논문 2008-2-3

확장논리에 기초한 순차디지털논리시스템 및 컴퓨터구조에 관한 연구

A Study on Sequential Digital Logic Systems and Computer Architecture based on Extension Logic

박춘명*

Chun-Myoung Park*

요약

본 논문에서는 2진논리의 확장을 Galis체상에서 해석하여 확장논리에 기초한 순차디지털논리시스템과 컴퓨터구조의 핵심인 연산알고리즘을 논의하였다. 순차디지털논리시스템은 Building Block으로서 T-gate를 사용하였으며, 차순상태함수, 출력함수를 도출하여 최종 궤환이 없는 Moore Model의 순차디지털논리시스템을 구성하였다. 그리고, 컴퓨터구조에서 중요한 연산알고리즘의 핵심인 가산, 감산, 승산 및 제산 알고리즘을 유한체의 수학적 성질을 토대로 각각 도출하였다. 특히, 유한체 GF(P^m)상에서 P=2인 경우는 기존의 2진디지털논리시스템에 적용이 용이하다는 장점이 있으며, mod2의 성질에 의해 감산 알고리즘은 가산 알고리즘과 동일하다. 제안한 방법은 기존의 2진논리를 확장할 수 있어 좀 더 효율적으로 디지털논리시스템을 구성할 수 있을 것으로 사료된다.

Abstract

This paper discuss the sequential digital logic systems and arithmetic operation algorithms which is the important material in computer architecture using analysis and synthesis which is based on extension logic for binary logic over galois fields. In sequential digital logic systems, we construct the moore model without feedback sequential logic systems after we obtain the next state function and output function using building block T-gate. Also, we obtain each algorithms of the addition, subtraction, multiplication, division based on the finite fields mathematical properties. Especially, in case of P=2 over GF(P^m), the proposed algorithm have a advantage which will be able to apply traditional binary logic directly.The proposed method can construct more efficiency digital logic systems because it can be extended traditional binary logic to extension logic.

Key Words : extension logic, sequential digital logic, arithmetic operation, finite fields

Ⅰ. 서 론

21세기의 세계 IT산업은 아나로그 기술의 퇴조와 더불어 디지털 기술과 인터넷 확산으로 반도체, 디지털 가전, 컴퓨터, 방송 등 다양한 산업분야가융합되어 새로운 부가가치를 창출하는 디지털 컨버젼스와 인간 중심의 Anytime, Anywhere, Anynetwork, Anydevice, Anyservice를 지향하는 유비쿼터스 컴퓨팅으로 변화 될 것이다.[1-3] 이러한 유비쿼터스 컴퓨팅 환경에서의 방대한 각종

*정회원: 충주대학교 컴퓨터공학과

접수일자: 2007.11.12, 수정완료일자: 2008.04.07

정보를 효과적으로 다루기 위해 현재의 2진디지털논리의 확장논리개념이 대두되기 시작하였다. 한편, 확장논리를 분석, 해석 및 종합하는 데는 여러 가지 수학적 배경이 요구되지만 그 중 일명 Galois체라고 하는 유한체를 사용하면 좀 더 효율적으로 분석, 해석 및 종합하기가 용이하다.[4-5] 특히 P가 소수이고 m이 양의 정수인 GF(P^m)상에서 P=2이고 m=1인 경우는 현존의 2진논리에 기본을 둔 2진논리디지털시스템의 근간이 되며 대수학적으론 부울대수가 이 범주에 속한다. 즉, P=2인 경우는 기존의 2진디지털논리시스템에 적용이 용이하다는 장점이 있다.[6-7] 본 논문에서는 2진논리의 확장을 Galis체상에서 해석하여 차세대 디지털논리시스템에서의 궤환이 없는 순차디지털논리시스템을 구성하였으며, 또한 컴퓨터구조의 핵심인 연산알고리즘을 논의하였다.

