



(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2007-0105415
(43) 공개일자 2007년10월31일

(51) Int. Cl.

G06F 7/50 (2006.01) G06F 7/52 (2006.01)

G06F 7/00 (2006.01)

(21) 출원번호 10-2006-0037460

(22) 출원일자 2006년04월26일

심사청구일자 2006년04월26일

(71) 출원인

고려대학교 산학협력단

서울 성북구 안암동5가1 고려대학교 내

(72) 발명자

임종인

서울 종로구 내수동 95 삼성파크팰리스 101-901

장남수

서울 마포구 성산2동 200-84

(뒷면에 계속)

(74) 대리인

현중철

전체 청구항 수 : 총 9 항

(54) 부호화를 이용하는 곱셈 장치 및 그 방법

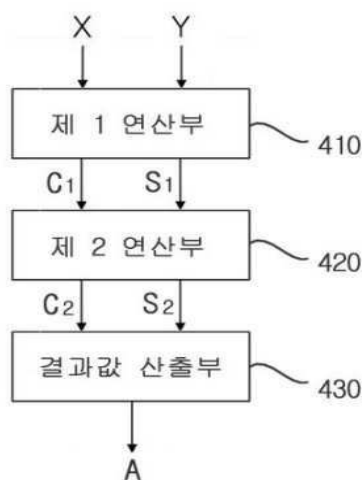
(57) 요약

부호화를 이용하는 덧셈 장치, 부호화를 이용하는 곱셈 장치 및 그 방법이 개시된다.

본 발명은 임의의 자리 비트인 i 번째 비트가 부호를 나타내는 부호비트와 크기를 나타내는 크기비트로 구성되는 부호수인 제1 부호수 및 제2 부호수에 대하여, 상기 제1 부호수의 i 번째 비트와 상기 제2 부호수의 i 번째 비트를 합산하여 음의수인 제1 합의 i 번째 비트와 부호수인 제1 캐리의 $i+1$ 번째 비트를 생성하는 제1연산부, 상기 제1 합의 i 번째 비트를 부호를 갖지 않는 이진수로 변환한 제2 합의 i 번째 비트를 생성하고, 상기 제2 합의 i 번째 비트에 따라 상기 제1 캐리의 $i+1$ 번째 비트를 변환시켜 부호수인 제2 캐리의 $i+1$ 번째 비트를 생성하는 제2연산부 및 상기 제2 캐리의 i 번째 비트와 상기 제2 합의 i 번째 비트를 합산하여 덧셈 결과값을 생성하는 결과값 산출부를 포함한다.

본 발명에 의하면, 공간 복잡도와 시간 복잡도를 줄여 성능을 향상시키고, 제조 비용을 감소시킬 수 있는 효과가 있다. 또한, 또한, 부호수의 특징을 이용하여 하이 래디스(High-radix) 연산을 적용 할 경우, 2의 보수 연산에서 고려해야 하는 캐리 전파를 고려하지 않아도 되는 추가적인 성능 향상을 가능하게 한다.

대표도 - 도4a



(72) 발명자

김성경

서울 성북구 안암동5가 11-5번지 리베르타 302호

지성연

경기 수원시 영통구 매탄2동 111-97

임대성

경기 안산시 상록구 이동 530-14 301호

이상진

서울 노원구 중계1동 롯데우성아파트 103-1202

특허청구의 범위

청구항 1

임의의 자리 비트인 i 번째 비트가 부호를 나타내는 부호비트와 크기를 나타내는 크기비트로 구성되는 부호수인 제1 부호수 및 제2 부호수에 대하여,

상기 제1 부호수의 i 번째 비트와 상기 제2 부호수의 i 번째 비트를 합산하여 음의수인 제1 합의 i 번째 비트와 부호수인 제1 캐리의 $i+1$ 번째 비트를 생성하는 제1연산부;

상기 제1 합의 i 번째 비트를 부호를 갖지 않는 이진수로 변환한 제2 합의 i 번째 비트를 생성하고, 상기 제2 합의 i 번째 비트에 따라 상기 제1 캐리의 $i+1$ 번째 비트를 변환시켜 부호수인 제2 캐리의 $i+1$ 번째 비트를 생성하는 제2연산부; 및

상기 제2 캐리의 i 번째 비트와 상기 제2 합의 i 번째 비트를 합산하여 덧셈 결과값을 생성하는 결과값 산출부를 포함하는 부호화를 이용하는 덧셈 장치.

청구항 2

제 1 항에 있어서,

상기 제1연산부는

상기 제1 부호수의 크기비트 및 제2 부호수의 크기비트의 배타적 논리합 연산을 수행하여 상기 제1 합의 i 번째 크기비트 및 부호비트를 생성하는 제1게이트부를 포함하는 것을 특징으로 하는 부호화를 이용하는 덧셈 장치.

청구항 3

제 1 항에 있어서,

상기 제1연산부는

상기 제1 부호수의 부호비트 및 제2 부호수의 부호비트의 논리곱을 연산하여 상기 제1 캐리의 $i+1$ 번째 부호비트를 생성하는 제2게이트부; 및

상기 제1 부호수의 부호비트 및 제2 부호수의 부호비트의 배타적 논리합의 인버스 연산 결과와 상기 제1 부호수의 크기비트 및 제2 부호수의 크기비트의 논리합 연산 결과 사이의 논리곱 연산을 수행하여 상기 제1 캐리의 $i+1$ 번째 크기비트를 생성하는 제3게이트부를 포함하는 것을 특징으로 하는 부호화를 이용하는 덧셈 장치.

청구항 4

제 1 항에 있어서,

상기 제2연산부는

상기 제2 합의 i 번째 부호비트에 비트값 0을 할당하고, 상기 제1 캐리의 크기비트 및 제1 합의 크기비트의 배타적 논리합을 연산하여 상기 제2 합의 i 번째 크기비트를 생성하는 제4게이트부; 및

상기 제1 캐리의 부호비트 및 제1 캐리의 크기비트의 배타적 논리합의 인버스 연산 결과와 상기 제1 캐리의 부호비트 및 제1 합의 부호비트의 논리합 연산 결과 사이의 논리곱 연산을 수행하여 상기 제2 캐리의 $i+1$ 번째 비트를 생성하는 제5게이트부를 포함하는 것을 특징으로 하는 부호화를 이용하는 덧셈 장치.

청구항 5

임의의 자리 비트인 i 번째 비트가 부호를 나타내는 부호비트와 크기를 나타내는 크기비트로 구성되는 부호수인 제1 부호수 및 제2 부호수에 대하여,

상기 제1 부호수의 i 번째 비트와 상기 제2 부호수의 i 번째 비트를 합산하여 음의수인 제1 합의 i 번째 비트와 부호수인 제1 캐리의 $i+1$ 번째 비트를 생성하는 제1연산부;

상기 제1 합을 i 번째 비트를 부호를 갖지 않는 이진수로 변환한 제2 합의 i 번째 비트를 생성하고, 상기 제2 합의 i 번째 비트에 따라 상기 제1 캐리의 $i+1$ 번째 비트를 변환시켜 부호수인 제2 캐리의 $i+1$ 번째 비트를 생성하는

제2연산부;

상기 제2 캐리의 i번째 비트와 상기 제2 합의 i번째 비트를 합산하여 덧셈 결과값을 생성하는 결과값 산출부;

제1입력값에 따라, 제2입력값, 상기 제1입력값 및 제2입력값을 이용한 모듈러값, 상기 모듈러값과 제2입력값을 합산한 값 또는 0 중 어느 하나를 선택하여 상기 제1연산부의 제2부호수로 출력하는 멀티플렉서;

상기 제1입력값에 따라 상기 멀티플렉서를 제어하여 제2입력값의 덧셈 여부를 결정하는 변환기;

상기 결과값 산출부의 덧셈 결과값을 쉬프트 연산한 쉬프트값을 상기 제1연산부의 제1부호수로 출력하는 쉬프트 레지스터; 및

상기 쉬프트값을 이용하여 제1입력값 및 제2입력값의 곱셈 연산 결과를 생성하는 입출력 버퍼부를 포함하는 부호화를 이용하는 곱셈 장치.

청구항 6

임의의 자리 비트인 i번째 비트가 부호를 나타내는 부호비트와 크기를 나타내는 크기비트로 구성되는 부호수인 제1 부호수 및 제2 부호수에 대하여,

상기 제1 부호수의 i번째 비트와 상기 제2 부호수의 i번째 비트를 합산하여 음의수인 제1 합의 i번째 비트와 부호수인 제1 캐리의 i+1번째 비트를 생성하는 단계;

상기 제1 합의 i번째 비트를 부호를 갖지 않는 이진수로 변환한 제2 합의 i번째 비트를 생성하고, 상기 제2 합의 i번째 비트에 따라 상기 제1 캐리의 i+1번째 비트를 변환시켜 부호수인 제2 캐리의 i+1번째 비트를 생성하는 단계; 및

상기 제2 캐리의 i번째 비트와 상기 제2 합의 i번째 비트를 합산하여 덧셈 결과값을 생성하는 단계를 포함하는 부호화를 이용하는 덧셈 방법.

청구항 7

제 6 항에 있어서,

상기 제1 캐리의 i+1번째 비트를 생성하는 단계는

상기 제1 부호수의 크기비트 및 제2 부호수의 크기비트의 배타적 논리합 연산을 수행하여 상기 제1 합의 i번째 크기비트 및 부호비트를 생성하는 단계;

상기 제1 부호수의 부호비트 및 제2 부호수의 부호비트의 논리곱을 연산하여 상기 제1 캐리의 i+1번째 부호비트를 생성하는 단계; 및

상기 제1 부호수의 부호비트 및 제2 부호수의 부호비트의 배타적 논리합의 인버스 연산 결과와 상기 제1 부호수의 크기비트 및 제2 부호수의 크기비트의 논리합 연산 결과 사이의 논리곱 연산을 수행하여 상기 제1 캐리의 i+1번째 크기비트를 생성하는 단계를 포함하는 것을 특징으로 하는 부호화를 이용하는 덧셈 방법.

