(19)대한민국특허청(KR)
(12) 등록특허공보(B1)

(51) 。Int. Cl.
G06F 9/38 (2006.01)

(45) 공고일자 2007년06월21일
(11) 등록번호 10-0731371
(24) 등록일자 2007년06월15일

(21) 출원번호 10-2000-7011394
(22) 출원일자 2000년10월13일
심사청구일자 2005년01월26일
번역문 제출일자 2000년10월13일
(86) 국제출원번호 PCT/EP2000/000590
국제출원일자 2000년01월26일
(65) 공개번호 10-2001-0042690
(43) 공개일자 2001년05월25일
(87) 국제공개번호 WO 2000/49496
국제공개일자 2000년08월24일

(81) 지정국 국내특허 : 일본, 대한민국,

EP 유럽특허 : 오스트리아, 벨기에, 스위스, 사이프러스, 독일, 덴마크, 스페인, 핀란드, 프랑스, 영국, 그리스, 아일랜드, 이탈리아, 룩셈부르크, 모나코, 네덜란드, 포르투칼, 스웨덴,

(30) 우선권주장 99200431.7 1999년02월15일 유럽특허청(EPO)(EP)

(73) 특허권자
코닌클리즈케 필립스 일렉트로닉스 엔.브이.
네덜란드 엔엘-5621 베에이 아인드호펜 그로네보드세베그 1

(72) 발명자
드올리베이라카스트룹페레이라베르나르도
네덜란드엔엘-5656에이에이아인드호펜홀스트란6

빙크아드리아누스제이
네덜란드엔엘-5656에이에이아인드호펜홀스트란6

후게르브루게잔
네덜란드엔엘-5656에이에이아인드호펜홀스트란6

(74) 대리인
김창세
장성구

(56) 선행기술조사문헌
EP 0825540
US 5,684,980
US 5,748,979
JP10105402
US 4,763,242
KR 10-1998-0018874

심사관 : 노지명

전체 청구항 수 : 총 10 항

(54) 구성가능 기능 유닛을 포함하는 프로세서를 사용해서 컴퓨터 프로그램을 실행하는 방법, 프로세서 및 컴퓨터 판독가능 기록 매체

(57) 요약

소정의 프로세서는 재구성가능 명령(reconfigurable instructions)을 실행할 수 있으며, 그 효과가 실행 시간에 구성 프로그램(configuration program)을 로딩(loading)함으로써 재정의될 수 있는 구성가능 기능 유닛(configurable functional unit)을 포함한다. 재구성가능 명령은 하나이상의 상이한 구성가능 명령의 조합(combination)으로 선택된다. 각각의 명령의 조합에 대한 각각의 구성 프로그램이 생성된다. 실행 중에 조합중 하나로부터 명령이 필요로 할 때, 그리고 구성가능 기능 유닛이 상기 조합에 대한 구성 프로그램으로 구성되지 않을 때마다, 상기 조합에 대한 모든 명령의 구성 프로그램은 구성가능 기능 유닛에 로딩된다. 재구성가능 명령은 조합의 어느 명령이 실행되어야할지를 선택한다.

특허청구의 범위

청구항 1.

구성가능 기능 유닛(configurable functional unit)을 포함하며, 재구성가능(reconfigurable) 명령을 실행할 수 있고, 실행시에 구성 프로그램(configuration program)을 로딩(loading)함으로써 그 효과가 재정의될 수 있는 프로세서(processor)를 사용해서 컴퓨터 프로그램을 실행하는 방법에 있어서,

상기 재구성가능 명령의 조합(combination)을 선택하는 단계와,

각각의 조합에 대한 각각의 구성 프로그램을 생성하는 단계와,

상기 컴퓨터 프로그램을 실행하는 단계와,

실행동안에 상기 조합 중 하나로부터의 명령이 필요하지만, 상기 구성가능 기능 유닛이 상기 조합에 대한 상기 구성 프로그램을 이용해서 구성되지 않을 때마다, 상기 조합의 모든 상기 명령에 대한 상기 구성 프로그램을 상기 구성가능 기능 유닛으로 로딩하는 단계를 포함하는 컴퓨터 프로그램 실행 방법.

청구항 2.

제 1 항에 있어서,

상기 구성 프로그램을 생성하는 단계는 적어도 하나의 상기 구성 프로그램내에서 상이한 명령에 대한 하드웨어 자원 이용을 상호최소화(cross-minimizing)하는 단계를 포함하는

컴퓨터 프로그램 실행 방법.

청구항 3.

제 2 항에 있어서,

프로그램 실행동안에 상기 조합으로부터 서로 다른 명령들을 선택하여, 상기 서로 다른 명령들에 따라 오퍼랜드(operand) 데이터를 처리하기 위한 하드웨어 자원 이용이 상호 최소화되는(cross-minimized)

컴퓨터 프로그램 실행 방법.

청구항 4.

제 2 항에 있어서,

상기 구성가능 기능 유닛은 오퍼랜드 데이터의 입력단과 연결 라인 사이에 교차점 스위치를 포함하되, 상기 연결 라인은 교차점 스위치의 각각의 출력단을 상이한 논리 조합 회로에 연결하고, 상기 교차점 스위치는 상기 구성 프로그램에 의해서 제어되고, 상기 교차점 스위치 내의 상기 연결의 프로그래밍(programming)은 상기 구성가능 기능 유닛 내에서 하드웨어 자원 이용을 상호 최소화하는 데에 이용되는

컴퓨터 프로그램 실행 방법.

청구항 5.

제 2 항에 있어서,

상기 조합으로부터 상기 구성가능 명령을 선택하는 적어도 일부의 비트가 상기 오퍼랜드 데이터와 교환 가능하게 상기 교차점 스위치에 공급되는

컴퓨터 프로그램 실행 방법.

청구항 6.

재구성가능 명령(reconfigurable instructions)을 실행할 수 있으며, 그 효과가 실행 시간에 구성 프로그램을 로딩함으로써 재정의될 수 있는 구성 가능 기능 유닛을 포함하는 프로세서로서,

상기 구성 가능 기능 유닛은, 구성가능 명령을 실행하는 데에 필요로하는 경우에 상기 재구성(reconfiguration) 프로그램이 아직 로딩되지 않았을 때에 상기 구성 프로그램의 로딩(loading)을 트리거(trigger)하는 로드 트리거링 회로(load triggering circuit)를 포함하며,

상기 재구성 프로그램은 적어도 2개의 조합된 구성가능 명령의 효과를 정의하며,

상기 로드 트리거링 회로는 상기 조합된 구성가능 명령 중 적어도 하나가 필요할 때에 모든 결합된 명령에 대하여 재구성 프로그램의 로딩을 트리거하는

프로세서.

청구항 7.

