(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2011년01월12일
(11) 등록번호 10-1007425
(24) 등록일자 2011년01월04일

(51) Int. Cl.
G06F 7/52 (2006.01) *G06F 17/00* (2006.01)
(21) 출원번호 10-2009-0081955
(22) 출원일자 2009년09월01일
심사청구일자 2009년09월01일
(56) 선행기술조사문헌
KR100907547 B1
KR100875928 B1

(73) 특허권자
인하대학교 산학협력단
인천 남구 용현동 253 인하대학교
(72) 발명자
이한호
인천시 남구 용현3동 인하대학교 정보통신공학부
안효진
인천시 남구 용현동 인하대학교 하이테크센터 528호
(74) 대리인
특허법인무한

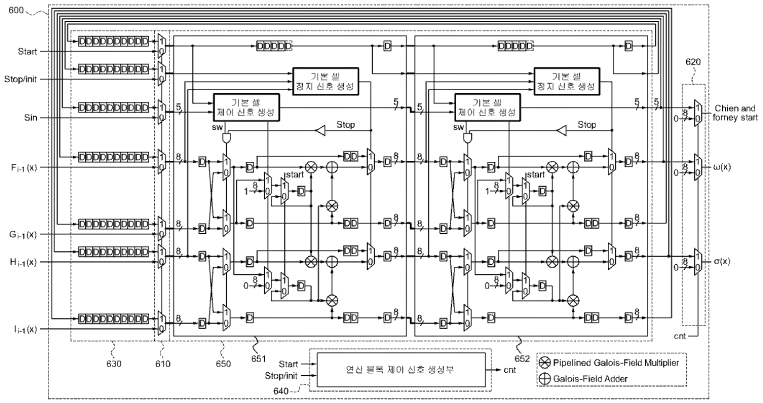
전체 청구항 수 : 총 5 항 심사관 : 임영희

(54) 고속 저면적 리드-솔로몬 복호기를 위한 폴딩 방식 수정 유클리드 알고리즘 연산 회로 및 그 방법

(57) 요 약

고속 저면적 리드-솔로몬 복호기를 위한 폴딩 방식 수정 유클리드 알고리즘 연산 회로는 수정 유클리드 알고리즘의 차수 비교 및 계산 부분이 제거된 기본 셀을 이용하여 반복적으로 사용하고, 연산 블록 제어 신호를 생성하여 종래의 시스토릭-어레이 구조보다 저면적의 효율적인 리드-솔로몬 복호기 구현이 가능한 폴딩 방식의 새로운 연산 블록 구조를 이용하여 획기적으로 면적을 줄이는 것이 가능하다.

대 표 도



이 발명을 지원한 국가연구개발사업
과제고유번호 IITA-2009-C1090-0902-0019
부처명 지식경제부 정보통신연구진흥원
연구관리전문기관
연구사업명 대학IT연구센터 육성지원사업
연구과제명 초광대역(UWB) 무선 전송 및 시스템 응용기술
기여율
주관기관 인하대학교 산학협력단
연구기간 2009.01.01-2009.12.31

특허청구의 범위

청구항 1

신드롬 다항식을 기초로 오류 값 다항식 및 오류 위치 다항식을 연산하기 위한 제1 부분 셀과 제2 부분 셀로 구성된 기본 셀을 반복적으로 사용하는 폴딩 구조로 구성되며,

상기 제1 부분 셀은 1차적으로 쉬프트 연산, 다항식 연산, 교환 연산을 수행하고, 연산 결과 값을 상기 제2 부분 셀의 입력으로 전달하며,

상기 제2 부분 셀은 2차적으로 쉬프트 연산, 다항식 연산, 교환 연산을 수행하고,

상기 기본 셀은 쉬프트 연산과 입력 다항식의 최고차항의 계수를 서로 상대방의 다항식과 곱하여 더하는 수식 연산을 동시에 수행하고, 두 번의 쉬프트 연산을 수행하는 수정 유클리드 알고리즘 연산 회로.

청구항 2

제1항에 있어서,

상기 기본 셀은 출력이 다시 입력으로 재귀하기 위한 출력 저장부를 더 포함하고,

상기 기본 셀의 출력 값을 순차적으로 저장하여 상기 기본 셀의 제1 부분 셀이 연산이 끝날 때까지 입력을 지연시키는 상기 출력 저장부에 포함된 레지스터의 수가 상기 부분 셀 내부의 파이프라인 단계에 따라 달라지는, 수정 유클리드 알고리즘 연산 회로.

청구항 3

제2항에 있어서,

상기 제1 부분 셀의 최초 입력 다항식F_0은 신드롬 다항식 S(x)에 상기 신드롬 다항식 S(x)의 변수인 x를 곱한 값, G_0는 오류 정정 가능한 심벌 수의 2배의 차수를 가지는 값, H_0는 상기 x, I_0는 0이며, 상기 최초 입력 다항식 이후의 입력 다항식은 상기 출력 저장부에 저장되어 있는 F_i, G_i, H_i, I_i ($i \geq 1$)값이 입력으로 전달되어지도록 하는 선택 제어 신호에 따라 입력이 결정되어지는 입력 신호 선택부를 더 포함하는 수정 유클리드 알고리즘 연산 회로.

청구항 4

제3항에 있어서,

상기 기본 셀의 반복 연산 중 오류 값 다항식과 오류 위치 다항식을 상기 기본 셀의 최종 결과값으로 출력하는 출력 신호 선택부를 더 포함하고,

상기 기본 셀의 연산 횟수가 정해진 횟수가 되면, 상기 제2 부분 셀의 출력 신호 선택부에 의해 최종 결과 값, 오류위치 다항식과 오류 값 다항식으로 $F_i(x)$, $H_i(x)$가 출력되는, 수정 유클리드 알고리즘 연산 회로.

청구항 5

제4항에 있어서,

상기 입력 신호 선택부의 제어 신호 및 상기 출력 신호 선택부의 제어 신호를 생성하는 연산 블록 제어 신호 생성부를 더 포함하고,

상기 기본 셀의 최초 입력 시 상기 입력 신호 선택부의 제어 신호를 생성하고, 상기 정해진 횟수만큼 연산을 수행한 후 상기 출력 신호 선택부의 제어 신호를 생성하여 재귀되어지는 상기 기본 셀의 입력 값에서 최종 오류 위치 다항식과 오류값 다항식의 결과 값을 출력하는, 수정 유클리드 알고리즘 연산 회로.

명 세 서

발명의 상세한 설명

기 술 분 야

[0001] 본 발명의 실시예들은 고속 리드-솔로몬 복호기를 위한 수정 유클리드 알고리즘을 이용한 폴딩 방식 연산 회로 및 그 방법에 관한 것으로, 더욱 상세하게는 종래의 차수 계산 부분 및 비교 부분이 제거된 수정 유클리드 연산 회로의 기본 셀을 이용하여 KES 블록의 새로운 폴딩 방식의 연산 회로 및 그 방법에 관한 것이다.

배 경 기 술

[0002] 일반적으로 리드-솔로몬 코드는 마그네틱, 광 저장매체, 광통신, 방송통신, 유선 및 위성 통신 등 다양한 응용분야에 널리 쓰이는 순방향 에러 보정(FEC: Forward Error Correction) 기술로서 통신로상에서 발생되는 랜덤 오류 및 연집 오류까지 정정 가능한 채널 부호화 방식이다. 리드-솔로몬 부호를 나타내는 전형적인 형태는 RS(n,k,t)이다. 여기서, n은 부호 심볼 수, k는 정보 심볼 수, t는 오류 정정 가능 심볼 수를 나타내며, t=(n-k)/2이다. 따라서, RS(255,239)에서 t=8이다.

[0003] 도 1은 종래의 리드-솔로몬 복호기(100)의 구조를 도시한 도면이다.

[0004] 도 1을 참조하면, 신드롬 다항식 계산(Syndrome Polynomial Computation)부(110)는 신드롬 다항식 S(x)를 생성하고, 수신된 코드워드(codeword)의 오류 패턴을 표현한다. 신드롬 다항식 S(x)는 리드-솔로몬 복호기(100)의 KES(Key_Equation Solver)부(120)에서 사용된다. KES(Key_Equation Solver)부(120)는 키 등식(key equation) $S(x)\sigma(x)=\omega(x) \bmod x^{2t}$을 해결하기 위해 유클리드 알고리즘, 수정 유클리드 알고리즘 또는 벨르캄프-메시(Berlecamp-Massay) 알고리즘 등을 이용한다. 이러한 알고리즘은 오류 위치 다항식(error locator polynomial) $\sigma(x)$와 오류 값 다항식(error value polynomial) $\omega(x)$을 구하기 위하여 사용된다.

