論文95-32B-6-5

테이블 대수형 스위칭 함수를 위한 배타적 논리합 연산 방법

(Method for Exclusive-OR Operation for Switching Equations Based on Tabular Algebra)

鄭 華 子 *, 鄭 己 鉉 **

(Hwaja Chung, and Gihyun Jung)

요 약

본 논문에서는 테이블 형으로 된 두 부울 함수 간의 배타적 논리합 연산 방법을 제안하였다. 이 방법을 적용함으로써 Unger의 보수 기법보다 직접적인 방법으로 스위칭 함수나 동시 함수의 해를 구할 수 있다.

Abstract

In this paper a method to perform Exclusive-OR operation between two tabular type Boolean expressions is presented. The proposed method allows to solve the switching equations and the simultaneous equations in a rather direct manner, compared with Unger's method.

I. 서 론

디지탈 회로의 테스터 패턴 생성이나 조합 회로의 신호 확률 예측과 같은 스위칭 이론에 관련된 많은 문제를 해결하기 위해 스위칭 함수가 널리 사용되고 있다. 1970년대와 1980년대에 많은 논문들[3]-[4]이 부울 함수의 해를 구하는 효과적인 방법들을 제안하였다. 이들에 따르면 스위칭 함수의 해를 구하기 위해 많이 사용되는 방법은 크게 부울 함수를 사용하는 대수적인 방법과 진리표를 사용하는 테이블 방식으로 나눌 수 있다. 대수적인 방법은 비교적 간단하고 일반성이 있다는 장점이 있으나 수작업으로 해를 구하기가 쉽지 않고 복잡하며, 컴퓨터를 사용하기에는 적절하지 못한 단점이 있다. 따라서 효과적으로 해를 구하는 방법이 제시되어 있지 않다. 이에 비해 테이블을 이용하는 방법은 컴퓨터를 사용하여 구현하기에 편하다는 장점이 있으나 n개의 변수의 진리 함수는 2^n개의 값을 가지게 되어 n의 값이 커질 때는 매우 큰 용량의 메모리를 요구하게 되므로 컴퓨터로 실현하는 것이 실용적이 아니다. 그러나, 테이블 방식에서 스위칭 함수에 필요한 변수의 수를 최대한 줄이고 이를 효과적으로 이용할 수 있다면 이는 매우 유용하게 사용될 수 있게 될 것이다. n개의 변수로 된 부울 함수의 해는 부울 대수로서 알려져 있는 대수학적인 방법 외에도 각 변수들의 값을 2^n개의 셀로 된 리스트에 사상시키는 테이블 방법을 사용하여 쉽게 구할 수 있다.

표 1.a B^3의 부분 벡터
Table 1.a Sub-vectors in B^3

y_1	y_2	y_3
0	-	-
1	-	0

표 1.b B^3의 부분 벡터
Table 1.b All vectors in B^3

y_1	y_2	y_3
-	-	-

Trabado 등[1]은 부울 함수의 해를 테이블 대수로

* 正會員, 서울産業大學校 電子計算學科
(Dept. of Computer Science, S.N.P.U.)
** 正會員, 亞洲大學校 電子工學科
(Dept. of Elec. Eng., Ajou Univ.)
接受日字: 1994年8月13日, 수정완료일: 1995年5月25日

구하는 새로운 방법을 처음 제안하였다. 이 방법은 표현이 매우 간략하고 종래의 방법보다 훨씬 간단히 해를 구할 수 있다는 장점이 있다. 부울 대수 Bn에는 2^n개의 벡터 변수들이 있다. $S = \{Y_i\}$, $i = 1, 2, \cdots, k$ (단, $k \leq 2^n$, $Y_i \in B^n$) 라 하면, S를 나타내는 가장 간단한 방법은 k개의 열과 n개의 행으로 된 테이블을 사용하는 것이다. 각 행은 하나의 벡터에 대응된다. 이 때 각 행의 요소 y_i (i = 1~3) 는 각기 truth, false, don't care 에 해당되는 "1", "0", "-" 의 3가지 중의 하나로 주어진다. 예를 들어 표 1.a는 B^3 중 벡터 y_1'와 y_1y_3'를 나타내고 표 1.b는 B^3 전체를 나타낸다.

