 블록(156)). 시프트된 입력 벡터 데이터 샘플 세트(86S(0)-86S(X))는 이전의 결과적 상관 출력 벡터 데이터 샘플 세트(132(0)-132(X))와 누산될 다음 결과적 상관 출력 벡터 데이터 샘플 세트(132(0)-132(X))를 계산하기 위해 제공된다. 탭핑-지연 라인들(78(0), 78(1))에서 입력 벡터 데이터 샘플들(86)의 시프트는 도 14에 관하여 앞서 상술되었다.

[0122] 도 15c는, 입력 벡터 데이터 샘플 세트(86(0)-86(X))가 다음 상관 프로세싱 연산(140)(온-=타입 입력 벡터 데이터 샘플들(86SOT)에 대해 R(OT)[n] = y[2] * x[2+n] 및 늦은 입력 벡터 데이터 샘플들(86SL)에 대해 R(L)[n] = y[3] * x[3+n])을 위해 새로운 시프트된 입력 벡터 데이터 샘플 세트(86S(0)-86S(X))가 되도록 상관 벡터 프로세싱 연산(140)의 제 2 프로세싱 스테이지에서 시프트될 때 탭핑-지연 라인들(78)의 컨텐츠들을 예시한다. 1차 탭핑-지연 라인(78(0))의 입력 벡터 데이터 샘플 세트(86(0)-86(X))는 2개의 입력 벡터 데이터 샘플들(86)에 의해 시프트된다. 예를 들어, X(2) 및 X(3)의 도 15b에서 입력 벡터 데이터 샘플(86OT(1))은 이제 도 15c의 입력 벡터 데이터 샘플(86S(0))로 시프트된다. 시프트된 입력 벡터 데이터 샘플 세트(86S(0)-86S(X))는 현재 입력

벡터 데이터 샘플 세트(86(0)-86(X))가 된다. 실행 유닛들(84(0)-84(X))에 제공되는 기준 벡터 데이터 샘플들(130)은 또한 이 예에서 Y(2) 및 Y(3)인 기준 벡터 데이터 샘플들(130)이다.

[0123] 도 12b를 예속 참조하면, 프로세스는 다음 기준 벡터 데이터 샘플들(130)과 곱셈되도록 1차 탭핑-지연 라인(78(0))으로부터(그리고 섀도우 탭핑-지연 라인(78(1))의 부분으로부터) 실행 유닛들(84(0)-84(X))로 다음 시프트된 입력 벡터 데이터 샘플 세트(86S(0)-86S(X))를 제공함으로써 반복되며(도 12a의 블록(150)), 결과적 상관된 출력 벡터 데이터 샘플 세트(132(0)-132(X))는 이전의 결과적 상관된 출력 벡터 데이터 샘플 세트(132(0)-132(X))와 누산된다(도 12b의 블록(152)). 도 15d는 예시적인 상관 벡터 프로세싱 연산(140)의 마지막 프로세싱 스테이지 동안 탭핑-지연 라인들(78(0), 78(1))에 존재하는 입력 벡터 데이터 샘플들(86)의 상태를 예시한다. 이 예에서, 도 15d에서 도시된 바와 같이, 상관 벡터 프로세싱 연산(140)에 대한 열여섯개(16)의 프로세싱 스테이지들이 있는데, 그 이유는 탭핑-지연 라인들(78)의 전체 데이터 폭이 입력 벡터 데이터 샘플 세트(86(0)-86(X))에 대해 이용되지만, 정시 및 늦은 입력 벡터 데이터 샘플들(86OT, 86L) 사이에서 분할되기 때문이다. Y(30) 및 Y(31)는 도 13의 예에서 기준 벡터 데이터 샘플들(130(15))인, 도 15d에서 도시된 바와 같이 상관 벡터 프로세싱 연산(140)의 마지막 기준 벡터 데이터 샘플들(130(X))이다. 시프트된 입력 벡터 데이터 샘플 세트(86S(0)-86S(X))는 열여섯번(16)(이 예에서, 벡터 데이터 레인들(100(0)-100(X))의 폭) 시프트되어서, 입력 벡터 데이터 샘플들(X(30) 및 X(31))은 상관 벡터 프로세싱 연산(140)에 대해 최종의 16번째 프로세싱 스테이지에서 1차 탭핑-지연 라인(78(0))의 시프트된 입력 벡터 데이터 샘플(86S(0))에 저장되게 된다.

[0124] 도 16은, 위의 예에서 예시적인 열여섯개(16)의 상관 벡터 프로세싱 스테이지들이 완전히 실행된 이후, 도 11의 VPE(22(2))에서 실행 유닛들(84(0)-84(X))의 누산기들의 컨텐츠의 개략도(즉, 결과적 상관된 출력 벡터 데이터 샘플들(132))이다. 결과적 상관된 출력 벡터 데이터 샘플 세트는 132(0)-132(X)로서 도시된다. 누산기들(Acc0-Acc3)은 도 16에서 도시되는데, 그 이유는 이 예에서, 각각의 실행 유닛(84(0)-84(X))은 각각의 벡터 데이터 레인(100(0)-100(X))에 대해 병렬로 배치된 4개의 누산기들을 갖기 때문이다. 누산된 결과적 출력 벡터 데이터 샘플들은 추가의 분석 및/또는 프로세싱을 위해 그 내부에 저장되도록 출력 데이터 흐름 경로들(98(0)-98(X)) 상에서 집합적인 결과적 상관된 출력 벡터 데이터 샘플 세트(132(0)-132(X))로서 벡터 데이터 파일들(82(0)-82(X))에 제공될 수 있다. 특수 벡터 명령은 원하는 경우, 벡터 데이터 파일들(82(0)-82(X))로부터 벡터 유닛 데이터 메모리(32)(도 2 참조)로 결과적 상관된 출력 벡터 데이터 샘플 세트(132(0)-132(X))의 행들을 이동시키도록 VPE(22(2))에 의해 지원될 수 있다.

[0125] 위에서 설명된 결과적 필터 벡터 출력 데이터 샘플 세트들(94(0)-94(X)) 및 결과적 상관된 출력 벡터 데이터 샘플 세트들(132(0)-132(X))을 비롯해서, 실행 유닛들(84(0)-84(X))에 의해 제공되는 결과적 출력 벡터 데이터 샘플 세트들은 VPE에 의해 실행되는 벡터 명령에 의존하여 상이한 인터리빙된 포맷들로 벡터 데이터 파일들(82(0)-82(X), 82(31))에 다시 저장될 수 있다. 'X'는 이 예에서 벡터 데이터 파일들(82(0)-82(X))을 제공하도록 삼십일(31)과 동일하며, 각각이 삼십이(32) 비트 폭이다. 예를 들어, 도 17a에서 예시된 바와 같이, 결과적 출력 벡터 데이터 샘플 세트(158(0)-158(X), 158(31))는 그의 실수("q") 및 허수("i") 컴포넌트들에 의해 분리되는 벡터 데이터 파일들(82(0)-82(X))에 저장될 수 있다. 결과적 출력 벡터 데이터 샘플 세트(158(0)-158(X))는 이 예에서 158(0), 158(1), … , 및 158(X)인 'X+1' 결과적 출력 벡터 데이터 샘플들(158)로 구성된다. 예컨대, 다음 벡터 명령들이 입력 벡터 데이터 샘플 세트로서 결과적 출력 벡터 데이터 샘플 세트(158(0)-158(X), 158(31))의 실수 및 허수 컴포넌트들 상에서 연산하는 경우, 효율성 목적을 위해 그의 실수("q") 및 허수("i") 컴포넌트들에 의해 분리되는 결과적 출력 벡터 데이터 샘플 세트(158(0)-158(X), 158(31))를 저장하는 것이 보다 더 효율적일 수 있다. 아니면, 그의 실수 및 허수 컴포넌트들로 결과적 출력 벡터 데이터 샘플(158)의 분리를 위해 벡터 데이터 파일(82)에 결과적 출력 벡터 데이터 샘플(158)을 저장하는 것이 가능하지 않을 수 있다. 예를 들어, 십육(16) 비트 벡터 데이터 샘플이 다른 십육(16) 비트 벡터 데이터 샘플에 의해 곱셈되는 경우, 삼십이(32) 비트의 결과적 벡터 데이터 샘플들이 발생한다. 예를 들어, 삼십이(32) 비트의 결과적 출력 벡터 데이터 샘플(158)은 도 17a에서 Y0일 수 있다. Y0의 허수 컴포넌트(Y0.i 158(I))는 벡터 데이터 파일(82(0))의 어드레스 '0'에 저장될 수 있고, Y0의 실수 컴포넌트(Y0.q 158(Q))는 어드레스 "A"와 같은 다른 어드레스에 저장될 수 있다.

[0126] 도 17a에서 결과적 출력 벡터 데이터 샘플 세트(158(0)-158(X), 158(31))는 짝수 및 홀수의 결과적 출력 벡터 데이터 샘플들에 의해 인터리빙되는 벡터 데이터 파일들(82(0)-82(X), 82(31))에 저장될 수 있다. 이는 도 17b의 예에 의해 예시된다. 도 17b에서 예시된 바와 같이, 결과적 출력 벡터 데이터 샘플 Y0-Y31(158(0)-158(X), 158(31))은 벡터 데이터 파일들(82(0)-82(31))의 어드레스 '0'과 어드레스 'A' 사이에서 짝수 및 홀수의 벡터 데이터 샘플들에 의해 인터리빙된 포맷으로 저장된다. 결과적 출력 벡터 데이터 샘플 Y0(158(0))은 벡터 데이

터 파일들(82(0))의 어드레스 '0'에 저장된다. 결과적 출력 벡터 데이터 샘플 Y1(158(1))은 벡터 데이터 파일들(82(1))의 어드레스 '0'이 아니라 벡터 데이터 파일들(82(0))의 어드레스 'A'에 저장된다. 결과적 출력 벡터 데이터 샘플 Y2(158(2))는 벡터 데이터 파일들(82(1))의 어드레스 '0'에 저장되는 식이다.

[0127] 특정한 무선 기저대역 연산들은 프로세싱되기 이전에 데이터 샘플들이 포맷-변환되도록 요구한다. 예를 들어, 도 17a 및 도 17b에서 인터리빙된 포맷으로 벡터 데이터 파일들(82(0)-82(X))에 저장된 결과적 출력 벡터 데이터 샘플 세트들(158(0)-158(X))은 다음 벡터 프로세싱 연산을 위해 디인터리빙될 필요가 있다. 예를 들어, 결과적 출력 벡터 데이터 샘플들(158(0)-158(X))이 CDMA 신호를 나타내는 경우, 결과적 출력 벡터 데이터 샘플들(158(0)-158(X))은 신호의 짝수 및 홀수 위상들을 분리하도록 디인터리빙될 필요가 있다. 디인터리빙된 신호는 또한, 예컨대, 도 11 내지 16에 관하여 상술된 예시적인 상관 벡터 프로세싱 연산으로 CDMA 시스템이 신호를 추출할 수 있는지를 결정하도록 상관 프로세싱 연산에서 로컬적으로 생성된 코드 또는 시퀀스 번호와 상관될 수 있다. 종래의 프로그램가능 프로세서들은 다수의 단계들에서 데이터 샘플의 포맷 변환을 구현하며, 이는 벡터 데이터 샘플 포맷 변환들에서 사이클, 전력 소비 및 데이터 흐름 복잡도를 부가한다. 다양한 프로세서들은 포맷-변환된 벡터 데이터 샘플들이 실행 유닛들에 제공되기 이전에 포맷 변환들을 제공하도록 벡터 데이터 샘플들을 사전-프로세싱할 수 있다. 포맷-변환된 벡터 데이터 샘플들은 벡터 데이터 메모리에 저장되고 실행 유닛들에 의해 프로세싱되도록 데이터 포맷 변환을 요구하는 벡터 프로세싱 연산의 부분으로서 재패치된다. 그러나 벡터 데이터 샘플들의 이러한 포맷 사전-프로세싱은 실행 유닛들에 의해 포맷-변환된 벡터 데이터 샘플들의 후속 프로세싱을 지연하고 실행 유닛들의 계산 컴포넌트들이 덜 활용되게 한다.

[0128] 여기에 그리고 아래에 개시된 실시예들은, 도 18a 및 도 18b에 예시된 것들과 같은 인터리빙된 벡터 데이터 샘플 세트들의 변환을 제공한다. 예컨대, 도 18a 및 도 18b는 상이한 포맷들로 벡터 데이터 파일들(82(0)-82(X))에 저장된 벡터 데이터 샘플 세트 D(0)-D(X)를 예시한다. 도 18a는 부호화된 복소(SC) 16 비트 샘플들(SC16)에 저장되고 실수 및 허수 성분들에 의해 포맷-인터리빙된 벡터 데이터 샘플 세트 D(0)-D(X)를 예시한다. 삼십이(32) 비트 벡터 데이터 샘플 D(0)의 십육(16) 비트 실수 및 허수 성분들 - D(0)(Q) 및 D(0)(I)는, 삼십이(32) 비트 벡터 데이터 파일(82(0))에 저장된다. 벡터 데이터 샘플 D(X)의 십육(16) 비트 실수 및 허수 성분들 - D(X)(Q) 및 D(X)(I)는, 삼십이(32) 비트 벡터 데이터 파일(82(X))에 저장된다. 도 18b는 SC 8 비트 샘플들(SC8)에 저장되고 실수 및 허수 성분들에 의해 포맷-인터리빙된 벡터 데이터 샘플 세트 D(0)-D(X)를 예시한다. 십육(16) 비트 벡터 데이터 샘플 D(0)(1)의 팔(8) 비트 실수 및 허수 성분들 - D(0)(1)(Q), D(0)(1)(I)는, 벡터 데이터 파일(82(0))에 저장된다. 십육(16) 비트 벡터 데이터 샘플 D(0)(0)의 팔(8) 비트 실수 및 허수 성분들 - D(0)(0)(Q), D(0)(0)(I)는 또한, 삼십이(32) 비트 벡터 데이터 파일(82(0))에 저장된다. 마찬가지로, 십육(16) 비트 벡터 데이터 샘플 D(X)(1)의 팔(8) 비트 실수 및 허수 성분들 - D(X)(1)(Q), D(X)(1)(I)는, 삼십이(32) 비트 벡터 데이터 파일(82(X))에 저장된다. 십육(16) 비트 벡터 데이터 샘플 D(X)(0)의 팔(8) 비트 실수 및 허수 성분들 - D(X)(0)(Q), D(X)(0)(I)는 또한 삼십이(32) 비트 벡터 데이터 파일(82(X))에 저장된다.

[0129] 이와 관련하여, 도 19는, 도 2의 VPE(22)로서 제공될 수 있는 다른 예시적인 VPE(22(3))의 개략도이다. 아래에서 더 상세히 설명될 바와 같이, 도 19의 VPE(22(3))는, 제거된 또는 감소된 벡터 데이터 샘플 재패치 및 감소된 전력 소비로, VPE(22(3))에서의 벡터 프로세싱 연산들에 대한 실행 유닛들에 제공되는 입력 벡터 데이터 샘플 세트들의 실시간 포맷 변환(예컨대, 디인터리빙)을 제공하도록 구성된다. 입력 벡터 데이터 샘플 세트들의 실시간 포맷 변환은, 벡터 데이터 메모리로부터 리트리브되는 입력 벡터 데이터 샘플 세트가, 실행을 위한 실행 유닛들에 제공되기 전에 벡터 데이터 메모리에 저장되고 그로부터 재패치될 필요 없이 포맷-변환됨을 의미한다. 전력 소비를 감소시키고 프로세싱 효율을 증가시키도록 벡터 데이터 파일로부터 입력 벡터 데이터 샘플들의 재패치를 제거 또는 감소시키기 위해, 벡터 데이터 파일들(82(0)-82(X))과 실행 유닛들(84(0)-84(X)) 사이의 벡터 데이터 레인들(100(0)-100(X)) 각각에 포맷 변환 회로(159(0)-159(X))가 포함된다. 아래에서 더 상세히 논의될 바와 같이, 입력 벡터 데이터 샘플 세트(86(0)-86(X))의 디인터리빙을 요구하는 벡터 프로세싱 연산을 위한 실행 유닛들(84(0)-84(X))에 포맷-변환된 입력 벡터 데이터 샘플 세트(86F(0)-86F(X))를 제공하기 위해, 벡터 데이터 파일들(82(0)-82(X))로부터의 입력 벡터 데이터 샘플 세트(86(0)-86(X))는 VPE(22(3))의 포맷 변환 회로(159(0)-159(X))에서 포맷-변환된다(예컨대, 디인터리빙된다). 포맷-변환된 입력 벡터 데이터 샘플들(86F) 모두는, 이 예에서는 포맷-변환된 입력 벡터 데이터 샘플 세트(86F(0)-86F(X))를 포함한다. 'X'+1은, 이 예에서, 입력 벡터 데이터 샘플들(86)의 프로세싱을 위해 VPE(22(3))에서 제공되는 병렬 입력 데이터 레인들의 최대 수이다.

[0130] 이러한 방식으로, VPE(22(3))에서 입력 벡터 데이터 샘플 세트들(86(0)-86(X))의 포맷 변환은, 사전-프로세싱, 저장 및 벡터 데이터 파일들(82(0)-82(X))로부터의 재패치를 요구하지 않아서, 전력 소비를 감소시킨다. 추가

로, 입력 벡터 데이터 샘플 세트들(86(0)-86(X))의 포맷 변환은, 사전-프로세싱, 저장, 및 벡터 데이터 파일(82(0)-82(X))로부터 포맷-변환된 입력 벡터 데이터 샘플 세트들(86(0)-86(X))의 재패치를 요구하지 않기 때문에, 실행 유닛들(84(0)-84(X))은 벡터 프로세싱 연산들을 수행하는 것으로부터 지연되지 않는다. 따라서, VPE(22(3))에서 데이터 흐름 경로들의 효율은, 입력 벡터 데이터 샘플 세트들(86(0)-86(X))의 포맷 변환 사전-프로세싱 지연들에 의해 제한되지 않는다. 포맷-변환된(예컨대, 디인터리빙된) 입력 벡터 데이터 샘플 세트들(86F(0)-86F(X))은 실행 유닛들(84(0)-84(X))에 제공되어 로컬화된다. 실행 유닛들(84(0)-84(X))에서의 벡터 프로세싱은, 데이터 흐름 제한들에 의하기 보다는 오직 연산 자원들에 의해서만 제한된다.

[0131] 1차 및 섀도우 탭핑-지연 라인들(78(0), 78(1))이 도 19의 VPE(22(3))에 예시되지만, 도 19의 VPE(22(3))에 탭핑-지연 라인을 포함시키는 것이 요구되는 것은 아님을 주목한다. 이 예에서, 도 19에 예시된 바와 같이, 포맷 변환 회로(159(0)-159(X))는 선택적인 1차 탭핑-지연 라인(78(0))에 포함될 수 있다. 이러한 어레인지먼트는, 벡터 데이터 파일들(82(0)-82(X))과 도 19의 VPE(22(3))의 실행 유닛들(84(0)-84(X)) 사이의 입력 데이터 흐름 경로들(80(0))-80(X))에 포맷 변환 회로(159(0)-159(X))를 제공한다. 1차 탭핑-지연 라인(78(0))의 연산은, VPE들(22(1) 및 22(2))에 대해 앞서 이미 설명되었다. 앞서 이미 논의된 바와 같이, 1차 및 섀도우 탭핑-지연 라인들(78(0), 78(1))은, 포맷-변환된 입력 벡터 데이터 샘플 세트들(86F(0)-86F(X))이 실행 유닛들(84(0)-84(X))에 제공되도록 요구하는 벡터 프로세싱 연산에 대해 이용될 수 있고, 다음, 이는 또한, 86SF(0)-86SF(X)로 지정된 포맷-변환된 시프트된 입력 벡터 데이터 샘플 세트들을 요구한다.

[0132] 도 19의 VPE(22(3))에 제공되는 것과 동일한 컴포넌트들 및 아키텍쳐가 도 11의 VPE(22(2))에 제공됨을 주목한다. 도 19의 VPE(22(3))와 도 11의 VPE(22(2)) 사이의 공통 컴포넌트들은, VPE(22(2))의 도 11의 컴포넌트들과 공통 엘리먼트 번호들을 갖는 것으로 도 19에 예시된다. 앞선 도 11의 VPE(22(2))에 대한 이러한 공통 컴포넌트들의 이전의 설명 및 논의는 또한, 도 19의 VPE(22(3))에 대해 적용가능하고, 따라서 여기서 다시 설명되지 않을 것이다.

[0133] 도 19의 VPE(22(3)), 및 이 실시예에서 입력 데이터 흐름 경로들(80(0)-80(X))의 실행 유닛들(84(0)-84(X))에 포맷-변환된 입력 벡터 데이터 샘플 세트(86F(0)-86F(X))를 제공하기 위한 탭핑-지연 라인들(78)의 추가적인 세부사항들 및 특징들의 추가적인 설명이 이제 설명될 것이다. 이와 관련하여, 도 20은, 입력 벡터 데이터 샘플 세트(86(0)-86(X))의 포맷 변환을 요구하는 예시적인 벡터 명령에 따라, 포맷 변환 회로(159(0)-159(X))를 이용하는 도 19의 VPE(22(3))에서 수행될 수 있는 예시적인 디인터리빙 포맷 변환 벡터 프로세싱 연산(160)을 예시하는 흐름도이다.

[0134] 도 20을 참조하면, 벡터 명령에 따른 벡터 프로세싱 연산(160)에 대한 입력 벡터 데이터 샘플 세트(86(0)-86(X))는 벡터 데이터 파일들(82(0)-82(X))로부터 입력 데이터 흐름 경로들(80(0)-80(X))로 패치된다(블록(162)). 예컨대, 벡터 프로세싱 연산(160)에 대한 포맷 변환은 디인터리빙 벡터 프로세싱 연산(160)일 수 있고, 입력 벡터 데이터 샘플 세트(86(0)-86(X))는, 벡터 데이터 파일들(82(0)-82(X))의 그의 인터리빙된 상태로부터 디인터리빙된 입력 벡터 데이터 샘플 세트(86F(0)-86F(X))로 디인터리빙된다. 벡터 프로세싱 연산(160)을 위해 포맷-변환될 입력 벡터 데이터 샘플 세트(86(0)-86(X))의 폭에 의존하는 벡터 명령의 프로그래밍에 따라 벡터 프로세싱 연산(160)을 제공하기 위해, 도 19의 VPE(22(3))에서 벡터 데이터 레인들(100(0)-100(X)) 중 어느 하나, 일부 또는 전부가 이용될 수 있다. 벡터 데이터 파일들(82(0)-82(X))의 전체 폭이 요구되면, 벡터 프로세싱 연산(160)을 위해 모든 벡터 데이터 레인들(100(0)-100(X))이 이용될 수 있다. 벡터 프로세싱 연산(160)은, 벡터 프로세싱 연산(160)에 대해 이용될 수 있는 벡터 데이터 레인들(100(0)-100(X))의 오직 서브세트만을 요구할 수 있다. 이것은, 입력 벡터 데이터 샘플 세트(86(0)-86(X))의 폭이 모든 벡터 데이터 파일들(82(0)-82(X)))의 폭보다 작기 때문일 수 있고, 여기서, 벡터 프로세싱 연산(160)과 병렬적으로 수행될 다른 벡터 프로세싱 연산들에 대해 추가적인 벡터 데이터 레인들(100)을 이용하는 것이 바람직하다. 현재의 예를 논의하기 위해, 벡터 프로세싱 연산(160)에 대한 입력 벡터 데이터 샘플 세트(86F(0)-86F(X))로 포맷-변환되는 입력 벡터 데이터 샘플 세트(86(0)-86(X))는 도 19의 VPE(22(3))의 모든 벡터 데이터 레인들(100(0)-100(X))을 수반하는 것으로 가정된다.

[0135] 도 20을 계속 참조하면, 패치된 입력 벡터 데이터 샘플 세트(86(0)-86(X))는, 포맷 변환 회로(159(0)-159(X))에 대한 입력 데이터 흐름 경로들(80(0)-80(X))로 제공되어, 벡터 프로세싱 연산(160)에 따라 포맷-변환된다(블록(164)). 비제한적인 예로서, 현재 입력 벡터 데이터 샘플 세트(86(0)-86(X))는 선택적으로, 벡터 프로세싱 연산(160)을 위해 실행 유닛들(84(0)-84(X))에 제공되기 전에, 포맷-변환된 입력 벡터 데이터 샘플 세트(86(0)-86(X))로서 1차 탭핑-지연 라인(78(0))에 로딩될 수 있다. 앞서 논의된 바와 같이, 다음 입력 벡터 데이터 샘플 세트(86(0)-86(X))는 또한, 선택적으로, 실행 유닛들(84(0)-84(X))에 의해 프로세싱될 다음 입력 벡터 데이

터 샘플 세트(86N(0)-86N(X))로서 새도우 탭핑-지연 라인(78(1))에 로딩될 수 있다. 앞서 먼저 논의된 바와 같이, 탭핑-지연 라인들(78)의 목적은, 입력 벡터 데이터 샘플 세트(86(0)-86(X))를, 시프트된 입력 벡터 데이터 샘플들(86S)에 대해 동작하는 벡터 프로세싱 연산(160)의 연산 동안 실행 유닛들(84(0)-84(X))에 제공될 시프트된 입력 벡터 데이터 샘플들(86S(0)-86S(X))로 시프트시키는 것이다. 포맷-변환된 입력 벡터 데이터 샘플 세트(86F(0)-86F(X))가 또한 벡터 프로세싱 연산(160) 동안 탭핑-지연 라인들(78)에서 시프트되면, 시프트된 포맷-변환된 입력 벡터 데이터 샘플 세트는 86SF(0)-86SF(X)로 지정된다.

[0136] 도 20을 계속 참조하면, 실행 유닛들(84(0)-84(X))은 다음으로, 포맷-변환된 입력 벡터 데이터 샘플 세트(86F(0)-86F(X))를 이용하여 벡터 프로세싱 연산(160)을 수행할 수 있다(블록(166)). 실행 유닛들(84(0)-84(X))은, 포맷-변환된 입력 벡터 데이터 샘플 세트(86F(0)-86F(X))를 이용하여 곱셈들 및/또는 누산을 제공하도록 구성될 수 있다. 벡터 프로세싱 연산(160) 동안 포맷-변환된 입력 벡터 데이터 샘플 세트(86F(0)-86F(X))를 시프트시키기 위해 탭핑-지연 라인들(78)이 이용되면, 실행 유닛들(84(0)-84(X))은, 벡터 프로세싱 연산(160)이 완료될 때까지 벡터 프로세싱 연산(160)의 각각의 프로세싱 스테이지 동안, 시프트된 포맷-변환된 입력 벡터 데이터 샘플 세트(86SF0)-86SF(X))를 수신할 수 있다(블록(168)). 벡터 프로세싱 연산(160)이 완료되면, 포맷-변환된 입력 벡터 데이터 샘플 세트(86F(0)-86F(X)) 또는 시프트된 포맷-변환된 입력 벡터 데이터 샘플 세트들(86SF(0)-86SF(X))과의 벡터 프로세싱에 기초하는 결과적 출력 벡터 데이터 샘플 세트(172(0)-172(X))는, 출력 데이터 흐름 경로들(98(0)-98(X))에 제공되어, 벡터 데이터 파일들(82(0)-82(X))에 제공 및 저장된다(블록(170)). 결과적 출력 벡터 데이터 샘플 세트(172(0)-172(X))는 'X+1'개의 결과적 출력 벡터 데이터 샘플들(172)로 이루어지고, 이는, 이 예에서 172(0), 172(1), … 및 172(X)이다.

[0137] 도 21은 1차 탭핑-지연 라인(78(0))으로부터 시프트된 입력 벡터 데이터 샘플 세트(86S(0)-86S(X))를 수신하는 예시적인 포맷 변환 회로(159(0)-159(X))의 개략도이다. 이 예에서, 포맷 변환 회로(159(0)-159(X))는 입력 데이터 흐름 경로들(80(0)-80(X))에서 1차 탭핑-지연 라인(78(0))의 출력 상에 제공된다. 예시적인 포맷 변환 회로(159(0)-159(X))가 이제 설명될 것이다.

[0138] 예시적인 포맷 변환 회로(159(0)-159(X))가 이제 설명될 것이다. 포맷 변환 회로(159(0))의 내부 컴포넌트들의 예시적인 세부사항이 도 21에 제공되지만, 이것은 또한 포맷 변환 회로(159(1)-159(X))에 대해 적용가능하다. 도 21의 포맷 변환 회로(159(0))를 일례로 고려하면, 이 예의 포맷 변환 회로(159(0))는 포맷-변환된 입력 벡터 데이터 샘플들(86F(0)) 또는 시프트된 포맷-변환된 입력 벡터 데이터 샘플들(86SF(0))을 각각 제공하기 위해, 벡터 데이터 레인(100(0))의 1차 파이프라인 레지스터들(120(0), 120(1) 120(2X+2), 120(2X+3))로부터 입력 벡터 데이터 샘플(86(0)) 또는 시프트된 입력 벡터 데이터 샘플들(86S(0))의 디인터리빙 및 부호 확장을 제공하도록 구성된다. 이와 관련하여, 이 예에서는 4개의 멀티플렉서들(174(3)-174(0))이 제공되고, 이들은 할당된 1차 파이프라인 레지스터(120(0)-120(2X+3))에 따라 각각 어레인지된다. 각각의 멀티플렉서(174(3)-174(0))는, 할당된 1차 파이프라인 레지스터(120(0), 120(1), 120(2X+2), 120(2X+3))에 인접한 1차 파이프라인 레지스터(120)에 저장할, 할당된 1차 파이프라인 레지스터(120(0), 120(1), , 120(2X+2), 120(2X+3))의 시프트된 입력 벡터 데이터 샘플(86S(0))의 일부 또는 시프트된 입력 벡터 데이터 샘플(86S(0))의 일부를 선택하도록 구성된다.

[0139] 예컨대, 1차 파이프라인 레지스터들(120(0), 120(1), 120(2X+2), 120(2X+3))이 인터리빙된 시프트된 입력 벡터 데이터 샘플(86S(0))을 실수 [15:8], 허수 [15:8], 실수 [7:0], 허수 [7:0]의 복소 인터리빙된 형태로 저장하고, 원하는 디인터리빙된 포맷이 실행될 벡터 명령에 따라 실수 [15:0] 및 허수 [15:0]이면, 멀티플렉서(174(3)-174(0)) 선택들은 다음과 같을 것이다. 멀티플렉서(174(3))는, 그의 할당된 1차 파이프라인 레지스터(120(0))에 저장된 시프트된 입력 벡터 데이터 샘플(86S)의 일부를 선택할 것이다. 그러나, 멀티플렉서(174(2))는, 1차 파이프라인 레지스터(120(1))에 저장된 시프트된 입력 벡터 데이터 샘플(86S)의 일부를 선택할 것이다. 이것은, 인접한 입력 데이터 흐름 경로들(80(0)(3), 80(0)(2))에 입력 벡터 데이터 샘플(86S(0))의 디인터리빙된 실수부(즉, 실수 [15:0])를 제공할 것이다. 유사하게, 멀티플렉서(174(0))는, 그의 할당된 1차 파이프라인 레지스터(120(2X+3))에 저장된 시프트된 입력 벡터 데이터 샘플(86S)의 일부를 선택할 것이다. 그러나, 멀티플렉서(174(1))는, 1차 파이프라인 레지스터(120(2X+2))에 저장된 시프트된 입력 벡터 데이터 샘플(86S)의 일부를 선택할 것이다. 이것은, 인접한 입력 데이터 흐름 경로들(80(0)(1), 80(0)(0))에 시프트된 입력 벡터 데이터 샘플(86S(0))의 디인터리빙된 허수부(즉, 허수 [15:0])를 제공할 것이다. 도 21에 예시된 바와 같이, 멀티플렉서들(176(1), 176(0))은, 할당되지 않고 인접하지 않은 1차 파이프라인 레지스터들(120(0), 120(1), 120(2X+2)-120(2X+3))로부터 시프트된 입력 벡터 데이터 샘플(86S(0))의 일부를 선택하는 능력을 각각의 멀티플렉서(174(3)-174(0))에 제공한다.

[0140] 도 21을 계속 참조하면, 포맷 변환 회로(159(0)-159(X))는 또한, 포맷-변환된 입력 벡터 데이터 샘플 세트들(86F(0)-86F(X))을 부호 확장하도록 구성될 수 있다. 예컨대, 입력 벡터 데이터 샘플 세트들(86(0)-86(X))의 포맷 변환이 작은 비트 폭들로부터 큰 비트 폭들로 변환된 부호화된 벡터 데이터 샘플들을 수반하면, 포맷 변환 회로(159(0)-159(X))는, 최상위 비트들을, 음이 아닌 수들에 대해 '0''들로 그리고 음인 수들에 대해 "F"'들로 확장함으로써 디인터리빙된 벡터 데이터 샘플들을 부호 확장하도록 구성될 수 있다. 포맷 변환 회로(159(0)-159(X))는, 포맷-변환된 입력 벡터 데이터 샘플 세트(86F(0)-86F(X))에 대해 부호 확장이 수행될지 여부를 표시하기 위해, 실행되고 있는 벡터 명령에 따라 설정된 부호 확장(SC) 입력(178(0)-178(X))을 가질 수 있다. SC 입력들(178(0)-178(X))은, 프로세싱되고 있는 벡터 명령에 따라 SC 입력들(178(0)-178(X))에 의해 제공되는 프로그래밍가능한 데이터 경로 구성에 따라 부호 확장을 수행하도록, 포맷 변환 회로(159(0)-159(X))에 제공된 부호 확장 회로(180(0)-180(X))에 제공될 수 있다. SC 입력들(178(0)-178(X))은, VPE(22(3))에 의한 벡터 프로세싱에서 유연성을 제공하기 위해 각각의 벡터 명령에 대해 구성 및 재구성될 수 있다. 예컨대, 포맷 변환 회로(159(0)-159(X))의 프로그래밍가능한 데이터 경로들은 SC 입력들(178(0)-178(X))에 의해 구성될 수 있고, 원한다면 실행 유닛들(84(0)-84(X))의 완전한 활용에 의해, 원하는 포맷 변환을 제공하기 위해, 원한다면 클록-사이클-바이-클록-사이클 기반으로, 벡터 명령의 각각의 클록-사이클에 대해 구성 및 재구성될 수 있다.

[0141] 그러나, 앞서 논의된 바와 같이, 포맷 변환 회로(159(0)-159(X))는, 1차 탭핑-지연 라인(78(0))의 일부로서 제공될 필요가 없다. 1차 및 섀도우 탭핑-지연 라인들(78(0), 78(1))은 선택적이다. 포맷 변환 회로(159(0)-159(X))는, 벡터 데이터 파일들(82(0)-82(X))로부터 직접 입력 벡터 데이터 샘플 세트들(86(0)-86(X))을 수신할 수 있다. 일예로 이러한 시나리오에서, 도 21을 참조하면, 입력 벡터 데이터 샘플 세트(86(0)-86(X))는, 벡터 레지스터 파일들(82(0)-82(X))로부터 1차 레지스터들(120(0)-120(4X+3))에 직접 로딩될 수 있다.

[0142] 추가로, 입력 벡터 데이터 샘플 세트들(86(0)-86(X))을 포맷 변환하기 위해 1차 탭핑-지연 라인(78(0))의 출력상에 포맷 변환 회로(159(0)-159(X))가 제공되지만, 이것이 요구되는 것은 아님을 주목한다. 도 21의 포맷 변환 회로(159(0)-159(X))는, 1차 및 섀도우 탭핑-지연 라인들(78(0), 78(1))의 입력측에 제공되어, 벡터 데이터 파일들(82(0)-82(X))로부터 패치된 입력 벡터 데이터 샘플 세트들(86(0)-86(X))은, 1차 및 섀도우 탭핑-지연 라인들(78(0), 78(1))로 로딩되기 전에 포맷 변환 회로(159(0)-159(X))에서 포맷 변환될 수 있다. 이 예에서, 입력 벡터 데이터 샘플 세트들(86(0)-86(X))은, 포맷-변환된 입력 벡터 데이터 샘플 세트들(86F(0)-86F(X))(또는 시프팅 이후 86SF(0)-86SF(X))로서 1차 및 섀도우 탭핑-지연 라인들(78(0), 78(1))에 저장될 것이다. 다음, 포맷-변환된 입력 벡터 데이터 샘플 세트들(86F(0)-86F(X))(또는 시프팅 이후 86SF(0)-86SF(X))은 직접적으로 1차 탭핑-지연 라인(78(0))으로부터 벡터 프로세싱 연산에서의 실행을 위해 직접적으로 실행 유닛들(84(0)-84(X))에 제공될 수 있다.

[0143] 앞서 논의된 바와 같이, 실행될 벡터 명령에 따라 포맷 변환 회로(159(0)-159(X))를 이용하기 위해, 입력 데이터 흐름 경로들(80(0)-80(X))은 프로그래밍가능한 입력 데이터 경로 구성에 따라 프로그래밍될 수 있다. 이와 관련하여, 도 22는, 도 19의 VPE(22(3))에서 입력 벡터 데이터 샘플 세트들(86(0)-86(X))의 시프팅 및 포맷 변환의 프로그래밍을 제어하기 위한 벡터 명령의 비트들의 예시적인 데이터 포맷을 제공하는 차트(182)이다. 차트(182)의 필드들에 제공되는 데이터는, 프로세싱될 벡터 명령에 대해 이들의 기능이 필요한지 여부에 따라, 입력 데이터 흐름 경로들(80(0)-80(X))에 포맷 변환 회로(159(0)-159(X)) 및/또는 탭핑-지연 라인들(78)이 포함되는지 여부를 제어하기 위한 프로그래밍을 VPE(22(3))에 제공한다.

[0144] 예컨대, 도 22에서, 탭핑-지연 라인들(78)에 의해 부호화된 복수 십육(16) 비트 포맷(SC16)을 이용하는 경우 산술 명령들에 대한 시프트 바이어스가 제공되는지 여부를 표시하기 위해, 벡터 명령 또는 벡터 프로그래밍의 비트들 [7:0]에 바이어스 필드(184)(BIAS_SC16)가 제공된다. 제 1 소스 데이터(즉, 입력 벡터 데이터 샘플 세트(86(0)-86(X))가 데시메이트(즉, 디인터리빙)되고 SC8로부터 SC16 포맷으로 변환되어야 하는지 여부를 표시하기 위해, 벡터 명령 또는 벡터 프로그래밍의 비트 [16]에 제 1 소스 데이터 포맷 변환 필드(186)(DECIMATE_SRC1)가 제공된다. 제 2 소스 데이터(즉, 입력 벡터 데이터 샘플 세트(86(0)-86(X))가 데시메이트(즉, 디인터리빙)되고 SC8로부터 SC16 포맷으로 변환되어야 하는지 여부를 표시하기 위해, 벡터 명령 또는 벡터 프로그래밍의 비트 [17]에 제 2 소스 데이터 포맷 변환 필드(188)(DECIMATE_SRC2)가 제공된다. 출력 소스 데이터(예컨대, 도 19의 VPE(22(3))에서 결과적 출력 벡터 데이터 샘플 세트(172(0)-172(X)))가 벡터 데이터 파일들(82(0)-82(X))에 저장되는 경우 SC16 포맷으로 저장되어야 하는지 또는 SC16으로부터 SC8 포맷으로 변환되고 재정렬되어야 하는지 여부를 표시하기 위해, 비트 [18]에 출력 데이터 포맷 필드(190)(DEST_FMT)가 제공된다. 입력 소스 데이터(즉, 입력 벡터 데이터 샘플 세트(86(0)-86(X))) 및 출력 데이터(예컨대, 도 19의 VPE(22(3))에서 결과적 출력 벡터 데이터 샘플 세트(172(0)-172(X)))가 짝수(예컨대, 정시) 및 홀수(예컨대, 늦음) 샘플들을 따라 데시메이트(즉,

디인터리빙)되어야 하는지 여부를 표시하기 위해, 비트 [19]에 위상 포맷 필드(192)(DECIMATE_PHASE)가 제공되고, 이는, 앞서 그리고 도 17b에서 이미 설명된 바와 같이, 특히 CDMA-특정 벡터 프로세싱 연산들에 대해 유용할 수 있다.

[0145] 앞서 논의된 바와 같이, VPE들(22)의 실행 유닛들(84(0)-84(X))이 입력 벡터 데이터 샘플 세트들에 대해 벡터 프로세싱을 수행하고, 결과로서 결과적 출력 벡터 데이터 샘플 세트들을 출력 데이터 흐름 경로들(98(0)-98(X)) 상에 제공한 후, 결과적 출력 벡터 데이터 샘플 세트들에 대해 후속적인 벡터 프로세싱 연산들이 수행될 필요가 있을 수 있다. 그러나, 결과적 출력 벡터 데이터 샘플 세트들은, 후속적인 벡터 프로세싱 연산들을 위해 재정렬될 필요가 있을 수 있다. 따라서, 이전의 프로세싱 연산들로부터 도출된 결과적 출력 벡터 데이터 샘플 세트들은, 벡터 데이터 파일들(82(0)-82(X))에 저장되고, 재정렬을 위해 패치되고, 재정렬된 포맷으로 벡터 데이터 파일들(82(0)-82(X))에 재저장되어야 한다. 예컨대, 도 17a 및 도 17b에서 앞서 논의된 바와 같이, 후속적인 프로세싱 연산들은, 이전에 프로세싱된 벡터 데이터 샘플들이 벡터 데이터 파일들(82(0)-82(X))에 저장되는 경우 인터리빙되도록 요구할 수 있다.

[0146] 다른 예로서, 후속 프로세싱 연산들은, 벡터 데이터 파일들(82(0)-82(X))에 저장될 때 이전에 프로세싱된 벡터 데이터 샘플들이 디인터리빙될 것을 요구할 수 있다. 예를 들어, CDMA 프로세싱 연산들에서, 신호를 나타내는 데이터 샘플들이 신호의 홀수(정시) 위상 및 홀수(늦음) 위상에 따라 저장되고 인터리빙될 필요가 있을 수 있다. 이 문제를 해결하기 위해서, 벡터 프로세서는 출력 벡터 데이터가 벡터 데이터 메모리에 저장된 후에 실행 유닛들로부터의 출력 벡터 데이터의 사후-프로세싱 재정렬을 수행하는 회로를 포함할 수 있다. 벡터 데이터 메모리에 저장된 사후-프로세싱된 출력 벡터 데이터 샘플들이 벡터 데이터 메모리로부터 패치되고, 재정렬되어, 벡터 데이터 메모리에 다시 저장된다. 이 사후-프로세싱은 실행 유닛들에 의해 재정렬된 벡터 데이터 샘플들의 후속 프로세싱을 지연시키고, 실행 유닛의 계산 컴포넌트로 하여금 제대로 활용되지 못하게 한다.

[0147] 이와 관련하여, 도 23은, 도 2의 VPE(22)로서 제공될 수 있는 다른 예시적인 VPE(22(4))의 개략도이다. 이하에서 보다 상세히 설명되는 바와 같이, 도 23의 VPE(22(4))는 VPE(22(4)) 내의 벡터 데이터 파일들(82(0)-82(X))에 저장될 벡터 프로세싱 연산들을 위해 실행 유닛들(84(0)-84(X))에 의해 제공되는 결과적 출력 벡터 데이터 샘플 세트들(194(0)-194(X))의 실시간 재정렬을 제공하도록 구성되어, 벡터 데이터 샘플 재패치가 제거되거나 또는 감소되고 전력 소비가 감소된다. 결과적 출력 벡터 데이터 샘플 세트(194(0)-194(X))는, 이 예에서, 194(0), 194(1),..., 및 194(X)인 'X+1'의 결과적 출력 벡터 데이터 샘플들(194)로 구성되어 있다. 예를 들어, 재정렬은 결과적 출력 벡터 데이터 샘플 세트들(194(0)-194(X))이 벡터 데이터 파일들(82(0)-82(X))에 저장되기 전에 그 결과적 출력 벡터 데이터 샘플 세트들(194(0)-194(X))을 인터리빙하는 것을 포함한다.

[0148] 도 23에 도시되고 아래에서 보다 구체적으로 설명되는 바와 같이, 재정렬 회로(196(0)-196(X))가, 벡터 데이터 레인들(100(0)-100(X)) 각각의 실행 유닛들(84(0)-84(X))과 벡터 데이터 파일들(82(0)-82(X)) 사이의 출력 데이터 흐름 경로들(98(0)-98(X))에 제공된다. 재정렬 회로(196(0)-196(X))는, 결과적 출력 벡터 데이터 샘플 세트(194(0)-194(0))의 재정렬을, 출력 데이터 흐름 경로들(98(0)-98(X))의 재정렬된 결과적 출력 벡터 데이터 샘플 세트(194R(0)194R(X))로서 제공하기 위해 실행될 벡터 명령에 따라 프로그래밍에 기초하여 구성된다. 도 23의 VPE 22(4)의 결과적 출력 벡터 데이터 샘플 세트(194(0)-194(X))의 실시간 재정렬은, 실행 유닛들(84(0)-84(X))에 의해 제공되는 결과적 출력 벡터 데이터 샘플 세트(194(0)-194(X))가 벡터 데이터 파일들(82(0)-82(X))에 저장되기 전에 재정렬된 결과적 출력 벡터 데이터 샘플 세트(194R(0)-194R(X))로서 재정렬되는 것을 의미한다. 이러한 방식으로, 결과적 출력 벡터 데이터 샘플 세트(194(0)-194(X))는 재정렬된 결과적 출력 벡터 데이터 샘플 세트(194R(0)-194R(X))로서 재정렬된 포맷으로 벡터 데이터 파일들(82(0)-82(X))에 저장된다. 비제한적인 예로서, 결과적 출력 벡터 데이터 샘플 세트들(194(0)-194(X))의 재정렬은 재정렬된 결과적 출력 벡터 데이터 샘플 세트들(194R(0)-194R(X))로서 벡터 데이터 파일들(82(0)-82(X))에 저장될 결과적 출력 벡터 데이터 샘플 세트들(194(0)-194(X))의 인터리빙 또는 디인터리빙을 포함할 수 있다.

[0149] 따라서, 출력 데이터 흐름 경로들(98(0)-98(X))에 제공된 재정렬 회로(196(0)-196(X))를 이용하여, 결과적 출력 벡터 데이터 샘플 세트(194(0)-194(X))는, 벡터 데이터 파일들(82(0)-82(X))에 먼저 저장될 필요가 없으며, 이후, 벡터 데이터 파일들(82(0)-82(X))로부터 패치되고, 재정렬되고, 벡터 데이터 파일들(82(0)-82(X))에서 복원된다. 결과적 출력 벡터 데이터 샘플 세트(194(0)-194(X))가, 벡터 데이터 파일들(82(0)-82(X))에 저장되기 전에 재정렬된다. 이러한 방식으로, 결과적 출력 벡터 데이터 샘플 세트들(194(0)-194(X))은 실행 유닛들(84(0)-84(X))에서 수행될 후속 벡터 프로세싱 연산들을 지연시킬 수 있는 추가 사후-프로세싱 단계들을 요구하지 않고 벡터 데이터 파일들(82(0)-82(X))에서 재정렬된 포맷으로 저장된다. 따라서, VPE(22(4)) 내의 데이터 흐름 경로의 효율이 결과적 출력 벡터 데이터 샘플 세트들(194(0)-194(X))의 재정렬에 의해 제한되지 않는다. 결과적

출력 벡터 데이터 샘플 세트들(194(0)-194(X))이 벡터 데이터 파일들(82(0)-82(X))의 재정렬된 결과적 출력 벡터 데이터 샘플 세트들(194R(0)-194R(X))로서 재정렬된 포맷으로 저장될 경우, 실행 유닛들(84(0)-84(X))의 후속하는 벡터 프로세싱이 데이터 흐름 제한들에 의해서 보다는 계산 리소스들에 의해 제한될 뿐이다.