[0005] 유클리드 알고리즘은 유한체(Galois-field) 연산의 역원이 사용되므로 이를 위한 룩업 테이블(Look Up Table: 이하 LUT라고 함)가 ROM의 형태로 필요하다. 그러나 수정 유클리드 알고리즘은 LUT를 필요로 하지 않으므로 LUT 사용으로 인한 속도 지연을 줄일 수 있고, 시스토릭-어레이(systolic-array) 구조 및 폴딩(Folding) 구조로 구현이 가능하여 벨르캄프-메시 알고리즘에 비해 파이프라인화(pipelined)가 용이하여 쉽게 고속 구현을 할 수 있다.

[0006] $\sigma(x)$와 $\omega(x)$의 두 다항식은 친 서치(Chien Search)부(130)와 포니(Formey) 알고리즘 연산부(140)를 이용하여 오류의 위치에 대응하는 오류들의 크기 값을 구하기 위해 사용된다.

[0007] KES부(120)에서 사용되는 수정된 유클리드 알고리즘은 최고차항을 소거하여 반복적으로 차수를 줄이는 방식으로 오류 위치 및 오류 값 다항식을 구하기 위해 하기 수학식 1과 같이 초기값을 정한다. 여기서, S(x)는 앞서 언급한 신드롬 다항식이다.

수학식 1

[0008] $A_0(x)=x^{2t}$, $B_0(x)=S(x)$, $C_0(x)=0$, $D_0(x)=1$

[0009] 하기 수학식 2 및 수학식 3은 반복적으로 i번 수행될 $A_i(x)$, $B_i(x)$, $C_i(x)$, $D_i(x)$에 대해 나타낸 것이다. a_i와 b_i는 각각 $A_i(x)$, $B_i(x)$의 최고차항의 계수이고 다항식 $A_i(x)$, $B_i(x)$, $C_i(x)$, $D_i(x)$의 값은 다항식 $A_{i-1}(x)$, $B_{i-1}(x)$의 차수에 따라 값이 결정된다. $\deg(A_i(x))$와 $\deg(B_i(x))$는 각각 다항식 $A_i(x)$, $B_i(x)$의 차수를 나타낸다. $\deg(A_i(x)) < \deg(B_i(x))$가 되면 반복 연산을 멈추게 되고 이때의 $A_i(x)$와 $C_i(x)$가 각각 오류 값 다항식과 오류 위치 다항식이 된다.

수학식 2

[0010] $$A_i(x) = [\sigma_{i-1}\ \sigma_{i-1}b_{i-1}A_{i-1}(x)-\overline{\sigma}_{i-1}a_{i-1}B_{i-1}(x) - x^{|1i-1|}[\sigma_{i-1}a_{i-1}B_{i-1}(x)-\overline{\sigma}_{i-1}b_{i-1}A_{i-1}(x)]$$

[0011] $$B_i(x) = \sigma_{i-1}B_{i-1}(x) - \overline{\sigma}_{i-1}A_{i-1}(x)$$

수학식 3

[0012] $C_i(x) = [\sigma_{i-1} b_{i-1}C_{i-1}(x) - \overline{\sigma}_{i-1} a_{i-1}D_{i-1}(x) - x^{|1i-1|}[\sigma_{i-1} a_{i-1}D_{i-1}(x) - \overline{\sigma}_{i-1} b_{i-1}C_{i-1}(x)]$

[0013] $D_i(x) = \sigma_{i-1} D_{i-1}(x) - \overline{\sigma}_{i-1} C_{i-1}(x)$

수학식 4

[0014] $l_{i-1}=Deg(A_{i-1})-Deg(B_{i-1})$

[0015] $\sigma_{i-1} = \begin{cases} 1, & \text{if } l_{i-1} \geqq 0 \\ 0, & \text{if } l_{i-1} < 0 \end{cases}$

[0016] 수정 유클리드 알고리즘은 상기 수학식 2부터 수학식 4까지 반복적으로 연산을 하는 것이다. 이와 같은 종래의 첫 번째 방식은 수학식 4의 두 다항식 $A_{i-1}(x)$, $B_{i-1}(x)$의 차수를 계산하고 l_{i-1}을 구하여 수학식 2와 수학식 3이 빠르게 연산될 수 있도록 제어 신호를 만든다. 상기 제어 신호는 반복 연산의 종료 조건인 $deg(A_i(x)) < deg(B_i(x))$에 따른 정지(stop) 신호도 포함하고 있다.

[0017] 도 2는 기본 셀의 출력을 다시 입력으로 재귀하여 하나의 기본 셀을 반복적으로 사용하는 종래의 재귀적인 구조를 도시한 도면이다.

[0018] 상기에 설명한 도 2의 수정 유클리드 알고리즘 연산 회로는 수학식 1부터 수학식 4를 수행하는 기본 셀로 구성되고, 기본 셀은 각각 다항식의 차수를 계산하고, 비교하는 연산 회로가 포함되어 있어 많은 면적을 차지한다.

수학식 5

[0019] $R_i(x) = b_{i-1}R_{i-1}(x) - x^{|1i-1|}a_{i-1}Q_{i-1}(x)$

[0020] $Q_i(x) = Q_{i-1}(x)$

수학식 6

[0021] $R_i(x) = a_{i-1}Q_{i-1}(x) - x^{|1i-1|}b_{i-1}R_{i-1}(x)$

[0022] $Q_i(x) = R_{i-1}(x)$

수학식 7

[0023] $if(l_{i-1}<0)\ SWAP(R_{i-1}(x),Q_{i-1}(x))$

[0024] $R_i(x) = b_{i-1}R_{i-1}(x) - x^{|1i-1|}a_{i-1}Q_{i-1}(x)$

[0025] $Q_i(x) = Q_{i-1}(x)$

수학식 8

[0026] $if(l_{i-1}<0)\ SWAP(R_{i-1}(x),Q_{i-1}(x))$

[0027] $L_i(x) = b_{i-1}L_{i-1}(x) - x^{|1i-1|}a_{i-1}U_{i-1}(x)$

[0028] $U_i(x) = U_{i-1}(x)$

[0029] 종래의 두 번째 방식은 수학식 2에서 σ_{i-1}가 1인 경우는 수학식 5의 형태가 되고 σ_{i-1}가 0인 경우는 수학식 6의 형태가 된다. l_{i-1}이 음수인 경우 $R_{i-1}(x)$와 $Q_{i-1}(x)$를 서로 바꾸어 수학식 5에 적용하면 $R_{i-1}(x)$new는 $Q_{i-1}(x)$이고, $Q_{i-1}(x)$new는 $R_{i-1}(x)$이고, a_{i-1}_new는 $Q_{i-1}(x)$의 최고차항의 계수 b_{i-1}이고, b_{i-1}_new는 $R_{i-1}(x)$의 최고차항의 계수 a_{i-1}이므로 $R_{i-1}(x)$와 $Q_{i-1}(x)$를 수학식 6에 적용한 것과 같은 값을 가지게 된다. 즉, 수학식 5과 수학식 6을 수학식 7의 형태로 표현할 수 있고 같은 방법으로 수학식 8를 나타낼 수 있다.

[0030] 도 3에서 기본 셀의 연산과 관계하여 수학식 7의 연산은 쉬프트 연산, 다항식 연산, 교환 연산의 세 가지로 분류될 수 있다. 즉, 기본 셀의 제어 신호는 세 가지가 된다.

[0031] 도 3의 기본 셀은 x를 곱하는 쉬프트 연산과 두 입력 다항식의 최고차항의 계수를 서로 상대방의 다항식과 곱하여 더하는 수식 연산을 동시에 할 수 없고 하나의 셀은 두 번 이상의 쉬프트 연산을 수행할 수 없다. 그리고 쉬프트 연산을 수행할 다항식은 멀티플렉서의 출력단에 위치해야 한다. l_{i-1}을 이용하지 않고 세 가지의 제어 신호를 생성하기 위해 다항식의 입력 패턴을 이용한다. 세 가지의 입력패턴은 다음의 수학식 10부터 수학식 12까지로 나타낸다.

