k-map상의 셀을 이용한 새로운 GRM 상수 생성 기법

정회원 이 철 우*, 차 문 철*, 김 흥 수*

A New Production method of GRM coefficients using k-map

Chol-U Lee*, Wenzhe Che*, Heung-Soo Kim* ***Regualr Members***

요 약

본 논문에서는 karnaugh map(k-map)상의 셀을 이용하여 2^n개의 서로 다른 극수(Polarity)를 갖는 GRM (Generalized Reed-Muller)상수를 생성하는 새로운 기법을 제안하였다. n개의 입력변수에 대한 일반적인 GRM 함수의 생성 방법은 단일 변수에 대한 변환 행렬을 구하고 이를 n번의 Kronecker 곱을 행한 변환 행렬을 이용하여 GRM 상수를 구하는 것이다. 이런 방법을 사용하는 경우, 변수의 숫자가 증가함에 따라 변환 행렬의 차수가 $2^n \times 2^n$로 커지는 단점을 갖는다. 이에 반하여 본 논문에서는 k-map상에서 변수를 축약시킨 셀 $[f_i]$을 구하고 이를 단일 변수 변환 행렬과 연산하여 GRM 상수를 구하는 새로운 기법을 제안한다. 본 논문에서 제안한 새로운 방법과 타 논문과의 비교를 한 결과, 기존 방법은 가산기, 승산기, KP(Kronecker 곱 승산기)회로가 필요한데 반하여 본 논문에서는 가산기만이 필요하므로 효율적인 VLSI 설계에 유리하다.

ABSTRACT

In this paper we propose a new method to derive GRM(Generalized Reed-Muller) coefficients for each 2^n *polarities using cell of karnaugh map(k-map).*

Generally, there are the serial and parallel method to derive GRM coefficients. As a serial method, Green method generates GRM coefficients using transform matrix. And as a parallel method, Besslich algorithm produces GRM coefficients of each polarity using the generated anteriorly.

Green's method generates GRM coefficients for n-variable by calculating transform matrix for one-variable and n-times kronecker product this matrix. And Besslich's method generates GRM coefficients of each polarity in order of Grey-code. But those methods have disadvantages that the number of variable exceeding four makes transform matrix large and there are so many operation steps.

In this paper, GRM coefficients is generated by producing cell $[f_i]$ minimizing variable on k-map and operating this cell $[f_i]$ and transform matrix for one-variable. So, we can generate GRM coefficients of all polarities easily by using the proposed method.

Ⅰ. 서 론

최근 집적회로 기술의 비약적인 발전에 따라 회로가 복잡해지고, 집적도 등이 증가하게 되었다. 이와 같은 문제는 칩의 VLSI화에 따른 내부 상호 연결선들의 증가로 인하여 신호지연, 전력소비, 잡음 등의 증가를 야기 시켰다.[1-3]

논리 회로 설계에서 함수는 입력 값의 조합에 의해 출력 값이 주어지는 진리표를 일반화한 연산영역(Operational domain)과 입력변수를 함수적으로 표현한 함수영역(Function domain)에서 해석이 가능하며 영역사이의 변환은 Reed-Muller(RM)전개식을

* 인하대학교 전자공학과 회로 및 시스템 연구실 (martin2k@nate.com)
논문번호 KICS2005-06-229, 접수일자 : 2005년 6월 2일

이용한다. 또한, RM함수는 극수에 따라 서로 다른 형태의 값을 갖는 함수식으로 구성되는데, 모든 극수의 함수를 표현하는 것을 일반화된 Reed Muller (Generalized Reed Muller : GRM)변환이라 한다.[4-5]

Besslich[6]와 Wu[7]등은 GRM계수의 생성 방법으로 극수의 변화에 따라서 일정한 모양으로 변하는 계수의 관계를 맵(map)상에서 해석하여 순차적인 GRM계수를 구하는 방법에 관하여 연구하였다. 그러나 맵을 이용한 방법은 변수의 개수가 증가하면 맵이 복잡해져서 맵 상에서 함수들의 상호관계를 표현하기 어렵다는 단점이 있다. Green[8-9]은 2진 함수에 대하여 Kronecker 곱에 의한 전달행렬을 구하고 이를 적용하여 RM변환과 GRM계수를 구하였는데 이 방법은 연속적으로 극수 생성이 가능하며 계수 생성 연산식이 간단하고, 특히 3치와 4치 함수에 대하여 효과적인 GRM계수 생성 방법을 제안하였다. Green이 제안한 방법은 전달행렬을 이용한 직렬형의 출력을 갖는 GRM계수의 생성 방법으로 직렬형의 알고리즘은 계수 생성에 있어서 많은 수의 연산자가 필요하며 계수의 생성 시간도 상대적으로 오래 걸린다는 단점이 있다. 이러한 문제에 대한 해결을 위하여 Hong[10]등은 *GF(3)*상에서 단일변수에 대한 극수 변환 과정을 확장하여 n변수에 대한 GRM 계수를 구하였으며 출력의 형태는 병렬형의 출력을 갖는 계수 생성 알고리즘을 제안하였다. 이 밖에도 Sasao[11-12]등에 의하여 결정도(Decision diagram)를 이용하여 RM 계수를 구하는 방법이 제시되었다. Miller[13]는 BDD (Binary decision diagram)의 확장 개념을 결정도에 적용한 MDD (Multiple-valued decision diagram)를 이용하여 GRM 계수를 구하는 방법도 제안하였다. MDD를 이용하면 결정도의 장점인 회로 구현의 용이성, 점검의 용이성 등을 쉽게 적용할 수 있지만 연속적인 GRM계수의 생성이 어려운 단점이 있다. 따라서 본 논문에서는 k-map상에서 변수를 축약시키면서 얻어지는 셀(cell) $f[i]$을 단일변수 변환 행렬과 연산하여 GRM 상수의 극수를 구하는 새로운 알고리즘을 제안하였다.

Ⅱ. Reed-Muller(RM) 상수와 Generalized Reed-Muller(GRM)상수

본 장에서는 RM 계수와 GRM 계수의 수식적인 생성 방법에 대하여 논의한다.

2.1 RM 상수의 생성

일반적으로 논리회로에서는 입력함수에 대한 출력 값을 일반화한 진리표로 나타나는 연산영역에서 함수의 해석과 입력변수를 함수적으로 표현하여 함수 영역에서 해석이 가능하며 두 영역 사이의 변환 과정을 RM변환이라 한다. 일반적으로 RM변환에 의하여 연산영역의 데이터를 함수영역으로 변환하는 과정은 변환을 위한 전달행렬을 먼저 구하고 연산영역의 데이터를 이용한 행렬연산을 행하여 함수영역의 데이터를 얻을 수 있다. 역으로 함수영역의 데이터를 연산영역의 값으로 변환할 때에는 역행렬을 이용하여 구하면 연산영역에서의 값을 얻을 수 있다. 이를 그림으로 표현하면 그림 1과 같다.