[0150] 도 23에 도시된 바와 같이 이 예에서, 재정렬 회로(196(0)-196(X))를 포함하는 VPE(22(4))는 또한, 1차 탭핑-지연 라인(78(0)) 및/또는 섀도우 탭핑-지연 라인(78(1))을 선택적으로 포함할 수 있다. 탭핑-지연 라인들(78(0), 78(1))의 동작은 VPE들(22(1) 및 22(2))과 관련하여 앞에서 상술되었다. 앞서 상술된 바와 같이, 탭핑-지연 라인들(78(0), 78(1))은 실행 유닛들(84(0)-84(X))로 제공될 시프트된 입력 벡터 데이터 샘플 세트들(86S(0)-86S(X))을 필요로 하는 벡터 프로세싱 연산을 위해 사용될 수 있다. 또한, 공통 컴포넌트들이, 도 4, 도 11 및 도 19의 VPE들(22(1)-22(3))에 제공되는 도 23의 VPE(22(4))에 제공된다는 것을 주목한다. 공통 컴포넌트들이 공통 엘리먼트 번호들을 이용하여 도 23의 VPE(22(4))에 도시된다. VPE들(22(1)-22(3))에 대하여 상술된 이러한 공통 컴포넌트들의 앞의 설명과 논의는 또한 도 23의 VPE(22(4))에 적용가능하고 따라서 여기서 다시 설명하지 않을 것이다.

[0151] 도 23을 계속해서 참조하면, 보다 구체적으로, 재정렬 회로(196(0)-196(X))는 출력 데이터 흐름 경로들(98(0)-98(X)) 상의 재정렬 회로 입력들(198(0)-198(X))을 통해 결과적 출력 벡터 데이터 샘플 세트들(194(0)-194(X))을 수신하도록 구성된다. 재정렬 회로(196(0)-196(X))는 재정렬된 결과적 출력 벡터 데이터 샘플 세트들(194R(0)-194R(X))을 제공하기 위해서 결과적 출력 벡터 데이터 샘플 세트들(194(0)-194(X))을 재정렬하도록 구성된다. 재정렬 회로(196(0)-196(X))는 저장을 위해서 벡터 데이터 파일들(82(0)-82(X))로 제공될 출력 데이터 흐름 경로들(98(0)-98(X))의 재정렬 회로 출력들(200(0)-200(X))을 통해 재정렬된 결과적 출력 벡터 데이터 샘플 세트들(194R(0)-194R(X))을 제공하도록 구성된다.

[0152] 본 실시예에서 재정렬된 결과적 출력 벡터 데이터 샘플 세트들(194R(0)-194R(X))을 출력 데이터 흐름 경로들(98(0)-98(X))에서 벡터 데이터 파일들(82(0)-82(X))로 제공하는 것에 대한 도 23의 VPE(22(4))의 부가적인 상세들 및 피처들의 추가 설명이 이제 설명될 것이다. 이와 관련하여, 도 24는 결과적 출력 벡터 데이터 샘플 세트(194(0)-194(X))의 재정렬을 필요로 하는 예시적인 벡터 명령에 따라 재정렬 회로(196(0)-196(X))를 사용하는 도 23의 VPE22(4)에서 수행될 수 있는 벡터 프로세싱 연산(202)으로부터 비롯된 결과적 출력 벡터 데이터 샘플 세트들(194(0)-194(X))의 예시적인 재정렬을 도시하는 흐름도이다.

[0153] 도 23 및 도 24를 참조하면, 벡터 명령에 따른 벡터 프로세싱 연산(202)에 따라 프로세싱될 입력 벡터 데이터 샘플 세트(86(0)-86(X))가 벡터 데이터 파일들(82(0)-82(X))로부터 패치되고 입력 데이터 흐름 경로들(80(0)-80(X))에 제공된다(도 24의 블록(204)). 예를 들어, 벡터 프로세싱 동작(202)은, 실행되는 벡터 명령에 따라 요구되는 임의의 벡터 프로세싱 연산을 포함할 수 있다. 필터, 상관, 포맷 변환 벡터 프로세싱의 연산들을 포함하는 비제한적 예들이 상술되었다. 도 23의 VPE(22(4))의 벡터 데이터 레인들(100(0)-100(X)) 중 하나, 일부, 또는 전부가 벡터 프로세싱 연산(202)을 위한 입력 벡터 데이터 샘플 세트(86(0)-86(X))의 폭에 의존하여 벡터 명령의 프로그래밍에 따른 벡터 프로세싱 연산(202)을 제공하기 위해 사용될 수 있다. 벡터 데이터 파일들(82(0)-82(X))의 전체 폭이 필요한 경우, 모든 벡터 데이터 레인들(100(0)-100(X))은 벡터 프로세싱 연산(202)을 위해 사용될 수 있다. 벡터 프로세싱 연산(202)은 단지 벡터 데이터 레인들(100(0)-100(X))의 서브세트만을 요구할 수 있다. 이는, 입력 벡터 데이터 샘플 세트(86(0)-86(X))의 폭이 모든 벡터 데이터 파일들(82(0)-82(X))의 폭보다 좁기 때문인데, 벡터 프로세싱 연산(202)과 병렬로 수행되는 다른 벡터 프로세싱 연산들을 위해 추가 벡터 데이터 레인들(100)을 사용하는 것이 바람직한 경우이다.

[0154] 도 23 및 도 24를 계속해서 참조하면, 패치된 입력 벡터 데이터 샘플 세트(86(0)-86(X))가 실행 유닛들(84(0)-84(X))에서 입력 데이터 흐름 경로들(80(0)-80(X))로부터 수신된다(도 24의 블록 206). 실행 유닛(84(0)-84(X))은 벡터 명령에 따라 제공되는 벡터 프로세싱 연산(202)에 따라서 수신된 입력 벡터 데이터 샘플 세트(86(0)-86(X))에 대해 벡터 프로세싱을 수행한다(도 24의 블록208). 비제한적 예로서, 입력 벡터 데이터 샘플(86(0)-86(X))은 선택적으로, 입력 벡터 데이터 샘플 세트(86(0)-86(X))의 시프팅을 포함하는 실행 유닛들(84(0)-84(X))에 의해 실행되는 벡터 프로세싱 연산(202)의 각각의 프로세싱 스테이지 동안에 벡터 프로세싱 연산(202)의 실행 동안 시프트될 입력 벡터 데이터 샘플 세트(86(0)-86(X))로서 1차 탭핑-지연 라인(78(0))으로 로딩된다. 상술한 바와 같이, 다음 입력 벡터 데이터 샘플 세트(86N(0)-86N(X))는 또한, 실행 유닛들(84(0)-84(X))에 의해 프로세싱될 다음 입력 벡터 데이터 샘플 세트(86N(0)-86N(X))로서 섀도우 탭핑-지연 라인(78(1))으로 선택적으로 로딩될 수 있다. 앞서 상술한 바와 같이, 탭핑-지연 라인들(78)의 목적은 시프트된 입력 벡터 데이터 샘플들(86S)에 대해 동작하는 벡터 프로세싱 연산(202)의 연산 동안 실행 유닛들(84(0)-84(X))로 제공될 시프트된 입력 벡터 데이터 샘플들(86S(0)-86S(X))로 입력 벡터 데이터 샘플 세트(86(0)-86(X))를 시프

트시키는 것이다.

[0155] 도 23 및 도 24를 계속해서 참조하면, 실행 유닛들((84(0)-84(X))은 입력 벡터 데이터의 샘플 세트(86(0)-86(X))를 이용한 곱셈 및/또는 누산을 제공하도록 구성될 수 있다. 탭핑-지연 라인들(78)이 벡터 프로세싱 연산(202) 동안 포맷-변환 입력 벡터 데이터 샘플 세트(86F(0)-86F(X))를 시프트시키기 위해 사용되는 경우, 실행 유닛들(84(0)-84(X))은, 예로서 상술된 바와 같이, 벡터 프로세싱 동작(202)이 완료될 때까지, 벡터 프로세싱 연산(202)의 각각의 프로세싱 스테이지 동안 시프트된 입력 벡터 데이터 샘플 세트(86S(0)-86S(X))를 수신할 수 있다. 벡터 프로세싱의 연산(202)이 완료되면, 입력 벡터 데이터 샘플 세트(86(0)-86(X))의 벡터 프로세싱, 또는 시프트된, 포맷-변환된 입력 벡터 데이터 샘플 세트들(86S(0)-86S(X))에 기초하여 결과적 출력 벡터 데이터 샘플 세트(194(0)-194(X))가 출력 데이터 흐름 경로들(98(0)-98(X))에서 제공된다.

[0156] 도 23 및 도 24를 계속해서 참조하면, 결과적 출력 벡터 데이터 샘플 세트(194(0)-194(X))가 벡터 데이터 파일들(82(0)-82(X))에 저장되기 전에, 그 결과적 출력 벡터 데이터 샘플 세트(194(0)-194(X))는 실행 유닛들(84(0)-84(X))과 벡터 데이터 파일들(82(0)-82(X)) 사이에 제공되는 출력 데이터 흐름 경로들(98(0)-98(X))에 제공되는 재정렬 회로(196(0)-196(X))로 제공된다. 재정렬 회로(196(0)-196(X))는, 아래에서 보다 구체적으로 설명되는 바와 같이, 벡터 명령이 벡터 데이터 파일들(82(0)-82(X))에 저장될 결과적 출력 벡터 데이터 샘플 세트(194(0)-194(X))의 재정렬을 요구하는 경우, 실행되는 벡터 명령에 따라 출력 데이터 흐름 경로들(98(0)-98(X))에 포함되도록 프로그래밍가능하다. 재정렬 회로(196(0)-196(X))는, 결과적 출력 벡터 데이터 샘플 세트(194(0)-194(X))가 벡터 데이터 파일들(82(0)-82(X))에 저장되지 않고 실행되는 벡터 명령에 따른 프로그래밍에서 제공되는 재정렬에 따라 결과적 출력 벡터 데이터 샘플 세트(194(0)-194(X))를 재정렬한다(도 24의 블록(210)). 이러한 방식으로, 결과적 출력 벡터 데이터 샘플 세트(194(0)-194(X))는 벡터 데이터 파일들(82(0)-82(X))에 먼저 저장되고, 재패치되고, 사후-프로세싱 연산에서 재정렬되고 벡터 데이터 파일들(82(0)-82(X)) 내에 재정렬된 포맷으로 저장될 필요(이들은 실행 유닛들(84(0)-84(X))에서 지연을 제공함)가 없다. 결과적 출력 벡터 데이터 샘플 세트(194(0)-194(X))가, 재정렬 사후-프로세싱이 필요없이, 벡터 데이터 파일들(82(0)-82(X)) 내에서 재정렬된 결과적 출력 벡터 데이터 샘플 세트(194R(0)-194R(X))로서 저장된다(도 24의 블록(212)). 예를 들어, 결과적 출력 벡터 데이터 샘플 세트(194(0)-194(X))는 재정렬 회로(196(0)-196(X))에 의해 재정렬되기 전에 도 18a 및 도 18b에 제공된 것과 같은 포맷으로 나타날 수 있다.

[0157] 재정렬 회로(196(0)-196(X))의 예를 이제, 도 25와 관련하여 설명할 것이다. 재정렬 회로(196(0)-196(X))의 내부 컴포넌트들의 예시적인 상세가, 도 25에 제공되는 벡터 데이터 레인(100(0))에 제공되는 재정렬 회로(196(0))의 일 경우에 대해 제공되지만, 이러한 것은 재정렬 회로(196(1)-196(X))에도 또한 적용가능하다. 도 25의 재정렬 회로(196(0))를 예로서 택하면, 본 예의 재정렬 회로(196(0))는, 재정렬된 결과적 출력 벡터 데이터 샘플(194R(0))을 제공하기 위해서 벡터 데이터 레인(100(0))의 출력 데이터 흐름 경로(98(0))의 실행 유닛(84(0))에 의해 제공되는 결과적 출력 벡터 데이터 샘플(194(0))을 재정렬하도록 구성된다. 이와 관련하여, 이 예에서 멀티플렉서들의 형태로 제공되는 4개의 출력 벡터 데이터 샘플 선택기들(214(3)-214(0))이 이 예에서 제공되는데, 그것들은 이 예에서 각각이 팔(8) 비트 폭들인 96(0)(3)-96(0)(0))로 이루어진 네개(4)의 실행 유닛 출력들(96(0))의 비트 폭들에 따라 배열된다. 각각의 출력 벡터 데이터 샘플 선택기(214(3)-214(0))는 할당된 실행 유닛 출력(96(0)(3)-96(0)(0))의 결과적 출력 벡터 데이터 샘플(194(0))의 부분, 또는 할당된 실행 유닛 출력(96(0)(3)-96(0)(0))과 인접한 실행 유닛 출력(96)으로부터 결과적으로 시프트되어 출력된 벡터 데이터 샘플(194(0))의 부분을 선택하도록 구성된다.

[0158] 예를 들어, 실행 유닛 출력들(96(0)(3)-96(0)(0))이 십육(16) 비트 할당 복소 포맷 실수[31:24], 실수[23:16], 허수[15:8], 허수[7:0]로 결과적 출력 벡터 데이터 샘플(194(0))을 제공하고 원하는 재정렬된(예를 들어, 인터리빙된) 포맷이 실행될 벡터 명령에 따라 실수[31:24], 허수[23:16], 실수[15:8], 허수[7:7]인 경우, 출력 벡터 데이터 샘플 선택기(214(3)-214(0))의 선택들은 다음과 같을 것이다. 출력 벡터 데이터 샘플 선택기((214)(3))는 출력 데이터 흐름 경로(98(0)(3))를 통한 제공을 위해서 실행 유닛 출력(96(0)(3))으로부터의 결과적 출력 벡터 데이터 샘플(194(0)(3))을 선택할 것이다. 그러나, 출력 벡터 데이터 샘플 선택기(214)(2))는 출력 데이터 흐름 경로(98(0)(2))를 통한 제공을 위해서 실행 유닛 출력부(96(0)(1)) 상의 결과적 출력 벡터 데이터 샘플(194(0)(1))의 부분을 선택할 것이다. 이는, 재정렬된 결과적 출력 벡터 데이터 샘플(194R(0))의 재정렬된 결과적 출력 벡터 데이터 샘플(194R(0)(3), 194R(0)(2))로서, 인접한 출력 데이터 흐름 경로들(98(0)(3), 98(0)(2)) 내의 결과적으로 시프트된 출력 벡터 데이터 샘플(194(0))(즉, 실수[31:24], 허수[23:16])의 인터리빙된 실수부를 제공한다. 유사하게, 출력 벡터 데이터 샘플 선택기((214)(0))는 출력 데이터 흐름 경로(98(0)(0))를 통한 제공을 위해서 실행 유닛 출력(96(0)(0))으로부터의 결과적 출력 벡터 데이터 샘플

(194(0)(0))을 선택할 것이다. 그러나, 출력 벡터 데이터 샘플 선택기((214)(1))는 출력 데이터 흐름 경로(98(0)(1))를 통한 제공을 위해서 실행 유닛 출력(96(0)(2))으로부터의 결과적 출력 벡터 데이터 샘플(194(0)(2))을 선택할 것이다. 이는, 재정렬된 결과적 출력 벡터 데이터 샘플(194R(0))의 재정렬된 결과적 출력 벡터 데이터 샘플(194R(0)(1), 194R(0)(0))로서, 인접한 출력 데이터 흐름 경로들(98(0)(1), 98(0)(0)) 내의 재정렬된, 인터리빙된 결과적 출력 벡터 데이터 샘플들(194(0)(2),194(0)(0))(즉, 실수[15:8], 허수[7:0])을 제공한다. 멀티플렉서들의 형태로도 또한 제공되는 출력 벡터 데이터 샘플 선택기(216(1), 216(0))는 또한, 도 25에 도시된 바와 같이, 할당되지 않고, 인접하지 않은 실행 유닛 출력부(96(0)(3)-96(0)(0))로부터의 결과적 출력 벡터 데이터 샘플(194(0)(3)-194(0)(0)) 사이에서 선택할 능력을 제공한다.

[0159] 도 23 및 도 25를 계속해서 참조하면, 재정렬 회로((196)(0)-196(X))는 실행되는 벡터 명령에 따라 결과적 출력 벡터 데이터 샘플 세트(194(0)-194(X))를 재정렬하지 않도록 구성되거나 또는 재구성되는 프로그래밍 가능한 것으로서 제공될 수 있다. 이 예에서, 재정렬 회로(196(0)-196(X))는, 어떠한 재정렬 동작들도 형성되지 않은 상태로 재정렬 회로(196(0)-196(X))로 직선으로 흐르도록 출력 데이터 흐름 경로들(98(0)-98(X))에 제공되도록 프로그래밍될 수 있다. 앞서 상술되고 도 22에 도시된 바와 같이, 차트(182)의 출력 데이터 포맷 필드(190)(DEST_FMT)는, 출력 소스 데이터(예를 들어, 도 23의 VPE22(4)에서 결과적 출력 벡터 데이터 샘플 세트(194(0)-194(X)))가 벡터 데이터 파일들(82(0)-82(X))에 저장될 경우, SC16 포맷으로 저장되거나 SC16으로부터 SC8 포맷으로 변환되고 재정렬되어야 하는지를 나타내기 위해서 비제한적인 예로서 벡터 명령의 비트[18]로 제공될 수 있다.

[0160] 이와 관련하여, 도 25의 프로그램가능 재정렬 데이터 경로 구성 입력 유닛(218(0))이, 출력 데이터 흐름 경로(98(0))의 결과적 출력 벡터 데이터 샘플들(194(0)(3)-194(0)(0))을 재정렬하거나 또는 재정렬하지 않도록 재정렬 회로(196(0))를 프로그래밍하기 위해서 재정렬 회로(196(0))로 제공될 수 있다. 프로그램가능 재정렬 데이터 경로 구성 입력부들(218(1)-218(X)(도시되지 않음)도 또한 유사하게, 각각, 출력 데이터 흐름 경로들(98(1)-98(X))의 결과적 출력 벡터 데이터 샘플 세트들(194(1)-194(X))을 재정렬하거나 또는 재정렬하지 않도록 재정렬 회로(196(1)-196(X))를 프로그래밍하기 위해서 재정렬 회로(196(1)-196(X))로 제공될 수 있다. 이러한 방식으로, 재정렬 회로(196(0)-196(X))는, 수행될 이러한 프로세싱에 벡터 명령이 제공되지 않는 경우, 결과적 출력 벡터 데이터 샘플 세트들(194(0)-194(X))을 재정렬하지 않도록 프로그래밍될 수 있다. 프로그램가능 재정렬 데이터 경로 구성 입력들(218(0)-218(X))은, 각각의 벡터 명령이 VPE(22(4))에 의한 벡터 프로세싱의 유연성을 제공하도록 구성 및 재구성될 수 있다. 예를 들어, 프로그램가능 재정렬 데이터 경로 구성 입력부들(218(0)-218(X))은, 필요한 경우, 실행 유닛들(84(0)-84(X))을 전적으로 활용하여, 원하는 대로 재정렬을 제공하기 위해서, 필요한 경우, 클록-사이클-바이-클록-사이클 기반으로, 벡터 명령의 각각의 클록-사이클에 대해 구성 및 재구성될 수 있다.

[0161] 다른 벡터 프로세싱 연산들은 또한, 추가 사후-프로세싱 단계들을 필요로하지 않고 실행 유닛들(84(0)-84(X))로부터의 결과적 출력 벡터 데이터 샘플 세트들의 실시간 프로세싱을 포함하여 제공될 수 있으며, 이는 실행 유닛들(84(0)-84(X))에서 수행될 후속 벡터 프로세싱 연산들을 지연시킬 수 있다. 예를 들어, 다양한 길이의 확산 신호 데이터 시퀀스들에 따른 칩 시퀀스들의 역확산을 요구하는 CDMA 무선 기저대역 동작들은 실시간 벡터 프로세싱으로부터 이익을 얻을 수 있다.

[0162] 예를 들어, CDMA를 이용하여 변조될 수 있는 데이터 신호(220)는 도 26의 a에 도시되어 있다. 데이터 신호(220)는 2T의 기간을 갖는다. 데이터 신호(220)는 이 예에서 데이터 시퀀스(1010)를 나타내며, 도 26의 a에 나타내어진 바와 같이, 하이 신호 레벨들은 논리 '1'을 나타내고, 로우 신호 레벨들은 논리 '0'을 나타낸다. CDMA 변조에서, 데이터 신호(220)는, 의사랜덤 코드일 수 있는 도 26의 b의 칩 시퀀스(222)와 같은 칩 시퀀스(222)에 의해 확산된다. 이 예에서, 칩 시퀀스(222)는, 이 예에서 데이터 신호(220)의 각각의 샘플에 대해 열개(10)의 칩들의 확산 레이트 또는 팩터를 갖는 칩 시퀀스(222)를 제공하기 위해서, 데이터 신호(220)의 기간보다 열(10)배 더 짧은 기간을 갖는다. 이 예의 데이터 신호(220)를 확산시키기 위해서, 데이터 신호(220)는, 도 26의 c에 도시된 바와 같이, 확산 송신된 데이터 신호(224)를 제공하기 위해서 칩 시퀀스(222)에 따라 배타적 OR이 된다(즉, XOR이 된다). 확산 송신된 데이터 신호(224)를 갖는 동일한 대역폭에서 송신되는 다른 사용자들을 위한 다른 데이터 신호들은, 서로 직교하는 다른 칩 시퀀스들 및 칩 시퀀스(222)로 확산된다. 이러한 방식으로, 원래의 데이터 신호(220)가 복원될 경우, 확산 송신된 데이터 신호(224)가, 도 11 내지 도 16과 관련하여 앞서 상술된 바와 같이, 시퀀스 번호들과 상관된다. 시퀀스 번호와 확산 송신된 데이터 신호(224) 사이에 높은 상관이 존재하는 경우, 칩 시퀀스(222)의 경우와 같이, 원래 데이터 신호(220)는 높은 상관 시퀀스 번호와 연관된 칩 시퀀스를 이용하여 복원될 수 있다. 확산 송신된 데이터 신호(224)는, 도 26의 d의 복원된 데이터 신호

(226)로서 원래 데이터 신호(220)를 복원하기 위해서, 본 실시예에서 칩 시퀀스(222)인 고도로 상관된 칩 시퀀스로 역확산된다.

[0163] 도 26의 c의 확산 송신된 데이터 신호(224)의 역확산은, 매우 상관적인 칩 시퀀스를 결정하기 위해, 도 11의 VPE(22(2))에 대해 위에서 설명된 상관 벡터 프로세싱 연산과 유사하게, 확산 송신된 데이터 신호(224)와 잠재적인 칩 시퀀스들 사이의 내적으로서 역확산 벡터 프로세싱 연산에서 수행될 수 있다. 확산 송신된 데이터 신호(224)는, 도 26의 d에서 복원된 데이터 신호(226)를 제공하기 위해, 본래의 데이터 신호(220)를 CDMA 변조하는데 사용되도록 결정되는 칩 시퀀스(222)를 이용하여 역확산될 수 있다.

[0164] CDMA 프로세싱 연산들을 포함하는 벡터 프로세서들에서, 벡터 프로세서들은, 실행 유닛들로부터 출력되고 벡터 데이터 메모리에 저장된 이후 확산 신호 벡터 데이터 시퀀스들의 역확산을 수행하는 회로를 포함할 수 있다. 이와 관련하여, 벡터 데이터 메모리에 저장된 확산 신호 벡터 데이터 시퀀스들은, 사후-프로세싱 연산에서 벡터 데이터 메모리로부터 패치되며, 본래의 데이터 신호를 복원하기 위해, 상관된 확산 코드 시퀀스 또는 칩 시퀀스로 역확산한다. 확산 이전의 본래의 데이터 샘플들인 역확산된 벡터 데이터 시퀀스들은 벡터 데이터 메모리에 다시 저장된다. 사후-프로세싱 연산은, 실행 유닛들에 의한 후속 벡터 연산 프로세싱을 지연시키며, 실행 유닛들 내의 계산 컴포넌트들이 충분히 이용되지 않게 할 수 있다. 추가적으로, 확산 코드 시퀀스를 사용하는 확산 신호 벡터 데이터 시퀀스들의 역확산은, 역확산될 확산 신호 벡터 데이터 시퀀스들이 실행 유닛들로부터 상이한 데이터 흐름 경로들에 걸쳐 크로스하므로, 병렬화시키기에 어렵다.

[0165] 이러한 문제를 해결하기 위해, 아래에 기재된 실시예들에서, VPE 내의 실행 유닛들과 벡터 데이터 메모리 사이의 데이터 흐름 경로들에 제공된 역확산 회로를 포함하는 VPE들이 제공된다. 역확산 회로는, 출력 벡터 데이터 샘플 세트가 실행 유닛들로부터 벡터 데이터 메모리로의 출력 데이터 흐름 경로들을 통해 제공되는 동안, 실시간으로 실행 유닛들로부터의 출력 벡터 데이터 샘플 세트를 사용하여 확산-스펙트럼 시퀀스들을 역확산하도록 구성된다. 출력 벡터 데이터 샘플 세트들의 실시간 역확산은, 출력 벡터 데이터 샘플 세트가 역확산 포맷으로 벡터 데이터 메모리에 저장되도록, 실행 유닛들에 의해 제공된 출력 벡터 데이터 샘플 세트가 벡터 데이터 메모리에 저장되기 전에 역확산된다는 것을 의미한다. 역확산 확산-스펙트럼 시퀀스들(DSSS)은, 부가적인 사후-프로세싱 단계들을 요구하지 않으면서 벡터 데이터 메모리에 역확산 포맷으로 저장될 수 있으며, 이는, 실행 유닛들에서 수행될 후속 벡터 프로세싱 연산들을 지연시킬 수있다. 따라서, VPE에서의 데이터 흐름 경로들의 효율은 확산-스펙트럼 시퀀스들의 역확산에 의해 제한되지 않을 수 있다. 실행 유닛들에서의 후속 벡터 프로세싱은, 역확산 확산-스펙트럼 시퀀스들이 벡터 데이터 메모리에 저장되는 경우, 데이터 흐름 제한들보다는 계산 리소스들에 의해서만 제한될 수있다.

[0166] 이와 관련하여, 도 27은, 도 2의 VPE(22)로서 제공될 수 있는 다른 예시적인 VPE(22(5))의 개략도이다. 아래에 더 상세히 설명될 바와 같이, 도 27의 VPE(22(5))는, 제거된 또는 감소된 벡터 데이터 샘플 재패치 및 감소된 전력 소비로, VPE(22(5)) 내의 벡터 데이터 파일들(82(0)-82(X))에 저장될 벡터 프로세싱 연산들을 위한 코드 시퀀스를 이용하여 실행 유닛들(84(0)-84(X))에 의해 제공된 결과적 출력 벡터 데이터 샘플 세트들(228(0)-228(X))의 실시간 역확산을 제공하도록 구성된다. 결과적 출력 벡터 데이터 샘플 세트들(228(0)-228(X))은, 이러한 예에서는 (228(0), 228(1), …, 및 228(X))인 'X+1' 입력 결과적 출력 벡터 데이터 샘플들(228)로 구성된다. 코드 시퀀스는 비제한적인 예로서, CDMA 역확산 벡터 프로세싱 연산을 위한 확산-스펙트럼 CDMA 칩 시퀀스일 수 있다. 도 27의 VPE(22(5))에서, 결과적 출력 벡터 데이터 샘플 세트(228(0)-228(X))는, 벡터 데이터 파일들(82(0)-82(X))에 저장되기 전에 코드 시퀀스를 이용하여 역확산될 수 있다.

[0167] 도 27에 도시되고 아래에서 더 상세히 논의되는 바와 같이, 역확산 회로(230)는, 벡터 데이터 레인들(100(0)-100(X)) 각각에서 실행 유닛들(84(0)-84(X))과 벡터 데이터 파일들(82(0)-82(X)) 사이의 출력 데이터 흐름 경로들(98(0)-98(X))에서 제공된다. 역확산 회로(230)는 상관 벡터 프로세싱 연산들에 대해 도 11-16에서 위에서 이전에 설명된 바와 같이, 벡터 명령에 따른 프로그래밍에 기초하여, 시퀀스 넘버 생성기(134)에 의해 생성된 기준 벡터 데이터 샘플 세트(130(0)-130(X))로서 제공된 코드 시퀀스를 이용한 결과적 출력 벡터 데이터 샘플 세트(228(0)-228(X))의 실시간 역확산을 제공하기 위해 실행되도록 구성된다. 역확산된 결과적 출력 벡터 데이터 샘플 세트(229(0)-229(Z))는, 출력 데이터 흐름 경로들(98(0)-98(X)) 내의 역확산 회로(230)에 의해 제공된다. 역확산된 결과적 출력 벡터 데이터 샘플 세트(229(0)-229(Z))는, 이러한 예에서는 (229(0), 229(1), …, 및 229(Z))인 'Z+1' 역확산된 결과적 출력 벡터 데이터 샘플들(229)로 구성된다. 도 27의 VPE(22(5))의 결과적 출력 벡터 데이터 샘플 세트(228(0)-228(X))의 실시간 역확산은, 실행 유닛들(84(0)-84(X))에 의해 제공된 결과적 출력 벡터 데이터 샘플 세트(228(0)-228(X))가 벡터 데이터 파일들(82(0)-82(X))에 저장되기 전에 결과적 벡터 데이터 샘플 세트(228(0)-228(X)) 내의 코드 시퀀스를 이용하여 역확산된다는 것을 의미한다. 이러한 방식

으로, 결과적 출력 벡터 데이터 샘플 세트(228(0)-228(X))는, 역확산된 결과적 출력 벡터 데이터 샘플 세트(229(0)-229(X))로서, 역확산된 포맷으로 벡터 데이터 파일들(82(0)-82(X))에 저장된다.

[0168] 따라서, 출력 데이터 흐름 경로들(98(0)-98(X))에서 제공된 역확산 회로(230)를 이용하면, 결과적 출력 벡터 데이터 샘플 세트(228(0)-228(X))는, 벡터 데이터 파일들(82(0)-82(X))에 먼저 저장되도록 요구되지 않으며, 그 후, 벡터 데이터 파일들(82(0)-82(X))로부터 패치되고, 역확산되며, 역확산된 포맷으로 벡터 데이터 파일들(82(0)-82(X))에 재저장된다. 결과적 출력 벡터 데이터 샘플 세트(228(0)-228(X))는, 벡터 데이터 파일들(82(0)-82(X))에 저장되기 전에 역확산된다. 이러한 방식으로, 역확산된 결과적 출력 벡터 데이터 샘플 세트(229(0)-229(Z))는, 부가적인 사후-프로세싱 단계들을 요구하지 않으면서, 벡터 데이터 파일들(82(0)-82(X))에 저장되며, 이는, 실행 유닛들(84(0)-84(X))에서 수행될 후속 벡터 프로세싱 연산들을 지연시킬 수있다. 따라서, VPE(22(5))에서의 데이터 흐름 경로들의 효율은 결과적 출력 벡터 데이터 샘플 세트(228(0)-228(X))의 역확산에 의해 제한되지 않는다. 실행 유닛들(84(0)-84(X))에서의 후속 벡터 프로세싱은, 결과적 출력 벡터 데이터 샘플 세트들(228(0)-228(X))이 벡터 데이터 파일들(82(0)-82(X))에 역확산된 결과적 출력 벡터 데이터 샘플 세트들(229(0)-229(Z))로서 역확산된 형태로 저장되는 경우, 데이터 흐름 제한들보다는 계산 리소스들에 의해서만 제한된다.

[0169] 추가적으로, 실행 유닛들(84(0)-84(X))과 벡터 데이터 파일들(82(0)-82(X)) 사이의 출력 데이터 흐름 경로들(98(0)-98(X))에 확산 회로(230)를 제공함으로써, 결과적 출력 벡터 데이터 샘플 세트(228(0)-228(X))는, 벡터 데이터 파일들(82(0)-82(X))과 실행 유닛들(84(0)-84(X)) 사이의 입력 데이터 흐름 경로들(80(0)-80(X))에서 벡터 데이터 레인들(100)을 크로스할 필요가 없다. 상이한 벡터 데이터 레인들(100) 사이에서 입력 벡터 데이터 샘플 세트(86(0)-86(X)) 내의 입력 벡터 데이터 샘플들(86)의 역확산을 위해 데이터 흐름 경로들을 제공하는 것은, 라우팅 복잡도들을 증가시킬 것이다. 결과로서, 실행 유닛들(84(0)-84(X))은, 역확산 동작들이 입력 데이터 흐름 경로들(80(0)-80(X))에서 수행되고 있는 동안 충분히 이용되지 않을 수 있다. 또한, 위에서 논의된 바와 같이, 입력 데이터 흐름 경로들(80(0)-80(X))에서의 결과적 출력 벡터 데이터 샘플 세트(228(0)-228(X))의 역확산은, 도 27의 VPE(22(5)) 내의 벡터 데이터 파일들(82(0)-82(X))에 먼저 저장되도록 결과적 출력 벡터 데이터 샘플 세트(228(0)-228(X))에게 요구할 것이며, 그에 의해, 재패치 및 확산되는 경우 전력 소비를 증가시키고 그리고/또는 역확산 연산들이 수행되고 있는 동안 지연될 수 있는 실행 유닛들(84(0)-84(X))의 충분치 못한 이용의 위험성을 갖게 한다.

[0170] 공통적인 컴포넌트들이 도 4, 11, 19, 및 23의 VPE들(22(1)-22(4))에서 제공되는 도 27의 VPE(22(5))에서 제공됨을 유의한다. 공통적인 컴포넌트들은, 공통적인 엘리먼트 넘버들을 가지면서 도 27의 VPE(22(5))에서 예시된다. VPE들(22(1)-22(4)) 내의 위의 이들 공통적인 컴포넌트들의 이전의 설명 및 논의는 또한, 도 27의 VPE(22(5))에 적용가능하며, 따라서, 여기서 다시 설명되지 않을 것이다.

[0171] 도 27을 계속 참조하면, 더 상세하게, 확산 회로(230)는, 출력 데이터 흐름 경로들(98(0)-98(X))을 통해 역확산 회로 입력들(232(0)-232(X)) 상의 결과적 출력 벡터 데이터 샘플 세트(228(0)-228(X))를 수신하도록 구성된다. 역확산 회로(230)는, 역확산된 결과적 출력 벡터 데이터 샘플 세트(229(0)-229(Z))를 제공하기 위해, 결과적 출력 벡터 데이터 샘플 세트(228(0)-228(X))를 역확산하도록 구성된다. 더 상세히 아래에서 논의되는 바와 같이, 역확산된 결과적 출력 벡터 데이터 샘플들(229)의 수는, 역확산된 결과적 출력 벡터 데이터 샘플 세트(229(0)-229(Z))에서 'Z+1'이다. 역확산된 결과적 출력 벡터 데이터 샘플 세트(229(0)-229(Z)) 내의 역확산된 결과적 출력 벡터 데이터 샘플들(229)의 수는, 결과적 출력 벡터 데이터 샘플 세트(228(0)-228(X))를 역확산하는데 사용되는 확산 팩터에 의존한다. 역확산 회로(230)는, 저장을 위해 벡터 데이터 파일들(82(0)-82(X))에 제공될 출력 데이터 흐름 경로들(98(0)-98(X)) 내의 역확산 회로 출력들(234(0)-234(X)) 상에서 역확산된 결과적 출력 벡터 데이터 샘플 세트(229(0)-229(Z))를 제공하도록 구성된다.

[0172] 이러한 실시예에서, 출력 데이터 흐름 경로들(98(0)-98(X))에서 역확산된 결과적 출력 벡터 데이터 샘플 세트(229(0)-229(Z))를 벡터 데이터 파일들(82(0)-82(X))에 제공하기 위한 도 27의 VPE(22(5))의 부가적인 세부사항들 및 특성들의 추가적인 설명이 이제 설명될 것이다. 이와 관련하여, 도 28은, 결과적 출력 벡터 데이터 샘플 세트(228(0)-228(X))의 역확산을 요구하는 예시적인 벡터 명령에 따라 역확산 회로(230)를 이용하여, 도 27의 VPE(22(5))에서 수행될 수 있는 역확산 벡터 프로세싱 연산(236)으로부터 초래하는 결과적 출력 벡터 데이터 샘플 세트들(228(0)-228(X))의 예시적인 역확산을 예시한 흐름도이다.

[0173] 도 27 및 28을 참조하면, 벡터 명령에 따라서 역확산 벡터 프로세싱 연산(236)에 따라 프로세싱될 입력 벡터 데이터 샘플 세트(86(0)-86(X))는, 벡터 데이터 파일들(82(0)-82(X))로부터 패치되고, 입력 데이터 흐름 경로들

(80(0)-80(X))에서 제공된다(도 28의 블록(238)). 도 27의 VPE(22(5)) 내의 벡터 데이터 레인들(100(0)-100(X)) 중 어느 하나, 일부, 또는 모두는, 결과적 역확산 벡터 프로세싱 연산(236)에 대한 결과적 출력 벡터 데이터 샘플 세트(228(0)-228(X))의 폭에 의존하여 벡터 명령의 프로그래밍에 따라 역확산 벡터 프로세싱 연산(236)을 제공하도록 이용될 수 있다. 역확산 벡터 프로세싱 연산(236)이 결과적 출력 벡터 데이터 샘플 세트(228(0)-228(X))에서 모든 결과적 출력 벡터 데이터 샘플들(228)의 역확산을 수행하는 것을 수반하면, 실행 유닛들(84(0)-84(X))로부터의 출력 데이터 흐름 경로들(98(0)-98(X))에서의 모든 벡터 데이터 레인들(100(0)-100(X))은 역확산 벡터 프로세싱 연산(236)을 위해 이용될 수 있다. 대안적으로, 역확산 벡터 프로세싱 연산(236)은, 결과적 출력 벡터 데이터 샘플 세트(228(0)-228(X)) 내의 결과적 출력 벡터 데이터 샘플들(228)의 서브세트를 역확산하는 것만을 수반할 수 있으며, 따라서, 결과적 출력 벡터 데이터 샘플들(228)의 서브세트에 대응하는 출력 데이터 흐름 경로들(98)에서 벡터 데이터 레인들(100)만을 수반한다.

[0174] 도 27 및 28을 계속 참조하면, 도 27의 VPE(22(5))에서 역확산 회로(230)에 의해 수행되는 역확산 벡터 프로세싱 연산 이전에, 패치된 입력 벡터 데이터 샘플 세트(86(0)-86(X))는 실행 유닛들(84(0)-84(X))에서 입력 데이터 흐름 경로들(80(0)-80(X))로부터 수신된다(도 28의 블록(240)). 실행 유닛들(84(0)-84(X))은, 벡터 명령에 따라서 제공된 벡터 프로세싱 연산에 따라 수신된 입력 벡터 데이터 샘플 세트(86(0)-86(X))에 대해 하나 또는 그 초과의 벡터 프로세싱 연산들을 수행한다(도 28의 블록(242)). 예를 들어, 실행 유닛들(84(0)-84(X))은, 결과적 출력 벡터 데이터 샘플 세트(228(0)-228(X))를 제공하도록 벡터 프로세싱 연산을 수행하기 위해 기준 벡터 데이터 샘플 세트(130(0)-130(X)) 내의 입력 벡터 데이터 샘플 세트(86(0)-86(X)) 및 코드 시퀀스를 사용하는 곱셈들 및/또는 누산들을 제공한다. 예를 들어, 결과적 출력 벡터 데이터 샘플 세트(228(0)-228(X))는, 도 27의 VPE(22(5))의 출력 데이터 흐름 경로들(98(0)-98(X))에서 제공되는 기준 벡터 데이터 샘플 세트(130(0)-130(X))를 이용한 입력 벡터 데이터 샘플 세트(86(0)-86(X))의 벡터 프로세싱에 기초할 수있다.

[0175] 도 27 및 도 28을 계속 참조하면, 결과적 출력 벡터 데이터 샘플 세트(228(0)-228(X))를 역확산하는 것이 요구되는 경우, 역확산 벡터 프로세싱 연산(236)은, 결과적 출력 벡터 데이터 샘플 세트(228(0)-228(X))가 벡터 데이터 파일들(82(0)-82(X))에 저장되기 전에 수행될 수 있다. 이러한 예에서, 결과적 출력 벡터 데이터 샘플 세트(228(0)-228(X))는, 도 27의 VPE(22(5)) 내의 실행 유닛들(84(0)-84(X))과 벡터 데이터 파일들(82(0)-82(X)) 사이에서 제공된 출력 데이터 흐름 경로들(98(0)-98(X))에서 제공되는 역확산 회로(230)에 제공된다. 역확산 회로(230)는, 실행되는 벡터 명령에 따라서, 그리고 벡터 명령이 벡터 데이터 파일들(82(0)-82(X))에 저장되도록 결과적 출력 벡터 데이터 샘플 세트(228(0)-228(X))의 역확산을 요청하면, 출력 데이터 흐름 경로들(98(0)-98(X))에서 결과적 출력 벡터 데이터 샘플 세트(228(0)-228(X))를 선택적으로 역확산하도록 프로그래밍된다. 역확산 회로(230)는, 결과적 출력 벡터 데이터 샘플 세트(228(0)-228(X))가 벡터 데이터 파일들(82(0)-82(X))에 저장되지 않으면서 실행되는 벡터 명령에 따른 역확산 프로그래밍에 따라 결과적 출력 벡터 데이터 샘플 세트(228(0)-228(X))를 역확산한다(도 28의 블록(244)).

[0176] 이러한 방식으로, 결과적 출력 벡터 데이터 샘플 세트(228(0)-228(X))는, 먼저 벡터 데이터 파일들(82(0)-82(X))에 저장되고, 재패치되고, 사후-프로세싱 연산에서 역확산되고, 벡터 데이터 파일들(82(0)-82(X))에 역확산된 포맷으로 저장될 필요가 없으며, 그에 의해, 실행 유닛들(84(0)-84(X))에서 지연을 제공한다. 결과적 출력 벡터 데이터 샘플 세트(228(0)-228(X))는, 요구되는 역확산 사후-프로세싱 없이 벡터 데이터 파일들(82(0)-82(X))에 역확산된 결과적 출력 벡터 데이터 샘플 세트(229(0)-229(Z))로서 저장된다(도 28의 블록(246)).

[0177] 도 29는, 도 27의 VPE(22(5)) 내의 실행 유닛들(84(0)-84(X))과 벡터 데이터 파일들(82(0)-82(X)) 사이의 출력 데이터 흐름 경로들(98(0)-98(X))에서 제공될 수 있는 예시적인 역확산 회로(230)의 개략도이다. 역확산 회로(230)는, 기준 벡터 데이터 샘플 세트(130(0)-130(X)) 내의 반복된 코드 시퀀스들의 상이한 확산 팩터들에 대해, 역확산된 결과적 출력 벡터 데이터 샘플 세트(229(0)-229(Z))를 제공하기 위해, 결과적 출력 벡터 데이터 샘플 세트(228(0)-228(X))의 역확산을 제공하도록 구성된다. 도 27에 예시된 바와 같이, 결과적 출력 벡터 데이터 샘플 세트(228(0)-228(X))는, 실행 유닛 출력들(96(0)-96(X))로부터 확산 회로(230)로 제공된다. 결과적 출력 벡터 데이터 샘플 세트(228(0)-228(X))의 확산 팩터가 알려지지 않을 수 있기 때문에, 도 27의 시퀀스 넘버 생성기(134)에 의해 생성된 기준 벡터 데이터 샘플 세트(130(0)-130(X)) 내의 반복 시퀀스 넘버들의 상이한 확산 팩터들을 이용하여 결과적 출력 벡터 데이터 샘플 세트(228(0)-228(X))를 역확산하는 것이 소망될 수있다.

[0178] 예를 들어, 결과적 출력 벡터 데이터 샘플 세트(228(0)-228(X))가 32개의 샘플들을 포함했고, 전체의 결과적 출력 벡터 데이터 샘플 세트(228(0)-228(X))가 사(4)의 확산 팩터를 가정하여 역확산되었다면, 역확산된 결과적 출력 벡터 데이터 샘플 세트(229(0)-229(Z))는, 결과적 출력 벡터 데이터 샘플 세트(228(0)-228(X))의 역확산이 수행된 이후, 여덟개(8)의 역확산 샘플들(즉, 32개의 샘플들/4의 확산 팩터)을 포함할 것이다. 그러나, 이러한

동일한 예에서, 전체의 결과적 출력 벡터 데이터 샘플 세트(228(0)-228(X))가 팔(8)의 확산 팩터를 가정하여 역확산되었다면, 역확산된 결과적 출력 벡터 데이터 샘플 세트(229(0)-229(Z))는, 결과적 출력 벡터 데이터 샘플 세트(228(0)-228(X))의 역확산이 수행된 이후, 네개(4)의 역확산 샘플들(즉, 32 개의 샘플들/8의 확산 팩터)을 포함할 것이다.

[0179] 따라서, 도 29를 계속 참조하면, 역확산 회로(230)는 상이한 수의 확산 팩터들에 대해 결과적 출력 벡터 데이터 샘플 세트(228(0)-228(X))를 역확산하도록 구성된다. 이러한 실시예에서, 역확산 회로(230)는, 하나의 벡터 프로세싱 동작/하나의 벡터 명령에서 상이한 확산 팩터들에 대한 역확산된 결과적 출력 벡터 데이터 샘플 세트(229(0)-229(Z))를 제공하도록 구성된다. 이와 관련하여, 역확산 회로(230)는, 결과적 출력 벡터 데이터 샘플 세트(228(0)-228(X))를 수신하기 위해 실행 유닛 출력들(96(0)-96(X))에 커플링된 가산기 트리(248)를 포함한다. 역확산 회로(230)의 가산기 트리(248)는, 그들 각각의 벡터 데이터 레인들(100(0)-100(X))에서 결과적 출력 벡터 데이터 샘플 세트(228(0)-228(X))의 각각의 샘플(228)을 수신하도록 구성된다. 제 1 가산기 트리 레벨(248(1))은 가산기 트리(248)에서 제공된다. 제 1 가산기 트리 레벨(248(1))은, 사(4)의 확산 팩터에 의해 결과적 출력 벡터 데이터 샘플 세트(228(0)-228(X))에서 샘플들(228)을 확산할 수 있기 위해, 가산기들(250(0)-250(((X+1)*2)-1), 250(7))로 구성된다. 래치들(251(0)-251(X))은, 출력 데이터 흐름 경로들(98(0)-98(X))로부터 결과적 출력 벡터 데이터 샘플 세트(228(0)-228(X))를 래치하기 위해 역확산 회로(230)에서 제공된다.