II. 순차디지털논리시스템구성

순차디지털논리시스템의 출력은 조합논리디지털시스템과는 달리 현재의 입력뿐만 아니라 과거의 입력에 의해서 결정된다. 따라서 현재의 입력은 지연소자 또는 기억소자에 의해서 그 정보가 입력단으로 궤환되어 다음 시간의 입력과 더불어 입력으로 동작을 하여야 한다. 일반적으로 순차디지털논리시스템은 다음 식(1)과 같은 5-tuple로 표시된다.

$$M=(S, I, Z, \delta, \lambda) \qquad (1)$$

여기서 각각 S는 상태, I는 입력, Z는 출력, δ는 차순상태함수, λ는 출력함수이다. 또한, S, I, $Z=e_i \in GF(P^m)$ (i=0, 1, 2, ... , P^m-1)이다.

한편, 위 식(1)의 δ는 다음 식(2)와 같은 사상(mapping) 관계를 가진다.

$$\delta : S_t \times I \longrightarrow S_{t+1} \qquad (2)$$

여기서 S_t는 현재상태이고 S_{t+1}은 차순상태이다.

또한, λ는 출력이 오직 현재상태의 함수로만 이루어지는 Moore model인 경우에 다음과 같이 표시된다.

$$\lambda : S_t \longrightarrow Z_t \qquad (3)$$

본 논문에서는 Moore model의 궤환이 없는 경우의 순차디지털논리시스템구성을 다룬다.

1. T-gate 구성

본 논문에서의 순차디지털논리시스템구성시에 사용되는 Building Block의 기본 논리소자는 T-gate이며 이에 대한 블럭도는 다음 그림1과 같다. 또한, 궤환이 없는 경우에 R는 V_k이며, V_k는 상태 digit code이고 k=0, 1, 2, ... , m-1이다.

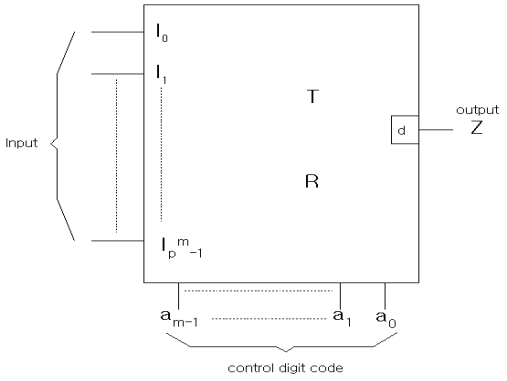

그림 1. 순차디지털논리시스템구성에 사용되는 Building Bolck T-gate의 블럭도
Fig. 1. The block diagram of the Building Block T-gate that is used constructing the sequential logic digital systems.

여기서, I_i는 입력, Z는 출력이다. 또한, I_i, $Z=e_i \in GF(P^m)$(i=0, 1, 2, ... , P^m-1)이고 a_j(j=0, 1, 2, ... , m-1)는 제어 digit code이며, $a_j \in \{0, 1, 2, ... , (P-1)\}$이다.

2. 궤환이 없는 순차디지털논리시스템구성

가. 차순상태함수

순차디지털논리시스템을 구성할 때 차순상태함

수를 구하는 것은 중요하다. 또한 순차디지털논리 시스템에서는 현재상태와 차순상태를 동시에 다루어야 하므로 시간 t를 사용하여 차순상태함수를 표현하면 다음 식(4)와 같이 표현 할 수 있다.

$$S_{t+1}=\delta(S_t, I_t) \quad (4)$$

여기서 S_{t+1}, S_t, $I_t=e_i \in GF(P^m)(i=0, 1, 2, \ldots\ldots, P^m-1)$이다.