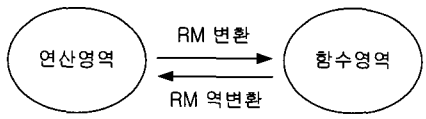


그림 1. RM전개식에 의한 영역사이의 변환
Fig. 1. Exchanges of domain for RM expansion.

단일변수 부울 함수식을 단일변수 RM 전개식으로 변환할 때의 계수 선택 행렬을 나타내기 위해 단일변수 부울 함수식을 이용하면 식(1)과 같다.

$$f(x_1)=d_0\overline{x_1}+d_1x_1 \tag{1}$$

여기서 d_0, d_1는 0 또는 1의 진리 값을 갖는다. 식(1)에서 드모르강의 법칙을 이용하면 식(2)과 같은 결과를 얻을 수 있다.

$$f(x_1)=\overline{\overline{d_0x_1}\cdot\overline{d_1x_1}} \tag{2}$$

GF(2)상에서 modulo-2 연산에서는 $\overline{d}=d\oplus 1$ 또는 $d\oplus 1=\overline{d}$이므로 식(2)을 다시 modulo-2 연산식으로 전개하면 식(3)과 같다.

$$\begin{aligned} f(x_1) &= [((d_0(x_1\oplus 1)\oplus 1)\cdot \\ &\quad (d_1x_1\oplus 1))\oplus 1] \\ &= d_0d_1x_1\oplus d_0d_1x_1\oplus d_0x_1\oplus d_0\oplus d_1x_1 \\ &= d_0\oplus(d_0\oplus d_1)x_1 \end{aligned} \tag{3}$$

$$f(x_1)=c_0\oplus c_1x_1 \text{ over GF(2)} \tag{4}$$

식(3)과 식(4)의 관계를 행렬로 나타내면 식(5)과 같다.

$$\begin{bmatrix} c_0 \\ c_1 \end{bmatrix}=\begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}\cdot\begin{bmatrix} d_0 \\ d_1 \end{bmatrix} \text{over GF(2)} \tag{5}$$

이 때, 단 변수에서 연산영역의 계수 d_0와 함수영역의 계수 c_0와의 관계는 식(5)에서 나타난 것처럼 다음의 행렬 형태로 표현되고 이를 변환 행렬(transform matrix)이라한다.

$$T_1=\begin{bmatrix}1&0\\1&1\end{bmatrix} \qquad T_0=[1] \tag{6}$$

여기서, n변수에 대한 행렬은 식(7)과 같다.

$$T_n=\begin{bmatrix}T_{n-1}&0\\T_{n-1}&T_{n-1}\end{bmatrix} \tag{7}$$

식(7)은 변수의 개수가 적을 때에는 변환 행렬을 구하는 것이 용이하지만 변수의 숫자가 증가함에 따라 행렬의 차수가 증가하므로 연산하기에 많은 어려움이 생긴다는 것을 보여준다.

2.2 GRM상수의 생성

만약 입력 변수 x_i를 $\overline{x_i}$로 대치한다면 다른 형태의 정규화된 형식으로 표현된다. 이와 같이 n개의 입력변수에 대해 2^n개의 서로 다른 입력형태가 만들어지며 이에 대한 RM상수를 구하는 것이 GRM 변환이다. GRM 상수 값의 결정은 보통 연산영역에서 극수에 대한 GRM 상수를 구하는 방법이 있지만 극수가 0인 고정극수(fixed polarity)를 구하고, 극수를 확장시켜 GRM 상수를 구하는 것이 최적화된 극수를 구하는데 더욱 효과적이다.

단일변수인 경우 GRM 변환은 식(8)과 같다.

$$f(x_1')=c_0\oplus c_1x_1' \tag{8}$$

x_1'는 입력형태가 x_1이거나 또는 $\overline{x_1}$인 2가지 형태 중에 하나를 의미한다.

① $x_1'=x_1$인 경우

식(8)은 식(9)과 같이 표현된다.

$$f(x_1')=c_0\oplus c_1x_1=a_0\oplus a_1x_1 \tag{9}$$

식(9)을 변환행렬을 이용하여 행렬식으로 표현하면 식(10)과 같이 표현된다.

$$\begin{bmatrix}a_0\\a_1\end{bmatrix}=\begin{bmatrix}1&0\\0&1\end{bmatrix}\begin{bmatrix}c_0\\c_1\end{bmatrix} \text{ 또는 } a=Z_0c \tag{10}$$

② $x_1'=\overline{x_1}$인 경우

modulo-2 성질에 의하여 $\overline{x_1}=x_1\oplus1$이 되고, 식(8)은 식(11)과 같다.

$$\begin{aligned}f(x_1') &= c_0\oplus c_1\overline{x_1}=c_0\oplus c_1(x_1\oplus1)\\ &= a_0\oplus a_1x_1\end{aligned} \tag{11}$$

같은 방법으로 식(11)을 행렬식으로 표현하면 식(12)과 같다.

$$\begin{bmatrix}a_0\\a_1\end{bmatrix}=\begin{bmatrix}1&1\\0&1\end{bmatrix}\begin{bmatrix}c_0\\c_1\end{bmatrix} \text{ 또는 } a=Z_1c \tag{12}$$

이 때, 식(10)과 식(12)의 변환 행렬을 각각 Z_0, Z_1으로 표현할 수 있다.

변수의 개수가 2인 경우에 대해서 보면 $n=2$이므로 총 2^2개의 경우를 갖는다.

① $x_2'\rightarrow x_2$, $x_1'\rightarrow x_1$이면 극수 $p=0$인 경우;

$$\begin{aligned}f(x_2',x_1') &= f(x_2,x_1)\\ &= c_0\oplus c_1x_1\oplus c_2x_2\oplus c_3x_2x_1\\ &= a_0\oplus a_1x_1\oplus a_2x_2\oplus a_3x_2x_1\end{aligned}$$

② $x_2'\rightarrow x_2$, $x_1'\rightarrow\overline{x_1}$이면 극수 $p=1$인 경우;

$$\begin{aligned}f(x_2',x_1') &= f(x_2,\overline{x_1})\\ &= c_0\oplus c_1\overline{x_1}\oplus c_2x_2\oplus x_2\overline{x_1}\\ &= (c_0\oplus c_1)\oplus c_1x_1\oplus(c_2\oplus c_3)x_2\\ &\quad\oplus c_3x_2x_1\\ &= a_0\oplus a_1x_1\oplus a_2x_2\oplus a_3x_2x_1\end{aligned}$$

③ $x_2'\rightarrow\overline{x_2}$, $x_1'\rightarrow x_1$이면 극수 $p=2$인 경우;

$$\begin{aligned}f(x_2',x_1') &= f(\overline{x_2},x_1)\\ &= c_0\oplus c_1\overline{x_1}\oplus c_2x_2\oplus x_2\overline{x_1}\\ &= (c_0\oplus c_1)\oplus(c_1\oplus c_3)x_1\\ &\quad\oplus c_2x_2\oplus c_3x_2x_1\\ &= a_0\oplus a_1x_1\oplus a_2x_2\oplus a_3x_2x_1\end{aligned}$$