[0180] 예를 들어, 결과적 출력 벡터 데이터 샘플 세트(228(0)-228(X)) 내의 각각의 샘플(228)이 폭이 32비트들이고 두개(2)의 16비트 복소 벡터 데이터(즉, 포맷 IQ8에 따른 제 1 벡터 데이터 및 포맷 I8Q8에 따른 제 2 벡터 데이터)로 구성되면, 사(4)의 확산 팩터는, 결과적 출력 벡터 데이터 샘플 세트(228(0)-228(X)) 내의 두개(2)의 결과적 출력 벡터 데이터 샘플들(228)에서 네개(4)의 벡터 데이터 샘플들을 하나의 역확산된 결과적 출력 벡터 데이터 샘플로 역확산하도록 적용될 수 있다. 예를 들어, 도 29에 예시된 바와 같이, 가산기(250(0))는, 그들 샘플들에 대해 사(4)의 확산 팩터에 의해 결과적 출력 벡터 데이터 샘플들(228(0) 및 228(1))을 역확산하도록 구성된다. 유사하게, 가산기(250(1))는, 그들 샘플들에 대해 사(4)의 확산 팩터에 의해 결과적 출력 벡터 데이터 샘플들(228(2) 및 228(3))을 역확산하도록 구성된다. 가산기(250(((X+1)/2)-1), 250(7))는, 사(4)의 확산 팩터를 갖는 역확산된 벡터 데이터 샘플 세트(252(0)-252(((X+1)/2)-1), 252(7))를 제공하기 위해 결과적 출력 벡터 데이터 샘플 세트(228(X-1) 및 228(X))를 역확산하도록 구성된다. 가산기들(250(((X+1)/2)-1), 250(7))에 의해 수행된 역확산으로부터의 역확산된 벡터 데이터 샘플 세트(252(0)-252(((X+1)/2)-1), 252(7))는 래치들(255(0)-255(((X+1)/2)-1), 255(7))로 래치된다.

[0181] 역확산된 벡터 프로세싱 연산(236)이 사(4)의 확산 팩터에 의한 결과적 출력 벡터 데이터 샘플 세트(228(0)-228(X))의 역확산을 요구하면, 아래에서 더 상세히 논의될 바와 같이, 역확산된 벡터 데이터 샘플 세트(252(0)-252(((X+1)/2)-1), 252(7))는 역확산된 결과적 출력 벡터 데이터 샘플 세트(229(0)-229(Z))로서 제공될 수 있으며, 여기서 'Z'는 칠(7)이다. 그러나, 역확산된 벡터 프로세싱 연산(236)이 더 높은 확산 팩터(예를 들어, 8, 16, 32, 64, 128, 256)를 요청하면, 역확산된 벡터 데이터 샘플 세트(252(0)-252(((X+1)/2)-1), 252(7))는 역확산된 결과적 출력 벡터 데이터 샘플 세트(229(0)-229(Z))로서 제공되지 않는다. 역확산된 벡터 데이터 샘플 세트(252(0)-252(((X+1)/2)-1), 252(7))는, 가산기들(254(0)-254(((X+1)/4)-1), 254(3))로의 제 2 가산기 트리 레벨(248(2))에 제공된다. 이와 관련하여, 가산기(254(0))는, 그들 샘플들에 대한 팔(8)의 확산 팩터를 갖는 결과적 역확산된 벡터 데이터 샘플(256(0))을 제공하기 위해, 역확산된 벡터 데이터 샘플들(252(0) 및 252(1))에 대해 역확산을 수행하도록 구성된다. 유사하게, 가산기(254(1))는, 그들 샘플들에 대한 팔(8)의 확산 팩터를 갖는 결과적 역확산된 벡터 데이터 샘플(256(2))을 제공하기 위해, 역확산된 벡터 데이터 샘플들(252(1) 및 252(3))에 대해 역확산을 수행하도록 구성된다. 가산기(254(((X+1)/4)-1), 254(3))는, 팔(8)의 확산 팩터를 갖는 결과적 역확산된 벡터 데이터 샘플(256(((X+1)/4)-1), 256(3))를 제공하기 위해, 역확산된 벡터 데이터 샘플 세트(252(((X+1)/4)-2), 252(((X+1)/4)-1), 252(3))에 대해 역확산을 수행하도록 구성된다. 가산기들(254(0)-254(((X+1)/4)-1), 254(3))에 의해 수행된 역확산으로부터의 결과적 역확산된 벡터 데이터 샘플 세트(256(0)-256(((X+1)/4)-1), 256(3))는 래치들(257(0)-257(((X+1)/4)-1), 257(3))로 래치된다.

[0182] 도 29를 계속 참조하면, 역확산 벡터 프로세싱 연산(236)이 팔(8)의 확산 팩터에 의한 결과적 출력 벡터 데이터 샘플 세트(228(0)-228(X))의 역확산을 요구하는 경우, 하기에서 더욱 상세히 논의될 바와 같이, 역확산 벡터 데이터 샘플 세트(256(0)-256(((X+1)/4)-1), 256(3))는 역확산 결과적 출력 벡터 데이터 샘플 세트(229(0)-229(Z))로서 제공될 수 있고, 여기서 'Z'는 삼(3)이다. 그러나, 역확산 벡터 프로세싱 연산(236)이 팔(8)보다 더 높은 확산 팩터(예컨대, 16, 32, 64, 128, 256)를 호출하는 경우, 역확산 벡터 데이터 샘플 세트(256(0)-256(((X+1)/4)-1), 256(3))는 역확산 결과적 출력 벡터 데이터 샘플 세트(229(0)-229(Z))로서 제공되지

않는다. 역확산 벡터 데이터 샘플 세트(256(0)-256(((X+1)/4)-1), 256(3))는, 가산기들(258(0)-258(((X+1)/8)-1), 258(1))에 대한 제 3 가산기 트리 레벨(248(3))에 제공된다. 이 점에 있어서, 가산기(258(0))는, 역확산 벡터 데이터 샘플들(256(0) 및 256(1))에 대해 역확산을 수행하여, 그러한 샘플들에 십육(16)의 확산 팩터를 제공하도록 구성된다. 마찬가지로, 가산기(258(1))는, 역확산 벡터 데이터 샘플 세트(260(0)-260(((X+1)/8)-1), 260(1))에 십육(16)의 확산 팩터를 제공하기 위해, 역확산 벡터 데이터 샘플들(256(2) 및 256(3))에 대해 역확산을 수행하도록 구성된다. 가산기들(258(0)-258(((X+1)/8)-1), 258(1))에 의해 수행되는 역확산으로부터의 역확산 벡터 데이터 샘플 세트(260(0)-260(((X+1)/8)-1), 260(1))는 래치들(259(0)-259(((X+1)/8)-1), 259(2))로 래칭된다.

[0183] 계속해서 도 29를 참조하면, 역확산 벡터 프로세싱 연산(236)이 십육(16)의 확산 팩터에 의한 결과적 출력 벡터 데이터 샘플 세트(228(0)-228(X))의 역확산을 요구하는 경우, 하기에서 더욱 상세히 논의될 바와 같이, 역확산 벡터 데이터 샘플 세트(260(0)-260(((X+1)/8)-1), 256(1))는 역확산 결과적 출력 벡터 데이터 샘플 세트(229(0)-229(Z))로서 제공될 수 있고, 여기서 'Z'는 일(1)이다. 그러나, 역확산 벡터 프로세싱 연산(236)이 십육(16)보다 더 높은 확산 팩터(예컨대, 32, 64, 128, 256)를 호출하는 경우, 역확산 벡터 데이터 샘플 세트(260(0)-260(((X+1)/8)-1), 260(1))는 역확산 결과적 출력 벡터 데이터 샘플 세트(229(0)-229(Z))로서 제공되지 않는다. 역확산 벡터 데이터 샘플 세트(260(0)-260(((X+1)/8)-1), 260(1))는 가산기(262)에 대한 제 4 가산기 트리 레벨(248(4))에 제공된다. 이 점에 있어서, 가산기(262)는, 역확산 벡터 데이터 샘플(264)에 삼십이(32)의 확산 팩터를 제공하기 위해, 역확산 벡터 데이터 샘플들(260(0) 및 260(1))에 대해 역확산을 수행하도록 구성된다. 가산기(262)에 의해 수행되는 역확산으로부터의 역확산 벡터 데이터 샘플(264)은 래치들(266 및 268)로 래칭된다.

[0184] 계속해서 도 29를 참조하면, 역확산 벡터 프로세싱 연산(236)이 삼십이(32)의 확산 팩터에 의한 결과적 출력 벡터 데이터 샘플 세트(228(0)-228(X))의 역확산을 요구하는 경우, 하기에서 더욱 상세히 논의될 바와 같이, 역확산 벡터 데이터 샘플(264)은 역확산 결과적 출력 벡터 데이터 샘플(229)로서 제공될 수 있다. 그러나, 역확산 벡터 프로세싱 연산(236)이 삼십이(32)보다 더 높은 확산 팩터(예컨대, 64, 128, 256)를 호출하는 경우, 역확산 벡터 데이터 샘플(264)은 역확산 결과적 출력 벡터 데이터 샘플 세트(229)로서 제공되지 않는다. 벡터 데이터 파일(82)에 저장될 필요 없이, 역확산 벡터 데이터 샘플(264)은 래치(268)로 래칭된 채로 남아 있다. 위에서 설명된 바와 같이, 다른 결과적 출력 벡터 데이터 샘플 세트(228(0)-228(X))는, 삼십이(32)의 확산 팩터를 사용하여 역확산되도록, 부가의 프로세싱 사이클들에 걸쳐 래치들(251(0)-251(X))로 로딩된다. 결과적 역확산 벡터 데이터 샘플(264')이 가산기(270)에 의해 제 5 가산기 트리(248(5))에서 이전의 역확산 벡터 데이터 샘플(264)에 가산되어, 육십사(64)의 확산 팩터를 갖는 역확산 벡터 데이터 샘플(272)이 제공된다. 선택기(273)는, 역확산 벡터 데이터 샘플(272)이 래치(274)로 래칭될 때, 삼십이(32)의 확산 팩터를 갖는 역확산 벡터 데이터 샘플(264)이 래칭되는지 또는 육십사(64)의 확산 팩터를 갖는 역확산 벡터 데이터 샘플(264')이 래칭되는지 여부를 제어한다. 부가의 결과적 출력 벡터 데이터 샘플 세트들(228(0)-228(X))을 래칭하고 이들을 역확산시키는 이러한 동일한 프로세스는, 원해진다면, 육십사(64)보다 더 큰 확산 팩터들을 달성하기 위해 수행될 수 있다. 역확산 벡터 데이터 샘플(272)은 궁극적으로, 역확산 벡터 프로세싱 연산(236)에 대한 원하는 확산 팩터에 따른 원하는 역확산 결과적 출력 벡터 데이터 샘플(229)로서 래치(274)로 래칭될 것이다.

[0185] 계속해서 도 29를 참조하면, 비록 어떤 확산 팩터가 역확산 벡터 프로세싱 연산(236)에서 호출될지라도, 역확산 결과적 출력 벡터 데이터 샘플 세트(229(0)-229(Z))는 도 27에서 벡터 데이터 파일들(82(0)-82(X))에 저장될 필요가 있을 것이다. 이제 논의될 바와 같이, 도 29의 역확산 회로(230)는 또한, 역확산 결과적 출력 벡터 데이터 샘플 세트(229(0)-229(Z))를 형성하기 위해, 결과적 출력 벡터 데이터 샘플들(228(0)-228(X))에 대해 역확산 벡터 프로세싱 연산(236)을 수행하는 것의 결과로서 제공되는 역확산 결과적 출력 벡터 데이터 샘플들(229)을 래치들(276(0)-276(X))로 로딩하도록 구성된다. 역확산 결과적 출력 벡터 데이터 샘플 세트(229(0)-229(Z))는, 저장되도록, 벡터 데이터 파일들(82(0)-82(X))에 제공될 수 있다. 이러한 방식으로, 역확산 회로(230)에 의해 생성되는 역확산 결과적 출력 벡터 데이터 샘플 세트(229(0)-229(Z))를 저장하기 위해, 벡터 데이터 파일들(82(0)-82(X))에 대해 단 한(1) 번의 쓰기가 요구된다. 어떤 확산 팩터가 역확산 벡터 프로세싱 연산(236)에서 호출되는지에 관계없이, 도 29의 역확산 회로(230)의 가산기 트리들(248(1)-248(5))은 확산 팩터들(4, 8, 16, 및 32) 전부에 대해 역확산 결과적 출력 벡터 데이터 샘플들(229)을 생성할 수 있다. 대안적으로, 원하는 확산 팩터에 따른 역확산 벡터 프로세싱 연산(236)을 수행할 필요가 없는 가산기 트리들의 가산기들은 디스에이블될 수 있거나 또는 0들을 가산하도록 구성될 수 있다. 그러나, 이들 역확산 결과적 출력 벡터 데이터 샘플들(229) 중 어느 것이 저장되도록 래치들(276(0)-276(X))에 제공될 것인지를 결정하기 위해, 이제 논의될 바와 같이, 선

택기들(278(0)-278(((X+1)/4)-1), 278(3))이 제공된다.

[0186] 이 점에 있어서, 계속해서 도 29를 참조하면, 선택기(278(0))는, 실행되고 있는 역확산 벡터 프로세싱 연산(236)에 기초하여, 각각, 가산기들(250(0), 254(0), 258(0))로부터의 확산 팩터들(4, 8, 및 16) 중 임의의 팩터에 대해, 그리고 가산기들(262, 270)로부터의 확산 팩터들(32, 64, 128, 256)에 대해, 역확산 결과적 출력 벡터 데이터 샘플들(229)을 선택할 수 있다. 선택기(278(1))는, 실행되고 있는 역확산 벡터 프로세싱 연산(236)에 기초하여, 각각, 가산기들(250(1), 254(1), 및 258(1))로부터의 확산 팩터들(4, 8, 및 16)에 대해, 역확산 결과적 출력 벡터 데이터 샘플들(229)을 선택할 수 있다. 선택기(278(2))는, 실행되고 있는 역확산 벡터 프로세싱 연산(236)에 기초하여, 각각, 가산기들(250(2) 및 254(2))로부터의 확산 팩터들(4 및 8)에 대해, 역확산 결과적 출력 벡터 데이터 샘플들(229)을 선택할 수 있다. 선택기(278(3))는, 실행되고 있는 역확산 벡터 프로세싱 연산(236)에 기초하여, 각각, 가산기들(250(3) 및 254(3))로부터의 확산 팩터들(4 및 8)에 대해, 역확산 결과적 출력 벡터 데이터 샘플들(229)을 선택할 수 있다. 선택기(278(4))는, 실행되고 있는 역확산 벡터 프로세싱 연산(236)에 기초하여, 각각, 가산기 트리들(248(1) 및 248(2))로부터의 확산 팩터들(4 및 8)에 대해, 역확산 결과적 출력 벡터 데이터 샘플들(229)을 선택할 수 있다. 선택기들은 가산기들(250(4)-250(7))로부터 제공되는 역확산 결과적 출력 벡터 데이터 샘플들(229)을 제어하기 위해서는 제공되지 않는데, 그 이유는 팔(8)의 확산 팩터를 제공하는 것이 선택기들(278(0)-278(3))에 의해 완전히 충족될 수 있기 때문이다.

[0187] 계속해서 도 29를 참조하면, 일련의 데이터 슬라이서들(280(0)-280(((X+1)/2)-1), 280(7))은, 각각, 선택기들(278(0)-278(((X+1)/4)-1), 278(3)) 및 가산기들(250(4)-250(((X+1)/2)-1), 250(7))에 의해 선택되는 역확산 결과적 출력 벡터 데이터 샘플들(229)을 수신하기 위해 제공된다. 데이터 슬라이서들(280(0)-280(((X+1)/2)-1), 280(7))은, 자신이 수신한 역확산 결과적 출력 벡터 데이터 샘플들(229)이 논리 하이 레벨(예컨대, 논리 '1')로서 특성화될 것인지 또는 논리 로우 레벨(예컨대, 논리 '0')로서 특성화될 것인지 여부를 선택하도록 구성된다. 이후, 역확산 결과적 출력 벡터 데이터 샘플들(229)은, 저장되도록, 크로스바(282)에 대한 연결들을 통해 래치들(276(0)-276(X)) 중 원하는 래치(276)로 라우팅된다. 크로스바(282)는, 역확산 벡터 프로세싱 연산(236)에 따른 역확산 결과적 출력 벡터 데이터 샘플들(229)을 상이한 래치들(276(0)-276(X))에 제공하기 위한 유연성을 제공한다. 이러한 방식으로, 역확산 결과적 출력 벡터 데이터 샘플들(229)은, 벡터 데이터 파일들(82(0)-82(X))에 저장되기 이전에, 역확산 벡터 프로세싱 연산들(236)의 상이한 반복들 중에 래치들(276(0)-276(X))에 스택될 수 있다. 예컨대, 역확산 결과적 출력 벡터 데이터 샘플 세트(229(0)-229(Z))는, 벡터 데이터 파일들(82(0)-82(X))에 저장되기 이전에, 역확산 벡터 프로세싱 연산들(236)의 상이한 반복들 중에 래치들(276(0)-276(X))에 스택될 수 있다. 이러한 방식으로, 역확산 결과적 출력 벡터 데이터 샘플 세트(229(0)-229(Z))를 저장하기 위한 벡터 데이터 파일들(82(0)-82(X))로의 액세스들이, 연산 효율성을 위해 최소화될 수 있다.

[0188] 예컨대, 도 29에 예시된 바와 같이, 크로스바(282)에 커플링된 선택기들(284(0)-284(X))은, 데이터 슬라이서(280(0))로부터의 역확산 결과적 출력 벡터 데이터 샘플(229)을 래치들(276(0)-276(X)) 중 임의의 래치에 저장하도록 제어될 수 있다. 크로스바(282)에 커플링된 선택기들(284(1), 284(3), 284(5), 284(7), 284(9), 284(11), 284(13), 284(15))은, 데이터 슬라이서들(280(1))로부터의 역확산 결과적 출력 벡터 데이터 샘플(229)을, 래치들(276(1), 276(3), 276(5), 276(7), 276(9), 276(11), 276(13), 및 276(15))에 저장되도록 저장하기 위해 제어될 수 있다. 크로스바(282)에 커플링된 선택기들(284(2), 284(6), 284(10), 284(14))은, 데이터 슬라이서(280(2))로부터의 역확산 결과적 출력 벡터 데이터 샘플(229)을 래치들(276(2), 276(6), 276(10), 및 276(14))에 저장하도록 제어될 수 있다. 크로스바(282)에 커플링된 선택기들(284(3), 284(7), 284(11), 284(15))은, 데이터 슬라이서(280(3))로부터의 역확산 결과적 출력 벡터 데이터 샘플(229)을 래치들(276(3), 276(7), 276(11), 및 276(15))에 저장하도록 제어될 수 있다. 크로스바(282)에 커플링된 선택기들(284(4) 및 284(12))은, 데이터 슬라이서(280(4))로부터의 역확산 결과적 출력 벡터 데이터 샘플(229)을 래치들(276(4) 및 276(12))에 저장하도록 제어될 수 있다. 크로스바(282)에 커플링된 선택기들(284(5) 및 284(13))은, 데이터 슬라이서(280(5))로부터의 역확산 결과적 출력 벡터 데이터 샘플(229)을, 래치들(276(5) 및 276(13))에 저장되도록 저장하기 위해 제어될 수 있다. 크로스바(282)에 커플링된 선택기들(284(6) 및 284(14))은, 데이터 슬라이서(280(6))로부터의 역확산 결과적 출력 벡터 데이터 샘플(229)을 래치(276(6) 또는 276(14))에 저장하도록 제어될 수 있다. 크로스바(282)에 커플링된 선택기들(284(7) 및 284(15))은, 데이터 슬라이서(280(7))로부터의 역확산 결과적 출력 벡터 데이터 샘플(229)을 래치들(276(7) 또는 276(15))에 저장하도록 제어될 수 있다.

[0189] 계속해서 도 29를 참조하면, 역확산 회로(230)는, 실행될 벡터 명령에 따라 결과적 출력 벡터 데이터 샘플들(228(0)-228(X))에 대해 역확산 연산들을 수행하거나 또는 수행하지 않기 위해 구성되도록 프로그래밍될 수 있

다. 이 점에 있어서, 결과적 출력 벡터 데이터 샘플들(228(0)-228(X))에 대해 역확산 연산들을 수행하기 위해, 또는 단순히, 벡터 데이터 파일들(82(0)-82(X))에 저장되도록, 각각, 결과적 출력 벡터 데이터 샘플들(228(0)-228(X))을 래치들(276(0)-276(X))에 제공하기 위해, 도 29의 역확산 구성 입력(286)이 역확산 회로(230)에 제공될 수 있다. 이러한 방식으로, 역확산 회로(230)는, 벡터 명령이 이러한 프로세싱이 수행되도록 제공하지 않는 경우, 결과적 출력 벡터 데이터 샘플 세트들(228(0)-228(X))을 역확산시키지 않도록 프로그래밍될 수 있다. 역확산 구성 입력(284)은, 각각의 벡터 명령에 대해, 도 27의 VPE(22(5))에 의한 벡터 프로세싱에서의 유연성을 제공하도록 구성 및 재구성될 수 있다. 예컨대, 역확산 구성 입력(284)은, 원해진다면, 벡터 명령의 각각의 클록-사이클에 대해, 원해진다면, 클록-사이클마다, 원하는 역확산에 실행 유닛들(84(0)-84(X))의 전체 사용 효율을 제공하도록 구성 및 재구성될 수 있다.

[0190] 특정한 다른 무선 베이스밴드 연산들은, 확산 스펙트럼 데이터 시퀀스들의 역확산 이외의 이유들로, 이전의 프로세싱 연산들로부터 결정된 데이터 샘플들의 병합을 요구한다. 예컨대, 그것은 벡터 데이터 레인들(100(0)-100(X))에 의해 제공되는 실행 유닛들(84(0)-84(X))에 대한 데이터 흐름 경로들보다 더 넓은 가변 폭들의 벡터 데이터 샘플들을 누산시키기 위해 요구될 수 있다. 다른 예로서, 그것은 벡터 프로세싱 연산들에서 출력 벡터 데이터의 병합을 제공하기 위해, 상이한 실행 유닛들(84(0)-84(X))로부터의 출력 벡터 데이터 샘플들의 내적 곱셈을 제공하기 위해 요구될 수 있다. VPE의 벡터 데이터 레인들(100(0)-100(X))은, 병합된 벡터 프로세싱 연산들을 제공하기 위해 벡터 데이터 레인들(100(0)-100(X))을 크로스 오버하기 위한 인트라벡터 데이터 경로들을 제공하기 위해서, 복잡한 라우팅을 포함할 수 있다. 그러나, 이는 복잡성을 증가시키고, 그리고 상이한 벡터 데이터 레인들을 크로스 오버하여 병합될 출력 벡터 데이터의 병렬화 어려움들 때문에, VPE의 효율성을 감소시킬 수 있다. 벡터 프로세서들은, 실행 유닛들로부터 벡터 데이터 메모리로 저장되는 출력 벡터 데이터의 사후-프로세싱 병합을 수행하는 회로를 포함할 수 있다. 벡터 데이터 메모리에 저장된, 사후-프로세싱된 출력 벡터 데이터 샘플들은 벡터 데이터 메모리로부터 패치되고, 원하는 대로 병합되며, 그리고 다시 벡터 데이터 메모리에 저장된다. 그러나, 이러한 사후-프로세싱은 VPE의 후속 벡터 프로세싱 연산들을 지연시킬 수 있고, 그리고 실행 유닛들의 계산 컴포넌트들이 충분히 이용되지 않게 할 수 있다.

[0191] 예컨대, 이전에 설명된 VPE의 벡터 데이터 파일들(82(0), 82(1))에 제공되는 두 개의 입력 벡터 데이터 샘플들(290(0), 290(1))이 도 30에 도시된다. 이들 두 개의 입력 벡터 데이터 샘플들(290(0), 290(1))을 서로 가산시키는 것이 원해질 수 있다. 이 예에서, 두 개의 입력 벡터 데이터 샘플들(290(0), 290(1))의 합은 '0x11250314E'이고, 이는 벡터 데이터 레인(100(0) 또는 100(1))보다 더 넓은 데이터 폭을 갖는다. 벡터 데이터 레인들(100(0), 100(1))에 걸쳐 두 개의 실행 유닛들(84(0), 84(1)) 사이에 캐리 논리를 제공하는 것을 비롯해, 실행 유닛들(84(0), 84(1))이 두 개의 입력 벡터 데이터 샘플들(290(0), 290(1))의 서로의 합의 실행을 수행하도록 허용하기 위해, 벡터 데이터 레인들(100(0), 100(1)) 사이에 벡터 데이터 라우팅을 제공하기 위해서, 데이터 흐름 경로들이 VPE(22)에 제공될 수 있다. 병합된 벡터 데이터 샘플들의 스칼라 결과를 제공하기 위해, 모든 벡터 데이터 레인들(100(0)-100(X))을 크로스하는 능력이 요구될 수 있고, 이는 데이터 흐름 경로들에서의 복잡성을 추가로 증가시킬 수 있다. 그러나, 위에서 논의된 바와 같이, 이것이 데이터 흐름 경로들에서의 복잡성을 가산시켜서, 복잡성은 증가되고 아마도 효율성은 감소된다.

[0192] 이 이슈를 다루기 위해, 하기에 개시되는 실시예들은, VPE의 벡터 데이터 메모리와 실행 유닛들 사이의 출력 데이터 흐름 경로들에 제공되는 병합 회로를 포함하는 VPE들을 포함한다. 병합 회로는, 출력 벡터 데이터 샘플 세트가 실행 유닛들로부터 출력 데이터 흐름 경로들을 통해 벡터 데이터 메모리로 제공되고 있는 동안에, 실시간 실행 유닛들에 의해 제공되는 출력 벡터 데이터 샘플 세트로부터의 출력 벡터 데이터 샘플들을 병합하도록 구성된다. 출력 벡터 데이터 샘플들의 실시간 병합은, 실행 유닛들에 의해 제공되는 출력 벡터 데이터 샘플들이 벡터 데이터 메모리에 저장되기 이전에 병합될 수 있고, 따라서 결과적 출력 벡터 데이터 샘플 세트가 벡터 데이터 메모리에 병합된 포맷으로 저장됨을 의미한다. 실행 유닛들에서 수행될 후속 벡터 프로세싱 연산들을 지연시킬 수 있는 부가의 사후-프로세싱 단계들을 요구하는 것 없이, 병합된 출력 벡터 데이터 샘플들은 벡터 데이터 파일들에 저장될 수 있다. 따라서, VPE에서의 데이터 흐름 경로들의 효율성은 벡터 데이터 병합 연산들에 의해 제한되지 않는다. 실행 유닛들에서의 후속 벡터 프로세싱은, 병합된 벡터 데이터 샘플들이 벡터 데이터 메모리에 저장될 때, 데이터 흐름 제한들에 의해서가 아니라, 계산 자원들에 의해서만 제한된다.

[0193] 이 점에 있어서, 도 31은, 도 2에서 VPE(22)로서 제공될 수 있는 다른 예시적 VPE(22(6))의 개략도이다. 하기에서 더욱 상세히 설명될 바와 같이, 도 31의 VPE(22(6))는, 제거된 또는 감소된 벡터 데이터 샘플 재패치 및 감소된 전력 소비량으로 VPE(22(6))의 벡터 데이터 파일들(82(0)-82(X))에 저장되도록, 벡터 프로세싱 연산들에 대한 코드 시퀀스를, 실행 유닛들(84(0)-84(X))에 의해 제공되는 결과적 출력 벡터 데이터 샘플 세트들(292(0)-

292(X))의 실시간 병합에 제공하도록 구성된다. 결과적 출력 벡터 데이터 샘플 세트(292(0)-292(X))는 결과적 출력 벡터 데이터 샘플들(292(0), ... , 292(X))로 구성된다. 비제한적 예들로서, 병합 벡터 프로세싱 연산은, 결과적 출력 벡터 데이터 샘플들(292)을 가산하는 것, 복수의 결과적 출력 벡터 데이터 샘플들(292) 중 최대 벡터 데이터 샘플 값을 결정하는 것, 또는 복수의 출력 벡터 데이터 샘플들(292) 중 최소 벡터 데이터 샘플 값을 결정하는 것을 포함할 수 있다. 도 31의 VPE(22(6))에서, 결과적 출력 벡터 데이터 샘플 세트(292(0)-292(X)) 중의 결과적 출력 벡터 데이터 샘플들(292)은, 벡터 데이터 파일들(82(0)-82(X))에 저장되기 이전에 병합될 수 있다.

[0194] 병합 회로(294)는, 결과적 출력 벡터 데이터 샘플 세트(228(0)-228(X)) 중 결과적 출력 벡터 데이터 샘플들(228)의 실시간 병합을 제공하기 위해 실행될 벡터 명령에 따른 프로그래밍에 기초하여 구성된다. 병합된 결과적 출력 벡터 데이터 샘플들(296(0)-296(Z))은 출력 데이터 흐름 경로들(98(0)-98(X))에서 병합 회로(294)에 의해 제공된다. 병합된 결과적 출력 벡터 데이터 샘플들(296(0)-296(Z))의 'Z'는, 병합된 결과적 출력 벡터 데이터 샘플 세트들(296(0)-296(Z))에서 병합된 결과적 출력 벡터 데이터 샘플들(296)의 개수를 표현한다. 병합된 결과적 출력 벡터 데이터 샘플 세트(296(0)-296(Z))는, 이 예에서 296(0), ..., 및 296(Z)인 결과적 출력 벡터 데이터 샘플들(296)로 구성된다. 병합된 결과적 출력 벡터 데이터 샘플 세트(296(0)-296(Z))에서 병합된 결과적 출력 벡터 데이터 샘플들(296)의 개수는, 결과적 출력 벡터 데이터 샘플 세트(292(0)-292(X))에 대해 수행되는 병합 연산들에 따라 좌우된다. 도 31에서 VPE(22(6))의 결과적 출력 벡터 데이터 샘플들(292)의 실시간 병합은, 실행 유닛들(84(0)-84(X))에 의해 제공되는 결과적 출력 벡터 데이터 샘플 세트(292(0)-292(X))의 결과적 출력 벡터 데이터 샘플들(292)이 벡터 데이터 파일들(82(0)-82(X))에 저장되기 이전에 서로 병합될 수 있음을 의미한다. 이러한 방식으로, 병합된 결과적 출력 벡터 데이터 샘플 세트(296(0)-296(Z))의 병합된 결과적 출력 벡터 데이터 샘플들(296)은, 병합된 결과적 출력 벡터 데이터 샘플 세트(296(0)-296(Z))로서 병합된 형태로 벡터 데이터 파일들(82(0)-82(X))에 저장될 수 있다.

[0195] 따라서, 출력 데이터 흐름 경로들(98(0)-98(X))에 제공되는 병합 회로(294)를 이용하여, 결과적 출력 벡터 데이터 샘플 세트(292(0)-292(X))는, 먼저 벡터 데이터 파일들(82(0)-82(X))에 저장되고 이후 벡터 데이터 파일들(82(0)-82(X))로부터 패치될 것을 요구받지 않는다. 원하는 결과적 출력 벡터 데이터 샘플들(292)이 병합되고, 그리고 결과적 출력 벡터 데이터 샘플들(292)이 병합된 형태로 벡터 데이터 파일들(82(0)-82(X))에 재저장된다. 결과적 출력 벡터 데이터 샘플 세트(292(0)-292(X))로부터의 결과적 출력 벡터 데이터 샘플들(292)은, 벡터 데이터 파일들(82(0)-82(X))에 저장되기 이전에 병합될 수 있다. 이러한 방식으로, 실행 유닛들(84(0)-84(X))에서 수행될 후속 벡터 프로세싱 연산들을 지연시킬 수 있는 부가의 사후-프로세싱 단계들을 요구하는 것 없이, 병합된 결과적 출력 벡터 데이터 샘플 세트(296(0)-296(Z))로부터의 병합된 결과적 출력 벡터 데이터 샘플들(296)은 벡터 데이터 파일들(82(0)-82(X))에 저장된다. 따라서, VPE(22(6))의 데이터 흐름 경로들의 효율성은 결과적 출력 벡터 데이터 샘플들(292)의 병합에 의해 제한되지 않는다. 실행 유닛들(84(0)-84(X))에서의 후속 벡터 프로세싱은, 결과적 출력 벡터 데이터 샘플들(292)이 병합된 형태로 벡터 데이터 파일들(82(0)-82(X))에 저장될 때, 데이터 흐름 제한들에 의해서가 아니라, 계산 자원들에 의해서만 제한된다.

[0196] 추가로, 실행 유닛들(84(0)-84(X))과 벡터 데이터 파일들(82(0)-82(X)) 사이의 출력 데이터 흐름 경로들(98(0)-98(X))에 병합 회로(294)를 제공함으로써, 결과적 출력 벡터 데이터 샘플 세트(292(0)-292(X))는, 벡터 데이터 파일들(82(0)-82(X))과 실행 유닛들(84(0)-84(X)) 사이의 입력 데이터 흐름 경로들(80(0)-80(X))에서 벡터 데이터 레인들(100)을 크로스할 필요가 없다. 상이한 벡터 데이터 레인들(100) 사이에 입력 벡터 데이터 샘플 세트(86(0)-86(X))의 입력 벡터 데이터 샘플들(86)의 병합을 위한 데이터 흐름 경로들을 제공하는 것은, 라우팅 복잡성들을 증가시킬 것이다. 그 결과, 병합 연산들이 입력 데이터 흐름 경로들(80(0)-80(X))에서 수행되고 있는 동안에, 실행 유닛들(84(0)-84(X))은 충분히 이용되지 않을 수 있다. 또한, 위에서 논의된 바와 같이, 입력 데이터 흐름 경로들(80(0)-80(X))에서 결과적 출력 벡터 데이터 샘플 세트(292(0)-292(X))로부터의 결과적 출력 벡터 데이터 샘플들(292)의 병합은, 결과적 출력 벡터 데이터 샘플 세트(292(0)-292(X))가 도 31의 VPE(22(6))의 벡터 데이터 파일들(82(0)-82(X))에 먼저 저장되도록 요구할 것이고, 이는 이로써, 재패치 및 병합될 때 전력 소비량을 증가시키고, 그리고/또는 병합 연산들이 수행되고 있는 동안에 지연될 수 있는 실행 유닛들(84(0)-84(X))의 충분하지 않은 이용의 위험성이 있다.

[0197] 도 4, 도 11, 도 19, 도 23, 및 도 27의 VPE들(22(1)-22(5))에서 제공되는 공통 컴포넌트들이 도 31의 VPE(22(6))에서 제공된다는 점이 주목된다. 공통 컴포넌트들이, 공통 엘리먼트 번호들을 이용하여 도 31의 VPE(22(6))에 예시된다. VPE들(22(1)-22(5))에서의 위의 이러한 공통 컴포넌트들의 이전 설명 및 논의는 또한 도 31의 VPE(22(6))에 적용가능하며, 따라서, 여기서 다시 설명되지 않을 것이다.

[0198] 도 31을 계속 참조하면, 더 구체적으로, 병합 회로(294)는 출력 데이터 흐름 경로들(98(0)-98(X)) 상의 병합 회로 입력들(300(0)-300(X)) 상에서 결과적 출력 벡터 데이터 샘플 세트(292(0)-292(X))를 수신하도록 구성된다. 병합 회로(294)는, 병합된 결과적 출력 벡터 데이터 샘플 세트들(296(0)-296(Z))을 제공하기 위해서 결과적 출력 벡터 데이터 샘플 세트들(292(0)-292(X))로부터의 원하는 결과적 출력 벡터 데이터 샘플들(292)을 병합하도록 구성된다. 병합된 결과적 출력 벡터 데이터 샘플 세트(296(0)-296(Z))에서의 'Z'는 병합된 결과적 출력 벡터 데이터 샘플 세트(296(0)-296(Z))의 비트 폭을 표현한다. 'Z'는 병합 연산들로 인하여, 'X'로 표현되는 결과적 출력 벡터 데이터 샘플 세트(292(0)-292(X))의 비트 폭보다 좁을 수 있다. 아래에서 더 상세하게 논의되는 바와 같이, 병합된 결과적 출력 벡터 데이터 샘플 세트(296(0)-296(Z))에서의 병합된 결과적 출력 벡터 데이터 샘플들(296)의 수 'Z+1'는 함께 병합될 결과적 출력 벡터 데이터 샘플 세트(292(0)-292(X))로부터의 결과적 출력 벡터 데이터 샘플들(292)에 의존한다. 병합 회로(294)는 저장을 위해서 벡터 데이터 파일들(82(0)-82(X))에 제공될 출력 데이터 흐름 경로들(98(0)-98(X))에서의 병합 회로 출력들(301(0)-301(X)) 상에서 병합된 결과적 출력 벡터 데이터 샘플 세트(296(0)-296(Z))를 제공하도록 구성된다.

[0199] 본 실시예에서 병합된 결과적 출력 벡터 데이터 샘플 세트(296(0)-296(X))를 출력 데이터 흐름 경로들(98(0)-98(X))의 벡터 데이터 파일들(82(0)-82(X))에 제공하는 것에 대한 도 31의 VPE(22(6))의 부가적인 세부사항들 및 특징들의 추가 설명이 이제 설명될 것이다. 이와 관련하여, 도 32는 결과적 출력 벡터 데이터 샘플들(292)의 병합을 요구하는 예시적 벡터 명령에 따라 병합 회로(294)를 이용하는 도 31의 VPE(22(6))에서 수행될 수 있는 벡터 프로세싱 연산(302)으로부터 발생하는 결과적 출력 벡터 데이터 샘플 세트들(292(0)-292(X))의 결과적 출력 벡터 데이터 샘플들(292)의 예시적인 병합을 예시하는 흐름도이다.

[0200] 도 31 및 도 32를 참조하면, 벡터 명령에 따른 벡터 프로세싱 연산(302)에 따라 프로세싱될 입력 벡터 데이터 샘플 세트(86(0)-86(X))가 벡터 데이터 파일들(82(0)-82(X))로부터 패치되며, 입력 데이터 흐름 경로들(80(0)-80(X))에서 제공된다(도 32의 블록(304)). 도 31의 VPE(22(6))의 벡터 데이터 레인들(100(0)-100(X)) 중 하나, 일부, 또는 전부가 벡터 프로세싱 연산(302)을 위한 입력 벡터 데이터 샘플 세트(86(0)-86(X))의 폭에 의존하여 벡터 명령의 프로그래밍에 따른 벡터 프로세싱 연산(302)을 제공하기 위해서 이용될 수 있다. 벡터 데이터 파일들(82(0)-82(X))의 전체 폭이 요구되는 경우, 모든 벡터 데이터 레인들(100(0)-100(X))은 벡터 프로세싱 연산(302)을 위해서 이용될 수 있다. 벡터 프로세싱 연산(302)은 단지 벡터 데이터 레인들(100(0)-100(X))의 서브세트만을 요구할 수 있다. 이는 입력 벡터 데이터 샘플 세트(86(0)-86(X))의 폭이 모든 벡터 데이터 파일들(82(0)-82(X))의 폭보다 좁기 때문인데, 여기서, 벡터 프로세싱 연산(302)과 동시에 수행될 다른 벡터 프로세싱 연산들을 위해서 부가적인 벡터 데이터 레인들(100)을 이용하는 것이 바람직하다.

[0201] 도 31 및 도 32를 계속 참조하면, 패치된 입력 벡터 데이터 샘플 세트(86(0)-86(X))가 실행 유닛들(84(0)-84(X))에서 입력 데이터 흐름 경로들(80(0)-80(X))로부터 수신된다(도 32의 블록(306)). 실행 유닛(84(0)-84(X))은 벡터 명령에 따라 제공되는 벡터 프로세싱 연산(302)에 따라 수신된 입력 벡터 데이터 샘플 세트(86(0)-86(X)) 상에서 벡터 프로세싱 연산(302)을 수행한다(도 32의 블록(308)). 실행 유닛들(84(0)-84(X))은 결과적 출력 벡터 데이터 샘플 세트(292(0)-292(X))를 제공하기 위해서 벡터 프로세싱 연산(302)을 위한 입력 벡터 데이터 샘플 세트(86(0)-86(X))를 사용하는 곱셈들 및/또는 누산을 제공할 수 있다. 일단 벡터 프로세싱 연산(302)이 완료되면, 입력 벡터 데이터 샘플 세트(86(0)-86(X)) 상에서 수행되는 벡터 프로세싱 연산(302)에 기초하는 결과적 출력 벡터 데이터 샘플 세트(292(0)-292(X))는 도 31의 VPE(22(6))의 출력 데이터 흐름 경로들(98(0)-98(X))에서 제공된다.

[0202] 도 31 및 도 32를 계속 참조하면, 결과적 출력 벡터 데이터 샘플 세트(292(0)-292(X))가 벡터 데이터 파일들(82(0)-82(X))에 저장되기 전에, 결과적 출력 벡터 데이터 샘플 세트(292(0)-292(X))가, 실행 유닛들(84(0)-84(X))과 벡터 데이터 파일들(82(0)-82(X)) 사이에 제공되는 출력 데이터 흐름 경로들(98(0)-98(X))에 제공되는 병합 회로(294)에 제공된다. 병합 회로(294)는, 아래에서 더 상세하게 논의되는 바와 같이, 벡터 명령이 벡터 데이터 파일들(82(0)-82(X))에 저장될 결과적 출력 벡터 데이터 샘플 세트(292(0)-292(X))로부터의 결과적 출력 벡터 데이터 샘플들(292)의 병합을 요구하면, 실행되는 벡터 명령에 따라 출력 데이터 흐름 경로들(98(0)-98(X))에 포함되도록 프로그램가능하다. 병합 회로(294)는, 결과적 출력 벡터 데이터 샘플 세트(292(0)-292(X))가 벡터 데이터 파일들(82(0)-82(X))에 저장되지 않고 실행되는 벡터 명령에 따라 결과적 출력 벡터 데이터 샘플 세트(292(0)-292(X))로부터의 결과적 출력 벡터 데이터 샘플들(292)을 병합한다(도 32의 블록(310)). 이러한 방식으로, 결과적 출력 벡터 데이터 샘플 세트(292(0)-292(X))는, 벡터 데이터 파일들(82(0)-82(X))에 먼저 저장되고, 재패치되고, 사후-프로세싱 연산에서 병합되고, 병합된 포맷으로 벡터 데이터 파일들(82(0)-82(X))에 저장될 필요(이들은 실행 유닛들(84(0)-84(X))에서 지연을 제공함)가 없다. 결과적 출력 벡터 데이터

샘플 세트(292(0)-292(X))가, 요구되는 사후-프로세싱을 병합하지 않고 병합된 결과적 출력 벡터 데이터 샘플 세트(296(0)-296(Z))로서 벡터 데이터 파일들(82(0)-82(X))에 저장된다(도 32의 블록(312)).

[0203] 도 33은 도 31의 VPE(22(6))의 실행 유닛들(84(0)-84(X))과 벡터 데이터 파일들(82(0)-82(X)) 사이의 출력 데이터 흐름 경로들(98(0)-98(X))에서 제공될 수 있는 예시적 병합 회로(294)의 개략도이다. 병합 회로(294)는 병합된 결과적 출력 벡터 데이터 샘플 세트(296(0)-296(Z))를 제공하기 위해서 결과적 출력 벡터 데이터 샘플 세트(292(0)-292(X))의 병합을 제공하도록 구성된다. 결과적 출력 벡터 데이터 샘플 세트(292(0)-292(X))는, 도 31에 예시되는 바와 같이, 실행 유닛 출력들(96(0)-96(X))로부터 병합 회로(294)에 제공된다.

[0204] 도 33을 계속 참조하면, 병합 회로(294)는 결과적 출력 벡터 데이터 샘플 세트(292(0)-292(X))를 병합하도록 구성된다. 이 실시예에서, 병합 회로(294)는 병합된 결과적 출력 벡터 데이터 샘플 세트(296(0)-296(Z))를 제공하도록 구성된다. 이와 관련하여, 병합 회로(294)는 결과적 출력 벡터 데이터 샘플 세트(292(0)-292(X))를 수신하기 위해서 실행 유닛 출력들(96(0)-96(X))에 커플링된 가산기 트리(318)를 포함한다. 병합 회로(294)의 가산기 트리(318)는 그들의 각각의 벡터 데이터 레인들(100(0)-100(X))에서 결과적 출력 벡터 데이터 샘플 세트(292(0)-292(X))의 각각의 샘플(292)을 수신하도록 구성된다. 제 1 가산기 트리 레벨(318(1))은 가산기 트리(318)에서 제공된다. 제 1 가산기 트리 레벨(318(1))은 결과적 출력 벡터 데이터 샘플 세트(292(0)-292(X))에서 인접 샘플들(292)을 병합할 수 있도록 병합 회로들(320(0)-320(((X+1)/2)-1), 320(7))로 구성된다. 래치들(321(0)-321(X))은 출력 데이터 흐름 경로들(98(0)-98(X))로부터의 결과적 출력 벡터 데이터 샘플 세트(292(0)-292(X))를 래치하도록 병합 회로(294)에서 제공된다.

[0205] 예컨대, 결과적 출력 벡터 데이터 샘플 세트(292(0)-292(X))에서의 각각의 샘플(292)이 32 비트 폭이며, 두개(2)의 16 비트 복소 벡터 데이터(즉, 포맷 I8Q8에 따른 제 1 벡터 데이터 및 포맷 I8Q8에 따른 제 2 벡터 데이터)로 구성되면, 병합 연산은 결과적 출력 벡터 데이터 샘플 세트(292(0)-292(X))에서의 두개(2)의 결과적 출력 벡터 데이터 샘플들(292)에서 네개(4)의 벡터 데이터 샘플들을 하나의 병합된 결과적 출력 벡터 데이터 샘플(296)로 병합하도록 적용될 수 있다. 예컨대, 도 33에 예시되는 바와 같이, 가산기(320(0))는 결과적 출력 벡터 데이터 샘플들(292(0) 및 292(1))을 병합하도록 구성된다. 마찬가지로, 가산기(320(1))는 이러한 샘플들에 대해 결과적 출력 벡터 데이터 샘플들(292(2) 및 292(3))을 병합하도록 구성된다. 가산기(320(((X+1)/2)-1), 320(7))는 병합 벡터 데이터 샘플 세트(322(0)-322(((X+1)/2)-1), 322(7))를 제공하기 위해서 결과적 출력 벡터 데이터 샘플 세트(292(X-1) 및 292(X))를 병합하도록 구성된다. 가산기들(320(((X+1)/2)-1), 320(7))에 의해 수행되는 병합으로부터의 병합 벡터 데이터 샘플 세트(322(0)-322(((X+1)/2)-1), 322(7))는 래치들(325(0)-325(((X+1)/2)-1), 325(7))로 래치된다.