한편, 상태 S를 $GF(P^m)$내 원소들을 P치 digit code 할당 알고리즘에 따라 상태 digit code $V_k(k=0, 1, 2, \ldots\ldots, m-1)$로 할당할 수 있으며 이는 현존의 2진논리에서의 각 상태를 bit code로 code화 한 것과 개념이 유사하다. 그러므로 상태 digit code로 각 상태를 표시한 후 상태 digit code 별로 그 값이 1에서부터 (P-1)까지에 해당하는 상태식을 구한다. 이때 상태와 상태의 합은 modP 합이다. 특히, P=2인 $GF(2^m)$상의 가산은 mod2 합이므로 이는 Exclusive-OR 연산으로 가능하다. 따라서 $GF(P^m)$상에서의 상태 digit code V_k는 $V_{m-1}, V_{m-2}, \ldots\ldots, V_1, V_0$로 생성되며 V_k 값에 따른 상태식을 구하면 다음 식(5)와 같다.

$$V_{kt}=(S_0 + S_1 + \ldots\ldots + S_{\zeta-1})_t \quad (5)$$

여기서, ζ는 P^m-1이고, k=0, 1, 2, ... , m-1이며 +는 modP 승이다.

또한, 차순상태함수는 현재상태와 이전상태와의 관계를 표시하는 전순상태표로 부터 앞에서 구한 상태식 (5)를 토대로 다음 식(6)을 얻을 수 있다.

$$\begin{aligned} S(V_k)_{t+1} &= (S_0 + S_1 + \ldots\ldots + S_{\zeta-1})_t \cdot I_0 \\ &+ (S_0 + S_1 + \ldots\ldots + S_{\zeta-1})_t \cdot I_1 + \ldots\ldots\ldots\ldots \\ &+ (S_0 + S_1 + \ldots\ldots + S_{\zeta-1})_t \cdot I_{\zeta-1} \\ &= \Big(\sum_{i=0}^{P^m-1} S_i\Big)_t \cdot I_0 + \Big(\sum_{i=0}^{P^m-1} S_i\Big)_t \cdot I_1 + \ldots\ldots\ldots\ldots \\ &+ \Big(\sum_{i=0}^{P^m-1} S_i\Big)_t \cdot I_{\zeta-1} = \sum_{j=0}^{P^m-1} \Big(\sum_{i=0}^{P^m-1} S_i\Big)_t \cdot I_j \end{aligned} \quad (6)$$

여기서, ζ는 P^m-1이고, V_k는 상태 digit code(k=0, 1, 2, ... , m-1)이고, $V_k \in \{0, 1, 2, \ldots\ldots, P-1\}$이며, $S_i=e_i \in GF(P^m)(i=0, 1, 2, \ldots\ldots, P^m-1)$이다. 또한, Σ, +는 modP 합이다. 또한, S_i는 V_k의 값에 따른 상태이다.

나. 출력함수

출력함수는 상태천이도에서 Goal 상태 modP 합으로 표현할 수 있다.

다. 순차디지털논리시스템의 회로실현

순차디지털논리시스템의 회로는 다음 그림 2와 같은 일반적인 블럭도로 표시할 수 있다.

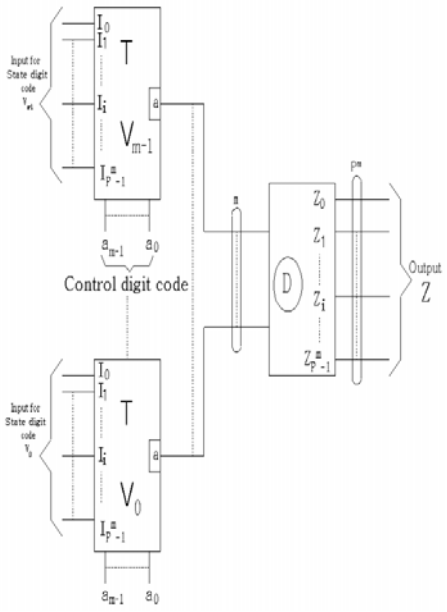

그림 2. 일반적인 $GF(P^m)$상의 순차디지털논리시스템의 회로 블럭도

Fig. 2. The block diagram of the generalized sequential logic digital systems over $GF(P^m)$.