Trabado 등의 논문에서는 가장 기본적인 표현법 및 연산법인 논리곱 (AND)과 논리합 (OR)에 대한 테이블 대수 연산 방법을 제시하였다. 한편, Unger[12]는 Trabado 등이 제안한 테이블 대수 식을 발전시켜 매트릭스 형의 테이블을 간소화는 방법과 테이블 표현식 (tabular expression)의 보수를 구하는 방법을 제안하였다. 테이블의 각 행들 중에서 공통 변수를 가지는 행들은 합병 원리 (즉, XY + XY′ = X) 를 적용하여 합병시킴으로써 테이블을 간소화한다. 또한 f (X)의 보수를 취하여 f′(X) = 0 인 함수의 해를 구함으로써 f (X) = 1 인 함수의 해를 얻을 수 있다. 이 보수 기법은 $f(X) = g(X)$ 형의 스위칭 함수의 해를 구하는 데에 매우 유용하게 사용될 수 있다.

본 논문에서는 배타적 논리합 (Exclusive OR) 연산을 통해 스위칭 함수의 해를 구하기 위해 테이블 대수식의 배타적 논리합 연산을 위한 규칙들을 제안하였다. 본 제안 규칙을 사용하는 경우 $f(X) = g(X)$ 와 같은 유형의 함수나 동시 함수의 해를 Unger가 제안한 보수 기법을 사용하는 것보다 더 직접적이고 효과적인 방법으로 구할 수 있다.

Ⅱ장에 Trabado 등이 제안한 테이블 대수와 Unger가 제안한 보수 기법 및 합병 이론에 대해 설명한다. Ⅲ장에서 본 논문에서 제안한 테이블 대수의 배타적 논리합 (TXOR)의 연산 규칙들에 대해 기술하고 Ⅳ장에서는 제안 방법을 사용하여 $f(X) = g(X)$ 형의 스위칭 함수의 해를 구하는 방법을 설명한다.

Ⅱ. 테이블 대수 연산

두개의 부울 변수 간의 논리합은 두 변수를 각기 테이블 대수로 표현한 후, 두 테이블의 모든 행을 더함으로써 얻어진다. 또한 이렇게 더해진 결과는 중첩을 방지하기 위해 분리되는 과정을 거치게 된다. 즉, 아래의 예 ①의 경우를 보면 두 테이블의 모든 행을 더한 결과는 $\begin{bmatrix} -1- \\ --1 \end{bmatrix}$ 이 되지만, {- - 1} 에서 {- 1 -} 와 중복되는 부분인 {- 1 1} 을 분리해 내면 {- 0 1} 가 남게 되므로 최종 결과는 $\begin{bmatrix} -1- \\ -01 \end{bmatrix}$ 이 됨을 알 수 있다[11]. 한편, 각 테이블 간의 논리곱은 두 테이블의 각 행들 간에 행해지며, 대상이 되는 두 행에서 대응되는 변수 간의 연산은 부울 대수에서와 마찬가지로 행해진다. 단, "-" 와 "1" (혹은 "0") 의 연산 결과는 "1" (혹은 "0") 로 나타난다. 다음은 테이블 대수 표현과 기본적인 연산과 분리의 과정을 예시한 것이다. "+" 와 "•" 는 Trabado[11]의 정의에서와 같이 OR 와 AND 연산 기호이다.

예: $f(A,B,C) = A \bullet (B + C) + AC' = 1$

① $B+C$:

A B C		A B C		A B C		A B C
- 1 -	+	- - 1	=	- 1 -	=	- 1 -
				- - 1		- 0 1

② $(B+C) \bullet A$:

A B C		A B C		A B C		A B C
- 1 -	+	1 - -	=	1 - -	=	1 1 -
- 0 1				- 1 -		1 0 1
				- 0 1		

③ $A \bullet C'$:

A B C		A B C		A B C		A B C
1 - 1	+	- - 0	=	1 - -	=	1 - 0
				- - 0		

④ $A \bullet (B+C) + A \bullet C'$:

A B C		A B C		A B C
1 1 0	+	1 - 0	=	1 1
1 0 1				1 0 1
				1 0 0

따라서 f (X) = 1 에 대한 해는 {110}, {111}, {100}, {101} 이 된다.

테이블의 행들 중에서 한 개의 변수만 다른 행들은 합병 원리 (즉, XY + XY′ = X) 를 적용하여 합병시킴으로써 테이블을 간소화할 수 있다.