청구항 8

제 6 항에 있어서,

상기 제2 캐리의 i+1번째 비트를 생성하는 단계는

상기 제2 합의 i번째 부호비트에 비트값 0을 할당하고, 상기 제1 캐리의 크기비트 및 제1 합의 크기비트의 배타적 논리합을 연산하여 상기 제2 합의 i번째 크기비트를 생성하는 단계; 및

상기 제1 캐리의 부호비트 및 제1 캐리의 크기비트의 배타적 논리합의 인버스 연산 결과와 상기 제1 캐리의 부호비트 및 제1 합의 부호비트의 논리합 연산 결과 사이의 논리곱 연산을 수행하여 상기 제2 캐리의 i+1번째 비트를 생성하는 단계를 포함하는 것을 특징으로 하는 부호화를 이용하는 덧셈 방법.

청구항 9

임의의 자리 비트인 i번째 비트가 부호를 나타내는 부호비트와 크기를 나타내는 크기비트로 구성되는 부호수에 대하여,

제1입력값에 따라, 제2입력값, 상기 제1입력값 및 제2입력값을 이용한 모듈러값, 상기 모듈러값과 제2입력값을 합산한 값 또는 0 중 어느 하나를 선택하여 제2부호수로 출력하는 단계;

상기 제1 부호수의 i번째 비트와 상기 제2 부호수의 i번째 비트를 합산하여 음의수인 제1 합의 i번째 비트와 부호수인 제1 캐리의 i+1번째 비트를 생성하는 단계;

상기 제1 합의 i번째 비트를 부호를 갖지 않는 이진수로 변환한 제2 합의 i번째 비트를 생성하고, 상기 제2 합의 i번째 비트에 따라 상기 제1 캐리의 i+1번째 비트를 변환시켜 부호수인 제2 캐리의 i+1번째 비트를 생성하는 단계; 및

상기 제2 캐리의 i번째 비트와 상기 제2 합의 i번째 비트를 합산하여 덧셈 결과값을 생성하는 단계를 반복하고, 상기 제1입력값에 따라 제어하여 제2입력값의 덧셈 여부를 결정하고,

상기 덧셈 결과값을 쉬프트 연산한 쉬프트값을 제1부호수로 출력하여 제1입력값 및 제2입력값의 곱셈 연산 결과를 생성하는 단계를 포함하는 부호화를 이용하는 곱셈 방법.

명세서

발명의 상세한 설명

발명의 목적

발명이 속하는 기술 및 그 분야의 종래기술

- <14> 본 발명은 연산 장치에 관한 것으로, 특히, 부호화를 이용하는 덧셈 장치, 부호화를 이용하는 곱셈 장치 및 그 방법에 관한 것이다.
- <15> RSA(Rivest-Shamir-Adleman) 암호 시스템은 키 분배와 관리의 용이, 인증과 디지털 서명(digital signature)이 가능한 장점으로 인해 널리 이용되고 있다. RSA 암호 시스템은 IC카드, 모바일 및 WPKI, 전자화폐, SET, SSL 시스템 등에 많이 이용된다. RSA 공개키 암호 시스템은 매우 큰 정수에 대한 소인수분해가 어렵고 해를 찾아내는 데 많은 시간을 요구한다는 점을 이용한 암호시스템이다. 그러나 최근 연산 환경의 발전 및 알고리즘의 효율성 개선 등으로 인수분해가 가능한 정수의 크기가 점차 커지고 있고 앞으로도 증가할 것이다. 따라서 RSA 암호 시스템의 안전성을 위해서 키 길이는 길어지게 되고 상대적으로 연산량이 증가하게 된다. 그러므로 RSA 암호 시스템의 효율적인 구성은 매우 중요한 연구 과제이다.
- <16> RSA 암호 시스템은 모듈러 지수승 연산을 통하여 수행이 되고, 모듈러 지수승 연산은 반복적인 모듈러 곱셈연산을 통하여 구성이 된다. 따라서 모듈러 곱셈 장치의 효율적인 구현과 반복 연산의 횟수는 RSA 암호 시스템의 효율성에 있어 중요한 문제이다.
- <17> 최근 RSA 암호시스템에서 이용되어지는 모듈러 연산을 위해 개발 된 방법 중에 몽고메리(Montgomery) 모듈러 곱셈 알고리즘이 효율적이라고 알려져 있다. 몽고메리(Montgomery) 곱셈 장치는 세 입력 값의 덧셈 연산이 필요하고, 덧셈 연산의 지연시간이 곱셈 장치의 시간 복잡도를 결정한다.
- <18> 몽고메리(Montgomery) 모듈러 곱셈 알고리즘의 시간 복잡도는 세 입력 값의 덧셈 과정에서 발생하는 지연시간에 의존하게 되며, 덧셈 연산을 위하여 구성되는 대표적인 덧셈 장치는 캐리 전파 가산기(Carry Propagation Adder; 캐리 전파 가산기(CPA))와 캐리 저장 가산기(Carry Save Adder; CSA)가 있다. 캐리 저장 가산기(CSA)를 이용하여 구성할 경우 캐리 전파 가산기(CPA)의 캐리 전파 지연 시간이 발생하지 않으므로 캐리 전파 가산기(CPA)보다 효율적인 곱셈 장치를 구성할 수 있다. 그러나, 캐리 저장 가산기(CSA)를 이용한 결과 캐리와 합 즉, 두개의 값으로 표현 되므로, 연산의 입력 값의 개수가 증가한다.
- <19> 종래의 캐리 저장 가산기(CSA)를 이용한 몽고메리(Montgomery) 곱셈 장치는 다음과 같다.
- <20> 몽고메리(Montgomery) 곱셈 알고리즘에서, 세 연산자의 덧셈과정에서 발생하는 캐리 전파를 처리하는 과정이 알고리즘 지연 시간의 대부분을 차지하므로, 알고리즘에서 세 입력 값에 대한 덧셈을 수행하는 과정이 알고리즘의 시간 복잡도에서 가장 큰 부분을 차지한다.

<21> 캐리 저장 가산기(CSA)는 세 개의 입력 값을 갖고, 연산($x_i + y_i + z_i = 2c_{i+1} + s_i$)을 수행하고 두 개의 값(캐리인 c_{i+1} , 합인 s_i)을 출력하는 덧셈 장치이다.

<22> 도 1a 및 도 1b는 종래의 곱셈 장치의 블록도이다.

<23> 도 1a는 캐리 저장 가산기(CSA) 덧셈 장치를 이용한 5개의 입력 값과 2개의 출력 값을 가지는 곱셈 장치(종래의(McIvor의 1) 방법)를 도시하고, 도 1b는 4개의 입력 값과 2개의 출력 값을 가지는 곱셈 장치(종래의(McIvor의 2) 방법)를 도시한다.

<24> 몽고메리(Montgomery) 곱셈 알고리즘에서 $(A + x_i Y + u_i M)$ 의 연산 결과는 캐리 저장 가산기(CSA)를 이용하여 합 A_S 과 캐리 A_C 로 나타낼 수 있고, $A = (A_S + A_C)$ 는 다음 곱셈의 입력으로 이용되므로 Y 또한 $Y = (Y_S + Y_C)$ 으로 표현한다. 따라서 $(A + x_i Y + u_i M)$ 의 연산은 x_i 를 X 의 i 번째 비트라 할 때 $((A_S + A_C) + x_i(Y_S + Y_C) + u_i M)$ 으로 나타낼 수 있으므로 다섯 개의 입력 값과 두 개의 출력 값을 갖는 종래의(McIvor의 1) 방법이 된다. 종래의(McIvor의 1) 방법은 도 1a과 같이 3단계의 캐리 저장 가산기(CSA) 지연 시간을 요구한다.

<25> 이와 같은 지연 시간을 줄이기 위한 종래의(McIvor의 2) 방법은 도 1b와 같은 2단계의 캐리 저장 가산기(CSA)로 구성된다. $((A_S + A_C) + x_i(Y_S + Y_C) + u_i M)$ 의 연산에서 x_i 와 u_i 의 값에 따라 덧셈에 필요한 피 연산자의 조합은 달라진다. $x_i = u_i = 0$ 인 경우 단지 $(A_S + A_C)$ 의 연산이 되고, $x_i = 1, u_i = 0$ 인 경우 $(A_S + A_C) + (Y_S + Y_C)$ 의 연산이 되며, $x_i = 0, u_i = 1$ 의 경우 $(A_S + A_C) + M$ 의 연산이 된다. 그러나, $x_i = y_i = 1$ 의 경우 다섯 개의 피 연산자로 구성된 $(A_S + A_C) + (Y_S + Y_C) + M$ 의 연산을 수행하게 된다. 피 연산자의 항의 개수가 증가하는 것을 방지하기 위하여 곱셈이 시작하기 전 사전 연산을 통하여 $(Y_S + Y_C) + M$ 의 연산을 수행하고 합 D_S 과 캐리 D_C 을 저장한 후 멀티플렉서(MUX)를 이용하여 x_i 와 u_i 의 모든 경우에 따라서 $(A_S + A_C) + (Y_C + Y_S)$, $(A_S + A_C) + (0 + 0)$, $(A_S + A_C) + (0 + M)$, $(A_S + A_C) + (D_C + D_S)$ 와 같이 연산한다.

<26> 그러나, 종래의 덧셈 장치 및 곱셈 장치는 캐리 저장 가산기(CSA)를 2단계 이상 요구하고, 1단계의 캐리 저장 가산기(CSA)마다 많은 면적과 게이트 지연시간이 필요하므로, 덧셈 장치 및 곱셈 장치의 성능을 저하시키고, 그 제조 비용을 증가시키는 문제점이 있다.

발명이 이루고자 하는 기술적 과제

<27> 따라서, 본 발명이 이루고자 하는 첫번째 기술적 과제는 공간 복잡도와 시간 복잡도를 줄여 성능을 향상시키고, 제조 비용을 감소시킬 수 있는 부호화를 이용하는 덧셈 장치를 제공하는데 있다.