④ $x_2'\rightarrow\overline{x_2}$, $x_1'\rightarrow\overline{x_1}$이면 극수 $p=3$인 경우;

$$\begin{aligned}f(x_2',x_1') &= f(\overline{x_2},x_1)\\ &= c_0\oplus c_1\overline{x_1}\oplus c_2\overline{x_2}\oplus c_3\overline{x_2}\overline{x_1}\\ &= (c_0\oplus c_1\oplus c_2\oplus c_3)\oplus(c_1\oplus c_3)x_1\end{aligned}$$

$$\oplus (c_2 \oplus c_3)x_2 \oplus c_3 x_2 x_1$$
$$= a_0 \oplus a_1 x_1 \oplus a_2 x_2 \oplus a_3 x_2 x_1$$

각각의 경우마다 계수 a_i는 달라지며 계수 c_i와의 관계를 다음과 같이 변환행렬로 표현할 수 있다.

1. $P=0 \quad Z_{00} = \begin{bmatrix} 1&0&0&0 \\ 0&1&0&0 \\ 0&0&1&0 \\ 0&0&0&1 \end{bmatrix} = \begin{bmatrix} Z_0 & 0 \\ 0 & Z_0 \end{bmatrix}$

2. $P=1 \quad Z_{01} = \begin{bmatrix} 1&1&0&0 \\ 0&1&0&0 \\ 0&0&1&1 \\ 0&0&0&1 \end{bmatrix} = \begin{bmatrix} Z_1 & 0 \\ 0 & Z_1 \end{bmatrix}$

3. $P=2 \quad Z_{10} = \begin{bmatrix} 1&0&1&0 \\ 0&1&0&1 \\ 0&0&1&0 \\ 0&0&0&1 \end{bmatrix} = \begin{bmatrix} Z_0 & Z_0 \\ 0 & Z_0 \end{bmatrix}$

4. $P=3 \quad Z_{11} = \begin{bmatrix} 1&1&1&1 \\ 0&1&0&1 \\ 0&0&1&1 \\ 0&0&0&1 \end{bmatrix} = \begin{bmatrix} Z_1 & Z_1 \\ 0 & Z_1 \end{bmatrix}$

위의 경우를 살펴보면 모든 경우마다 행렬은 단일변수 변환 행렬 Z_0와 Z_1의 조합으로 이루어져 있다는 것을 알 수 있다. 단일변수 변환행렬을 이용하여 n변수에 대한 임의의 극수 $p=i$의 GRM 변환에 대한 일반식은 식(13)과 같다. 이 때 변환행렬은 $2^n \times 2^n$의 차수를 갖는 행렬이며 단일 변수 변환 행렬의 Kronecker 곱에 의해 생성된다.

$$Z_{n,n-1,n-2,\cdots,1} = Z_{\langle i \rangle} = Z_n \otimes Z_{n-1} \otimes \cdots \otimes Z_1 \quad (13)$$

여기서 $n, n-1, \cdots, 1$은 0또는 1이며 $\otimes$는 Kronecker 곱이다.[4]

Ⅲ. GRM 함수 생성을 위한 단위연살 셀 $[f_i]$ 생성 기법

일반적인 3변수 함수는 다음과 같이 표현된다.

$$f(x_3, x_2, x_1) = d_0 \oplus d_1 x_1 \oplus d_2 x_2 \oplus d_3 x_2 x_1 \oplus d_4 x_3 \oplus d_5 x_3 x_1 \oplus d_6 x_3 x_2 \oplus d_7 x_3 x_2 x_1 \quad (14)$$

3변수 함수인 경우에 축약 방법은 x_3, x_2, x_1 변수 각각에 대한 3가지 축약 방법으로 구분되며 이를 구하기 위해 식(14)을 식(15)과 같이 표현된다.

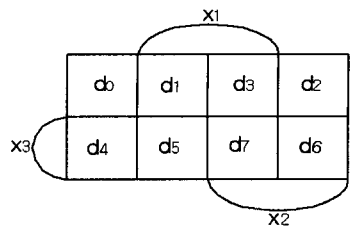


그림 2. 3변수 함수의 k-map 표현
Fig. 2. k-map expression for 3 variables function

$$\begin{aligned} f(x_3, x_2, x_1) &= d_0 \oplus d_1 x_1 \oplus d_2 x_2 \oplus d_3 x_2 x_1 \oplus d_4 x_3 \\ &\quad \oplus d_5 x_3 x_1 \oplus d_6 x_3 x_2 \oplus d_7 x_3 x_2 x_1 \\ &= d_0 x_3 x_2 x_1 \oplus d_1 x_3 x_2 x_1 \oplus d_2 x_3 x_2 x_1 \\ &\quad \oplus d_3 x_3 x_2 x_1 \oplus d_4 x_3 x_2 x_1 \\ &\quad \oplus d_5 x_3 x_2 x_1 \oplus d_6 x_3 x_2 x_1 \oplus d_7 x_3 x_2 x_1 \end{aligned} \quad (15)$$

① 변수 x_1에 대한 축약

식(15)에서 변수 x_1를 축약하면 다음과 같이 된다.

$$\begin{aligned} f(x_3, x_2, x_1) &= (d_0 \overline{x_1} OPLUS d_1 x_1) \overline{x_3 x_2} \\ &\quad \oplus (d_2 x_1 \oplus d_3 x_1) x_3 x_2 \oplus (d_4 x_1 \oplus d_3 x_1) x_3 \overline{x_2} \\ &\quad \oplus (d_6 x_1 \oplus d_7 x_1) x_3 x_2 \\ &= f_0 x_3 x_2 \oplus f_1 x_3 x_2 \oplus f_2 x_3 x_2 \oplus f_3 x_3 x_2 \end{aligned}$$

$$f_0 = d_0 \overline{x_1} \oplus d_1 x_1, \quad f_1 = d_2 \overline{x_1} \oplus d_3 x_1,$$
$$f_2 = d_4 \overline{x_1} \oplus d_5 x_1, \quad f_3 = d_6 \overline{x_1} \oplus d_7 x_1$$

② 변수 x_2에 대한 축약

같은 방법을 이용해 변수 x_2를 축약하면 다음과 같은 결과를 얻는다.