예: $A'D' + AB'D' + A'C'DE' + ABD'E' + AB'C'DE' + ABC'DE'$

$= A'D' + AB'D' + A'C'DE' + ABD'E' + AC'DE'$

$= A'D' + AB'D' + C'DE' + ABD'E'$

$$\begin{vmatrix} 0 & - & - & 0 & - \\ 1 & 0 & - & 0 & - \\ 0 & - & 0 & 1 & 0 \\ 1 & 1 & - & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 \end{vmatrix} = \begin{vmatrix} 0 & - & - & 0 & - \\ 1 & 0 & - & 0 & - \\ 0 & - & 0 & 1 & 0 \\ 1 & 1 & - & 0 & 0 \\ 1 & - & 0 & 1 & 0 \end{vmatrix} = \begin{vmatrix} 0 & - & - & 0 & - \\ 1 & 0 & - & 0 & - \\ - & - & 0 & 1 & 0 \\ 1 & 1 & - & 0 & 0 \end{vmatrix}$$

Unger는 Trabado 등이 제안한 테이블 대수 식을 발전시켜 매트릭스 형의 테이블을 간소화는 합병 원리와 보수를 구하는 방법을 제안하였다[2]. 논문 [1]에서 제안된 방법은 f (X) = 1 인 함수의 해를 구할 수 있었으나, 이 방법은 바로 f (X)의 보수를 취함으로서 f (X) = 0 인 함수의 해를 구하는 데에 활용될 수 있다.

한 행이 n개의 부울 변수로 된 테이블 T1의 보수가 되는 테이블 T1′는 n개 이하의 행들로 구성된다. T1에서 행의 원래 위치에 있는 요소가 "-" 면 T1′에서 해당 행의 대응하는 위치에 있는 요소도 "-" 가 된다. T1에서 "-" 가 아닌 첫 요소의 보수를 취하여 T1′의 첫 행에서 대응 위치에 있는 요소로 만들고 나머지 요소들은 모두 "-" 로 만든다. T1에서 그 다음 요소 중 "-" 가 아닌 요소가 발견되면 T1′의 다음 행의 해당 위치에 그 요소의 보수를 취한 값을 복사하고 이후의 요소들은 모두 "-" 로 만든다. 이때 T1′의 먼저 행에서 보수를 취했던 요소는 다시 보수를 취하여 (즉, 원래의 값이 된다) 이를 현재 행의 해당 위치에 복사한다. 마찬가지 방식으로 반복해 나가면 T1′에서의 행의 수는 T1에서 "-" 가 아닌 요소의 수와 같게 된다.

예:

$$(C'EFG')' = C + C'E' + C'EF' + C'EFG$$

$$T_1 = \frac{ABCDEFG}{|--0-110|} = C'EFG'$$

$$T_1' = \left|\begin{array}{cccc} - & - & 1 & - \\ - & - & 0 & - \\ - & - & 0 & 1 \\ - & - & 1 & 1 \end{array}\right| \quad C + C'E' + C'EF' + C'EFG$$

한편 두개 이상의 행으로 된 테이블에 대한 보수를 구하기 위해서는 드모르간의 정리를 이용하여 각 행들의 보수를 구한 다음 그 결과들을 AND 연산한다. 다음은 그 예를 나타낸 것이다.

예:

$$T_2 = \left|\begin{array}{cccc} 0 & 1 & - & 0 \\ 1 & - & - & 0 \\ - & - & 1 & 1 \end{array}\right|$$

$$T_2' = \left|\begin{array}{cccc} 1 & - & - & - \\ 0 & 0 & - & - \\ 0 & 1 & - & 1 \end{array}\right| \left[\begin{array}{cccc} 0 & - & - & - \\ 1 & - & - & 1 \end{array}\right] \left[\begin{array}{cccc} - & - & 0 & 1 \end{array}\right] \left|\begin{array}{cccc} 1 & - & 0 & 1 \\ 0 & 0 & 0 & - \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{array}\right|$$

스위칭 함수의 일반적인 형태인 $f(X) = g(X)$ 의 해는 $f(X)g'(X) + f(X)g(X) = 1$의 해와 같다[2]. 예를 들어 $A'BC + AB'D = B'C' + CD$ 의 경우를 생각해 보자. $A'BC + AB'D$ 를 $f(X)$ 라 하고 $B'C' + CD$ 를 $g(X)$ 라 하여 이를 테이블 형으로 나타내고 해를 구하면 다음과 같다.