제 6 항에 있어서,

상기 구성가능 기능 유닛은 상기 조합으로부터 명령을 선택하는 명령 선택 입력단, 오퍼랜드 데이터 입력단, 조합 논리 및 결과 출력단을 포함하되,

상기 오퍼랜드 데이터 입력단 및 상기 명령 선택 입력단은 모두 상기 조합 논리를 경유하여 상기 결과 출력단에 연결되어서, 명령 선택 비트 및 오퍼랜드 데이터 비트가 상호 교환 가능하게 사용될 수 있는

프로세서.

청구항 8.

제 7 항에 있어서,

상기 구성가능 기능 유닛은 한편의 상기 조합 논리와 다른 한편의 상기 오퍼랜드 데이터 입력단 및 상기 명령 선택 입력단 사이의 교차점 스위치를 포함하여, 상기 교차점 스위치가 상기 명령 선택 비트 및 오퍼랜드 데이터 비트를 상기 조합 논리에 기능적으로 상호 교환 가능하게 연결할 수 있는

프로세서.

청구항 9.

제 6 항에 있어서,

상기 구성가능 기능 유닛은, 상기 조합의 상기 구성가능 명령을 실행하는 가능한 연결부의 분류 중 어느 것이 상기 구성 프로그램의 제어하에 연결되는지에 관계없이, 실질적으로 고정된 오퍼랜드 결과 지연(operand-result delay)을 전하는 프로그래밍 가능 논리 장치(programmable logic device)를 포함하는

프로세서.

청구항 10.

구성가능 프로세싱 유닛을 포함하는 프로세서용 기계 코드(machine code)를 생성하며 - 상기 기계 코드는 다수의 구성가능 명령을 포함함 - , 상기 구성가능한 프로세싱 유닛으로의 조합된 로딩을 위해 구성가능한 명령의 조합을 선택하도록 배열된 컴퓨터 프로그램을 구비한 컴퓨터 판독 가능 기록 매체에 있어서,

상기 선택하는 단계는

다수의 구성가능 명령 각각에 대하여 상기 구성가능 명령의 입력 오퍼랜드 중 어느 비트가 상기 구성가능 명령의 결과에 영향을 미치는지를 결정하는 단계와,

상이점의 정도(measure of dissimilarity)가 계산되는 모든 구성가능 명령의 결과에 영향을 미치지 않는 상기 입력 오퍼랜드의 비트의 수에 대응하는, 상기 구성가능 명령들 사이의 상이점의 정도를 결정하는 단계와,

상대적으로 낮은 상이점을 가진 가능한 조합이 상대적으로 더 높은 상이점을 가진 가능한 조합보다 선호되도록, 상이점의 정도에 기초하여 구성가능 명령의 조합을 선택하는 단계

를 포함하는 컴퓨터 프로그램을 구비한 컴퓨터 판독 가능 기록 매체.

명세서

기술분야

본 발명은 구성가능 기능 유닛(configurable functional unit)을 포함하며, 재구성가능 명령(reconfigurable instruction)을 실행할 수 있고, 실행시에 그 효과가 재정의될 수 있는 프로세서(processor)로 컴퓨터 프로그램을 실행하는 방법에 관한 것이다. 본 발명은 또한 이러한 방법을 이용하는 데이터 프로세서에 관한 것이다.

배경기술

구성가능 기능 유닛을 포함하는 프로세서로 컴퓨터 프로그램을 실행하는 방법은 마이클 제이 윈쓰린(Michael J. Wirthlin)과 브래드 엘 허칭스(Brad L. Hutchings)에 의해서 씌여졌으며, 1995년에 존 쉐웰(John Schewel)에 의해서 편집된 "Proceedings FPGAs for fast board development and reconfigurable computing"(Proceedings SPIE 2607)의 92 내지 103 페이지에 발행된 "동적 명령 세트 컴퓨터(DISC:The dynamic instruction set computer)"라는 제목의 논문으로부터 공지되어 있다.

상기 논문은 필드 프로그래밍 가능 게이트 어레이(field programmable gate array(FPGA))를 포함하는 기능 유닛을 가진 데이터 프로세서를 기술한다. FPGA는 출력 신호를 입력 신호의 함수로서 생성하는 회로이다. FPGA는 구성가능(configurable) 회로 소자의 행(row)과 열(column)로 구성된 행렬로 구성된다. 입력과 출력의 관계는 정보를 FPGA의 상이한 회로 소자와 이들 회로 소자들의 기능들 사이의 연결을 제어하는 메모리 셀(memory cell)로 로딩(loading)함으로써 구성될 수 있다.

구성 프로그램(configuration program)의 이용은 마이크로 프로그램(microprogarm)과 구분되어야 한다. 잘 알려진 바와 같이, 마이크로 프로그램은 기능 회로를 제어하는 데에 이용되는 개별적인 제어 신호를 정의한다. 상이한 마이크로 코드(microcode)의 실행 단계 및 상이한 명령들에 대하여 상이한 제어 신호가 정의된다. 이와는 대조적으로, 구성 프로그램의 비트를 저장하는 관계 메모리 셀은 입력 출력 관계에 영구적인 제어를 가진다. 즉, 실행되는 명령 또는 실행 단계에 관계없이 회로 소자를 영구히 제어한다. 통상적으로 제어된 입력 출력 관계는 시간 연속 회로 특성을 가진다.

구성 프로그램은 상이한 구성가능 명령들에 영향을 미치도록 발생된다. 윈쓰린(Winthlin)등의 논문에 의하면, FPGA 행렬은 회로 소자의 다수의 밴드의 행(row)으로 나누어진다. 각각의 구성 프로그램은 하나이상의 밴드를 차지하며 어떤 밴드 내에도 위치할 수 있다. 실행 시간에, 특정한 구성가능 명령과 직면하면, 이 명령에 대한 구성 프로그램이 이미 하나의 밴드에 로딩되었는지가 검사된다. 명령에 대한 구성 프로그램이 이미 하나의 밴드에 로딩된 경우에는 구성 프로그램을 이용하여 명령이 실행된다. 명령에 대한 구성 프로그램이 이미 하나의 밴드에 로딩되지 않은 경우에는 이 명령에 대한 구성 프로그램이 로딩되어서 구성 프로그램을 이용하여 명령이 실행된다.

단지 제한된 수의 구성 프로그램만이 동시에 로딩될 수 있다. 새로운 구성 프로그램을 로딩할 여유가 없는 경우에는, 다른 구성 명령에 대한 구성 프로그램이 밴드로부터 제거되어 새로운 구성 프로그램에 대한 여유를 만든다.

구성 프로그램이 로딩될 때마다 상당한 오버헤드(overhead)가 존재한다. 상기 논문에 따르면, 이 오버헤드는 구성 프로그램을 제거하여 다른 구성 프로그램을 로딩하기전에 구성 프로그램을 가능한한 오래동안 로딩된 상태로 유지함으로써 최소화된다. 이렇게 하여, 일종의 구성 프로그램의 캐슁(caching)이 구현되는데, 이것은 구성가능 명령이 되풀이해서 이용될 경우에 오버로드를 최소화한다. 하지만 여전히 구성 프로그램을 로딩하는 데에 상당한 오버헤드가 존재한다.

