(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2015-0078681
(43) 공개일자 2015년07월08일

(51) 국제특허분류(Int. Cl.)
G06F 11/10 (2006.01) *G06F 7/523* (2006.01)
(21) 출원번호 10-2013-0168272
(22) 출원일자 2013년12월31일
심사청구일자 2013년12월31일

(71) 출원인
고려대학교 산학협력단
서울특별시 성북구 안암로 145, 고려대학교 (안암동5가)
(72) 발명자
홍석희
서울특별시 도봉구 마들로 859-19 103-107(도봉동, 한신아파트)
이상엽
서울특별시 성북구 안암로 145 고려대학교 정보보호대학원 (안암동5가)
(뒷면에 계속)
(74) 대리인
특허법인충현

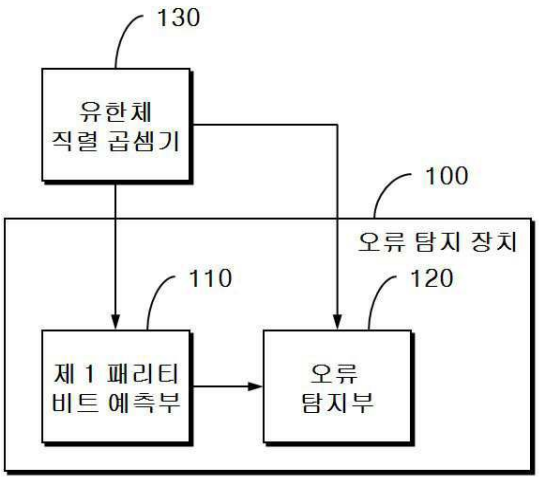
전체 청구항 수 : 총 10 항

(54) 발명의 명칭 **가우시안 정규기저를 갖는 GF(2^n) 직렬 곱셈기에 대한 오류 탐지**

(57) 요 약

본 발명은 유한체의 직렬 곱셈기에 대한 오류 탐지에 관한 것으로서, 제 1 입력의 비트 중 하나의 비트와 제 2 입력의 비트 중 하나의 비트를 이용하여 제 1 패리티 비트를 예측하는 단계, 및 상기 제 1 입력과 상기 제 2 입력의 직렬 곱셈을 통해 산출된 결과의 각 비트 값과 제 1 패리티 비트를 비교하여 오류가 있는지를 탐지하는 단계를 포함함으로써, 패리티 비트를 이용하여 직렬 곱셈기에 대한 오류주입공격에 안전할 수 있다.

대 표 도 - 도1



(72) 발명자

김창한

서울 강남구 삼성로 212, 21동 507호 (대치동, 은마아파트)

박영호

경기도 군포시 오금로 16 324-1404 (금정동, 다산주공아파트)

장남수

서울특별시 마포구 월드컵북로27길 16 (성산동)

조성민

서울특별시 성북구 화랑로41길 9 2층 201호 (장위동)

이 발명을 지원한 국가연구개발사업

과제고유번호	NIPA-2013-H0301-13-3007
부처명	미래창조과학부
연구관리전문기관	정보통신산업진흥원
연구사업명	IT융합 고급인력과정 지원사업
연구과제명	스마트그리드 기술 연구
기 여 율	1/1
주관기관	고려대학교 산학협력단
연구기간	2013.01.01 ~ 2013.12.31

명 세 서

청구범위

청구항 1

유한체의 직렬 곱셈기에 있어서,

상기 직렬 곱셈기의 제 1 입력의 비트 중 하나의 비트와 상기 직렬 곱셈기의 제 2 입력의 비트 중 하나의 비트를 이용하여 제 1 패리티 비트를 예측하는 제 1 패리티 비트 예측부; 및

상기 제 1 입력과 상기 제 2 입력의 직렬 곱셈을 통해 산출된 결과의 각 비트 값과 상기 제 1 패리티 비트를 비교하여 오류가 있는지를 탐지하는 오류 탐지부를 포함하는 오류 탐지 장치.

청구항 2

제 1 항에 있어서,

상기 제 1 패리티 비트 예측부는,

상기 제 1 입력의 비트 중 하나의 비트와 상기 제 2 입력의 비트 중 하나의 비트의 곱셈 연산을 수행하는 곱셈 연산부;

상기 곱셈 연산부의 곱셈 값과 제 1 패리티 비트 레지스터의 저장 값의 배타적 논리합 연산을 수행하는 배타적 논리합 연산부; 및

상기 배타적 논리합 연산부의 배타적 논리합 연산 값을 저장하는 제 1 패리티 비트 레지스터를 포함하고,

상기 직렬 곱셈 연산의 종료 시, 상기 제 1 패리티 비트 레지스터에 저장된 값이 상기 제 1 패리티 비트인 것을 특징으로 하는 오류 탐지 장치.

청구항 3

제 1 항에 있어서,

상기 제 1 패리티 비트 예측부는,

상기 제 1 입력과 상기 제 2 입력의 비트 중 대응하는 자리의 비트를 이용하여 상기 제 1 패리티 비트를 예측하는 것을 특징으로 하는 오류 탐지 장치.

청구항 4

제 1 항에 있어서,

상기 오류 탐지부는,

상기 직렬 곱셈을 통해 산출된 결과의 각 비트 값들과 상기 제 1 패리티 비트를 2 개씩 짝을 이루어 배타적 논리합 연산을 순차적으로 수행함으로써 최종 산출되는 배타적 논리합 연산 값으로부터 상기 오류가 있는지를 탐지하는 것을 특징으로 하는 오류 탐지 장치.

청구항 5

제 1 항에 있어서,

상기 제 1 입력과 상기 제 2 입력의 직렬 곱셈을 통해 산출된 결과 값을 이용하여 제 2 패리티 비트를 생성하는 제 2 패리티 비트 생성부를 더 포함하고,

상기 오류 탐지부는,

상기 제 1 패리티 비트와 제 2 패리티 비트를 비교하여 오류가 있는지를 탐지하는 것을 특징으로 하는 오류 탐지 장치.

청구항 6

제 5 항에 있어서,

상기 제 2 패리티 비트 생성부는,

상기 제 1 입력과 상기 제 2 입력의 직렬 곱셈을 통해 산출된 결과 값의 각 비트 값들의 배타적 논리합 연산을 통해 제 2 패리티 비트를 생성하는 것을 특징으로 하는 오류 탐지 장치.

청구항 7

유한체의 직렬 곱셈기에 있어서,

상기 직렬 곱셈기의 제 1 입력의 비트 중 하나의 비트와 상기 직렬 곱셈기의 제 2 입력의 비트 중 하나의 비트를 이용하여 제 1 패리티 비트를 예측하는 단계; 및

상기 제 1 입력과 상기 제 2 입력의 직렬 곱셈을 통해 산출된 결과의 각 비트 값과 제 1 패리티 비트를 비교하여 오류가 있는지를 탐지하는 단계를 포함하는 오류 탐지 방법.

청구항 8

제 7 항에 있어서,

상기 제 1 패리티 비트를 예측하는 단계는,

상기 제 1 입력의 비트 중 하나의 비트와 상기 제 2 입력의 비트 중 하나의 비트의 곱셈 연산을 수행하는 단계;

상기 곱셈 연산의 곱셈 값과 제 1 패리티 비트 레지스터의 저장 값의 배타적 논리합 연산을 수행하는 단계; 및

상기 배타적 논리합 연산의 배타적 논리합 연산 값을 제 1 패리티 비트 레지스터에 저장하는 단계를 포함하고,

상기 직렬 곱셈 연산의 종료 시, 상기 제 1 패리티 비트 레지스터에 저장된 값이 상기 제 1 패리티 비트인 것을 특징으로 하는 오류 탐지 방법.

청구항 9

제 7 항에 있어서,

상기 오류가 있는지를 탐지하는 단계는,

상기 직렬 곱셈을 통해 산출된 결과의 각 비트 값들과 상기 제 1 패리티 비트를 2 개씩 짝을 이루어 배타적 논리합 연산을 순차적으로 수행함으로써 최종 산출되는 배타적 논리합 연산 값으로부터 상기 오류가 있는지를 탐지하는 것을 특징으로 하는 오류 탐지 방법.

청구항 10

제 7 항 내지 제 9 항 중 어느 한 항의 방법을 컴퓨터에서 실행시키기 위한 프로그램을 기록한 컴퓨터로 읽을 수 있는 기록매체.

발명의 설명

기 술 분 야

[0001] 본 발명은 직렬 곱셈기에 대한 오류 탐지에 관한 것으로서, 더욱 상세하게는 패리티 비트를 이용하여 직렬 곱셈기에 대한 오류주입공격에 안전한 오류 탐지 방법 및 그 장치에 관한 것이다.