$$f(X) = \left[\begin{array}{cccc} 0 & 1 & 1 & - \\ 1 & 0 & - & 1 \end{array}\right], \quad g(X) = \left[\begin{array}{cccc} - & 0 & 0 & - \\ - & - & 1 & 1 \end{array}\right]$$

따라서, $f'(X)g'(X) + f(X)g(X) = 1$ 의 해는 다음과 같이 된다.

$$\left|\begin{array}{cccc} 1 & - & - & - \\ 0 & 0 & - & - \\ 0 & 1 & 0 & - \end{array}\right| \left|\begin{array}{cccc} 0 & - & - & - \\ 1 & 1 & - & - \\ 1 & 0 & - & 0 \end{array}\right| \left[\begin{array}{cccc} - & 1 & - & - \\ - & 0 & 1 & - \end{array}\right] \left[\begin{array}{cccc} - & - & 0 & - \\ - & - & 1 & 0 \end{array}\right]$$

$$+ \left[\begin{array}{cccc} 0 & 1 & 1 & - \\ 1 & 0 & - & 1 \end{array}\right] \left[\begin{array}{cccc} - & 0 & 0 & - \\ - & - & 1 & 1 \end{array}\right]$$

$$= \left|\begin{array}{cccc} 1 & 1 & 0 & - \\ 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & - \end{array}\right| + \left|\begin{array}{cccc} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 \end{array}\right| = \left|\begin{array}{cccc} - & 1 & 0 & - \\ 1 & - & 1 & 0 \\ 0 & 0 & 1 & 0 \end{array}\right| + \left[\begin{array}{cccc} 0 & 1 & 1 & 1 \\ 1 & 0 & - & 1 \end{array}\right]$$

$$= \left|\begin{array}{cccc} 0 & 1 & 1 & 1 \\ 1 & 0 & - & 1 \\ - & 1 & 0 & - \\ 1 & - & 1 & 0 \\ 0 & 0 & 1 & 0 \end{array}\right| \qquad (2)$$

이는 해의 유일한 표현은 아니지만 부울 연산법에 의한 해와 같음을 알 수 있다.

Ⅲ. 배타적 논리합의 연산 방법

위에서와 같은 $f(X) = g(X)$ 와 같은 일반적인 형의 함수나 동시 함수의 해를 구하는 데 있어 Unger의 보수 기법은 보수와 함께 드모르간의 법칙을 사용해야 하므로 복잡한 면이 있다. 이는 테이블 대수식 간의 배타적 논리합 연산 (exclusive OR)에 의해 개선될 수 있다. 본 논문에서는 비교적 간단히 구현할 수 있는 테이블 식을 위한 배타적 논리합 (이하 TXOR라 하고 ⊕T 로 표시한다)을 위한 규칙들을 제안한다.

먼저 하나의 행으로 구성된 두 테이블 대수식을 각기 T_a 와 T_b라 하고 T_a 와 T_b를 TXOR 연산한 결과를 T_{ab} 라 하자. 즉,

$$T_{ab} = T_a TXOR T_b = T_a \oplus_T T_b$$

테이블의 각 행이 k개의 변수를 가질 때 이 연산 결과로 얻어지는 테이블 T_{ab}는 $2k$개 이하의 행으로 구성된다. T_{ab}의 각 행의 변수는 두 테이블 T_a와 T_b 내의 각기 i 번째 대응하는 두 변수 간의 TXOR 연산에 의해 결정된다 (이때 i는 k보다 작은 수이다). T_a 에서 "-" 를 제외한 모든 변수들 (즉, "0" 과 "1")을 한번에 하나씩 선택한 후, T_a의 모든 변수와 다른 테이블 T_b의 모든 변수들과 각각 TXOR 한다. TXOR 연산의 일반 규칙은 표 2와 같다. 이렇게 하면 T_a에서 선택된 한 변수당 한 개의 행이 T_{ab}에 만들어진다. T_a에서 T_b로의 연산이 완료되고 나면 Tb에서 T_a로도 같은 처리를 반복한다. 즉, TXOR 연산의 일반 규칙

자체는 교환성이 없다. 그러나 TXOR 연산은 XOR 연산과 같이 교환성이 있다. 이렇게 하여 만들어진 T_{ab}의 행들 중에서 중복되는 행이 있으면 하나만 남기고 제거한다. 따라서 테이블 T_{ab}의 행 수는 총 2k개 이하가 된다.