<28> 본 발명이 이루고자 하는 두번째 기술적 과제는 상기의 덧셈 장치를 이용하여 덧셈 연산을 수행하는 부호화를

이용하는 곱셈 장치를 제공하는데 있다.

<29> 본 발명이 이루고자 하는 세번째 기술적 과제는 상기의 덧셈 장치에 적용되는 부호화를 이용하는 덧셈 방법을 제공하는데 있다.

<30> 본 발명이 이루고자 하는 세번째 기술적 과제는 상기의 곱셈 장치에 적용되는 부호화를 이용하는 곱셈 방법을 제공하는데 있다.

발명의 구성 및 작용

<31> 상기의 첫번째 기술적 과제를 이루기 위하여, 본 발명은 임의의 자리 비트인 i 번째 비트가 부호를 나타내는 부호비트와 크기를 나타내는 크기비트로 구성되는 부호수인 제1 부호수 및 제2 부호수에 대하여, 상기 제1 부호수의 i 번째 비트와 상기 제2 부호수의 i 번째 비트를 합산하여 음의수인 제1 합의 i 번째 비트와 부호수인 제1 캐리의 $i+1$ 번째 비트를 생성하는 제1연산부, 상기 제1 합의 i 번째 비트를 부호를 갖지 않는 이진수로 변환한 제2 합의 i 번째 비트를 생성하고, 상기 제2 합 of i 번째 비트에 따라 상기 제1 캐리의 $i+1$ 번째 비트를 변환시켜 부호수인 제2 캐리의 $i+1$ 번째 비트를 생성하는 제2연산부 및 상기 제2 캐리의 i 번째 비트와 상기 제2 합 of i 번째 비트를 합산하여 덧셈 결과값을 생성하는 결과값 산출부를 포함하는 부호화를 이용하는 덧셈 장치를 제공한다.

<32> 상기의 두번째 기술적 과제를 이루기 위하여, 본 발명은 상기 제1 부호수의 i 번째 비트와 상기 제2 부호수의 i 번째 비트를 합산하여 음의수인 제1 합 of i 번째 비트와 부호수인 제1 캐리의 $i+1$ 번째 비트를 생성하는 제1연산부, 상기 제1 합 of i 번째 비트를 부호를 갖지 않는 이진수로 변환한 제2 합 of i 번째 비트를 생성하고, 상기 제2 합 of i 번째 비트에 따라 상기 제1 캐리의 $i+1$ 번째 비트를 변환시켜 부호수인 제2 캐리의 $i+1$ 번째 비트를 생성하는 제2연산부, 상기 제2 캐리의 i 번째 비트와 상기 제2 합 of i 번째 비트를 합산하여 덧셈 결과값을 생성하는 결과값 산출부, 제1입력값에 따라, 제2입력값, 상기 제1입력값 및 제2입력값을 이용한 모듈러값, 상기 모듈러값과 제2입력값을 합산한 값 또는 0 중 어느 하나를 선택하여 상기 제1연산부의 제2부호수로 출력하는 멀티플렉서, 상기 제1입력값에 따라 상기 멀티플렉서를 제어하여 제2입력값의 덧셈 여부를 결정하는 변환기, 상기 결과값 산출부의 덧셈 결과값을 쉬프트 연산한 쉬프트값을 상기 제1연산부의 제1부호수로 출력하는 쉬프트 레지스터 및 상기 쉬프트값을 이용하여 제1입력값 및 제2입력값의 곱셈 연산 결과를 생성하는 임출력 버퍼부를 포함하는 부호화를 이용하는 곱셈 장치를 제공한다.

<33> 상기의 세번째 기술적 과제를 이루기 위하여, 본 발명은 상기 제1 부호수의 i 번째 비트와 상기 제2 부호수의 i 번째 비트를 합산하여 음의수인 제1 합 of i 번째 비트와 부호수인 제1 캐리의 $i+1$ 번째 비트를 생성하는 단계, 상기 제1 합 of i 번째 비트를 부호를 갖지 않는 이진수로 변환한 제2 합 of i 번째 비트를 생성하고, 상기 제2 합 of i 번째 비트에 따라 상기 제1 캐리의 $i+1$ 번째 비트를 변환시켜 부호수인 제2 캐리의 $i+1$ 번째 비트를 생성하는 단계 및 상기 제2 캐리의 i 번째 비트와 상기 제2 합 of i 번째 비트를 합산하여 덧셈 결과값을 생성하는 단계를 포함하는 부호화를 이용하는 덧셈 방법을 제공한다.

<34> 상기의 네번째 기술적 과제를 이루기 위하여, 본 발명은 제1입력값에 따라, 제2입력값, 상기 제1입력값 및 제2입력값을 이용한 모듈러값, 상기 모듈러값과 제2입력값을 합산한 값 또는 0 중 어느 하나를 선택하여 제2부호수로 출력하는 단계, 상기 제1 부호수의 i 번째 비트와 상기 제2 부호수의 i 번째 비트를 합산하여 음의수인 제1 합 of i 번째 비트와 부호수인 제1 캐리의 $i+1$ 번째 비트를 생성하는 단계, 상기 제1 합 of i 번째 비트를 부호를 갖지 않는 이진수로 변환한 제2 합 of i 번째 비트를 생성하고, 상기 제2 합 of i 번째 비트에 따라 상기 제1 캐리의 $i+1$ 번째 비트를 변환시켜 부호수인 제2 캐리의 $i+1$ 번째 비트를 생성하는 단계 및 상기 제2 캐리의 i 번째 비트와 상기 제2 합 of i 번째 비트를 합산하여 덧셈 결과값을 생성하는 단계를 반복하고, 상기 제1입력값에 따라 제어하여 제2입력값의 덧셈 여부를 결정하고, 상기 덧셈 결과값을 쉬프트 연산한 쉬프트값을 제1부호수로 출력하여 제1입력값 및 제2입력값의 곱셈 연산 결과를 생성하는 단계를 포함하는 부호화를 이용하는 곱셈 방법을 제공한다.

<35> 본 발명은 부호화 기법을 이용하여 종래의(McIvor의 2) 방법에 비해 게이트 수와 지연시간을 줄인 덧셈 장치 즉, 덧셈 장치(Nadder)를 제공하고, 본 발명에 따른 덧셈 장치(Nadder)를 이용하여 구성된 새로운 곱셈 장치를 제공한다. 또한, 본 발명은 본 발명에 따른 덧셈 장치(Nadder) 및 본 발명에 따른 곱셈 장치에 적용되는 방법을 제공한다.

<36> 이하에서는 도면을 참조하여 본 발명의 바람직한 실시예를 설명하기로 한다.

<37> 도 2는 본 발명에 따른 덧셈 장치의 블록도이다.

<38> 본 발명에 따른 덧셈 장치(Nadder)는 도 2와 같이 두 개의 입력 값과 하나의 출력 값을 가지는 덧셈 장치이다.

몽고메리 곱셈에서 $(A+x_iY+u_iM)$ 의 연산은 x_iY+u_iM 을 한 개의 값으로 표현하면 두개의 피 연산자 덧셈으로 표현이 가능하다. 그러나, 양수인 두 값에 대해 캐리 전과 없이 덧셈을 수행하는 것은 불가능하므로, 두 수를 캐리 전과 없이 한 값으로 표현하는 부호화 기법을 이용한다.

<39> 따라서, 본 발명에 따른 덧셈 장치(Nadder)의 두 입력은 부호수가 되고, 출력 역시 부호수가 된다.

<40> 부호화 기법을 이용하기 위해서 n 비트 X와 Y의 덧셈 결과를 (n+1)비트 캐리 C와 n비트 합 S로 나타낸다. C와 S는 각각 $C=(c_n c_{n-1} \dots c_1 c_0)$ 와 $S=(s_{n-1} s_{n-2} \dots s_1 s_0)$ 라고 가정한다.

<41> 두 비트 x_i 와 y_i 의 덧셈 결과를 캐리 c_{i+1} 와 합 s_i 으로 표현하면 수학적 식 1과 같이 표현된다.

수학적 식 1

<42> $x_i + y_i = c_{i+1} \cdot 2 + s_i = (c_{i+1}, s_i)$, $0 \leq i < n-1$

<43> 도 3a 및 도 3b는 도 2에 적용되는 부호화 기법을 표로 도시한 것이다.

<44> 도 3a는 부호화 기법을 이용한 두 부호수 x_i, y_i 의 덧셈에 대한 표이고, 도 3b는 부호화 기법을 이용한 부호수 $c1_i$ 와 음의수 $s1_i$ 의 덧셈에 대한 표이다. 이때, 제1 캐리인 C1, 제1 합인 S1, 제2 캐리인 C2, 제2 합인 S2는 중간 과정에서 제거된다.

<45> 부호수를 표현하기 위해 x_i, y_i 를 부호자리와 값의 자리를 이용하여 각각 $[x_i^1, x_i^0]$ 와 $[y_i^1, y_i^0]$ 와 같이 표현한다(즉, $1=[0,1]$, $0=[0,0]$, $\overline{1}=[1,1]$). 연산 과정의 $c1_i, s1_i, c2_i, s2_i$ 와 출력 결과인 a_i 또한 부호가 있는 형태로 표현한다.

<46> 본 발명에 이용되는 부호화 기법부호화 기법을 설명하기 위해 0과 $\overline{1}$ 로 이루어진 수를 음의수라고 가정한다. $x_i + y_i = 1$ 인 경우 $x_i + y_i = (0,1)$ 이며, 부호수를 이용하여 $(0,1) = (1, \overline{1})$ 로 표현할 수 있다. 두

양수 X, Y 덧셈의 $x_i + y_i$ 에서 s_i 가 1인 경우는 $(x_i, y_i) = (1,0)$, $(x_i, y_i) = (0,1)$ 두 가지가 있고,

이 때 합은 $x_i + y_i = (0,1)$ 이 되고, c_{i+1} 는 항상 0이므로 부호수로 표기하는 경우, 캐리 전과 없이

s_i 를 음수로 표현할 수 있다.