수학식 9

[0032] $C_0(x)=xQ(x)$, $D_0(x)=R(x)$, $E_0(x)=x$, $F_0(x)=0$

수학식 10

[0033] $R_0(x)=x^{2t}$

[0034] $$Q(x) = S_{2t-1}x^{2t-1} + S_{2t-2}x^{2t-2} + \cdots S_1x^1 + S_0 (S_i \neq 0, 0 \leq i \leq 2t-1)$$

수학식 11

[0035] $$R_0(x) = x^{2t}$$

[0036] $$Q(x) = S_{2t-1}x^{2t-1} + S_{2t-2}x^{2t-2} + \cdots S_1x^1 + S_0 \begin{cases} S_i = 0, & 2t-1-k \leq 2t-1 \ \ k \geq 0, k\text{은 정수} \\ S_i \neq 0, & otherwise \end{cases}$$

수학식 12

[0037] $$R_0(x) = x^{2t}$$

[0038] $$Q(x) = S_{2t-1}x^{2t-1} + S_{2t-2}x^{2t-2} + \cdots S_1x^1 + S_0 \begin{cases} S_i = 0, & 2t-1-m \leq 2t-1 \ \ m \geq 0, m\text{은 정수} \\ S_i \neq 0, & otherwise \end{cases}$$

[0039] $Q_0(x)$의 차수는 항상 $R_0(x)$보다 작으므로 도 3의 기본 셀1은 수학식 9의 입력을 가진다. 수학식 10의 경우는 S_i의 값이 모두 0이 아니다. 이는 수학식 7의 연산이 $R_0(x)$의 최고차항만을 소거하는 것을 의미한다. 다시 수학식 7을 적용하여 $R_2(x)$와 $Q_2(x)$를 구할 수 있고, $R_2(x)$가 $R_1(x)$의 최고차항만을 소거한 형태라면 $R_2(x)$와 $Q_2(x)$를 서로 스왑한 후 다시 수학식 10과 같은 입력 패턴이 된다. 만약 $R_2(x)$가 $R_1(x)$의 최고차항뿐만 아니라 두 번째 항까지 소거하였다면 $R_2(x)$와 $Q_2(x)$는 서로 스왑한 후 수학식 11의 입력 패턴이 된다.

[0040] 도 3의 기본 셀과 관련하여 $C_1(x)$와 $D_1(x)$는 각각 $R_1(x)$와 $Q_1(x)$가 되고 $C_2(x)$와 $D_2(x)$는 각각 $R_2(x)$와 $Q_2(x)$가 된다. 다항식 연산 또는 쉬프트 연산을 수행하는 경우 $D_i(x)$의 출력은 $C_i(x)$의 출력에 비해 1 클럭 지연하는 데, 이는 두 가지의 역할을 수행한다. 첫 번째는 수학식 7에서 덧셈연산을 위해 $Q_{i-1}(x)$에 곱해진 x를 소거하는 것이고, 두 번째는 $R_{i-1}(x)$와 $Q_{i-1}(x)$를 스왑하기 전에 미리 $R_{i-1}(x)$에 x를 곱하여 다항식 연산을 쉬프트 연산 없이 바로 수행할 수 있게 한다.

[0041] 도 4에서 S0상태는 연산이 처음으로 시작되거나 새 연산이 시작될 때의 상태를 나타내고 출력은 Sw 신호와 Sht 신호이다.

[0042] 도 4의 출력은 기본 셀의 제어 신호 Sw 및 Sht와 같고, Sw가 1일 때 교환 연산, Sht가 1일 때 쉬프트 연산, Sht가 0일 때 다항식 연산이 된다. S0상태에서 두 입력 다항식의 차수가 같은지 아닌지 여부에 따라 S1 또는 S2상태로 천이되고 두 입력 다항식의 차수가 같은지 아닌지를 나타내는 입력 신호는 SI이다. 수학식 10의 경우는 두 입력 다항식의 차수가 같으므로 SI는 1이 되고 S1상태로 천이된다. 이 경우 연산은 다항식 연산이므로 Sht 신호는 0이 되고, $C_0(x)$는 $xQ_0(x)$이므로 다항식 연산을 위해 Sw 신호는 1이 된다. $R_1(x)$가 $R_0(x)$에서 최고차항만 소거된 경우라면 다항식 연산을 수행하게 되고, 도 4에서 S1상태에서 SI가 1일 때의 제어신호가 기본 셀2에 적용된다. 계속해서 $R_i(x)$가 $R_{i-1}(x)$의 최고차항만 소거되는 경우 도 4의 상태는 S0-S1을 반복하게 된다.

[0043] 수학식 11의 입력 패턴을 가지는 경우 기본 셀1의 입력인 $xQ_0(x)$의 차수가 $R_0(x)$의 차수보다 낮으므로 $xQ_0(x)$에 x를 곱하는 연산을 수행한다. 그러므로 도4에서 초기 S0상태에서 SI 입력이 0인 경우 S2상태가 되고 쉬프트 연산에 대한 제어신호가 출력이 된다. 기본 셀2의 두 입력 다항식 $x^2Q_0(x)$와 $R_0(x)$의 차수가 같으면 다항식 연산을 수행한다. $Q_1(x)$와 $R_1(x)$의 차수가 같으면 다항식 연산을 수행하고, 그렇지 않으면 $R_1(x)$의 차수가 $Q_1(x)$의 차수보다 낮으므로 스왑 연산을 수행한다. k가 1인 경우의 상태 천이는 S0-S2-S3-S1-S0이다.

[0044] 수학식 12의 입력 패턴을 가지는 경우 기본 셀1은 수학식 10일 때와 같게 된다. 하지만 이때의 $R_1(x)$의 차수는 $Q_1(x)$의 차수보다 낮으므로 $R_1(x)$와 $Q_1(x)$를 다항식 연산이 아닌 스왑 연산을 수행해야 한다. 스왑 연산과 다항식 연산은 동시에 수행 가능하지만 다항식 연산과 쉬프트 연산은 동시에 수행될 수 없으므로 스왑하기 이전에 바로 다항식 연산을 수행하도록 쉬프트 연산을 수행한다. 그러므로 도 4에서 S1상태의 SI 입력이 0인 경우 쉬프트 연산에 대한 제어신호가 출력이 된다. 기본 셀2의 연산이 끝난 후 S0상태에 있게 된다.

[0045] t개의 오류가 발생한 경우 오류 위치 다항식과 오류 값 다항식은 2t개 기본 셀 연산을 수행한 후 얻을 수 있다. 하지만 v(< t)개의 오류가 발생한 경우 2t보다 적은 수의 기본 셀 연산으로 오류 위치 다항식과 오류 값 다항식을 얻게 된다. 오류 값 다항식의 차수가 오류 위치 다항식의 차수보다 낮다는 것을 기초로 하여 FSM으로 다항식의 연산의 종료를 나타내는 정지(stop) 신호가 생성되었다.

[0046] 도 5는 기본 셀의 두 입력 $C_{i-1}(x)$와 $E_{i-1}(x)$ 중 먼저 0이 아닌 값이 들어오는 다항식이 어떤 다항식인지를 확인하고 $E_{i-1}(x)$의 값이 먼저 들어오면 상태는 초기 S0에서 S2상태로 이동하고 정지 리셋 신호가 들어올 때까지 S2상태를 유지한다. 반면에 $C_{i-1}(x)$의 값이 먼저 들어오거나 동시에 들어오면 S1상태로 이동하고 마찬가지로 정지 리셋 신호가 들어올 때까지 S1상태를 유지한다.

[0047] 도 3은 차수 계산이 필요 없는(degree computationless) 수정 유클리드 알고리즘의 연산 회로를 도시한 도면이다.

[0048] t개 오류가 발생한 경우 오류 위치 다항식과 오류값 다항식은 2t개의 기본 셀 연산을 수행한 후 얻을 수 있다. 하지만 t개 이하의 오류가 발생한 경우 2t보다 적은 수의 기본 셀 연산으로 오류 위치 다항식과 오류 값 다항식을 얻게 된다. 그리고 연산이 끝나게 되면 나머지 기본 셀들은 쉬프트 연산을 수행하여 2t번째 기본 셀의 출력까지 값이 전달되어지는 연산만 수행하게 된다. t개 이하의 오류가 발생한 경우에는 t개의 오류가 발생한 경우의 기본 셀 구성 내에서 연산이 모두 끝나게 된다는 것이다. t개 이하의 오류가 발생하였을 경우에는 다양한 역할을 할 수 있는 기본 셀이 단순한 쉬프트 역할만 하게 되어 기본 셀 효율성이 떨어지게 된다. 또한 2t개의 기본 셀들로 구성된 시스토릭 어레이 구조로 KES 블록은 복호기에서 하드웨어의 대부분을 차지하고 있다.

발명의 내용

해결 하고자하는 과제

[0049] 본 발명의 일실시예는 종래의 차수 계산 부분 및 비교 부분이 제거된 수정 유클리드 연산 회로의 기본 셀을 이용하여 KES 블록의 새로운 폴딩 방식의 연산 회로 및 그 방법을 제공한다.

과제 해결수단

[0050] 본 발명의 일실시예에 따른 수정 유클리드 연산 회로는 신드롬 다항식을 기초로 오류값 다항식 및 오류 위치 다항식을 연산하기 위해 2개의 부분 셀들로 구성된 기본 셀을 이용하고, 상기 기본 셀은 셀의 입력을 다항식의 출력으로 재귀되어 들어오는 것으로 받을 것인지에 대한 입력 신호 선택부; 상기 기본 셀의 출력 다항식을 출력 저장부에 저장하거나 최종 결과값으로 출력하기 위한 출력 신호 선택부; 상기 입력 신호 선택부와 상기 출력 신호 선택부를 제어하기 위한 연산 블록 제어 신호 생성부를 포함하고, 상기 기본 셀 중 제1 부분 셀의 최초 입력 다항식F_0은 신드롬 다항식 S(x)에 상기 신드롬 다항식 S(x)의 변수인 x를 곱한 값이고, G_0는 오류 정정 가능한 심벌 수의 2배의 차수를 가지는 값이고, H_0는 상기 x이고, I_0는 '0'이며, 상기 최초 입력 다항식 이후 반복 연산을 위하여 상기 제1 부분 셀의 입력 값은 상기 기본 셀의 제2 부분 셀의 출력 다항식이 상기 출력 저장부에 저장되어 있다가 상기 제1 부분 셀의 연산 수행이 끝남과 동시에 입력으로 재귀되어 들어온다.