표 2. TXOR 규칙
Table 2. TXOR rules

규칙	기준 테이블	대상 테이블	결과
1	선택된 "1"	"-"	"0"
2	선택된 "0"	"-"	"1"
3	"-"	"1"	"1"
4	"-"	"0"	"0"
5	모든 "0"	"1"	"1"
6	모든 "1"	"0"	"0"
7	선택 안된 "1"	"-"	"-"
8	선택 안된 "0"	"-"	"-"
9	모든 "1"	"1"	"1"
10	모든 "0"	"0"	"0"
11	"-"	"-"	"1"

(주: 모든 "0" : 선택된 "0" 과 선택 안된 "0")

예를 들어 T_a가 {1 0 0 - 1} 이고 T_b가 {- 0 1 1 -} 라 한다면, 이는 각각 부울 식 $AB'C'E$ 와 $B'CD$ 에 해당된다. T_{ab} (즉 $T_{ab} = Ta \oplus_T Tb = AB'C'E \oplus B'CD$) 의 첫째 행은 T_a의 첫 변수 "1" 을 선택함으로써 얻어진다. 표 2의 규칙에 의해 결과 테이블 T_{ab}의 첫째 행의 첫 변수는, 선택된 "1" 과 T_b의 첫 변수 "-" 간의 TXOR 연산에 의해 "0" 이 된다. T_{ab}의 첫째 행의 두 번째 변수는 T_a의 두 번째 변수가 "0" 이고 T_b의 상대 변수도 "0" 이므로 "0" 이 된다. 마찬가지 방식으로 하여 셋째와 넷째 변수는 "0" 과 "1" 로부터 "1"이 되고, "-" 와 "1" 로부터 "1" 이 되며, 끝으로 다섯째 변수는 선택되지 않은 "1" 과 "-" 를 TXOR 하므로 "-" 가 된다. 이와 같이 하여 T_{ab}의 첫 행 {0 0 1 1 -} 이 만들어지고 난 다음, T_a의 두 번째 변수 "0" 이 선택되고 T_a과 T_b의 상대 변수들을 각기 TXOR 하여 T_{ab}의 둘째 행 {- 0 1 1 -} 이 얻어진다. 이런 식으로 T_a의 마지막 변수 "1" 까지 선택이 되고 나면 T_{ab}의 넷째 행까지 만들어진다. 이 과정에서 T_{ab}의 둘째 행과 셋째 행은 {- 0 1 1 -} 로서 중복이 되므로 이 중 하나만 선택하면 세개의 행으로 줄어들게 된다. 다음에는 T_b를 기준으로 하여 한번에 하나씩 차례로 변수를 선택하면서 T_a의 상대 변수와 TXOR 연산을 해 나가면 {1 0 0 - 1}, {1 0 0 - 1}, {1 0 0 0 1} 의 세 행을 얻게 되고 이 중에서 중복 행 {1 0 0 - 1} 을 제거하면, T_{ab}의 나머지 행은 {1 0 0 - 1} 과 {1 0 0 0 1} 이 된다. 이렇게 해서 얻어진 총 5개의 행은 Unger의 간소화 방법을 사용하여 다시 다음과 같이 두개의 행으로 줄일 수 있다.

$$T_{ab} = [\,1\ 0\ 0\ -\ 1\,] \oplus_T [\,-\ 0\ 1\ 1\ -\,]$$

$$= \begin{vmatrix} 0&0&1&1&- \\ -&0&1&1&- \\ -&0&1&1&- \\ -&0&1&1&0 \\ 1&0&0&-&1 \\ 1&0&0&-&1 \\ 1&0&0&0&1 \end{vmatrix} = \begin{vmatrix} 0&0&1&1&- \\ -&0&1&1&- \\ -&0&1&1&0 \\ 1&0&0&-&1 \\ 1&0&0&0&1 \end{vmatrix} = \begin{bmatrix} -&0&1&1&- \\ 1&0&0&-&1 \end{bmatrix}$$

다음으로, 각기 m 개의 행과 n 개의 행으로 된 두 테이블 T_a와 T_b를 경우를 생각해 보자. 결과 테이블 T_{ab} (즉, $T_{ab} = Ta \oplus_T Tb$)는 다음과 같이 하여 구할 수 있다.