[0206] 병합 벡터 프로세싱 연산(302)이 아래에서 더 상세하게 논의될 바와 같이, 결과적 출력 벡터 데이터 샘플 세트(292(0)-292(X))의 병합을 요구하면, 병합 벡터 데이터 샘플 세트(322(0)-322(((X+1)/2)-1), 322(7))는 병합된 결과적 출력 벡터 데이터 샘플 세트(296(0)-296(Z))로서 제공될 수 있으며, 여기서, 'Z'는 칠(7)이다. 그러나, 병합 벡터 프로세싱 연산(302)이 결과적 출력 벡터 데이터 샘플 세트(292(0)-292(X))에서의 비-인접 결과적 출력 벡터 데이터 샘플들(292)의 병합을 요구하면, 병합 벡터 데이터 샘플 세트(322(0)-322(((X+1)/2)-1), 322(7))는 병합된 결과적 출력 벡터 데이터 샘플 세트(296(0)-296(Z))로서 제공되지 않는다. 병합 벡터 데이터 샘플 세트(322(0)-322(((X+1)/2)-1), 322(7))는 가산기들(324(0)-324(((X+1)/4)-1), 324(3))에 대한 제 2 가산기 트리 레벨(318(2))에 제공된다. 이와 관련하여, 가산기(324(0))는 결과적 병합 벡터 데이터 샘플(326(0))을 제공하기 위해서 병합 벡터 데이터 샘플들(322(0) 및 322(1))에 대해 병합을 수행하도록 구성된다. 마찬가지로, 가산기(324(1))는 결과적 병합 벡터 데이터 샘플(326(1))을 제공하기 위해서 병합 벡터 데이터 샘플들(322(2) 및 322(3))에 대해 병합을 수행하도록 구성된다. 가산기(324(((X+1)/4)-1), 324(3))는 결과적 병합 벡터 데이터 샘플(326(((X+1)/4)-1), 326(3))을 제공하기 위해서 병합 벡터 데이터 샘플(322(((X+1)/4)-2), 322(((X+1)/4)-1), 322(3))에 대해 병합을 수행하도록 구성된다. 가산기들(324(0)-324(((X+1)/4)-1), 324(3))에 의해 수행되는 병합으로부터의 결과적 병합 벡터 데이터 샘플 세트(326(0)-326(((X+1)/4)-1), 326(3))는 래치들(327(0)-327(((X+1)/4)-1), 327(3))로 래치된다.

[0207] 도 33을 계속 참조하면, 병합 벡터 프로세싱 연산(302)이 아래에서 더 상세하게 논의될 바와 같이, 팔(8)의 병합 팩터에 의한 결과적 출력 벡터 데이터 샘플 세트(292(0)-292(X))의 병합을 요구하면, 병합 벡터 데이터 샘플 세트(326(0)-326(((X+1)/4)-1), 326(3))는 병합된 결과적 출력 벡터 데이터 샘플 세트(296(0)-296(Z))로서 제공될 수 있으며, 여기서, 'Z'는 삼(3)이다. 그러나, 병합 벡터 프로세싱 연산(302)이 팔(8)보다 높은 병합 팩터(예컨대, 16, 32, 64, 128, 256)를 요구하면, 병합 벡터 데이터 샘플 세트(326(0)-326(((X+1)/4)-1), 326(3))는 병합된 결과적 출력 벡터 데이터 샘플 세트(296(0)-296(Z))로서 제공되지 않는다. 병합 벡터 데이터

샘플 세트(326(0)-326(((X+1)/4)-1), 326(3))는 가산기들(328(0)-328(((X+1)/8)-1), 328(1))에 대한 제 3 가산기 트리 레벨(318(3))에 제공된다. 이와 관련하여, 가산기(328(0))는 십육(16)의 병합 팩터를 이러한 샘플들에 제공하기 위해서 병합 벡터 데이터 샘플들(326(0) 및 326(1))에 대해 병합을 수행하도록 구성된다. 마찬가지로, 가산기(328(1))는 십육(16)의 병합 팩터를 병합 벡터 데이터 샘플 세트(330(0)-330(((X+1)/8)-1), 330(1))에 제공하기 위해서 병합 벡터 데이터 샘플들(326(2) 및 326(3))에 대해 병합을 수행하도록 구성된다. 가산기들(328(0)-328(((X+1)/8)-1), 328(1))에 의해 수행되는 병합으로부터의 병합 벡터 데이터 샘플 세트(330(0)-330(((X+1)/8)-1), 330(1))는 래치들(329(0)-329(((X+1)/8)-1), 329(1))로 래치된다.

[0208] 도 33을 계속 참조하면, 병합 벡터 프로세싱 연산(302)이 아래에서 더 상세하게 논의될 바와 같이, 십육(16)의 병합 팩터에 의한 결과적 출력 벡터 데이터 샘플 세트(292(0)-292(X))의 병합을 요구하면, 병합 벡터 데이터 샘플 세트(330(0)-330(((X+1)/8)-1), 330(1))는 병합된 결과적 출력 벡터 데이터 샘플 세트(296(0)-296(Z))로서 제공될 수 있으며, 여기서, 'Z'는 일(1)이다. 그러나, 병합 벡터 프로세싱 연산(236)이 십육(16)보다 높은 병합 팩터(예컨대, 32, 64, 128, 256)를 요구하면, 병합 벡터 데이터 샘플 세트(330(0)-330(((X+1)/8)-1), 330(1))는 병합된 결과적 출력 벡터 데이터 샘플 세트(296(0)-296(Z))로서 제공되지 않는다. 병합 벡터 데이터 샘플 세트(330(0)-330(((X+1)/8)-1), 330(1))는 가산기(332)에 대한 제 4 가산기 트리 레벨(318(4))에 제공된다. 이와 관련하여, 가산기(332)는 삼십이(32)의 병합 팩터를 병합 벡터 데이터 샘플(334)에 제공하기 위해서 병합 벡터 데이터 샘플들(330(0) 및 330(1))에 대해 병합을 수행하도록 구성된다. 가산기(332)에 의해 수행되는 병합으로부터의 병합 벡터 데이터 샘플(334)은 래치들(336 및 338)로 래치된다.

[0209] 도 33을 계속 참조하면, 병합 벡터 프로세싱 연산(302)이 아래에서 더 상세하게 논의될 바와 같이, 삼십이(32)의 병합 팩터에 의한 결과적 출력 벡터 데이터 샘플 세트(292(0)-292(X))의 병합을 요구하면, 병합 벡터 데이터 샘플(334)은 병합된 결과적 출력 벡터 데이터 샘플(296)로서 제공될 수 있다. 그러나, 병합 벡터 프로세싱 연산(302)이 삼십이(32)보다 높은 병합 팩터(예컨대, 64, 128, 256)를 요구하면, 병합 벡터 데이터 샘플(334)은 병합된 결과적 출력 벡터 데이터 샘플 세트(296)로서 제공되지 않는다. 병합 벡터 데이터 샘플(334)은 벡터 데이터 파일(82)에 저장될 필요 없이 래치(338)로 래치된 상태를 유지한다. 또 다른 결과적 출력 벡터 데이터 샘플 세트(292(0)-292(X))는 위에서 설명된 바와 같이, 삼십-이(32)의 병합 팩터를 사용하여 병합될 부가적인 프로세싱 사이클들 상에서 래치들(321(0)-321(X))로 로딩된다. 결과적 병합 벡터 데이터 샘플(334')은 제 5 가산기 트리(318(5))에서의 가산기(340)에 의해, 육십-사(64)의 병합 팩터를 갖는 병합 벡터 데이터 샘플(342)을 제공하기 위해 이전 병합 벡터 데이터 샘플(334)에 가산된다. 선택기(343)는, 병합 벡터 데이터 샘플(342)이 래치(344)로 래치될 때, 삼십-이(32)의 병합 팩터를 갖는 병합 벡터 데이터 샘플(334)이 래치되는지 아니면 육십-사(64)의 병합 팩터를 갖는 병합 벡터 데이터 샘플(334')이 래치되는지를 제어한다. 부가적인 결과적 출력 벡터 데이터 샘플 세트들(292(0)-292(X))을 래치하고 이들을 병합하는 이러한 동일한 프로세스는 요구에 따라, 육십-사(64)보다 큰 병합 팩터들을 달성하도록 수행될 수 있다. 병합 벡터 데이터 샘플(342)은 결국, 병합 벡터 프로세싱 연산(302)에 대해 요구되는 병합 팩터에 따라 원하는 병합된 결과적 출력 벡터 데이터 샘플(296)로서 래치(344)로 래치될 것이다.

[0210] 도 33을 계속 참조하면, 어떠한 병합 벡터 프로세싱 연산이 병합 벡터 프로세싱 연산(302)에서 요구될지라도, 병합된 결과적 출력 벡터 데이터 샘플 세트(296(0)-296(Z))는 벡터 데이터 파일들(82(0)-82(X))에 저장될 필요가 있을 것이다. 이제 논의될 바와 같이, 도 33의 병합 회로(294)는 또한, 병합된 결과적 출력 벡터 데이터 샘플 세트(296(0)-296(Z))를 형성하기 위해, 결과적 출력 벡터 데이터 샘플들(292(0)-292(X))에 대해 병합 벡터 프로세싱 연산(302)을 수행한 결과로서 제공되는 병합된 결과적 출력 벡터 데이터 샘플들(296)을 래치들(346(0)-346(X))로 로딩하도록 구성된다. 병합된 결과적 출력 벡터 데이터 샘플 세트(296(0)-296(Z))는 저장될 벡터 데이터 파일들(82(0)-82(X))에 제공될 수 있다. 이러한 방식으로, 병합 회로(294)에 의해 생성되는 병합된 결과적 출력 벡터 데이터 샘플 세트(296(0)-296(Z))를 저장하기 위해 벡터 데이터 파일들(82(0)-82(X))에의 단지 한 번(1)의 기록이 요구된다. 도 33의 병합 회로(294)에서의 가산기 트리들(318(1)-318(5))은 어떤 병합 팩터가 병합 벡터 프로세싱 연산(302)에서 요구되는지에 관계없이, 병합 팩터들(4, 8, 16 및 32) 전부에 대해, 병합된 결과적 출력 벡터 데이터 샘플들(296)을 생성할 수 있다. 대안적으로, 원하는 병합 팩터에 따라 병합 벡터 프로세싱 연산(302)을 수행하는데 필요하지 않은 가산기 트리들에서의 가산기들은 디스에이블되거나, 또는 0들을 가산하도록 구성될 수 있다. 그러나, 이러한 병합된 결과적 출력 벡터 데이터 샘플들(296) 중 어떤 것이 저장될 래치들(346(0)-346(X))에 제공될 것인지를 결정하기 위해, 선택기들(348(0)-348(((X+1)/4)-1), 348(3))이 이제 논의되는 바와 같이 제공된다.

[0211] 이와 관련하여, 도 33을 계속 참조하면, 선택기(348(0))는 실행되고 있는 병합 벡터 프로세싱 연산(302)에 기초

하여, 가산기들(320(0), 324(0), 328(0)) 각각으로부터의 병합 팩터들(4, 8 및 16) 중 임의의 것에 대한 병합된 결과적 출력 벡터 데이터 샘플들(296) 및 가산기들(332, 340)로부터의 병합 팩터들(32, 64, 128, 256)을 선택할 수 있다. 선택기(348(1))는 실행되고 있는 병합 벡터 프로세싱 연산(302)에 기초하여, 가산기들(320(1), 324(1) 및 328(1)) 각각으로부터의 병합 팩터들(4, 8 및 16)에 대해 병합된 결과적 출력 벡터 데이터 샘플들(296)을 선택할 수 있다. 선택기(348(2))는 실행되는 병합 벡터 프로세싱 연산(302)에 기초하여, 가산기들(320(2) 및 324(2)) 각각으로부터의 병합 팩터들(4 및 8)에 대해 병합된 결과적 출력 벡터 데이터 샘플들(296)을 선택할 수 있다. 선택기(348(3))는 실행되는 병합 벡터 프로세싱 연산(302)에 기초하여, 가산기들(320(3) 및 324(3)) 각각으로부터의 병합 팩터들(4 및 8)에 대해 병합된 결과적 출력 벡터 데이터 샘플들(296)을 선택할 수 있다. 팔(8)의 병합 팩터를 제공하는 것이 선택기들(348(0)-348(3))에 의해 완전히 만족될 수 있기 때문에, 선택기들은 가산기들(320(4)-320(7))로부터 제공되는 병합된 결과적 출력 벡터 데이터 샘플들(296)을 제어하도록 제공되지 않는다.

[0212] 도 33을 계속 참조하면, 병합 벡터 프로세싱 연산들에 대해 제공되는 데이터 슬라이서들(350(0)-350(((X+1)/2)-1), 350(7))은 선택기들(348(0)-348(((X+1)/4)-1), 348(3)) 및 가산기들(320(4)-320(((X+1)/2)-1), 320(7)) 각각에 의해 선택되는 수신된 병합된 결과적 출력 벡터 데이터 샘플들(296)에 대해 데이터 스플라이싱을 수행하지 않도록 구성되거나 또는 바이패싱될 수 있다. 그 다음, 병합된 결과적 출력 벡터 데이터 샘플들(296)은 저장될 래치들(346(0)-346(X)) 사이의 원하는 래치(346)로의 크로스바(352)로 연결들을 통해 라우팅된다. 크로스바(352)는 병합 벡터 프로세싱 연산(302)에 따른 병합된 결과적 출력 벡터 데이터 샘플들(296)을 상이한 래치들(346(0)-346(X))에 저장하기 위해 유연성을 제공한다. 이러한 방식으로, 병합된 결과적 출력 벡터 데이터 샘플들(296)은 벡터 데이터 파일들(82(0)-82(X))에 저장되기 전에 병합 벡터 프로세싱 연산들(302)의 상이한 반복들 사이에서 래치들(346(0)-346(X))에 스택될 수 있다. 예컨대, 병합된 결과적 출력 벡터 데이터 샘플 세트(296(0)-296(Z))는 벡터 데이터 파일들(82(0)-82(X))에 저장되기 전에 병합 벡터 프로세싱 연산들(302)의 상이한 반복들 사이에서 래치들(346(0)-346(X))에 스택될 수 있다. 이러한 방식으로, 병합된 결과적 출력 벡터 데이터 샘플 세트(296(0)-296(Z))를 저장하기 위한 벡터 데이터 파일들(82(0)-82(X))로의 액세스들은 동작 효율성을 위해 최소화될 수 있다.

[0213] 예컨대, 도 33에 예시되는 바와 같이, 크로스바(352)에 커플링된 선택기들(354(0)-354(X))은 래치들(346(0)-346(X)) 중 임의의 것에 선택기(348(0))로부터의 병합된 결과적 출력 벡터 데이터 샘플(296)을 저장하도록 제어될 수 있다. 크로스바(352)에 커플링된 선택기들(354(1), 354(3), 354(5), 354(7), 354(9), 354(11), 354(13), 354(15))은 래치들(346(1), 346(3), 346(5), 346(7), 346(9), 346(11), 346(13) 및 346(15))에 저장될 선택기(348(1))로부터의 병합된 결과적 출력 벡터 데이터 샘플(296)을 저장하도록 제어될 수 있다. 크로스바(352)에 커플링된 선택기들(354(2), 354(6), 354(10), 354(14))은 래치들(346(2), 346(6), 346(10) 및 346(14))에 선택기(348(2))로부터의 병합된 결과적 출력 벡터 데이터 샘플(296)을 저장하도록 제어될 수 있다. 크로스바(352)에 커플링된 선택기들(354(3), 354(7), 354(11), 354(15))은 래치들(346(3), 346(7), 346(11) 및 346(15))에 선택기(348(3))로부터의 병합된 결과적 출력 벡터 데이터 샘플(296)을 저장하도록 제어될 수 있다. 크로스바(352)에 커플링된 선택기들(354(4) 및 354(12))은 래치들(346(4) 및 346(12))에 가산기(320(4))로부터의 병합된 결과적 출력 벡터 데이터 샘플(296)을 저장하도록 제어될 수 있다. 크로스바(352)에 커플링된 선택기들(354(5) 및 354(13))은 래치들(346(5) 및 346(13))에, 저장될 가산기(320(5))로부터의 병합된 결과적 출력 벡터 데이터 샘플(296)을 저장하도록 제어될 수 있다. 크로스바(352)에 커플링된 선택기들(354(6) 및 354(14))은 래치(346(6) 또는 346(14))에 가산기(320(6))로부터의 병합된 결과적 출력 벡터 데이터 샘플(296)을 저장하도록 제어될 수 있다. 크로스바(352)에 커플링된 선택기들(354(7) 및 354(15))은 래치들(346(7) 또는 346(15))에 가산기(320(7))로부터의 병합된 결과적 출력 벡터 데이터 샘플(296)을 저장하도록 제어될 수 있다.

[0214] 도 33에서의 병합 회로(294)에서, 가산기들은 결과적 출력 벡터 데이터 샘플 세트(292(0)-292(X))에서의 인접하지 않은 결과적 출력 벡터 데이터 샘플들(282)이 병합되게 허용하도록 구성될 수 있다는 것을 유의한다. 예컨대, 결과적 출력 벡터 데이터 샘플들(292(9))과 결과적 출력 벡터 데이터 샘플들(292(0))을 병합하도록 요구되는 경우에, 가산기 트리 레벨들(318(1)-318(3))에서의 가산기들은 결과적 출력 벡터 데이터 샘플들(292(9))과 결과적 출력 벡터 데이터 샘플들(292(0))의 병합을 가산기 트리 레벨(318(4))로 단순히 패스하도록 구성될 수 있다. 그 후에, 가산기 트리 레벨(318(4))에서의 가산기(332)는 병합된 출력 벡터 데이터 샘플들(296)을 제공하기 위해 결과적 출력 벡터 데이터 샘플들(292(9))과 결과적 출력 벡터 데이터 샘플(292(0))을 병합할 수 있다.

[0215] 또한, 벡터 및/또는 스칼라 가산 이외의 다른 타입들의 벡터 병합 연산들을 제공하는 병합 회로가 벡터 데이터

파일들(82(0)-82(X))과 실행 유닛들(84(0)-84(X)) 사이의 출력 데이터 흐름 경로들(98(0)-98(X))에 제공될 수 있다. 예컨대, 도 33에서의 병합 회로(294)는 최대 또는 최소 벡터 및/또는 스칼라 병합 연산들을 제공하도록 구성될 수 있다. 예컨대, 도 33에서의 가산기 트리(318)에서의 가산기 트리 레벨들(318(1)-318(5))에서의 가산기들은 최대 또는 최소 함수 회로로 대체될 수 있다. 즉, 회로는 결과적 출력 벡터 데이터 샘플 세트(292(0)-292(X))로부터의 2개의 결과적 출력 벡터 데이터 샘플들(292) 중 더 큰 것 또는 더 작은 것 중 어느 하나를 패스하도록 선택할 것이다. 예컨대, 결과적 출력 벡터 데이터 샘플 세트(292(0)-292(X))로부터의 2개의 결과적 출력 벡터 데이터 샘플들(292)이 도 30에서의 2개의 입력 벡터 데이터 샘플들(290(0), 290(1))이었던 경우에, 병합 회로(294)가 최대 벡터 데이터 샘플을 선택하도록 구성되는 경우, 병합 회로(294)는 벡터 데이터 샘플들(290(1))을 선택하도록 구성될 수 있다.

[0216] 이와 관련하여, 도 34를 참조하면, 도 34에서 예시된 바와 같이, 도 33에서의 제 1 가산기 트리 레벨(318(1))에서의 가산기들(320(0)-320(((X+1)/2)-1), 320(7))은 최대 또는 최소 병합 선택기 가산기들(320'(0)-320'(((X+1)/2)-1), 320'(7))로 대체될 수 있다. 도 34에서 예시된 바와 같이, 제 2 가산기 트리 레벨(318(2))에서의 가산기들(324(0)-324(((X+1)/4)-1), 324(3))은 최대 또는 최소 선택기들(324'(0)-324'(((X+1)/4)-1), 324'(3))로 대체될 수 있다. 도 34에서 예시된 바와 같이, 제 3 가산기 트리 레벨(318(3))에서의 가산기들(328(0)-328(((X+1)/8)-1), 328(1))은 최대 또는 최소 선택기들(328'(0)-328'(((X+1)/8)-1), 328'(1))로 대체될 수 있다. 도 34에서 예시된 바와 같이, 제 4 가산기 트리 레벨(318(4))에서의 가산기(332)는 최대 또는 최소 선택기(332')로 대체될 수 있다. 도 34에서 예시된 바와 같이, 제 5 가산기 트리 레벨(318(5))에서의 가산기(340)는 최대 또는 최소 선택기(340')로 대체될 수 있다. 도 34에서의 병합 회로(294)에서, 가산기들은 결과적 출력 벡터 데이터 샘플 세트(292(0)-292(X))에서의 인접하지 않은 결과적 출력 벡터 데이터 샘플들(292) 사이의 최대 또는 최소 결과적 출력 벡터 데이터 샘플(292)이 병합되게 선택하도록 구성될 수 있다는 것을 유의한다. 예컨대, 결과적 출력 벡터 데이터 샘플들(292(9))과 결과적 출력 벡터 데이터 샘플들(292(0))을 최대 병합하도록 요구되는 경우에, 가산기 트리 레벨들(318(1)-318(3))에서의 가산기들은 결과적 출력 벡터 데이터 샘플들(292(9))과 결과적 출력 벡터 데이터 샘플들(292(0))의 병합을 가산기 트리 레벨(318(4))로 단순히 패스하도록 구성될 수 있다. 그 후에, 가산기 트리 레벨(318(4))에서의 가산기(332')는 병합된 출력 벡터 데이터 샘플들(264)을 제공하기 위해 결과적 출력 벡터 데이터 샘플들(292(9))과 결과적 출력 벡터 데이터 샘플(292(0))을 최대 병합할 수 있다.

[0217] 위에서 논의된 바와 같이, 실행 유닛들(84(0)-84(X))이 입력 벡터 데이터 샘플 세트들(86(0)-86(X))에 대해 벡터 프로세싱 연산들을 수행하기 위해 VPE들(22(1)-22(6))에 제공된다. 실행 유닛들(84(0)-84(X))은 또한, 실행 유닛들(84(0)-84(X))이 상이한 벡터 프로세싱 연산들을 위한 공통 회로 및 하드웨어에 의한 연산의 다수의 모드들을 제공하게 허용하는 프로그램가능 데이터 경로 구성들을 포함한다. 실행 유닛들(84(0)-84(X)), 및 공통 회로 및 하드웨어에 의한 연산의 다수의 모드들을 제공하기 위한 이들의 프로그램가능 데이터 경로 구성들에 관한 더 예시적인 세부사항이 이제 논의된다.

[0218] 이와 관련하여, 도 35는 VPE들(22(1)-22(6))에서의 실행 유닛들(84(0)-84(X)) 각각에 대해 제공될 수 있는 예시적인 실행 유닛의 예시적인 개략도를 예시한다. 도 35에서 예시된 바와 같이 그리고 아래에서 도 36 내지 도 39에서 더 상세히 설명될 바와 같이, 실행 유닛(84)은 프로그램가능 데이터 경로 구성들로 구성될 수 있는 예시적인 벡터 프로세싱 블록들을 갖는 복수의 예시적인 벡터 파이프라인 스테이지들(460)을 포함한다. 아래에서 더 상세히 논의될 바와 같이, 벡터 프로세싱 블록들에 제공되는 프로그램가능 데이터 경로 구성들은, 특정한 회로들 및 하드웨어가, 도 2에서의 벡터 유닛 데이터 메모리(32)로부터 수신된 벡터 데이터(30)에 대해 상이한 특정한 벡터 프로세싱 연산들을 수행하는 것을 지원하도록 프로그램 및 재프로그램되게 허용한다.

[0219] 예컨대, 특정한 벡터 프로세싱 연산들은 일반적으로, 곱셈된 벡터 데이터 결과들의 누산이 뒤따르는 벡터 데이터(30)의 곱셈을 요구할 수 있다. 그러한 벡터 프로세싱의 비제한적인 예들은 무선 통신 알고리즘들에 대한 FFT(Fast Fourier Transform) 연산들을 수행하는데 일반적으로 사용되는 필터링 연산들, 상관 연산들, 및 래딕스-2 및 래딕스-4 버터플라이 연산들을 포함하며, 여기에서, 병렬 곱셈들의 시리즈에 뒤따라 곱셈 결과들의 병렬 누산들의 시리즈가 제공된다. 또한, 아래에서 도 39 및 도 40에 관하여 더 상세히 논의될 바와 같이, 도 35에서의 실행 유닛(84)은 또한, 캐리-절약 누산기들에 리던던트 캐리-절약 포맷을 제공하기 위해 곱셈기들과 캐리-절약 누산기들을 결합하는 옵션을 가진다. 캐리-절약 누산기들에 리던던트 캐리-절약 포맷을 제공하는 것은 누산의 각각의 단계 동안에 캐리 전파 경로 및 캐리 전파 가산 연산을 제공할 필요를 제거할 수 있다.

[0220] 이와 관련하여, 도 35를 더 참조하면, VPE(22)의 M0 곱셈 벡터 파이프라인 스테이지(460(1))가 먼저 설명될 것이다. M0 곱셈 벡터 파이프라인 스테이지(460(1))는, 각각이 프로그램가능 데이터 경로 구성들을 가지는, 임의

의 원하는 수의 곱셈기 블록들(462(A)-462(0))의 형태로 복수의 벡터 프로세싱 블록들을 포함하는 제 2 벡터 파이프라인 스테이지이다. 곱셈기 블록들(462(A)-462(0))은 실행 유닛(84)에서 벡터 곱셈 연산들을 수행하도록 제공된다. 복수의 곱셈기 블록들(462(A)-462(0))은 최대 열두개(12)의 곱셈 벡터 데이터 샘플 세트들(34(Y)-34(0))의 곱셈을 제공하기 위해 M0 곱셈 벡터 파이프라인 스테이지(460(1))에서 서로에 대해 병렬로 배치된다. 이 실시예에서, 'A'는 삼(3)과 동등하고, 이는, 이 예에서의 M0 곱셈 벡터 파이프라인 스테이지(460(1))에 네개(4)의 곱셈기 블록들(462(3)-462(0))이 포함되는 것을 의미한다. 곱셈 벡터 데이터 샘플 세트들(34(Y)-34(0))은, 실행 유닛(84)에서의 제 1 벡터 파이프라인 스테이지(460(0))인 입력 판독(RR) 벡터 파이프라인 스테이지에 제공된 복수의 래치들(464(Y)-464(0))로의 벡터 프로세싱을 위해 실행 유닛(84)에 로딩된다. 이 실시예에서 실행 유닛(84)에 열두개(12)의 래치들(464(11)-464(0))이 존재하고, 이는, 이 실시예에서 'Y'는 열하나(11)와 동등한 것을 의미한다. 래치들(464(11)-464(0))은 벡터 데이터 입력 샘플 세트들(466(11)-466(0))로서 벡터 레지스터들(도 2의 벡터 데이터 파일들(28)을 참조)로부터 리트리브된 곱셈 벡터 데이터 샘플 세트들(34(11)-34(0))을 래치하도록 구성된다. 이 예에서, 각각의 래치(464(11)-464(0))는 폭이 8 비트이다. 총 96 비트 폭의 벡터 데이터(30)(즉, 12 래치들 x 각 8비트)에 대해, 래치들(464(11)-464(0))은 각각 개별적으로, 곱셈 벡터 데이터 입력 샘플 세트들(466(11)-466(0))을 래치하도록 구성된다.

[0221] 도 35를 계속하여 참조하면, 복수의 곱셈기 블록들(462(3)-462(0))은 벡터 곱셈 연산들을 제공하기 위해 벡터 데이터 입력 샘플 세트들(466(11)-466(0))의 특정한 조합들을 수신할 수 있도록 구성되고, 여기에서, 이 예에서 'Y'는 열하나(11)와 동등하다. 곱셈 벡터 데이터 입력 샘플 세트들(466(11)-466(0))은 실행 유닛(84)의 디자인에 따라, 복수의 입력 데이터 경로들(A3-A0, B3-B0 및 C3-C0)에 제공된다. 도 35에서 예시된 바와 같이, 벡터 데이터 입력 샘플 세트들(466(3)-466(0))은 입력 데이터 경로들(C3-C0)에 대응한다. 도 35에서 예시된 바와 같이, 벡터 데이터 입력 샘플 세트들(466(7)-466(4))은 입력 데이터 경로들(B3-B0)에 대응한다. 도 35에서 예시된 바와 같이, 벡터 데이터 입력 샘플 세트들(466(11)-466(8))은 입력 데이터 경로들(A3-A0)에 대응한다. 복수의 곱셈기 블록들(462(3)-462(0))은 벡터 곱셈 연산들을 제공하기 위해, 복수의 곱셈기 블록들(462(3)-462(0))에 제공되는 입력 데이터 경로들(A3-A0, B3-B0, C3-C0) 각각에 따라, 수신된 벡터 데이터 입력 샘플 세트들(466(11)-466(0))을 프로세싱하도록 구성된다.

[0222] 도 37 및 도 38에 관하여 아래에서 더 상세히 논의될 바와 같이, 도 35에서의 곱셈기 블록들(462(3)-462(0))에 제공된 프로그램가능 내부 데이터 경로들(467(3)-467(0))은 상이한 데이터 경로 구성들을 가지도록 프로그램될 수 있다. 이러한 상이한 데이터 경로 구성들은 각각의 곱셈기 블록(462(3)-462(0))에 제공된 특정한 입력 데이터 경로들(A3-A0, B3-B0, C3-C0)에 따라 곱셈기 블록들(462(3)-462(0))에 제공된 특정한 수신된 벡터 데이터 입력 샘플 세트들(466(11)-466(0))의 곱셈의 상이한 조합들 및/또는 상이한 비트 길이들을 제공한다. 이와 관련하여, 복수의 곱셈기 블록들(462(3)-462(0))은 벡터 데이터 입력 샘플 세트들(466(11)-466(0))의 특정한 조합을 함께 곱셈하는 것의 곱셈 결과를 포함하는 벡터 결과 출력 샘플 세트로서 벡터 곱셈 출력 샘플 세트들(468(3)-468(0))을 제공한다.

[0223] 예컨대, 곱셈기 블록들(462(3)-462(0))의 프로그램가능 내부 데이터 경로들(467(3)-467(0))은 도 2에서의 베이스밴드 프로세서(20)의 명령 디스패치 회로(48)에서의 벡터 명령 디코더로부터 제공된 세팅들에 따라 프로그램될 수 있다. 이러한 실시예에서, 곱셈기 블록들(462(3)-462(0))의 네개(4)의 프로그램가능 내부 데이터 경로들(467(3)-467(0))이 존재한다. 벡터 명령은 실행 유닛(84)에 의해 수행될 특정한 타입의 연산을 특정한다. 따라서, 실행 유닛(84)은 매우 효율적인 방식으로 동일한 공통 회로에 의한 상이한 타입들의 벡터 곱셈 연산들을 제공하도록, 곱셈기 블록들(462(3)-462(0))의 프로그램가능 내부 데이터 경로들(467(3)-467(0))을 구성하도록 프로그램되고 재프로그램될 수 있다. 예컨대, 실행 유닛(84)은 명령 디스패치 회로(48)에서의 명령 파이프라인에서의 벡터 명령들의 디코딩에 따라, 실행되는 각각의 벡터 명령에 대해 클록-사이클-바이-클록-사이클 기초로 곱셈기 블록들(462(3)-462(0))의 프로그램가능 내부 데이터 경로들(467(3)-467(0))을 구성하고 재구성하도록 프로그램될 수 있다. 따라서, 실행 유닛(84)에서의 M0 곱셈 벡터 파이프라인 스테이지(460(1))가 매 클록 사이클마다 벡터 데이터 입력 샘플 세트들(466)을 프로세싱하도록 구성되는 경우에, 결과적으로, 곱셈기 블록들(462(3)-462(0))은 명령 디스패치 회로(48)에서의 명령 파이프라인에서의 벡터 명령들의 디코딩에 따라 매 클록 사이클마다 벡터 곱셈 연산들을 수행한다.

[0224] 곱셈기 블록들(462)은 실수 및 복소수 곱셈들을 수행하도록 프로그램될 수 있다. 도 35를 계속해서 참조하면, 하나의 벡터 프로세싱 블록 데이터 경로 구성에서, 곱셈기 블록(462)은 2개의 8 비트 벡터 데이터 입력 샘플 세트들(466)을 함께 곱셈하도록 구성될 수 있다. 하나의 곱셈 블록 데이터 경로 구성에서, 곱셈기 블록(462)은, 8 비트 벡터 데이터 입력 샘플 세트들(466)의 제 2 쌍에 의해 곱셈된 8 비트 벡터 데이터 입력 샘플 세트들

(466)의 제 1 쌍으로부터 형성된 2개의 16 비트 벡터 데이터 입력 샘플 세트들(466)을 함께 곱셈하도록 구성될 수 있다. 이는 도 38에서 예시되고, 아래에서 더 상세히 논의된다. 다시, 곱셈기 블록들(462(3)-462(0))에 프로그램가능 데이터 경로 구성들을 제공하는 것은, 곱셈기 블록들(462(3)-462(0))이 상이한 타입들의 곱셈 연산들을 수행하도록 구성되고 재구성될 수 있어서, 실행 유닛(84)에서의 영역을 감소시킬 수 있고, 가능하게는, 원하는 벡터 프로세싱 연산들을 수행하기 위해 베이스밴드 프로세서(20)에 더 적은 실행 유닛들(84)이 제공되게 허용할 수 있다는 점에서, 유연성을 제공한다.

[0225] 다시 도 35를 참조하면, 복수의 곱셈기 블록들(462(3)-462(0))은 다음 벡터 프로세싱 스테이지(460) 또는 출력 프로세싱 스테이지 중 어느 하나로의 프로그램가능 출력 데이터 경로들(470(3)-470(0))에 벡터 곱셈 출력 샘플 세트들(468(3)-468(0))을 제공하도록 구성된다. 복수의 곱셈기 블록들(462(3)-462(0))에 의해 실행되는 벡터 명령에 기초하여 프로그램된 구성에 따라, 프로그램가능 출력 데이터 경로들(470(3)-470(0))에 벡터 곱셈 출력 샘플 세트들(468(3)-468(0))이 제공된다. 이 예에서, 아래에서 논의될 바와 같이, 프로그램가능 출력 데이터 경로들(470(3)-470(0))에서의 벡터 곱셈 출력 샘플 세트들(468(3)-468(0))은 누산을 위해 M1 누산 벡터 파이프라인 스테이지(460(2))에 제공된다. 실행 유닛(84)의 이러한 특정 디자인에서, 곱셈된 결과들의 누산이 뒤따르는 벡터 데이터 입력들의 곱셈들을 필요로 하는 특정된 벡터 명령들을 지원하기 위해, 누산기들이 뒤따르는 복수의 곱셈기 블록들(462(3)-462(0))을 제공하는 것이 요구된다. 예컨대, FFT 연산들을 제공하기 위해 일반적으로 사용되는 래딕스-2 및 래딕스-4 버터플라이 연산들은 곱셈 결과들의 누산이 뒤따르는 곱셈 연산들의 시리즈를 포함한다. 그러나, 실행 유닛(84)에 제공되는 벡터 프로세싱 블록들의 이러한 조합들은 예시적이고, 제한적이지 않다는 것을 유의한다. 프로그램가능 데이터 경로 구성들을 가지는 VPE는 벡터 프로세싱 블록들을 가지는 하나 또는 임의의 다른 수의 벡터 프로세싱 스테이지들을 포함하도록 구성될 수 있다. 벡터 프로세싱 블록들은 실행 유닛에 의해 지원되도록 디자인된 특정한 벡터 명령들 및 디자인에 따라 임의의 타입의 연산들을 수행하도록 제공될 수 있다.

[0226] 도 35를 계속해서 참조하면, 이러한 실시예에서, 벡터 곱셈 출력 샘플 세트들(468(3)-468(0))은, M1 누산 벡터 프로세싱 스테이지(460(2))인 다음 벡터 프로세싱 스테이지에 제공되는 복수의 누산기 블록들(472(3)-472(0))에 제공된다. 복수의 누산기 블록들(472(A)-472(0)) 중 각각의 누산기 블록은 2개의 누산기들(472(X)(1) 및 472(X)(0))(즉, 472(3)(1), 472(3)(0), 472(2)(1), 472(2)(0), 472(1)(1), 472(1)(0), 및 472(0)(1), 472(0)(0))을 포함한다. 복수의 누산기 블록들(472(3)-472(0))은 벡터 곱셈 출력 샘플 세트들(468(3)-468(0))의 결과들을 누산한다. 아래에서 도 39 및 도 40에 관하여 더 상세히 논의될 바와 같이, 복수의 누산기 블록들(472(3)-472(0))은 캐리-절약 누산기들로서 제공될 수 있고, 여기에서, 누산 연산이 완료될 때까지, 누산 프로세스 동안에 캐리 프로덕트가 본질적으로 세이브되고, 전파되지 않는다. 복수의 누산기 블록들(472(3)-472(0))에 리던던트 캐리-절약 포맷을 제공하기 위해, 복수의 누산기 블록들(472(3)-472(0))은 또한, 도 35 및 도 37에서의 복수의 곱셈기 블록들(462(3)-462(0))과 결합되는 옵션을 가진다. 복수의 누산기 블록들(472(3)-472(0))에 리던던트 캐리-절약 포맷을 제공하는 것은, 누산의 각각의 단계 동안에 복수의 누산기 블록들(472(3)-472(0))에 캐리 전파 경로 및 캐리 전파 가산 연산을 제공할 필요성을 제거할 수 있다. M1 누산 벡터 프로세싱 스테이지(460(2)) 및 M1 누산 벡터 프로세싱 스테이지의 복수의 누산기 블록들(472(3)-472(0))이 이제 도 35를 참조하여 소개될 것이다.

[0227] 도 35를 참조하면, M1 누산 벡터 프로세싱 스테이지(460(2))에서의 복수의 누산기 블록들(472(3)-472(0))은, 다음 벡터 프로세싱 스테이지(460) 또는 출력 프로세싱 스테이지 중 어느 하나에 누산기 출력 샘플 세트들(476(3)-476(0))(즉, 476(3)(1), 476(3)(0), 476(2)(1), 476(2)(0), 476(1)(1), 476(1)(0), 및 476(0)(1), 476(0)(0))을 제공하기 위해, 프로그램가능 출력 데이터 경로 구성들에 따라, 프로그램가능 출력 데이터 경로들(474(3)-474(0))(즉, 474(3)(1), 474(3)(0), 474(2)(1), 474(2)(0), 474(1)(1), 474(1)(0), 및 474(0)(1), 474(0)(0))에서 벡터 곱셈 출력 샘플 세트들(468(3)-468(0))을 누산하도록 구성된다. 이러한 예에서, 누산기 출력 샘플 세트들(476(3)-476(0))은 ALU 프로세싱 스테이지(460(3))인 출력 프로세싱 스테이지에 제공된다. 예컨대, 아래에서 더 상세히 논의되는 바와 같이, 비제한적인 예로서, 누산기 출력 샘플 세트들(476(3)-476(0))은 또한 도 2에서의 베이스밴드 프로세서(20)에서의 스칼라 프로세서(44)에서의 ALU(46)에 제공될 수 있다. 예컨대, ALU(46)는 더 일반적인 프로세싱 연산들에서 사용되도록, 실행 유닛(84)에 의해 실행되는 특정된 벡터 명령들에 따라, 누산기 출력 샘플 세트들(476(3)-476(0))을 취할 수 있다.

[0228] 다시 도 35를 참조하면, 누산기 블록들(472(3)-472(0))의 프로그램가능 입력 데이터 경로들(478(3)-478(0)) 및/또는 프로그램가능 내부 데이터 경로들(480(3)-480(0))은, 곱셈기 블록들(462(3)-462(0))로부터 누산기 블록들(472(3)-472(0))로 제공된 벡터 곱셈 출력 샘플 세트들(468(3)-468(0))의 상이한 조합들 및/또는 비트 길이들을

수신하도록 재구성되도록 프로그램될 수 있다. 각각의 누산기 블록(472)이 2개의 누산기들(472(X)(1), 472(X)(0))로 구성되기 때문에, 프로그램가능 입력 데이터 경로들(478(A)-478(0))은 478(3)(1), 478(3)(0), 478(2)(1), 478(2)(0), 478(1)(1), 478(1)(0), 및 478(0)(1), 478(0)(0)로서 도 35에서 도시된다. 유사하게, 프로그램가능 내부 데이터 경로들(480(3)-480(A))은 480(3)(1), 480(3)(0), 480(2)(1), 480(2)(0), 480(1)(1), 480(1)(0), 480(0)(1), 480(0)(0)로서 도 35에서 도시된다. 누산기 블록들(472(3)-472(0))에 프로그램가능 입력 데이터 경로들(478(3)-478(0)) 및/또는 프로그램가능 내부 데이터 경로들(480(3)-480(0))을 제공하는 것은 도 39 및 도 40에 관하여 아래에서 더 상세히 논의된다. 이러한 방식으로, 누산기 블록들(472(3)-472(0))의 프로그램가능 입력 데이터 경로들(478(3)-478(0)) 및/또는 프로그램가능 내부 데이터 경로들(480(3)-480(0))에 따라, 누산기 블록들(472(3)-472(0))은 누산된 벡터 곱셈 출력 샘플 세트들(468(3)-468(0))의 프로그램된 조합에 따라 누산기 출력 샘플 세트들(476(3)-476(0))을 제공할 수 있다. 다시, 이는, 누산기 블록들(472(3)-472(0))이 프로그램가능 입력 데이터 경로들(478(3)-478(0)) 및/또는 프로그램가능 내부 데이터 경로들(480(3)-480(0))의 프로그래밍에 기초하여 상이한 타입들의 누산 연산들을 수행하도록 구성되고 재구성될 수 있어서, 실행 유닛(84)에서의 영역을 감소시킬 수 있고, 가능하게는, 원하는 벡터 프로세싱 연산들을 수행하기 위해 베이스밴드 프로세서(20)에 더 적은 실행 유닛들(84)이 제공되게 허용할 수 있다는 점에서, 유연성을 제공한다.

[0229] 예컨대, 하나의 누산기 모드 구성에서, 2개의 누산기 블록들(472)의 프로그램가능 입력 데이터 경로(478) 및/또는 프로그램가능 내부 데이터 경로들(480)은 비제한적인 예로서 단일 40 비트 누산기를 제공하도록 프로그램될 수 있다. 다른 누산기 모드 구성에서, 2개의 누산기 블록들(472)의 프로그램가능 입력 데이터 경로(478) 및/또는 프로그램가능 내부 데이터 경로(480)은 비제한적인 예로서 이중 24 비트 누산기들을 제공하도록 프로그램될 수 있다. 다른 누산기 모드 구성에서, 2개의 누산기 블록들(472)의 프로그램가능 입력 데이터 경로(478) 및/또는 프로그램가능 내부 데이터 경로(480)는 단일 24 비트 누산기가 뒤따르는 16 비트 캐리-절약 가산기를 제공하도록 프로그램될 수 있다. 곱셈들 및 누산 연산들의 특정한 상이한 조합들은 또한, 곱셈기 블록들(462(3)-462(0)) 및 누산기 블록들(472(3)-472(0))(예컨대, 16 비트 누산과 16 비트 복소수 곱셈, 및 16 비트 누산과 32 비트 복소수 곱셈)의 프로그래밍에 따라 실행 유닛(84)에 의해 지원될 수 있다.

[0230] 누산기 블록들(472(3)-472(0))의 프로그램가능 입력 데이터 경로들(478(3)-478(0)) 및/또는 프로그램가능 내부 데이터 경로들(480(3)-480(0))은 도 2에서의 베이스밴드 프로세서(20)의 명령 디스패치 회로(48)에서의 벡터 명령 디코더로부터 제공된 세팅들에 따라 프로그램될 수 있다. 벡터 명령은 실행 유닛(84)에 의해 수행될 특정한 타입의 연산을 특정한다. 따라서, 실행 유닛(84)은 명령 디스패치 회로(48)에서의 명령 파이프라인에서의 벡터 명령의 디코딩에 따라 실행되는 각각의 벡터 명령에 대해, 누산기 블록들(472(3)-472(0))의 프로그램가능 내부 데이터 경로들(480(3)-480(0)) 및/또는 프로그램가능 입력 데이터 경로들(478(3)-478(0))을 재프로그램하도록 구성될 수 있다. 벡터 명령은 실행 유닛(84)의 하나 또는 그 초과의 클록 사이클들에 걸쳐 실행될 수 있다. 또한 이 예에서, 실행 유닛(84)은 클록-사이클-바이-클록 사이클 기초로 벡터 명령의 각각의 클록 사이클 동안, 누산기 블록들(472(3)-472(0))의 프로그램가능 입력 데이터 경로들(478(3)-478(0)) 및/또는 프로그램가능 내부 데이터 경로들(480(3)-480(0))을 재프로그램하도록 구성될 수 있다. 따라서, 예컨대, 실행 유닛(84)에서의 M1 누산 벡터 프로세싱 스테이지(460(2))에 의해 실행되는 벡터 명령이 매 클록 사이클마다 벡터 곱셈 출력 샘플 세트들(468(3)-468(0))을 프로세싱하는 경우에, 결과적으로, 누산기 블록들(472(3)-472(0))의 프로그램가능 입력 데이터 경로들(478(3)-478(0)) 및/또는 프로그램가능 내부 데이터 경로들(480(3)-480(0))은 벡터 명령의 실행 동안에 매 클록 사이클마다 재구성될 수 있다.

[0231] 도 36은 예시적 벡터 프로세싱의 추가의 예시를 제공하기 위해 도 2 및 35의 실행 유닛(84)에서의 곱셈기 블록들(462(A)-462(0)) 및 누산기 블록들(472(A)(1)-472(0)(0))의 예시적 벡터 프로세싱을 예시하는 흐름도이다. 곱셈기 블록들(462(A)-462(0)) 및 누산기 블록들(472(A)(1)-472(0)(0)) 각각은 프로그램가능 데이터 경로 구성들을 가지며, 도 2 및 35의 예시적 실행 유닛(84)의 상이한 벡터 프로세싱 스테이지들에 제공된다. 예컨대, FFT 벡터 연산들은 누산 연산들이 뒤따르는 곱셈 연산들을 수반한다.

[0232] 이와 관련하여, 도 36에 관해, 벡터 프로세싱은 입력 프로세싱 스테이지(460(0))의 복수의 입력 데이터 경로들(A3-C0) 중 입력 데이터 경로에서 벡터 어레이의 폭의 복수의 곱셈 벡터 데이터 샘플 세트들(34(Y)-34(0))을 수신하는 단계를 수반한다(블록(501)). 그 다음으로, 벡터 프로세싱은 복수의 곱셈기 블록들(462(A)-462(0))에서 복수의 입력 데이터 경로들(A3-C0)로부터 곱셈 벡터 데이터 샘플 세트들(34(Y)-34(0))을 수신하는 단계를 포함한다(블록(503)). 그 다음으로, 벡터 프로세싱은 벡터 프로세싱 스테이지(460(1))에 의해 실행되는 벡터 명령에 따른 곱셈기 블록들(462(A)-462(0))에 대한 프로그램가능 데이터 경로 구성들에 기초하여, 복수의 곱셈 출력 데이터 경로들(470(A)-470(0)) 중 곱셈 출력 데이터 경로들(470(A)-470(0))에 곱셈 벡터 결과 출력 샘플 세트들

(468(A)-468(0))을 제공하기 위해 곱셈 벡터 데이터 샘플 세트들(34(Y)-34(0))을 곱셈하는 단계를 포함한다(블록(505)). 다음 벡터 프로세싱은 복수의 누산기 블록들(472(A)(1)-472(0)(0))에서 복수의 곱셈 출력 데이터 경로들(470(A)-470(0))로부터 곱셈 벡터 결과 출력 샘플 세트들(468(A)-468(0))을 수신하는 단계를 포함한다(블록(507)). 다음 벡터 프로세싱은 제 2 벡터 프로세싱 스테이지(460(2))에 의해 실행되는 벡터 명령에 따른 누산기 블록들(472(A)(1)-472(0)(0))에 대한 프로그램가능 입력 데이터 경로들(478(A)(1)-478(0)(0)), 프로그램가능 내부 데이터 경로들(480(A)(1)-480(0)(0)), 및 프로그램가능 출력 데이터 경로들(474(A)(1)-474(0)(0)) 구성들에 기초하여, 누산기 출력 샘플 세트들(476(A)(1)-476(0)(0))을 제공하기 위해서 곱셈 벡터 결과 출력 샘플 세트들(468(A)-468(0))을 함께 누산하는 단계를 포함한다(블록(509)). 그 다음으로, 벡터 프로세싱은 프로그램가능 출력 데이터 경로들(474(A)(1)-474(0)(0))에서 누산기 출력 샘플 세트들(476(A)(1)-476(0)(0))을 제공하는 단계를 포함한다(블록(511)). 그 다음으로, 벡터 프로세싱은 출력 벡터 프로세싱 스테이지(460(3))에서 누산기 블록들(472(A)(1)-472(0)(0))로부터 누산기 출력 샘플 세트들(476(A)(1)-476(0)(0))을 수신하는 단계를 포함한다(블록(513)).