효 과

[0051] 본 발명의 일실시예에 따르면, 종래의 차수 계산 및 비교 부분이 제거된 수정 유클리드 알고리즘 연산 회로의 기본 셀을 이용하여 KES 블록의 새로운 폴딩 방식의 연산 회로 및 그 방법을 제시함으로써 효율적인 기본 셀의 연산 수행으로 획기적으로 면적을 줄일 수 있다.

발명의 실시를 위한 구체적인 내용

[0052] 이하에서, 본 발명에 따른 실시예들을 첨부된 도면을 참조하여 상세하게 설명한다. 그러나, 본 발명이 실시예들에 의해 제한되거나 한정되는 것은 아니다. 각 도면에 제시된 동일한 참조 부호는 동일한 부재를 나타낸다.

[0053] 도 6은 본 발명의 일실시예에 따른 폴딩 방식 수정 유클리드 알고리즘의 연산 회로의 구조를 나타내는 도면이다.

[0054] 도 6을 참조하면, 본 발명의 일실시예에 따른 폴딩 방식 수정 유클리드 알고리즘의 연산 회로(600)는 입력 신호 선택부(610), 출력 신호 선택부(620), 출력 저장부(630), 연산 블록 제어 신호 생성부(640) 및 기본 셀 연산부(650)를 포함한다.

[0055] 입력 신호 선택부(610)는 기본 셀 연산부(650)의 입력을 최초 입력다항식(F_0,G_0,H_0,I_0)으로 받은 것인지, 기본 셀 연산부(650)의 이전 연산 결과인 출력을 재귀하여 들어오는 것을 받을 것인지를 선택한다.

[0056] 출력 신호 선택부(620)는 기본 셀 연산부(650)의 연산 결과인 출력 다항식을 최종 결과 값으로 출력할 것인지를 선택한다.

[0057] 출력 저장부(630)는 기본 셀 연산부(650)의 출력을 다시 입력으로 재귀하기 위해 저장한다. 출력 저장부(630)는 기본 셀 연산부(650)의 출력을 저장하기 위해 레지스터로 구성된다. 출력 저장부(630)는 기본 셀 연산부(650)의 연산이 수행된 후 출력 값을 상기 레지스터에 저장하며, 상기 레지스터의 수는 (2t-1)-m이며, m은 기본 셀 중 제1, 2 부분 셀(651, 652)의 파이프라인 단계에 따라 달라질 수 있다. 제1 부분 셀(651) 및 제2 부분 셀(652)의 파이프라인 단계는 동일하다. 출력 저장부(630)는 상기 파이프라인 단계가 늘어남에 따라 레지스터의 수를 줄일 수 있으므로 효율적인 설계를 할 수 있다. 출력 저장부(630)는 기본 셀 연산부(650)의 출력 값을 순차적으로 저장함으로써 기본 셀의 제1 부분 셀(651)의 연산 수행이 끝날 때까지 입력을 지연시키는 역할을 수행한다.

[0058] 연산 블록 제어 신호 생성부(640)는 제1 부분 셀(651)의 입력부와 연결되는 입력 신호 선택부(610)에 대한 제어 신호 및 제2 부분 셀(652)의 출력부와 연결되는 출력 신호 선택부(620)에 대한 제어 신호를 생성한다. 즉, 연산 블록 제어 신호 생성부(640)는 제1 부분 셀(651)의 입력부와 연결되는 입력 신호 선택부(610)의 멀티플렉서에 대한 제어 신호와 제2 부분 셀(652)의 출력부와 연결되는 출력 신호 선택부(620)의 멀티플렉서에 대한 제어 신호를 생성한다.

[0059] 연산 블록 제어 신호 생성부(640)는 신드롬 다항식 계산 블록의 연산 결과 값이 기본 셀의 입력으로 전달되면, 입력 신호 선택부(610)에 제어 신호를 전달하여 F_0, G_0, H_0, I_0의 입력을 받을 수 있도록 하고, 그 이후로는 출력 저장부(630)의 F_i, G_i, H_i, I_i를 입력으로 받도록 한다. 또한 연산 블록 제어 신호 생성부(640)는 출력 신호 선택부(620)로 제어 신호를 전달하여 기본 셀 연산부(650)의 출력이 친서치 블록과 포니 알고리즘 연산블록으로 전달되도록 한다.

[0060] 입력 신호 선택부(610)는 연산 블록 제어 신호 생성부(640)에서 생성된 제어 신호에 의해 기본 셀 연산부(650)

의 입력을 F_0, G_0, H_0, I_0 또는 출력 저장부(630)의 값으로 선택한다. 일례로 입력 신호 선택부(610)는 기본 셀의 연산 횟수가 t번이 되면, 연산 블록 제어 신호 생성부(640)의 제어 신호에 의해 기본 셀의 입력부와 연결된 멀티플렉서가 신드롬 다항식 계산 블록의 결과값을 입력으로 받도록 선택할 수 있다.

[0061] 출력 신호 선택부(620)는 연산 블록 제어 신호 생성부(640)에서 생성된 제어 신호에 의해 기본 셀 연산부(650)의 최종 결과 값인 오류 위치 다항식과 오류 값 다항식의 출력을 선택한다.

[0062] 신드롬 다항식 계산 블록의 연산이 모두 끝나게 되면 기본 셀 연산부(650)의 첫 번째 입력 신호는 입력 신호 선택부(610)의 제어 신호에 의해서 F_0, G_0, H_0, I_0가 들어오게 된다. F_0, G_0, H_0, I_0의 입력은 제1 부분 셀(651)의 연산 수행을 거쳐 제2 부분 셀(652)의 결과가 출력되면 출력 저장부(630)에 순차적으로 저장된다. F_0, G_0, H_0, I_0의 입력이 모두 제1 부분 셀(651)로 입력되면, 이후부터 입력 신호 선택부(610)는 연산 블록 제어 신호 생성부(640)의 제어 신호에 의해 출력 저장부(630)에 저장된 다항식 값을 기본 셀 연산부(650)의 입력으로 선택한다. 이후 기본 셀 연산부(650)의 연산 횟수가 정해진 횟수(t번)가 되면, 연산 블록 제어 신호 생성부(640)의 제어 신호에 의해 입력 신호 선택부(610)는 신드롬 다항식 계산 블록의 결과 값을 입력으로 받도록 선택한다.

[0063] 제2 부분 셀(652)의 출력부에 연결된 멀티플렉서인 출력 신호 선택부(620)에 대한 제어 신호는 기본 셀 연산부(650)의 출력이 친서치 블록과 포니 알고리즘 연산블록으로 전달되도록 제어하기 위한 신호이다. 기본 셀의 연산 횟수가 정해진 t번이 되면, 그 결과 값은 제2 부분 셀(652)의 출력부에 연결된 멀티플렉서인 출력 신호 선택부(620)에 의해 최종 결과 값, 오류 값 다항식과 오류 위치 다항식으로 출력된다.

[0064] 기본 셀 연산부(650)는 제1 부분 셀(651) 및 제2 부분 셀(652)로 구성된 기본 셀에 대한 연산을 수행한다. 즉, 기본 셀 연산부(650)는 신드롬 다항식을 기초로 오류 값 다항식 및 오류 위치 다항식을 연산하기 위한 제1 부분 셀(651)과 제2 부분 셀(652)로 구성된 기본 셀을 반복적으로 사용하는 폴딩 구조로 구성된다.

[0065] 제1 부분 셀(651)은 1차적으로 쉬프트 연산, 다항식 연산, 교환 연산을 수행하고, 상기 연산 수행 결과 값을 제2 부분 셀(652)의 입력으로 전달한다.

[0066] 제2 부분 셀(652)은 제1 부분 셀(651)로부터 전달된 상기 연산 수행 결과 값을 이용하여 2차적으로 쉬프트 연산, 다항식 연산, 교환 연산을 수행한다.

[0067] 기본 셀 연산부(650)는 제1 부분 셀(651) 및 제2 부분 셀(652)을 이용하여 쉬프트 연산과 입력 다항식의 최고차항의 계수를 서로 상대방의 다항식과 곱하여 더하는 수식 연산을 동시에 수행할 수 있고, 두 번의 쉬프트 연산을 수행할 수 있다.

[0068] 기본 셀 연산부(650)는 스테이지가 정해진 횟수 이하인 경우일 때에 한하여 기본 셀 연산을 수행한다.

