(19)대한민국특허청(KR)
(12) 등록특허공보(B1)

(51) 。Int. Cl.[6]
G06F 9/00

(45) 공고일자 2005년06월16일
(11) 등록번호 10-0486697
(24) 등록일자 2005년04월22일

(21) 출원번호 10-1998-0019040
(22) 출원일자 1998년05월26일
(65) 공개번호 10-1999-0086179
(43) 공개일자 1999년12월15일

(73) 특허권자 삼성전자주식회사
경기도 수원시 영통구 매탄동 416

(72) 발명자 이경희
경기도 수원시 팔달구 영통동 벽적골 한신아파트 816동 1205호

차영태
경기도 성남시 분당구 이매촌 삼성아파트 1001동 804호

(74) 대리인 이영필

심사관 : 성경아

(54) 모듈러연산장치및그방법

요약

공개키 암호화/복호화 및 디지털 서명 시스템에 사용되는 모듈러 연산장치에 관한 것으로서, 본 발명에 의한 모듈러 연산 $A \cdot B \cdot 2^{-k} \bmod N$을 계산하는 장치는 k비트의 저장용량을 지니고, 각각 A, B, N을 저장하는 A메모리수단, B메모리수단, N메모리수단; 두 개의 w비트의 값을 병렬로 입력받아 곱한 2w비트의 결과를 병렬로 출력하는 곱셈기; 두 개의 2w비트의 값을 병렬로 입력받아 더한 결과를 병렬로 출력하는 덧셈기; k+2w비트의 저장용량을 지니고, 한 클럭 내에 그 중 2w비트를 결정하여 덧셈기로 출력하고, 덧셈기로부터 입력된 값을 출력한 비트의 위치에 저장하는 누산기; w비트의 저장용량을 지니고, 미리 계산된 $N_0^{-1} \bmod 2^w$ 저장하는 J메모리수단; w비트의 저장용량을 지니고, 곱셈기의 출력으로부터 상위 w비트를 저장하는 q메모리수단; J메모리수단, B메모리수단 및 N메모리수단 중에서 하나를 선택하여 곱셈기에 접속하는 제1선택수단; A메모리수단, q메모리수단 및 누산기 중에서 하나를 선택하여 곱셈기에 접속하는 제2선택수단; 및 q메모리수단 및 덧셈기 중에서 하나를 선택하여 곱셈기 접속하는 제3선택수단을 포함함을 특징으로 한다.

본 발명에 의하면, 디지털 서명기기, 공개키 암호화/복호화기기에서 필요로 하는 모듈러 곱셈 또는 모듈러 멱승을 계산하기 위한 회로인 $A \cdot B \cdot 2^{-k} \bmod N$을 계산하는 회로를 간단하게 구현함으로써 디지털 서명기기 및 공개키 암호화/복호화기기 제작에 요구되는 비용을 줄일 수 있다.

대표도

도 1

명세서

도면의 간단한 설명

도 1은 본 발명에 의한 모듈러 연산장치의 블록 구성도이다.

도 2는 도 1의 모듈러 연산장치에서의 타이밍도이다.

발명의 상세한 설명

발명의 목적

발명이 속하는 기술 및 그 분야의 종래기술

본 발명은 모듈러 연산장치에 관한 것으로서, 특히 공개키 암호화/복호화 및 디지털 서명 시스템에 사용되는 모듈러 연산장치 및 그 방법에 관한 것이다.

1976년 디피(Diffie)와 헬만(Hellman)은 수학적으로 매우 풀기 어려운 문제의 일방성을 이용한 공개키 암호시스템(Public Key Cryptosystem)의 개념을 처음으로 소개하여 현대 암호학의 새로운 전기를 마련하였다. 기존의 관용(대칭형) 암호시스템(conventional or symmetric cryptosystem)에서는 통신하고자 하는 두 사용자가 동일한 비밀키를 공유하여야 하므로 키관리가 어렵고, 디지털 서명(digital signature)을 구현하기 어렵다는 등의 단점이 있었다. 그런데, 공개키 암호시스템에서는 수학적으로 풀기 어려운 문제의 일방성을 이용하여 공개키와 비밀키를 계산하고, 공개키는 누구나 이용할 수 있게 공개하며 비밀키만 각 사용자가 보관하게 된다. 따라서, 공개된 상대방의 공개키를 가진 사용자는 누구나 그 상대방과 비밀통신을 할 수 있게 된다.