배 경 기 술

[0002] 유한체 $GF(2^n)$의 연산은 덧셈과 곱셈, 제곱으로 구성되어 있다. 덧셈과 제곱은 간단하게 구성되는 반면 곱셈은 유한체의 표현 방법에 따라 다양하게 구성된다. 유한체의 표현은 어떤 기저를 사용하여 표현하느냐에 따라 구분하는데, 대표적으로 다항식기저와 정규기저를 사용하고 있다. 즉, $GF(2^n)$을 구성하는 GF(2) 위의 기약다항식을

$f(x)=x^n+a_{n-1}x^{n-1}+\cdots+a_1x+1$ 이라 할 때, f(x)의 근을 α라 하면, $GF(2^n)$의 원소 $A=a_0+a_1\alpha+\cdots+a_{n-1}\alpha^{n-1},\ a_i\in GF(2)$ 와 같이 표현하는 것이 다항식기저를 이용한 표현이다. 이때 A를 벡터로 표시하면 $A=(a_0,a_1,\cdots,a_{n-1})$ 이다. 반면에 정규기저를 이용한 표현은 $\beta\in GF(2^n)$의 conjugate들이 GF(2)위에서 $GF(2^n)$의 기저가 될 때, 이 기저를 사용하는 것이다. 즉, $\{\beta,\beta^2,\cdots,\beta^{2^{n-1}}\}$ 이 기저가 되어 $GF(2^n)$의 원소 $A=a_0\beta+a_1\beta^2+\cdots+a_{n-1}\beta^{2^{n-1}},a_i\in GF(2)$ 와 같이 표현하는 것이다. 이때 β를 정규기저 생성자라 한다. 또한 A를 벡터로 표시하면 $A=(a_0,a_1,\cdots,a_{n-1})$ 이다.

[0003] 정규기저를 이용할 경우 제곱 연산은 계수들의 위치이동으로 표시되므로 하드웨어적으로는 연산이 없는 장점이 있다. 정규기저를 구성하는 방법은 여러 방법이 있으나 다음 [정리 1]에 제시된 Gaussian 정규기저 방법이 가장 널리 쓰인다.

[0004] [정리 1] n,k는 nk+1이 2 보다 큰 소수인 조건을 만족하는 양의 정수이고, α는 $GF(2^{nk})$에서 nk+1의 원시근이라 하자. Z_{nk+1}에서 2의 위수(order)를 e라 할 때 $\gcd(nk/e,n)=1$ 라 하자. 그러면 Z_{nk+1}에서 k의 원시근 τ에 대하여 $\beta=\sum_{i=0}^{k-1}\alpha^{\tau^i}$ 는 GF(2) 위에서 $GF(2^n)$의 정규기저 생성자이다.

[0005] [참고] 양의 정수 n에 대하여 정리 1을 만족하는 k가 존재할 때 $GF(2^n)$은 GF(2) 위에서 타입 k인 가우시안 정규기저를 갖는다고 한다.

[0006] 정규기저를 이용한 곱셈의 경우

[0007] $$C=A\cdot B,\ A=\sum_{i=0}^{n-1}a_i\beta^{2^i},\ B=\sum_{i=0}^{n-1}b_i\beta^{2^i},\ C=\sum_{i=0}^{n-1}c_i\beta^{2^i}$$

[0008] 라 하면

[0009] $$c_i=(a_0,a_1,\cdots,a_{n-1})\cdot M\cdot(b_0,b_1,\cdots,b_{n-1})^T,\quad M=(m_{ij}):n\times n$$

[0010] 로 표시되고, 이때 행렬 M의 1의 개수(Weight)를 기저 $N=\{\beta,\beta^2,\cdots,\beta^{2^{n-1}}\}$ 의 곱셈에 대한 복잡도(Complexity) C_N이라 한다.

[0011] 모든 정규기저 N의 복잡도는 $C_N\geq 2n-1$ 이다. 특히 $C_N=2n-1$ 인 경우 N을 최적정규기저라 한다. 위 정리에서 k=1,2인 경우 최적정규기저가 되며 각각을 타입 I, II의 최적정규기저라 한다. 위 정리에서 k가 짝수인 경우 $C_N\leq kn-1$ 이다.

[0012] 본 발명과 관련된 선행기술로는 '타입 케이 가우시안 정규기저를 갖는 유한체의 직렬 곱셈연산기(한국공개특허 10-2007-0102912)' 등이 있다.

발명의 내용

해결하려는 과제

[0013] 본 발명이 해결하고자 하는 첫 번째 과제는 패리티 비트를 이용하여 직렬 곱셈기에 대한 오류주입공격에 안전한 오류 탐지 장치를 제공하는 것이다.

[0014] 본 발명이 해결하고자 하는 두 번째 과제는 패리티 비트를 이용하여 직렬 곱셈기에 대한 오류주입공격에 안전한

오류 탐지 방법을 제공하는 것이다.

과제의 해결 수단

[0015] 본 발명은 상기 첫 번째 과제를 달성하기 위하여, 유한체의 직렬 곱셈기에 있어서, 제 1 입력의 비트 중 하나의 비트와 제 2 입력의 비트 중 하나의 비트를 이용하여 제 1 패리티 비트를 예측하는 제 1 패리티 비트 예측부; 및 상기 제 1 입력과 상기 제 2 입력의 직렬 곱셈을 통해 산출된 결과의 각 비트 값과 상기 제 1 패리티 비트를 비교하여 오류가 있는지를 탐지하는 오류 탐지부를 포함하는 오류 탐지 장치를 제공한다.

[0016] 본 발명의 실시예에 의하면, 상기 제 1 패리티 비트 예측부는, 상기 제 1 입력의 비트 중 하나의 비트와 상기 제 2 입력의 비트 중 하나의 비트의 곱셈 연산을 수행하는 곱셈 연산부; 상기 곱셈 연산부의 곱셈 값과 제 1 패리티 비트 레지스터의 저장 값의 배타적 논리합 연산을 수행하는 배타적 논리합 연산부; 및 상기 배타적 논리합 연산부의 배타적 논리합 연산 값을 저장하는 제 1 패리티 비트 레지스터를 포함하고, 상기 직렬 곱셈 연산의 종료 시, 상기 제 1 패리티 비트 레지스터에 저장된 값이 상기 제 1 패리티 비트인 것을 특징으로 하는 오류 탐지 장치일 수 있다.

[0017] 본 발명의 실시예에 의하면, 상기 제 1 패리티 비트 예측부는, 상기 제 1 입력과 상기 제 2 입력의 비트 중 대응하는 자리의 비트를 이용하여 상기 제 1 패리티 비트를 예측하는 것을 특징으로 하는 오류 탐지 장치일 수 있다.

[0018] 본 발명의 실시예에 의하면, 상기 오류 탐지부는, 상기 직렬 곱셈을 통해 산출된 결과의 각 비트 값들과 상기 제 1 패리티 비트를 2 개씩 짝을 이루어 배타적 논리합 연산을 순차적으로 수행함으로써 최종 산출되는 배타적 논리합 연산 값으로부터 상기 오류가 있는지를 탐지하는 것을 특징으로 하는 오류 탐지 장치일 수 있다.

[0019] 본 발명의 실시예에 의하면, 상기 제 1 입력과 상기 제 2 입력의 직렬 곱셈을 통해 산출된 결과 값을 이용하여 제 2 패리티 비트를 생성하는 제 2 패리티 비트 생성부를 더 포함하고, 상기 오류 탐지부는, 상기 제 1 패리티 비트와 제 2 패리티 비트를 비교하여 오류가 있는지를 탐지하는 것을 특징으로 하는 오류 탐지 장치일 수 있다.

[0020] 본 발명은 상기 두 번째 과제를 달성하기 위하여, 유한체의 직렬 곱셈기에 있어서, 제 1 입력의 비트 중 하나의 비트와 제 2 입력의 비트 중 하나의 비트를 이용하여 제 1 패리티 비트를 예측하는 단계; 및 상기 제 1 입력과 상기 제 2 입력의 직렬 곱셈을 통해 산출된 결과의 각 비트 값과 제 1 패리티 비트를 비교하여 오류가 있는지를 탐지하는 단계를 포함하는 오류 탐지 방법을 제공한다.