[0069] 정해진 횟수만큼 반복적인 연산을 수행한 결과 출력 다항식은 친서치 블록과 포니 알고리즘 연산 블록으로 전달되고, 신드롬 다항식 계산 블록의 출력이 기본 셀의 입력으로 전달될 때까지 대기한다.

[0070] 기본 셀의 입, 출력 전달 과정은 신드롬 다항식 계산 블록의 출력 다항식 또는 출력 저장부(630)의 저장 값이 제1 부분 셀(651)의 입력으로 전달되면, 연산이 수행된 후 결과 값이 제2 부분 셀(652)의 입력으로 전달된다. 제2 부분 셀(652)의 출력 값은 연산이 수행된 후 제1 부분 셀(651)의 입력 값으로 재귀되어 들어가거나 기본 셀의 최종 출력 값, 오류 위치 다항식과 오류 값 다항식으로 출력된다.

[0071] 제1 부분 셀(651)의 입력부에는 입력 신호 선택부(610)의 멀티플렉서가 연결되어 있어서 초기 다항식의 입력 값 또는 출력 저장부(630)에 저장되어 있는 다항식 값이 연산 블록 제어 신호 생성부(640)의 제어 신호에 의해 선택되어 들어온다.

[0072] 제2 부분 셀(652)의 출력부에는 출력 신호 선택부(620)의 멀티플렉서가 연결되어 있어서 모든 연산이 끝나게 되면, 결과 값은 KES 블록의 최종 출력 값인 오류 위치 다항식 $\sigma(x)$과 오류값 다항식 $\omega(x)$으로 출력된다.

[0073] 제1 부분 셀(651)의 입력부와 연결되는 입력 신호 선택부(610)의 멀티플렉서에 대한 제어 신호 및 제2 부분 셀(652)의 출력부와 연결되는 출력 신호 선택부(620)의 멀티플렉서에 대한 제어 신호는 연산 블록 제어 신호 생성부(640)에 의해 생성된다.

[0074] 기본 셀 연산부(650)에서 신드롬 다항식에 대한 연산이 모두 끝나게 되면 기본 셀의 첫 번째 입력 신호는 멀티플렉서의 제어 신호에 의해서 F_0, G_0, H_0, I_0가 들어오게 된다. F_0, G_0, H_0, I_0의 입력은 제1 부분 셀(651)의 연

산 수행을 거쳐 제2 부분 셀(652)의 결과가 출력되면, 출력 저장부(630)에 순차적으로 저장된다. F_0, G_0, H_0, I_0의 입력이 모두 제1 부분 셀(651)로 입력되면, 이후부터 입력 신호 선택부(610)는 연산 블록 제어 신호 생성부(640)에 의해 출력 저장부(630)에 저장된 다항식 값이 입력되도록 선택한다. 이후 기본 셀의 연산 횟수가 정해진 횟수(t)가 되면, 연산 블록 제어 신호 생성부(640)의 제어 신호에 의해 입력 신호 선택부(610)는 신드롬 다항식 계산 블록의 결과 값을 입력으로 받도록 선택한다.

[0075] 제2 부분 셀(652)의 출력부와 연결되는 출력 신호 선택부(620)의 멀티플렉서에 대한 제어 신호는 기본 셀에 입력 신호(F_0, G_0, H_0, I_0)가 제1 부분 셀(621)을 통과하여 들어오게 되면, 제2 부분 셀(652)의 출력 값이 순차적으로 출력 저장부(630)로 전달되도록 한다. 이후 기본 셀의 연산 횟수가 정해진 횟수가 되면, 그 결과 값은 제2 부분 셀(652)의 출력부와 연결되는 출력 신호 선택부(620)의 멀티플렉서에 의해 최종 결과 값, 오류 값 다항식과 오류 위치 다항식으로 $F_i(x)$, $H_i(x)$가 출력된다.

[0076] 출력 저장부(630)는 기본 셀의 출력이 다시 입력으로 재귀하기 위해 레지스터로 구성된다. 출력 저장부(630)는 기본 셀 연산부(650)에서 기본 셀의 연산이 수행된 후 출력 값을 상기 레지스터에 저장한다. 상기 레지스터의 개수는 (2t-1)-m이고, t는 정해진 연산 횟수이고, m은 기본 셀 중 부분 셀 파이프라인 단계에 따라 달라질 수 있다. 제1 부분 셀(651)과 제2 부분 셀(652)의 파이프라인 단계는 동일하다. 상기 파이프라인 단계가 늘어남에 따라 출력 저장부(630)의 레지스터의 개수를 줄일 수 있는 효율적인 설계를 할 수 있다.

[0077] 도 7은 본 발명의 일실시예에 따른 수정 유클리드 알고리즘의 반복 연산에 관한 데이터 흐름을 나타내는 도면이다.

[0078] 도 7 및 하기 표 1을 참조하면, 단계(S700)에서 기본 셀 연산부(650)는 입력된 값($x \cdot S(x)$, x^{2t})을 이용하여 다항식을 초기화하고($F_0(x)=x \cdot S(x)$, $G_0(x)=x^{2t}$, $H_0(x)=x$, $I_0(x)=0$), 단계(S710)에서 기본 셀 연산부(650)는 스테이지를 '1'로 설정하고, i를 '0'으로 설정하고, 루프를 '1'로 설정한다(Stage=1, i=0, loop=1)

[0079] 단계(S720)에서 기본 셀 연산부(650)는 $F_i(x)$, $G_i(x)$, $H_i(x)$, $I_i(x)$에 대한 교환, 쉬프트, 다항식 연산을 수행하고, 단계(S730)에서 i를 '1' 증가시키고, 루프도 '1' 증가시킨다(i=i+1, Loop=loop+1).

[0080] 단계(S740)에서 기본 셀 연산부(650)는 루프가 '2'보다 큰지 판단한다.

[0081] 상기 루프가 '2'보다 크지 않은 경우, 기본 셀 연산부(650)는 단계(S720)부터의 동작을 다시 수행한다.

[0082] 상기 루프가 '2'보다 큰 경우, 단계(S750)에서 기본 셀 연산부(650)는 루프를 '1'로 설정하고, 스테이지를 '1' 증가시킨다(loop=1, Stage=Stage+1).

[0083] 단계(S760)에서 기본 셀 연산부(650)는 스테이지가 정해진 횟수(t)보다 큰지 여부를 판단한다.

[0084] 상기 스테이지가 상기 정해진 횟수보다 크지 않은 경우, 기본 셀 연산부(650)는 단계(S720)부터의 동작을 다시 수행한다.

[0085] 상기 스테이지가 상기 정해진 횟수보다 큰 경우, 단계(S770)은 기본 셀 연산부(650)는 최종 결과값, 오류값 다항식과 오류 위치 다항식으로 $F_i(x)=\omega(x)$, $H_i(x)=\sigma(x)$를 출력한다.

표 1

[0086]

```
Input : x · S(x), x^2t
Initialization : F0(x)=x · S(x), G0(x)=x^2t, H0(x)=x, I0(x)=0;
Index 'i' is initialized to 0;
Start algorithm:
for stage=1 step 1 until t
begin
Step1 : for loop=1 step 1 until 2
        begin
        i=i+1;
        Step2 : if (stop?1)
                begin
                Step2.1 : if(sw=1)
                          begin
                            swap(Fi-1(x),Gi-1(x));
                            swap(Hi-1(x),Ii-1(x));
                            swap(ai-1,bi-1);
                          end
                Step2.2 : else if (Sht=1)
                          begin
                            ai-1=0;bi-1=1;
                          end
                Step2.3 : Fi(x)=bi-1Fi-1(x)-ai-1Gi-1(x);
                Step2.4 : Gi(x)=Gi-1(x);
                Step2.5 : Hi(x)=bi-1Hi-1(x)-ai-1Ii-1(x);
                Step2.6 : Ii(x)=Ii-1(x);
                end
         Step3 : else
                 begin
                 Step3.1 : Fi(x)=Fi-1(x);Gi(x)=Gi-1(x);
                 Step3.2 : Hi(x)=Hi-1(x);Ii(x)=Ii-1(x);
                 end
        end
end
Output: σ(x), ω(x);
```

[0087] 도 8은 본 발명에 따른 오류 위치 다항식과 오류값 다항식을 계산하는 지연 시간을 나타내는 도면이다.

[0088] 도 8을 참조하면, KES 블록에서 필요한 총 계산 시간은 '(2m+SR)·(t-1)+2m'로 계산되며, m은 제1 부분 셀(651) 및 제2 부분 셀(652)의 파이프라인 단계이며,'SR'은 출력 저장부(630)의 레지스터 수, 't'는 RS 코드의 최대 오류 정정 능력을 나타내기도 하고 폴딩 방식 KES 블록 내부의 기본 셀의 반복 연산 횟수를 나타낸다. 따라서, 본 발명에 따른 폴딩 방식 KES 블록의 계산 시간은 'm', 'SR', 't' 세가지 요소에 의해 결정될 수 있는데 이는 KES 블록의 계산 시간이 정해짐에 따라 RS 디코더 전체 계산 시간에도 영향을 줄 수 있다.