$$\begin{aligned} f(x_3, x_2, x_1) &= (d_0 \overline{x_2} \oplus d_2 x_2) \overline{x_3 x_1} \\ &\quad \oplus (d_1 \overline{x_2} \oplus d_3 x_2) \overline{x_3 x_1} \\ &\quad \oplus (d_4 \overline{x_2} \oplus d_6 x_2) x_3 \overline{x_1} \\ &\quad \oplus (d_5 \overline{x_2} \oplus d_7 x_2) x_3 \overline{x_1} \\ &= f_0 x_3 x_1 \oplus \oplus f_1 x_3 x_1 \oplus f_2 x_3 \overline{x_1} \\ &\quad \oplus f_3 x_3 x_1 \end{aligned}$$

$$f_0 = d_0 \overline{x_2} \oplus d_2 x_2, \quad f_1 = d_1 \overline{x_2} \oplus d_3 x_2.$$
$$f_2 = d_4 \overline{x_2} \oplus d_6 x_2, \quad f_3 = d_5 \overline{x_2} \oplus d_7 x_2$$

③ 변수 x_3에 대한 축약

마찬가지로 같은 방법을 사용하여 변수 x_3에 대한 축약을 실행하면 다음과 같다.

$$\begin{aligned} f(x_3, x_2, x_1) &= (d_0 \overline{x_3} \oplus d_4 x_3) \overline{x_2 x_1} \oplus (d_1 \overline{x_3} \oplus d_5 x_3) \overline{x_2} x_1 \\ &\quad \oplus (d_2 \overline{x_3} \oplus d_6 x_3) x_2 x_1 \oplus (d_3 \overline{x_3} \oplus d_7 x_3) x_2 x_1 \\ &= f_0 x_2 x_1 \oplus f_1 x_2 x_1 \oplus f_2 x_2 x_1 \oplus f_3 x_2 x_1 \end{aligned}$$

$$f_0 = d_0\overline{x_3} \oplus d_4 x_3,\quad f_1 = d_1\overline{x_3} \oplus d_5 x_3,$$
$$f_2 = d_2\overline{x_3} \oplus d_6 x_3,\quad f_3 = d_3\overline{x_3} \oplus d_7 x_3$$

위의 3가지 방법으로 구해진 축약방법들을 그림 3에 도시하였다.

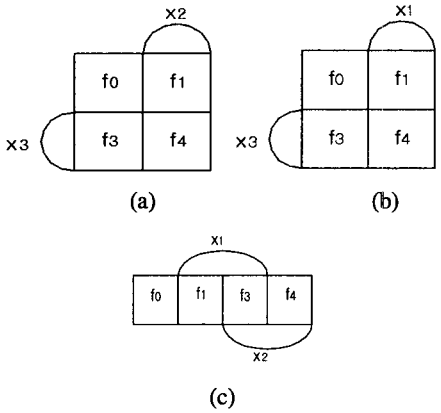


그림 3, 3변수 함수의 축약, (a) 변수 x_1의 축약 (b) 변수 x_2의 축약 (c) 변수 x_3의 축약

Fig. 3. Minimization of 3 variables function, (a) for variable x_1 (b) for variable x_2 (c) for variable x_3

각각의 방법으로 축약된 함수들은 $[f_i] = \begin{bmatrix} d_k \\ d_j \end{bmatrix} [\overline{x_a} \quad x_a]$의 행렬식으로 표현할 수 있다. 여기서 $[f_i]$는 n변수 함수에 대하여 $n-1$개의 개수를 갖는 셀(cell)이며, d_k, d_i는 함수의 계수, x_a는 축약시키고자 하는 변수이다.

본 논문에서는 축약된 변수 f_i는 $\begin{bmatrix} d_k \\ d_j \end{bmatrix}[\overline{x_a} \; x_a]$의 수식을 갖는 단위 연산 셀(cell)이라 정의하기로 한다.

Ⅳ. 단위연산 셀 $[f_i]$을 이용한 새로운 GRM 상수의 생성

단일변수 함수에서 극수 $p=0$와 $p=1$의 계수 간에는 식(16)과 같은 관계가 성립한다.

$$\begin{bmatrix} a_0 \\ a_1 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} d_0 \\ d_1 \end{bmatrix} \tag{16}$$

위의 식에서 좌변의 계수 a_i는 극수 $p=1$의 계수이며, 우변의 계수 d_i는 극수 $p=0$의 계수이다.

계수 d_i는 지난 장에서 보았던 축약하고자 했던 변수의 셀 $[f_i]$이므로 극수 $p=1$의 계수에 대해 다음과 같은 식이 성립된다.

$$[f_i'] = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} d_k \\ d_j \end{bmatrix}[\overline{x_a} \quad x_a]$$
$$[f_i'] = [Z]\begin{bmatrix} d_k \\ d_j \end{bmatrix}[\overline{x_a} \; x_a] \tag{17}$$
$$= [Z][f_i]$$

단일 변수 함수에서는 얻을 수 있는 극수가 극수 $p=1$ 뿐이므로 셀 $[f_i]$는 항상 일정하므로 2변수와 3변수 일 때를 알아본다.

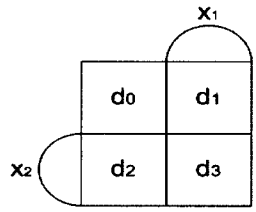


그림 4. 2변수 함수의 k-map 표현

Fig. 4. K-map expression of 2 variables function

① 2변수인 경우

2변수인 경우에는 변수에 따라 3개의 극수가 존재한다.

먼저 구하려는 극수에 대한 셀 $[f_i]$을 구해보면 다음과 같다. 위의 그림에서 $p=1$인 경우에는 $\begin{bmatrix} d_o \\ d_1 \end{bmatrix}[\overline{x_1} \; x_1]$, $\begin{bmatrix} d_2 \\ d_3 \end{bmatrix}[\overline{x_1} \; x_1]$이 각각 셀 $[f_0]$, $[f_1]$이 된다.

$p=2$인 경우에는 $\begin{bmatrix} d_o \\ d_2 \end{bmatrix}[\overline{x_2} \; x_2]$, $\begin{bmatrix} d_1 \\ d_3 \end{bmatrix}[\overline{x_2} \; x_2]$인 셀 $[f_0]$, $[f_1]$가 구해지고, $p=3$인 경우에는 변수 x_1, x_2 모두에 보수가 취해지므로 위의 2가지 경우를 반복한다.

이처럼 셀 $[f_i]$가 구해졌으면 각각의 극수마다 GRM 상수를 구해야 되는데 이 연산은 얻어진 셀 $[f_i]$와 단일변수 변환 행렬을 곱하면 얻을 수 있다.