<u>발명의 개요</u>

본 발명의 목적은 구성 프로그램을 로딩하는 데에 필요한 오버헤드를 줄이는 것이다. 본 발명의 부가되는 목적은 함께 로딩된 상태로 유지될 수 있는 구성 프로그램의 수를 증가시켜 구성 프로그램이 좀 더 적게 로딩되도록 하는 것이다. 본 발명의 다른 목적은 컴퓨터 프로그램에 필요한 모든 구성 프로그램을 저장하는 데 필요한 메모리량을 줄이는 것이다.

발명의 상세한 설명

본 발명에 다른 컴퓨터 프로그램을 실행하는 방법의 실시예가 청구의 범위의청구항 1에 청구되어 있다. 이러한 실시예에 따라, 구성가능 명령들의 조합이 정의되며 개별적이지 않게 조합 내에 로딩된다. 상기 프로그램을 실행하기 전에, 각각은 적어도 2개의 구성가능 명령을 가지는 하나 혹은 그 이상의 조합이 선택된다. 통상적으로 각각의 조합은 컴퓨터 프로그램의 명령의 하나 혹은 그 이상의 순차적인 영역과 결합되어 있다. 프로그램의 특정한 영역이 실행될 때에, 이 영역에 대한 관계 조합에 대한 모든 구성가능 명령에 대한 구성 프로그램이 로딩된다.

명령의 조합과 그 관련 영역은 프로그램을 실행하기 전에 구성 프로그램을 로딩하는 데에 대한 오버헤드가 최소화되는 방식으로 선택될 수 있다. 즉, 다른 구성가능 명령들이 다른 조합을 로딩한 필요성을 야기하였을 경우에, 상기 조합에 대하여 선택된 구성가능 명령은 상기 조합에 속하지 않는 다른 구성가능 명령에 의한 중단없이 순서대로 발생한다. 따라서, 오버헤드를 줄이기 위하여 행해지는 작업은 실행 시간(run time)이라기 보다는 컴파일 시간(compile time)에 행해진다.

부가적으로, 많은 컴퓨터 프로그램에 대하여, 명령 싸이클 계수(instruction cycle count)는 오퍼랜드(operand)내의 같은 위치로부터의 비트의 사용 또는 유사하지만 조금 다른 논리 함수의 계산과 같은 강한 유사점을 가진 명령의 조합에 의해 최소화될 수 있다. 이들 명령은 조합내의 모든 명령에 의해 공동으로 사용된 하드웨어 자원과 개별적인 명령(또는 명령의 서브세트(subset))에 특유한 몇몇 하드웨어로 구현될 수 있다. 따라서, 상기 조합 내에 로딩될 수 있는 명령의 수는 증가된다.

본 발명에 따른 방법의 또 다른 실시예에 따라, 명령의 조합에 대한 구성 프로그램은 재구성가능 기능 유닛내의 조합 내에서 상이한 명령의 재구성가능 하드웨어 자원 이용을 상호 최소화(cross-minimize)하도록 선택되어진다. 다수의 기능의 자원 이용의 상호 최소화(cross-minimization)는 자원 이용이 기능에 대해 독립적으로 최소화되지 않고, 모든 기능을 수행하는 모든 구성 프로그램의 설계 공간(design space)내에서 최소값이 추구된다는 것을 의미한다. 조합 내의 상이한 명령간의 상호 최소화의 결과로서, 자원 이용이 각각의 명령에 대하여 독립적으로 최소화되는 경우에 조합에 대해 필요로하는 것보다 더 적은 하드웨어 자원을 필요로한다.

구성가능 기능 유닛내에서의 하드웨어 자원의 예는 회로 소자와 프로그래밍 가능한 연결(programmable connection)이다. 통상적인 구성가능 기능 유닛은 온(on) 혹은 오프(off)로 구성될 수 있으며 회로 소자를 서로 연결하거나, 회로 소자를 다른 타입의 회로 소자에 연결하거나 혹은 회로 소자를 기능 유닛의 입력단 또는 출력 단자에 연결하는 연결부를 가진 많은 동일 회로 소자를 포함한다. 통상적으로, 이러한 연결의 단지 제한된 수의 연결만이 구성될 수 있다. 예를 들면 단지 몇몇 회로소자가 입력단 혹은 출력단 혹은 주어진 다른 회로 소자에 직접 연결될 수 있다.

구성 프로그램이 상이한 명령에 대하여 독립적으로 선택된 경우에는, 하드웨어 자원이 실제로 이용되지 않더라도 조합 내에서 다른 명령에 대한 다른 구성 프로그램에 의해서 이용될 수 있도록, 조합 내의 명령에 대한 각각의 구성 프로그램은 하드웨어 자원을 이용하지 않고서 남겨두어야 한다. 상호 최소화에 의해서, 하나의 명령에 대한 구성 프로그램은 다른 명령에서 이용되지 않은 어떠한 하드웨어라도 이용할 수 있다.

심지어는, 조합 내의 다른 명령을 고려하지 않은 채 하나의 명령에 대한 구성 프로그램내에서 이용될 회로 소자를 선택하게 되면 이 선택된 소자를 입력단 혹은 출력단에 고정할 때에 부가적인 자원이 이용될 수 있다. 이것은 다른 명령에 대한 입력/출력 연결을 최선으로 선택함으로써 하드웨어 자원 이용을 최소화하는 가능성을 제거한다.

동일한 조합 내의 상이한 명령의 하드웨어 자원 이용을 상호 최소화함으로써, 하드웨어 자원의 낭비를 막을 수 있다. 더욱이, 상이한 명령간의 공통의 하드웨어 자원의 공유가 가능해진다. 하드웨어 이용을 상호 최소화함으로써, 공통 하드웨어 자원이 명령의 조합에 대해 1 회이상 할당되어야 하는 것이 회피된다.

본 발명에 따른 부가되는 실시예에 따르면, 조합 내의 상이한 명령의 선택 및 상기 상이한 명령에 따른 오퍼랜드 데이터의 프로세싱(processing)에 대한 하드웨어 자원 이용은 상호 최소화된다. 통상적으로, 명령 선택은 연산 코드를 오퍼랜드 데이터 프로세싱 회로를 인에이블(enable)시키는 신호로 복호화하는 과정을 포함한다. 실시예에서, 명령과 오퍼랜드 데이터 프로세싱의 구성 프로그램의 하드웨어 자원 이용이 서로 독립적으로 최소화되는 경우보다, 명령 선택과 오퍼랜드 데이터 프로세싱 모두에 좀 더 적은 하드웨어 자원이 필요로 한다.