공개키 암호시스템에서 가장 널리 이용되는 어려운 문제로는 이산대수 문제(Discrete Logarithm Problem)와 소인수분해 문제가 있다. 대표적인 공개키 암호시스템으로는 이산대수 문제에 근거한 엘가말(ElGamal)형의 암호시스템과 소인수분해 문제에 근거한 알.에스.에이(RSA:Revest Shamir Adleman) 암호시스템이 있다. 표준으로 채택된 것도 국제표준으로는 ISO(the International Organization for Standardization:국제표준기구)/IEC(the International Electrotechnical Commission:국제전자기술위원회) 9796, 엘가말형의 변형인 미국의 DSA, 러시아의 GOST 등이 있으며, 한국에서는 KCDSA가 있다.

이러한 공개키 암호시스템들은 대부분 모듈러 멱승(modular exponentiation: m^e mod N) 연산을 필요로 하고, 이 모듈러 멱승 연산을 위해서는 모듈러 곱셈 (AB mod N)을 수행하는 것이 반드시 필요하다.

모듈러 곱셈을 위한 알고리듬으로는 고전적인 알고리듬, 바레트(Barret)의 알고리듬, 그리고 몽고메리(Montgomery) 알고리듬 등이 제안되어 있다.

상기 고전적인 알고리듬은 보통 연필로 나눗셈을 하여 나머지를 구하듯이 한자리씩 몫을 추정하여 나머지를 구하는 과정을 반복함으로써 모듈러 감소를 하는 방법이다. 이는 법 M에 대한 제약이 없으며 사전(事前) 계산이나 사후(事後) 계산이 필요없으므로 어느 경우에나 적용될 수 있는 가장 일반적인 모듈러 감소 알고리듬이다. 그러나, 몫을 추정하는 과정에서 (곱셈에 비해 속도가 느린) 나눗셈이 필요하고 추정된 몫이 정확한 값이 아닌 경우 추가적인 덧셈이나 뺄셈이 필요하므로 비교적 속도가 느린 편이다.

바레트 알고리듬은 고정된 법에 대한 사전계산값을 이용하여 전체 몫을 한꺼번에 추정하여 곱셈만으로 모듈러 감소를 수행한다. 이는 법 M이 고정되어 있는 경우 또는 같은 법에 대해 많은 수의 모듈러 곱셈이 필요한 모듈러 멱승 연산시 고전적인 알고리듬에 비해 좀 더 나은 성능을 보여준다.

몽고메리 알고리듬은 수체제의 변환을 통해 나눗셈없이 나머지를 구하는 알고리듬으로 다은 알고리듬에 비해 속도가 빠르므로 모듈러 멱승이 필요한 공개키 암호시스템의 구현시 가장 널리 이용된다. 즉 주어진 수들을 곱셈만으로 모듈러 감소를 할 수 있는 다은 수체제로 변환하여 거기서 모듈러 감소시킨 후 이를 다시 원래의 수체제로 역변환시켜 원하는 결과를 얻게 된다. 대부분의 공개키 암호시스템에서 요구되는 모듈러 멱승의 연산시는 이러한 사전/사후 계산은 전체의 수행속도에 거의 영향을 미치지 못하므로 이 알고리듬은 전체적으로 다른 알고리듬에 비해 매우 좋은 성능을 보여준다.

발명이 이루고자 하는 기술적 과제

본 발명은 몽고메리 알고리듬을 이용하여 공개키 암호/복호 및 디지털 서명에 이용되는 모듈러 멱승 및 모듈러 곱셈 연산을 효율적으로 수행할 수 있으며, 그 구성이 간단한 모듈러 연산장치 및 그 방법을 제공함을 그 목적으로 한다.