[0089] 출력 저장부(630)는 기본 셀 연산부(650)의 출력 값을 순차적으로 저장함으로써 제1 부분 셀(651)의 연산 수행이 끝날 때까지 입력을 지연시키는 역할을 수행한다.

[0090] 제1 부분 셀(651) 및 제2 부분 셀(652)은 차수 비교 및 계산 부분을 대신하는 제어 신호를 생성함으로써 제어 신호가 차수의 정보를 의미할 수 있기 때문에 출력 저장부(630)의 구성으로 인해 기본 셀 연산부(650)의 출력이 다시 입력으로 재귀하여 쉬프트 연산, 다항식 연산, 교환 연산을 올바르게 수행할 수 있다.

[0091] t개 이하의 오류가 발생한 경우, t보다 적은 수의 반복 연산으로 오류 위치 다항식과 오류 값 다항식을 얻을 수 있지만 차수 비교 및 계산 부분이 제거된 기본 셀을 사용하기 때문에 t개의 오류가 발생한 경우와 같은 반복 횟수의 연산으로 오류 위치 다항식과 오류 값 다항식을 얻을 수 있다.

[0092] 이와 같이, 본 발명의 일실시예에 따른 종래의 차수 계산 및 비교 부분이 제거된 수정 유클리드 알고리즘 연산 회로의 기본 셀을 이용하여 KES 블록의 새로운 폴딩 방식의 연산 회로 및 그 방법을 제시함으로써 효율적인 기본 셀의 연산 수행으로 획기적으로 하드웨어 복잡도를 감소시킬 수 있다.

[0093] 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.

[0094] 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.

도면의 간단한 설명

[0095] 도 1은 종래의 리드-솔로몬 복호기의 구조를 도시한 도면이다.

[0096] 도 2는 기본 셀의 출력을 다시 입력으로 재귀하여 하나의 기본 셀을 반복적으로 사용하는 종래의 구조를 도시한 도면이다.

[0097] 도 3은 차수 계산이 필요 없는 수정 유클리드 알고리즘의 종래의 연산 회로를 도시한 도면이다.

[0098] 도 4는 도 3의 차수 계산이 필요 없는 수정 유클리드 알고리즘의 제어 신호를 생성하기 위한 각 경우별 다항식의 연산을 도시한 도면이다.

[0099] 도 5는 정지 신호 생성을 위한 유한 상태 기계를 도시한 도면이다.

[0100] 도 6은 본 발명의 일실시예에 따른 폴딩 방식 수정 유클리드 알고리즘의 연산 회로 구조를 나타내는 도면이다.

[0101] 도 7은 본 발명의 일실시예에 따른 수정 유클리드 알고리즘의 반복 연산에 관한 데이터 흐름을 나타내는 도면이다.

[0102] 도 8은 본 발명에 따른 오류 위치 다항식과 오류값 다항식을 계산하는 지연 시간을 나타내는 도면이다.