[0021] 본 발명의 실시예에 의하면, 상기 제 1 패리티 비트를 예측하는 단계는, 상기 제 1 입력의 비트 중 하나의 비트와 상기 제 2 입력의 비트 중 하나의 비트의 곱셈 연산을 수행하는 단계; 상기 곱셈 연산의 곱셈 값과 제 1 패리티 비트 레지스터의 저장 값의 배타적 논리합 연산을 수행하는 단계; 및 상기 배타적 논리합 연산의 배타적 논리합 연산 값을 제 1 패리티 비트 레지스터에 저장하는 단계를 포함하고, 상기 직렬 곱셈 연산의 종료 시, 상기 제 1 패리티 비트 레지스터에 저장된 값이 상기 제 1 패리티 비트이고, 상기 오류가 있는지를 탐지하는 단계는, 상기 직렬 곱셈을 통해 산출된 결과의 각 비트 값들과 상기 제 1 패리티 비트를 2 개씩 짝을 이루어 배타적 논리합 연산을 순차적으로 수행함으로써 최종 산출되는 배타적 논리합 연산 값으로부터 상기 오류가 있는지를 탐지하는 것을 특징으로 하는 오류 탐지 방법일 수 있다.

발명의 효과

[0022] 본 발명에 따르면, n-bit 출력의 직렬 곱셈기의 경우 1 개의 AND gate, n+1 개의 XOR gate, 1 개의 1-bit Register를 추가하여 오류 주입 공격에 대비할 수 있다. 이때, 시간복잡도는 타입 IV 정규 기저를 갖는 n=163인 경우 기존의 직렬곱셈기에 비하여 1.23%만이 증가한다.

도면의 간단한 설명

[0023] 도 1은 본 발명의 일 실시예에 따른 유한체의 직렬 곱셈기에 대한 오류 탐지 장치의 블록도이다.

도 2는 본 발명의 실시예에 따른 유한체의 직렬 곱셈기에 대한 오류 탐지 장치의 블록도이다.

도 3은 본 발명의 실시예에 따른 유한체의 직렬 곱셈기에 대한 오류 탐지 과정을 도시한 것이다.

도 4는 유한체의 직렬 곱셈기이고, 도 5는 본 발명의 실시예에 따른 오류 탐지를 유한체의 직렬 곱셈기에 적용한 것이다.

도 6은 본 발명의 일 실시예에 따른 유한체의 직렬 곱셈기에 대한 오류 탐지 방법의 흐름도이다.

도 7은 본 발명의 실시예에 따른 유한체의 직렬 곱셈기에 대한 오류 탐지 방법의 흐름도이다.

발명을 실시하기 위한 구체적인 내용

[0024] 본 발명에 관한 구체적인 내용의 설명에 앞서 이해의 편의를 위해 본 발명이 해결하고자 하는 과제의 해결 방안의 개요 혹은 기술적 사상의 핵심을 우선 제시한다.

[0025] 본 발명의 일 실시예에 따른 인지 유한체의 직렬 곱셈기에 대한 오류 탐지 방법은 제 1 입력의 비트 중 하나의 비트와 제 2 입력의 비트 중 하나의 비트를 이용하여 제 1 패리티 비트를 예측하는 단계, 및 상기 제 1 입력과 상기 제 2 입력의 직렬 곱셈을 통해 산출된 결과의 각 비트 값과 제 1 패리티 비트를 비교하여 오류가 있는지를 탐지하는 단계를 포함한다.

[0026] 이하 첨부된 도면을 참조하여 본 발명이 속하는 기술 분야에서 통상의 지식을 가진 자가 본 발명을 용이하게 실시할 수 있는 실시 예를 상세히 설명한다. 그러나 이들 실시예는 본 발명을 보다 구체적으로 설명하기 위한 것으로, 본 발명의 범위가 이에 의하여 제한되지 않는다는 것은 당업계의 통상의 지식을 가진 자에게 자명할 것이다.

[0027] 본 발명이 해결하고자 하는 과제의 해결 방안을 명확하게 하기 위한 발명의 구성을 본 발명의 바람직한 실시예에 근거하여 첨부 도면을 참조하여 상세히 설명하되, 도면의 구성요소들에 참조번호를 부여함에 있어서 동일 구성요소에 대해서는 비록 다른 도면상에 있더라도 동일 참조번호를 부여하였으며 당해 도면에 대한 설명시 필요한 경우 다른 도면의 구성요소를 인용할 수 있음을 미리 밝혀둔다. 아울러 본 발명의 바람직한 실시 예에 대한 동작 원리를 상세하게 설명함에 있어 본 발명과 관련된 공지 기능 혹은 구성에 대한 구체적인 설명 그리고 그 이외의 제반 사항이 본 발명의 요지를 불필요하게 흐릴 수 있다고 판단되는 경우, 그 상세한 설명을 생략한다.

[0028] 도 1은 본 발명의 일 실시예에 따른 유한체의 직렬 곱셈기에 대한 오류 탐지 장치의 블록도이고, 도 2는 본 발명의 실시예에 따른 유한체의 직렬 곱셈기에 대한 오류 탐지 장치의 블록도이다.

[0029] 정규기저로 표현된 $A,B\in GF(2^n)$ 의 곱셈 결과를 $C=A\cdot B$ 라 하고, 각각의 패리티 비트(parity bit)를 다음과 같이 나타낼 수 있다.

[0030] $$A_p=\sum_{i=0}^{n-1}a_i,\ B_p=\sum_{i=0}^{n-1}b_i,\ C_p=\sum_{i=0}^{n-1}c_i,\ A_p,B_p,C_p\in GF(2)$$

[0031] 예상되는 패리티 비트를 제 1 패리티 비트, C_P(pred)하 하고, A와 B의 곱셈 연산에서 얻은 C의 패리티 비트를 제 2 패리티 비트, C_P(Act)라 하면, 오류 없이 정상적인 곱셈이 이루어지면 C_P(Act)=C_P(pred)이다. 직렬 곱셈기의 연산에 오류가 있는지를 탐지하는 과정은 도 3과 같이 나타낼 수 있다. CUT(circuit under test)은 곱셈 과정이고, PP(parity prediction)는 C_P(pred)를 구하는 과정이며, PG(parity generation)는 직렬 곱셈기의 경우 곱셈의 결과로부터 C_P(Act)를 구하는 과정이다.

[0032] 상기 과정을 직렬 곱셈기에 적용하기 위하여, 본 발명의 일 실시예에 따른 유한체의 직렬 곱셈기에 대한 오류 탐지 장치(100)는 제 1 패리티 비트 예측부(110) 및 오류 탐지부(120)로 구성된다.

[0033] 제 1 패리티 비트 예측부(110)는 제 1 입력의 비트 중 하나의 비트와 제 2 입력의 비트 중 하나의 비트를 이용하여 제 1 패리티 비트를 예측한다.

[0034] 보다 구체적으로, 유한체 직렬 곱셈기(130)에 입력되는 제 1 입력의 비트 중 하나의 비트와 제 2 입력의 비트 중 하나의 비트를 이용하여 제 1 패리티 비트를 예측한다. 제 1 패리티 비트를 예측하는데 이용되는 상기 제 1 입력의 비트와 상기 제 2 입력의 비트는 서로 대응하는 자리의 비트이다. 즉, 제 1 입력의 비트가 a1 내지 a5로 구성되고, 제 2 입력의 비트가 b1 내지 b5로 구성되는 경우, a1과 b1을 이용하거나 a5와 b5를 이용하는 것과 같이 대응하는 자리의 비트를 이용한다. 상기 유한체 직렬 곱셈기의 각 비트는 기준클락에 따라 상위비트로 시프

트(shift)되는바, 어느 자리의 비트를 이용하더라도 제 1 패리티 비트의 결과는 동일하다.

[0035] 제 1 패리티 비트를 산출하는데, 도 2와 같이, 제 1 패리티 비트 예측부(110)는 곱셈 연산부(111), 배타적 논리합 연산부(112), 및 제 1 패리티 비트 레지스터(113)로 구성될 수 있다. 즉, 상기 직렬 곱셈기의 출력이 n-bit인 경우, 1 개의 곱셈 연산부, 1 개의 배타적 논리합 연산부, 및 1 개의 1-bit 레지스터를 이용하여 제 1 패리티 비트를 산출할 수 있다.

[0036] 곱셈 연산부(111)는 상기 제 1 입력의 비트 중 하나의 비트와 상기 제 2 입력의 비트 중 하나의 비트의 곱셈 연산을 수행한다.

[0037] 보다 구체적으로, 상기 제 1 입력의 비트와 제 1 입력의 비트에 대응하는 제 2 입력의 비트의 곱셈 연산을 수행한다. 곱셈 연산부(111)는 AND 게이트이고, 입력되는 2 개의 입력의 곱셈 연산을 수행한다.