발명의 구성 및 작용

상기의 목적을 달성하기 위하여, 본 발명에 의한 모듈러 연산 $A \cdot B \cdot 2^{-k} \bmod N$을 계산하는 장치는 k비트의 저장용량을 지니고, 상기 A값을 병렬로 입력받고, 소정의 클럭마다 w비트 단위로 하위비트 방향으로 쉬프트하며, 최하위 w비트를 병렬로 출력하는 A메모리수단; k비트의 저장용량을 지니고, 상기 B값을 병렬로 입력받고, 소정의 클럭마다 w비트 단위로 하위비트 방향으로 로테이트하며, 최하위 w비트를 병렬로 출력하는 B메모리수단; k비트의 저장용량을 지니고, 상기 N값을 병렬로 입력받고, 소정의 클럭마다 w비트 단위로 하위비트 방향으로 로테이트하며, 최하위 w비트를 병렬로 출력하는 N메모리수단; 두 개의 w비트의 값을 병렬로 입력받아 곱한 2w비트의 결과를 병렬로 출력하는 곱셈기; 두 개의 2w비트의 값을 병렬로 입력받아 더한 결과를 병렬로 출력하는 덧셈기; k+ 2w비트의 저장용량을 지니고, 한 클럭 내에 그 중 2w비트를 결정하여 상기 덧셈기로 출력하고, 상기 덧셈기로부터 입력된 값을 출력한 비트의 위치에 저장하는 누산기; w비트의 저장용량을 지니고, 미리 계산된 $N_0^{-1} \bmod 2^w$를 병렬로 입력받고 병렬로 출력하는 J메모리수단(여기에서 N_0는 N의 최하위 w비트이다); w비트의 저장용량을 지니고, 상기 곱셈기의 출력으로부터 상위 w비트를 병렬로 입력받고 병렬로 출력하는 q메모리수단; 상기 J메모리수단, 상기 B메모리수단 및 상기 N메모리수단의 w비트 출력들 중에서 하나의 출력을 선택하여 상기 곱셈기에 전달하는 제1선택수단; 상기 A메모리수단, 상기 q메모리수단 및 상기 누산기의 w비트 출력들 중에서 하나의 출력을 선택하여 상기 곱셈기에 전달하는 제2선택수단; 및 상기 q메모리수단 및 상기 덧셈기 중에서 하나를 선택하여 상기 곱셈기의 출력을 전달하는 제3선택수단을 포함함을 특징으로 한다(여기에서, k = w·s이고, k,w,s는 모두 2이상의 정수).

상기의 다른 목적을 달성하기 위하여, 본 발명에 의한 모듈러 연산장치를 이용하여 모듈러 연산 $A \cdot B \cdot 2^{-k} \bmod N$을 계산하는 방법은 (a) k비트인 A,B,N을 각각 상기 A메모리수단, 상기 B메모리수단 및 상기 N메모리수단에 저장하고, $N_0^{-1} \bmod 2^w$를 미리 계산하여 상기 J메모리수단에 저장하고, 상기 누산기를 '0'으로 초기화하는 단계; (b) (b.1) 상기 곱셈기가 상기 A메모리수단에 저장된 최하위 w비트의 A_0와 상기 B메모리수단에 저장된 최하위 w비트의 B_0의 곱셈을 수행하는 단계; 및 (b.2) 상기 덧셈기가 상기 곱셈기의 계산한 결과와 상기 누산기의 2w비트의 S_iS_{i-1}(단, i는 반복회수를 나타낸다)에 저장된 값과 더하여, 상기 누산기의 S_iS_{i-1}(단, i는 반복회수를 나타낸다)에 저장하는 단계를 수행하되, 매 클럭마다 상기 B메모리수단을 w비트 하위비트측으로 쉬프트시키고, 상기 누산기의 입출력되는 위치를 w비트 상위비트측으로 이동하면서 s번 반복 수행하여 A_0B 값을 계산하는 단계; (c) 상기 곱셈기가 상기 누산기의 최하위 w비트 S_0와 상기 J메모리수단의 J_0을 곱하여 상기 q메모리수단에 저장하는 단계; (d) (d.1) 상기 곱셈기가 상기 N메모리수단에 저장된 최하위 w비트의 N_0와 상기 q메모리수단에 저장된 q_0의 곱셈을 수행하는 단계; 및 (d.2) 상기 덧셈기가 상기 곱셈기의 계산한 결과와 상기 누산기의 2w비트의 S_iS_{i-1}(단, i는 반복회수를 나타낸다)에 저장된 값과 더하여, 상기 누산기의 S_iS_{i-1}(단, i는 반복회수를 나타낸다)에 저장하는 단계를 수행하되, 매 클럭마다 상기 N메모리수단을 w비트 하위비트측으로 쉬프트시키고, 상기 누산기의 입출력되는 위치를 w비트 상위비트측으로 이동하면서 s번 반복 수행하여 q_0N 값을 계산하는 단계; (e) 상기 누산기에 저장된 값을 w비트 하위비트측으로 쉬프트하는 단계; (f) 상기 A메모리수단에 저장된 A 값을 w비트 하위비트측으로 쉬프트하면서, 상기 (b) 단계 내지 상기 (e) 단계를 s번 수행하는 단계; 및 (g) 상기 누산기에 저장된 값이 N보다 크면, S = S - N을 수행하는 단계를 포함함을 특징으로 하는 모듈러 연산방법.