바람직하게, 상기 프로세서는 파이프라인(pipeline)된다. 이것은 명령 프로세싱이 명령 복호화와 오퍼랜드 페치(fetch) 단계, 명령 실행 단계 및 결과 기록 단계와 같은 연속하는 단계로 분할되는 것을 의미한다. 파이프라인 프로세서에서 연속하는 명령의 다른 단계의 명령 프로세싱은 서로 병렬적으로 실행된다. 명령 프로세싱의 구성가능 부분은 실행 단계에서 발생한다. 본 발명의 실시예에 따라, 오퍼랜드 데이터 프로세싱 및 명령 선택 비트를 이용하여 상이한 명령들 사이에서 구분하는 것은 구성가능 명령의 프로세싱 실행 단계에서 발생한다.

본 발명에 따른 방법의 또 다른 실시예에 따라, 재구성가능 기능 유닛은 오퍼랜드 데이터의 입력단과 교차점 스위치(cross-point switch)의 각각의 출력단을 상이한 논리 조합 회로에 연결하는 연결 라인(line) 사이에 재구성가능 교차점 스위치를 포함한다.

실시예

도 1은 구성가능 명령을 지원하는 프로세서 구조의 실시예를 도시한다. 본 발명에 영향을 미치지 않는 이 프로세서 구조의 다양한 측면은 명료하게 하기 위하여 생략된 것에 주의하여야 한다. 예를 들면, 파이프라인 RSIC 구조가 고려되었지만, 본 발명은 이러한 구조에 한정되지는 않는다. 예를 들면, CISC 구조 혹은 DSP 구조가 대신 이용될 수 있다. 실시예는 상이한 파이프라인 단계를 분리하는 3 개의 레지스터(register)(10,14,19)를 가진 파이프라인 프로세서를 나타낸다. 명령 레지스터(10)는 파이프라인의 소스에 존재한다. 이 명령 레지스터(10)의 오퍼랜드 참조 필드 출력단은 레지스터 파일(12)의 입력단에 결합된다(예를 들면 이들 필드는 5 비트이다.). 이 레지스터 파일(12)의 출력단(w 비트, 즉 w=32)은 명령 레지스터(10)의 부가되는 출력단과 함께 실행 단계 레지스터(14)에 결합된다. 부가되는 출력단은 결과 주소 출력단(예를 들면 5 비트) 및 구성가능 명령 선택 코드(예를 들면 11 비트)를 포함한다.

레지스터 파일(12)로부터의 데이터를 통과시키는 실행 단계 레지스터(14)의 출력단은 ALU 기능 함수 유닛 및 구성가능 기능 유닛(18)에 병렬로 결합된다. ALU 기능 유닛(16)은 입력단이 각 멀티플렉서(multiplexer)(162,164)로 결합된 ALU(160)를 포함한다. 각각의 멀티플렉서(162,164)는 실행 단계 레지스터(14)의 출력단에 결합된 입력단을 가진다. 더구나 상기 멀티플렉서(162,164)는 명령 레지스터로부터 프로그램 계수 값 및 순간 값을 각각 수신하는 입력단을 가진다(이 입력단의 연결은 도시되지 않음).

실행 단계 레지스터(14)의 또 다른 출력단은 구성가능 명령 선택 코드를 구성가능 기능 유닛(18)으로 보내며, 결과 주소는 기록 단계 레지스터(writeback stage register)(19)로 보내진다. ALU 기능 유닛의 출력단 및 구성가능 기능 유닛(18)은 기록 단계 레지스터(19)에 연결된다. 상기 실행 단계 레지스터는 레지스터 파일(12)에 연결되어(도시되지 않음) ALU 기능 유닛(16) 또는 구성가능 기능 유닛(18)의 결과를 결과 오퍼랜드 주소에 의해서 지시된 위치의 레지스터 파일(12)에 기록한다.

또 다른 기능 유닛(예를 들면, 메모리 액세스 유닛) 제어 라인, 분기 회로(branching circuits), 명령 복호화 회로 및 레지스터(14,19)에 대한 입력단을 선택하는 멀티플렉서 등과 같은 다양한 회로는 명료성을 위하여 도 1에서 생략되었다.

동작시에, 도 1의 구조는 파이프라인화된 실행 메카니즘을 수행한다. 연속적인 명령이 연속적인 클럭 싸이클로 명령 레지스터(10)에 로딩된다. 명령이 로딩된 후 클럭 싸이클에서는 오퍼랜드 레퍼런스(operand reference)가 이용되어 상기 레지스터 파일(12)로부터 오퍼랜드를 로딩한다. 이 클럭 싸이클은 또한 명령 복호화를, 예를 들면 어느 기능 유닛(16,18)(혹은 도시되지 않은 다른 것들)이 명령을 실행할 것인지에 대한 선택을 포함할 수 있다. 이 클럭 싸이클의 마지막에서, 오퍼랜드, 결과 오퍼랜드 주소 및 구성가능 명령 선택 코드가 실행하는 데에 필요한 어떤 다른 데이터(도시되지 않음)와 함께 실행 단계 레지스터(14)에 로딩된다. 다음 클럭 싸이클에서, 이 정보는 기능 유닛(16,18)(및/또는 도시되지 않은 기능 유닛)에 보내져서 처리되어 결과를 획득한다. 이 다음 클럭 싸이클에서 선택된 기능 유닛(16,18) 및 결과 오퍼랜더 주소는 기록 단계 레지스터(19)에 로딩된다. 이 다음 싸이클후의 클럭 싸이클에서 결과가 레지스터 파일(12)에 기록된다.

명령의 연산 코드 필드가 구성가능 명령을 선택하는 경우, 구성가능 기능 실행 유닛(18)은 선택되어 명령을 수행하며 결과를 생성한다. 이 경우에, 명령 내의 구성가능 명령 선택 코드가 이용되어 어느 특정 구성가능 명령이 실행될지를 결정한다.

물론 본 발명의 범주에서 벗어나지 않고서 하나 이상의 구성가능 기능 유닛이, 각각 상이한 명령의 조합으로 구성되어 하나 이상의 조합이 동시에 이용가능하게끔, 요청된 구성 프로그램을 교환하는 오버헤드없이 병렬로 제공될 것이다.

도 2는 구성가능 기능 유닛의 실시예를 도시한다. 이것은 기본적으로 CPLD(Complex Programmable Logic Device) 코어(core)인데 그 자체로 공지되어 있다. 구성가능 기능 유닛은 입력 단자(20a,b,22)를 가져서 w 비트의 각각의 오퍼랜드와 N 비트의 구성가능 명령 선택 코드(예를 들면 N=4)을 수신한다. 상기 입력 단자는 교차점 스위치(24)의 입력단에 연결된다. 이 교차점 스위치(24)는 다수의 출력단을 가진다. 상기 교차점 스위치(24)는 각각의 2*w+ N 입력단이 구성가능 기능 유닛에 로딩된 구성 프로그램의 제어하에 어느 하나의 출력단에 연결되도록 설계된다.