$p=2$인 경우에는

$$[f_0] = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} d_0 \\ d_2 \end{bmatrix}\begin{bmatrix} \overline{x_2} \; x_2 \end{bmatrix}$$
$$= \begin{bmatrix} d_0 + d_2 \\ d_2 \end{bmatrix}\begin{bmatrix} \overline{x_2} \; x_2 \end{bmatrix} = \begin{bmatrix} d_0 + d_2 \\ d_2 \end{bmatrix} \tag{21}$$

위의 식에서 우변에 있던 변수 행렬은 계수를 선택하기 위한 행렬이었고 연산이 끝나면 그 결과가 다

시 본래의 위치로 돌아가기 때문에 직접적으로 연산 결과에는 포함시키지 않는다.

$$[f_1'] = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} d_1 \\ d_3 \end{bmatrix} \begin{bmatrix} \overline{x_2} & x_2 \end{bmatrix} = \begin{bmatrix} d_1 + d_3 \\ d_3 \end{bmatrix}$$

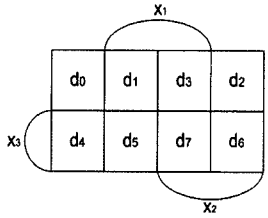


그림 5. 3변수 함수의 k-map 표현
Fig. 5. K-map expression of 3 variables function

② 3변수인 경우

3변수인 경우에도 2변수인 경우와 같은 방법을 사용하지만 2변수보다 더 많은 극수가 존재하므로 조금 더 많은 셀 $[f_i']$이 필요하다.

3변수 함수는 총 8개의 극수를 가지므로 8번의 연산을 해야 하지만 모두 같은 방법을 사용하기 때문에 하나의 극수($p=5$)를 선택하여 알아본다.

극수 $p=5$인 경우 보수가 취해지는 변수는 x_1과 x_3이므로 먼저 x_1에 대한 셀 $[f_i]$을 구하고 얻어진 결과를 x_3에 대하여 셀 $[f_i]$을 구하면 된다.

$$[f_0] = \begin{bmatrix} d_0 \\ d_1 \end{bmatrix} \begin{bmatrix} \overline{x_1} & x_1 \end{bmatrix}, [f_1] = \begin{bmatrix} d_3 \\ d_2 \end{bmatrix} \begin{bmatrix} \overline{x_1} & x_1 \end{bmatrix},$$
$$[f_2] = \begin{bmatrix} d_4 \\ 5 \end{bmatrix} \begin{bmatrix} \overline{x_1} & x_1 \end{bmatrix}, [f_3] = \begin{bmatrix} d_7 \\ d_6 \end{bmatrix} \begin{bmatrix} \overline{x_1} & x_1 \end{bmatrix}$$

얻어진 셀 $[f_i]$을 단일변수 변환행렬 $[Z]$와 연산하여 셀 $[f_i']$을 얻는다.

$$[f_0] = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} d_0 \\ d_1 \end{bmatrix} = \begin{bmatrix} d_0 \oplus d_1 \\ d_1 \end{bmatrix},$$
$$[f_1] = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} d_3 \\ d_2 \end{bmatrix} = \begin{bmatrix} d_3 \oplus d_2 \\ d_2 \end{bmatrix},$$
$$[f_2] = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} d_4 \\ d_5 \end{bmatrix} = \begin{bmatrix} d_4 \oplus d_5 \\ d_5 \end{bmatrix},$$
$$[f_3] = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} d_7 \\ d_6 \end{bmatrix} = \begin{bmatrix} d_7 \oplus d_6 \\ d_6 \end{bmatrix}$$

일단 변수 x_1에 대한 셀 $[f_i']$을 구했으므로 변수 x_3에 대해 반복한다.

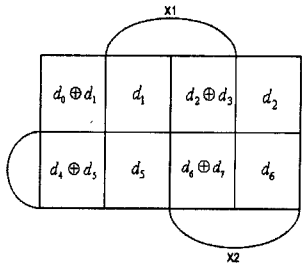


그림 6. 변수 x_1를 축약하여 얻어진 결과
Fig. 6. Result derived minimizing variable x_1

$$[f_0] = \begin{bmatrix} d_0 \oplus d_1 \\ d_4 \oplus d_5 \end{bmatrix} [\overline{x_3}\, x_3], \; [f_1] = \begin{bmatrix} d_1 \\ d_5 \end{bmatrix} [\overline{x_3}\, x_3],$$
$$[f_2] = \begin{bmatrix} d_3 \oplus d_2 \\ d_7 \oplus d_6 \end{bmatrix} [\overline{x_3}\, x_3], \; [f_3] = \begin{bmatrix} d_2 \\ d_6 \end{bmatrix} [\overline{x_3}\, x_3]$$

마찬가지로 변환행렬 $[Z]$와 연산하여 셀$[f_i']$을 구한다.

$$[f_0'] = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} d_0 \oplus d_1 \\ d_4 \oplus d_5 \end{bmatrix} = \begin{bmatrix} d_0 + d_1 \oplus d_4 \oplus d_5 \\ d_4 \oplus d_5 \end{bmatrix},$$
$$[f_1'] = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} d_1 \\ d_5 \end{bmatrix} = \begin{bmatrix} d_1 \oplus d_5 \\ d_5 \end{bmatrix}$$
$$[f_2'] = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} d_3 \oplus d_2 \\ d_7 \oplus d_6 \end{bmatrix} = \begin{bmatrix} d_3 \oplus d_2 \oplus d_7 \oplus d_6 \\ d_7 \oplus d_6 \end{bmatrix},$$
$$[f_1'] = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} d_2 \\ d_6 \end{bmatrix} = \begin{bmatrix} d_2 \oplus d_6 \\ d_6 \end{bmatrix}$$

얻어진 셀들을 원래의 k-map에 대입하게 되면 원하는 극수 $p=5$의 계수를 얻을 수 있다.

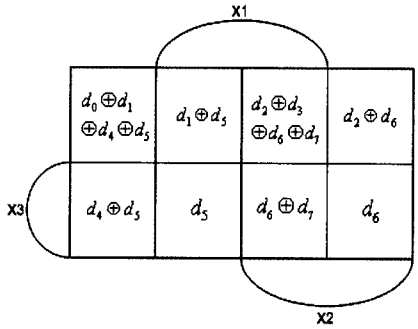


그림 7. 극수 $p=5$인 3변수 함수의 k-map 표현
Fig. 7. K-map expression of 3 variables function with polarity $p=5$

그림 7에서 볼 수 있듯이 일반적인 3변수 함수 $f(x_3, x_2, x_1)$에 대한 극수 $p=5$의 함수식은 다음과 같다.

$$f(\overline{x_3}, x_2, \overline{x_1}) = (d_0 \oplus d_1 \oplus d_4 \oplus d_5) \oplus (d_1 \oplus d_5)x_1 \oplus (d_2 \oplus d_6)x_2 \oplus (d_2 \oplus d_3 \oplus d_6 \oplus d_7)x_1x_2 \oplus (d_4 \oplus d_5)x_3 \oplus d_5x_1x_3 \oplus d_6x_2x_3 \oplus (d_7 \oplus d_6)x_1x_2x_3$$

③ $n(n>3)$ 변수인 경우

$n>3$인 n변수에 대해서는 k-map의 특성상 변수가 증가하게 되면 같은 형태로 증가하기 때문에 3변수의 경우를 확장하게 되면 n변수에 대해서도 셀 방식을 적용할 수 있다.