이하에서 첨부된 도면을 참조하여 본 발명을 상세히 설명한다.

본 발명은 각각 k비트인 수

$$A = a_{k-1} \cdot 2^{k-1} + \ldots + a_1 \cdot 2 + a_0,$$

$$B = b_{k-1} \cdot 2^{k-1} + \ldots + b_1 \cdot 2 + b_0,$$

$$N = n_{k-1} \cdot 2^{k-1} + \ldots + n_1 \cdot 2 + n_0$$

에 대하여 몽고메리 알고리듬을 이용하여 A·B mod N을 효율적으로 구현하기 위한 것이다. 이때, A,B,N은 각각 $A = \sum_{i=0}^{i=s-1} A_i r^i$, $B = \sum_{i=0}^{i=s-1} B_i r^i$, $N = \sum_{i=0}^{i=s-1} N_i r^i$으로 나타낼 수 있는 큰 수이고, w비트인 워드 s개로 나타낼 수 있다. 즉, $r=2^w$일 때, k=s·w이고, k,w,s는 모두 2이상의 정수이다. 몽고메리 함수 $f_m(A,B,N)$은 $A \cdot B \cdot R^{-1} \bmod N$을 계산하는 함수이다(단, $R=2^k$).

몽고메리 함수 $f_m(\cdot)$을 이용하여 모듈러 곱셈 A·B mod N을 수행하기 위해서는 $P=R^2 \bmod N$을 계산하여 미리 저장해 두며, $T = f_m(A,B,N) = A \cdot B \cdot R^{-1} \bmod N$을 계산한 후, $P \cdot T \cdot R^{-1} \bmod N = A \cdot B \bmod N$으로 계산할 수 있다.

몽고메리 알고리듬에 의하면, $T = f_m(A,B,N) = A \cdot B \cdot R^{-1} \bmod N$은 다음처럼 계산할 수 있다. 단, $r = 2^w$, $R = r^s$, $J_0 \equiv N_0^{-1} \bmod r$이고, S는 중간값을 저장하는 누산기(accumulator)이다.

S = 0

for i = 0 to s-1

S = S + $A_i \times B$

$q_i = S_0 \times J_0 \bmod r$

S = S + $q_i \times N$

$S = S/2^w$

endfor

if S > N then S = S - N

본 발명은 상기 알고리듬을 하드웨어적으로 구현하기 위해 몇 개의 레지스터(또는 메모리), MUX, w비트의 병렬곱셈기, 2w비트의 덧셈기를 이용하는 방법을 제안하고 있다.

도 1에 의하면, 본 발명에 의한 모듈러 연산장치는 하드웨어의 복잡도를 최소화하기 위하여 곱셈기와 덧셈기를 각각 하나씩만 사용하여 구성하였다.

A메모리수단(10), B메모리수단(12) 및 N메모리수단(14)은 각각 k비트의 레지스터 또는 유사한 형태의 메모리로 구현되며, 각 클럭마다 한 워드 단위인 w비트 단위로 A메모리수단(10)은 오른쪽으로 쉬프트(shift)하며, B메모리수단(12) 및 N메모리수단(14)은 각각 오른쪽으로 로테이트(rotate)한다.

누산기(20)는 k+ 2w비트의 레지스터로서, 계산 과정의 임시값을 저장하며, 한 클럭 내에 덧셈기(18)와 상호 2w비트 단위로 읽기와 쓰기를 한다.

J메모리수단(22)은 미리 계산된 값을 저장하는 w비트의 레지스터 또는 유사한 형태의 메모리로 구현되고, q메모리수단(24)은 매 클럭마다 계산되는 값을 저장하는 w비트의 레지스터 또는 유사한 형태의 메모리로 구현된다.

곱셈기(16)는 두 개의 w비트의 값을 병렬로 입력받아 2w비트의 결과를 한 클럭에 계산하는 것이다.

참조번호 26, 28은 각각 멀티플렉서이고, 참조번호 30은 디멀티플렉서이다.

이하에서 본 발명에 의한 모듈러 연산장치의 동작을 설명한다.

본 발명에 의한 모듈러 연산장치는 각각 w비트인 워드 s개로 이루어진 A, B, N을 입력받아 $s\cdot(2s+4)$클럭 내에 $A\cdot B\cdot R^{-1} \bmod N$을 계산한다.

도 1은 $T = f_m(A,B,N) = A\cdot B\cdot R^{-1} \bmod N$을 계산하기 위한 회로로서, 다음의 알고리듬을 바탕으로 하고 있다.