교차점 스위치(24)의 출력단은 각각의 논리 블록(26a-b)에 연결된다. 논리 블록(26a-b)의 출력단은 구성가능 기능 유닛의 출력 단자에 결합된다. 예를 들면 2개의 논리 블록(26a-b)이 도시되는데, 각각은 36 개의 입력단과 w/2(예를 들면, 16)개의 출력단을 가진다. 2개의 논리 블록(26a-b)의 w/2 비트 출력단이 모여서 w(예를 들면, 32) 비트의 결과 출력을 구성한다.

구성가능 기능 유닛은 재구성 제어 회로(reconfiguration control circuit)(23)를 포함한다. 이 재구성 제어 회로(23)는 교차점 스위치(24)에 제공되지 않는 구성가능 명령 선택 코드의 비트를 수신하는 입력단을 포함한다. 재구성 제어 회로(23)는 교차점 스위치(24) 및 논리 블록(26a,26b)에 연결된 출력단을 가진다.

동작시에, 재구성 제어 회로(23)는 수신된 명령 선택 코드의 비트를 비교하며, 이 비트들을 구성가능 기능 유닛이 현재 실행하려고 하는 해당하는 재구성가능 명령의 조합의 비트와 비교한다. 바람직하게, 명령 선택 코드 비트의 서브세트(subset)가 이용되어 조합 및 조합 내에 구성가능 명령을 지시하는 남은 비트을 지시한다. 선택 코드가 상이한 조합으로부터의 명령이 실행되어야 한다고 지시하는 경우, 새로운 조합에 필요할 때에 재구성 제어 회로(23)는 메모리(도시되지 않음)로부터의 새로운 조합의 모든 명령에 대한 구성 프로그램을 로딩하며, 교차점 스위치(23) 및 논리 블록(26a,b)을 재프로그램한다. 이어서, 상기 새로운 조합으로부터의 명령이 실행될 것이다.

일단 새로운 조합이 로딩되거나 명령이 이미 로딩된 조합으로부터 선택되면, 구성가능 기능 유닛은 상기 명령을 처리한다. 이 경우에, 교차점 스위치에 인가된 N 명령 비트는 (구성가능 기능 유닛에 로딩된 명령의 조합으로부터의) 어느 명령에 따라 오퍼랜드가 처리되어야 할지를 결정한다.

구성 프로그램을 로딩하는 가장 쉬운 실시예는 구성 프로그램이 로딩될 때까지 프로세서에 의한 부가되는 명령 실행을 정지하는 것을 포함한다. 그러나, 더 적은 명령 싸이클 오버헤드를 필요로하는 다른 실시가 이용될 수도 있다. 예를 들면, 프리커서(precursor) 명령은 구성 프로그램의 로딩을 트리거(trigger)하는 데에 이용될 수 있다. 프리커서 명령 자체는 구성 프로그램을 필요로하지 않지만 지시된 구성 프로그램의 로딩을 트리거한다.

다른 예에서 프로세서는 정규(예를 들면, ALU) 명령의 서브루틴으로 점프(jump)하는데, 이것은 구성 프로그램이 로딩되거나 로딩되고 있는 경우에 구성가능 명령을 실행한다. 이것은 어드레스된(addressed) 오퍼랜드 레지스터의 내용을 서브루틴 호출 스택(stack)에 두며, 서브루틴을 호출하며, 서브루틴으로부터 되돌려준 후 호출 스택으로부터 구성가능 명령의 어드레스된 결과 레지스터로 결과를 되돌림으로써 실시될 것이다.

도 3은 도 2(그것 자체가 또한 CPLD로부터 공지 되었다.)의 구성가능 유닛에서 이용되는 논리 블록의 예의 실시예를 도시한다. 이러한 논리 블록은 PAL 어레이(array)(30) 및 PLA 어레이(32)를 포함하는데, 양자 모두 교차점 스위치(24)에 결합되었다. 어레이(30,32)의 출력단에서 AND 게이트(34a-b,35a-c)가 기호에 의해 도시된다. 예를 들면, PAL 어레이(30)에 대한 64 개의 AND 게이트(34a-b)와 PLA 어레이(32)에 대한 32개의 AND 게이트(35a-c)가 존재한다.

어레이(30,32)는 열(column) 도전체와 행(row) 도전체로 구성되는데(도시되지 않음), 각 열(column)은 교차점 스위치(2)의 각 출력단에 해당하며 각 행(row)은 논리 블록의 AND 게이트(34a-b,35a-c)에 해당한다. 행(row)과 열(column)의 교차점에는 트랜지스터와 메모리 셀이 존재한다(도시되지 않음). 메모리 셀은 트랜지스터가 활성화되었는지를 제어하며, 활성화되었을 때는 트랜지스터는 AND 게이트(34a-b,35a-c)의 입력단을 형성하며, AND 게이트(34a-b,35a-c)는 트랜지스터가 활성화된 열(column) 도전체의 논리 레벨의 AND 연산값을 출력한다.

PLA 어레이의 AND 게이트의 출력단(35a-c)은 행렬(33)의 행(row) 도전체에 연결된다. 이러한 어레이의 열(column) 도전체는 OR 게이트(36a-c)에 연결되어 도시되어 있다. 행(row)과 열(column)의 교차점에 트랜지스터와 메모리 셀이 존재한다(도시되지 않음). 메모리 셀은 트랜지스터가 활성화되었는지를 제어하며, 활성화되었을 때에 트랜지스터는 OR 게이트(36a-c)의 입력단을 형성하며, OR 게이트(36a-c)는 트랜지스터가 활성화된 행(row) 도전체의 논리 레벨의 OR 연산 값을 출력한다.

PAL 어레이(30)의 출력단은 4 개의 그룹으로 또 다른 OR 게이트(38a-d)에 연결된다. 각 OR 게이트(36a-d)는 부가되는 OR 게이트(38a-d)중의 각각의 하나의 입력단에 결합된 출력단을 가진다. 예를 들면, 16 개의 OR 게이트(36a-d)와 16 개의 또 다른 OR 게이트(38a-b)가 존재한다. 각각의 또 다른 OR 게이트(38a-b)는 프로그래밍 가능 인버터(inverter)/논인버터(noninverter)(39a-b)를 경유하여 논리 블록의 출력단 비트 라인에 연결된다. 메모리 셀(도시되지 않음)이 각각의 인버터/논인버터(39a-b)에 대하여 제공되며, 이들 메모리 셀의 내용이 이들 인버터/논인버터(39a-b)가 반전할지 그렇지 않을지를 제어한다.

논리 블록의 논리 기능은 구성 프로그램의 비트를 PAL 행렬(30)과 PLA 행렬(32)과 행렬(33)내의 교차점에서의 트랜지스터의 활성화 및 인버터/논인버터(39a-b)의 기능을 제어하는 메모리 셀에 로딩함으로써 프로그램된다.