[0233] 이제 도 35의 예시적인 실행 유닛(84) 및 프로그램가능 데이터 경로 구성들을 갖는 벡터 프로세싱 블록들을 이용한 도 36의 벡터 프로세싱의 개요가 설명되었고, 설명의 나머지는 도 37 내지 도 40의 이러한 벡터 프로세싱 블록들의 더 예시적이며, 비제한적인 세부사항들을 설명한다.

[0234] 이와 관련하여, 도 37은 도 35의 실행 유닛(84)의 M0 곱셈 벡터 프로세싱 스테이지(460(1))의 복수의 곱셈기 블록들(462(3)-462(0))의 더 상세한 개략도이다. 도 38은 도 37의 곱셈기 블록(462)의 내부 컴포넌트들의 개략도이다. 도 37에서 예시된 바와 같이, 특정 입력 데이터 경로들(A3-A0, B3-B0, C3-C0)에 따라 곱셈기 블록들(462(3)-462(0))에 의해 수신된 벡터 데이터 입력 샘플 세트들(466(11)-466(0))이 도시된다. 도 38과 관련하여 아래에서 더 상세하게 논의될 바와 같이, 이 예에서 곱셈기 블록들(462(3)-462(0)) 각각은 네(4) 개의 8 비트×8 비트 곱셈기들을 포함한다. 다시 도 37을 참조하면, 이 예에서 곱셈기 블록들(462(3)-462(0)) 각각은 피승수 입력 'A'에, 피승수 입력 'B' 또는 피승수 입력 'C' 중 하나를 곱셈하도록 구성된다. 곱셈기 블록(462)에서 모두 곱셈될 수 있는 피승수 입력들 'A' 및 'B' 또는 'C'는 도 37에서 도시된 바와 같이, 어느 입력 데이터 경로들(A3-A0, B3-B0, C3-C0)이 곱셈기 블록들(462(3)-462(0))에 연결되는지 에 의해 제어된다. 피승수 선택기 입력(482(3)-482(0))은, 피승수 입력 'A'에 피승수 입력 'B'가 곱셈되도록 선택되는지 또는 피승수 입력 'C'가 곱셈되도록 선택되는지를 선택하기 위해 각각의 곱셈기 블록(462(3)-462(0))에서 프로그램가능 내부 데이터 경로들(467(3)-467(0))을 제어하도록 각각의 곱셈기 블록(462(3)-462(0))에 입력으로서 제공된다. 이러한 방식으로, 곱셈기 블록들(462(3)-462(0))에는, 원하는 바에 따라 상이한 곱셈 연산들을 제공하도록 곱셈기 블록들의 프로그램가능 내부 데이터 경로들(467(3)-467(0))이 재프로그램되게 하는 능력이 제공된다.

[0235] 도 37을 계속해서 참조하면, 예로서 곱셈기 블록(462(3))을 사용하면, 입력 데이터 경로들(A3 및 A2)은 입력들(AH 및 AL)에 각각 연결된다. 입력(AH)은 피승수 입력(A)의 하이 비트(high bit)들을 나타내며, AL은 입력 피승수 입력('A')의 로우 비트(low bit)들을 의미한다. 입력 데이터 경로들(B3 및 B2)은 입력들(BH 및 BL)에 각각 연결된다. 입력(BH)은 피승수 입력('B')의 하이 비트들을 나타내며, AL은 입력 피승수 입력('B')의 로우 비트들을 나타낸다. 입력 데이터 경로들(C3 및 C2)은 입력들(CI 및 CQ)에 각각 연결된다. 입력(CI)은 이 예에서 입력 피승수 입력('C')의 실수 비트 부분을 나타낸다. CQ는 이 예에서 입력 피승수 입력('C')의 허수 비트 부분을 나타낸다. 도 38과 관련하여 아래에서 더 상세하게 논의될 바와 같이, 피승수 선택기 입력(482(3))은 또한 이 예에서, 곱셈기 블록(462(3))의 프로그램가능 내부 데이터 경로(467(3))가 피승수 입력('A')에 대해 피승수 입력('B') 또는 피승수 입력('C')과의 8 비트 곱셈을 수행하도록 구성되는지 여부, 또는 곱셈기 블록(462(3))이 피승수 입력('A')에 대해 피승수 입력('B') 또는 피승수 입력('C')과의 16 비트 곱셈을 수행하도록 구성되는지 여부를 제어한다.

[0236] 도 37을 계속해서 참조하면, 곱셈기 블록들(462(3)-462(0))은 곱셈기 블록들의 프로그램가능 내부 데이터 경로들(467(3)-467(0))의 구성에 기초하여 곱셈 연산의 캐리('C') 및 합('S') 벡터 출력 샘플 세트들로서 벡터 곱셈 출력 샘플 세트들(468(3)-468(0))을 각각 생성하도록 구성된다. 도 39 및 40과 관련하여 아래에서 더 상세하게 논의될 바와 같이, 벡터 곱셈 출력 샘플 세트들(468(3)-468(0))의 캐리('C') 및 합('S')은 결합되며, 이는 복수의 누산기 블록들(472(3)-472(0))에 리던던트 캐리-절약 포맷을 제공하기 위해, 캐리('C') 및 합('S')이 복수의 누산기 블록들(472(3)-472(0))에 리던던트 캐리-절약 포맷으로 제공되는 것을 의미한다. 아래에서 더 상세하게 논의될 바와 같이, 복수의 누산기 블록들(472(3)-472(0))에 리던던트 캐리-절약 포맷을 제공하는 것은 복수의 누산기 블록들(472(3)-472(0))에 의해 수행되는 누산 연산들 동안 캐리 전파 경로 및 캐리 전파 가산 연산을 제공할 필요성을 제거할 수 있다.

[0237] 곱셈기 블록들(462(3)-462(0))의 프로그램가능 내부 데이터 경로들(467(3)-467(0))의 구성에 기초하는 곱셈 연산의 캐리('C') 및 합('S') 벡터 출력 샘플 세트들로서 벡터 곱셈 출력 샘플 세트들(468(3)-468(0))을 생성하는 곱셈기 블록들(462(3)-462(0))의 예들이 도 37에 도시된다. 예컨대, 곱셈기 블록(462(3))은 8 비트 곱셈들에 대한 32 비트 값들로서 캐리(C00) 및 합(S00)을 생성하고 16 비트 곱셈들에 대한 64 비트 값들로서 캐리(C01) 및 합(S01)을 생성하도록 구성된다. 다른 곱셈기 블록들(462(2)-462(0))은 이 예에서 동일한 능력을 갖는다. 이와 관련하여, 곱셈기 블록(462(2))은 8 비트 곱셈들에 대한 32 비트 값들로서 캐리(C10) 및 합(S10)을 생성하고 16 비트 곱셈들에 대한 64 비트 값들로서 캐리(C11) 및 합(S11)을 생성하도록 구성된다. 곱셈기 블록(462(1))은 8 비트 곱셈들에 대한 32 비트 값들로서 캐리(C20) 및 합(S20)을 생성하고 16 비트 곱셈들에 대한 64 비트 값들로서 캐리(C21) 및 합(S21)을 생성하도록 구성된다. 곱셈기 블록(462(0))은 8 비트 곱셈들에 대한 32 비트 값들로서 캐리(C30) 및 합(S30)을 생성하고 16 비트 곱셈들에 대한 64 비트 값들로서 캐리(C31) 및 합(S31)을 생성하도록 구성된다.

[0238] 도 37의 곱셈기 블록(462)에 제공되는 프로그램가능 데이터 경로 구성들의 더 예시적인 세부사항을 설명하기 위해, 도 38이 제공된다. 도 38은 8 비트에 8 비트 벡터 데이터 입력 샘플 세트(466)를 곱셈하고 그리고 16 비트에 16 비트 벡터 데이터 입력 샘플 세트(466)를 곱셈하는 것이 가능한 프로그램가능 데이터 경로 구성들을 갖는 도 37의 곱셈기 블록(462)의 내부 컴포넌트들의 개략도이다. 이와 관련하여, 곱셈기 블록(462)은 이 예에서 4개의 8x8 비트 곱셈기들(484(3)-484(0))을 포함한다. 임의의 바람직한 수의 곱셈기들(484)이 제공될 수 있다. 제 1 곱셈기(484(3))는 8 비트 벡터 데이터 입력 샘플 세트(466A[H])(이는 입력 피승수 입력('A')의 하이 비트들임)를 수신하고, 벡터 데이터 입력 샘플 세트(466A[H])를 8 비트 벡터 데이터 입력 샘플 세트(466B[H])(이는 입력 피승수 입력('B')의 하이 비트들임) 또는 8 비트 벡터 데이터 입력 샘플 세트(466C[I])(이는 입력 피승수 입력('C')의 하이 비트들임) 중 하나와 곱셈하도록 구성된다. 곱셈기(484(3))에 피승수로서 제공되는 8 비트 벡터 데이터 입력 샘플 세트(466B[H]) 또는 8 비트 벡터 데이터 입력 샘플 세트(466C[I]) 중 하나를 선택하도록 구성된 멀티플렉서(486(3))가 제공된다. 멀티플렉서(486(3))는 이 실시예에서 피승수 선택기 입력(482)에서 하이 비트인 피승수 선택기 입력(482[3])에 의해 제어된다. 이러한 방식으로, 8 비트 벡터 데이터 입력 샘플 세트(466B[H]) 또는 8 비트 벡터 데이터 입력 샘플 세트(466C[I])가, 수신된 벡터 데이터 입력 샘플 세트(466A[H])와 곱셈되는지 여부를 제어하기 위해, 멀티플렉서(486(3)) 및 피승수 선택기 입력(482[3])은 곱셈기(484(3))에 대한 프로그램가능 내부 데이터 경로(467[0]) 구성을 제공한다.

[0239] 도 38을 계속해서 참조하면, 다른 곱셈기들(484(2)-484(0))은 또한 제 1 곱셈기(484(3))를 위해 제공된 것과 유사한 프로그램가능 내부 데이터 경로들(467[2]-467[0])을 포함한다. 곱셈기(484(2))는 피승수 입력('A')의 로우 비트들인 8 비트 벡터 데이터 입력 샘플 세트(466A[L])와 곱셈될 8 비트 벡터 데이터 입력 샘플 세트(466B[H]) 또는 8 비트 벡터 데이터 입력 샘플 세트(466C[I]) 중 하나를 프로그램가능 내부 데이터 경로(467[1])에 제공하기 위해 프로그램가능 구성을 갖는 프로그램가능 내부 데이터 경로(467[2])를 포함한다. 이 실시예에서, 선택은 피승수 선택기 입력(482)의 피승수 선택기 입력(482[2])에 따라 멀티플렉서(486(2))에 의해 제어된다. 곱셈기(484(1))는 8 비트 벡터 데이터 입력 샘플 세트(466A[H])와 곱셈될 피승수 입력('B')의 로우 비트들인 8 비트 벡터 데이터 입력 샘플 세트(466B[L]) 또는 피승수 입력('C')의 로우 비트들인 8 비트 벡터 데이터 입력 샘플 세트(466C[Q]) 중 하나를 프로그램가능 내부 데이터 경로(467[1])에 제공하도록 프로그램가능한 프로그램가능 내부 데이터 경로(467[1])를 포함한다. 이 실시예에서, 선택은 피승수 선택기 입력(482)의 피승수 선택기 입력(482[1])에 따라 멀티플렉서(486(1))에 의해 제어된다. 게다가, 곱셈기(484(0))는 8 비트 벡터 데이터 입력 샘플 세트(466A[L])와 곱셈될 8 비트 벡터 데이터 입력 샘플 세트(466B[L]) 또는 8 비트 벡터 데이터 입력 샘플 세트(466C[Q]) 중 하나를 프로그램가능 내부 데이터 경로(467[0])에 제공하도록 프로그램가능한 프로그램가능 내부 데이터 경로(467[0])를 포함한다. 이 실시예에서, 선택은 피승수 선택기 입력(482)의 피승수 선택기 비트 입력(482[0])에 따라 멀티플렉서(486(0))에 의해 제어된다.

[0240] 도 38을 계속해서 참조하면, 위에서 논의된 바와 같이, 곱셈기들(484(3)-484(0))은 상이한 비트 길이 곱셈 연산들을 수행하도록 구성될 수 있다. 이와 관련하여, 각각의 곱셈기(484(3)-484(0))는 비트 길이 곱셈 모드 입력들(488(3)-488(0))을 각각 포함한다. 이 예에서, 각각의 곱셈기(484(3)-484(0))는 각각 프로그램가능 데이터 경로들(490(3)-490(0), 491, 및 492(3)-492(0))의 구성을 제어하는 입력들에 따라, 8 비트×8 비트 모드로 프로그램될 수 있다. 각각의 곱셈기(484(3)-484(0))는 또한, 각각 프로그램가능 데이터 경로들(490(3)-490(0), 491, 및 492(3)-492(0))의 구성을 제어하는 입력들에 따라, 16 비트×16 비트 모드 및 24 비트×8 비트 모드를 포함한 더 큰 비트 곱셈 연산의 일부를 제공하도록 프로그램될 수 있다. 예컨대, 각각의 곱셈기(484(3)-484(0))가 프로그램가능 데이터 경로들(490(3)-490(0))의 구성에 따라 8 비트×8 비트 곱셈 모드로 구성되는 경

우, 유닛으로서 복수의 곱셈기들(484(3)-484(0))은 곱셈기 블록(462)의 부분으로서 두(2) 개의 개별적인 8 비트×8 비트 곱셈기들을 포함하도록 구성될 수 있다. 각각의 곱셈기(484(3)-484(0))가 프로그램가능 데이터 경로(491)의 구성에 따라 16 비트×16 비트 곱셈 모드로 구성되는 경우, 유닛으로서 복수의 곱셈기들(484(3)-484(0))은 곱셈기 블록(462)의 부분으로서 단일의 16 비트×16 비트 곱셈기를 포함하도록 구성될 수 있다. 곱셈기들(484(3)-484(0))이 프로그램가능 데이터 경로들(492(3)-492(0))의 구성에 따라 24 비트×8 비트 곱셈 모드로 구성되는 경우, 유닛으로서 복수의 곱셈기들(484(3)-484(0))은 곱셈기 블록(462)의 부분으로서 하나(1)의 16 비트×24 비트×8 비트 곱셈기를 포함하도록 구성될 수 있다.

[0241] 도 38을 계속해서 참조하면, 이 예에서의 곱셈기들(484(3)-484(0))은 16 비트×16 비트 곱셈 모드로 구성되는 것으로 도시된다. 십육(16) 비트 입력 합들(494(3), 494(2)) 및 입력 캐리들(496(3), 496(2))은 각각의 곱셈기(484(3), 484(2))에 의해 각각 생성된다. 십육(16) 비트 입력 합들(494(1), 494(0)) 및 입력 캐리들(496(1), 496(0))은 각각의 곱셈기(484(1), 484(0))에 의해 각각 생성된다. 입력 합들(494(3)-494(0)) 및 입력 캐리들(496(3)-496(0))을 함께 가산하기 위해, 16 비트 입력 합들(494(1), 494(0)) 및 입력 캐리들(496(1), 496(0))과 함께 16 비트 입력 합들(494(3), 494(2)) 및 입력 캐리들(496(3), 496(2))이 또한 24 비트 4:2 압축기(515)에 제공된다. 프로그램가능 데이터 경로(491)가 입력 합들(494(3)-494(0)) 및 입력 캐리들(496(3)-496(0))과 함께 활성화 및 게이팅될 때, 가산된 입력 합들(494(3)-494(0)) 및 입력 캐리들(496(3)-496(0))은 단일의 합(498) 및 단일의 캐리(500)를 16 비트×16 비트 곱셈 모드로 제공한다. 프로그램가능 데이터 경로(491)는 24 비트 4:2 압축기(515)에 제공되도록 16 비트 워드로서 조합된 입력 합들(494(3), 494(2))을 갖는 제 1 AND-기반 게이트(502(3))에 의해 그리고 16 비트 워드로서 조합된 입력 캐리들(496(3), 496(2))을 갖는 제 2 AND-기반 게이트(502(2))에 의해 게이팅된다. 프로그램가능 데이터 경로(491)는 또한, 24 비트 4:2 압축기(515)에 제공되도록 16 비트 워드로서 조합된 입력 합들(494(1), 494(0))을 갖는 제 3 AND-기반 게이트(502(1))에 의해 그리고 16 비트 워드로서 조합된 입력 캐리들(496(1), 496(0))을 갖는 제 4 AND-기초 게이트(502(0))에 의해 게이팅된다. 곱셈기 블록(462)이 16 비트×16 비트 또는 24 비트×8 비트 곱셈 모드로 구성되는 경우, 프로그램가능 출력 데이터 경로(470[0])에는 압축된 32 비트 합(S0) 및 32 비트 캐리(C0) 부분 곱(partial product)으로서 벡터 곱셈 출력 샘플 세트(468[0])가 제공된다.

[0242] 곱셈기 블록(462)의 곱셈기들(484(3)-484(0))이 8 비트×8 비트 곱셈 모드로 구성되는 경우, 프로그램가능 출력 데이터 경로(470[1]) 구성은 압축 없이 부분 곱들로서 16 비트 입력 합들(494(3)-494(0)) 및 대응하는 16 비트 입력 캐리들(496(3)-496(0))로서 제공된다. 곱셈기 블록(462)의 곱셈기들(484(3)-484(0))이 8 비트×8 비트 곱셈 모드로 구성되는 경우, 프로그램가능 출력 데이터 경로(470[1])는 압축 없이 벡터 곱셈 출력 샘플 세트들(468[1])로서 16 비트 입력 합들(494(3)-494(0)) 및 대응하는 16 비트 입력 캐리들(496(3)-496(0))로서 제공된다. 곱셈기 블록(462)의 곱셈 모드에 의존하는 벡터 곱셈 출력 샘플 세트들(468[0], 468[1])은 실행되는 벡터 명령에 따라 합 및 캐리 곱들의 누산을 위해 누산기 블록들(472(3)-472(0))에 제공된다.

[0243] 이제 프로그램가능 데이터 경로 구성들을 갖는 도 37 및 38의 곱셈기 블록들(462(3)-462(0))이 설명되었고, 리던던트 캐리-절약 포맷으로 구성된 누산기 블록들(472(3)-472(0))과 결합될 실행 유닛(84)의 곱셈기 블록들(462(3)-462(0))의 특징들이 이제 도 39와 관련하여 일반적으로 설명된다.

[0244] 이와 관련하여, 도 39는 위에서 설명된 실행 유닛들(84(0)-84(X))의 곱셈기 블록 및 누산기 블록의 일반화된 개략도이며, 누산기 블록은 캐리 전파를 감소시키기 위해 리던던트 캐리-절약 포맷을 이용하는 캐리-절약 누산기 구조를 이용한다. 이전에 논의되고 도 38에서 도시된 바와 같이, 곱셈기 블록들(462)은, 피승수 입력들(466[H] 및 466[L])을 곱셈하고, 벡터 곱셈 출력 샘플 세트들(468)로서 적어도 하나의 입력 합(494) 및 적어도 하나의 입력 캐리(496)를 프로그램가능 출력 데이터 경로(470)에 제공하도록 구성된다. 각각의 누산 단계에 있어서 누산기 블록(472)에 캐리 전파 경로 및 캐리 전파 가산기를 제공할 필요성을 제거하기 위해, 프로그램가능 출력 데이터 경로(470)의 벡터 곱셈 출력 샘플 세트들(468)의 적어도 하나의 입력 합(494) 및 적어도 하나의 입력 캐리(496)가 리던던트 캐리-절약 포맷으로 적어도 하나의 누산기 블록(472)에 결합된다. 다시 말해서, 벡터 곱셈 출력 샘플 세트들(468)의 캐리(496)는 캐리-절약 포맷의 벡터 입력 캐리(496)로서 누산기 블록(472)에 제공된다. 이러한 방식으로, 벡터 곱셈 출력 샘플 세트들(468)의 입력 합(494) 및 입력 캐리(496)는 누산기 블록(472)의 압축기(508)에 제공될 수 있으며, 이 실시예에서 압축기(508)는 복소 게이트 4:2 압축기이다. 압축기(508)는 입력 합(494) 및 입력 캐리(496)를 각각 이전의 누산된 벡터 출력 합(512) 및 이전의 시프트된 누산된 벡터 출력 캐리(517)와 함께 누산하도록 구성된다. 이전의 시프트된 누산된 벡터 출력 캐리(517)는 본질적으로, 누산 연산 동안 절약된 캐리 누산이다.

[0245] 이러한 방식으로, 누산기 블록(472)에 의해 생성된 누산의 부분으로서 입력 합(494)에 수신된 입력 캐리(496)를

전파하기 위해, 단지 단일의 최종 캐리 전파 가산기만이 누산기 블록(472)에 제공되도록 요구된다. 이 실시예에서, 누산기 블록(472)에서 각각의 누산 단계 동안 캐리 전파 가산 연산을 수행하는 것과 연관된 전력 소비가 감소된다. 또한, 이 실시예에서, 누산기 블록(472)에서 각각의 누산 단계 동안 캐리 전파 가산 연산을 수행하는 것과 연관된 게이트 지연이 또한 제거된다.

[0246] 도 39를 계속해서 참조하면, 압축기(508)는 리던던트 형태의 입력 합(494) 및 입력 캐리(496)를 각각 이전의 누산된 벡터 출력 합(512) 및 이전의 시프트된 누산된 벡터 출력 캐리(517)와 누산하도록 구성된다. 시프트된 누산된 벡터 출력 캐리(517)는, 다음 수신되는 입력 합(494) 및 입력 캐리(496)의 다음 누산이 압축기(508)에 의해 수행되기 전에, 누산된 벡터 출력 캐리(514)를 시프트함으로써 압축기(508)에 의해 생성된 누산된 벡터 출력 캐리(514)에 의해 생성된다. 최종 시프트된 누산된 벡터 출력 캐리(517)는, 최종 누산된 벡터 출력 합(512)을 최종 누산기 출력 샘플 세트(476) 2의 보수 표기법으로 변환하기 위해, 캐리 누산을 최종 시프트된 누산된 벡터 출력 캐리(517)에 전파하도록 누산기 블록(472)에 제공된 단일의 최종 캐리 전파 가산기(519)에 의해 최종 누산된 벡터 출력 합(512)에 가산된다. 최종 누산된 벡터 출력 합(512)은 프로그램가능 출력 데이터 경로(474)(도 35 참조)에 누산기 출력 샘플 세트(476)로서 제공된다.

[0247] 이제 리던던트 캐리-절약 포맷으로 구성된 누산기 블록(472)을 갖는 곱셈기 블록들(462)의 결합을 예시하는 도 39가 설명되었고, 누산기 블록들(472(3)-472(0))에 관한 더 예시적인 세부사항이 이제 도 40과 관련하여 일반적으로 설명된다. 도 40은 도 35의 실행 유닛(84)에 제공된 누산기 블록(472)의 예시적인 내부 컴포넌트들의 상세한 개략도이다. 이전에 논의되고 아래에서 더 상세하게 논의되는 바와 같이, 누산기 블록(472)이 프로그램가능 입력 데이터 경로들(478(3)-478(0)) 및/또는 프로그램가능 내부 데이터 경로들(480(3)-480(0))로 구성되어서, 누산기 블록(472)은 벡터 누산 연산들의 특정한, 상이한 유형들을 수행하도록 설계된 전용 회로로서 동작하도록 프로그램될 수 있다. 예컨대, 누산기 블록(472)은, 부호가 있는 그리고 부호가 없는 누산 연산들을 포함한 다수의 상이한 누산들 및 가산들을 제공하도록 프로그램될 수 있다. 상이한 유형들의 누산 연산들을 제공하도록 구성되는 누산기 블록(472)의 프로그램가능 입력 데이터 경로들(478(3)-478(0)) 및/또는 프로그램가능 내부 데이터 경로들(480(3)-480(0))의 특정 예들이 개시된다. 또한, 누산기 블록(472)은, 감소된 조합 로직을 갖는 고속 누산 연산들을 제공하기 위해 캐리 전파를 회피하거나 또는 감소시키도록 리던던트 캐리 산술을 제공하기 위해 캐리-절약 누산기들(472[0], 472[1])을 포함하도록 구성된다.

[0248] 누산기 블록(472)의 예시적인 내부 컴포넌트들이 도 40에 도시된다. 도면에 예시된 바와 같이, 이러한 실시예에서의 누산기 블록(472)은, 제 1 입력 합(494[0]) 및 제 1 입력 캐리(496[0]), 및 곱셈기 블록(462)으로부터의 제 2 입력 합(494[1]) 및 제 2 입력 캐리(496[1])를 수신하여 함께 누산되게 하도록 구성된다. 도 40과 관련하여, 입력 합들(494[0], 494[1]) 및 입력 캐리들(496[0], 496[1])은 벡터 입력 합들(494[0], 494[1]) 및 벡터 입력 캐리들(496[0], 496[1])로 지칭될 것이다. 앞서 설명되고 도 39에서 예시된 바와 같이, 이러한 실시예에서의 벡터 입력 합들(494[0], 494[1]) 및 벡터 입력 캐리들(496[0], 496[1]) 각각은 길이가 16 비트이다. 이러한 예에서의 누산기 블록(472)은 2개의 24 비트 캐리-절약 누산기 블록들(472[0], 472[1])로서 제공되며, 각각은 공통 엘리먼트 넘버들을 갖는 유사한 컴포넌트들을 포함하는데, '[0]'은 캐리-절약 누산기 블록(472[0])에 지정되고 '[1]'은 캐리-절약 누산기 블록(472[1])에 대해 지정된다. 캐리-절약 누산기 블록들(472[0], 472[1])은 벡터 누산 연산들을 동시에 수행하도록 구성될 수 있다.

[0249] 도 40의 캐리-절약 누산기 블록(472[0])을 참조하면, 벡터 입력 합(494[0]) 및 벡터 입력 캐리(496[0])는, 프로그램가능 내부 데이터 경로(480[0])의 일부로서 제공되는 멀티플렉서(504(0))의 입력이다. 배타적 OR-기반 게이트들로 구성될 수 있는 부정(negation) 회로(506(0))는 또한, 네거티브 벡터 입력 합(494[0]') 및 네거티브 벡터 입력 캐리(496[0]')를 요구하는 누산 연산들에 대한 멀티플렉서(504(0))로의 입력들로서, 입력(521(0))에 따라 네거티브 벡터 입력 합(494[0]') 및 네거티브 벡터 입력 캐리(496[0]')를 생성하도록 제공된다. 멀티플렉서(504(0))는, 벡터 명령 디코딩의 결과로서 생성되는 선택기 입력(510(0))에 따라 압축기(508(0))에 제공될, 벡터 입력 합(494[0]) 및 벡터 입력 캐리(496[0]) 또는 네거티브 벡터 입력 합(494[0]') 및 네거티브 벡터 입력 캐리(496[0]') 중 어느 하나를 선택하도록 구성된다. 이와 관련하여, 선택기 입력(510(0))은, 캐리-절약 누산기 블록(472[0])의 프로그램가능 입력 데이터 경로(478[0])가, 누산기 블록(472)에 의해 수행되도록 구성되는 누산 연산에 따라 벡터 입력 합(494[0]) 및 벡터 입력 캐리(496[0]) 또는 네거티브 벡터 입력 합(494[0]') 및 네거티브 벡터 입력 캐리(496[0]') 중 어느 하나를 압축기(508(0))에 제공하도록 프로그램가능하게 한다.

[0250] 도 40을 계속 참조하면, 이러한 실시예에서의 캐리-절약 누산기 블록(472[0])의 압축기(508(0))는, 복소 게이트 4:2 압축기이다. 이와 관련하여, 압축기(508(0))는, 리던던트 캐리-절약 연산들에서 합들 및 캐리들을 누산하도록 구성된다. 압축기(508(0))는, 현재 벡터 입력 합(494[0]) 및 벡터 입력 캐리(496[0]), 또는 현재 네거티

브 벡터 입력 합(494[0]') 및 네거티브 벡터 입력 캐리(496[0]')를 이전의 누산된 벡터 입력 합(494[0]) 및 벡터 입력 캐리(496[0]), 또는 누산된 네거티브 벡터 입력 합(494[0]') 및 네거티브 벡터 입력 캐리(496[0]')와 함께 압축기(508(0))에 대한 네개(4)의 입력들로서 함께 누산하도록 구성된다. 압축기(508(0))는, 누산기 출력 샘플 세트들(476(3)-476(0))을 제공하기 위해, 누산된 벡터 출력 합(512(0)) 및 누산된 벡터 출력 캐리(514(0))를 프로그램가능 출력 데이터 경로(474[0])(도 35 참조)에서의 누산기 출력 샘플 세트(476[0])로서 제공한다. 누산된 벡터 출력 캐리(514(0))는, 시프트된 누산 벡터 출력 캐리(517(0))를 제공하여 각각의 누산 단계 동안 비트 폭 증가를 제어하기 위해, 누산 연산들 동안 비트 시프터(516(0))에 의해 시프트된다. 예컨대, 이러한 실시예에서의 비트 시프터(516(0))는 리던던트 캐리-절약 포맷에서 압축기(508(0))에 결합되는 배럴-시프터(barrel-shifter)이다. 이러한 방식으로, 시프트된 누산된 벡터 출력 캐리(517(0))는, 누산기 블록(472[0])에 의해 수행되는 누산 연산 동안, 누산된 벡터 출력 합(512(0))에 전파되어야 할 필요 없이 본질적으로 절약된다. 이러한 방식으로, 누산기 블록(472[0])에서의 각각의 누산 단계 동안 캐리 전파 부가 연산을 수행하는 것과 연관된 전력 소비 및 게이트 지연은 이러한 실시예에서 제거된다.

[0251] 부가적인 후속 벡터 입력 합들(494[0]) 및 벡터 입력 캐리들(496[0]) 또는 네거티브 벡터 입력 합들(494[0]') 및 네거티브 벡터 입력 캐리들(496[0]')은 현재 누산된 벡터 출력 합(512(0)) 및 현재 누산된 벡터 출력 캐리(517(0))에 누산될 수 있다. 벡터 입력 합들(494[0]) 및 벡터 입력 캐리들(496[0]), 또는 네거티브 벡터 입력 합들(494[0]') 및 네거티브 벡터 입력 캐리들(496[0]')은, 벡터 명령 디코딩의 결과로서 생성되는 합-캐리 선택기(520(0))에 따라 프로그램가능 내부 데이터 경로(480[0])의 일부로서 멀티플렉서(518(0))에 의해 선택된다. 현재 누산된 벡터 출력 합(512(0)) 및 현재 시프트된 누산된 벡터 출력 캐리(517(0))는, 업데이트된 누산된 벡터 출력 합(512(0)) 및 누산된 벡터 출력 캐리(514(0))를 제공하기 위해, 캐리-절약 누산기 블록(472[0])에 대한 압축기(508(0))에 입력들로서 제공될 수 있다. 이와 관련하여, 합-캐리 선택기(520(0))는, 누산기 블록(472[0])의 프로그램가능 내부 데이터 경로(480[0])가, 누산기 블록(472)에 의해 수행되도록 구성되는 누산 연산에 따라 벡터 입력 합(494[0]) 및 벡터 입력 캐리(496[0])를 압축기(508(0))에 제공하도록 프로그램가능하게 한다. 이러한 실시예에서, 멀티플렉서(518(0))로 하여금 홀드(hold) 상태 입력(526(0))에 따라, 누산된 벡터 출력 합(512(0)) 및 시프트된 누산된 벡터 출력 캐리(517(0))의 현재 상태를 유지하게 하여, 캐리-절약 누산기 블록(472[0])에서의 누산 연산 타이밍을 제어하기 위한 홀드 게이트들(522(0), 524(0))이 또한 제공된다.

[0252] 도 40을 계속 참조하면, 캐리-절약 누산기 블록(472[0])의 누산된 벡터 출력 합(512(0)) 및 시프트된 누산된 벡터 출력 캐리(517(1)), 및 캐리-절약 누산기 블록(472[1])의 누산된 벡터 출력 합(512(1)) 및 시프트된 누산된 벡터 출력 캐리(517(0))는 제어 게이트들(534(0), 536(0) 및 534(1), 536(1))에 의해 각각 게이팅된다. 제어 게이트들(534(0), 536(0) 및 534(1), 536(1))은, 압축기들(508(0), 508(1))로 반환되는 누산된 벡터 출력 합(512(0)) 및 시프트된 누산된 벡터 출력 캐리(517(0)), 및 누산된 벡터 출력 합(512(1)) 및 시프트된 누산된 벡터 출력 캐리(517(1))를 각각 제어한다.

[0253] 요약하면, 도 40의 누산기 블록(472)의 누산기 블록들(472[0], 472[1])의 프로그램가능 입력 데이터 경로들(478[0], 478[1]) 및 프로그램가능 내부 데이터 경로들(480[0], 480[1])에 대해, 누산기 블록(472)은 상이한 모드들로 구성될 수 있다. 누산기 블록(472)은, 도 40에 예시된 공통 누산기 회로에 따른 특정한 벡터 프로세싱 명령에 따라 상이한 누산 연산들을 제공하도록 구성될 수 있다.

[0254] 본원에 논의된 개념들 및 실시예들에 따른 VPE들은, 임의의 프로세서-기반 디바이스에 제공되거나 그에 통합될 수 있다. 제한이 아닌 예들은, 셋 톱 박스, 엔터테인먼트 유닛, 내비게이션 디바이스, 통신 디바이스, 고정 위치 데이터 유닛, 모바일 위치 데이터 유닛, 모바일 폰, 셀룰러 폰, 컴퓨터, 휴대용 컴퓨터, 데스크톱 컴퓨터, 개인 휴대 정보 단말(PDA), 모니터, 컴퓨터 모니터, 텔레비전, 튜너, 라디오, 위성 라디오, 뮤직 플레이어, 디지털 뮤직 플레이어, 휴대용 뮤직 플레이어, 디지털 비디오 플레이어, 비디오 플레이어, 디지털 비디오 디스크(DVD) 플레이어, 및 휴대용 디지털 비디오 플레이어를 포함한다.

[0255] 이와 관련하여, 도 41은 프로세서-기반 시스템(550)의 예를 예시한다. 이러한 예에서, 프로세서-기반 시스템(550)은 각각이 하나 또는 그 초과의 프로세서들 또는 코어들(554)을 포함하는 하나 또는 그 초과의 프로세싱 유닛(PU)들(552)을 포함한다. PU(들)(552)는 비제한적인 예로서 도 2의 기저대역 프로세서(20)일 수 있다. 프로세서(554)는 비제한적인 예로서 도 2에 제공된 기저대역 프로세서(20)와 같은 벡터 프로세서일 수 있다. 이와 관련하여, 프로세서(554)는 또한, 도 2의 실행 유닛(84)을 포함하지만 이에 제한되지 않는 VPE(556)를 포함할 수 있다. PU(들)(552)는 일시적으로 저장된 데이터에 대한 빠른 액세스를 위해 프로세서(들)(554)에 커플링되는 캐시 메모리(558)를 가질 수 있다. PU(들)(552)는, 시스템 버스(560)에 커플링되고, 프로세서-기반 시스템(550)에 포함되는 마스터 및 슬레이브 디바이스들을 상호커플링시킬 수 있다. 잘 알려진 바와 같이, PU

(들)(552)는 시스템 버스(560)를 통해 어드레스, 제어, 및 데이터 정보를 교환함으로써 이러한 다른 디바이스들과 통신한다. 예컨대, PU(들)(552)는 슬레이브 디바이스의 예로서 메모리 제어기(562)에 버스 트랜잭션 요청들을 통신할 수 있다. 도 41에 예시되진 않았지만, 각각의 시스템 버스(560)가 상이한 구조를 구성하는 다수의 시스템 버스들(560)이 제공될 수 있다.

[0256] 다른 마스터 및 슬레이브 디바이스들은 시스템 버스(560)에 연결될 수 있다. 도 41에 예시된 바와 같이, 이러한 디바이스들은 예들로서, 메모리 시스템(564), 하나 또는 그 초과의 입력 디바이스들(566), 하나 또는 그 초과의 출력 디바이스들(568), 하나 또는 그 초과의 네트워크 인터페이스 디바이스들(570), 및 하나 또는 그 초과의 디스플레이 제어기들(572)을 포함할 수 있다. 메모리 시스템(564)은 메모리 제어기(562)에 의해 액세스가능한 메모리(565)를 포함할 수 있다. 입력 디바이스(들)(566)는, 입력 키들, 스위치들, 보이스 프로세서들 등을 포함하지만 이에 제한되지 않는 임의의 타입의 입력 디바이스를 포함할 수 있다. 출력 디바이스(들)(568)는, 오디오, 비디오, 다른 시각적 표시기(visual indicator)들 등을 포함하지만 이에 제한되지 않는 임의의 타입의 출력 디바이스를 포함할 수 있다. 네트워크 인터페이스 디바이스(들)(570)는 네트워크(574)로의 그리고 그로부터의 데이터의 교환이 허용되도록 구성되는 임의의 디바이스들일 수 있다. 네트워크(574)는 유선 또는 무선 네트워크, 개인 또는 공용 네트워크, 로컬 영역 네트워크(LAN), WLAN(wide local area network), 및 인터넷을 포함하지만 이에 제한되지 않는 임의의 타입의 네트워크일 수 있다. 네트워크 인터페이스 디바이스(들)(570)는 요구되는 임의의 타입의 통신 프로토콜을 지원하도록 구성될 수 있다.

[0257] PU들(552)은 또한, 하나 또는 그 초과의 디스플레이들(578)에 전송되는 정보를 제어하기 위해, 시스템 버스(560)를 통해 디스플레이 제어기(들)(572)에 액세스하도록 구성될 수 있다. 디스플레이 제어기(들)(572)는 하나 또는 그 초과의 비디오 프로세서들(580)을 통해 디스플레이될 정보를 디스플레이(들)(578)에 전송하며, 그 비디오 프로세서들(580)은, 디스플레이(들)(578)에 적절한 포맷으로 디스플레이되도록 정보를 프로세싱한다. 디스플레이(들)(578)는 CRT(cathode ray tube), LCD(liquid crystal display), 플라즈마 디스플레이 등을 포함하지만 이에 제한되지 않는 임의의 타입의 디스플레이를 포함할 수 있다.

[0258] 당업자들은, 본원에 개시된 듀얼 전압 도메인 메모리 버퍼들의 실시예들과 관련하여 설명된 다양한 예시적인 로직 블록들, 모듈들, 회로들, 및 알고리즘들이 전자 하드웨어, 메모리 또는 다른 컴퓨터-판독가능 매체에 저장되고 프로세서 또는 다른 프로세싱 디바이스에 의해 실행되는 명령들, 또는 이 둘의 결합들로서 구현될 수 있음을 추가적으로 인식할 것이다. 본원에 설명된 아비터(arbiter)들, 마스터 디바이스들, 및 슬레이브 디바이스들은, 예들로서, 임의의 회로, 하드웨어 컴포넌트, 집적 회로(IC), 또는 IC 칩에서 이용될 수 있다. 본원에 개시된 메모리는 임의의 타입 및 크기의 메모리일 수 있으며, 임의의 타입의 원하는 정보를 저장하도록 구성될 수 있다. 이러한 상호교환가능성을 명확히 예시하기 위해, 다양한 예시적인 컴포넌트들, 블록들, 모듈들, 회로들, 및 단계들은 그들의 기능의 관점들에서 일반적으로 상술되었다. 그러한 기능이 어떻게 구현되는지는 특정 애플리케이션, 설계 선택들, 및/또는 전체 시스템에 부과된 설계 제약들에 의존한다. 당업자들은 설명된 기능을 각각의 특정한 애플리케이션에 대해 다양한 방식들로 구현할 수 있지만, 그러한 구현 결정들이 본 개시내용의 범위를 벗어나게 하는 것으로서 해석되지는 않아야 한다.

[0259] 본원에 개시된 실시예들과 관련하여 설명된 다양한 예시적인 로직 블록들, 모듈들, 및 회로들은 프로세서, DSP, 주문형 집적회로(ASIC), FPGA 또는 다른 프로그래밍가능 로직 디바이스, 이산 게이트 또는 트랜지스터 로직, 이산 하드웨어 컴포넌트들, 또는 본원에 설명된 기능들을 수행하도록 설계된 이들의 임의의 결합으로 구현되거나 수행될 수 있다. 프로세서는 마이크로프로세서일 수 있지만, 대안적으로, 프로세서는 임의의 종래의 프로세서, 제어기, 마이크로제어기, 또는 상태 머신일 수 있다. 또한, 프로세서는 컴퓨팅 디바이스들의 결합, 예컨대 DSP와 마이크로프로세서의 결합, 복수의 마이크로프로세서들, DSP 코어와 결합된 하나 또는 그 초과의 마이크로프로세서들, 또는 임의의 다른 그러한 구성으로서 구현될 수 있다.

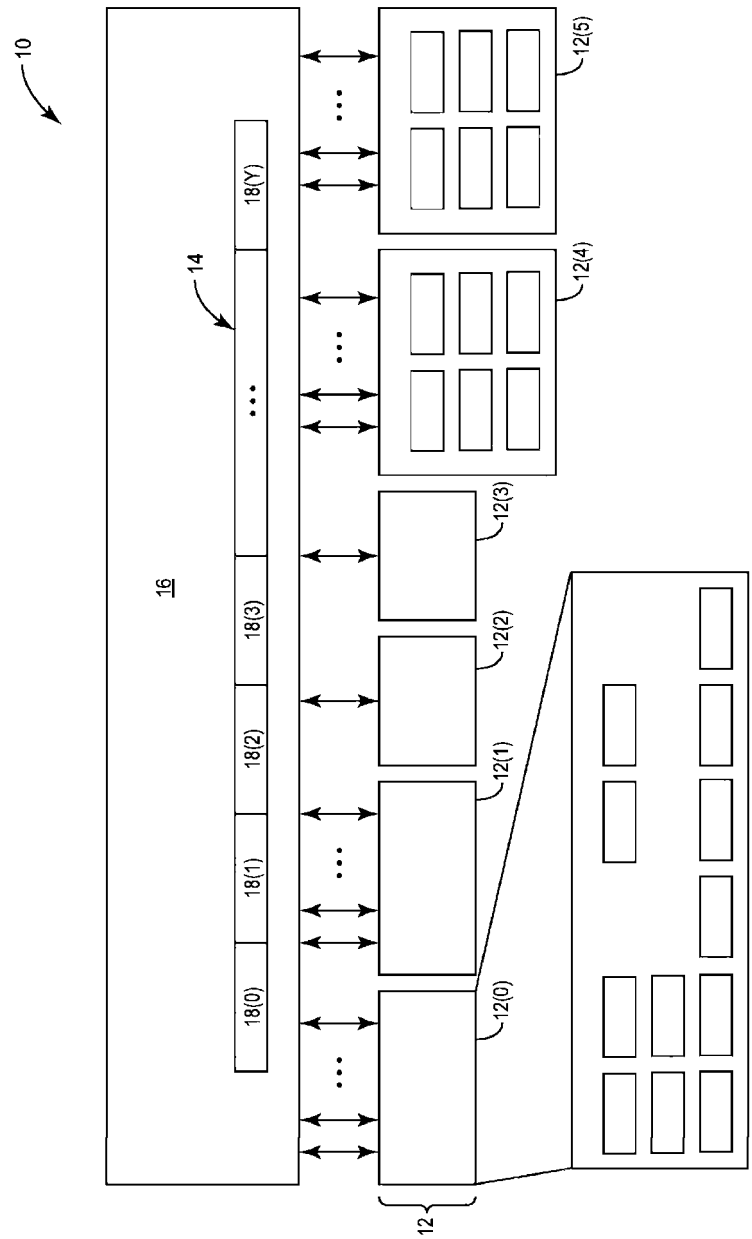
[0260] 본원에 개시된 실시예들은 하드웨어, 및 하드웨어에 저장된 명령들로 구현될 수 있으며, 예컨대, 랜덤 액세스 메모리(RAM), 플래시 메모리, 판독 전용 메모리(ROM), 전기적으로 프로그래밍가능 ROM(EPROM), 전기적으로 소거가능한 프로그래밍가능 ROM(EEPROM), 레지스터들, 하드 디스크, 착탈형 디스크, CD-ROM, 또는 당업계에 알려진 임의의 다른 형태의 컴퓨터 판독가능 매체에 상주할 수 있다. 예시적인 저장 매체는, 프로세서가 저장 매체로부터 정보를 판독하고, 저장 매체에 정보를 기입할 수 있도록 프로세서에 커플링된다. 대안적으로, 저장 매체는 프로세서에 통합될 수 있다. 프로세서 및 저장 매체는 ASIC에 상주할 수 있다. ASIC은 원격 스테이션에 상주할 수 있다. 대안적으로, 프로세서 및 저장 매체는 원격 스테이션, 기지국, 또는 서버 내의 이산 컴포넌트들로서 상주할 수 있다.

[0261] 본원의 예시적인 실시예들 중 임의의 실시예에서 설명된 동작 단계들은 예들 및 설명을 제공하기 위해 설명됨을 또한 유의한다. 설명된 동작들은 예시된 시퀀스들 이외에 다수의 상이한 시퀀스들로 수행될 수 있다. 또한, 단일 동작 단계로 설명된 동작들은 실제로, 다수의 상이한 단계들로 수행될 수 있다. 부가적으로, 예시적인 실시예들에서 논의된 하나 또는 그 초과의 동작 단계들은 결합될 수 있다. 흐름도 도면들에서 예시된 동작 단계들이, 당업자에게 용이하게 명백할 바와 같이 다수의 상이한 변형들을 겪을 수 있음이 이해될 것이다. 당업자들은, 정보 및 신호들이 다양한 상이한 기술들 및 기법들 중 임의의 기술 및 기법을 사용하여 표현될 수 있음을 또한 이해할 것이다. 예컨대, 상기 설명 전반에 걸쳐 참조될 수 있는 데이터, 명령들, 커맨드들, 정보, 신호들, 비트들, 심볼들, 및 칩들은 전압들, 전류들, 전자기파들, 자기장들 또는 자기 입자들, 광학 필드들 또는 광학 입자들, 또는 이들의 임의의 결합에 의해 표현될 수 있다.