<47> 따라서 $X+Y=C+S$ 이므로 S를 부호수로 만들면 C+S단계에서 캐리 전과 없이 부호수로 표현된다. 본 발명에 따른 곱셈 장치에서 초기 Y와 M은 양수이지만 Y+M은 부호수가 되므로 곱셈 장치의 모든 입출력 값을 부호수로 고려한다. 종래의(McIvor의 2) 방법에서 A는 캐리를 포함하는 (A_C, A_S) 의 형태인 반면, 본 발명의 A는 두 부

호수의 덧셈 결과로써 한 개의 부호수 A로 표현된다.

- <48> 두 부호수의 덧셈은 $x_i=y_i=1$ 인 경우와 $x_i=y_i=\overline{1}$ 에서 캐리가 발생할 수 있다. 그러나, 도 3a 및 도 3b의 두 가지 부호화 기법을 이용한 두 부호수의 덧셈은 캐리 전과 없이 하나의 값으로 출력할 수 있다. 본 발명에 따른 방법은 두 부호수의 덧셈을 부호수와 음의수의 덧셈으로 변환하고 그 결과를 부호수와 이진수로 변환시켜 캐리 전과 없이 더하는 방법이다.
- <49> 도 4a는 도 2의 상세 블록도이다.
- <50> 먼저, 임의의 자리 비트인 i번째 비트가 부호를 나타내는 부호비트와 크기를 나타내는 크기비트로 구성되는 부호수인 제1 부호수(X) 및 제2 부호수(Y)를 가정한다.
- <51> 제1연산부(410)는 제1 부호수의 i번째 비트와 제2 부호수의 i번째 비트를 합산하여 음의수인 제1 합의 i번째 비트($s1_i$)와 부호수인 제1 캐리의 i+1번째 비트($c1_{i+1}$)를 생성한다.
- <52> 제2연산부(420)는 제1 합의 i번째 비트($s1_i$)를 부호를 갖지 않는 이진수로 변환한 제2 합의 i번째 비트($s2_i$)를 생성하고, 제2 합의 i번째 비트($s2_i$)에 따라 제1 캐리의 i+1번째 비트($c1_{i+1}$)를 변환시켜 부호수인 제2 캐리의 i+1번째 비트($c2_{i+1}$)를 생성한다.
- <53> 결과값 산출부(430)는 제2 캐리의 i번째 비트($c2_i$)와 제2 합의 i번째 비트($s2_i$)를 합산하여 덧셈 결과값(A)을 생성한다.
- <54> 도 4b는 본 발명의 일 실시예에 따른 덧셈 장치의 회로도이다.
- <55> 이하에서는, $0 \leq i \leq n+1$ 인 i에 대하여, i=0인 경우 즉, 0번째 비트를 중심으로 설명한다.
- <56> 본 발명의 일 실시예에 따르면, 제1연산부(410)는 제1게이트부(411), 제2게이트부(412) 및 제3게이트부(413)를 포함하고, 제2연산부(420)는 제4게이트부(424) 및 제5게이트부(425)를 포함하며, 결과값 산출부(430)는 제6게이트부(436)를 포함한다.
- <57> 제1게이트부(411)는 제1 부호수의 크기비트 및 제2 부호수의 크기비트의 배타적 논리합 연산을 수행하여 제1 합의 i번째 크기비트 및 부호비트를 생성한다. 바람직하게는, 제1게이트부(411) XOR 게이트를 포함할 수 있다.
- <58> 제2게이트부(412)는 제1 부호수의 부호비트 및 제2 부호수의 부호비트의 논리곱을 연산하여 제1 캐리의 i+1번째 부호비트를 생성한다. 바람직하게는, 제2게이트부(412)는 AND 게이트를 포함할 수 있다.
- <59> 제3게이트부(413)는 제1 부호수의 부호비트 및 제2 부호수의 부호비트의 배타적 논리합의 인버스 연산 결과와 제1 부호수의 크기비트 및 제2 부호수의 크기비트의 논리합 연산 결과 사이의 논리곱 연산을 수행하여 제1 캐리의 i+1번째 크기비트를 생성한다. 바람직하게는, 제3게이트부(413)는 OR 게이트, XOR 게이트, NOT 게이트, AND 게이트를 포함할 수 있다.
- <60> 제4게이트부(424)는 제2 합의 i번째 부호비트에 비트값 0을 할당하고, 제1 캐리의 크기비트 및 제1 합의 크기비트의 배타적 논리합을 연산하여 제2 합의 i번째 크기비트를 생성한다. 바람직하게는, 제4게이트부(424)는 XOR 게이트를 포함할 수 있다.
- <61> 제5게이트부(425)는 제1 캐리의 부호비트 및 제1 캐리의 크기비트의 배타적 논리합의 인버스 연산 결과와 제1 캐리의 부호비트 및 제1 합의 부호비트의 논리합 연산 결과 사이의 논리곱 연산을 수행하여 제2 캐리의 i+1번째 비트를 생성한다. 바람직하게는, 제5게이트부(425)는 OR 게이트, XOR 게이트, NOT 게이트, AND 게이트를 포함할 수 있다.
- <62> 제6게이트부(436)는 제2 캐리의 i번째 비트와 제2 합의 i번째 비트를 합산하여 부호비트와 크기비트로 구성된

덧셈 결과값(A)을 생성한다. 바람직하게는, 제6게이트부(436)는 XOR 게이트, NOT 게이트, AND 게이트를 포함할 수 있다.

<63> $1 \leq i \leq n+1$ 인 경우도 상술한 바와 같은 구성을 갖는다. 다만, $i = n+1$ 인 경우, 부호 비트를 입력으로 하지 않는 점, 제1 캐리의 $n+2$ 번째 비트와 제2 캐리의 $n+2$ 번째 사이의 베타적 논리합 연산에 따라 덧셈 결과값(A)의 $n+2$ 번째 크기 비트를 생성하는 점은 나머지 자리 비트와 구별된다.

<64> 도 5a는 본 발명에 따른 곱셈 장치의 블록도이다.

<65> 본 발명에 따른 덧셈 장치(Nadder)를 이용하는 곱셈 장치는 입력 값을 제어하기 위하여 Y+M의 사전 연산이 요구된다. 이 연산은 곱셈 장치의 반복 연산을 수행하기 전에 연산되어 저장되고 x_i, u_i 에 따라 사전 연산 값들을 선택해서 이용해야 하므로 멀티플렉서(MUX, 530)가 필요하다.

<66> 따라서 본 발명에 따른 곱셈 장치는 본 발명에 따른 덧셈 장치(Nadder, 540), 4:1 멀티플렉서(MUX, 530), 1비트 변환기(521)를 포함한다.

<67> 이하에서는, 임의의 자리 비트인 i 번째 비트가 부호를 나타내는 부호비트와 크기를 나타내는 크기비트로 구성되는 부호수인 제1 부호수 및 제2 부호수를 가정한다. 또한, 곱셈 연산의 두 입력값을 제1입력값, 제2입력값으로 정의한다.

<68> 입출력 버퍼부(500)는 제2입력값, 모듈러값, 미리 연산된 제2입력값과 모듈러값의 덧셈값, 0를 각각 Y 레지스터(511), M 레지스터(512), Y+M 레지스터(513), 0 레지스터(514)에 전송하고, 쉬프트 레지스터(550)에 의한 쉬프트값을 이용하여 제1입력값 및 제2입력값의 곱셈 연산 결과를 생성하여 출력한다.

<69> Y 레지스터(511), M 레지스터(512), Y+M 레지스터(513), 0 레지스터(514)는 각각, 제2입력값, 모듈러값, 미리 연산된 제2입력값과 모듈러값의 덧셈값, 0를 저장한다.

<70> 변환기(521) 및 컨트롤러(522)는 제1입력값에 따라 멀티플렉서(530)를 제어한다. 이에 따라, 제2입력값의 덧셈 여부가 결정된다.

<71> 멀티플렉서(530)는 제1입력값에 따라, 제2입력값, 상기 제1입력값 및 제2입력값을 이용한 모듈러값, 상기 모듈러값과 제2입력값을 합산한 값 또는 0 중 어느 하나를 선택하여 본 발명에 따른 덧셈 장치(Nadder, 540)에 제2 부호수로 입력한다.

<72> 본 발명에 따른 덧셈 장치(Nadder, 540)는 도 4a의 제1연산부(410), 제2연산부(420) 및 결과값 산출부(430)를 포함한다.

<73> 쉬프트 레지스터(550)는 본 발명에 따른 덧셈 장치(Nadder, 540)의 덧셈 결과값을 쉬프트 연산한 쉬프트값을 본 발명에 따른 덧셈 장치(Nadder, 540)에 제1부호수로 다시 입력한다.

<74> 곱셈이 시작되기 전 한 번의 연산으로 Y+M을 연산하고 레지스터에 저장한다. 두 $(n+1)$ 비트의 덧셈이므로 캐리

전파를 고려하면 $(n+2)$ 비트까지 확장된다. x_i 는 부호수이므로 변환기(521)를 이용해서 한 비트씩 이진수로 표

현하고, 이진수 x_i, u_i 에 따라 멀티플렉서(MUX, 530)를 이용하여 피 연산자 $\{Y, Y+M, M, 0\}$ 중에서 어느 하나를 선택한다.

<75> 중간 결과 A와 멀티플렉서(MUX, 530)를 통해 x_i, u_i 에 따라 $\{Y, Y+M, M, 0\}$ 에서 선택된 피 연산자가 본 발명에 따른 덧셈 장치(Nadder)에 입력된다. Y+M은 최대 $(n+2)$ 비트의 크기를 갖고 A는 최대 $(n+3)$ 비트까지 확장되어 쉬프트 연산을 수행하면 $(n+2)$ 비트가 된다. 또한, 변형 몽고메리(Montgomery) 곱셈 알고리즘을 이용한 본 발명에 따른 곱셈 장치에서 출력 결과 A는 $(n+1)$ 비트가 된다.