N개의 비트의 명령 선택 코드가 교차점 스위치(24)에 인가된다. 이들 N개의 비트는 오퍼랜드의 비트와 같은 방식으로 인가된다. 구성가능 기능 유닛의 구성 프로그램은 이들 N개의 비트를 마치 오퍼랜드 비트처럼 자유로이 처리할 수 있다. N개의 비트가 먼저 결합하여 특정한 하나의 명령을 검출하며, 검출 결과는 데이터가 어떻게 처리되는지를 제어하는데에 이용되는 것은 필요하지 않다. 이와는 반대로, N개의 비트의 각각의 비트는 N개의 비트 중 다른 비트에 무관하게 논리 기능에서 오퍼랜드 비트를 가진 인자로 참여할 수 있다.

도 5는 명령의 조합으로부터 명령을 실행하도록 프로그램되었을 때에 기능 유닛의 하드웨어 기능 설명의 모델을 도시한다. 이 도식에 나타난 구조는 단지 기능적이며 물리적이지 않다는 점에 주의하여야 한다. 상이한 기능 블록으로의 분리는 구성가능 기능 유닛내에서 수행되는 회로의 구조의 분리에 해당할 필요가 없으며, 상이한 블록이 구성가능 기능 유닛내에서 동일한 물리적 회로 소자를 공유할 수 있다.

모델은 2개의 소스 오퍼랜드에 대한 입력단(50a,b)과 하나의 구성가능 명령 각각을 실행하는 많은 블록(52a-c)과 이들 명령에 대한 단자(54a-c)와 하나의 결과를 출력단(58)에 보내는 멀티플렉서(56)를 도시한다. 멀티플렉서(56)는 N 비트의 명령 선택 코드에 의해서 제어된다.

이 모델은 그 기능을 수행하게 될 연결의 리스트(list)로 변환될 것이다. 이 변환동안에, 자원 이용은 도 5의 다양한 블록 사이에서 상호최소화될 것이다. 멀티플렉서(56)의 기능은 블록(52a-c)의 기능과 (부분적으로)합쳐질 것이며, 이들 블록(52a-c)의 기능은 서로간에 합쳐질 것이다.

함께 로딩된 구성가능 명령 및 이들 명령의 조합은 바람직하게 프로세서에서 실행되는 각각의 특정 프로그램에 대하여 무관하게 선택된다. 아래에서 이들 명령 및 조합은 또한 각각 "커스텀 명령(custom instruction)" 과 "클러스터(cluster)"로 불리어질 것이다. 커스텀 명령과 클러스터의 선택은 바람직하게 컴퓨터 프로그램의 컴파일 과정에서, 즉 프로세서가 프로그램을 실행하기 전에 발생한다.

도 4는 컴파일된 프로그램을 생성하는 흐름도를 도시한다. 도 4의 흐름도는 다음 단계를 수행한다.

41 단계. 소스 코드(source code)(특히 C로 기록된)가 컴파일러(compiler) 프론트 엔드(front-end)에 의해서 처리되어 데이터 흐름도로 나타난 중간 코드를 생성한다.

42 단계. 중간 코드는 잠정적으로 하드웨어 합성(후보(candidate))에 적당한 데이터 흐름 세그먼트(segment)를 찾는 클러스터 검출/선택 모듈(module)에 의해서 판독된다. 각각의 "후보"는 커스텀 명령을 정의한다. 바람직하게 애플리케이션의 임계 경로(critical path)내에서 순전히 산술 또는 논리 연산만으로 구성된 세그먼트만이 고려된다. 프로파일(profile) 데이터는 검색을 안내하는 데에 이용된다. 후보는 특정한 기준에 따라서(이후의 논의 참조) 커스텀 명령의 클러스터로 함께 그룹(group)지워진다.

43 단계. 클러스터는 데이터 흐름 세그먼트의 산술 연산을 HDL(표준 하드웨어 정의 언어(a standard Hardware Definition Language))내의 하드웨어 기술(hardware description)로 변환하는 번역기(translator)에 의해서 처리된다. 이 하드웨어 기술에 복호화 논리가 부가되어 상이한 커스텀 명령이 독립적으로 실행된다. 도 5는 이 단계에서 생성된 회로 기술의 모델의 예를 도시한다.

44 단계. 결과로서 생긴 회로 기술이 타이밍(timing) 및 피팅(fitting) 리포트뿐만 아니라 회로 넷리스트(circuit netlist)가 생성되는 하드웨어 합성 도구에 의해서 처리된다. 이 단계에서 회로 기술내의 멀티플렉서의 기능은 도 5의 기능 블록의 자원 이용으로 상호최소화될 것이다. 이러한 자원의 상호최소화는 그 자체가 일반적인 기능을 가진 프로그래밍 가능 논리를 프로그래밍하는 것으로 알려져 있다.

45 단계. 타이밍(timing) 및 피팅(fitting) 정보가 클러스터 검출/선택 모듈(module)로 되돌려지는데, 이 클러스터는 재배열되거나 버려지거나 새로운 클러스터가 형성된다. 클러스터의 최종 세트가 선택될 때까지 싸이클은 계속된다.

46 단계. 클러스터의 최종 세트가 선택되면, 최종 선택된 클러스터내에서 실행된 중간 코드내의 데이터 흐름 세그먼트는 그들의 등가 커스텀 명령 레벨에 의해 대체된다.

47 단계. 그런 다음 결과로써 생긴 코드는 백 엔드(back-end)에 의해서 후처리된다(레지스터 할당(register allocation), 어셈블리 코드 이미션(assembly code emission), 명령 스케쥴링(instruction scheduling) 및/또는 적용될 수 있는 다른 후처리).

48 단계. 결과로서 생기는 어셈블리는 새로운 합성된 커스텀 명령 레벨을 인지하는 변형된 어셈블러에 보내진다. 하드웨어 합성 도구에 의해서 생성된 넷리스트는 어셈블리와 결합하여 최종적으로 실행가능한 것을 생성한다.

하드웨어 합성 단계는 인간 프로그래머로부터 완전히 숨겨질 수 있다. 이와는 달리, 바람직하게 아래에 기술된 지침에 따라 인간 프로그래머가 후보 및/또는 클러스터를 수동으로 선택하는 것이 필요할 수도 있다.

프로그램의 데이터 흐름도로부터의 후보 구성가능 명령의 선택은 그 자체로 공지되어 있다. 기본적으로 이것은 프로그램의 데이터 흐름도로부터 서브그래프(subgraph)를 선택하는 단계를 포함하는데, 여기서 서브그래프는 단지 2개의 가변 오

퍼랜드 입력단을 가진다(이것은 구성가능 기능 유닛이 2개의 오퍼랜드 입력단을 가지는 경우이다. 구성가능 기능 유닛이 더 적거나 더 많은 수의 오퍼랜드 입력단을 가지는 경우에는 이에 해당하는 더 많거나 더 적은 입력단을 가진 서브그래프가 선택될 것이다.).