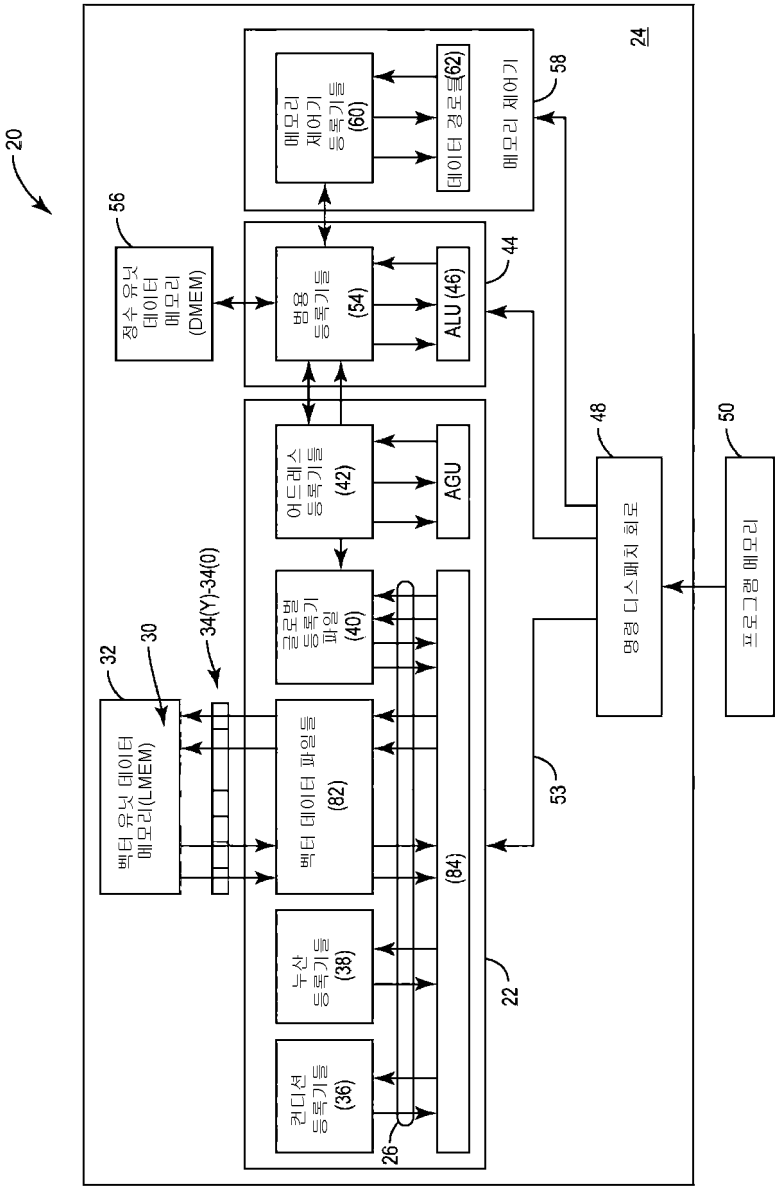
[0262] 본 개시내용의 이전 설명은 임의의 당업자가 본 개시내용을 사용 또는 실시할 수 있도록 제공된다. 본 개시내용에 대한 다양한 변형들은 당업자들에게 용이하게 명백할 것이며, 본원에 정의된 일반적인 원리들은 본 개시내용의 사상 또는 범위를 벗어나지 않으면서 다른 변경들에 적용될 수 있다. 따라서, 본 개시내용은 본원에 설명된 예들 및 설계들로 제한되도록 의도되는 것이 아니라, 본원에 개시된 원리들 및 신규한 특성들과 일치하는 가장 넓은 범위에 부합할 것이다.

도면

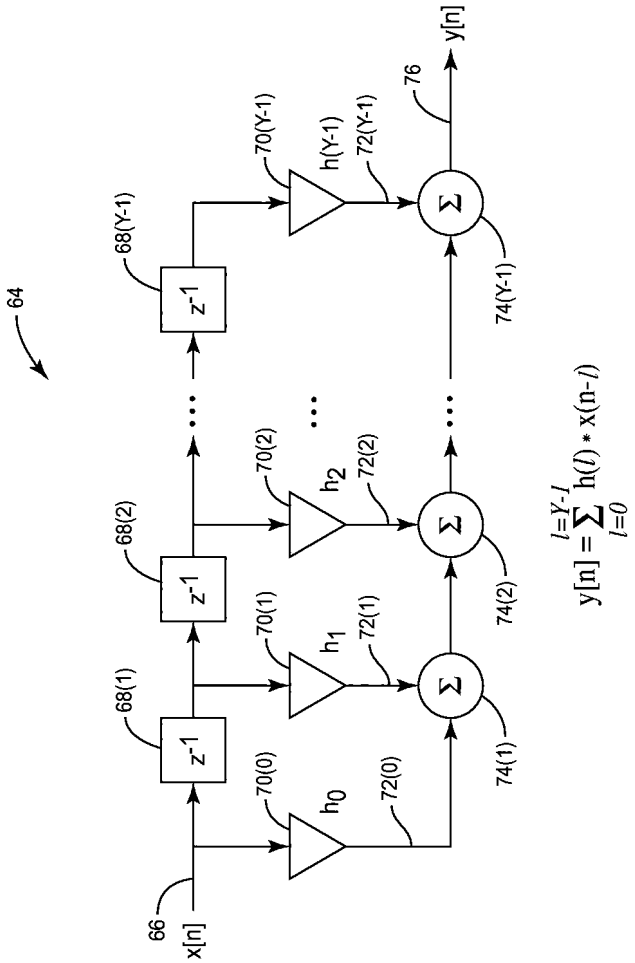
도면1



도면2

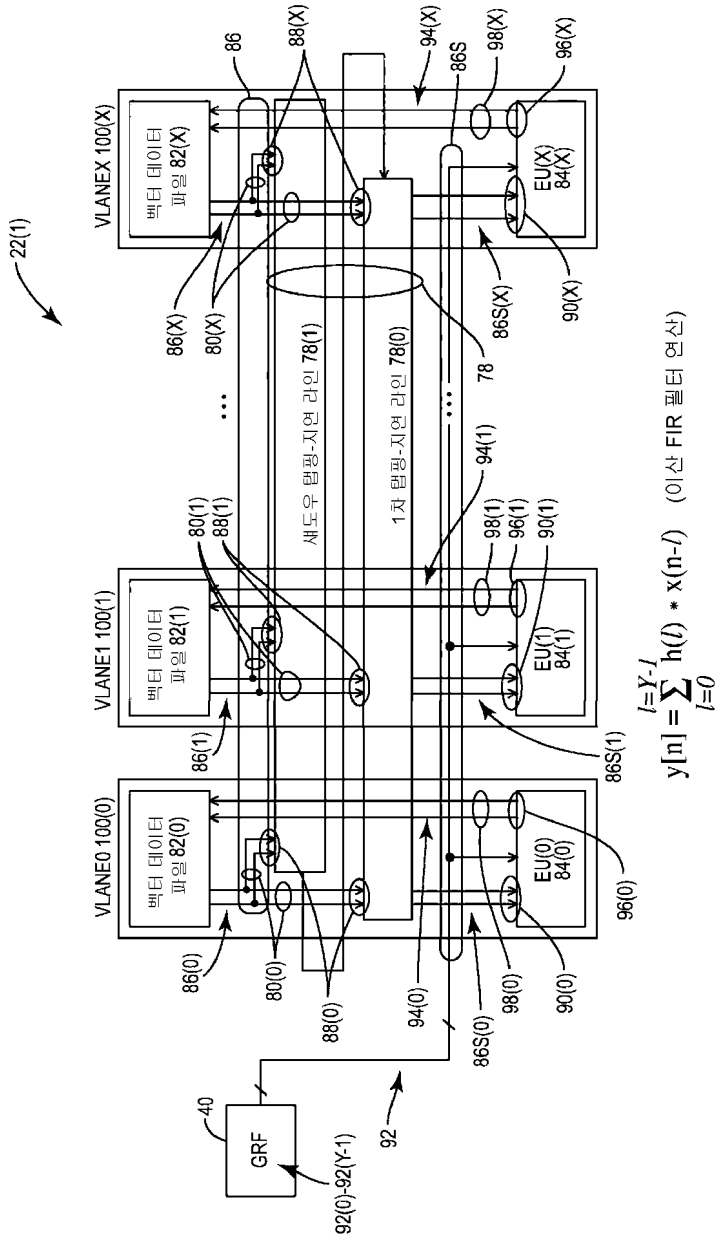

20
24
메모리 제어기
58
메모리 제어기 등록기들 (60)
데이터 경로들(62)
56
정수 유닛 데이터 메모리 (DMEM)
범용 등록기들 (54)
ALU (46)
44
어드레스 등록기들 (42)
AGU
48
명령 디스패치 회로
50
프로그램 메모리
34(Y)-34(0)
32
30
벡터 유닛 데이터 메모리(LMEM)
글로벌 등록기 파일 (40)
벡터 데이터 파일들 (82)
(84)
53
누산 등록기들 (38)
컨디션 등록기들 (36)
22
26

도면3



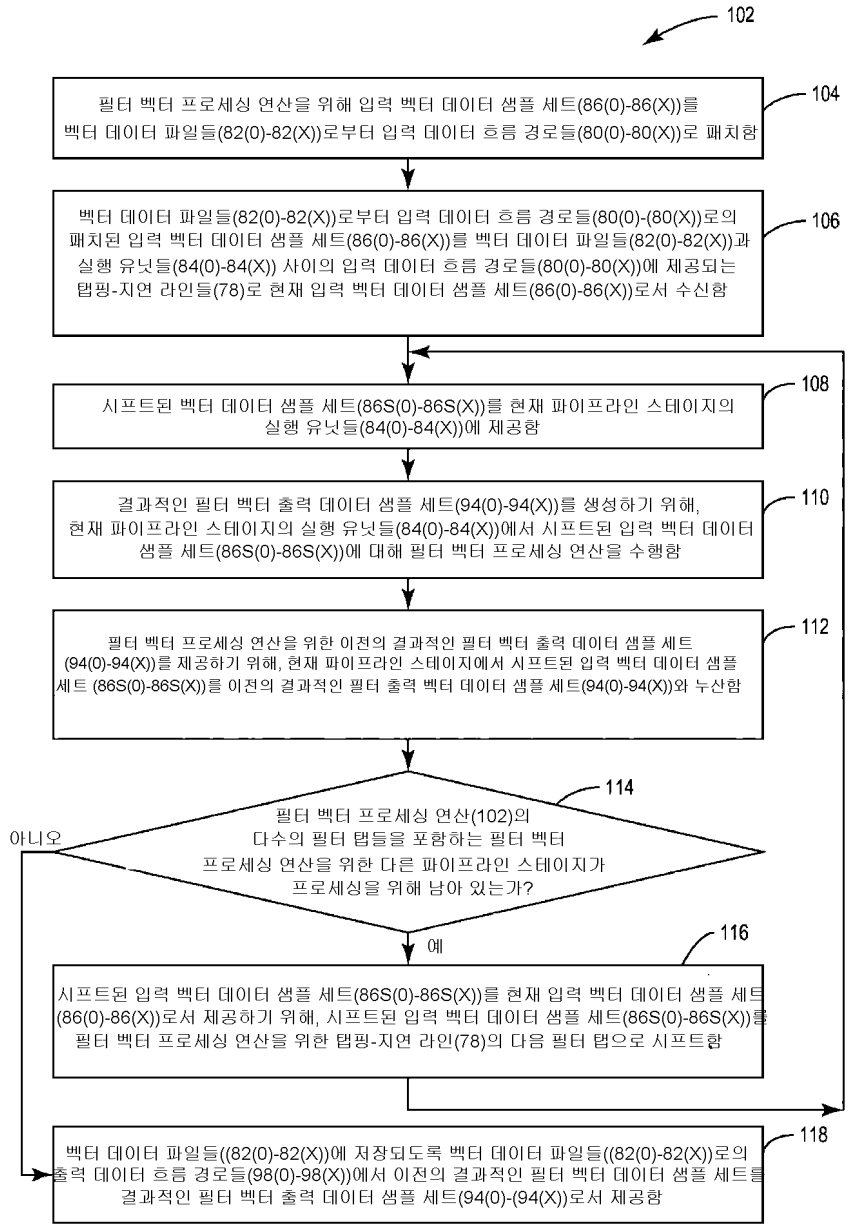
$$y[n] = \sum_{l=0}^{l=Y-1} h(l) * x(n-l)$$

도면4

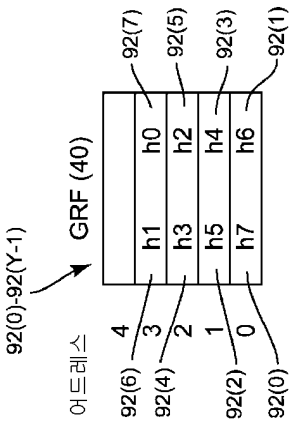


$$y[n] = \sum_{l=0}^{l=Y-1} h(l) * x(n-l) \quad \text{(이산 FIR 필터 연산)}$$

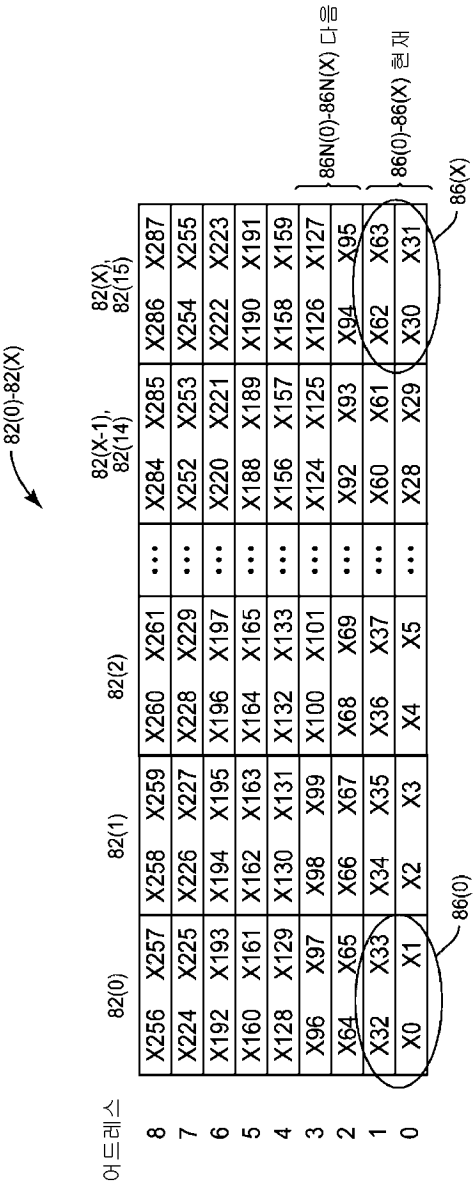
도면5


102
필터 벡터 프로세싱 연산을 위해 입력 벡터 데이터 샘플 세트(86(0)-86(X))를 벡터 데이터 파일들(82(0)-82(X))로부터 입력 데이터 흐름 경로들(80(0)-80(X))로 패치함
104
벡터 데이터 파일들(82(0)-82(X))로부터 입력 데이터 흐름 경로들(80(0)-(80(X))로의 패치된 입력 벡터 데이터 샘플 세트(86(0)-86(X))를 벡터 데이터 파일들(82(0)-82(X))과 실행 유닛들(84(0)-84(X)) 사이의 입력 데이터 흐름 경로들(80(0)-80(X))에 제공되는 탭핑-지연 라인들(78)로 현재 입력 벡터 데이터 샘플 세트(86(0)-86(X))로서 수신함
106
시프트된 벡터 데이터 샘플 세트(86S(0)-86S(X))를 현재 파이프라인 스테이지의 실행 유닛들(84(0)-84(X))에 제공함
108
결과적인 필터 벡터 출력 데이터 샘플 세트(94(0)-94(X))를 생성하기 위해, 현재 파이프라인 스테이지의 실행 유닛들(84(0)-84(X))에서 시프트된 입력 벡터 데이터 샘플 세트(86S(0)-86S(X))에 대해 필터 벡터 프로세싱 연산을 수행함
110
필터 벡터 프로세싱 연산을 위한 이전의 결과적인 필터 벡터 출력 데이터 샘플 세트 (94(0)-94(X))를 제공하기 위해, 현재 파이프라인 스테이지에서 시프트된 입력 벡터 데이터 샘플 세트 (86S(0)-86S(X))를 이전의 결과적인 필터 출력 벡터 데이터 샘플 세트(94(0)-94(X))와 누산함
112
필터 벡터 프로세싱 연산(102)의 다수의 필터 탭들을 포함하는 필터 벡터 프로세싱 연산을 위한 다른 파이프라인 스테이지가 프로세싱을 위해 남아 있는가?
114
아니오
예
시프트된 입력 벡터 데이터 샘플 세트(86S(0)-86S(X))를 현재 입력 벡터 데이터 샘플 세트 (86(0)-86(X))로서 제공하기 위해, 시프트된 입력 벡터 데이터 샘플 세트(86S(0)-86S(X))를 필터 벡터 프로세싱 연산을 위한 탭핑-지연 라인(78)의 다음 필터 탭으로 시프트함
116
벡터 데이터 파일들((82(0)-82(X))에 저장되도록 벡터 데이터 파일들((82(0)-82(X))로의 출력 데이터 흐름 경로들(98(0)-98(X))에서 이전의 결과적인 필터 벡터 데이터 샘플 세트를 결과적인 필터 벡터 출력 데이터 샘플 세트(94(0)-(94(X))로서 제공함
118

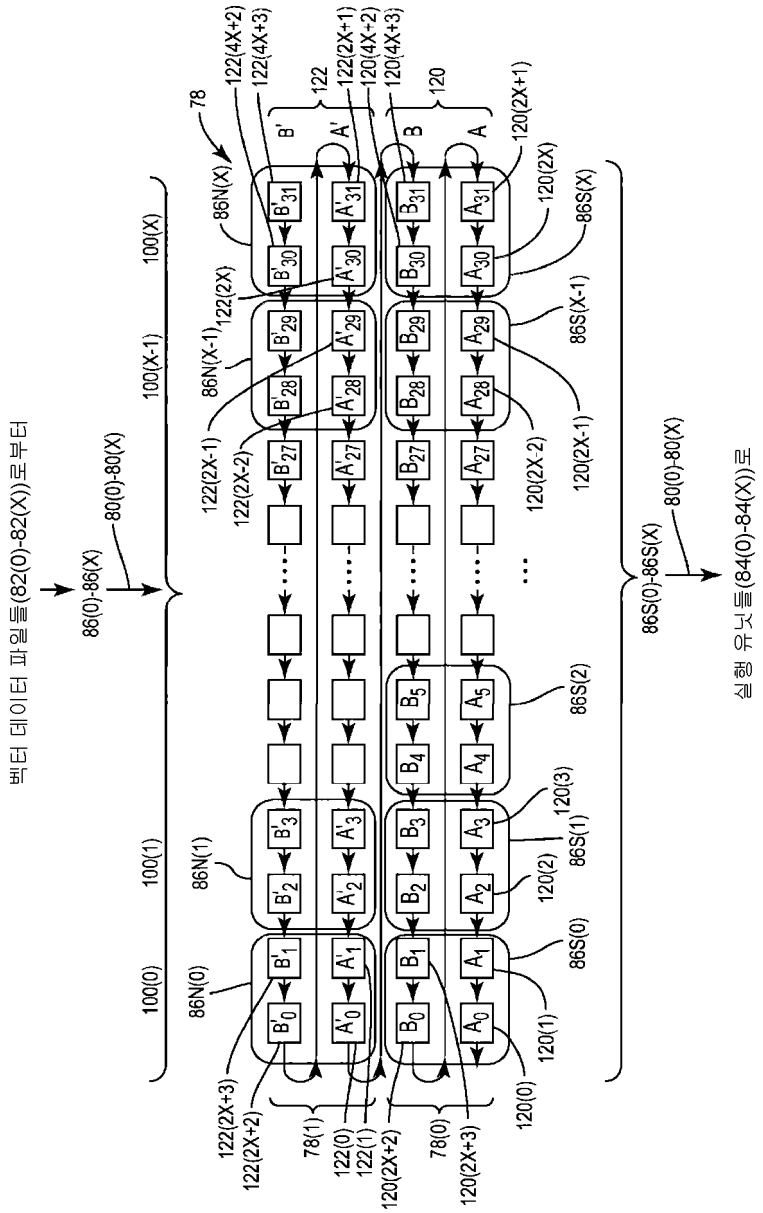
도면6a


92(7)
92(5)
92(3)
92(1)
h0
h2
h4
h6
GRF (40)
h1
h3
h5
h7
92(0)-92(Y-1)
4
3
2
1
0
어드레스
92(6)
92(4)
92(2)
92(0)

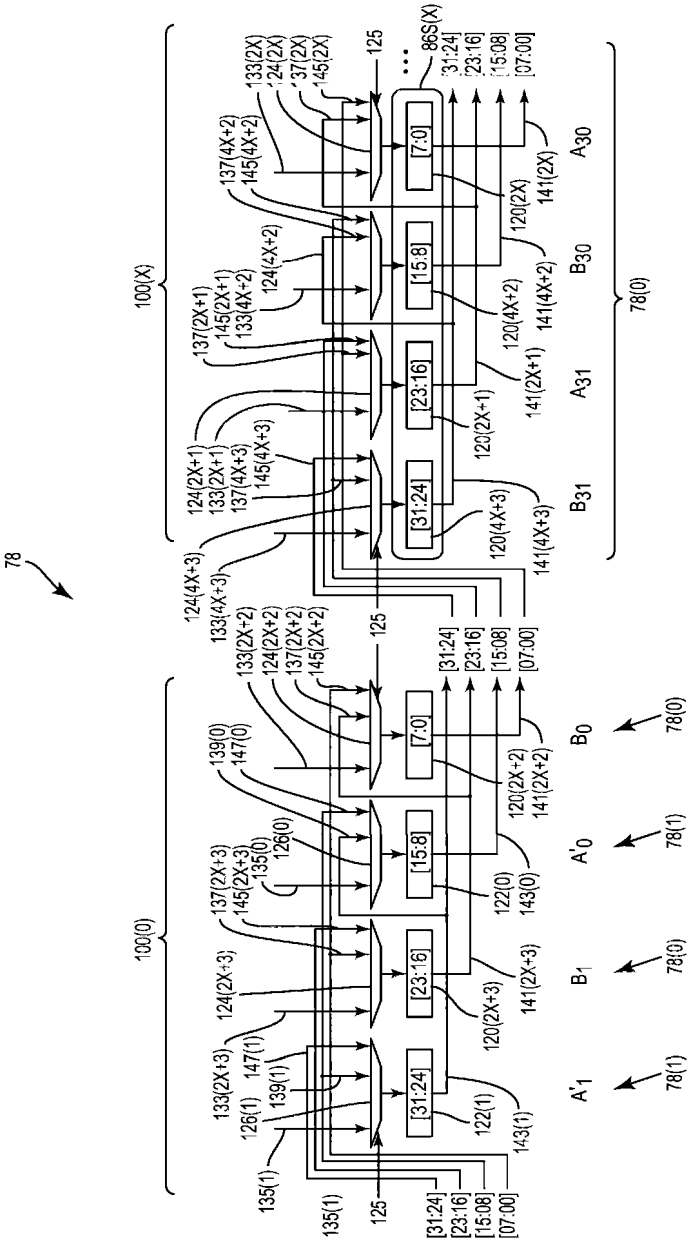
도면6b


82(0)-82(X)
어드레스
82(0)
82(1)
82(2)
82(X-1), 82(14)
82(X), 82(15)
8 X256 X257 X258 X259 X260 X261 ... X284 X285 X286 X287
7 X224 X225 X226 X227 X228 X229 ... X252 X253 X254 X255
6 X192 X193 X194 X195 X196 X197 ... X220 X221 X222 X223
5 X160 X161 X162 X163 X164 X165 ... X188 X189 X190 X191
4 X128 X129 X130 X131 X132 X133 ... X156 X157 X158 X159
3 X96 X97 X98 X99 X100 X101 ... X124 X125 X126 X127
2 X64 X65 X66 X67 X68 X69 ... X92 X93 X94 X95
1 X32 X33 X34 X35 X36 X37 ... X60 X61 X62 X63
0 X0 X1 X2 X3 X4 X5 ... X28 X29 X30 X31
86N(0)-86N(X) 다음
86(0)-86(X) 현재
86(X)
86(0)

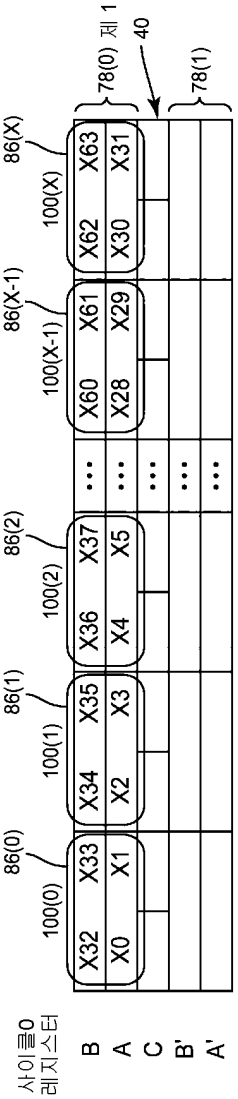
도면7


벡터 데이터 파일들(82(0)-82(X))로부터
86(0)-86(X)
80(0)-80(X)
100(0)
100(1)
100(X-1)
100(X)
86N(0)
86N(1)
86N(X-1)
86N(X)
78
122
120
B'
A'
B
A
122(4X+2)
122(4X+3)
122(2X+1)
120(4X+2)
120(4X+3)
120(2X+1)
120(2X)
86S(X)
86S(X-1)
120(2X-1)
120(2X-2)
122(2X)
122(2X-1)
122(2X-2)
86S(2)
86S(1)
86S(0)
120(3)
120(2)
120(1)
120(0)
122(2X+3)
122(2X+2)
78(1)
122(0)
122(1)
120(2X+2)
78(0)
120(2X+3)
86S(0)-86S(X)
80(0)-80(X)
실행 유닛들(84(0)-84(X))로

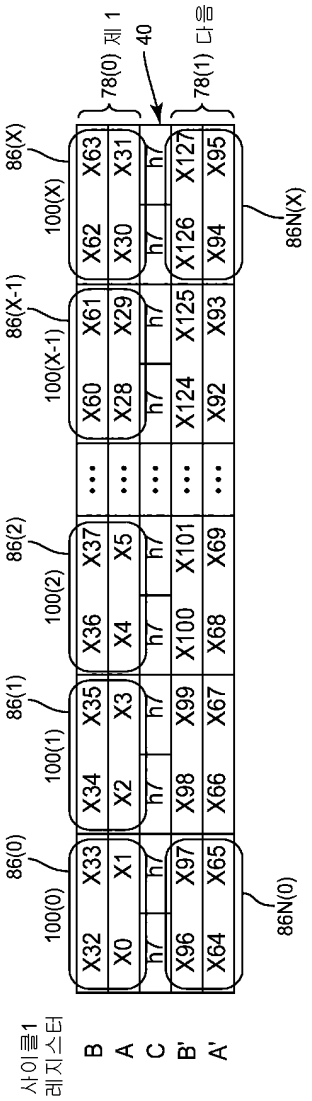
도면8


78
100(X)
100(0)
125
[31:24]
[23:16]
[15:08]
[07:00]
[7:0]
[15:8]
[23:16]
[31:24]
86S(X)
A30
B30
A31
B31
A0
B0
A1
B1
78(0)
78(1)

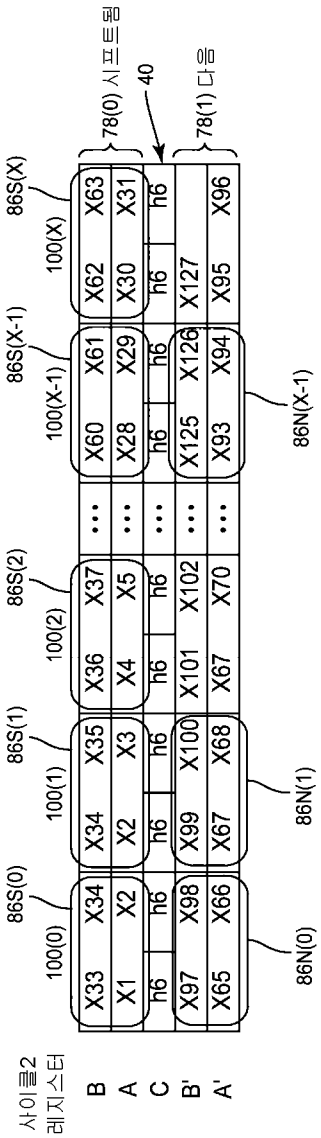
도면9a


78(0) 제 1
40
78(1)
86(X)
100(X)
86(X-1)
100(X-1)
86(2)
100(2)
86(1)
100(1)
86(0)
100(0)
X63
X31
X62
X30
X61
X29
X60
X28
X37
X5
X36
X4
X35
X3
X34
X2
X33
X1
X32
X0
B
A
C
B'
A'
사이클0
레지스터

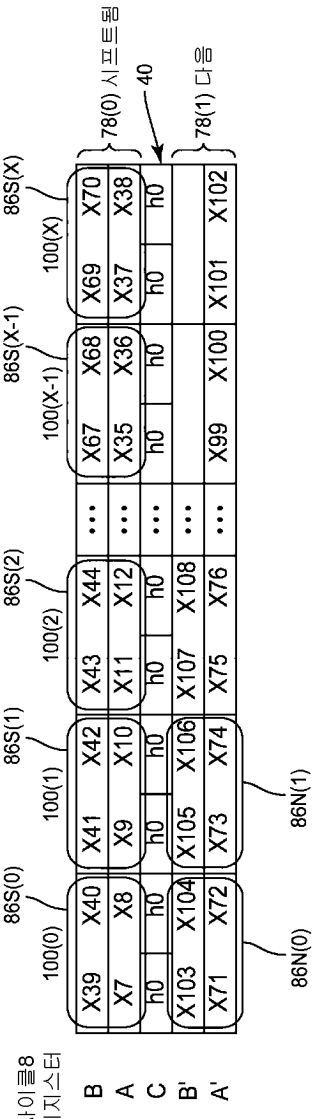
도면9b


78(0) 제 1
40
78(1) 다음
86(X)
100(X)
86(X-1)
100(X-1)
86(2)
100(2)
86(1)
100(1)
86(0)
100(0)
86N(X)
86N(0)
B
A
C
B'
A'
사이클1
레지스터
X63 X31 h7 X127 X95
X62 X30 h7 X126 X94
X61 X29 h7 X125 X93
X60 X28 h7 X124 X92
X37 X5 h7 X101 X69
X36 X4 h7 X100 X68
X35 X3 h7 X99 X67
X34 X2 h7 X98 X66
X33 X1 h7 X97 X65
X32 X0 h7 X96 X64

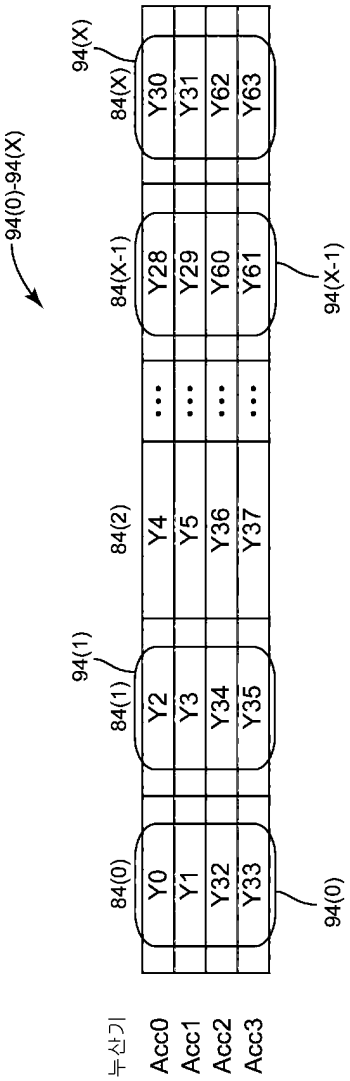
도면9c


78(0) 시프트됨
40
78(1) 다음
86S(X)
100(X)
86S(X-1)
100(X-1)
86S(2)
100(2)
86S(1)
100(1)
86S(0)
100(0)
86N(X-1)
86N(1)
86N(0)
사이클2
레지스터
B
A
C
B'
A'
X33
X34
X1
X2
X97
X98
X65
X66
X34
X35
X2
X3
X99
X100
X67
X68
X36
X37
X4
X5
X101
X102
X67
X70
X60
X61
X28
X29
X125
X126
X93
X94
X62
X63
X30
X31
X127
X95
X96
h6

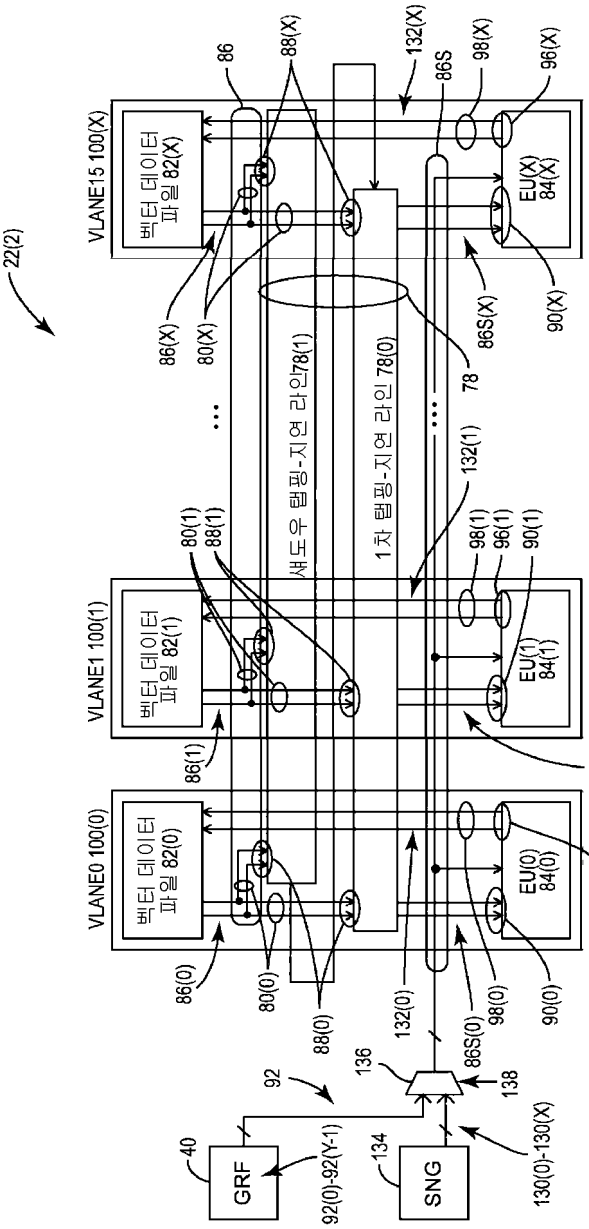
도면9d


사이클8
레지스터
B
A
C
B'
A'
86S(0)
100(0)
86S(1)
100(1)
86S(2)
100(2)
86S(X-1)
100(X-1)
86S(X)
100(X)
86N(0)
86N(1)
78(0) 시프트됨
78(1) 다음
40
X39 X40
X7 X8
h0 h0
X103 X104
X71 X72
X41 X42
X9 X10
h0 h0
X105 X106
X73 X74
X43 X44
X11 X12
h0 h0
X107 X108
X75 X76
X67 X68
X35 X36
h0 h0
X99 X100
X69 X70
X37 X38
h0 h0
X101 X102

도면10


94(0)-94(X)
94(X)
84(X)
84(X-1)
94(X-1)
84(2)
94(1)
84(1)
84(0)
94(0)
Y0
Y1
Y32
Y33
Y2
Y3
Y34
Y35
Y4
Y5
Y36
Y37
Y28
Y29
Y60
Y61
Y30
Y31
Y62
Y63
누산기
Acc0
Acc1
Acc2
Acc3

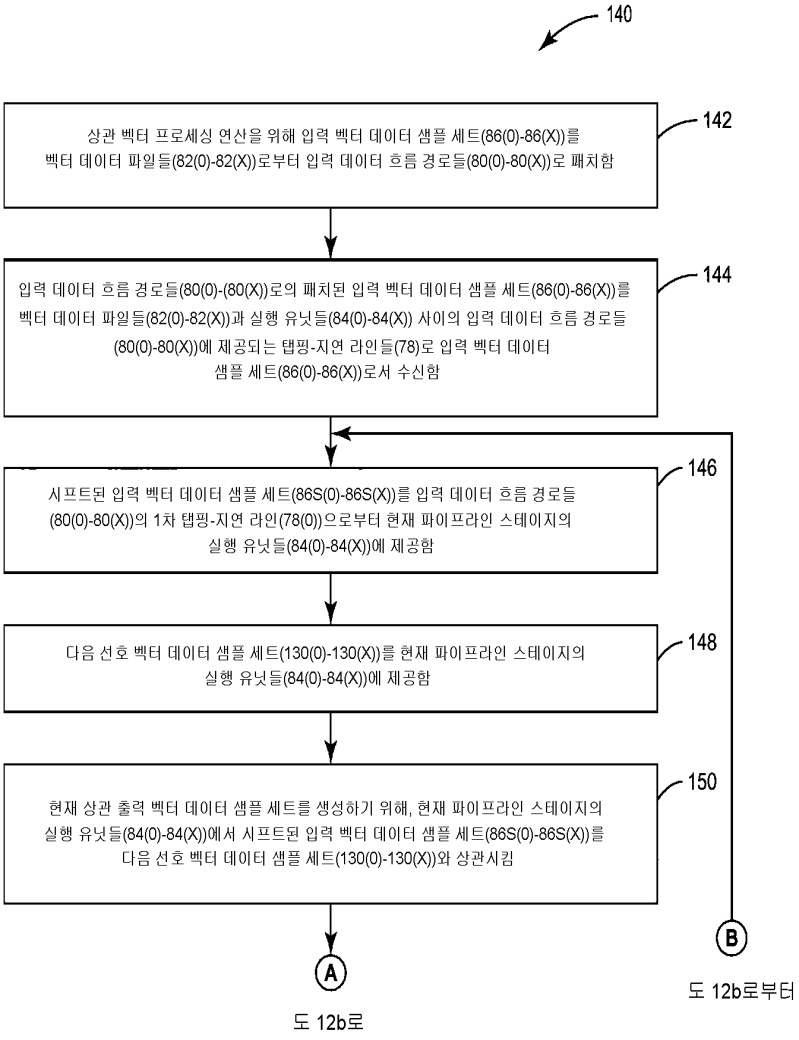
도면11



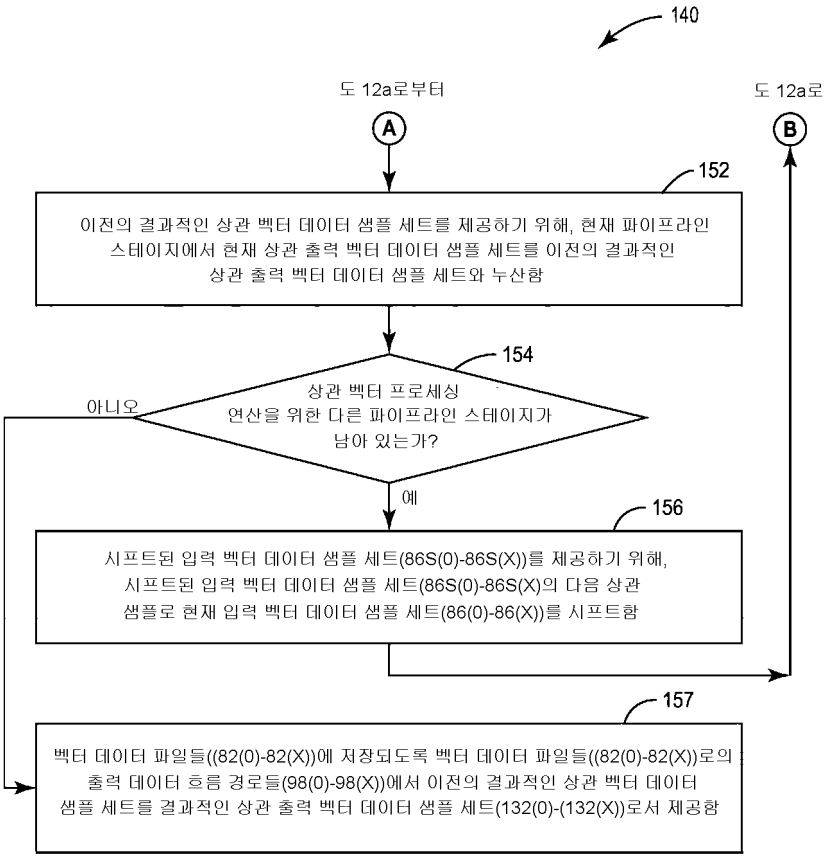
$$R_{xy}^{OT}(n) = \sum_{l=0}^{l=511} y(2l) * x(2l+n)$$ 정시

$$R_{xy}^{LT}(n) = \sum_{l=0}^{l=511} y(2l+1) * x(2l+1+n)$$ 늦음

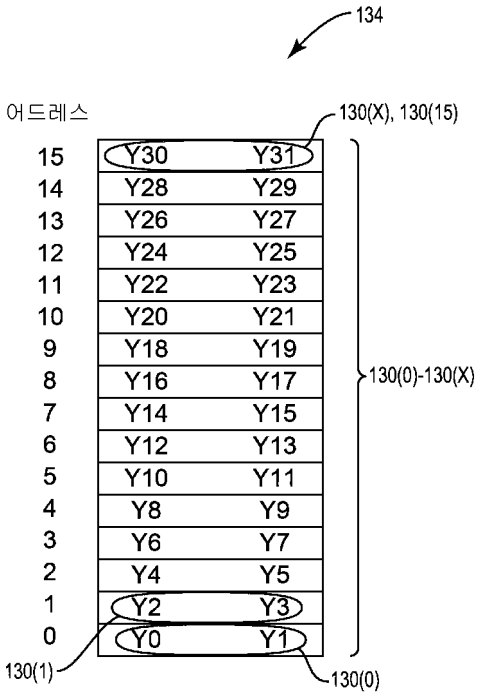
도면12a


140
상관 벡터 프로세싱 연산을 위해 입력 벡터 데이터 샘플 세트(86(0)-86(X))를 벡터 데이터 파일들(82(0)-82(X))로부터 입력 데이터 흐름 경로들(80(0)-80(X))로 패치함
142
입력 데이터 흐름 경로들(80(0)-(80(X))로의 패치된 입력 벡터 데이터 샘플 세트(86(0)-86(X))를 벡터 데이터 파일들(82(0)-82(X))과 실행 유닛들(84(0)-84(X)) 사이의 입력 데이터 흐름 경로들(80(0)-80(X))에 제공되는 탭핑-지연 라인들(78)로 입력 벡터 데이터 샘플 세트(86(0)-86(X))로서 수신함
144
시프트된 입력 벡터 데이터 샘플 세트(86S(0)-86S(X))를 입력 데이터 흐름 경로들(80(0)-80(X))의 1차 탭핑-지연 라인(78(0))으로부터 현재 파이프라인 스테이지의 실행 유닛들(84(0)-84(X))에 제공함
146
다음 선호 벡터 데이터 샘플 세트(130(0)-130(X))를 현재 파이프라인 스테이지의 실행 유닛들(84(0)-84(X))에 제공함
148
현재 상관 출력 벡터 데이터 샘플 세트를 생성하기 위해, 현재 파이프라인 스테이지의 실행 유닛들(84(0)-84(X))에서 시프트된 입력 벡터 데이터 샘플 세트(86S(0)-86S(X))를 다음 선호 벡터 데이터 샘플 세트(130(0)-130(X))와 상관시킴
150
A
도 12b로
B
도 12b로부터

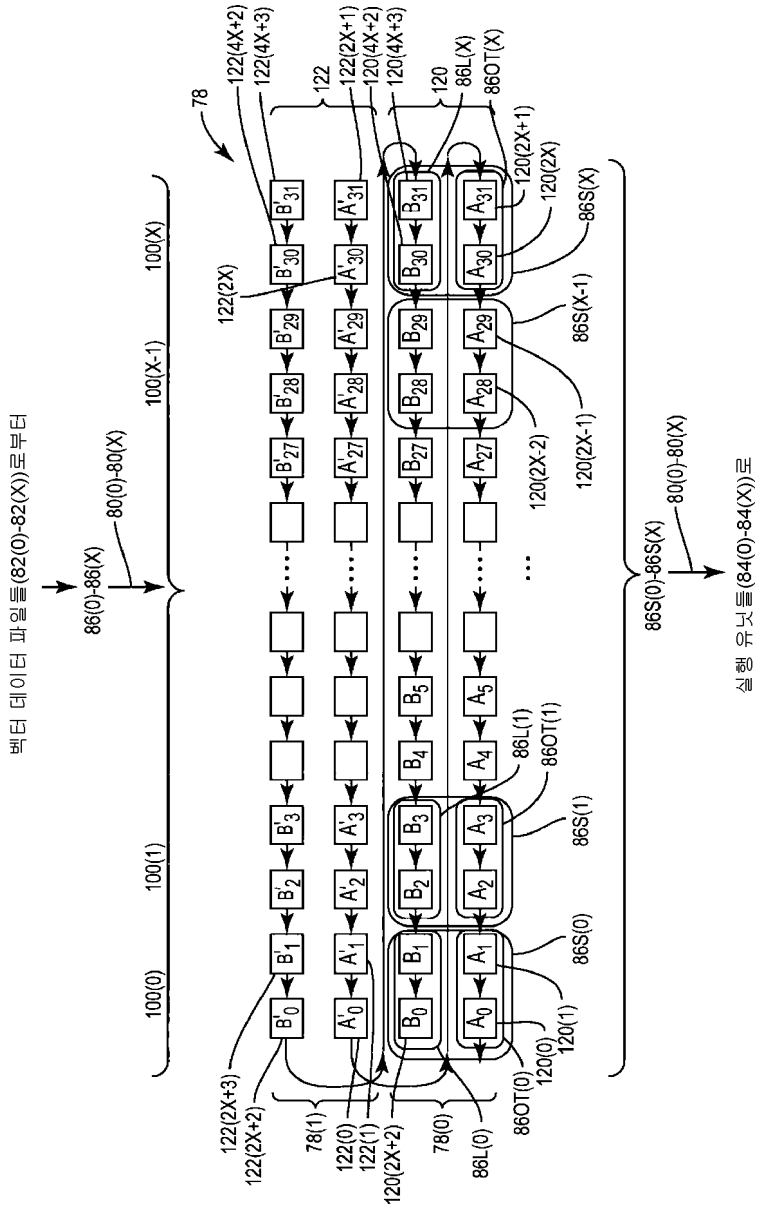
도면12b


140
도 12a로부터
A
도 12a로
B
152
이전의 결과적인 상관 벡터 데이터 샘플 세트를 제공하기 위해, 현재 파이프라인 스테이지에서 현재 상관 출력 벡터 데이터 샘플 세트를 이전의 결과적인 상관 출력 벡터 데이터 샘플 세트와 누산함
154
상관 벡터 프로세싱 연산을 위한 다른 파이프라인 스테이지가 남아 있는가?
아니오
예
156
시프트된 입력 벡터 데이터 샘플 세트(86S(0)-86S(X))를 제공하기 위해, 시프트된 입력 벡터 데이터 샘플 세트(86S(0)-86S(X)의 다음 상관 샘플로 현재 입력 벡터 데이터 샘플 세트(86(0)-86(X))를 시프트함
157
벡터 데이터 파일들((82(0)-82(X))에 저장되도록 벡터 데이터 파일들((82(0)-82(X))로의 출력 데이터 흐름 경로들(98(0)-98(X))에서 이전의 결과적인 상관 벡터 데이터 샘플 세트를 결과적인 상관 출력 벡터 데이터 샘플 세트(132(0)-(132(X))로서 제공함