라. 순차디지털논리시스템구성 알고리즘

[단계1] 상태천이도에 나타난 상태를 Ⅱ장의 digit code 할당 알고리즘에 의해 상태 digit code V_k로 할당하여 표로 작성한다.

[단계2] 단계1에서 구한 표에서 상태 digit code V_k에 따른 상태식을 구한다.

[단계3] 상태천이도로부터 전순표를 구한다.

[단계4] 단계3에서 구한 전순표로부터 단계2의 상태식을 토대로 차순상태함수를 구하고, 상태천이도의 Goal 상태로부터 출력함수를 구한다.

[단계5] 단계4까지에서 구한 내용과 그림2를 토대로 최종 순차디지털논리시스템의 회로를 실현한다.

Ⅲ. 연산알고리즘

1. 수학적 배경

[P1] 다음 식(7)을 인수분해하여 m차 기약다항식을 구한 후 이를 0으로 하는 한 근을 α라 할 때 식(8)과 같은 원시기약다항식을 얻을 수 있으며, 이를 벡터공간으로 표시하면 다음 식(9)와 같다.

$$X^{\beta}-X=X(X-1)(X^{\beta-2}+X^{\beta-3}+.....+X+1)=0 \qquad (7)$$

여기서 $\beta=P^m$, P 는 소수, m 은 양의정수

$$F(\alpha)=\sum_{i=0}^{m-1}\delta_i\alpha^i$$

$$=\delta_0+\delta_1\alpha^1+.....+\delta_{m-2}\alpha^{m-2}+\delta_{m-1}\alpha^{m-1} \qquad (8)$$

$$\underline{F(\alpha)}=[\delta_0\delta_1.....\delta_{m-2}\delta_{m-1}] \qquad (9)$$

여기서 δ_i(i=0,1, ... ,m-1)는 α^i의 계수이다.

[P2] $GF(P^m)$상에서 원소생성은 식(8)을 0으로 하는 한 근 α의 멱승으로 구해지며, 원소의 개수는 P^m개이며 다음 식(10)과 같다.

$$GF(P^m)=\{0,\alpha^1,\alpha^2,.....,\alpha^{\beta-2}=\alpha^{-1},\alpha^{\beta-1}=1\} \qquad (10)$$

여기서 $\beta=P^m$ 이다.

[P3] 역원의 존재

(1) $\Theta+(-\Theta)=0$인 가법에 대한 역원 $-\Theta$가 존재한다.($\forall\Theta\in GF(P^m)$)

(2) $\Theta.(\Theta^{-1})=1$인 승법에 대한 역원 $\Theta-1$이 존재한다.($\forall\Theta\in GF(P^m)$)

[P4] $(\Theta+\Psi)\beta=\Theta^{\beta}+\Psi^{\beta}=\Theta+\Psi$ ($\forall\Theta,\Psi\in GF(P^m)$)

[P5] $\Theta i.\Theta j=\Theta^{i+j(\mathrm{mod}\beta-1)}$ ($\beta=P^m$)

여기서 r=i+j이라 하면 i+j(mod β-1)은 r(mod β-1)이며 $0\le r\le P^m-1$이다.

이상의 수학적 성질과 그 외의 본 논문을 전개하는데 필요한 수학적 성질은 참고문헌[8-10]을 참고하였다.

2. 가산 및 감산 연산알고리즘

피가산원소를 e_i, 가산원소를 e_j, 가산후원소를 e_a라 하고 이들을 벡터공간으로 표현한 것을 각각 $\underline{e_i}(a_V)$, $\underline{e_j}(b_V)$, $\underline{e_a}(A_V)$라 하면 두 원소 e_i와 e_j의 가산은 다음 식(11)과 같다.

$$e_i\oplus e_j=\underline{e_i}(a_V)\oplus\underline{e_j}(b_V)=\underline{e_a}(A_V) \qquad (11)$$

여기서 i,j=0,1,.....,2^m-2,2^m-1이고, $a_V,b_V,A_V\in GF(2)$ (V=0,1,.....,m-2,m-1)이고, ⊕는 modP 합이다.