<76> 본 발명에 따른 곱셈 장치에서, 두 $(n+1)$ 비트 피 연산자의 덧셈은 곱셈을 수행하는 동안 $(n+3)$ 비트까지 연산이 확장되며 곱셈 수행이 종료되어 출력되는 결과는 $2M$ 보다 작은 $(n+1)$ 비트 값이 된다.

<77> 본 발명에 따른 곱셈 장치에서 입력의 최상위 두 비트는 [1, 1]이 되는 경우가 나타나지 않는다. 따라서 최상위 두 비트가 [1, 1]이 아닌 경우, 즉 [1, 0], [1, $\bar{1}$]은 연산 2단계에서 $c2_{n+2} = \bar{1}$, $s2_{n+2} = 1$ 가 되며

로 XOR 연산을 통해서 0이 된다. 따라서 A_{n+1} 은 (n+2)비트가 되고 쉬프트 연산을 통해 (n+1)비트를 유지할 수 있다.

<78> 도 5b는 도 5a의 변환기(521)에 대한 상세 블록도이다.

<79> 도 5b는 본 발명에 따른 부호수를 이진수로 바꾸는 변환기를 도시한 것이다. 몽고메리(Montgomery) 곱셈에서 Y의 덧셈 여부를 결정하는 x_i 는 부호수를 이용하므로 $\bar{1}$ 이 될 수 있다. $x_i = \bar{1}$ 는 Y의 뺄셈을 의미한다. 따라서, Y의 음수 값을 구하기 위하여 Y의 2의 보수를 연산해야 하는 비용을 추가로 부담해야 하는 문제가 발생한다. 변환기(521)를 본 발명에 따른 곱셈 장치에 적용 가능하면, 추가적인 연산을 줄이고, 효율적인 구성을 가능하게 할 수 있다.

<80> X는 부호수로 표현되지만 양수이므로 X는 부호수에서 이진수로 표현이 가능하다. 예를 들어, X가 $(1, \bar{1}, 1, \bar{1}, 1, \bar{1})$ 이라고 가정하면, 최하위 비트가 $\bar{1}$ 이므로 $(\bar{1}, 1)$ 과 같이 표현 가능하고, 따라서 곱셈 장치에 이용할 x_0 은 1이 되며, 다음 비트로 넘어가는 캐리는 $\bar{1}$ 이 된다. 두 번째 비트가 $\bar{1}$ 이므로 이전 캐리 $\bar{1}$ 과 덧셈을 수행하면 합, 즉 곱셈 장치에 이용되는 x_1 은 0이 되고 캐리는 $\bar{1}$ 이 된다. 이와 같은 방법으로 곱셈 장치가 반복 연산하는 동안 한 비트씩 연산되어 Y의 덧셈 여부를 결정하게 된다.

<81> 도 6a는 본 발명에 따른 덧셈 방법의 흐름도이다.

<82> 먼저, 임의의 자리 비트인 i번째 비트가 부호를 나타내는 부호비트와 크기를 나타내는 크기비트로 구성되는 부호수인 제1 부호수 및 제2 부호수를 가정한다.

<83> 다음, 제1 부호수의 i번째 비트와 제2 부호수의 i번째 비트를 합산하여 음의수인 제1 합의 i번째 비트와 부호수인 제1 캐리의 i+1번째 비트를 생성한다(610 과정).

<84> 제1 합의 i번째 비트와 제1 캐리의 i+1번째 비트가 생성되면, 제1 합의 i번째 비트를 부호를 갖지 않는 이진수로 변환한 제2 합의 i번째 비트를 생성하고, 제2 합의 i번째 비트에 따라 제1 캐리의 i+1번째 비트를 변환시켜 부호수인 제2 캐리의 i+1번째 비트를 생성한다(620 과정).

<85> 마지막으로, 제2 캐리의 i번째 비트와 제2 합의 i번째 비트를 합산하여 덧셈 결과값을 생성한다(630 과정).

<86> 도 6b는 도 6a의 상세 알고리즘을 도시한 것이다.

<87> 논리 회로에서 이용하는 기호에 대해 $\vee$ 는 OR 연산, $\wedge$ 는 AND 연산, (+)는 XOR 연산, $\sim$ 는 NOT 연산이라 정의한다.

<88> $[x_i^1, x_i^0]$ 와 $[y_i^1, y_i^0]$ 에 대한 덧셈의 경우의 수는 총 9가지이며 결과는 캐리와 합으로 나타낸다. 논리 회로

는 다음과 같이 구성된다. $c1_{i+1}^1$ 은 $(x_i^1 \text{ AND } y_i^1)$ 으로 생성되고 $c1_{i+1}^0$ 은 $\sim(x_i^1 \text{ XOR } y_i^1) \text{ AND } (x_i^0 \text{ OR } y_i^0)$ 으로

로 생성된다. 합 $[s1_i^1, s1_i^0]$ 은 음의 수 이므로 동일한 값을 갖는다. 따라서 $(x_i^0 \text{ XOR } y_i^0)$ 으로 두 값이 동시

에 생성되고, $c1_{i+1}^0$ 을 제외한 나머지 연산은 게이트 지연 시간 1단계에서 수행되고 $c1_{i+1}^0$ 의 정확한 값

은 게이트 지연 시간 2단계에서 결정된다.

<89> i번째 비트의 연산에서 연산 2단계의 두 입력은 $c1_i$ 와 $s1_i$ 이다. $s1_i$ 는 i번째 비트의 연산 1단계의 결과이

고 $c1_i$ 는 (i-1)번째 비트에서 연산된 값이다. 연산 2단계의 연산의 논리 회로는 다음과 같이 구성된다. 캐리

$c2_{i+1}$ 는 음의 수 이므로 $c2_{i+1}^1$ 와 $c2_{i+1}^0$ 이 $\sim(c1_i^1 XOR c1_i^0) AND (c1_i^1 OR s1_i^1)$ 의 연산을 통해

동일한 값을 갖는다. 합 $s2_i$ 는 이진수이므로 부호 비트는 0이 되고 $s2_i^0 (c1_i^0 XOR s1_i^0)$ 의 연산을 통해 결정된다.

<90> 연산 3단계는 연산 2단계의 출력들에 대한 XOR 연산을 통해 하나의 값으로 표현한다. a_i^1 는

$(\sim s2_i^0 AND c2_i^1)$ 의 연산을 통해서 구할 수 있고 $a_i^0 (c2_i^0 XOR s2_i^0)$ 의 연산을 통해서 구할 수 있다.

<91> 피 연산자 X, Y는 2M 보다 작은 범위에 있고 (n+1)비트의 크기이므로 본 발명에 따른 곱셈 장치는 (n+2)번째 자리까지 고려해야 한다. 그러나, 피 연산자는 음수가 아니므로 피 연산자의 최상위 비트는 $\overline{1}$ 이 될 수 없다. 그러므로 부호 비트는 고려하지 않는다.

<92> 최상위 두 비트에 대한 연산 또한 3단계와 지연 시간 5단계로 구성된다. 연산 1단계는 앞에서 언급한 바와 같이 음수가 아니므로 두 이진수의 덧셈에 대한 부호화 기법을 이용한다. $c1_{n+2}$ 는 이진수로 표현되고

$s1_{n+1}$ 는 음의수로 표현되므로 $c1_{n+2}^1$ 은 0이 되고 $s1_{n+1}^1$ 과 $s1_{n+1}^0$ 은 $(x_{n+1}^0 XOR y_{n+1}^0)$ 의 연산을 통해 동일한 값을 갖는다. $c1_{n+2}^0$ 은 $(x_{n+1}^0 AND y_{n+1}^0) OR s1_{n+1}^0$ 을 통해 연산된다. 연산 2단계와 3단계는 n번째 자리까지의 연산과 동일하게 수행된다. 연산 1단계에서 발생하는 캐리와 연산 2단계에서 발생하는 캐리는 XOR 연산을 통해 $s1_{n+2}$ 를 결정한다.

<93> 도 6c는 도 6b에 의한 연산 과정의 일 예를 도시한 것이다.

<94> 두 부호수의 덧셈을 단계 1이라 하고 부호수와 음의수의 덧셈을 단계 2라고 가정한다. 또한, 단계 1의 결과를 $(C1, S1)$ 이라 하고 단계 2의 결과를 $(C2, S2)$ 라고 가정한다. 최종 결과를 A라 할 때, 두 부호수 $X=(1\overline{1}01100\overline{1}1111)$ 와 $Y=(0111\overline{1}00\overline{1}110\overline{1})$ 의 덧셈은 도 6c과 같이 수행된다.

<95> (x_i, y_i) 값이 (0,1) 또는 (1,0)인 경우, 출력 결과 (c_{i+1}, s_i) 를 (0,1)에서 도 3a와 같이 $(1, \overline{1})$

로 대체하면, $c1_{i+1}$ 은 부호수가 되고 $s1_i$ 은 음의수가 된다. 부호수 $c1_i$ 와 음의수 $s1_i$ 의 덧셈 결과를

$(c2_{i+1}, s2_i)$ 라 할 때, 도 3b와 같이 $(c1_i, s1_i)$ 가 $(\overline{1}, 0)$ 또는 $(0, \overline{1})$ 인 경우 출력 결과의 $(0, \overline{1})$ 에서 $(\overline{1}, 1)$ 로 대체하면 $c2_{i+1}$ 는 음의수가 되고 $s2_i$ 는 이진수가 된다. 따라서 $c2_i$ 와 $s2_i$ 의 덧셈은 캐리 전과 없이 수행된다.