바람직하게, 후보는 후보가 발생하는 프로그램내의 순차 명령의 영역에 기초하여 클러스터로 그룹지워진다. 동시에 로딩될 수 있는 구성 프로그램의 수보다 더 많은 클러스터가 정의되어서는 안된다(예를 들면, 단지 하나의 구성가능 기능 유닛이 존재하고 동시에 단지 하나의 구성 프로그램으로 구성될 수 있는 경우에는 단 하나만의 클러스터). 프로그램 영역의 크기와 상기 영역에 대하여 선택된 후보의 수는 모든 후보가 동시에 로딩된 클러스터의 최대의 수로 프로그램될 수 있도록 선택되어져야 한다.

원리적으로, 후보 명령의 선택과 그들의 클러스터로의 조합은 제한된 최적화 문제이다. 많은 세트의 구성가능 명령의 조합이 가능하다. 프로그램(실행 프로파일(profile)에 의해서 정의된)의 특정한 실행에 필요한 명령 싸이클의 수를 최소화하는 세트를 찾는 것이 목적이다. 명령 싸이클 계수의 최소화는 클러스터에 대한 모든 선택된 후보가 구성 프로그램에 적합하다는 제한을 받는다.

이러한 목적을 위하여, 각각이 단지 정규 명령만이 이용되는 경우에 더 많은 수의 명령 싸이클을 필요로하는 하나의 명령 싸이클에서 유효한 구성가능 명령을 고려할 것이다. 커스텀 명령에 의해서 대체될 프로파일내에서 모든 정규의 명령을 수행하는 데에 필요한 부가적인 명령 싸이클의 수는 구성 프로그램을 로딩하는 오버헤드 싸이클보다 많아야 한다. 그렇지 않다면, 어떠한 조합도 선택될 수 없다. 조합 내의 모든 명령에 대하여 로딩은 한 차례 발생할 필요가 있고, 상호 최소화에 기인하여 평균적으로 명령마다 보다 적은 구성 프로그램 공간을 차지하기때문에, 오버헤드 싸이클의 수는 같은 수의 명령을 개별적으로 로딩하는 것보다 조합을 로딩하는 것이 더 적다는 것에 주의하여야 한다.

클러스터로 결합될 수 있는 후보의 수는 구성가능 기능 유닛내의 자원 이용의 최소화에 달려있을 것이다. 구성가능 명령의 수가 더 "적거나" 좀 더 유사하다면, 더 많은 구성가능 명령이 하나의 클러스터로 결합될 수 있다.

클러스터를 선택하는 알고리즘(algorithm)의 예는

1) 컴파일러에 의해서 생성된 중간 코드내의 영역의 개시 명령과 종료 명령에 의하여 프로그램의 영역으로부터 영역을 선택한다. 바람직하게 자주 실행된 루프(loop)내의 명령 또는 자주 실행된 서브루틴(subroution)내의 명령을 선택하여야 하지만, 매우 자주 실행되지는 않지만 반복되는 유사한 명령을 포함하는 영역 또한 좋은 후보가 된다.

2) 선택된 영역내의 명령의 데이터 흐름 세그먼트에 대한 다수의 후보 커스텀 명령을 선택한다.

3) 선택된 커스텀 명령을 결합하는 클러스터에 대한 구성 프로그램이 생성될 수 있는지를 결정하여 선택된 커스텀 명령 모두가 함께 구성가능 기능 유닛내에 맞추어지며 완성하는 데에 파이프라인 사이클보다 적게 걸리도록 한다. 구성 프로그램이 생성될 수 있다면, 정규 명령의 조합을 선택된 후보 커스텀 명령으로 대체함으로써 프로파일에서 획득한 명령 싸이클의 수를 결정한다.

4) 더 큰 그리고 더 작은 영역 및 상이한 선택된 명령에 대하여 1 내지 3 단계를 되풀이하며, 선택된 영역 및 선택된 명령을 유지하며, 대부분의 명령 싸이클을 획득한 클러스터를 유지한다. 이러한 단계는 영역에서 발견된 가장 이득이 되는 클러스터에서 시작하며 이것을 부가되는 커스텀 명령과 함께 확장하여 발견적으로(heuristically) 속도가 높아질 것이며, 양자 모두는 동일한 영역으로부터 또는 이 영역의 확장으로 부터이다.

5) 이 모든 상이한 영역에 대하여 각각의 클러스터를 유지하면서 프로그램의 상이한 겹치지 않는 영역에 대하여 1 내지 4 단계를 되풀이한다.

최소화는 전체 프로그램에 걸치지 않고 프로그램의 영역에 국한되는 로컬 프로세스(local process)임에 주의하여야 한다. 요점은 클러스터가 특정 영역에 대하여 명령 싸이클 계수를 줄인다는 점이다. 다른 영역에서 무엇이 발생하는가는 중요하지 않는데 이것은 이들 영역에 대한 클러스터는 로딩될 필요가 없기 때문이다. 사실, 상이한 영역에 대한 상이한 클러스터는 동일한 효과를 가진 몇몇 커스텀 명령을 포함한 것이다. 동일한 효과를 가진 이들 커스텀 명령들중 하나가 실행된다면, 실행되는 영역은 어느 클러스터가 로딩되는지를 결정한다. 그러므로, 실행되어야하는 특정 명령에 의해서라기 보다는 실행되고있는 영역에 의해서 로딩된 구성 프로그램이 결정된다.

클러스터 및 영역의 선택은 이들 포인트를 선택하는 다수의 발견적(heuristic) 기준을 이용하여 단순화될 수 있다. 다양한 기준이 후보를 클러스터로 그룹(group)짓는 데에 이용될 수 있다.

예를 들면,

- 공통된 루프(loop)내의 후보는 동일한 클러스로 그룹지어진다(이것은 루프내의 재배치 오버헤드를 회피한다.).

- 서브루틴에서 발생하는 후보는 그 서브루틴에 대한 클러스터 또는 클러스터들로 그룹지어진다.

- 좀 더 낮은 논리 복잡도를 가진 후보는 좀 더 큰 클러스터로 그룹지어진다(좀 더 많은 후보들로).

- 좀 더 높은 복잡도를 가진 후보는 더 작은 클러스터로 그룹지어진다(좀 더 적은 후보들로).

- 유사한 후보들(논리의 관점에서)의 동일한 클러스터로의 합성이 선호되어 논리 상호최소화 기회를 좀 더 이용하여야 한다.