V. K-map에서 얻어진 셀 $[f_i]$을 이용한 GRM 상수 생성 알고리즘

그림 8의 블록도를 단계별로 표시하면 다음과 같다.

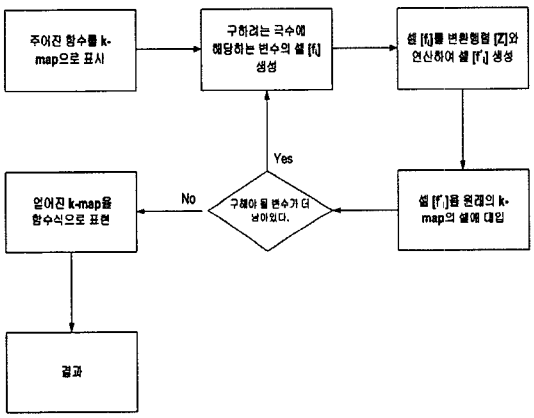


그림 8. K-map에서 얻어진 셀 $[f_i]$을 이용한 GRM 상수 생성 알고리즘의 블록도
Fig. 8. Block diagram of derivation method algorithm of GRM coefficients using cell $[f_i]$ of k-map

[단계1] 주어진 함수를 k-map으로 표시한다.

[단계2] 구하려는 극수에 해당하는 변수의 셀 $[f_i]$을 구한다.

[단계3] 셀 $[f_i]$을 단일변수 변환행렬 $[Z]$와 연산하여 셀$[f_i']$을 구한다.

[단계4] 셀$[f_i']$을 원래의 k-map의 셀에 대입한다.

[단계5] 구하려는 극수에 해당되는 변수가 2개 이상이면 [단계2]~[단계4]를 반복한다.

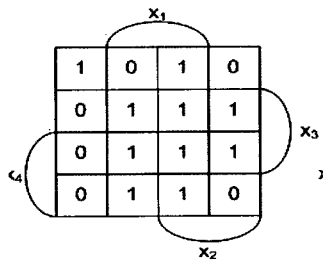
[단계6] 축약해야 될 변수가 더 이상 남지 않게 되면 [단계4]에서 얻어진 k-map을 함수식으로 표현하고 알고리즘을 종료한다.

다음 예제를 통하여 주어진 함수식에서 [k-map에서 얻어진 셀$[f_i]$을 이용한 GRM 상수 생성 알고리즘]을 이용하여 GRM 계수를 생성하는 과정에 대해 알아본다.

[예제1] [k-map에서 얻어진 셀 $[f_i]$을 이용한 GRM 상수 생성 알고리즘]을 이용하여 다음과 같은 4변수 함수의 $p=5$ 계수를 구한다.

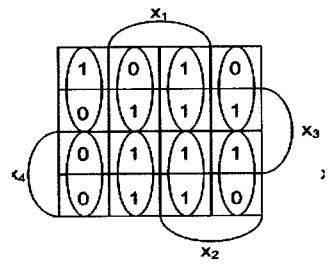
$$f(x_4, x_3, x_2, x_1) = 1 \oplus x_2x_1 \oplus x_3x_2 \oplus x_3x_2x_1 \oplus x_4x_1 \oplus x_4x_2x_1 \oplus x_4x_3x_1 \oplus x_4x_3x_2 \oplus x_4x_3x_2x_1$$

[단계 1] 주어진 함수를 k-map으로 표시한다.



[단계 2] 구하려는 극수에 해당하는 변수의 셀 $[f_i]$을 구한다.

극수 $p=5$의 계수를 구하려면 변수 x_3, x_1에 보수를 취해야만 하므로 먼저 x_3에 대한 보수를 취한다.

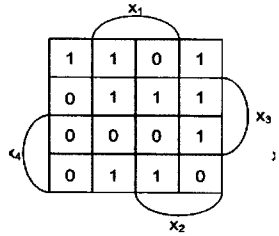


[단계 3] 셀 $[f_i]$을 단일변수 변환행렬 $[Z]$와 연산하여 셀$[f_i']$을 구한다.

$$[f_0'] = \begin{bmatrix}1\\0\end{bmatrix},\ [f_1'] = \begin{bmatrix}1\\1\end{bmatrix},\ [f_2'] = \begin{bmatrix}0\\1\end{bmatrix},\ [f_3'] = \begin{bmatrix}1\\1\end{bmatrix},$$

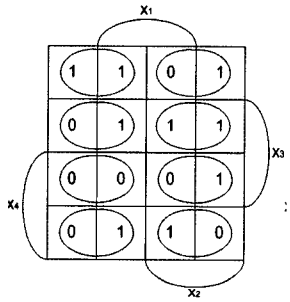
$$[f_4'] = \begin{bmatrix}0\\0\end{bmatrix},\ [f_5'] = \begin{bmatrix}0\\1\end{bmatrix},\ [f_6'] = \begin{bmatrix}0\\1\end{bmatrix},\ [f_7'] = \begin{bmatrix}1\\0\end{bmatrix}$$

[단계 4] 셀 $[f_i']$을 원래의 k-map의 셀에 대입한다.



[단계5] 구하려는 극수에 해당되는 변수가 2개 이상이므로 [단계2]~[단계4]를 반복한다.

[단계2] 구하려는 극수에 해당하는 변수의 셀 $[f_i]$을 구한다.

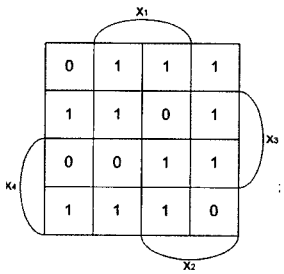


[단계3] 셀 $[f_i]$을 단일변수 변환행렬 $[Z]$와 연산하여 셀$[f_i']$을 구한다.

$$[f_0'] = \begin{bmatrix}0\\1\end{bmatrix},\ [f_1'] = \begin{bmatrix}1\\1\end{bmatrix},[f_2'] = \begin{bmatrix}1\\1\end{bmatrix},\ [f_3'] = \begin{bmatrix}0\\1\end{bmatrix},$$

$$[f_4'] = \begin{bmatrix}0\\0\end{bmatrix},\ [f_5'] = \begin{bmatrix}1\\1\end{bmatrix},[f_6'] = \begin{bmatrix}1\\1\end{bmatrix},\ [f_7'] = \begin{bmatrix}1\\0\end{bmatrix}$$

[단계4] 셀 $[f_i']$을 원래의 k-map의 셀에 대입한다.