- <96> 도 6d는 종래의 덧셈 방법 및 본 발명에 따른 덧셈 방법의 공간 복잡도와 시간 복잡도를 도시한 표이다.
- <97> 도 6d에서와 같이 본 발명에 따른 덧셈 장치(Nadder)의 공간 복잡도는 게이트의 개수는 종래의(McIvor의 2) 방법보다 약 8% 감소하고, 지연 시간은 종래의(McIvor의 2) 방법보다 약 17% 줄어든다.
- <98> 도 7은 본 발명에 따른 곱셈 방법을 도시한 것이다.
- <99> 입력으로 n+1비트의 제1입력값(X) 및 제2입력값(Y), 제1입력값(X) 및 제2입력값(Y)에 따른 n비트의 모듈러 값(M)을 가정한다.
- <100> 먼저, 곱셈 연산 결과 변수 A를 0으로 초기화시킨다(1).
- <101> 다음, 본 발명에 따른 덧셈 방법(Nadder)을 이용하여 Y와 M을 더한 값을 변수 D에 저장한다(2).
- <102> 다음의 과정은 제1입력값(X) 및 제2입력값(Y)의 0번째 비트부터 n+1번째 비트까지 반복하여 수행한다(3).
- <103> 곱셈 연산 결과 변수 A의 0번째 비트, 제1입력값의 i번째 비트, 제2입력값의 0번째 비트를 이용하여 변수 ui를 결정한다(3.1).
- <104> 제1입력값의 i번째 비트값과 변수 ui의 값에 따라 결과 변수 A를 갱신한다(3.2).
- <105> 이때, 제1입력값의 i번째 비트값이 0이고 변수 ui의 값이 0이면, A+0을 본 발명에 따른 덧셈 방법(Nadder)을 이용하여 연산한 결과를 쉬프트 시킨 값을 A에 저장한다.
- <106> 이때, 제1입력값의 i번째 비트값이 1이고 변수 ui의 값이 0이면, A+Y를 본 발명에 따른 덧셈 방법(Nadder)을 이용하여 연산한 결과를 쉬프트 시킨 값을 A에 저장한다.
- <107> 이때, 제1입력값의 i번째 비트값이 0이고 변수 ui의 값이 1이면, A+M을 본 발명에 따른 덧셈 방법(Nadder)을 이용하여 연산한 결과를 쉬프트 시킨 값을 A에 저장한다.
- <108> 이때, 제1입력값의 i번째 비트값이 1이고 변수 ui의 값이 1이면, A+D를 본 발명에 따른 덧셈 방법(Nadder)을 이용하여 연산한 결과를 쉬프트 시킨 값을 A에 저장한다.
- <109> 위의 과정을 제1입력값(X) 및 제2입력값(Y)의 0번째 비트부터 n+1번째 비트까지 반복하여 수행한다.
- <110> 도 8은 종래의 곱셈 방법 및 본 발명에 따른 곱셈 방법의 지연 시간을 도시한 표이다.
- <111> 캐리 저장 가산기(CSA)를 이용하는 전형적인 알고리즘 종래의 방법 1은 클럭사이클이 n+34로 길고, 32개의 전가산기에 의한 지연시간을 필요로 한다. 또한, 캐리 저장 가산기(CSA)와 멀티플렉서를 이용하는 종래의 방법 2는 캐리 전파에 의한 지연 시간을 필요로 한다.
- <112> 종래의 방법 3(McIvor의 2)을 이용하는 곱셈 장치 본 발명에 따른 덧셈 장치(Nadder)를 이용한 곱셈 장치를 비교하여 효율성을 알아보면 다음과 같다. 이때, 곱셈 장치의 다른 부분을 제외하고 덧셈 부분만 비교 하여 두 곱셈 장치의 효율성 비교가 가능하다. 본 발명에 따른 덧셈 장치(Nadder)를 이용한 곱셈 장치는 종래의 방법 3(McIvor의 2)을 이용한 곱셈 장치의 덧셈 연산 부분과 비교하여 공간 복잡도는 8% 감소하고, 시간 복잡도는 17% 감소한다.
- <113> 따라서 본 발명에 따른 덧셈 장치(Nadder)를 이용한 곱셈 장치는 종래의 곱셈 방법 중 고비도에서 효율적인 종래의(McIvor의 2) 방법을 이용한 곱셈 장치보다 효율적이다.
- <114> 특히, 복잡도를 줄이기 위해서 연산을 수행하는 동안, 부호수의 특징을 이용하여 하이 래디스(High-radix) 연산에 적용할 경우, 2의 보수 연산에서 이진수로 표현할 때 고려해야 하는 캐리 전파를 고려하지 않아도 되는 추가적인 성능 향상을 갖는다. 부호수를 이용한 다양한 부호화 기법은 여러 분야에 적용될 수 있다.
- <115> 바람직하게는, 본 발명의 부호화를 이용하는 덧셈 방법을 컴퓨터에서 실행시키기 위한 프로그램을 컴퓨터로 읽

을 수 있는 기록매체에 기록하여 제공할 수 있다.

- <116> 바람직하게는, 본 발명의 부호화를 이용하는 곱셈 방법을 컴퓨터에서 실행시키기 위한 프로그램을 컴퓨터로 읽을 수 있는 기록매체에 기록하여 제공할 수 있다.
- <117> 본 발명은 소프트웨어를 통해 실행될 수 있다. 소프트웨어로 실행될 때, 본 발명의 구성 수단들은 필요한 작업을 실행하는 코드 세그먼트들이다. 프로그램 또는 코드 세그먼트들은 프로세서 판독 가능 매체에 저장되거나 전송 매체 또는 통신망에서 반송파와 결합된 컴퓨터 데이터 신호에 의하여 전송될 수 있다.
- <118> 본 발명은 도면에 도시된 일 실시예를 참고로 하여 설명하였으나 이는 예시적인 것에 불과하며 당해 분야에서 통상의 지식을 가진 자라면 이로부터 다양한 변형 및 실시예의 변형이 가능하다는 점을 이해할 것이다. 그러나, 이와 같은 변형은 본 발명의 기술적 보호범위내에 있다고 보아야 한다. 따라서, 본 발명의 진정한 기술적 보호범위는 첨부된 특허청구범위의 기술적 사상에 의해서 정해져야 할 것이다.

발명의 효과

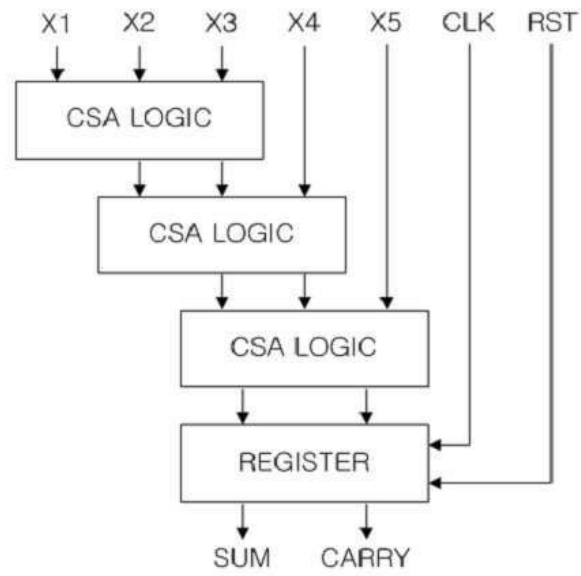
- <119> 상술한 바와 같이, 본 발명에 의하면, 부호와 기법을 적용하여 덧셈 연산을 하거나 부호와 기법을 적용한 덧셈 연산을 이용하여 곱셈 연산을 수행함으로써, 공간 복잡도와 시간 복잡도를 줄여 성능을 향상시키고, 제조 비용을 감소시킬 수 있는 효과가 있다. 또한, 또한, 부호수의 특징을 이용하여 하이 래디스(High-radix) 연산을 적용할 경우, 2의 보수 연산에서 고려해야 하는 캐리 전파를 고려하지 않아도 되는 추가적인 성능 향상을 가능하게 하는 효과가 있다.

도면의 간단한 설명

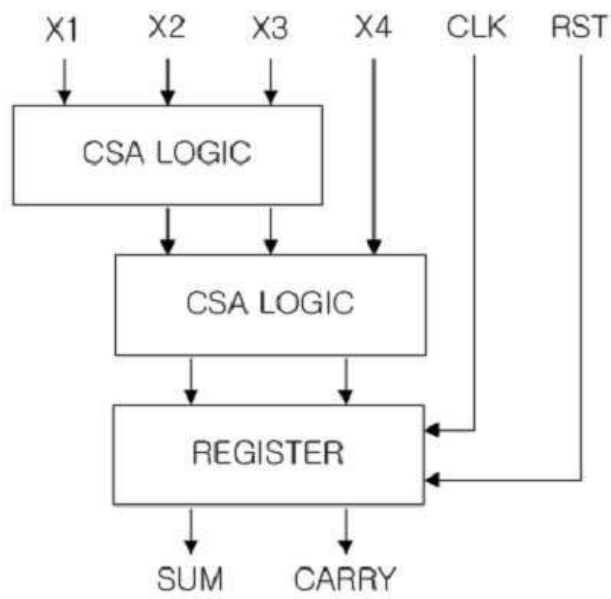
- <1> 도 1a 및 도 1b는 종래의 곱셈 장치의 블록도이다.
- <2> 도 2는 본 발명에 따른 덧셈 장치의 블록도이다.
- <3> 도 3a 및 도 3b는 도 2에 적용되는 부호화 기법을 표로 도시한 것이다.
- <4> 도 4a는 도 2의 상세 블록도이다.
- <5> 도 4b는 본 발명의 일 실시예에 따른 덧셈 장치의 회로도이다.
- <6> 도 5a는 본 발명에 따른 곱셈 장치의 블록도이다.
- <7> 도 5b는 도 5a의 변환기에 대한 상세 블록도이다.
- <8> 도 6a는 본 발명에 따른 덧셈 방법의 흐름도이다.
- <9> 도 6b는 도 6a의 상세 알고리즘을 도시한 것이다.
- <10> 도 6c는 도 6b에 의한 연산 과정의 일 예를 도시한 것이다.
- <11> 도 6d는 종래의 덧셈 방법 및 본 발명에 따른 덧셈 방법의 공간 복잡도와 시간 복잡도를 도시한 표이다.
- <12> 도 7은 본 발명에 따른 곱셈 방법을 도시한 것이다.
- <13> 도 8은 종래의 곱셈 방법 및 본 발명에 따른 곱셈 방법의 지연 시간을 도시한 표이다.