실제로 효율적으로 작용하는 것으로 알려진 클러스터를 선택하는 기준은 커스텀 명령을 커스텀 명령의 결과에 영향을 미치는 오퍼랜드의 입력 비트의 유사성에 의존하여 클러스터를 선택하는 것이다. 주어진 커스텀 명령에 대하여 입력 오퍼랜드의 어느 비트가 결과에 영향을 미치며, 어느 것이 결과에 영향을 미치지 않는지를 결정하는 것은 간단하다. 커스텀 명령의 상이점은 비공유 입력 비트의 수에 의해서 측정될 수 있다. 따라서 커스텀 명령의 클러스터는 각 명령의 어느 입력 비트가 결과에 영향을 미치는지를 계산하며(결과에 영향을 미치는 이들 비트는 "관계 비트(relavant bit)"라고 불림), 비공유 관계 비트의 수에 의해서 명령들 사이의 상이점의 정도를 계산하며, 설정된 상이점의 양보다 적은 상이점을 갖는 커스텀 명령의 클러스터를 선택함으로써 바람직하게 선택된다.

규칙적이며 예측가능한 CPLD 코어(core)의 타이밍 모델은 클러스터의 형성을 선호한다. 다수의 후보의 하나의 구조로의 그룹화는 특유의 FPGA 구조내에서 자신의 실시 지연을 상당히 변화시키는데, 이것은 자동 클러스터 형성 알고리즘을 좀 더 어렵게 만드는데 이것은 이 경우에 너무 많은 지연을 발생시키지 않는다는 제한하에 상호최소화가 수행되어야하기 때문이다. CPLD와 함께, 주어진 구성에 좀 더 많은 커스텀 명령을 첨가하는 것은 PAL 또는 PLA로부터 좀 더 많은 승항(product term)(PTs)을 단순히 요구한다. 회로가 코어내에 맞추어지는 동안 지연은 교차점 스위치 및 PLA(Tpd_pla)를 통한 시간 지연에 제한될 것이며, 알고리즘은 클러스트를 형성하는 동안에 제기한 지연 변화를 고려할 필요가 없다.

도 2의 기능 유닛에서, 심지어 상이한 비트가 오퍼랜드내의 매우 상이한 위치에 걸쳐 불규칙적으로 확산(spread out)될 때에도 오퍼랜드의 상이한 비트에 대한 신호들을 하나의 논리 블록(26a-b)에 가져오는 것이 가능하기 때문에 교차점 스위치(24)는 특히 유용하다. 이것은 ALU(16)에 의해서 동일한 효과가 구현되어야만 하는 경우에 상당히 많은 정규 명령을 필요로하는 구성가능 명령을 실시하는 것을 가능케한다.

또한, 교차점 스위치(24)는 명령 선택 비트를 상이한 블록(26a,b)내의 오퍼랜드 데이터 비트와 자유로이 혼합하는 것을 가능케한다. 따라서, 명령 식별과 오퍼랜드 프로세싱을 통합함으로써 하드웨어 자원 이용의 좀 더 나은 상호최소화가 가능하다.

도면의 간단한 설명

본 발명의 많은 측면이 아래의 도면을 이용하여 설명될 것이다.

도 1은 구성가능 명령을 지원하는 프로세서 구조,

도 2는 구성가능 기능 유닛,

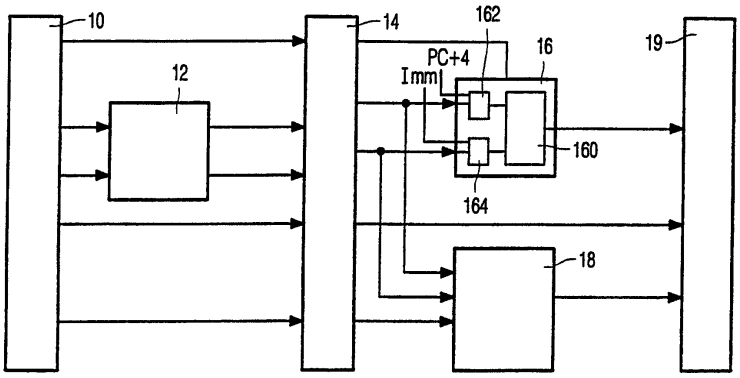
도 3은 구성가능 논리 블록,

도 4는 컴파일된 프로그램을 생성하는 흐름도,

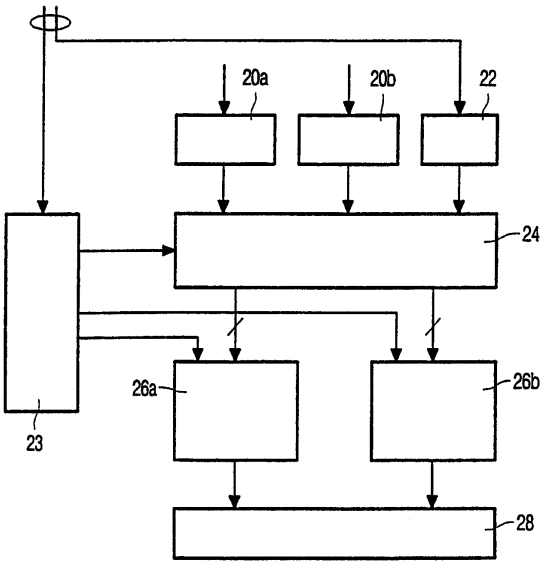
도 5는 명령의 조합을 실행하는 기능 유닛의 모델(model).

도면

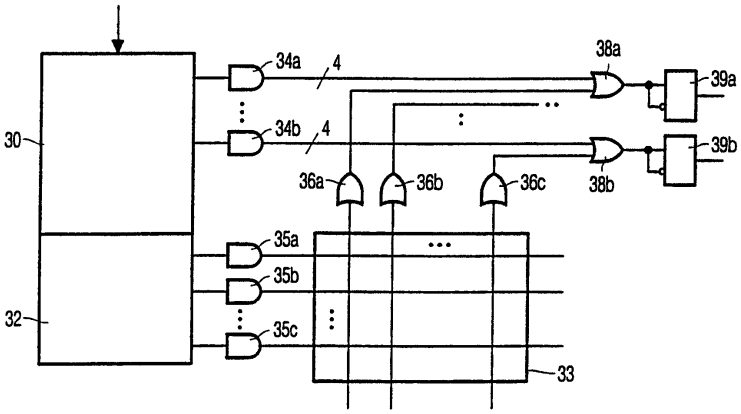
도면1



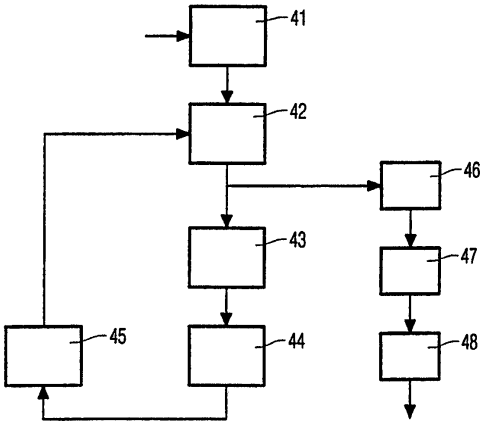
도면2



도면3



도면4



도면5