특히, P=2인 경우에는 식(11)에서 살펴 본 바와 같이 $A_V=a_V\oplus b_V$ 이다. 따라서 b_V를 가산연산시의 제어입력으로 사용하면 b_V 값에 따라 a_V 값을 그대로 유지하거나 2의 보수를 취한 값이 되고, 이는 mod2 합의 수학적 성질과 같으며, 식(12)와 같다.

$$A_V = \begin{cases} a_V & \text{iff } b_V=0 \\ a_V' & \text{iff } b_V\neq 0 \end{cases} \qquad (12)$$

따라서, P=2인 경우의 가산알고리즘을 도출하면 다음과 같다.

[단계1] 피가산원소 e_i와 가산원소 e_j를 각각 비트벡터공간으로 표현한 $\underline{e_i}(a_V)$와 $\underline{e_j}(b_V)$로 표시한다.

[단계2] 가산원소 $\underline{e_j}(b_V)$를 제어입력으로 사용하여 b_V의 값이 "0"이면 피가산원소 $\underline{e_i}(a_V)$의 해당 비트값을 그대로 유지하고 "1"이면 2의 보수를 취한다.

[단계3] STEP2를 행한 후의 결과가 최종 가산후의 원소 $\underline{e_a}(A_V)$가 된다.

P=2인 경우의 감산연산은 Mod2의 수학적 성질에 의해 가산연산과 같다.

3. 승산 연산알고리즘

피승산원소를 e_i, 승산원소를 e_j, 승산후원소를 e_M이라 하고 이들을 벡터공간으로 표현한 것을 각각 $\underline{e_i}(aV)$, $\underline{e_j}(bV)$, $\underline{e_M}(M_V)$라 하면 두 원소 e_i와 ej의 승산은 다음 식(13)과 같다.

$$e_i \otimes e_j = \underline{e_i}(a_V) \otimes \underline{e_j}(b_V) = \underline{e_m}(M_V) \quad (13)$$

한편, 피승산원소, 승산원소, 승산후원소의 기약다항식을 각각 $F(\alpha)=\sum_{i=0}^{m-1} a_i\alpha^i$, $G(\alpha)=\sum_{j=0}^{m-1} b_j\alpha^j$와 $H(\alpha)=\sum_{k=0}^{m-1} M_k\alpha^k$라 하면 식(13)은 다음 식(14)와 같이 표현 할 수 있다.

$$F(\alpha)\otimes G(\alpha)=\left(\sum_{i=0}^{m-1} a_i\alpha^i\right)\otimes\left(\sum_{j=0}^{m-1} b_j\alpha^j\right)$$
$$=\sum_{i=0}^{m-1}\left(\sum_{j=0}^{m-1} b_j\right)a_i\alpha^{i+j}=\sum_{i+j=0}^{2m-2} a_ib_j\alpha^{i+j} \quad (14)$$

여기서 a_i, $b_j \in GF(2)$이고 i,j=0,1,.....,m-2,m-1이다. 또한, r=i+j라 하면 식(14)는 식(15)와 같고 이는 H(α)와 같아야 한다.

$$F(\alpha)\otimes G(\alpha)=\sum_{r=0}^{2m-2} a_ib_j\alpha^r=H(\alpha)=\sum_{k=0}^{m-1} M_k\alpha^k \quad (15)$$

따라서 α^r의 r은 $m \le r1 \le 2m-2$ 부분과 $0 \le r2 \le m-1$ 부분으로 분할 할 수 있으며 α^{r1}항을 수학적 성질로부터 α^{r2}항으로 표현하여 α^k항과 일치시킬 수 가 있다. 또한, 이들 α^{r2}항들이 승산기 모듈 중 제어입력생성 모듈의 입력이 되고 이 제어입력 C_L에 의해 최종 승산후 원소 $\underline{e_m}(M_V)$를 얻는다. 이를 토대로 승산 알고리즘을 도출하면 다음과 같다.