$S = 0$

for $i = 0$ **to** $s-1$

for $j = 0$ **to** $s-1$

$S = S + A_i \times B_j$

endfor

$q_i = S_0 \times J_0 \bmod r$

for $j = 0$ **to** $s-1$

$S = S + q_i \times N_j$

endfor

$S = S/2^w$

endfor

if $S > N$ **then** $S = S - N$

다음은 도 1을 바탕으로 $T = f_m(A,B,N) = A\cdot B\cdot R^{-1} \bmod N$을 계산하는 과정을 설명한다.

(a) 먼저, 초기화단계로서 k비트인 A,B,N을 각각 A메모리수단(10), B메모리수단(12) 및 N메모리수단(14)에 저장한다. 누산기(20)은 '0'으로 초기화한다.

(b) 각각의 메모리수단(10, 12, 14)에 모든 데이터가 입력되었을 때, 제1멀티플렉서(26)는 B메모리수단(12)에 저장된 w비트의 B_0, 제2멀티플렉서는 A메모리수단(10)에 저장된 w비트의 A_0를 선택한다.

(c) 곱셈기(16)는 A_0와 B_0에 저장된 값을 병렬 입력하여 곱셈을 수행하고, 그 결과인 $A_0\times B_0$는 디멀티플렉서(30)에 의해 선택된 덧셈기(18)로 전달되어 누산기(20)의 2w비트의 S_iS_{i-1}(단, i는 반복회수를 나타낸다)에 저장된 값과 더하여진다. 이때 더한 결과값은 다음 클럭에 누산기(20)의 S_iS_{i-1}(단, i는 반복회수를 나타낸다)에 저장되고, 다음의 덧셈을 위하여 캐리(carry)값은 보관된다.

(d) 매 클럭마다 B메모리수단(12)를 w비트 오른쪽으로 쉬프트시키고 (b)와 (c)의 과정을 s번 반복 수행하여 A_0B 값을 계산한다. 이 과정 중 누산기(20)가 덧셈기(18)과 주고 받는 데이터는 매 클럭마다 w비트 단위로 상위비트측으로 이동한다.

(e) A_0B 값을 계산한 후, 제2멀티플렉서를 이용하여 S_0 값을 선택하고, 제1멀티플렉서를 이용하여 J_0값을 선택하여 s번째 클럭에서 곱셈기(16)에 의해 q_0값을 계산한 후, 그 다음 클럭에 디멀티플렉서(30)에 의해 선택된 q메모리수단(24)에 저장한다.

(f) 제1멀티플렉서는 N메모리수단(14)에 저장된 N을 선택하고, 제2멀티플렉서는 q메모리수단(24)에 저장된 q_0값을 선택한다.

(g) 다른 s클럭 동안 (b)~(d) 과정에서 A_0B를 계산하는 것과 비슷하게 q_0N을 계산한다.

(h) 누산기(20)에 저장된 값을 한 워드 단위인 w비트 오른쪽으로 쉬프트한다.

(i) A메모리수단(10)에 저장된 A 값을 한 워드 단위인 w비트 오른쪽으로 쉬프트하면서, (b)~(h) 과정을 s번 수행하면, 누산기(20)에는 $T = f_m(A,B,N) = A \cdot B \cdot R^{-1} \bmod N$ 값이 저장된다.

(j) 누산기(20)에 저장된 값이 N보다 크면 S = S - N을 수행한다.

본 발명에서 메모리수단을 제외한 모든 소자는 단순한 조합회로로 구현되므로 회로의 잘못된 동작을 방지하기 위하여 연결된 조합회로의 전파지연시간을 충분히 보장하는 정도의 클럭을 제공하여야 한다.

도 2에는 이러한 계산과정의 타이밍 관계가 도시되어 있다.

따라서, 본 발명에 의한 모듈러 연산장치를 이용하여 모듈러 곱셈 $A \cdot B \bmod N$을 계산하는 과정은 다음과 같다.

(1) 먼저, 미리 $P = 2^{2k} \bmod N$을 계산해 둔다.

(2) 다음, 도 1에 도시된 회로를 이용하여 $C = A \cdot B \cdot 2^{-k} \bmod N$을 계산한다.

(3) 마지막으로, $P \cdot C \cdot 2^{-k} \bmod N = A \cdot B \bmod N$을 계산한다.

도 1의 모듈러 연산장치를 이용하여 모듈러 멱승 $m^e \bmod N$을 계산하는 과정은 다음과 같다.