[0103] <도면의 주요 부분에 대한 부호의 설명>

[0104] 600: 폴딩 방식 수정 유클리드 알고리즘의 연산 회로

[0105] 610: 입력 신호 선택부

[0106] 620: 출력 신호 선택부

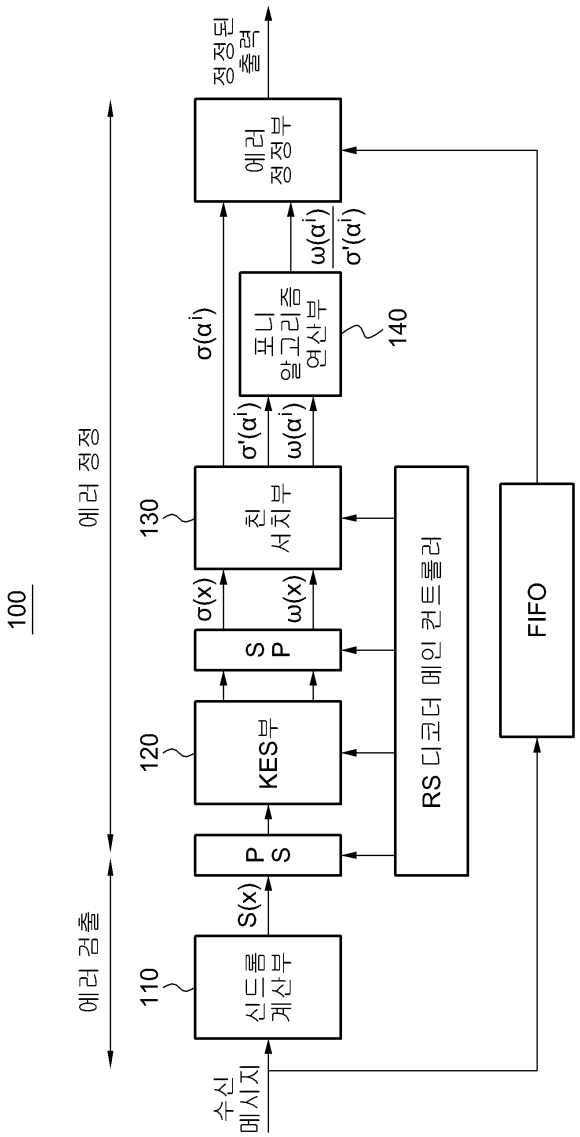
[0107] 630: 출력 저장부

[0108] 640: 연산 블록 제어 신호 생성부

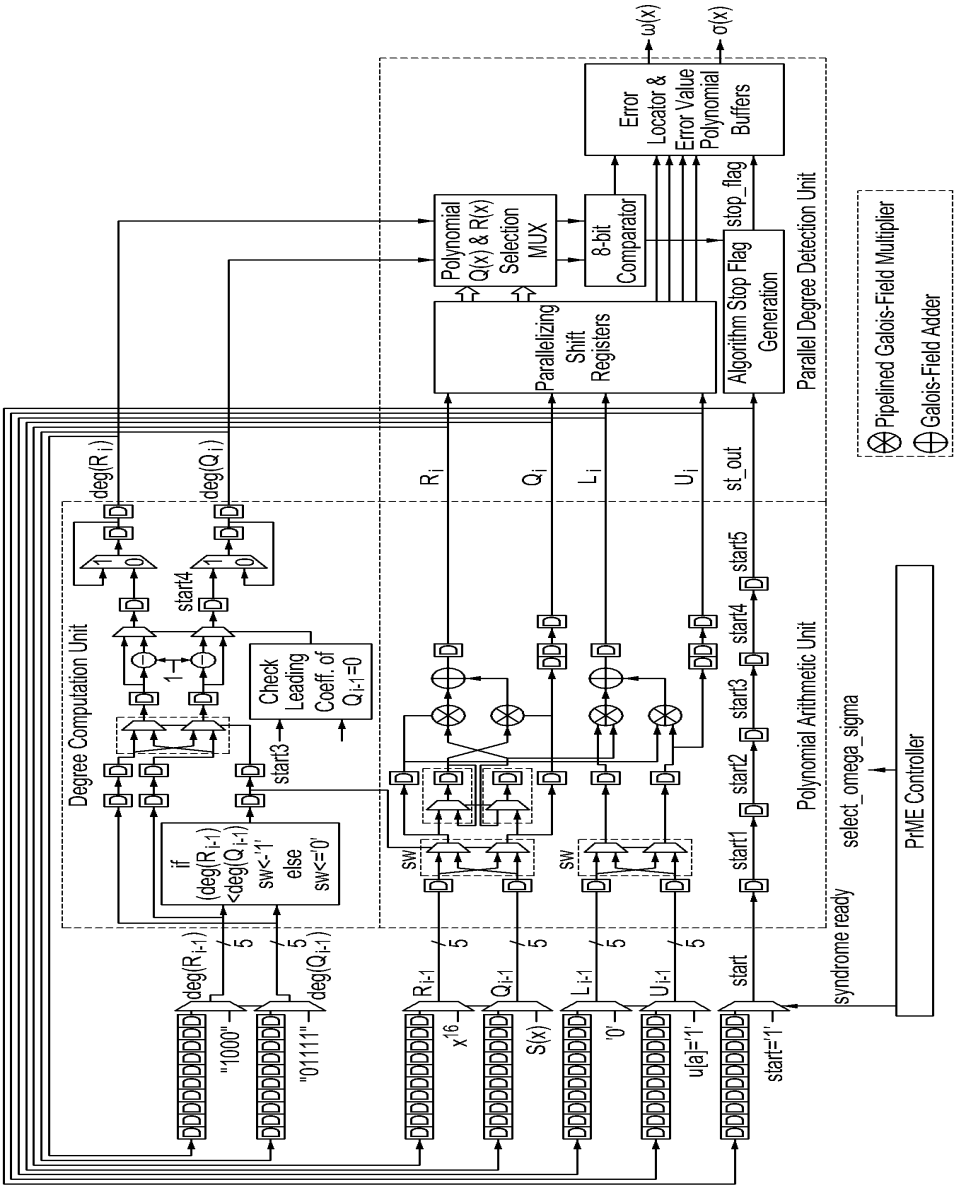
[0109] 650: 기본 셀 연산부

도면

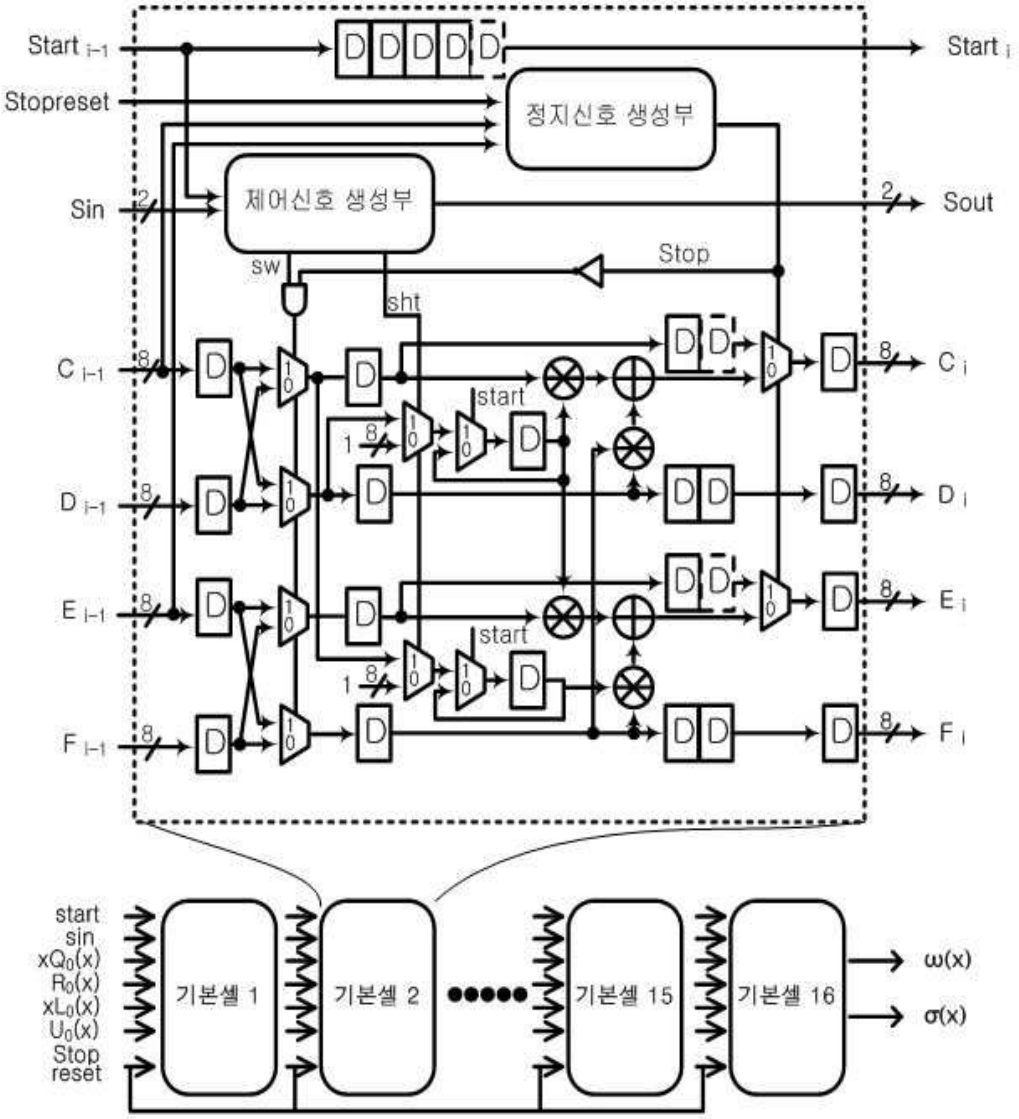
도면1


100
에러 검출
에러 정정
수신 메시지
110
신드롬 계산부
S(x)
P S
120
KES부
S P
σ(x)
ω(x)
130
친 서치부
σ'(αⁱ)
ω(αⁱ)
σ(αⁱ)
140
포니 알고리즘 연산부
ω(αⁱ)/σ'(αⁱ)
에러 정정부
정정된 출력
RS 디코더 메인 컨트롤러
FIFO

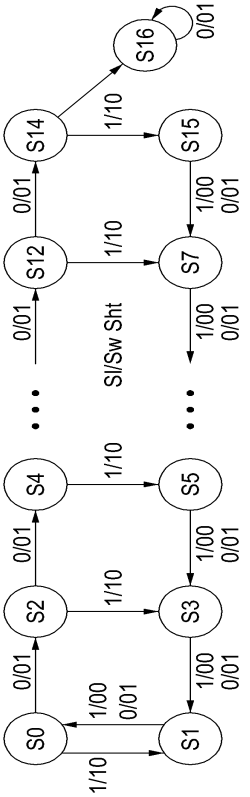
도면2


ω(x)
σ(x)
Error Locator & Error Value Polynomial Buffers
Polynomial Q(x) & R(x) Selection MUX
8-bit Comparator
stop_flag
Parallelizing Shift Registers
Algorithm Stop Flag Generation
Parallel Degree Detection Unit
Pipelined Galois-Field Multiplier
Galois-Field Adder
deg(R i)
deg(Q i)
Ri
Qi
Li
Ui
st_out
start4
start5
Degree Computation Unit
Check Leading Coeff. of Qi-1=0
start3
start2
start1
Polynomial Arithmetic Unit
if (deg(Ri-1) <deg(Qi-1) sw<-'1' else sw<='0'
sw
select_omega_sigma
PrME Controller
deg(Ri-1)
deg(Qi-1)
Ri-1
Qi-1
Li-1
Ui-1
start
syndrome ready
"1000"
"01111"
x16
S(x)
'0'
u[a]='1'
start='1'

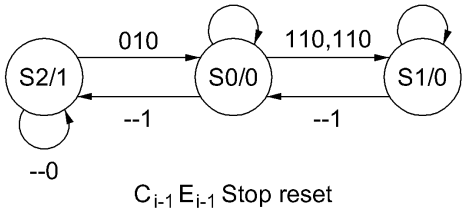
도면3


Start i-1
Stopreset
Sin
2
Start i
Sout
정지신호 생성부
제어신호 생성부
Stop
sw
sht
C i-1
D i-1
E i-1
F i-1
8
C i
D i
E i
F i
start
1
start
sin
xQ0(x)
R0(x)
xL0(x)
U0(x)
Stop
reset
기본셀 1
기본셀 2
기본셀 15
기본셀 16
ω(x)
σ(x)