[0038] 배타적 논리합 연산부(112)는 곱셈 연산부(111)의 곱셈 값과 제 1 패리티 비트 레지스터(113)의 저장 값의 배타적 논리합 연산을 수행한다.

[0039] 보다 구체적으로, 동일한 비트 자리를 갖는 제 1 입력의 비트와 제 2 입력의 비트의 곱셈 연산 값들을 순차적으로 배타적 논리합 연산을 수행하기 위하여, 곱셈 연산부(111)의 곱셈 값과 제 1 패리티 비트 레지스터(113)의 저장 값의 배타적 논리합 연산을 수행한다. 제 1 패리티 비트 레지스터(113)는 이전까지의 배타적 논리합 연산 결과 값을 저장하고 있는바, 이에 곱셈 연산부(111)는 이번 주기(클락)에 곱셈한 곱셈 값을 상기 제 1 패리티 비트 레지스터(113)가 저장하고 있는 저장 값과 배타적 논리합 연산을 수행함으로써 제 1 패리티 비트를 산출할 수 있다. 배타적 논리합 연산부(112)는 XOR 게이트이고, 입력되는 2 개의 입력의 배타적 논리합 연산을 수행한다.

[0040] 제 1 패리티 비트 레지스터(113)는 배타적 논리합 연산부(112)의 배타적 논리합 연산 값을 저장한다.

[0041] 보다 구체적으로, 배타적 논리합 연산부(112)에서 순차적으로 수행되는 제 1 입력의 비트와 제 2 입력의 비트의 곱셈 연산 값들에 대한 배타적 논리합 연산 값을 저장한다. 상기 저장된 배타적 논리합 연산 값은 배타적 논리합 연산부(112)에서 곱셈 연산부(111)의 곱셈 값과 배타적 논리합 연산을 수행하는데 이용된다. 유한체 직렬 곱셈기(130)의 연산이 종료되는 시점에 제 1 패리티 비트 레지스터(113)에 저장된 값이 제 1 패리티 비트이다.

[0042] 오류 탐지부(120)는 상기 제 1 입력과 상기 제 2 입력의 직렬 곱셈을 통해 산출된 결과의 각 비트 값과 상기 제 1 패리티 비트를 비교하여 오류가 있는지를 탐지한다.

[0043] 보다 구체적으로, 유한체 직렬 곱셈기(130)의 직렬 곱셈을 통해 산출되는 결과의 각 비트 값과 상기 제 1 패리티 비트를 비교하여 오류가 있는지를 탐지한다.

[0044] 상기 직렬 곱셈을 통해 산출되는 결과의 각 비트 값들로 생성되는 패리티 비트와 제 1 패리티 비트가 동일한지 여부로 오류가 있는지를 판단할 수 있다. 이를 위하여, 상기 제 1 입력과 상기 제 2 입력의 직렬 곱셈을 통해 산출된 결과 값을 이용하여 제 2 패리티 비트를 생성하는 제 2 패리티 비트 생성부를 더 포함할 수 있다. 상기 제 2 패리티 비트 생성부는 상기 제 1 입력과 상기 제 2 입력의 직렬 곱셈을 통해 산출된 결과 값의 각 비트 값들의 배타적 논리합 연산을 통해 제 2 패리티 비트를 생성할 수 있다. 즉, 제 2 패리티 비트 생성부에서 제 2 패리티 비트를 생성하고, 오류 탐지부(120)가 제 1 패리티 비트와 제 2 패리티 비트가 동일한지 비교하여 오류가 있는지를 탐지할 수 있다. 상기 제 1 입력과 상기 제 2 입력의 직렬 곱셈을 통해 산출된 결과 값들을 구성하는 각 비트 값들의 배타적 논리합 연산을 수행하면 제 2 패리티 비트를 생성할 수 있다. 제 1 패리티 비트 예측부(110)가 예측한 제 1 패리티 비트와 동일한지를 비교함으로써 오류가 있는지를 탐지할 수 있다.

[0045] 반면, 제 2 패리티 비트 생성부가 제 2 패리티 비트를 생성한 후, 제 1 패리티 비트와 비교하는 것과 다르게, 상기 오류가 있는지를 판단하기 위하여, 오류 탐지부(120)는 상기 직렬 곱셈을 통해 산출된 결과의 각 비트 값들과 상기 제 1 패리티 비트를 2 개씩 짝을 이루어 배타적 논리합 연산을 순차적으로 수행함으로써 최종 산출되는 배타적 논리합 연산 값으로부터 상기 오류가 있는지를 탐지할 수 있다. 즉, 직렬 곱셈의 결과 비트들로부터 제 2 패리티 비트를 생성하여 제 1 패리티 비트와 비교하는 것이 아니고, 상기 직렬 곱셈의 결과 비트들을 제 1 패리티 비트와 바로 배타적 논리합 연산을 수행하여 오류가 있는지 탐지할 수 있다. 이를 위하여, 상기 직렬 곱셈을 통해 산출된 결과의 각 비트 값들과 상기 제 1 패리티 비트를 2 개씩 짝을 이루어 배타적 논리합 연산을 순차적으로 수행한다. 2 개씩 짝을 이루어 수행한 배타적 논리합 연산의 결과들에 다시 2 개씩 짝을 이루어 배타적 논리합 연산을 수행하는 것을 반복하여 최종 산출되는 배타적 논리합 연산 값이 0인지 1인지로 오류가 존재하는지를 판단할 수 있다. 즉, 최종 산출되는 배타적 논리합 연산 값이 0인 경우, 오류가 존재하지 않는 것으

로 판단하고, 최종 산출되는 배타적 논리합 연산 값이 1인 경우, 오류가 존재하는 것으로 판단한다. 배타적 논리합 연산을 수행할 입력이 홀수인 경우, 2 개씩 짝을 이루고, 나머지 한 개는 다음 단계에서 배타적 논리합 연산을 수행한다. 상기와 같이, 상기 직렬 곱셈을 통해 산출된 결과의 각 비트 값들과 상기 제 1 패리티 비트를 2 개씩 짝을 이루어 배타적 논리합 연산을 순차적으로 수행함으로써 최종 산출되는 배타적 논리합 연산 값으로부터 상기 오류가 있는지를 탐지함으로써, 제 2 패리티 비트를 생성한 후 제 2 패리티 비트와 제 1 패리티 비트를 비교하는 것보다, 과정이 간단하며, 그에 따른 면적과 시간복잡도를 향상시킬 수 있다.

[0046] 본 발명의 일 실시예에 따른 오류 탐지 장치(100)가 유한체 직렬 곱셈기(130)에 적용가능함은 도 5에 대한 상세한 설명에서 자세히 다루도록 한다.

[0047] 도 4는 유한체의 직렬 곱셈기이고, 도 5는 본 발명의 실시예에 따른 오류 탐지를 유한체의 직렬 곱셈기에 적용한 것이다.

[0048] 본 발명의 실시예에 따른 오류 탐지를 유한체의 직렬 곱셈기에 적용하면, 도 5와 같이, n-bit 출력의 직렬 곱셈기의 경우 1 개의 AND gate, n+1 개의 XOR gate, 1 개의 1-bit Register만을 추가하여 오류 주입 공격에 대비할 수 있다. 제 1 패리티 비트를 예측하기 위한 제 1 패리티 비트 예측부(110)의 곱셈 연산부(111)에 1 개의 AND 게이트, 배타적 논리합 연산부(112)에 1 개의 XOR 게이트, 및 제 1 패리티 비트 레지스터(113)에 1 개의 1-bit 레지스터가 필요하고, 오류 탐지부(120)에 n 개의 XOR 게이트가 필요하다. 이때, 시간복잡도는 타입 IV 정규 기저를 갖는 n=163인 경우 도 4의 유한체 직렬 곱셈기에 비하여 1.23%만이 증가한다.

[0049] 이하, $GF(2^n)$이 타입 k(even)의 가우시안 정규기저를 갖는, n-bit 출력의 직렬 곱셈기의 경우 1 개의 AND gate, 1 개의 XOR gate, 1 개의 1-bit Register만을 이용하여 제 1 패리티 비트를 예측할 수 있음을 다루도록 한다.

[0050] n+1을 소수이고, α를 $x^n+x^{n-1}+\cdots+1$ 의 완전 확창체(Splitting field) F의 n+1의 원시근((n+1)-th primitive root)라 하면, F는 $GF(2^n)$의 부분체이다. 즉,

[0051] $\overline{A}=\sum_{i=0}^{n}A_i\alpha^i,\ \overline{B}=\sum_{i=0}^{n}B_i\alpha^i,\ A_i,B_i\in GF(2)$ 라 하면 $\overline{A},\overline{B}\in GF(2^n)$ 이다.