[단계1] 피승산원소 e_i와 승산원소 e_j를 각각 비트벡터공간으로 표현한 $\underline{e_i}(a_V)$와 $\underline{e_j}(b_V)$로 표시한다.

[단계2] α^{r1}항과 α^{r2}항을 각각 구한다.

[단계3] α^{r1}항을 수학적 성질로부터 α^{r2}항으로 표현하여 제어입력 C_L을 구한다.

[단계4] 단계3에서 구한 C_L의 값이 "0"이면 해당 α^{r2}항의 비트값을 그대로 유지하고 "1"이면 2의 보수를 취한다.

[단계5] 단계4를 행한 후의 결과가 최종 승산후 원소 $\underline{e_m}(M_V)$가 된다.

한편, P=2인 경우의 제어입력 C_L(L=0,1,.....,m-2,m-1)은 식(15)의 α^{r1}항으로 부터 구할 수 있다. 즉, $\sum_{r1=m}^{2m-2} R_{r1}\alpha^{r1}$을 α^{r2}항으로 표현해 이들을 modP합함으로써 쉽게 구할 수 있으며 이를 식으로 표현하면 다음 식(16)과 같고 제어입력 C_L의 개수는 m개이다.

$$\sum_{r1=m}^{2m-2} R_{r1}\alpha^{r1}=\sum_{r1=m}^{2m-2} R_{r1}\left(\sum_{L=0}^{m-1}\alpha^L\right) \quad (16)$$

따라서 승산 연산은 α^r생성 부분과 제어입력C_L 생성 부분을 합성하여 구할 수 있으며 C_L의 값이 "0"이면 식(15)의 M_k 값은 R_{r2} 값을 유지하고 "1"이면 R_{r2} 값에 2의 보수를 취한 값이 되고 이를

식으로 표현하면 다음 식(17)과 같다.

$$M_k = R_{r2} \text{ iff } C_L=0$$
$$R_{r2}' \text{ iff } C_L=1 \quad (17)$$

여기서 M_k, C_L, Rr2, $Rr2' \in GF(2)$이고 k, L, r2=0,1,.....,m-2,m-1이다.

4. 제산 연산알고리즘

피제산원소, 제산원소 및 제산후원소를 각각 $\underline{e_i}(a_V)$, $\underline{e_j}(b_V)$, $\underline{e_d}(D_V)$라 하면 두 원소 e_i와 e_j의 제산은 다음 식(18)과 같다.

$$e_i \ominus e_j = e_i(a_V) \ominus e_j(b_V) = e_i \otimes e_j^{-1}$$
$$= \underline{e_i}(a_V) \otimes \underline{e_i}(b_V^*) = \underline{e_d}(D_V) \quad (18)$$

여기서 e_j^{-1}은 e_j의 역원이며 b_V^*는 b_V에 대한 역원비트벡터공간이다.

한편, 피제산원소, 제산원소 및 제산후원소의 원시기약다항식을 각각 F(α), G(α)와 H(α)라 하면 식(18)은 다음 식(19)와 같다.

$$F(\alpha) \ominus G(\alpha) = F(\alpha) \otimes [G(\alpha)]^{-1} = H(\alpha) \quad (19)$$

여기서 $[G(\alpha)]^{-1}$은 G(α)에 대한 승법역원생성다항식(multiplicative inverse element generation polynomial)이다. 특히, P=2인 경우의 $[G(\alpha)]^{-1}$는 식(20)과 같다.

$$[G(\alpha)]^{-1} = \left(\sum_{j=0}^{m-1} b_j \alpha^j \right)^{K-2} (\text{where, } K=2^m) \quad (20)$$

위의 내용을 토대로 제산 알고리즘을 도출하면 다음과 같다.