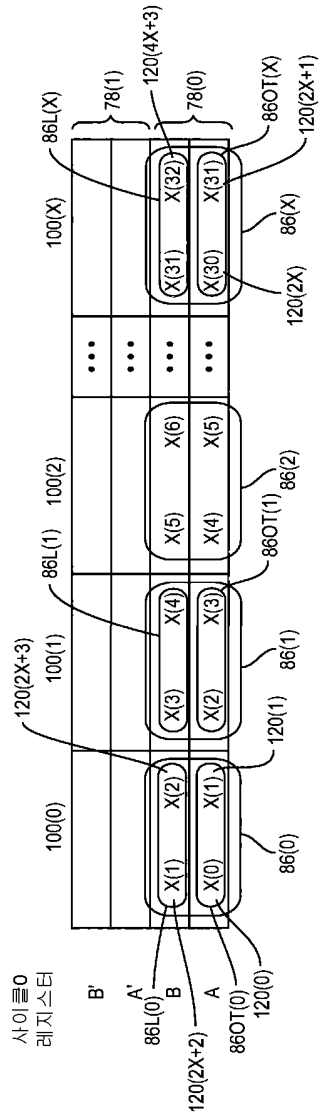
도면13


134
어드레스
130(X), 130(15)
15 Y30 Y31
14 Y28 Y29
13 Y26 Y27
12 Y24 Y25
11 Y22 Y23
10 Y20 Y21
9 Y18 Y19
8 Y16 Y17
7 Y14 Y15
6 Y12 Y13
5 Y10 Y11
4 Y8 Y9
3 Y6 Y7
2 Y4 Y5
1 Y2 Y3
0 Y0 Y1
130(0)-130(X)
130(1)
130(0)

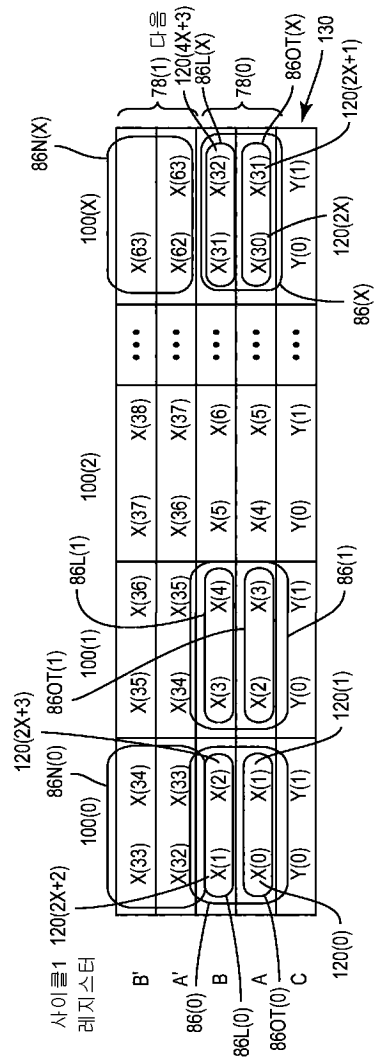
도면14


벡터 데이터 파일들(82(0)-82(X))로부터
86(0)-86(X)
80(0)-80(X)
100(0)
100(1)
100(X-1)
100(X)
78
122(4X+2)
122(4X+3)
122
122(2X+1)
120(4X+2)
120(4X+3)
120
86L(X)
86OT(X)
120(2X+1)
120(2X)
86S(X)
122(2X)
86S(X-1)
120(2X-2)
120(2X-1)
86S(0)-86S(X)
80(0)-80(X)
실행 유닛들(84(0)-84(X))로
122(2X+3)
122(2X+2)
78(1)
122(0)
122(1)
120(2X+2)
78(0)
86L(0)
86OT(0)
120(0)
120(1)
86S(0)
86S(1)
86L(1)
86OT(1)

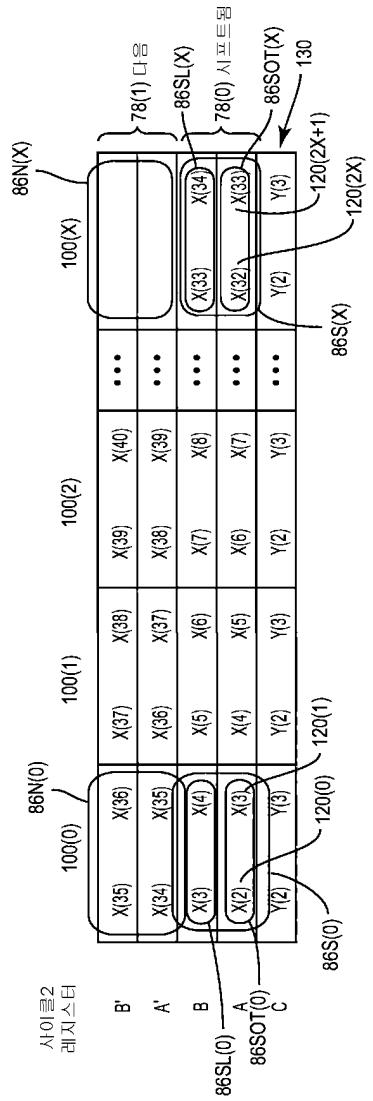
도면15a


사이클0
레지스터
B'
A'
B
A
86L(0)
120(2X+2)
86OT(0)
120(0)
120(2X+3)
100(0)
100(1)
86L(1)
100(2)
100(X)
86L(X)
78(1)
120(4X+3)
78(0)
86OT(X)
120(2X+1)
86(X)
120(2X)
86(2)
86OT(1)
86(1)
120(1)
86(0)
X(0)
X(1)
X(2)
X(3)
X(4)
X(5)
X(6)
X(30)
X(31)
X(32)

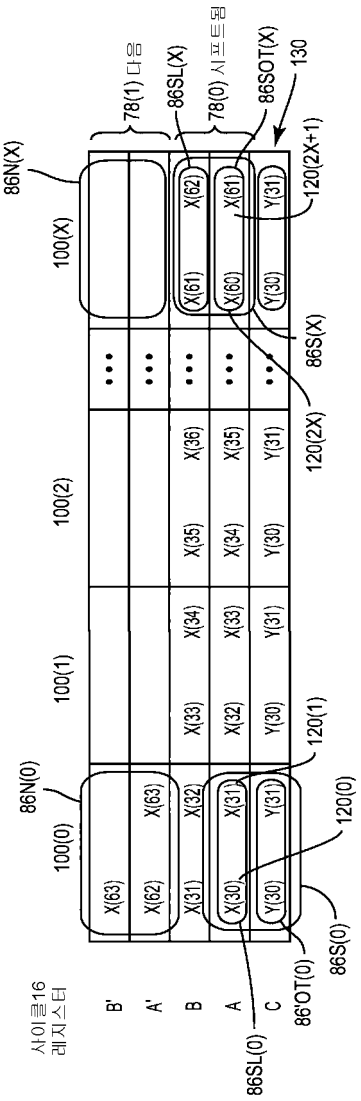
도면15b


사이클1
레지스터
B'
A'
B
A
C
86(0)
86L(0)
86OT(0)
120(0)
120(1)
120(2)
120(2X)
120(2X+1)
120(2X+2)
120(2X+3)
120(4X+3)
130
78(0)
78(1) 다음
86(X)
86(1)
86L(1)
86L(X)
86N(0)
86N(X)
86OT(1)
86OT(X)
100(0)
100(1)
100(2)
100(X)
X(63)
X(62)
X(31)
X(30)
Y(1)
Y(0)
X(32)
X(33)
X(34)
X(35)
X(36)
X(37)
X(38)
X(5)
X(4)
X(3)
X(2)
X(1)
X(0)
X(6)
X(37)
X(63)

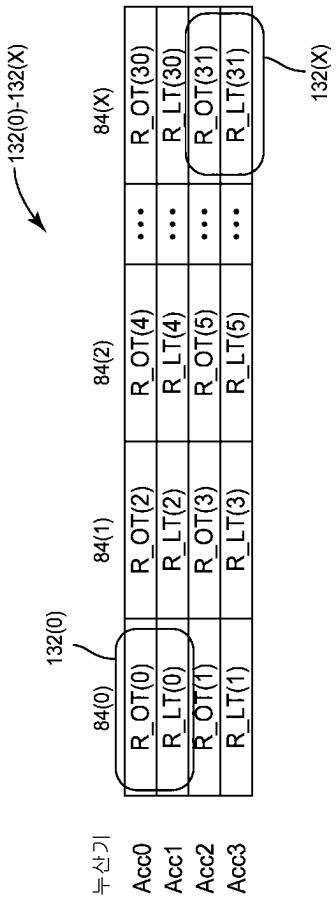
도면15c


78(1) 다음
86SL(X)
78(0) 시프트됨
86SOT(X)
130
86N(X)
100(X)
120(2X+1)
120(2X)
86S(X)
100(2)
100(1)
86N(0)
100(0)
120(1)
120(0)
86S(0)
사이클2
레지스터
B'
A'
B
A
C
86SL(0)
86SOT(0)

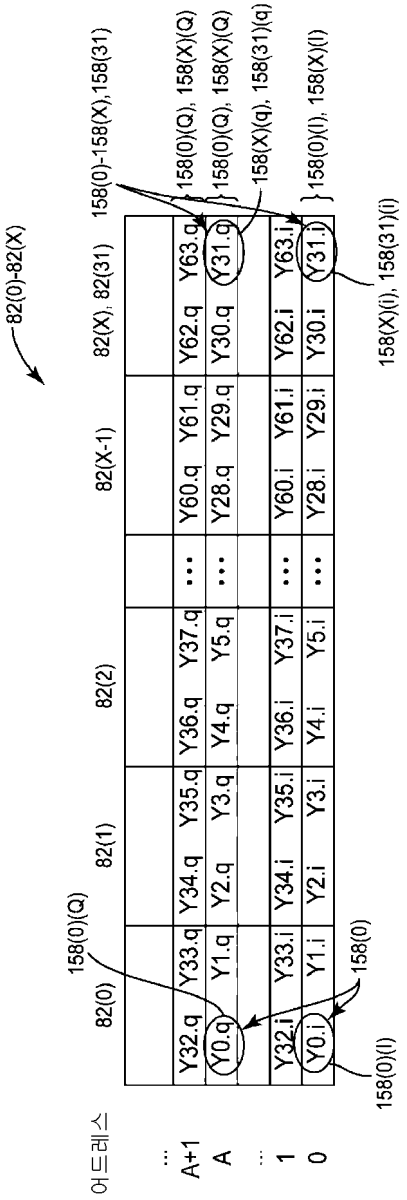
도면15d


78(1) 다음
86SL(X)
78(0) 시프트됨
86SOT(X)
130
86N(X)
100(X)
120(2X+1)
86S(X)
120(2X)
100(2)
100(1)
86N(0)
120(1)
100(0)
120(0)
86S(0)
86'OT(0)
86SL(0)
사이클16
레지스터
B'
A'
B
A
C
X(63)
X(62)
X(61)
X(60)
X(36)
X(35)
X(34)
X(33)
X(32)
X(31)
X(30)
Y(31)
Y(30)

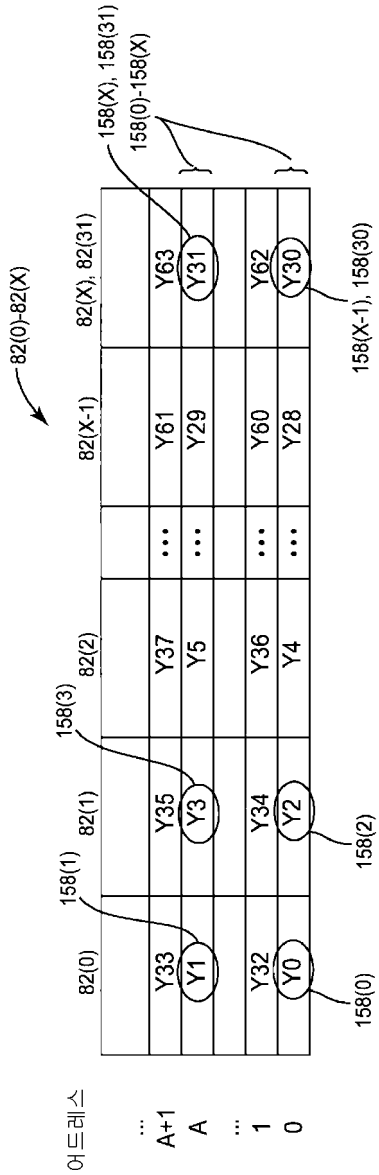
도면16


132(0)-132(X)
84(0)
84(1)
84(2)
84(X)
132(0)
132(X)
누산기
Acc0
Acc1
Acc2
Acc3
R_OT(0)
R_LT(0)
R_OT(1)
R_LT(1)
R_OT(2)
R_LT(2)
R_OT(3)
R_LT(3)
R_OT(4)
R_LT(4)
R_OT(5)
R_LT(5)
R_OT(30)
R_LT(30)
R_OT(31)
R_LT(31)

도면17a


82(0)-82(X)
158(0)-158(X),158(31)
158(0)(Q), 158(X)(Q)
158(0)(Q), 158(X)(Q)
158(X)(q), 158(31)(q)
158(0)(I), 158(X)(I)
158(X)(i), 158(31)(i)
158(0)(Q)
158(0)
158(0)(I)
어드레스
82(0)
82(1)
82(2)
82(X-1)
82(X), 82(31)
...
A+1
A
...
1
0
Y32.q Y33.q Y34.q Y35.q Y36.q Y37.q ... Y60.q Y61.q Y62.q Y63.q
Y0.q Y1.q Y2.q Y3.q Y4.q Y5.q ... Y28.q Y29.q Y30.q Y31.q
Y32.i Y33.i Y34.i Y35.i Y36.i Y37.i ... Y60.i Y61.i Y62.i Y63.i
Y0.i Y1.i Y2.i Y3.i Y4.i Y5.i ... Y28.i Y29.i Y30.i Y31.i

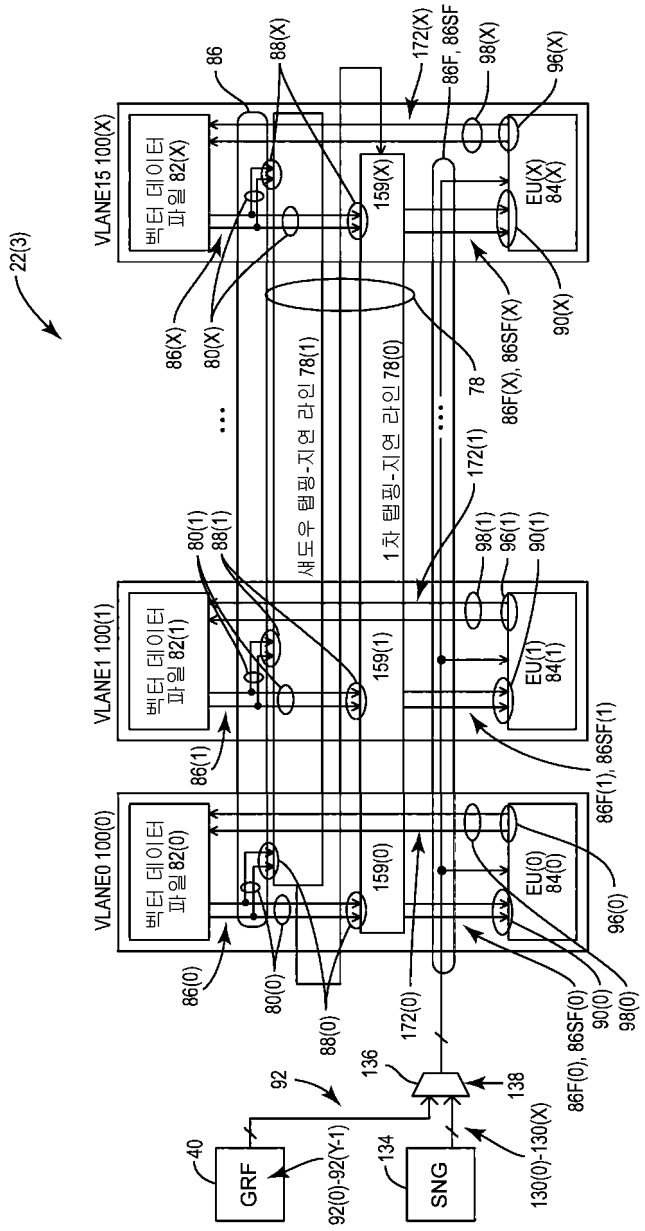
도면17b


82(0)-82(X)
158(X), 158(31)
158(0)-158(X)
82(X), 82(31)
82(X-1)
82(2)
158(3)
82(1)
158(1)
82(0)
158(2)
158(0)
158(X-1), 158(30)
어드레스
...
A+1
A
...
1
0
Y33
Y1
Y35
Y3
Y37
Y5
Y61
Y29
Y63
Y31
Y32
Y0
Y34
Y2
Y36
Y4
Y60
Y28
Y62
Y30

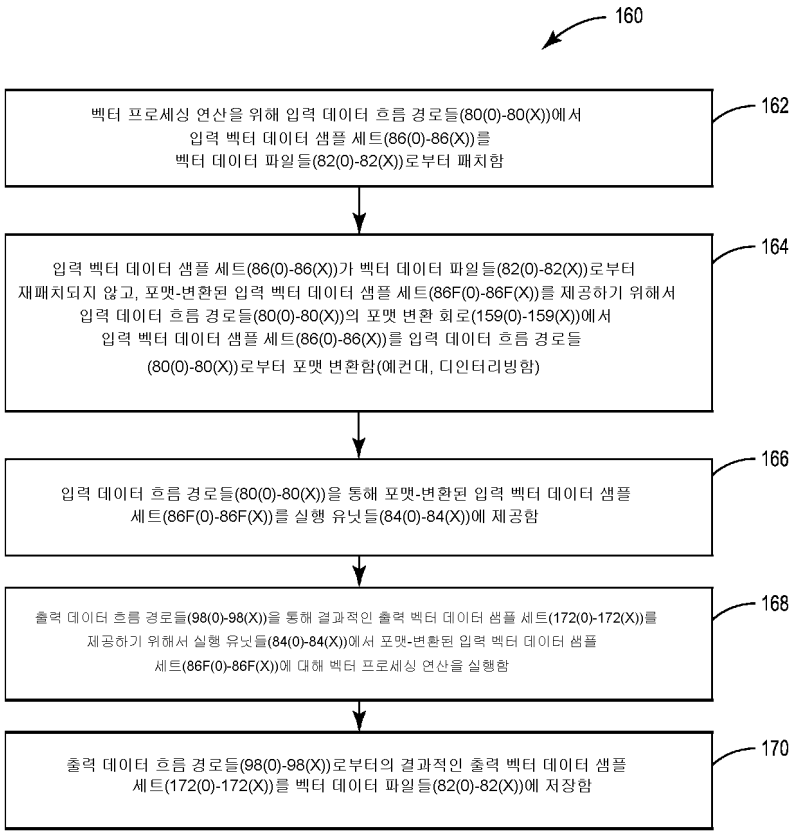
도면18a

도면18b

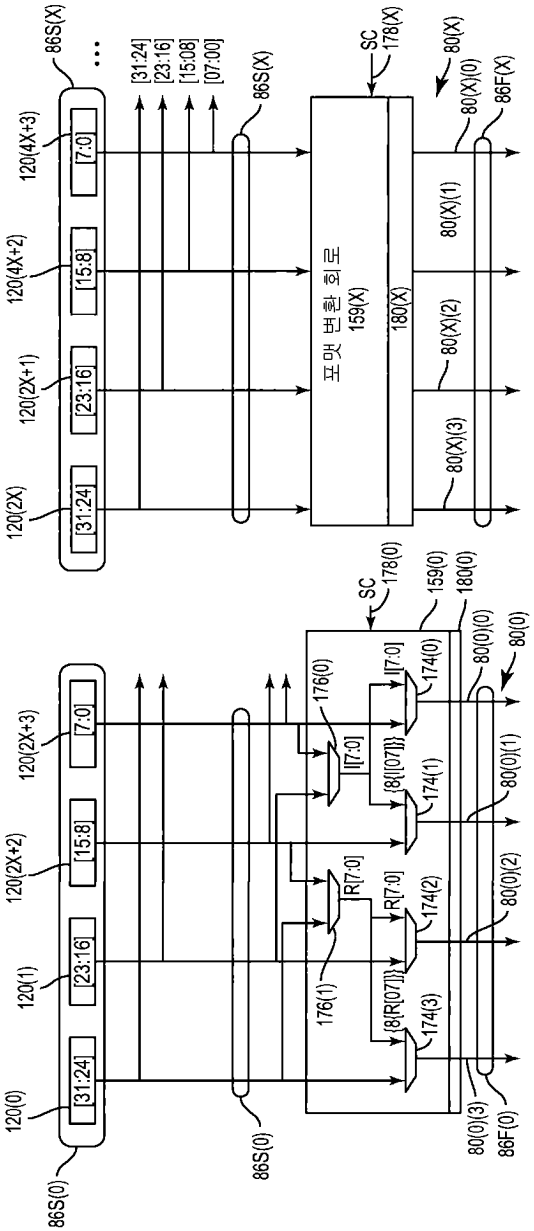
도면19


22(3)
VLANE15 100(X)
벡터 데이터 파일 82(X)
VLANE1 100(1)
벡터 데이터 파일 82(1)
VLANE0 100(0)
벡터 데이터 파일 82(0)
86
88(X)
172(X)
86F, 86SF
98(X)
96(X)
159(X)
EU(X) 84(X)
86(X)
80(X)
90(X)
86F(X), 86SF(X)
78
섀도우 탭핑-지연 라인 78(1)
1차 탭핑-지연 라인 78(0)
172(1)
80(1)
88(1)
98(1)
96(1)
90(1)
159(1)
EU(1) 84(1)
86(1)
86F(1), 86SF(1)
159(0)
EU(0) 84(0)
96(0)
86(0)
80(0)
88(0)
172(0)
86F(0), 86SF(0)
90(0)
98(0)
136
92
138
40
GRF
92(0)-92(Y-1)
134
SNG
130(0)-130(X)

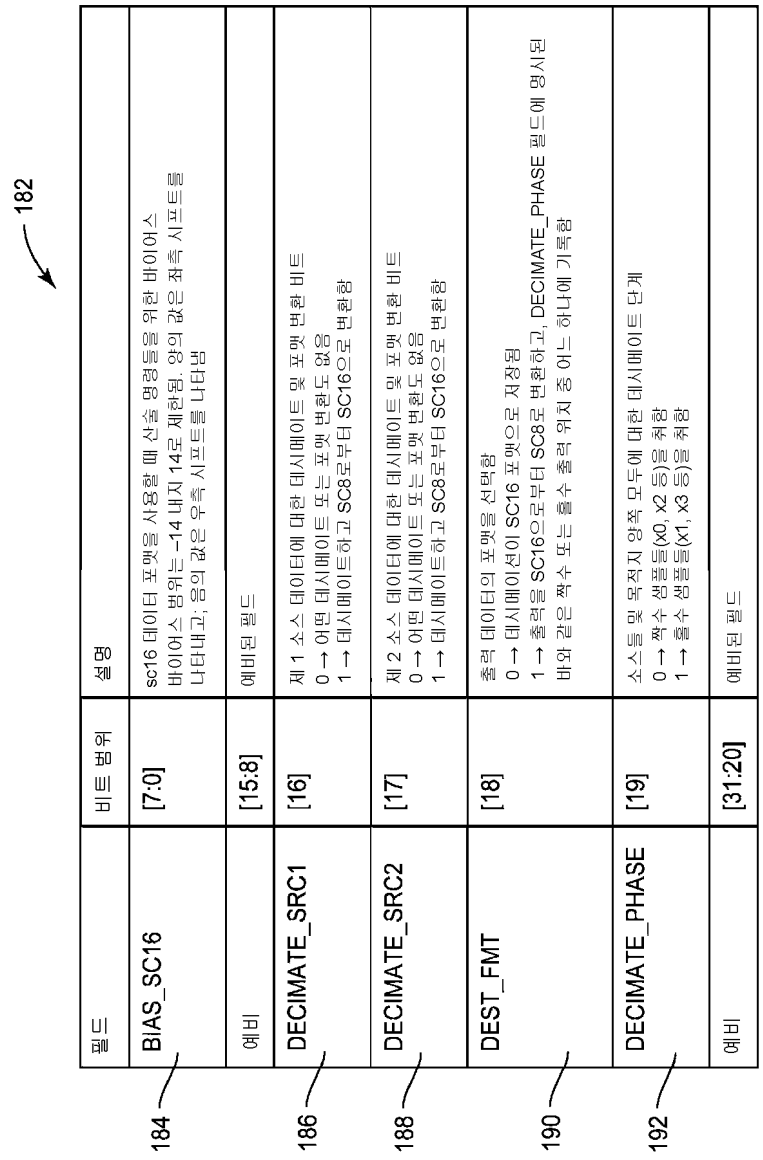
도면20


160
벡터 프로세싱 연산을 위해 입력 데이터 흐름 경로들(80(0)-80(X))에서
입력 벡터 데이터 샘플 세트(86(0)-86(X))를
벡터 데이터 파일들(82(0)-82(X))로부터 패치함
162
입력 벡터 데이터 샘플 세트(86(0)-86(X))가 벡터 데이터 파일들(82(0)-82(X))로부터
재패치되지 않고, 포맷-변환된 입력 벡터 데이터 샘플 세트(86F(0)-86F(X))를 제공하기 위해서
입력 데이터 흐름 경로들(80(0)-80(X))의 포맷 변환 회로(159(0)-159(X))에서
입력 벡터 데이터 샘플 세트(86(0)-86(X))를 입력 데이터 흐름 경로들
(80(0)-80(X))로부터 포맷 변환함(예컨대, 디인터리빙함)
164
입력 데이터 흐름 경로들(80(0)-80(X))을 통해 포맷-변환된 입력 벡터 데이터 샘플
세트(86F(0)-86F(X))를 실행 유닛들(84(0)-84(X))에 제공함
166
출력 데이터 흐름 경로들(98(0)-98(X))을 통해 결과적인 출력 벡터 데이터 샘플 세트(172(0)-172(X))를
제공하기 위해서 실행 유닛들(84(0)-84(X))에서 포맷-변환된 입력 벡터 데이터 샘플
세트(86F(0)-86F(X))에 대해 벡터 프로세싱 연산을 실행함
168
출력 데이터 흐름 경로들(98(0)-98(X))로부터의 결과적인 출력 벡터 데이터 샘플
세트(172(0)-172(X))를 벡터 데이터 파일들(82(0)-82(X))에 저장함
170

도면21

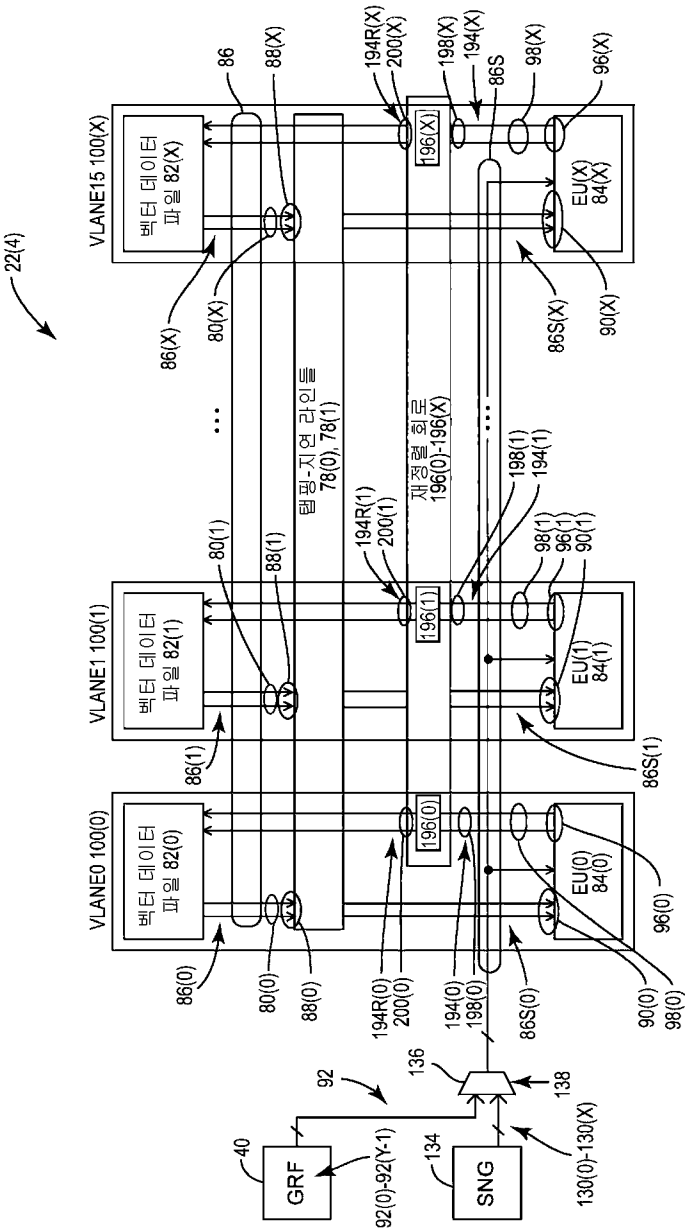

86S(X)
120(4X+3)
120(4X+2)
120(2X+1)
120(2X)
[7:0]
[15:8]
[23:16]
[31:24]
[31:24]
[23:16]
[15:08]
[07:00]
86S(X)
SC
178(X)
80(X)
80(X)(0)
86F(X)
포맷 변환 회로
159(X)
180(X)
80(X)(1)
80(X)(2)
80(X)(3)
SC
178(0)
159(0)
180(0)
80(0)(0)
80(0)
176(0)
I[7:0]
174(0)
I[7:0]
{8{I[07]}}
174(1)
80(0)(1)
R[7:0]
R[7:0]
174(2)
80(0)(2)
176(1)
{8{R[07]}}
174(3)
120(2X+3)
120(2X+2)
120(1)
120(0)
[7:0]
[15:8]
[23:16]
[31:24]
86S(0)
86S(0)
80(0)(3)
86F(0)

도면22

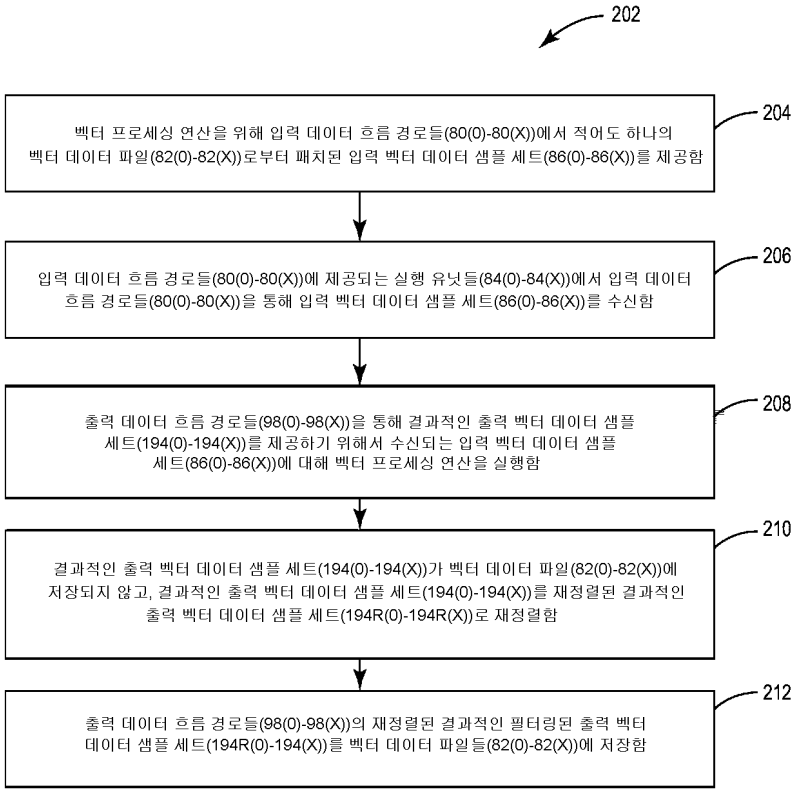


필드	비트 범위	설명
BIAS_SC16	[7:0]	sc16 데이터 포맷을 사용할 때 산출 명령들을 위한 바이어스 바이어스 범위는 -14 내지 14로 제한됨. 양의 값은 좌측 시프트를 나타내고; 음의 값은 우측 시프트를 나타냄
예비	[15:8]	예비된 필드
DECIMATE_SRC1	[16]	제 1 소스 데이터에 대한 데시메이트 및 포맷 변환 비트 0 → 어떤 데시메이트 또는 포맷 변환도 없음 1 → 데시메이트하고 SC8로부터 SC16으로 변환함
DECIMATE_SRC2	[17]	제 2 소스 데이터에 대한 데시메이트 및 포맷 변환 비트 0 → 어떤 데시메이트 또는 포맷 변환도 없음 1 → 데시메이트하고 SC8로부터 SC16으로 변환함
DEST_FMT	[18]	출력 데이터의 포맷을 선택함 0 → 데시메이션이 SC16 포맷으로 저장됨 1 → 출력을 SC16으로부터 SC8로 변환하고, DECIMATE_PHASE 필드에 명시된 바와 같은 짝수 또는 홀수 출력 위치 중 어느 하나에 기록함
DECIMATE_PHASE	[19]	소스들 및 목적지 양쪽 모두에 대한 데시메이트 단계 0 → 짝수 샘플들(x0, x2 등)을 취함 1 → 홀수 샘플들(x1, x3 등)을 취함
예비	[31:20]	예비된 필드

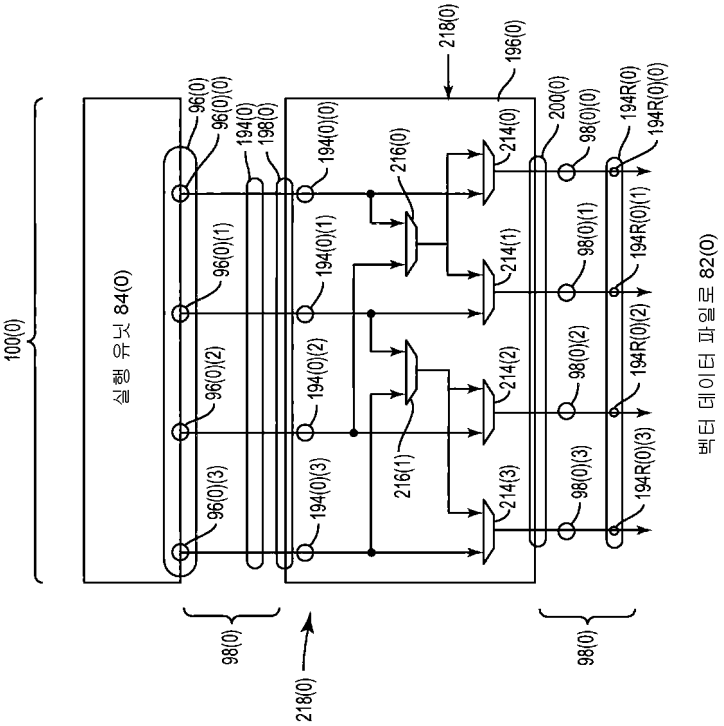
도면23


22(4)
VLANE15 100(X)
VLANE1 100(1)
VLANE0 100(0)
벡터 데이터 파일 82(X)
벡터 데이터 파일 82(1)
벡터 데이터 파일 82(0)
EU(X) 84(X)
EU(1) 84(1)
EU(0) 84(0)
86
88(X)
194R(X)
200(X)
198(X)
194(X)
86S
98(X)
96(X)
86(X)
80(X)
86S(X)
90(X)
80(1)
88(1)
194R(1)
200(1)
198(1)
194(1)
98(1)
96(1)
90(1)
86(1)
86S(1)
86(0)
80(0)
88(0)
194R(0)
200(0)
194(0)
198(0)
86S(0)
90(0)
98(0)
96(0)
탭핑-지연 라인들 78(0), 78(1)
재정렬 회로 196(0)-196(X)
196(X)
196(1)
196(0)
92
136
138
40
GRF
92(0)-92(Y-1)
134
SNG
130(0)-130(X)

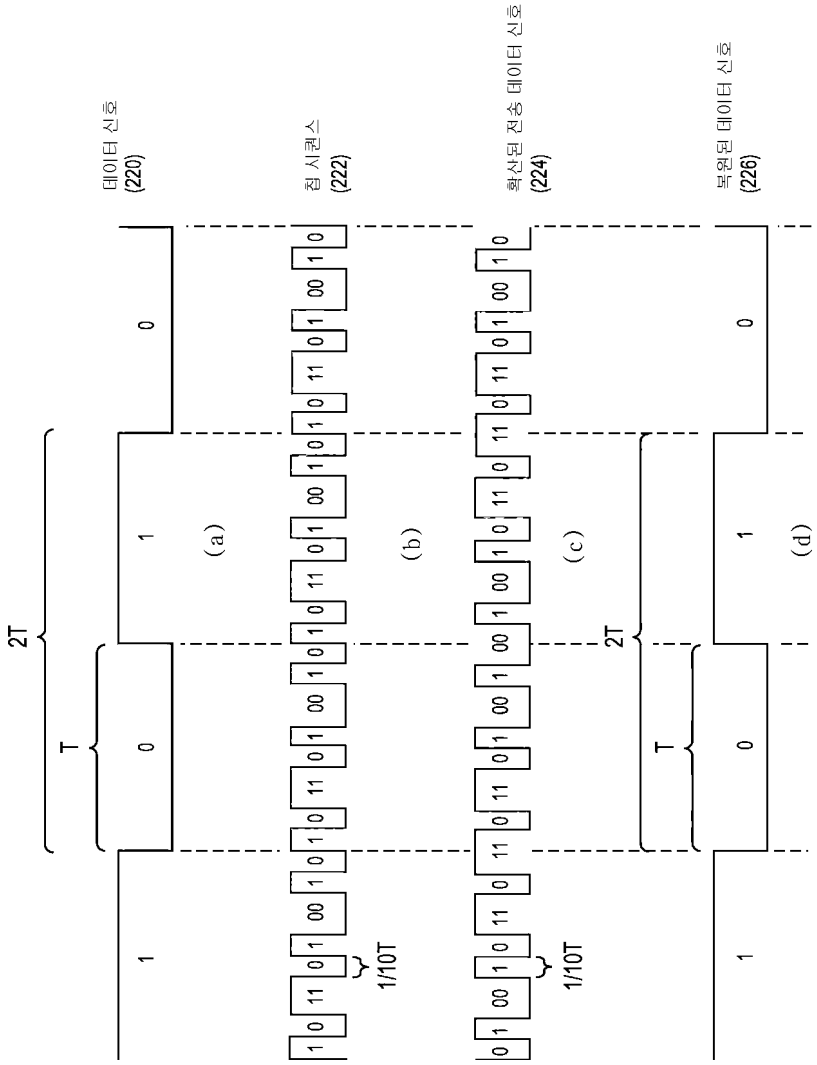
도면24


202
벡터 프로세싱 연산을 위해 입력 데이터 흐름 경로들(80(0)-80(X))에서 적어도 하나의 벡터 데이터 파일(82(0)-82(X))로부터 패치된 입력 벡터 데이터 샘플 세트(86(0)-86(X))를 제공함
204
입력 데이터 흐름 경로들(80(0)-80(X))에 제공되는 실행 유닛들(84(0)-84(X))에서 입력 데이터 흐름 경로들(80(0)-80(X))을 통해 입력 벡터 데이터 샘플 세트(86(0)-86(X))를 수신함
206
출력 데이터 흐름 경로들(98(0)-98(X))을 통해 결과적인 출력 벡터 데이터 샘플 세트(194(0)-194(X))를 제공하기 위해서 수신되는 입력 벡터 데이터 샘플 세트(86(0)-86(X))에 대해 벡터 프로세싱 연산을 실행함
208
결과적인 출력 벡터 데이터 샘플 세트(194(0)-194(X))가 벡터 데이터 파일(82(0)-82(X))에 저장되지 않고, 결과적인 출력 벡터 데이터 샘플 세트(194(0)-194(X))를 재정렬된 결과적인 출력 벡터 데이터 샘플 세트(194R(0)-194R(X))로 재정렬함
210
출력 데이터 흐름 경로들(98(0)-98(X))의 재정렬된 결과적인 필터링된 출력 벡터 데이터 샘플 세트(194R(0)-194(X))를 벡터 데이터 파일들(82(0)-82(X))에 저장함
212

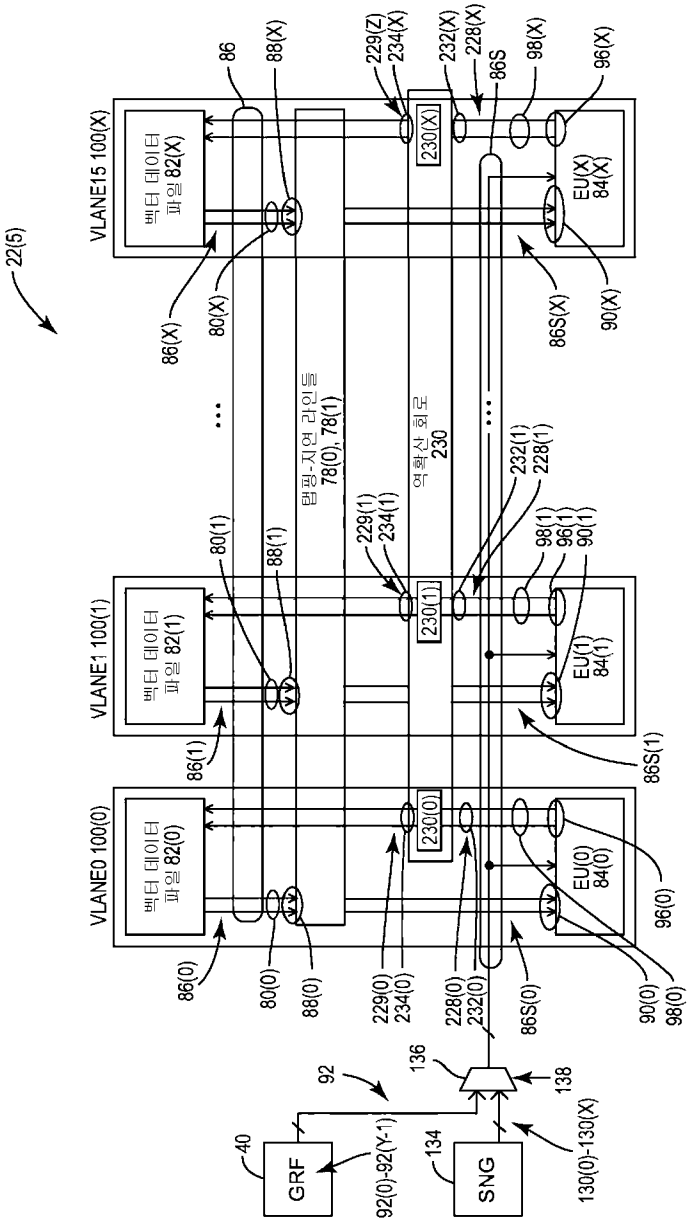
도면25


100(0)
실행 유닛 84(0)
96(0)
96(0)(0)
96(0)(1)
96(0)(2)
96(0)(3)
194(0)
198(0)
194(0)(0)
194(0)(1)
194(0)(2)
194(0)(3)
216(0)
216(1)
218(0)
196(0)
214(0)
214(1)
214(2)
214(3)
200(0)
98(0)
98(0)(0)
98(0)(1)
98(0)(2)
98(0)(3)
194R(0)
194R(0)(0)
194R(0)(1)
194R(0)(2)
194R(0)(3)
벡터 데이터 파일로 82(0)

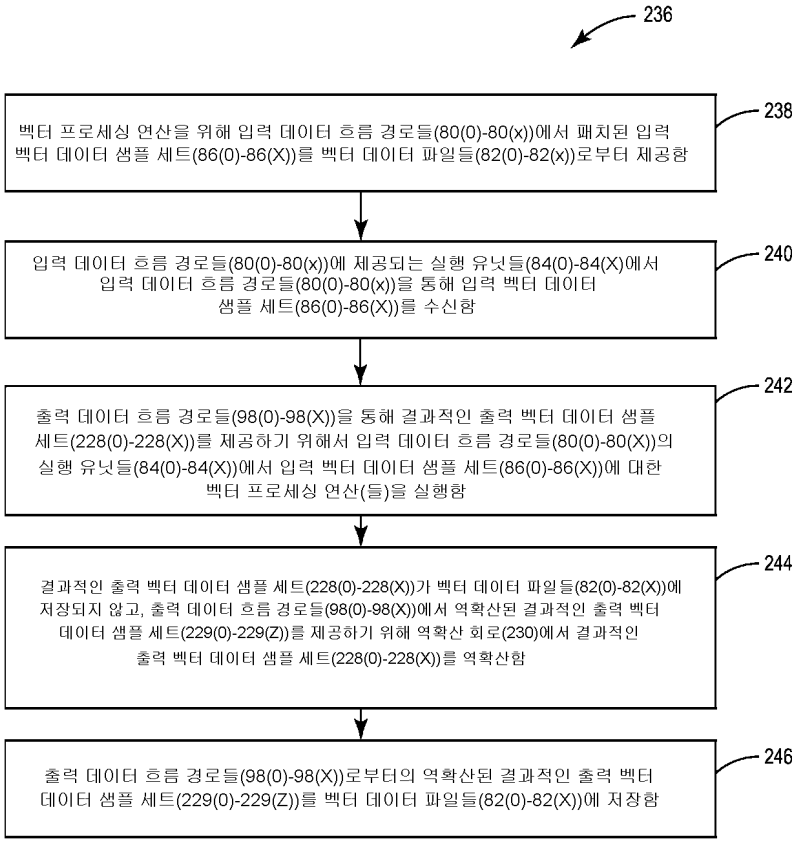
도면26


데이터 신호
(220)
칩 시퀀스
(222)
확산된 전송 데이터 신호
(224)
복원된 데이터 신호
(226)
2T
T
1/10T
(a)
(b)
(c)
(d)