[0052] $\overline{C}=\overline{A}\cdot\overline{B}=\sum_{k=0}^{n}C_k\alpha^k$ 라 하면,

[0053] $$(C_0\ C_1\cdots C_n)^T=\begin{pmatrix}A_0 & A_n & \cdots & A_2 & A_1\\ A_1 & A_0 & \cdots & A_3 & A_2\\ & & & & \\ A_n & A_{n-1} & \cdots & A_1 & A_0\end{pmatrix}\begin{pmatrix}B_0\\ B_1\\ \\ B_n\end{pmatrix}$$ 이다.

[0054] 유한체 원소 A를 정규기저로 표현한 것과 같이 parity bit를 $\overline{A}$ 에 대하여 다음과 같이 정리할 수 있다.

[0055] [정리 1]

[0056] $\overline{A}\in GF(2^n)$ 인 경우, $\overline{A_P}=\sum_{i=0}^{n}A_i,\ \overline{A_P}\in GF(2)$ 이다.

[0057] 이를 정리하면 다음과 같이 정리할 수 있다.

[0058] [정리 2]

[0059] $\overline{C_P}=\overline{A_P}\cdot\overline{B_P}$

[0060] 정리 1의 가정을 만족하는 n,k,τ의 경우, $Z_{2^k+1}^*$의 원소는 $r = \tau^{j2^i},\ 0 \le j \le k-1, 0 \le i \le n-1,$ 로 유일하게 표현된다.

[0061] **타입 I**

[0062] 타입 I인 경우는 n+1이 소수이고 2가 Z_{n+1}^*의 생성자이므로 $Z_{n+1}^* = \{1,2,\cdots,n\} = \{2^i \mid 0 \le i \le n-1\}$ 이다. 따라서 GF(2) 위에서 $GF(2^n)$의 정규기저 생성자 β=α이므로 정규기저 N은 다음과 같다.

[0063] $N = \{\alpha, \alpha^2, \alpha^{2^2}, \cdots, \alpha^{2^{n-1}}\},\ \alpha \in GF(2^n) : (n+1)th\ primitive\ root\ of\ unity.$

[0064] 또한, 집합적으로는 $Z_{n+1}^* = <2>$ 이므로

[0065] $\{\alpha, \alpha^2, \alpha^{2^2}, \cdots, \alpha^{2^{n-1}}\} = \{\alpha, \alpha^2, \cdots, \alpha^n\}$

[0066] 이다. 그러므로 $GF(2^n)$의 원소 A의 계수 위치를 정리하면 $A = \sum_{i=1}^{n} a_i \alpha^i$ 이고 A의 parity bit는 동일하게 표시된다. 또한, $\overline{A} = \sum_{i=0}^{n} A_i \alpha^i$ 로 표시하면 $A_0 = 0$ 이고 $A_i = a_i,\ 1 \le i \le n$ 이다.

[0067] A, B를 $\overline{A}, \overline{B}$ 로 표시하고 A와 B의 곱을 표시한 결과를 $\overline{C} = \sum_{i=0}^{n} C_i \alpha^i$ 라 하면,

[0068] $$(C_0 C_1 \cdots C_n)^T = \begin{pmatrix} 0 & A_n & \cdots & A_2 & A_1 \\ A_1 & 0 & \cdots & A_3 & A_2 \\ & & & & \\ A_n & A_{n-1} & \cdots & A_1 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ B_1 \\ \\ B_n \end{pmatrix}$$

[0069] 이다. 그리고 $1 = \alpha + \alpha^2 + \cdots + \alpha^n$ 이므로 실제로

[0070] $$C = A \cdot B = \sum_{i=1}^{n} (C_0 + C_i)\alpha^i$$

[0071] 이다. 이 경우 $\overline{A_F} = A_F, \overline{B_F} = B_F$, n이 짝수,

[0072] $$C_0 = \sum_{i=1}^{n} A_{n+1-i} B_i = \sum_{i=1}^{n} a_{n+1-i} b_i$$

[0073] 이므로

[0074] $$C_F = \sum_{i=1}^{n} (C_0 + C_i) = \sum_{i=0}^{n} C_i + \sum_{i=2}^{n} C_0 = \overline{C_F} + C_0 = \overline{A_F B_F} + C_0 = A_F B_F + \sum_{i=1}^{n} a_{n+1-i} b_i$$

[0075] [정리 3] $A, B \in GF(2^n)$ 에 대해 $C = A \cdot B$ 라 하면

[0076] $$C_p(mod) = A_p \cdot B_p + \sum_{i=1}^{n} a_{n+1-i} b_i$$ 이다.

[0077] **타입 II**

[0078] 타입 II인 경우 Z_{2n+1}에서 $\tau = -1$이므로 $\beta = \alpha + \alpha^{-1}$ 이다. 그리고 보조정리 1에 의하여 $Z_{2n+1} = \{\pm 2^i \mid 0 \le i \le n-1\}$ 이므로 타입 I의 경우와 같이 GF(2^n)의 정규기저는 집합적으로 다음과 같다.

[0079] $$N = \{\beta, \beta^2, \beta^{2^2}, \cdots, \beta^{2^{n-1}}\}$$
$$= \{\alpha + \alpha^{-1}, \alpha^2 + \alpha^{-2}, \cdots, \alpha^n + \alpha^{-n}\}.$$

[0080] 따라서 GF(2^n)의 원소 A의 계수 위치를 다시 정리하면

[0081] $$A = \sum_{i=1}^{n} a_i (\alpha^i + \alpha^{-i})$$

[0082] 이다. 또한, A를 GF(2^{2n})의 원소 $\overline{A}$ 로 표시하면 다음과 같다.

[0083] $$\overline{A} = \sum_{k=1}^{2n} A_k \alpha^k$$

[0084] 이 경우, $1 \le i \le n$ 에 대해 $A_i = A_{2n+1-i} = a_i$이다. 그러므로 정리 3의 타입 I 경우의 곱의 parity check bit를 적용하면 다음 수학식 1과 같다.

[0085] [수학식 1]

[0086] $$\overline{C_p} = \overline{A_p} \cdot \overline{B_p} + \sum_{k=1}^{2n} A_{2n+1-k} B_k$$

[0087] [정리 4] n이 타입 II 최적 정규기저의 경우 $C = A \cdot B$ 의 parity bit는 다음과 같다.

[0088] $$C_p(mod) = \sum_{i=1}^{n} a_i b_i$$

[0089] [증명] GF(2^n)의 원소 $A = \sum_{i=1}^{n} a_i (\alpha^i + \alpha^{-i})$ 를 GF(2^{2n})의 원소로 표시하면

[0090] $$\overline{A} = \sum_{k=1}^{2n} A_k \alpha^k, \quad A_i = A_{2n+1-i} = a_i, \quad 1 \le i \le n$$

[0091] 이므로 정수로 $\overline{A_p} = 2A_p$이고, 수학식 1과 $1 \le i \le n$ 에 대하여 $A_i = A_{2n+1-i} = a_i$, $B_i = B_{2n+1-i} = b_i$이므로 정수로 $2C_p = 2A_p \cdot 2B_p + 2\sum_{i=1}^{n} a_i b_i$ 이다. 그러므로

$C_p(mod) = \sum_{i=1}^{n} a_i b_i$ 이다.

[0092] **타입 k, $k \geq 4$, k는 짝수인 경우**

[0093] 타입 k인 경우 [정리 1]에 의하여 $\beta = \sum_{i=0}^{k-1} \alpha^i$ 가 정규기저 생성자이다.

[0094] [정리 5] $A = \sum_{i=1}^{n} a_i \beta^{i-1} \in GF(2^n)$ 라 하자. A를 $\overline{A} = \sum_{i=1}^{nk} A_i \alpha^i \in GF(2^{nk})$ 의 원소로 표시하면, $1 \leq j \leq nk$ 인 j에 대하여 $j=2^i \tau^l$을 만족하는 $i \in \{0, \cdots, n-1\}$, $l \in \{0, \cdots, k-1\}$ 이 존재하면 $A_j = a_{i+1}$ 이다.

[0095] [증명]

[0096] $A = \sum_{i=1}^{n} a_i \beta^{i-1} = \sum_{i=1}^{n} a_i \sum_{l=0}^{k-1} \alpha^{i \tau^l} = \sum_{i=0}^{n-1} \sum_{l=0}^{k-1} a_{i+1} \alpha^{2^i \tau^l}$ 이고 [보조정리 1]에 의하여

[0097] $Z_{nk+1} = \{\tau^l 2^i \mid 0 \leq l \leq k-1,\ 0 \leq i \leq n-1\}$

[0098] 이므로 고정된 $0 \leq i \leq n-1$ 에 대하여 $j=2^i \tau^l, 0 \leq l \leq k-1$ 인 경우 $A_j=a_{i+1}$이다.