[단계1] 피제산원소 e^i와 제산원소 e_j를 각각 비트벡터공간으로 표현한 $\underline{e_i}(a_V)$와 $\underline{e_i}(b_V)$로 표시한다.

[단계2] 제산원소의 원시기약다항식 G(α)에 대한 승법역원생성다항식 $[G(\alpha)]^{-1}$과 역원비트벡터공간 b_V^*로 표시된 $\underline{e_i}(b_V^*)$를 구한다.

[단계3] 단계2에서 구한 $\underline{e_i}(b_V^*)$를 승산 알고리즘의 승산원소로 한다.

[단계4] 이 후 부터는 승산 알고리즘의 단계2 이후와 동일하다.

IV. 결 론

본 논문에서는 2진논리의 확장을 Galis체상에서 해석하여 확장논리에 기초한 순차디지털논리시스템과 컴퓨터구조의 핵심인 연산알고리즘을 논의하였다. 순차디지털논리시스템은 Building Block으로서 T-gate를 사용하였으며, 차순상태함수, 출력함수를 도출하여 최종 궤환이 없는 Moore Model의 순차디지털논리시스템을 구성하였다. 그리고, 컴퓨터구조에서 중요한 연산알고리즘의 핵심인 가산, 감산, 승산 및 제산 알고리즘을 유한체의 수학적 성질을 토대로 각각 도출하였다. 특히, 유한체 $GF(P^m)$상에서 P=2인 경우는 기존의 2진디지털논리시스템에 적용이 용이하다는 장점이 있으며, mod2의 성질에 의해 감산 알고리즘은 가산 알고리즘과 동일하다. 제안한 방법은 기존의 2진논리를 확장할 수 있어 좀 더 효율적으로 디지털논리시스템을 구성할 수 있을 것으로 사료된다.

참고문헌

[1] T. B. Fower,"Convergence in the Information Technology and Telecommunications World : Separating Reality from Hype," Telecommunications Review, pp.11-30, 2002.

[2] A. Henten, R. samarajiva, W. Melody,"Designing Next Generation Telecom. regulation : ICT Convergence or Multisector Utility. Report on WDR Dialogue, 2003.

[3] V. Coroama, J. Bohn, F. Matten,"Living in a Smart Environment-Implications for the Coming Ubiquitous Information Society," IEEE SMC

2004, The Hague, The Netherlands, pp.5633-5638,Oct., 10-13, 2004.
[4] E.Artin, *Galois Theory*, NAPCO Graphics arts, Inc., Wisconsin. 1971.
[5] M.davio, Jean-Pierre, Deschamps and A.Thayse, *Discrete and Switching Functions*, McGraw-Hill international Book company, 1978.
[6] M.D.Ercegovac and T.Lang, Digital *Systems and Hardware/Firmware Algorithms*, Wiley, 1985.
[7] R.J.McEliece, *Finite Fields for Computer Science and Engineers*, Kluer Academic publishers, 1987.
[8] FJ.A.Pineiro, J.D.Bruguera, F.lamberti and P.Moutuschi, A Radix-2 Digit-by-Digit Architecture for Cube Root," IEEE Trans Comput., pp.562-566,vol.57,No.4,Apr.,2008.
[9] P.Longa and A.miri, Fast and Flexible Elliptic Curve Point Arithmetic over Prime Fields," IEEE Trans Comput., pp.289-302, vol.57, No.3, March 2008.
[10] N.Petra,D.De Caro and A.G.M.Strollo,"A Novel Architecture for Galois Fields GF(2^m) Multipliers Based on Mastrovito Scheme," IEEE Trans. Comput., vol.56, No.11, Nov. 2007.

※ 이 논문은 충주대학교 대학구조개혁지원사업비(교육인적자원부 지원)의 지원을 받아 수행한 연구임.

저 자 소 개

박춘명(정회원) 제6권 제2호 참조