(1) 지수 e를 레지스터 또는 유사한 형태의 메모리에 저장한다.

(2) 레지스터 N에 법 N을 저장한다.

(3) 누산기 S를 '0'으로 초기화한다.

(4) 몽고메리 모듈러 곱셈 $m' = f_m(m, P, N) = m \cdot P \cdot R^{-1} \bmod N$을 수행한다. 단, 멱승 연산의 밑 P는 모듈러 곱셈을 계산하는 과정 중 (1)에서 미리 계산한 값과 동일한 값이다.

(5) m'을 레지스터 B에 로드한다.

(6) 레지스터 B에 로드된 값을 이용하여 모듈러 제곱 연산을 수행한다. 이때, 몽고메리 모듈러 곱셈에 필요한 A는 레지스터 B에서 로드한다.

(7) 지수 e의 최상위비트(Most Significant Bit:MSB)인 '1'을 무시하고, 다음 비트를 현재비트로 한다.

(8) 레지스터 B에 저장된 값을 승수 및 피승수로 하여 모듈러 제곱 연산을 수행한다. 결과값은 레지스터 B에 로드한다.

(9) 지수 e의 현재비트가 '1'인 경우에는 멱승 연산의 밑 m'을 승수로하고, 레지스터 B의 값을 피승수로 하여 모듈러 곱셈 연산을 수행한다. 결과값은 레지스터 B에 로드한다.

(10) 지수 e의 모든 비트에 대하여 단계 (8) 내지 단계 (9)를 수행한 후, 1을 승수로 하고, 레지스터 B의 값을 피승수로 하여 모듈러 곱셈 연산을 수행한다.

단계 (1) 내지 단계 (10)을 수행한 후, 누산기 S에 남아 있는 값이 최종적인 모듈러 멱승 $m^e \bmod N$이 된다.

발명의 효과

본 발명에 의하면, 디지털 서명기기, 공개키 암호화/복호화기기에서 필요로 하는 모듈러 곱셈 또는 모듈러 멱승을 계산하기 위한 회로인 $A \cdot B \cdot 2^{-k} \bmod N$을 계산하는 회로를 간단하게 구현함으로써 디지털 서명기기 및 공개키 암호화/복호화기기 제작에 요구되는 비용을 줄일 수 있다.

(57) 청구의 범위

청구항 1.

모듈러 연산 $A \cdot B \cdot 2^{-k} \bmod N$을 계산하는 장치에 있어서,

k비트의 저장용량을 지니고, 상기 A값을 병렬로 입력받고, 소정의 클럭마다 w비트 단위로 하위비트 방향으로 쉬프트하며, 최하위 w비트를 병렬로 출력하는 A메모리수단;

k비트의 저장용량을 지니고, 상기 B값을 병렬로 입력받고, 소정의 클럭마다 w비트 단위로 하위비트 방향으로 로테이트하며, 최하위 w비트를 병렬로 출력하는 B메모리수단;

k비트의 저장용량을 지니고, 상기 N값을 병렬로 입력받고, 소정의 클럭마다 w비트 단위로 하위비트 방향으로 로테이트하며, 최하위 w비트를 병렬로 출력하는 N메모리수단;

두 개의 w비트의 값을 병렬로 입력받아 곱한 2w비트의 결과를 병렬로 출력하는 곱셈기;

두 개의 2w비트의 값을 병렬로 입력받아 더한 결과를 병렬로 출력하는 덧셈기;

k+2w비트의 저장용량을 지니고, 한 클럭 내에 그 중 2w비트를 결정하여 상기 덧셈기로 출력하고, 상기 덧셈기로부터 입력된 값을 출력한 비트의 위치에 저장하는 누산기;

w비트의 저장용량을 지니고, 미리 계산된 $N_0^{-1} \bmod 2^w$를 병렬로 입력받고 병렬로 출력하는 J메모리수단(여기에서 N_0는 N의 최하위 w비트이다);

w비트의 저장용량을 지니고, 상기 곱셈기의 출력으로부터 상위 w비트를 병렬로 입력받고 병렬로 출력하는 q메모리수단;

상기 J메모리수단, 상기 B메모리수단 및 상기 N메모리수단의 w비트 출력들 중에서 하나의 출력을 선택하여 상기 곱셈기에 전달하는 제1선택수단;

상기 A메모리수단, 상기 q메모리수단 및 상기 누산기의 w비트 출력들 중에서 하나의 출력을 선택하여 상기 곱셈기에 전달하는 제2선택수단; 및

상기 q메모리수단 및 상기 덧셈기 중에서 하나를 선택하여 상기 곱셈기의 출력을 전달하는 제3선택수단을 포함함을 특징으로 하는 모듈러 연산장치(여기에서, k = w·s이고, k,w,s는 모두 2이상의 정수).