[0099] k가 짝수이므로 $\tau^{nk/2} = -1 \in Z_{nk+1}$ 이다. 위의 정리에 의하여 $\overline{A}$의 계수는 A의 각 계수가 k번 반복되는 것을 알 수 있다. 즉, 정수로 $\overline{A_F} = kA_F$ 이다.

[0100] [정리 6] n이 타입 k인 경우 $C = AB \in GF(2^n)$ 인 경우 parity bit는 $C_p(mod) = \sum_{i=1}^{n} a_i b_i$ 이다.

[0101] [증명] A, B, C를

[0102] $\overline{A} = \sum_{i=1}^{nk} A_i \alpha^i, \overline{B} = \sum_{i=1}^{nk} B_i \alpha^i, \overline{C} = \sum_{i=1}^{nk} C_i \alpha^i$

[0103] 로 표시하면 [정리 3]의 타입 I 경우의 Parity bit를 적용하면 다음과 같다.

[0104] $\overline{C_p} = \overline{A_F} \cdot \overline{B_F} + \sum_{j=1}^{nk} A_{nk+1-j} B_j$

[0105] 그리고 정수로 $\overline{A_F} = kA_F$ 이고 $j = 2^i \tau^l$, $1 \leq j \leq nk$, $0 \leq i \leq n-1, 0 \leq l \leq k-1$ 인 경우

[0106] $nk+1-j = \tau^{nk/2} 2^i \tau^l = 2^i \tau^{(nk/2)+l} \bmod nk+1$

[0107] 이고 $A_i = A_{qk+1-i}$이므로

[0108] $$\sum_{i=1}^{qk} A_{qk+1-i}B_i = \sum_{i=1}^{qk} A_iB_i$$

[0109] 이다. 따라서 정수로 보면

[0110] $$kC_P = \overline{C_P} = \overline{A_P} \cdot \overline{B_P} + \sum_{i=1}^{qk} A_{qk+1-i}B_i$$
$$= kA_P \cdot kB_P + k\sum_{i=1}^{q} a_ib_i$$

[0111] 이다. 그러므로 $C_P(pred) = \sum_{i=1}^{q} a_ib_i$ 이다.

[0112] 타입 k (even) 가우시안 정규기저를 갖는 유한체 GF(2^n)의곱셈 오류 탐지를 위하여 $\sum_{i=1}^{q} a_ib_i$ 를 계산하는 것이 필요하다. 그리고 곱셈기 결과 값과 비교하기 위한 C_P(Act) 계산이 필요하고 마지막으로 C_P(Act)와 C_P(pred)의 XOR 연산이 필요하다. 구체적으로 보면 $\sum_{i=1}^{q} a_ib_i$ 계산은 곱셈기가 연산하는 동안 1 번의 AND와 XOR로 구성된다. C_P(Act)의 계산은 곱셈이 끝난 후에 이루어진다. GF(2^5)의 구체적인 예는 도 4와 같다. 도 4는 곱셈기이고, 도 5는 오류 탐지가 포함된 곱셈기이다. GF(2^5)의 경우는 타입 II 최적 정규기저를 갖는 유한체로 전체 gate(AND+XOR)의 개수는 m+(3m-1)/2 개이고 $1T_A+2T_X$의 지연이 있는 곱셈기이다. 2006년에 Reyhani-Masoleh는 전체 gate(AND+XOR)의 개수가 타입 IV인 경우 (7n-3)/2이고 일반적으로는 (C_n+3m)/2인 것을 제안하였다. 이러한 모든 경우에 오류 탐지C_P(pred)에 필요한 gate 수는 AND gate 1개, XOR gate 1개, 1-bit register 1개가 필요하다. 구체적인 것은 아래의 표 1에 제시하였다.

표 1

Serial multipliers		Complexities = original + [overhead]			Overhead*(%)
Reyhani-Masoleh [2]	kim[3]	General	AND	n + [1]	
			XOR	#XOR+[n+1], #XOR≤(C_n+n)/2	
			#1-bit reg.	3n + [1]	
			Delay	$D_G + [\lceil \lg n+1 \rceil]T_X$, $D_G \le n(T_A+(1+\lceil \lg k \rceil)T_X)$	
	Yang[8], Kwon[4]	Type II	AND	n + [1]	0.4
			XOR	(3n-1)/2 + [n+1]	67
			#1-bit reg.	3n + [1]	0.14
			Delay	$n(T_A+2T_X)+(\lceil \lg(n+1) \rceil)T_X$	1.1
	Kim[3]	Type IV	AND	n + [1]	0.2(0.6)
			XOR	(5n-7)/2+[n+1]	40.2(40.59)
			#1-bit reg.	3n + [1]	0.08(0.2)
			Delay	$n(T_A+3T_X)+(\lceil \lg(n+1) \rceil)T_X$	0.7(1.23)
		Type X	AND	n + [1]	0.175
			XOR	(11n-73)/2+[n+1]	18.4
			#1-bit reg.	3n + [1]	0.058
			Delay	$n(T_A+5T_X)+(\lceil \lg(n+1) \rceil)T_X$	0.29

* GF(2^n):NIST recommended field II:233, IV:409,163, X:571, overhead/original×100%

[0113]

[0114] 시간 복잡도는 타입 k의 경우 1 clock 당 $T_A+(1+\lg\lceil k \rceil)T_X$ 이 필요하나 전체 계산 결과를 출력하는데 필요한 n clock을 고려하면 n($T_A+(1+\lg\lceil k \rceil)T_X$) 의 시간 복잡도가 필요하다. 오류 탐지 곱셈기는 A와 B의 곱 결과의 bit와 C_P(pred)의 XOR에 필요한 $\lg\lceil n+1 \rceil T_X$ 가 추가로 더 필요하다. 전체적인 복잡도는 표 1과 같다.

[0115] 도 6은 본 발명의 일 실시예에 따른 유한체의 직렬 곱셈기에 대한 오류 탐지 방법의 흐름도이다.

[0116] 610 단계는 유한체의 직렬 곱셈기에 있어서, 제 1 입력의 비트 중 하나의 비트와 제 2 입력의 비트 중 하나의 비트를 이용하여 제 1 패리티 비트를 예측하는 단계이다.

[0117] 보다 구체적으로, 유한체의 직렬 곱셈기의 오류를 감지하기 위하여, 유한체의 직렬 곱셈기의 입력인 제 1 입력의 비트 중 하나의 비트와 제 2 입력의 비트 중 하나의 비트를 이용하여 제 1 패리티 비트를 예측한다. 본 단계에 대한 상세한 설명은 도 1 및 도 2의 제 1 패리티 비트 예측부(110)에 대한 상세한 설명에 대응하는바, 도 1 및 도 2의 제 1 패리티 비트 예측부(110)에 대한 상세한 설명으로 대신한다.

[0118] 620 단계는 상기 제 1 입력과 상기 제 2 입력의 직렬 곱셈을 통해 산출된 결과의 각 비트 값과 제 1 패리티 비트를 비교하여 오류가 있는지를 탐지하는 단계이다.

[0119] 보다 구체적으로, 상기 제 1 입력과 상기 제 2 입력의 직렬 곱셈을 통해 산출된 결과의 각 비트 값과 610 단계에서 예측한 제 1 패리티 비트를 비교하여 오류가 있는지를 판단한다. 실제로 직렬 곱셈기에서 산출된 결과값의 패리티 비트와 제 1 패리티 비트가 같은지를 통해 오류가 있는지를 판단한다. 본 단계에 대한 상세한 설명은 도 1 및 도 2의 오류 탐지부(120)에 대한 상세한 설명에 대응하는바, 도 1 및 도 2의 오류 탐지부(120)에 대한 상세한 설명으로 대신한다.

[0120] 도 7은 본 발명의 실시예에 따른 유한체의 직렬 곱셈기에 대한 오류 탐지 방법의 흐름도이다.

[0121] 710 단계는 제 1 입력의 비트 중 하나의 비트와 제 2 입력의 비트 중 하나의 비트의 곱셈 연산을 수행하는 단계이다.

[0122] 보다 구체적으로, 상기 제 1 입력과 상기 제 2 입력의 비트 중 대응하는 자리의 비트를 이용하여 곱셈 연산을 수행한다. 본 단계에 대한 상세한 설명은 도 2의 곱셈 연산부(111)에 대한 상세한 설명에 대응하는바, 도 2의 곱셈 연산부(111)에 대한 상세한 설명으로 대신한다.