도면

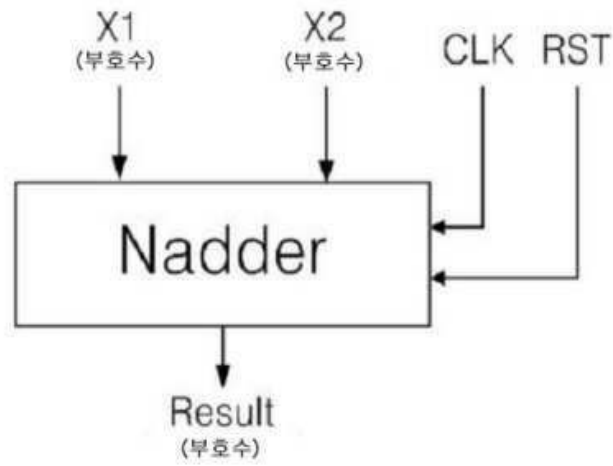
도면1a



도면1b



도면2



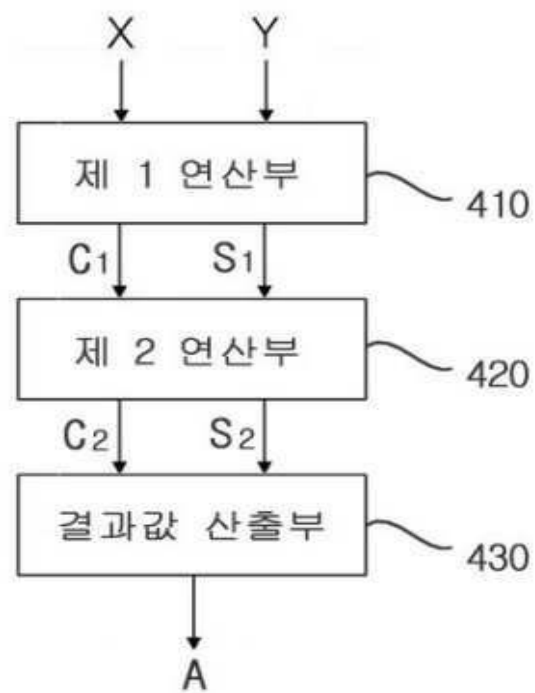
도면3a

x_i	y_i	$c1_{i+1}$	$s1_i$
0	0	0	0
0	1	1	$\overline{1}$
0	$\overline{1}$	0	$\overline{1}$
1	0	1	$\overline{1}$
1	1	1	0
1	$\overline{1}$	0	0
$\overline{1}$	0	0	$\overline{1}$
$\overline{1}$	1	0	0
$\overline{1}$	$\overline{1}$	$\overline{1}$	0

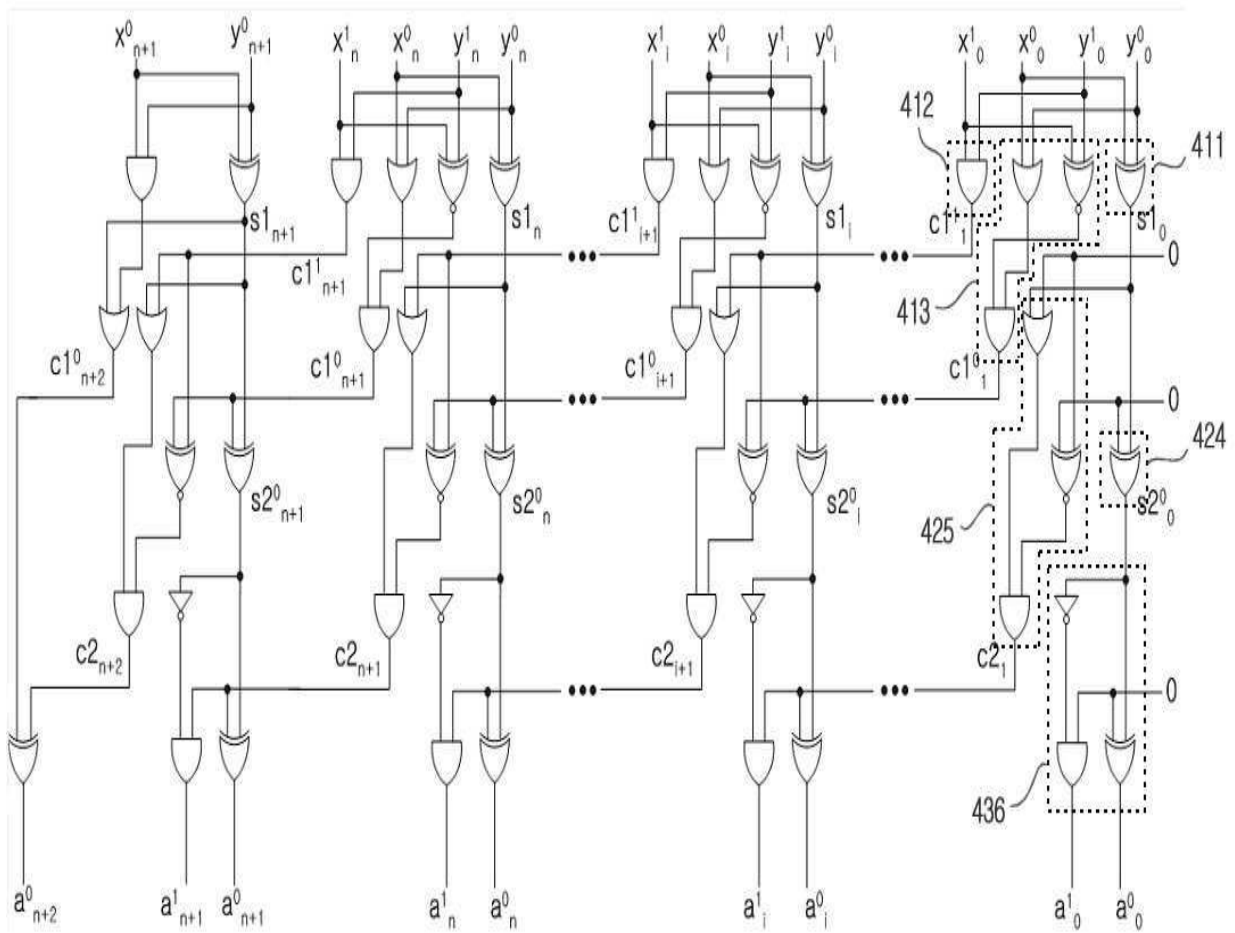
도면3b

$c1_i$	$s1_i$	$c2_{i+1}$	$s2_i$
0	0	0	0
0	$\overline{1}$	$\overline{1}$	1
1	0	0	1
1	$\overline{1}$	0	0
$\overline{1}$	0	$\overline{1}$	1
$\overline{1}$	$\overline{1}$	$\overline{1}$	0