[단계6] 구해야 되는 변수가 더 이상 남지 않게 되면 [단계4]에서 얻어진 k-map을 함수식으로 표현하고 알고리즘을 종료한다.

$$f(x_4, x_3, x_2, x_1) = x_1 \oplus x_2 \oplus x_2x_1 \oplus x_3 \oplus x_3x_1 \oplus x_3x_2 \oplus x_4 \oplus x_4x_1 \oplus x_4x_2x_1 \oplus x_4x_3x_2 \oplus x_4x_3x_2x_1$$

Ⅵ. GRM 상수 생성 회로 설계

지난 장에서 제안한 바와 같이 GRM 상수를 구하는 방법은 주어진 함수에서 셀 $[f_i]$를 구하고 이를 변환행렬 $[Z]$와 연산하여 셀 $[f_i']$을 구하는 방법이 있다. 여기서 셀 $[f_i]$와 셀 $[f_i']$는 변환행렬 $[Z]$의 관계를 가지고 있는 것을 알 수 있다.

$$[f_i'] = [Z][f_i]$$

여기서 변환행렬 $[Z]$는 $[Z_0]$와 $[Z_1]$의 두 가지 경우로 나타낼 수 있다. 이 두가지 경우의 변환행렬들을 회로로 구현하면 그림 9과 같다.

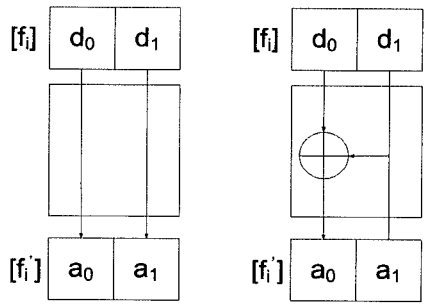


그림 9. 변환 행렬 $[Z]$의 회로 구현, (a)$[Z_0]$의 회로 구현 (b)$[Z_1]$의 회로 구현

Fig. 9. Implemented circuit of transform matrix $[Z]$ (a)circuit for Z_0 (b)circuit for Z_1

위의 그림에서 $[Z_0]$에 해당하는 회로는 by-pass 회로이다.

지난 절에서 보았던 [k-map에서 얻어진 셀 $[f_i]$을 이용한 GRM 상수 생성 알고리즘]에서 [단계3]에서 그림 9의 회로를 이용하면 보다 간단하게 GRM 상수를 얻을 수 있다.

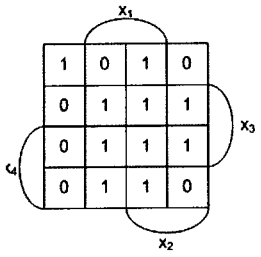
이렇게 구현된 회로를 이용하여 지난 절에서 보았던 예제를 구해보면 다음과 같다.

[예제2] 다음과 같은 함수

$$f(x_4, x_3, x_2, x_1) = 1 \oplus x_2x_1 \oplus x_3x_2 \oplus x_3x_2x_1 \oplus x_4x_1 \oplus x_4x_2x_1 \oplus x_4x_3x_1 \oplus x_4x_3x_2 \oplus x_4x_3x_2x_1$$

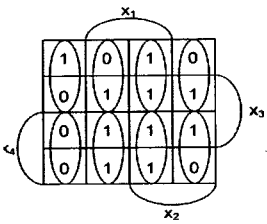
의 극수 $P=4$의 GRM 상수를 구한다.

[단계 1] 주어진 함수를 k-map으로 표시한다.

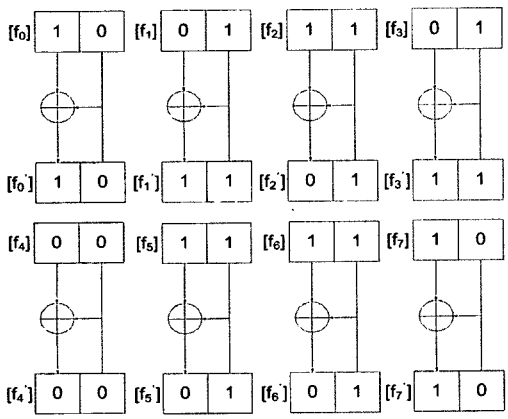


[단계 2] 구하려는 극수에 해당하는 변수의 셀 $[f_i]$을 구한다.

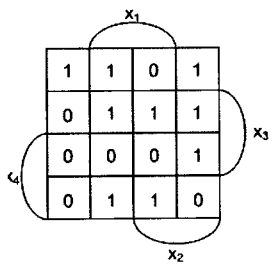
극수 $P=4$의 GRM 상수를 구하려면 변수 x_3에 보수를 취해야 한다.



[단계 3] 셀 $[f_i]$을 단일변수 변환행렬 $[Z]$와 연산하여 셀$[f_i']$을 구한다.



[단계 4] 셀 $[f_i']$을 원래의 k-map의 셀에 대입한다.



[단계 5] 구하려는 극수의 변수가 1개 이므로 다음 단계로 넘어간다.

[단계 6] 구해야되는 변수가 더 이상 남지 않게 되면 [단계4]에서 얻어진 k-map을 함수식으로 표현하고 알고리즘을 종료한다.

$$f(x_4,x_3,x_2,x_1)=1\oplus x_1\oplus x_2\oplus x_3x_1\oplus x_3x_2 \oplus x_3x_2x_1\oplus x_4x_1\oplus x_4x_2\oplus x_4x_3x_2$$

[예제2]에서 볼 수 있듯이 4변수 함수에서 극수 $P=4$인 GRM 상수를 구하는데 8개의 가산기만을 사용하였다. 모든 극수의 GRM 상수를 구하려면 그림 10과 같은 과정을 거치면 모든 극수의 GRM 상수를 구할 수 있다.

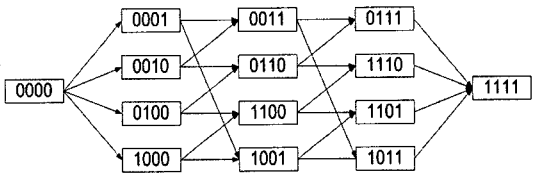


그림 10. 4변수 함수에서 극수의 변환 과정
Fig. 10. Transform process of polarities with 4 variables function

그림 10은 4변수 함수에서 극수들의 변환 과정을 표시한 것이다. 0000은 4개의 변수 $x_4.x_3,x_2,x_1$를 의미하고 1111은 $\overline{x_4},\overline{x_3},\overline{x_2},\overline{x_1}$를 의미한다. 또한 화살표는 한 개의 변수의 GRM 변환으로 [예제2]에서 본 것과 같이 8개의 가산기로 구성되어 있다. 0000부터 1111까지 총 24개의 화살표로 되어있지만 중간에 있는 화살표들은 중복되었기 때문에 총 16개의 화살표, 즉 128개의 가산기를 사용하면 모든 극수의 GRM 함수를 구할 수 있다.

Ⅷ. 비교 및 검토

GRM 상수를 구하는 목적은 최적의 극수를 선정하여 최적화된 GRM 함수를 구성하기 위한 것이다. 최적의 함수를 생성하기 위한 기존의 방법을 살펴보면 행렬과 같은 수식연산을 이용하여 모든 극수의 GRM 함수를 구한 후 이를 통해 최적의 함수를 결정하였다. 이러한 기존의 방식과는 달리 본 논문에서는 k-map에서 얻어진 셀(cell)을 이용하여 GRM 함수의 극수를 구한 후, 이를 통해 최적의 함수를 결정하는 새로운 방법을 제안하였다.