[0123] 720 단계는 상기 곱셈 연산의 곱셈 값과 제 1 패리티 비트 레지스터의 저장 값의 배타적 논리합 연산을 수행하는 단계이다.

[0124] 보다 구체적으로, 710 단계의 곱셈 값과 제 1 패리티 비트 레지스터의 저장 값의 배타적 논리합 연산을 수행한다. 제 1 입력과 제 2 입력의 각각의 비트의 곱셈 연산 값들의 배타적 논리합 연산을 수행하기 위하여, 상기 곱셈 연산의 곱셈 값과 제 1 패리티 비트 레지스터의 저장 값의 배타적 논리합 연산을 수행한다. 본 단계에 대한 상세한 설명은 도 2의 배타적 논리합 연산부(112)에 대한 상세한 설명에 대응하는바, 도 2의 배타적 논리합 연산부(112)에 대한 상세한 설명으로 대신한다.

[0125] 730 단계는 상기 배타적 논리합 연산의 배타적 논리합 연산 값을 제 1 패리티 비트 레지스터에 저장하는 단계이다.

[0126] 보다 구체적으로, 제 1 입력과 제 2 입력의 각각의 비트의 곱셈 연산 값들의 배타적 논리합 연산을 수행하기 위하여, 710 단계에서 연산된 배타적 논리합 연산의 배타적 논리합 연산 값을 제 1 패리티 비트 레지스터에 저장한다. 상기 직렬 곱셈 연산의 종료 시, 상기 제 1 패리티 비트 레지스터에 저장된 값이 상기 제 1 패리티 비트로 최종 예측한다. 본 단계에 대한 상세한 설명은 도 2의 제 1 패리티 비트 레지스터(113)에 대한 상세한 설명에 대응하는바, 도 2의 제 1 패리티 비트 레지스터(113)에 대한 상세한 설명으로 대신한다.

[0127] 740 단계는 상기 직렬 곱셈을 통해 산출된 결과의 각 비트 값들과 상기 제 1 패리티 비트를 2 개씩 짝을 이루어 배타적 논리합 연산을 순차적으로 수행함으로써 최종 산출되는 배타적 논리합 연산 값으로부터 상기 오류가 있는지를 탐지하는 단계이다.

[0128] 보다 구체적으로, 710 단계 내지 730 단계의 반복을 통해 최종 예측된 제 1 패리티 비트와 직렬 곱셈을 통해 산출된 결과의 각 비트 값들을 2 개씩 짝을 이루어 배타적 논리합 연산을 순차적으로 수행한다. 상기 배타적 논리합 연산을 통해 최종 산출되는 배타적 논리합 연산 값이 0인지 1인지에 따라 오류가 있는지를 판단한다. 최종 산출되는 배타적 논리합 연산 값이 0이면, 오류가 존재하지 않는 것으로 판단하고, 최종 산출되는 배타적 논리합 연산 값이 1이면, 오류가 존재하는 것으로 판단한다. 본 단계에 대한 상세한 설명은 도 1 및 도 2의 오류 탐지부(120)에 대한 상세한 설명에 대응하는바, 도 1 및 도 2의 오류 탐지부(120)에 대한 상세한 설명으로 대신한다.

[0129] 본 발명의 실시예들은 다양한 컴퓨터 수단을 통하여 수행될 수 있는 프로그램 명령 형태로 구현되어 컴퓨터 판독 가능 매체에 기록될 수 있다. 상기 컴퓨터 판독 가능 매체는 프로그램 명령, 데이터 파일, 데이터 구조 등을 단독으로 또는 조합하여 포함할 수 있다. 상기 매체에 기록되는 프로그램 명령은 본 발명을 위하여 특별히 설계되고 구성된 것들이거나 컴퓨터 소프트웨어 당업자에게 공지되어 사용 가능한 것일 수도 있다. 컴퓨터 판독 가능 기록 매체의 예에는 하드 디스크, 플로피 디스크 및 자기 테이프와 같은 자기 매체(magnetic media), CD-ROM, DVD와 같은 광기록 매체(optical media), 플롭티컬 디스크(floptical disk)와 같은 자기-광 매체(magneto-optical media), 및 롬(ROM), 램(RAM), 플래시 메모리 등과 같은 프로그램 명령을 저장하고 수행하도록 특별히 구성된 하드웨어 장치가 포함된다. 프로그램 명령의 예에는 컴파일러에 의해 만들어지는 것과 같은 기계어 코드뿐만 아니라 인터프리터 등을 사용해서 컴퓨터에 의해서 실행될 수 있는 고급 언어 코드를 포함한다. 상기된 하드웨어 장치는 본 발명의 동작을 수행하기 위해 하나 이상의 소프트웨어 모듈로서 작동하도록 구성될 수 있으며, 그 역도 마찬가지이다.

[0130] 이상과 같이 본 발명에서는 구체적인 구성 요소 등과 같은 특정 사항들과 한정된 실시예 및 도면에 의해 설명되었으나 이는 본 발명의 보다 전반적인 이해를 돕기 위해서 제공된 것일 뿐, 본 발명은 상기의 실시예에 한정되는 것은 아니며, 본 발명이 속하는 분야에서 통상적인 지식을 가진 자라면 이러한 기재로부터 다양한 수정 및 변형이 가능하다.

[0131] 따라서, 본 발명의 사상은 설명된 실시예에 국한되어 정해져서는 아니되며, 후술하는 특허청구범위뿐 아니라 이

특허청구범위와 균등하거나 등가적 변형이 있는 모든 것들은 본 발명 사상의 범주에 속한다고 할 것이다.