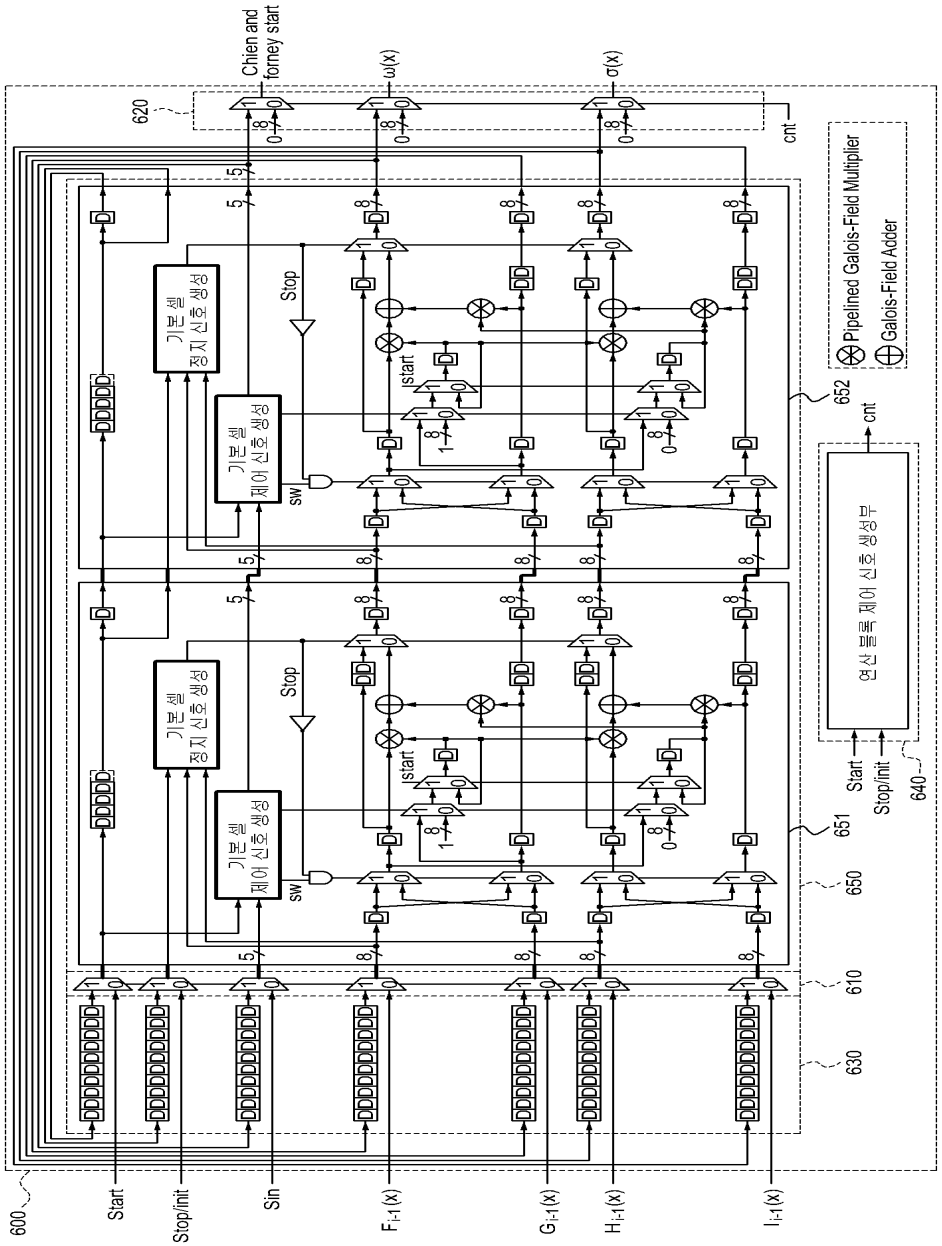
도면4


S16
0/01
S14
1/10
S15
0/01
1/00
0/01
S12
1/10
S7
0/01
1/00
0/01
Sl/Sw Sht
S4
1/10
S5
0/01
1/00
0/01
S2
1/10
S3
0/01
1/00
0/01
S0
1/00
0/01
S1
1/10

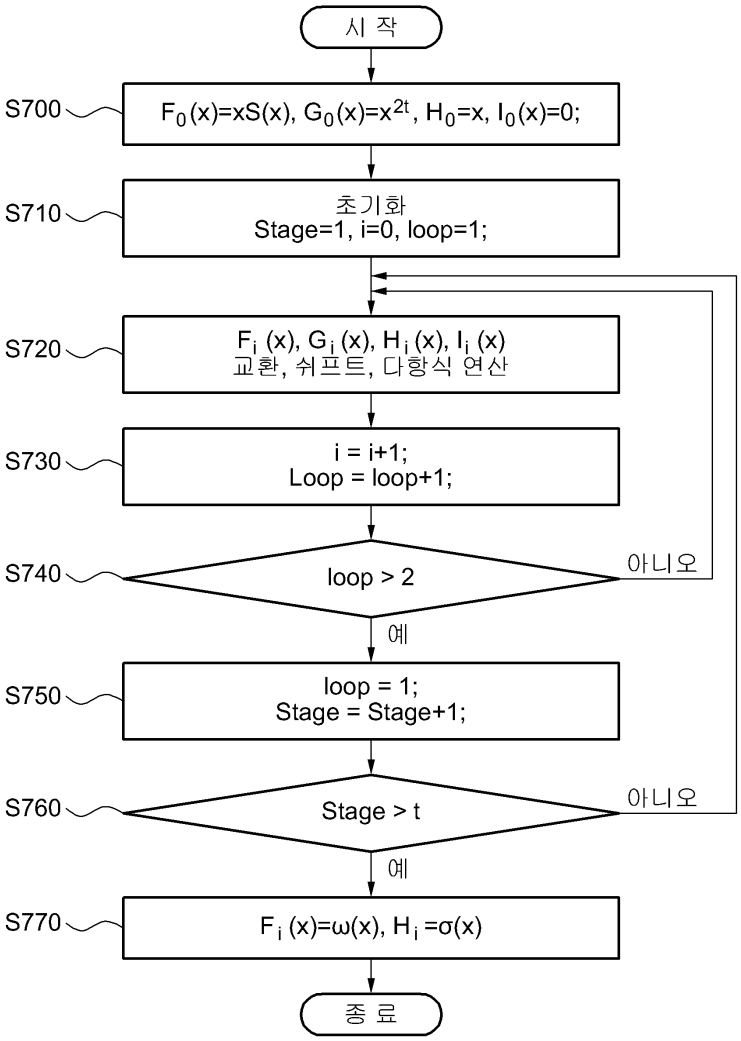
도면5


S2/1
010
S0/0
110,110
S1/0
--1
--1
--0
C_{i-1} E_{i-1} Stop reset

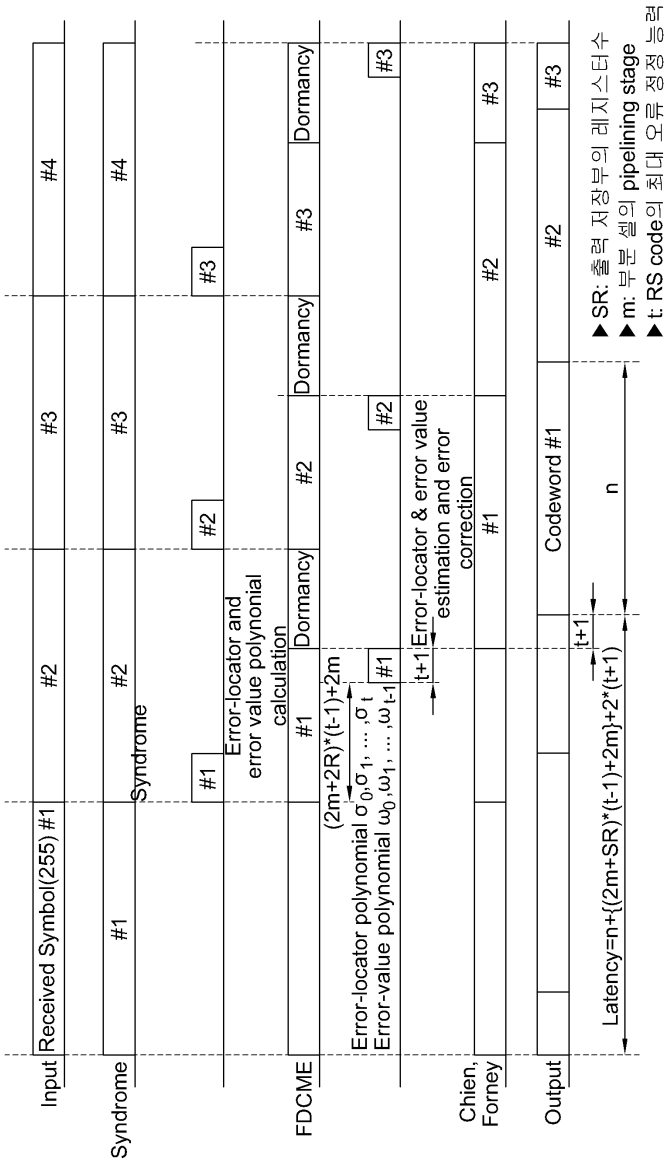
도면6


Chien and forney start
ω(x)
σ(x)
620
cnt
Pipelined Galois-Field Multiplier
Galois-Field Adder
652
기본셀 정지 신호생성
기본셀 제어 신호생성
Stop
sw
start
651
650
610
630
연산 블록 제어 신호 생성부
Start
Stop/init
640
600
Sin
F[-1](x)
G[-1](x)
H[-1](x)
I[-1](x)

도면7


시 작
S700
$F_0(x)=xS(x), G_0(x)=x^{2t}, H_0=x, I_0(x)=0;$
S710
초기화
Stage=1, i=0, loop=1;
S720
$F_i(x), G_i(x), H_i(x), I_i(x)$
교환, 쉬프트, 다항식 연산
S730
i = i+1;
Loop = loop+1;
S740
loop > 2
아니오
예
S750
loop = 1;
Stage = Stage+1;
S760
Stage > t
아니오
예
S770
$F_i(x)=\omega(x), H_i=\sigma(x)$
종 료

도면8


Input
Syndrome
FDCME
Chien, Forney
Output
Received Symbol(255) #1
#2
#3
#4
Syndrome
Error-locator and error value polynomial calculation
Dormancy
Error-locator polynomial σ0,σ1, ... ,σt
Error-value polynomial ω0,ω1, ... ,ωt-1
(2m+2R)*(t-1)+2m
t+1
Error-locator & error value estimation and error correction
Codeword #1
Latency=n+{(2m+SR)*(t-1)+2m}+2*(t+1)
n
▲ SR: 출력 저장부의 레지스터수
▲ m: 부분 셀의 pipelining stage
▲ t: RS code의 최대 오류 정정 능력