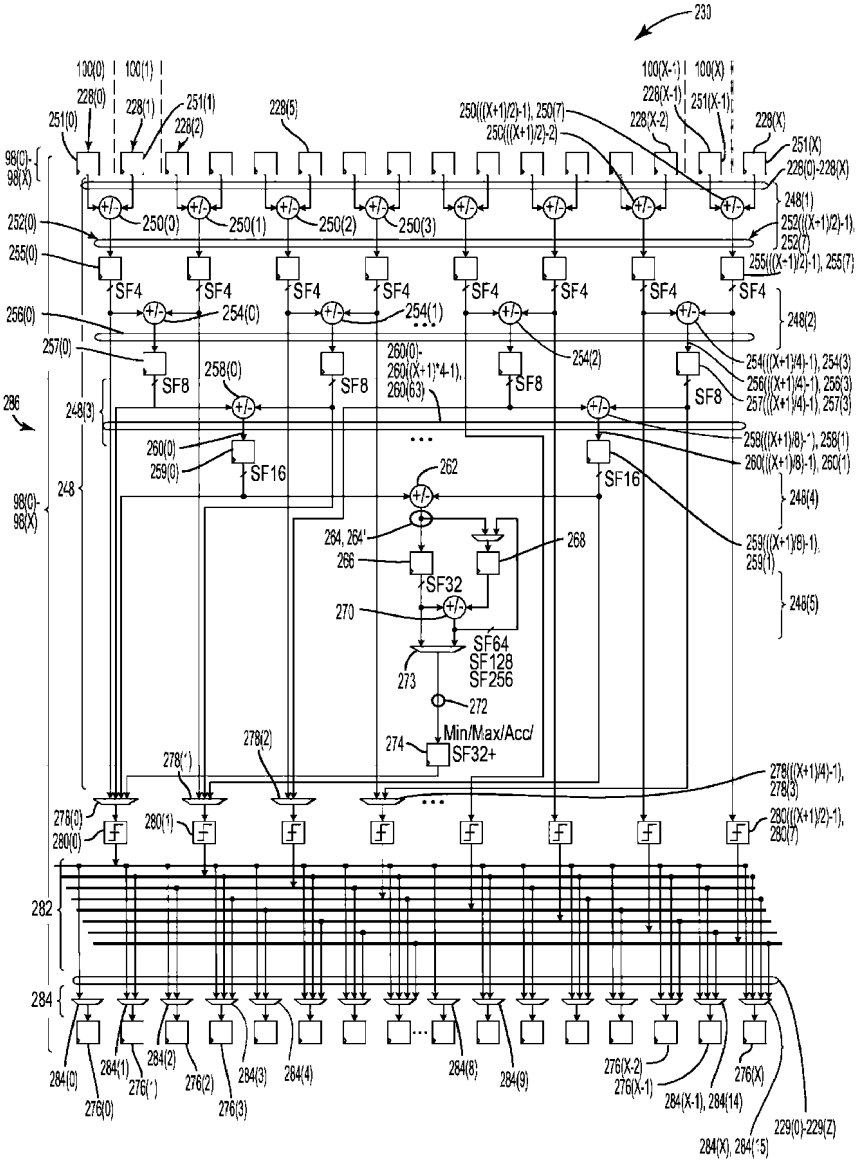
도면27


22(5)
VLANE15 100(X)
VLANE1 100(1)
VLANE0 100(0)
벡터 데이터 파일 82(X)
벡터 데이터 파일 82(1)
벡터 데이터 파일 82(0)
EU(X) 84(X)
EU(1) 84(1)
EU(0) 84(0)
230(X)
230(1)
230(0)
역확산 회로 230
탭핑-지연 라인들 78(0), 78(1)
86
88(X)
229(Z)
234(X)
232(X)
228(X)
86S
98(X)
96(X)
86(X)
80(X)
86S(X)
90(X)
80(1)
88(1)
229(1)
234(1)
232(1)
228(1)
98(1)
96(1)
90(1)
86(1)
86S(1)
96(0)
86(0)
80(0)
88(0)
229(0)
234(0)
228(0)
232(0)
86S(0)
90(0)
98(0)
92
136
138
40
GRF
92(0)-92(Y-1)
134
SNG
130(0)-130(X)

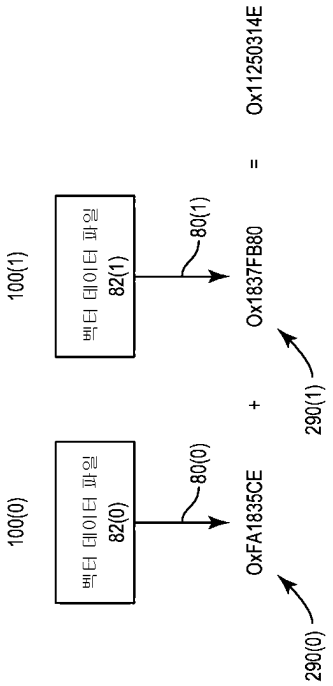
도면28


236
벡터 프로세싱 연산을 위해 입력 데이터 흐름 경로들(80(0)-80(x))에서 패치된 입력 벡터 데이터 샘플 세트(86(0)-86(X))를 벡터 데이터 파일들(82(0)-82(x))로부터 제공함
238
입력 데이터 흐름 경로들(80(0)-80(x))에 제공되는 실행 유닛들(84(0)-84(X)에서 입력 데이터 흐름 경로들(80(0)-80(x))을 통해 입력 벡터 데이터 샘플 세트(86(0)-86(X))를 수신함
240
출력 데이터 흐름 경로들(98(0)-98(X))을 통해 결과적인 출력 벡터 데이터 샘플 세트(228(0)-228(X))를 제공하기 위해서 입력 데이터 흐름 경로들(80(0)-80(X))의 실행 유닛들(84(0)-84(X))에서 입력 벡터 데이터 샘플 세트(86(0)-86(X))에 대한 벡터 프로세싱 연산(들)을 실행함
242
결과적인 출력 벡터 데이터 샘플 세트(228(0)-228(X))가 벡터 데이터 파일들(82(0)-82(X))에 저장되지 않고, 출력 데이터 흐름 경로들(98(0)-98(X))에서 역확산된 결과적인 출력 벡터 데이터 샘플 세트(229(0)-229(Z))를 제공하기 위해 역확산 회로(230)에서 결과적인 출력 벡터 데이터 샘플 세트(228(0)-228(X))를 역확산함
244
출력 데이터 흐름 경로들(98(0)-98(X))로부터의 역확산된 결과적인 출력 벡터 데이터 샘플 세트(229(0)-229(Z))를 벡터 데이터 파일들(82(0)-82(X))에 저장함
246

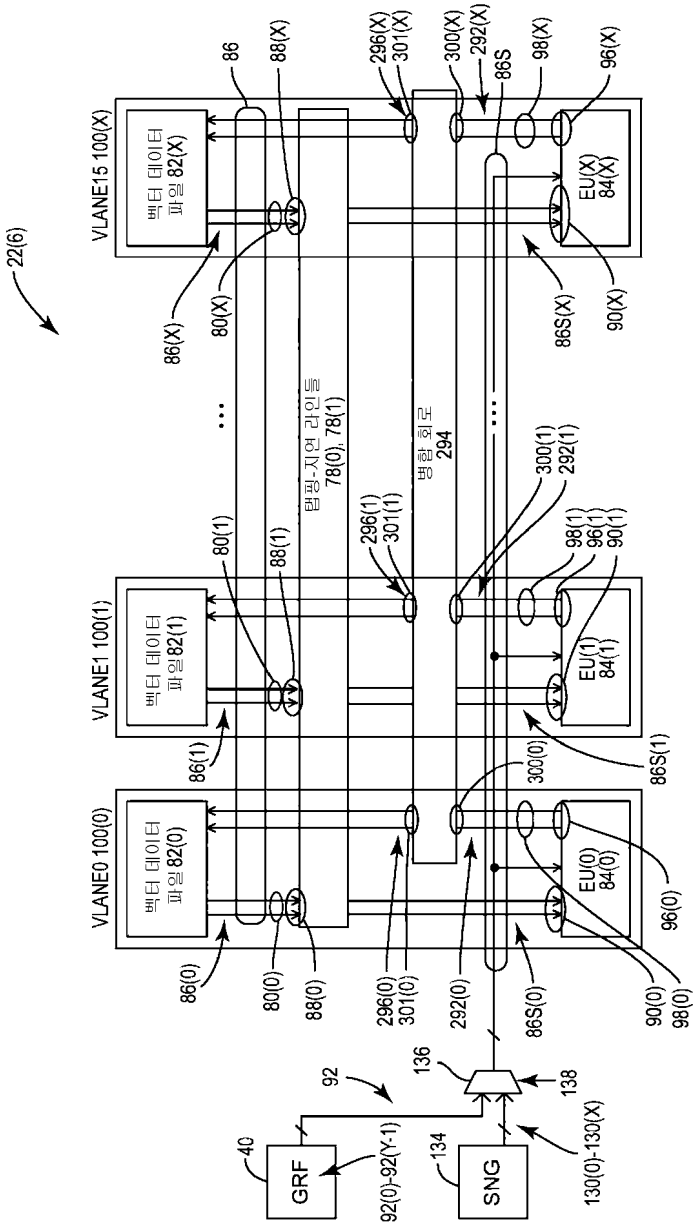
도면29


230
100(0)
100(1)
228(0)
228(1)
251(0)
251(1)
228(2)
228(5)
250((X+1)/2)-1), 250(7)
250((X+1)/2)-2)
100(X-1)
100(X)
228(X-1)
251(X-1)
228(X-2)
228(X)
251(X)
229(0)-228(X)
98(0)-98(X)
248(1)
252(0)
250(0)
250(1)
250(2)
250(3)
252((X+1)/2)-1),
252(7)
255(0)
255(((X-1)/2)-1), 255(7)
SF4
256(0)
254(0)
254(1)
248(2)
257(0)
254(2)
254(((X+1)/4)-1), 254(3)
256(((X+1)/4)-1), 256(3)
257(((X+1)/4)-1), 257(3)
SF8
258(0)
260(0)-
260(X+1)*4-1),
260(63)
286
248(3)
258(((X+1)/8)-1), 258(1)
260(((X+1)/8)-1), 260(1)
260(0)
259(0)
SF16
262
248
248(4)
264, 264'
266
268
259(((X+1)/8)-1),
259(1)
SF32
270
248(5)
SF64
SF128
SF256
273
272
Min/Max/Acc/
SF32+
274
278(0)
278(1)
278(2)
278(((X+1)/4)-1),
278(3)
276(0)
280(0)
280(1)
280(((X+1)/2)-1),
280(7)
282
284
284(0)
284(1)
284(2)
284(3)
284(4)
284(8)
284(9)
276(0)
276(1)
276(2)
276(3)
276(X-2)
276(X-1)
276(X)
284(X-1), 284(14)
284(X), 284(15)
229(0)-229(Z)

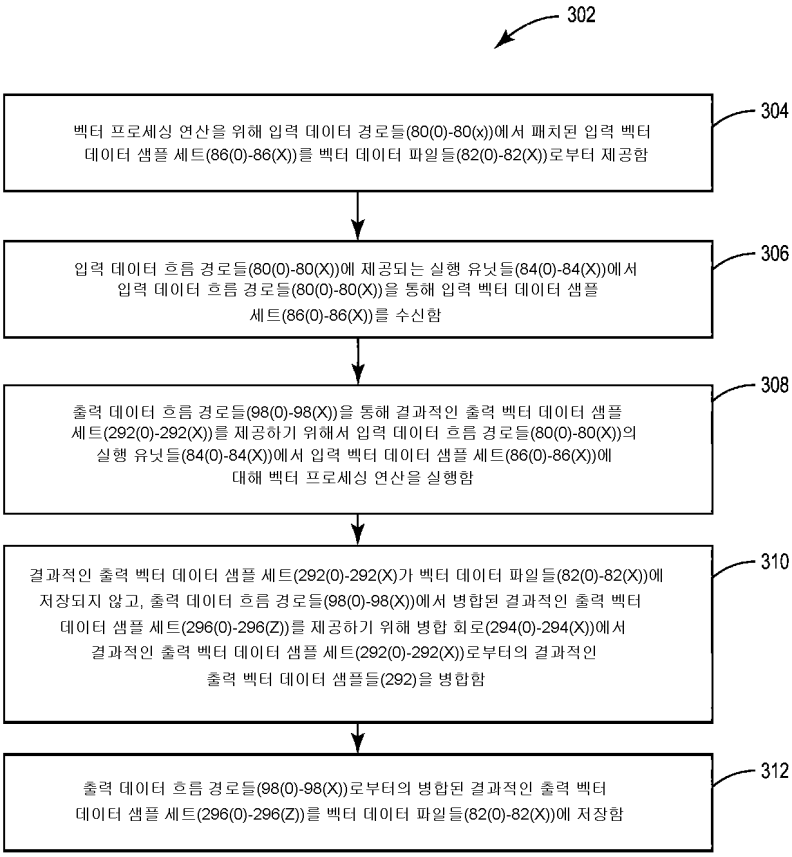
도면30


100(0)
백터 데이터 파일
82(0)
80(0)
OxFA1835CE
290(0)
+
100(1)
백터 데이터 파일
82(1)
80(1)
Ox1837FB80
290(1)
=
Ox11250314E

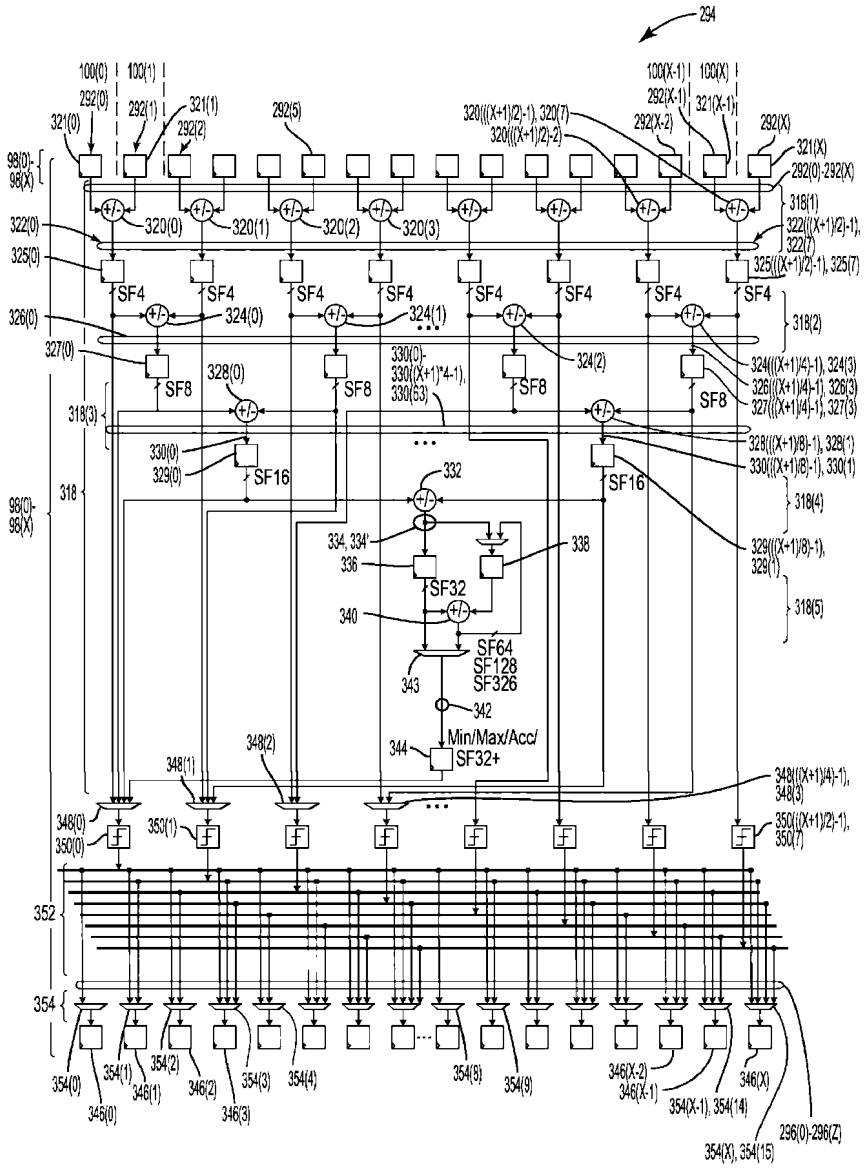
도면31


22(6)
VLANE15 100(X)
VLANE1 100(1)
VLANE0 100(0)
벡터 데이터 파일 82(X)
벡터 데이터 파일 82(1)
벡터 데이터 파일 82(0)
EU(X) 84(X)
EU(1) 84(1)
EU(0) 84(0)
탭핑-지연 라인들 78(0), 78(1)
병합 회로 294
GRF
SNG
40
92
92(0)-92(Y-1)
134
136
138
130(0)-130(X)
86
86(X)
86(1)
86(0)
80(X)
80(1)
80(0)
88(X)
88(1)
88(0)
296(X)
301(X)
296(1)
301(1)
296(0)
301(0)
300(X)
292(X)
300(1)
292(1)
292(0)
86S
86S(X)
86S(1)
86S(0)
98(X)
96(X)
90(X)
98(1)
96(1)
90(1)
300(0)
96(0)
90(0)
98(0)

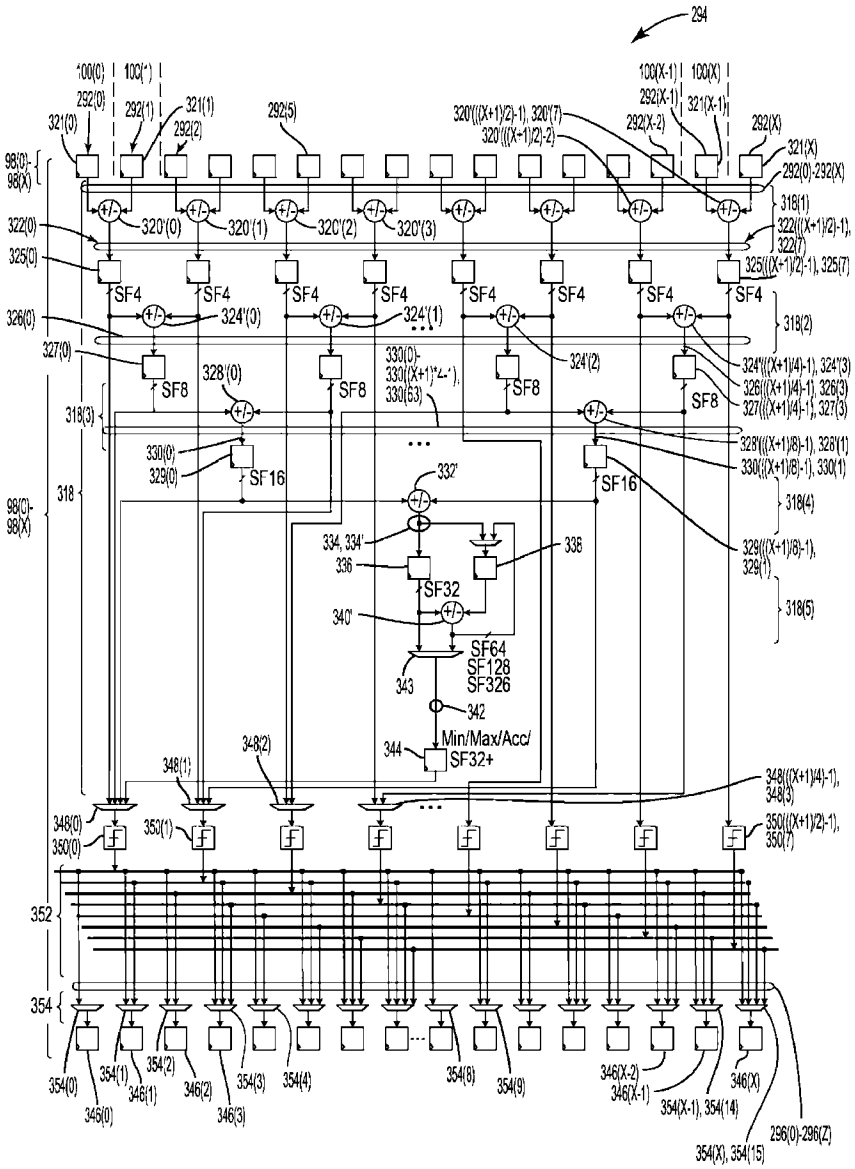
도면32


302
벡터 프로세싱 연산을 위해 입력 데이터 경로들(80(0)-80(x))에서 패치된 입력 벡터 데이터 샘플 세트(86(0)-86(X))를 벡터 데이터 파일들(82(0)-82(X))로부터 제공함
304
입력 데이터 흐름 경로들(80(0)-80(X))에 제공되는 실행 유닛들(84(0)-84(X))에서 입력 데이터 흐름 경로들(80(0)-80(X))을 통해 입력 벡터 데이터 샘플 세트(86(0)-86(X))를 수신함
306
출력 데이터 흐름 경로들(98(0)-98(X))을 통해 결과적인 출력 벡터 데이터 샘플 세트(292(0)-292(X))를 제공하기 위해서 입력 데이터 흐름 경로들(80(0)-80(X))의 실행 유닛들(84(0)-84(X))에서 입력 벡터 데이터 샘플 세트(86(0)-86(X))에 대해 벡터 프로세싱 연산을 실행함
308
결과적인 출력 벡터 데이터 샘플 세트(292(0)-292(X)가 벡터 데이터 파일들(82(0)-82(X))에 저장되지 않고, 출력 데이터 흐름 경로들(98(0)-98(X))에서 병합된 결과적인 출력 벡터 데이터 샘플 세트(296(0)-296(Z))를 제공하기 위해 병합 회로(294(0)-294(X))에서 결과적인 출력 벡터 데이터 샘플 세트(292(0)-292(X))로부터의 결과적인 출력 벡터 데이터 샘플들(292)을 병합함
310
출력 데이터 흐름 경로들(98(0)-98(X))로부터의 병합된 결과적인 출력 벡터 데이터 샘플 세트(296(0)-296(Z))를 벡터 데이터 파일들(82(0)-82(X))에 저장함
312

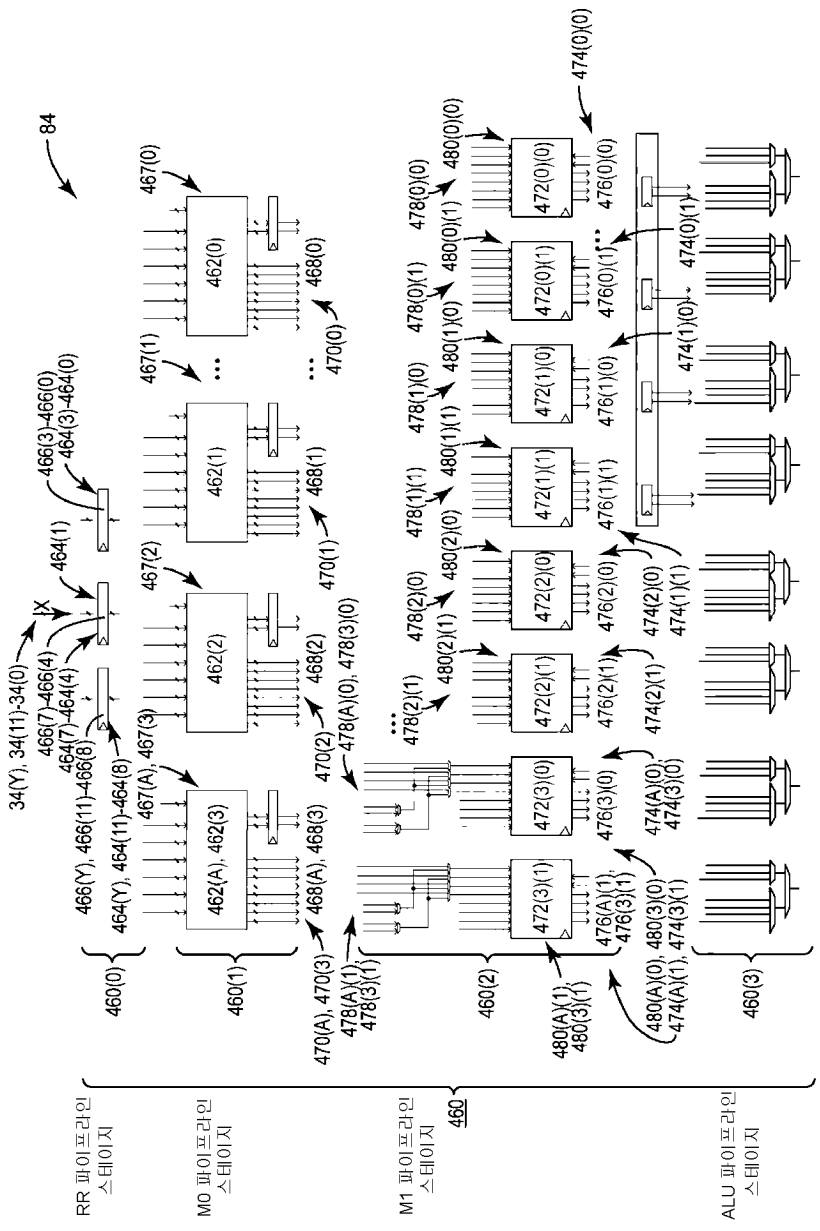
도면33



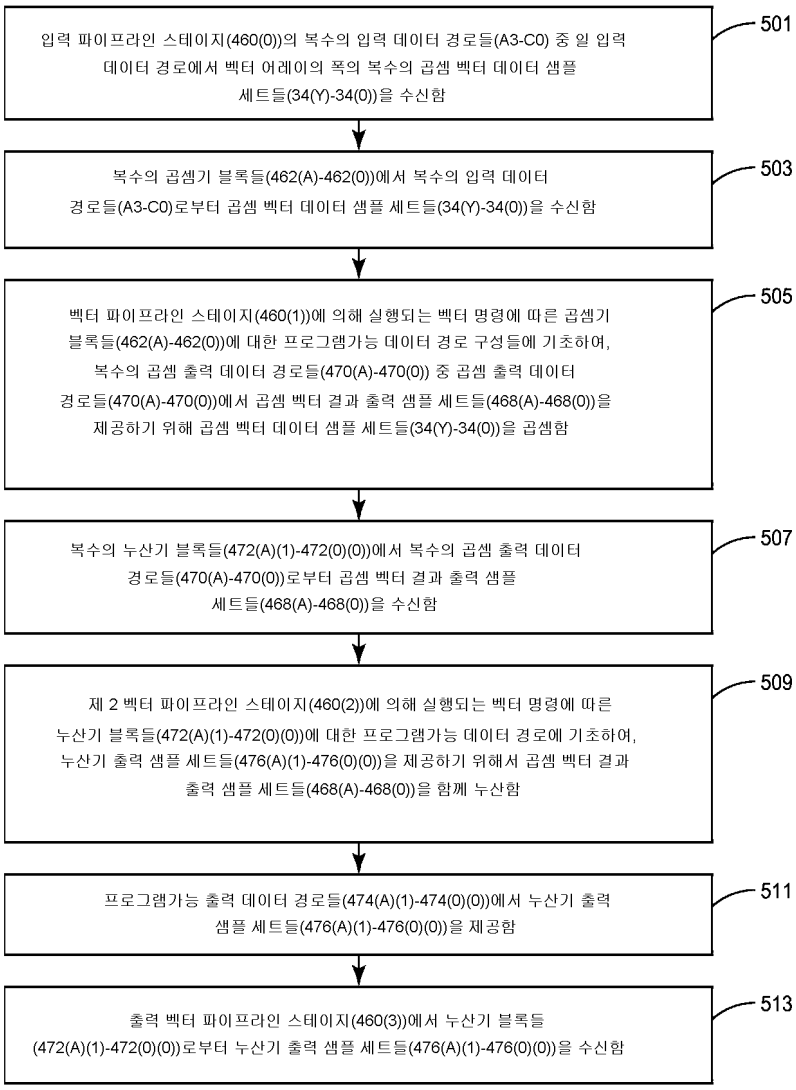
도면34



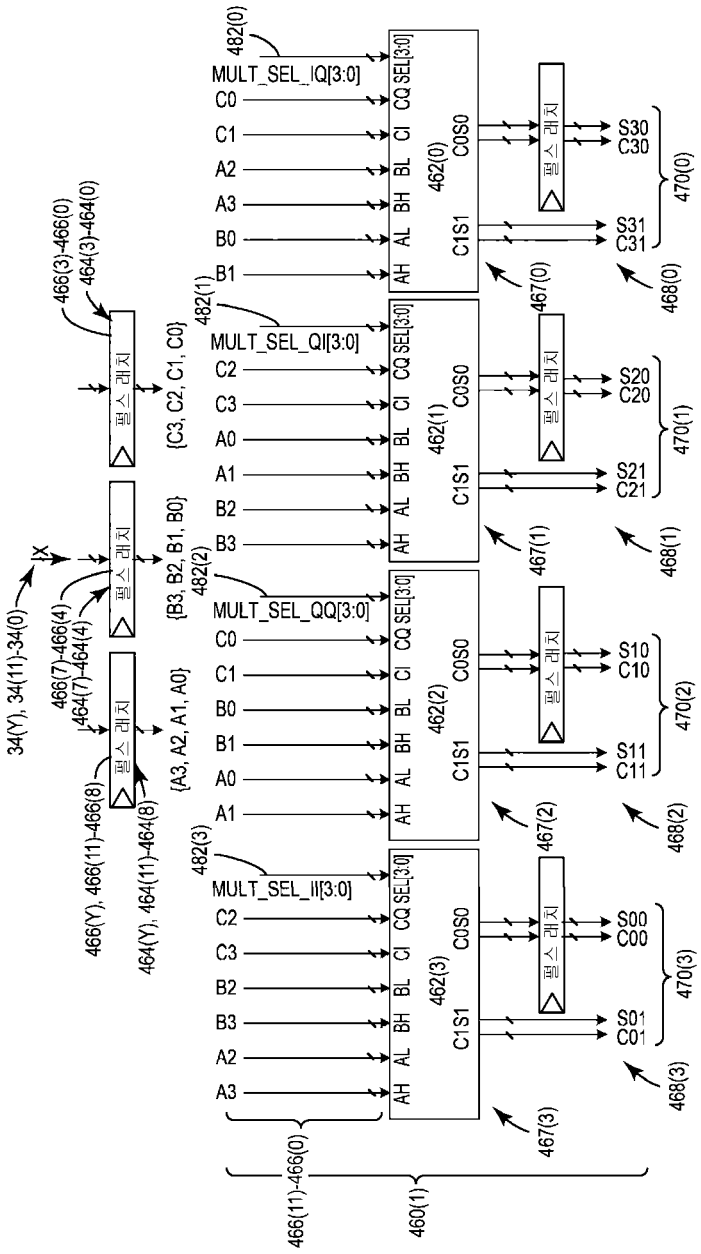
도면35


84
34(Y), 34(11)-34(0)
466(Y), 466(11)-466(8)
464(Y), 464(11)-464(8)
466(7)-466(4)
464(7)-464(4)
466(3)-466(0)
464(3)-464(0)
464(1)
462(A), 462(3)
462(2)
462(1)
462(0)
467(A), 467(3)
467(2)
467(1)
467(0)
468(A), 468(3)
468(2)
468(1)
468(0)
470(A), 470(3)
470(2)
470(1)
470(0)
478(A)(1), 478(3)(1)
478(A)(0), 478(3)(0)
478(2)(1)
478(2)(0)
478(1)(1)
478(1)(0)
478(0)(1)
478(0)(0)
480(A)(1), 480(3)(1)
480(A)(0), 480(3)(0), 474(A)(1), 474(3)(1)
480(2)(1)
480(2)(0)
480(1)(1)
480(1)(0)
480(0)(1)
480(0)(0)
476(A)(1), 476(3)(1)
476(3)(0), 474(A)(0), 474(3)(0)
476(2)(1)
474(2)(1)
476(2)(0)
474(2)(0)
476(1)(1)
474(1)(1)
476(1)(0)
474(1)(0)
476(0)(1)
474(0)(1)
476(0)(0)
474(0)(0)
472(3)(1)
472(3)(0)
472(2)(1)
472(2)(0)
472(1)(1)
472(1)(0)
472(0)(1)
472(0)(0)
460(0)
460(1)
460(2)
460(3)
460
RR 파이프라인 스테이지
M0 파이프라인 스테이지
M1 파이프라인 스테이지
ALU 파이프라인 스테이지

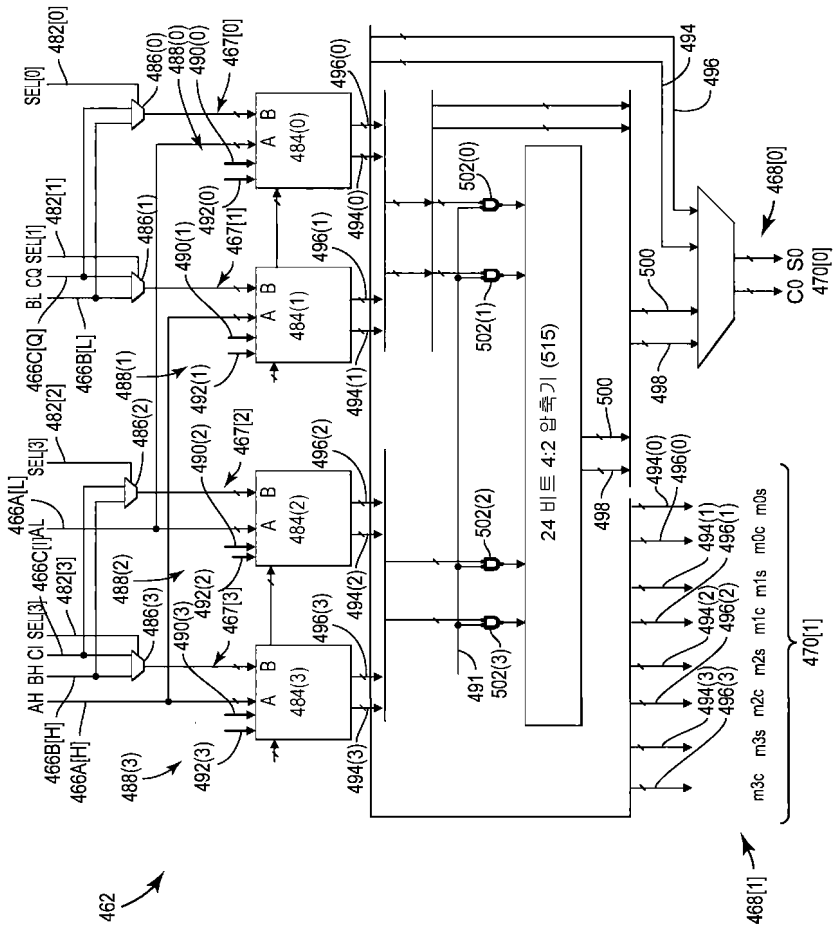
도면36


입력 파이프라인 스테이지(460(0))의 복수의 입력 데이터 경로들(A3-C0) 중 일 입력 데이터 경로에서 벡터 어레이의 폭의 복수의 곱셈 벡터 데이터 샘플 세트들(34(Y)-34(0))을 수신함
501
복수의 곱셈기 블록들(462(A)-462(0))에서 복수의 입력 데이터 경로들(A3-C0)로부터 곱셈 벡터 데이터 샘플 세트들(34(Y)-34(0))을 수신함
503
벡터 파이프라인 스테이지(460(1))에 의해 실행되는 벡터 명령에 따른 곱셈기 블록들(462(A)-462(0))에 대한 프로그램가능 데이터 경로 구성들에 기초하여, 복수의 곱셈 출력 데이터 경로들(470(A)-470(0)) 중 곱셈 출력 데이터 경로들(470(A)-470(0))에서 곱셈 벡터 결과 출력 샘플 세트들(468(A)-468(0))을 제공하기 위해 곱셈 벡터 데이터 샘플 세트들(34(Y)-34(0))을 곱셈함
505
복수의 누산기 블록들(472(A)(1)-472(0)(0))에서 복수의 곱셈 출력 데이터 경로들(470(A)-470(0))로부터 곱셈 벡터 결과 출력 샘플 세트들(468(A)-468(0))을 수신함
507
제 2 벡터 파이프라인 스테이지(460(2))에 의해 실행되는 벡터 명령에 따른 누산기 블록들(472(A)(1)-472(0)(0))에 대한 프로그램가능 데이터 경로에 기초하여, 누산기 출력 샘플 세트들(476(A)(1)-476(0)(0))을 제공하기 위해서 곱셈 벡터 결과 출력 샘플 세트들(468(A)-468(0))을 함께 누산함
509
프로그램가능 출력 데이터 경로들(474(A)(1)-474(0)(0))에서 누산기 출력 샘플 세트들(476(A)(1)-476(0)(0))을 제공함
511
출력 벡터 파이프라인 스테이지(460(3))에서 누산기 블록들 (472(A)(1)-472(0)(0))로부터 누산기 출력 샘플 세트들(476(A)(1)-476(0)(0))을 수신함
513

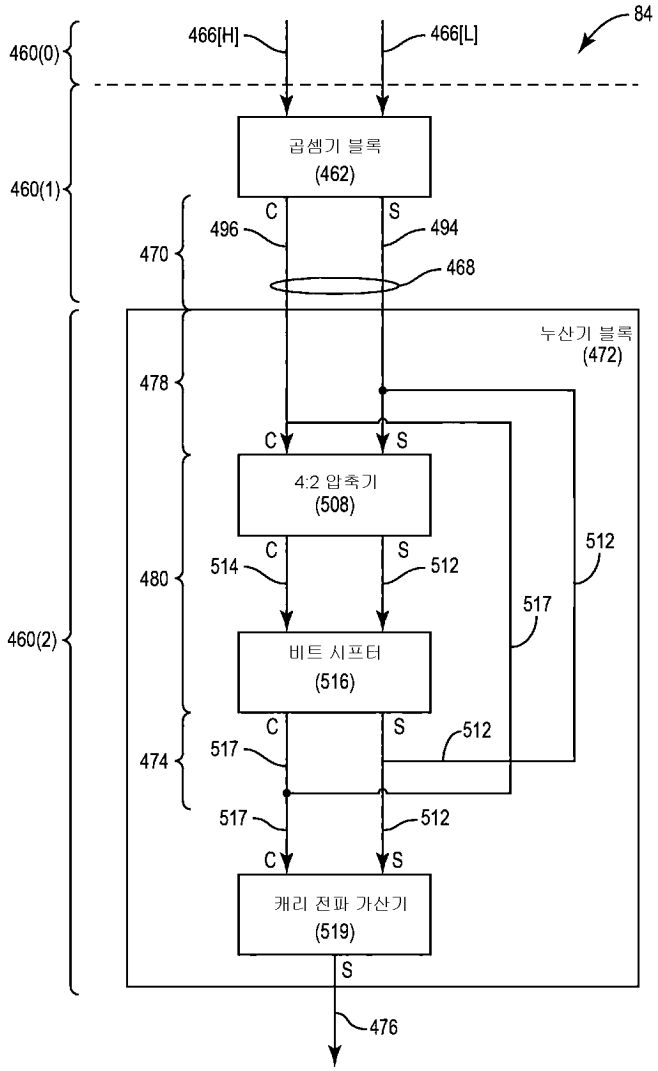
도면37


482(0)
MULT_SEL_IQ[3:0]
C0
C1
A2
A3
B0
B1
462(0)
C0S0
C1S1
467(0)
468(0)
470(0)
S30
C30
S31
C31
482(1)
MULT_SEL_QI[3:0]
C2
C3
A0
A1
B2
B3
462(1)
467(1)
468(1)
470(1)
S20
C20
S21
C21
482(2)
MULT_SEL_QQ[3:0]
C0
C1
B0
B1
A0
A1
462(2)
467(2)
468(2)
470(2)
S10
C10
S11
C11
482(3)
MULT_SEL_II[3:0]
C2
C3
B2
B3
A2
A3
462(3)
467(3)
468(3)
470(3)
S00
C00
S01
C01
466(11)-466(0)
460(1)
34(Y), 34(11)-34(0)
466(3)-466(0)
464(3)-464(0)
466(7)-466(4)
464(7)-464(4)
466(Y), 466(11)-466(8)
464(Y), 464(11)-464(8)
{C3, C2, C1, C0}
{B3, B2, B1, B0}
{A3, A2, A1, A0}
펄스 래치
AH AL BH BL CI CO SEL[3:0]

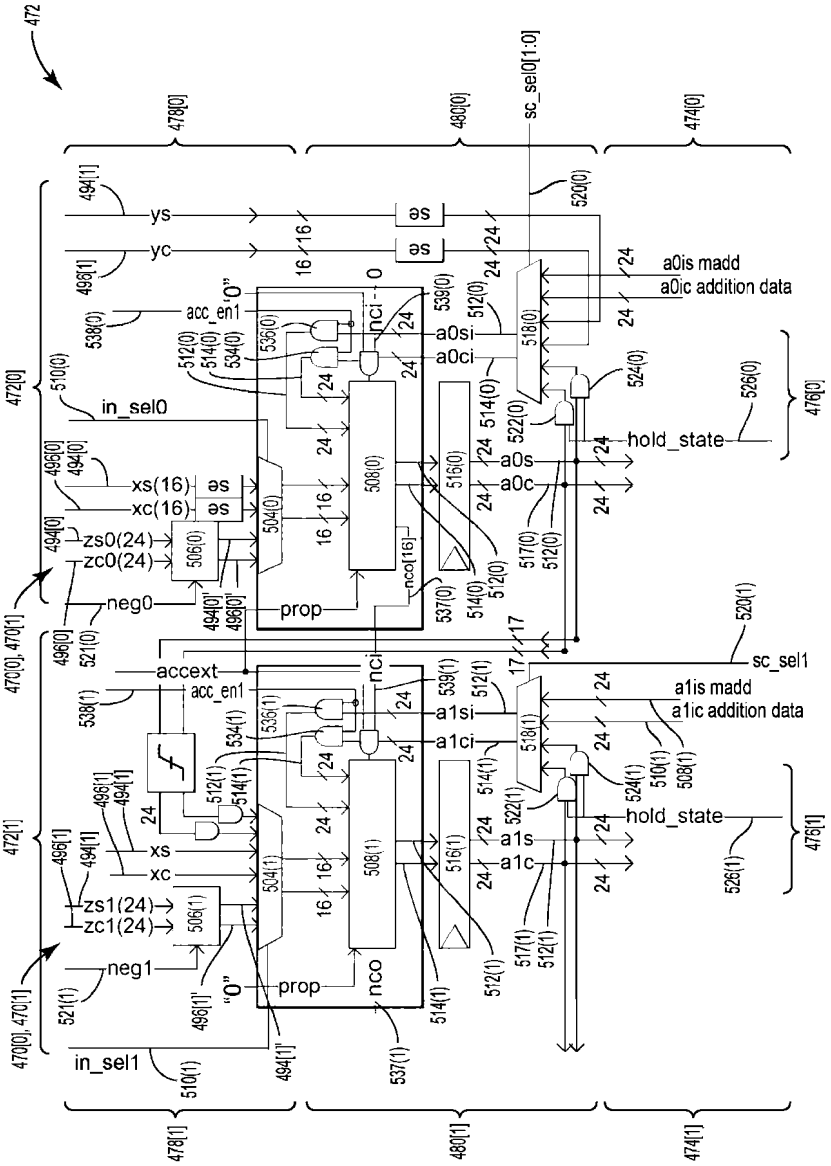
도면38


462
SEL[0]
482[0]
486(0)
488(0)
490(0)
467[0]
484(0)
496(0)
494(0)
482[1]
SEL[1]
BL CQ
466C[Q]
466B[L]
486(1)
490(1)
492(0)
467[1]
484(1)
496(1)
494(1)
488(1)
492(1)
482[2]
SEL[2]
466A[L]
466C[I]AL
486(2)
490(2)
467[2]
484(2)
496(2)
494(2)
488(2)
492(2)
482[3]
SEL[3]
AH BH CI
466B[H]
466A[H]
486(3)
490(3)
467[3]
484(3)
496(3)
494(3)
488(3)
492(3)
502(0)
502(1)
502(2)
502(3)
491
24 비트 4:2 압축기 (515)
500
498
494
496
468[0]
C0 S0
470[0]
494(0)
496(0)
494(1)
496(1)
494(2)
496(2)
494(3)
496(3)
m0s m0c m1s m1c m2s m2c m3s m3c
470[1]
468[1]

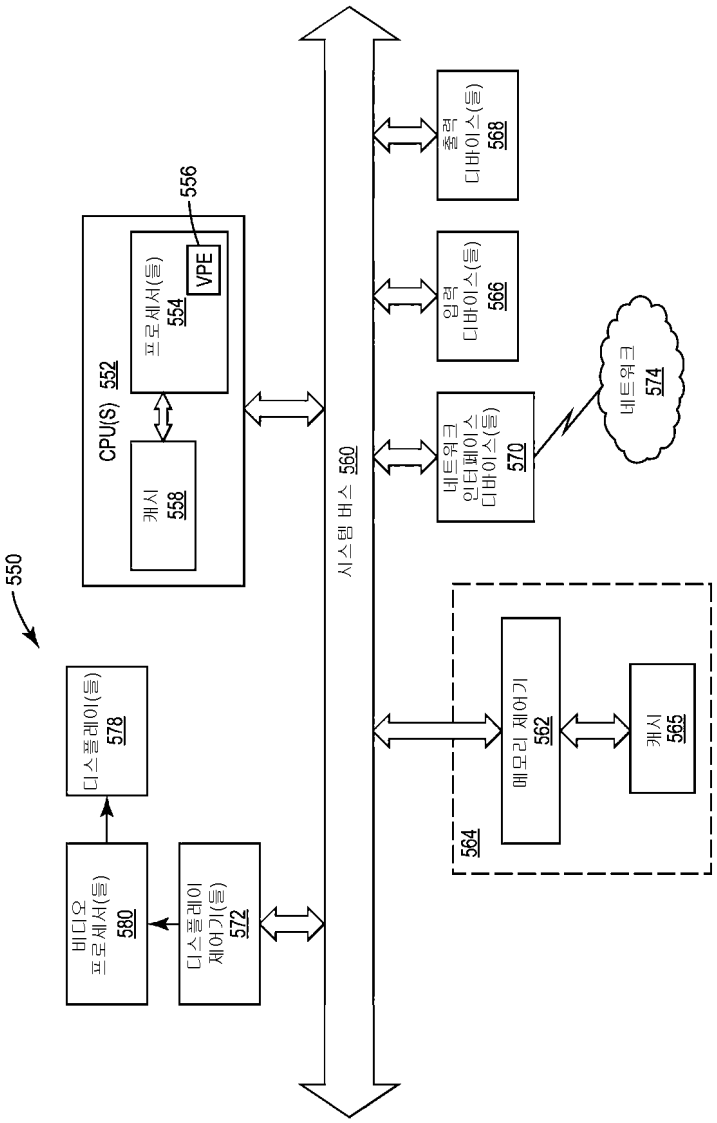
도면39


84
466[H]
466[L]
460(0)
곱셈기 블록
(462)
460(1)
C
S
496
494
470
468
누산기 블록
(472)
478
C
S
4:2 압축기
(508)
C
S
514
512
480
512
517
460(2)
비트 시프터
(516)
C
S
512
517
474
517
512
C
S
캐리 전파 가산기
(519)
S
476

도면40


472
sc_sel0[1:0]
478[0]
480[0]
474[0]
494[1]
ys
yc
496[1]
acc_en1
538(0)
536(0)
512(0)
514(0)
534(0)
nci
"0"
539(0)
a0si
a0ci
518(0)
a0is madd
a0ic addition data
520(0)
524(0)
526(0)
476[0]
472[0]
510(0)
in_sel0
hold_state
522(0)
a0s
a0c
516(0)
508(0)
504(0)
506(0)
xs(16)
xc(16)
zs0(24)
zc0(24)
neg0
prop
nco[16]
537(0)
517(0)
494[0]
496[0]
521(0)
470[0], 470[1]
accext
sc_sel1
520(1)
538(1)
536(1)
534(1)
539(1)
a1si
a1ci
518(1)
a1is madd
a1ic addition data
524(1)
510(1)
508(1)
514(1)
522(1)
a1s
a1c
516(1)
504(1)
506(1)
xs
xc
zs1(24)
zc1(24)
neg1
nco
526(1)
476[1]
517(1)
512(1)
537(1)
in_sel1
510(1)
478[1]
480[1]
474[1]

도면41


550
CPU(S) 552
프로세서(들) 554
VPE
556
캐시 558
시스템 버스 560
출력 디바이스(들) 568
입력 디바이스(들) 566
네트워크 인터페이스 디바이스(들) 570
네트워크 574
564
메모리 제어기 562
캐시 565
디스플레이(들) 578
비디오 프로세서(들) 580
디스플레이 제어기(들) 572