부호의 설명

[0132] 100: 오류 탐지 장치

110: 제 1 패리티 비트 예측부

111: 곱셈 연산부

112: 배타적 논리합 연산부

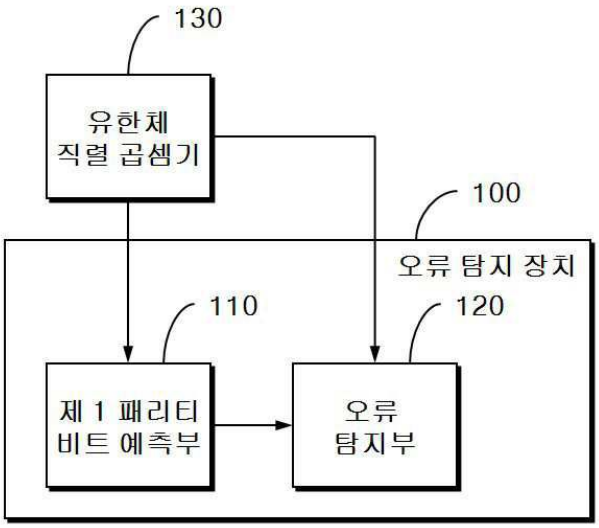
113: 제 1 패리티 비트 레지스터

120: 오류 탐지부

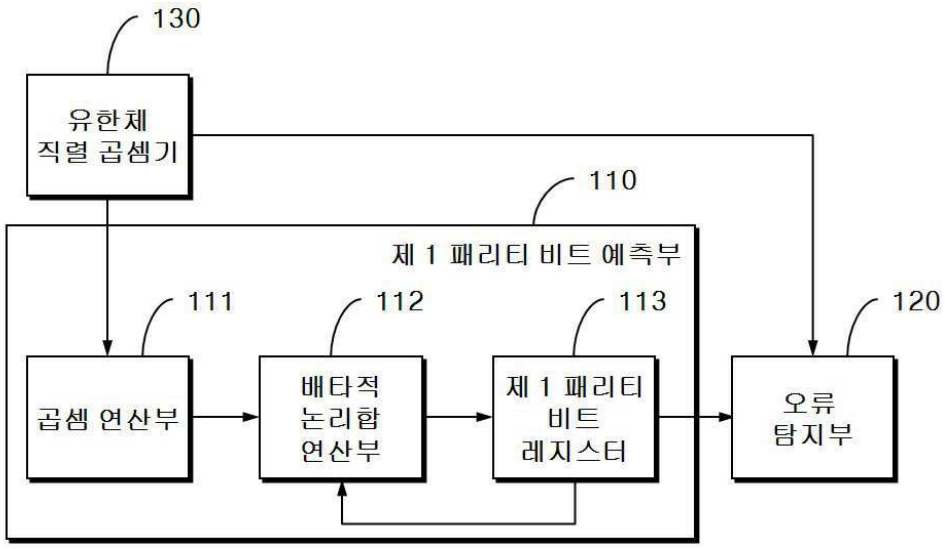
130: 유한체 직렬 곱셈기

도면

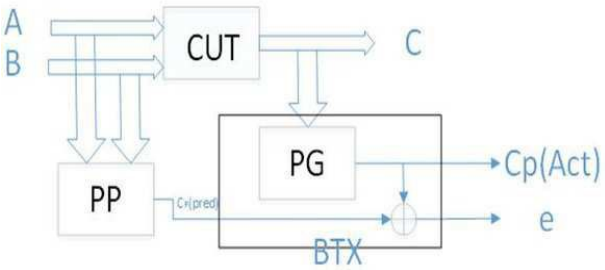
도면1



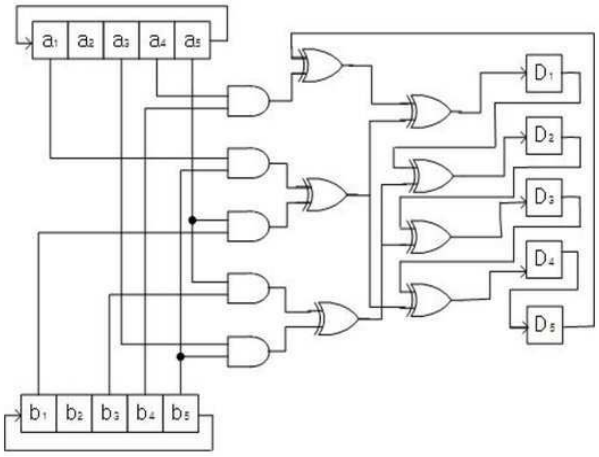
도면2


130
유한체
직렬 곱셈기
110
제 1 패리티 비트 예측부
111
112
113
120
곱셈 연산부
배타적
논리합
연산부
제 1 패리티
비트
레지스터
오류
탐지부

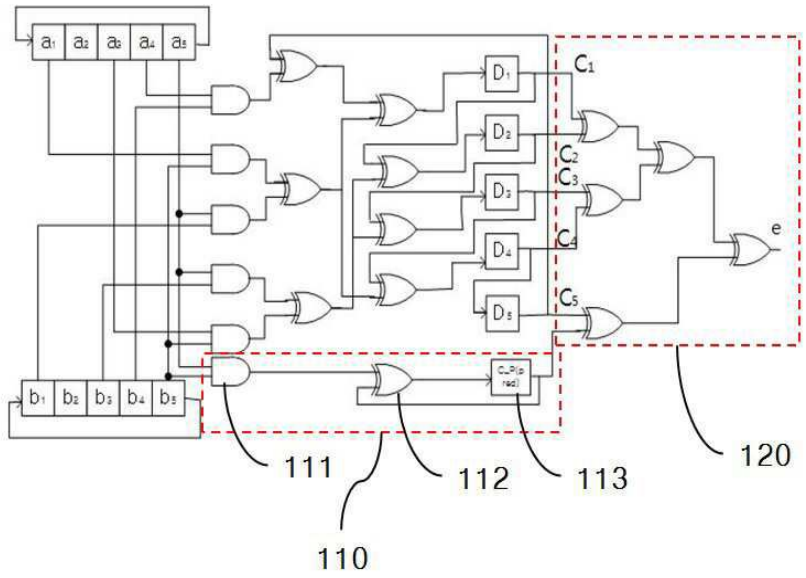
도면3


A
B
CUT
C
PG
Cp(Act)
PP
e
BTX

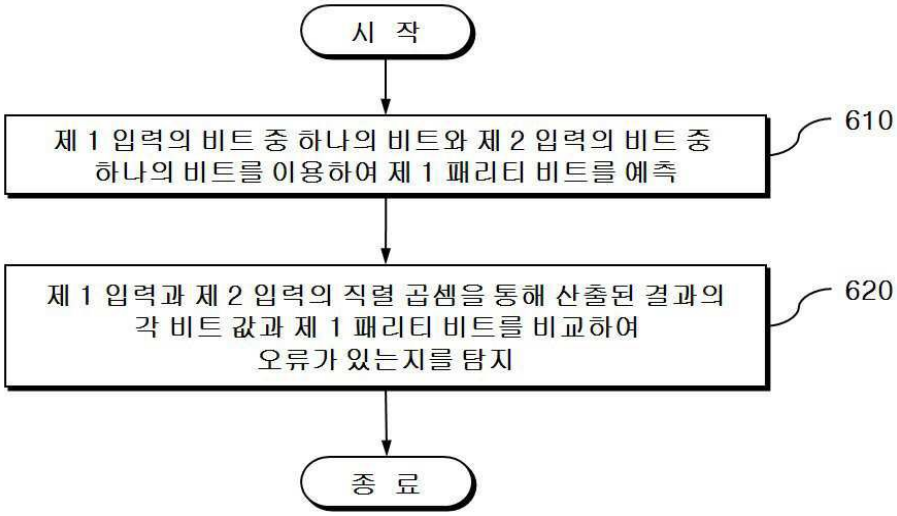
도면4


a1 a2 a3 a4 a5
b1 b2 b3 b4 b5
D1
D2
D3
D4
D5

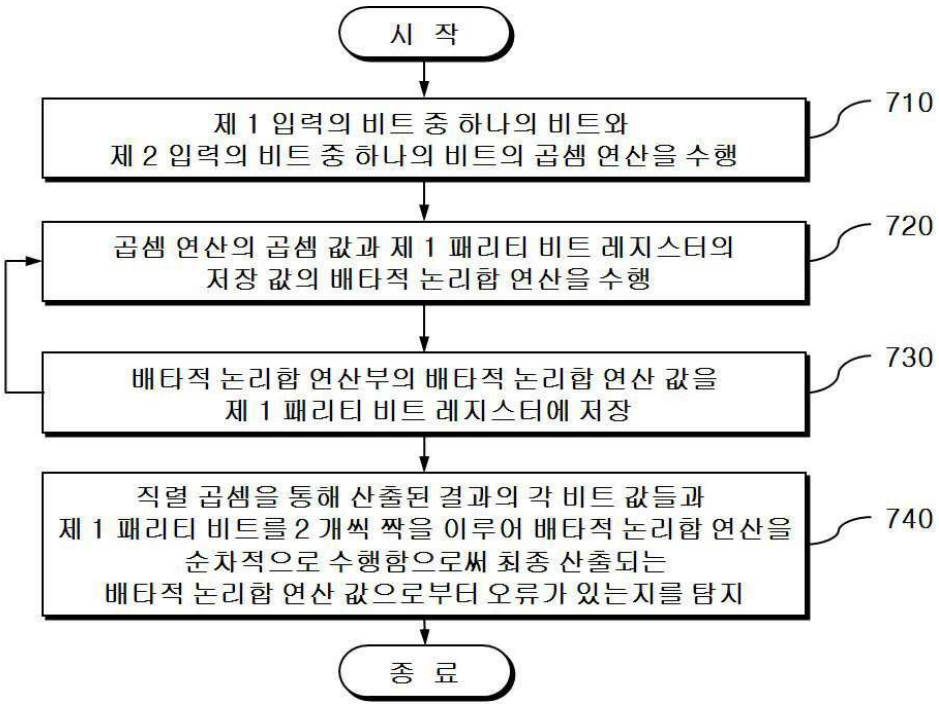
도면5


a1 a2 a3 a4 a5
b1 b2 b3 b4 b5
D1
D2
D3
D4
D5
C1
C2
C3
C4
C5
e
111
112
113
120
110

도면6


시 작
제 1 입력의 비트 중 하나의 비트와 제 2 입력의 비트 중 하나의 비트를 이용하여 제 1 패리티 비트를 예측
610
제 1 입력과 제 2 입력의 직렬 곱셈을 통해 산출된 결과의 각 비트 값과 제 1 패리티 비트를 비교하여 오류가 있는지를 탐지
620
종 료

도면7


시 작
제 1 입력의 비트 중 하나의 비트와
제 2 입력의 비트 중 하나의 비트의 곱셈 연산을 수행
710
곱셈 연산의 곱셈 값과 제 1 패리티 비트 레지스터의
저장 값의 배타적 논리합 연산을 수행
720
배타적 논리합 연산부의 배타적 논리합 연산 값을
제 1 패리티 비트 레지스터에 저장
730
직렬 곱셈을 통해 산출된 결과의 각 비트 값들과
제 1 패리티 비트를 2 개씩 짝을 이루어 배타적 논리합 연산을
순차적으로 수행함으로써 최종 산출되는
배타적 논리합 연산 값으로부터 오류가 있는지를 탐지
740
종 료