청구항 2.

제1항에 있어서, 상기 덧셈기 및 상기 곱셈기는 각각

한 클럭 내에 덧셈 및 곱셈을 수행하는 것을 특징으로 하는 모듈러 연산장치.

청구항 3.

제1항에 있어서, 상기 제1선택수단 및 상기 제2선택수단은 각각

멀티플렉서이고,

상기 제3선택수단은

디멀티플렉서임을 특징으로 하는 모듈러 연산장치.

청구항 4.

청구항 1의 모듈러 연산장치를 이용하여 모듈러 연산 $A \cdot B \cdot 2^{-k} \bmod N$을 계산하는 방법에 있어서,

(a) k비트인 A,B,N을 각각 상기 A메모리수단, 상기 B메모리수단 및 상기 N메모리수단에 저장하고, $N_0^{-1} \bmod 2^w$를 미리 계산하여 상기 J메모리수단에 저장하고, 상기 누산기를 '0'으로 초기화하는 단계;

(b) (b.1) 상기 곱셈기가 상기 A메모리수단에 저장된 최하위 w비트의 A_0와 상기 B메모리수단에 저장된 최하위 w비트의 B_0의 곱셈을 수행하는 단계; 및

(b.2) 상기 덧셈기가 상기 곱셈기의 계산한 결과와 상기 누산기의 2w비트의 S_iS_{i-1}(단, i는 반복회수를 나타낸다)에 저장된 값과 더하여, 상기 누산기의 S_iS_{i-1}(단, i는 반복회수를 나타낸다)에 저장하는 단계를 수행하되, 매 클럭마다 상기 B메모리수단을 w비트 하위비트측으로 쉬프트시키고, 상기 누산기의 입출력되는 위치를 w비트 상위비트측으로 이동하면서 s번 반복 수행하여 A_0B 값을 계산하는 단계

(c) 상기 곱셈기가 상기 누산기의 최하위 w비트 S_0와 상기 J메모리수단의 J_0을 곱하여 상기 q메모리수단에 저장하는 단계;

(d) (d.1) 상기 곱셈기가 상기 N메모리수단에 저장된 최하위 w비트의 N_0와 상기 q메모리수단에 저장된 q_0의 곱셈을 수행하는 단계; 및

(d.2) 상기 덧셈기가 상기 곱셈기의 계산한 결과와 상기 누산기의 2w비트의 S_iS_{i-1}(단, i는 반복회수를 나타낸다)에 저장된 값과 더하여, 상기 누산기의 S_iS_{i-1}(단, i는 반복회수를 나타낸다)에 저장하는 단계를 수행하되, 매 클럭마다 상기 N메모리수단을 w비트 하위비트측으로 쉬프트시키고, 상기 누산기의 입출력되는 위치를 w비트 상위비트측으로 이동하면서 s번 반복 수행하여 q_0N 값을 계산하는 단계;