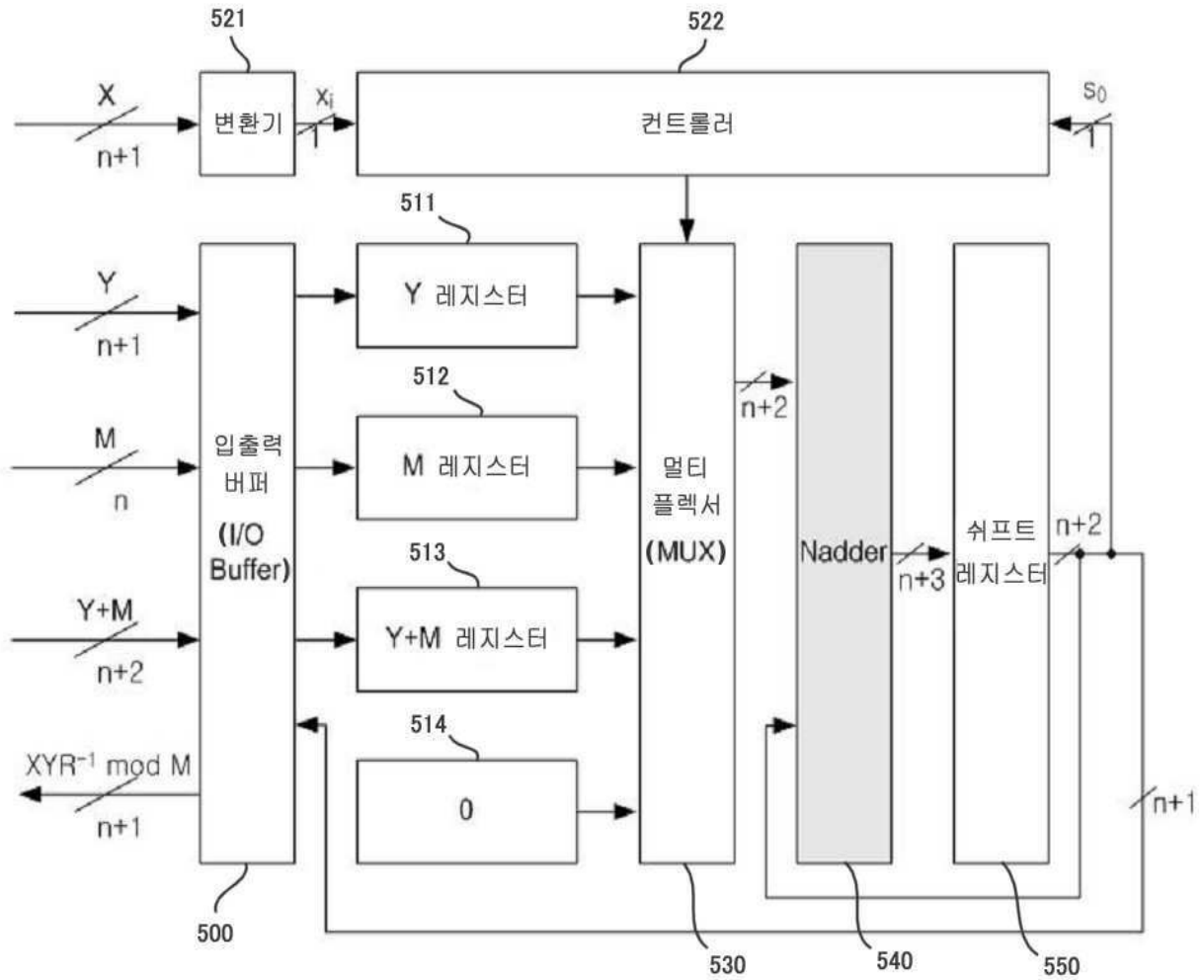
도면4a



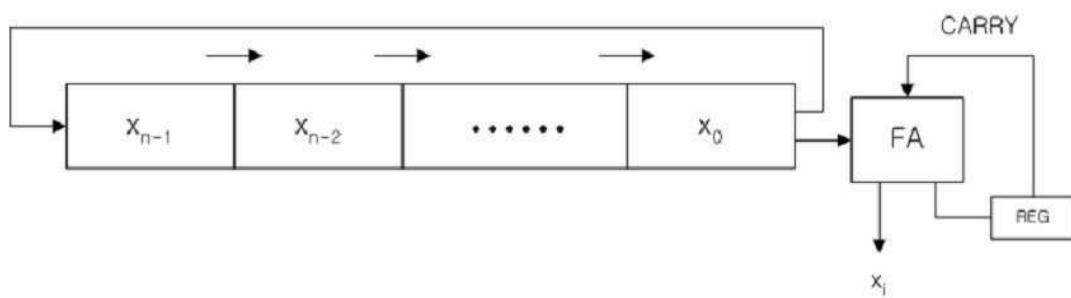
도면4b



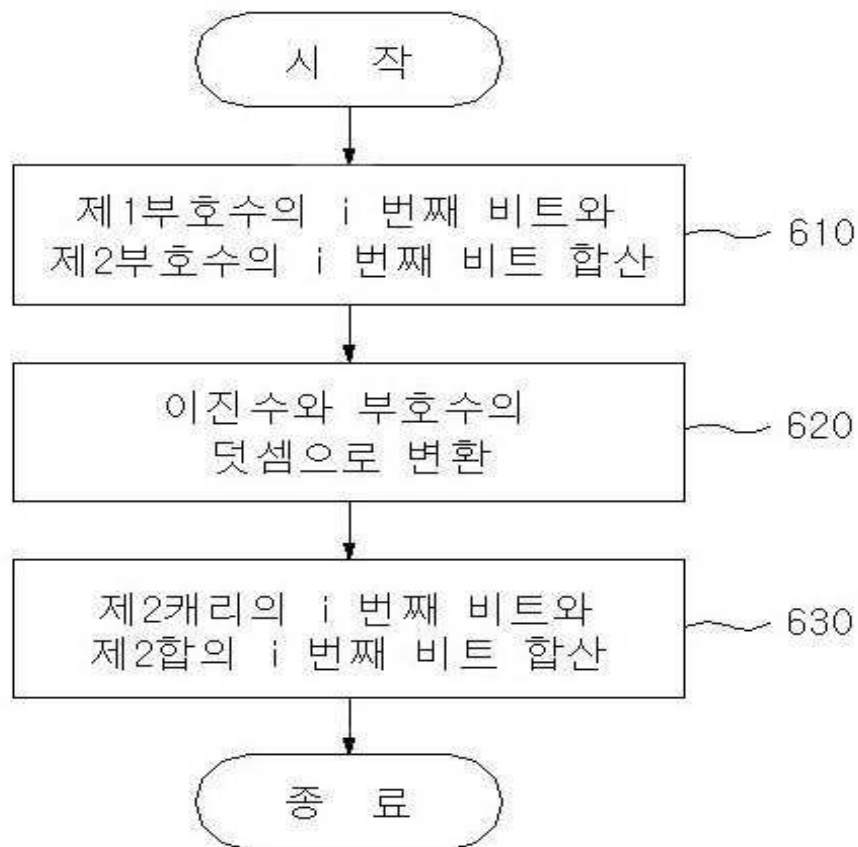
도면5a



도면5b



도면6a



도면6b

입력 : $X = (x_{n+1} \cdots x_1 x_0)_2$, $Y = (y_{n+1} \cdots y_1 y_0)_2$

출력 : $A = (a_{n+2} \cdots a_1 a_0) = X + Y$

Process 1 : (x_i, y_i)

Begin

- 1.1. $(s1_i^1, s1_i^0) \leftarrow (x_i^0 \oplus y_i^0)$.
- 1.2. $c1_{i+1}^1 \leftarrow (x_i^1 \wedge y_i^1)$, $c1_{i+1}^0 \leftarrow \sim (x_i^1 \oplus y_i^1) \wedge (x_i^0 \vee y_i^0)$.
- 1.3. $s2_i^1 \leftarrow 0$, $s2_i^0 \leftarrow (c1_i^0 \oplus s1_i^0)$.
- 1.4. $(c2_{i+1}^1, c2_{i+1}^0) \leftarrow \sim (c1_i^1 \oplus c1_i^0) \wedge (c1_i^1 \vee s1_i^1)$.
- 1.5. $a_i^1 \leftarrow \sim s2_i^0 \wedge c2_i^1$, $a_i^0 \leftarrow c2_i^0 \oplus s2_i^0$.

End process.

Process 2 : (x_{n+1}, y_{n+1})

Begin

- 2.1. $(s1_{n+1}^1, s1_{n+1}^0) \leftarrow (x_{n+1}^0 \oplus y_{n+1}^0)$.
- 2.2. $c1_{n+2}^1 \leftarrow 0$, $c1_{n+2}^0 \leftarrow (x_{n+1}^0 \wedge y_{n+1}^0) \vee s1_{n+1}^0$.
- 2.3. $s2_{n+1}^1 \leftarrow 0$, $s2_{n+1}^0 \leftarrow (c1_{n+1}^0 \oplus s1_{n+1}^0)$.
- 2.4. $(c2_{n+2}^1, c2_{n+2}^0) \leftarrow \sim (c1_{n+1}^1 \oplus c1_{n+1}^0) \wedge (c1_{n+1}^1 \vee s1_{n+1}^1)$.
- 2.5. $a_{n+1}^1 \leftarrow (\sim s2_{n+1}^0 \wedge c2_{n+1}^1)$, $a_{n+1}^0 \leftarrow (c2_{n+1}^0 \oplus s2_{n+1}^0)$, $a_{n+2}^0 \leftarrow (c1_{n+2}^0 \oplus c2_{n+1}^0)$.

End process...

도면6c

X		1	$\bar{1}$	0	1	1	0	0	$\bar{1}$	$\bar{1}$	1	1	1
Y	+	0	1	1	1	$\bar{1}$	0	0	$\bar{1}$	$\bar{1}$	1	0	$\bar{1}$
C1		1	0	1	1	0	0	0	$\bar{1}$	$\bar{1}$	1	0	0
S1			$\bar{1}$	0	$\bar{1}$	0	0	0	0	0	0	$\bar{1}$	0
C2	0	$\bar{1}$	0	0	0	0	$\bar{1}$	$\bar{1}$	0	0	$\bar{1}$	0	0
S2		1	1	1	0	0	0	1	1	1	1	1	0
A		1	1	0	0	0	$\bar{1}$	0	1	1	0	1	0

1단계

2단계

도면6d

	구성		1비트 입력의 복잡도		n비트 입력의 복잡도	
McIvor의 2 방법	종류	CSA	area	4AND+8XOR	area	$4 \cdot n\text{AND} + 8 \cdot n\text{XOR}$
	개수	2	delay	$2T_A + 4T_X$	delay	$2T_A + 4T_X$
본 발명의 방법	종류	Nadder	area	5XOR+2OR+4AND	area	$5 \cdot n\text{XOR} + 2 \cdot n\text{OR} + 4 \cdot n\text{AND}$
	개수	1	delay	$2T_A + 3T_X$	delay	$2T_A + 3T_X$

도면7

입력 : $X = (x_n \cdots x_1 x_0)_2$, $Y = (y_n \cdots y_1 y_0)_2$, $M = (m_{n-1} \cdots m_1 m_0)_2$, $R = 2^{n+2}$, $\gcd(M, R) = 1$,

$$0 \leq X, Y < 2M,$$

출력 : $A = XYR^{-1} \bmod M$.

1. $A \leftarrow 0$.
2. $D \leftarrow \text{Nadder}(Y + M)$.
3. For i from 0 to $n+1$ do : (단, $x_{n+1} = 0$).
 - 3.1. $u_i \leftarrow (a_0 + x_i y_0) \bmod 2$.
 - 3.2. if $x_i = 0, u_i = 0$ then

$$A \leftarrow \text{Nadder}(A + 0) / 2,$$
 elseif $x_i = 1, u_i = 0$ then

$$A \leftarrow \text{Nadder}(A + Y) / 2,$$
 elseif $x_i = 0, u_i = 1$ then

$$A \leftarrow \text{Nadder}(A + M) / 2,$$
 else $A \leftarrow \text{Nadder}(A + D) / 2,$
4. Return(A).

도면8

알고리즘	루프 지연 시간	변환 지연 시간	클럭 사이클	크리티컬 지연 시간
종래의 방법 1	2 CSAs	32비트 CPA의 $n/32$ 반복 연산 수	$n+34$ (1024비트에 대하여)	32FAs
종래의 방법 2	1 CSA + 4:1MUX	n 비트 CPA	언급하지 않음	n 비트 CPAs의 캐리 전파
종래의 방법 3 (McIvor의 2 방법)	2 CSAs + 4:1MUX + 2 XORs + 1 AND	없음	$n+2$	$8T_X + T_A + 4:1MUX$
본 발명의 곱셈장치	Nadder + 4:1MUX + 2 XORs + 1 AND	없음	$n+2$	$5T_X + 3T_A + 4:1MUX$