표 1. 연산자 비교표
Table 1. Comparison table in number of operators

n		Green	Besslich	본 논문
2	가산	16	3	4
	승산	32	0	0
	KP	12	0	0
3	가산	128	28	12
	승산	256	0	0
	KP	21	0	0
$n>3$	가산	$2^n \times 2^{n-2}$	$2^n(2^{n-1}+\frac{n}{2})$	$n \cdot 2^n$
	승산	$4^n \times 2^{n-1}$	0	0
	KP	$n \times (2^n-1)$	0	0

본 논문에서 제안한 알고리즘은 최적화된 GRM 함수를 생성하는데 있어 2개의 변수만을 이용하여 GRM 함수의 계수를 생성하므로 고려해야 할 경우의 수를 최소화 할 수 있고, 이에 따라 연산의 개수를 최소화 할 수 있다.

표 1에는 본 논문에서 제안한 알고리즘과 기존의 Besslich, Green이 제안한 알고리즘의 연산자의 수를 비교하였다.

본 논문에서는 주어진 함수식을 k-map상에서 셀 $[f_i]$로 구분하고, 이 셀을 단일 변수 변환 행렬과 연산하여 결과를 얻어내므로 보다 적은 수의 가산기를 사용한다.

기존의 여러 방법들은 변수가 4변수로만 증가해도 Green은 64개의 가산기, 2048개의 승산기, 59개의 KP(kronecker곱 승산기)를 사용하였고 Besslich는 192개의 승산기를 사용하였지만 본 논문에서 제안한 알고리즘은 승산의 필요 없이 32개의 가산기만으로 구할 수 있다.

Ⅶ. 결 론

GRM상수를 구하는 기존의 방법으로는 Green이 제안한 변환행렬의 Kronecker 곱을 이용한 방법 등의 생성 알고리즘을 이용한 방법이 있다. 이런 방법은 변수의 수가 많아지면 변환행렬의 차수도 증가한다는 점과 변환행렬 자체를 구하는 것도 어려워지는 단점을 갖고 있었다. 그러므로 본 논문에서는 k-map에서 얻어진 셀을 이용하여 GRM 상수를 구하는 알고리즘을 제안하였다. 이와 같이 셀 방식을 이용하면 3변수 함수인 경우 Green의 방법보다 가산기의 수가 1/42, Besslich의 알고리즘보다 1/9의 가산기만을 이용하여 GRM 함수를 구하였다. 또한 회로 구현이 용이하고 연산자의 개수도 감소하므로 처리속도도 향상될 수 있다.

추후 연구과제로는 $GF(2)$, 즉 이진 논리에서 셀 방식을 적용한 것과 같이 임의의 소수 p의 $GF(p)$ 상에서 적용할 수 있는 알고리즘을 개발하는 것과 최적화된 GRM 상수를 구할 수 있는 연구가 진행되어야 할 것으로 생각된다.

참 고 문 헌

[1] K.C Sith, "Multiple-Valued logic:a tutorial and appreciation", computers, pp. 17-27, Apr, 1988

[2] D. Etiemble, "On the performance of the Multivalued intergrated circuits:past, present and future:, 22th ISMVL pp.154-164, Sendai Japan, May, 1992

[3] Qinhua Hong, Benchu Fei, Haomin Wu, Markek A. Perkowski, Nan Zhuang, "Fast Synthesis for Ternary Reed-Muller Expansion," IEEE Proc. of Symposium on Multiple-Valued Logic, Sacramento, California, pp.14-16, May 1993

[4] David Green, Modern Logic Design, Addison-Wesley Publishing company, Inc.1986

[5] 이철우, 김영건, 박동영, 강성수, 김홍수, "단일변수 변환 행렬을 이용한 GRM 상수 생성 방법," 대한전자공학회 추계학술논문집, vol. 21, no. 2, pp. 807-810, 1998년 11월

[6] W.Besslich, "Efficient computer method for ExOR logic design," IEE Proceedings, vol 130, Pt. E, No. 6, pp.203-206, November 1983

[7] X.Wu, X.Chen, S.L.Hurst, "Mapping of Reed-Muller coefficients and the minimisation of exclusive OR-switching functions," IEE PROC.,Vol. 129, Pt. E, No. 1, January 1982

[8] D.H.Green, I.S.Taylor, "Multiple-valued switching circuit design by means of generalized Reed-Muller expansions," Digital Processes, 2, pp.63-81, 1976

[9] D.H.Green, I.S.Taylor, "Modular representation of multiple-valued logic systems," IEE Proceedings. vol 121, pp. 409-418, 1973

[10] Qinhua Hong, Benchu Fei, Haomin Wu,

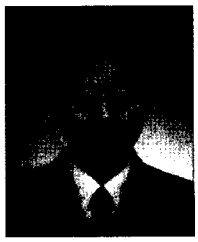
Markek A. Perkowski, Nan Zhuang, "Fast Synthesis for Ternary Reed-Muller Expansion," IEEE Proc. of Symposium on Multiple-Valued Logic, Sacramento, California, pp.14-16, May 1993

[11] T.Sasao, "Calculation of Reed-Muller-Fourier Coefficients of Multiple-Valued Functions through Multiple-Place Decision", IEEE Proc. of International Symposium on Multiple-Valued Logic, Boston, Massachusetts, USA, pp.82-88, May 1994.

[12] T. Sasao, "Optimization of Multiple-Valued AND-EXOR Expressions using Multiple-Place Decision Diagrams," IEEE Proc. of Symposium on Multiple-Valued Logic, Sendai Japan,. pp.451-458, May. 1992.

[13] D.M.Miller, "Multiple-Valued Logic Design Tools", IEEE Proc. of Symposium on Multiple-Valued Logic, Sacramto, California, pp.2-11, May. 1993

이 철 우(Chol-U Lee) 정회원



1998년 2월 인하대학교 전자공학과 (공학사)
2000년 2월 인하대학교 전자공학과 (공학석사)
2000년 3월~현재 인하대학교 전자공학과 박사과정 재학중
<관심분야> 회로 설계, 디지털 로직, 정보 및 부호이론

차 문 철(Wenzhe Che) 정회원

1999년 7월 중국 심양건축대학교 자동제어학과(공학사)
2002년 2월 인하대학교 전자공학과 (공학석사)
2002년 3월~현재 인하대학교 전자공학과 박사과정 재학중
<관심분야> 퍼지 이론, 퍼지 제어, 회로설계

김 흥 수(Heung-Soo Kim) 정회원

한국통신학회 논문지 제28권 제2A호 참조
현재 인하대학교 전자공학과 교수
<관심분야> 회로 및 시스템, 스위칭이론, 논리회로 설계, 퍼지논리, 다치논리 등