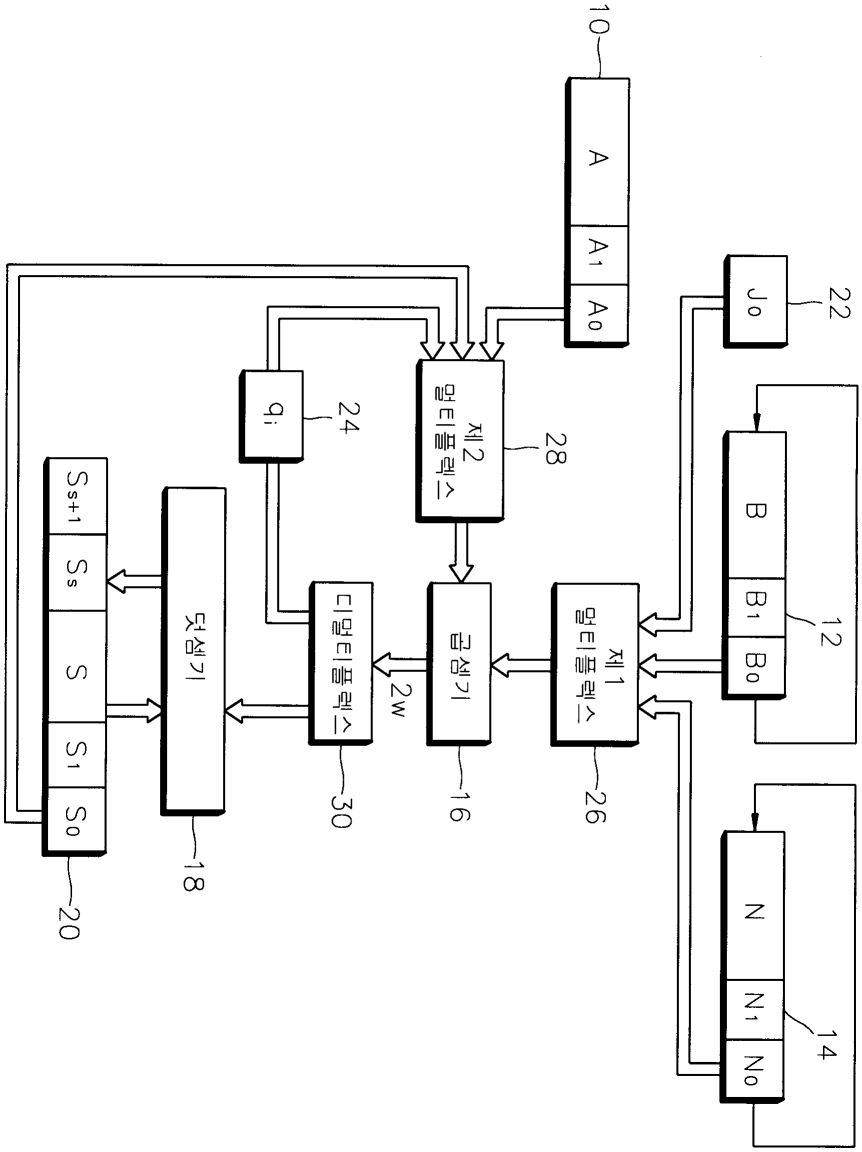
(e) 상기 누산기에 저장된 값을 w비트 하위비트측으로 쉬프트하는 단계;

(f) 상기 A메모리수단에 저장된 A 값을 w비트 하위비트측으로 쉬프트하면서, 상기 (b) 단계 내지 상기 (e) 단계를 s번 수행하는 단계; 및

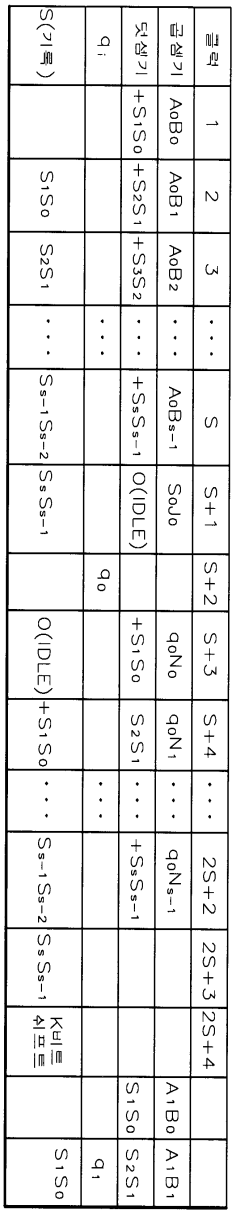
(g) 상기 누산기에 저장된 값이 상기 N값보다 큰 경우, 상기 누산기에 저장된 값에서 상기 N값을 감산한 후, 상기 결과값을 상기 누산기에 저장하는 단계를 포함함을 특징으로 하는 모듈러 연산방법.

도면

도면1


10
A
A1
A0
22
J0
24
qi
28
제2 멀티플렉스
B
B1
B0
12
Ss+1
Ss
S
S1
S0
덧셈기
디멀티플렉스
2w
곱셈기
제1 멀티플렉스
N
N1
N0
14
18
20
30
16
26

도면2


클럭
1
2
3
S
S+1
S+2
S+3
S+4
2S+2
2S+3
2S+4
곱셈기
AoBo
AoB1
AoB2
AoBs-1
SoJo
qoNo
qoN1
qoNs-1
덧셈기
+S1So
+S2S1
+S3S2
+SsSs-1
O(IDLE)
+S1So
+SsSs-1
qi
qo
S(기록)
S1So
S2S1
Ss-1Ss-2
SsSs-1
O(IDLE)+S1So
Ss-1Ss-2
SsSs-1
K비트 쉬프트
A1Bo
A1B1
S1So
S2S1
q1