$$\begin{aligned}
T_{AB} = {} & (T_{A,1} \oplus_{TL} T_{B,1}) \cdot (T_{A,2} \oplus_{TL} T_{B,1}) \\
& \cdot \cdots \cdot (T_{A,(m-1)} \oplus_{TL} T_{B,1}) \cdot (T_{A,m} \oplus_{TL} T_{B,1}) \\
+ \{ & (T_{A,1} \oplus_{TL} T_{B,2}) \cdot (T_{A,2} \oplus_{TL} T_{B,2}) \\
& \cdot \cdots \cdot (T_{A,(m-1)} \oplus_{TL} T_{B,2}) \cdot (T_{A,m} \oplus_{TL} T_{B,2}) \} \\
& \vdots \\
+ \{ & (T_{A,1} \oplus_{TL} T_{B,n}) \cdot (T_{A,2} \oplus_{TL} T_{B,n}) \\
& \cdot \cdots \cdot (T_{A,(m-1)} \oplus_{TL} T_{B,n}) \cdot (T_{A,m} \oplus_{TL} T_{B,n}) \} \\
+ \{ & (T_{B,1} \oplus_{TL} T_{A,1}) \cdot (T_{A,2} \oplus_{TL} T_{A,1}) \\
& \cdot \cdots \cdot (T_{B,(n-1)} \oplus_{TL} T_{A,1}) \cdot (T_{B,n} \oplus_{TL} T_{A,1}) \} \\
+ \{ & (T_{B,1} \oplus_{TL} T_{A,2}) \cdot (T_{A,2} \oplus_{TL} T_{A,2}) \\
& \cdot \cdots \cdot (T_{B,(n-1)} \oplus_{TL} T_{A,2}) \cdot (T_{B,n} \oplus_{TL} T_{A,2}) \} \\
& \vdots \\
+ \{ & (T_{B,1} \oplus_{TL} T_{A,m}) \cdot (T_{B,2} \oplus_{TL} T_{A,m}) \\
& \cdot \cdots \cdot (T_{B,(n-1)} \oplus_{TL} T_{A,m}) \cdot (T_{B,n} \oplus_{TL} T_{A,m}) \}
\end{aligned}$$

단, "+" 와 "•" 는 Trabado의 정의에서와 같이 OR 와 AND 연산 기호이다. T_{ij}는 i 테이블의 j 번째 행을 나타내며 $0 \le i \le m$ 이고 $0 \le j \le n$ 이다. 한편, "$\oplus_{TL}$" 는 양 방향 모두에 적용되는 TXOR 연산을 의미하는 반면, "$\oplus_{TL}$" 은 왼쪽 열에서 오른쪽 열로의 방향에만 적용되는 TXOR 연산을 의미한다. 만일 동일한 두 행을 TXOR 연산하면 결과 행은 모두

"0" 이 된다. 다음절에서 그 예제를 볼 수 있다.

Ⅳ. 예제 ; 스위칭 함수의 해

Unger의 논문에서는 일반적인 스위칭 함수의 형태인 f (X) = g (X) 를 만족하는 해는 함수 $f(X)g'(X) + f(X)g(X) = 1$을 만족하는 해와 같다[2]. 또, $f(X)g'(X) + f(X)g(X) = 0$ 을 만족하는 해는 $f(X)g'(X) + f(X)g(X) = 1$을 만족하는 해의 보수가 된다. 그리고 이는 f (X) ⊕ (X) = 0 로도 나타낼 수 있다. 즉, 만일 f (X)와 g(X)가 테이블 형이면 f(X) = g(X) 를 만족하는 해는 f(X) $\oplus_T$ g(X) = 0 을 만족하는 해의 보수와 같다.

표 3. T_{CD} 의 진리표
Table 3. Truth Table for T_{CD}

	A B C D	T_{CD}
1	0 0 0 0	1
2	0 0 0 1	1
3	0 0 1 0	0
4	0 0 1 1	1
5	0 1 0 0	0
6	0 1 0 1	0
7	0 1 1 0	1
8	0 1 1 1	0
9	1 0 0 0	0
10	1 0 0 1	1
11	1 0 1 0	0
12	1 0 1 1	0
13	1 1 0 0	0
14	1 1 0 1	0
15	1 1 1 0	1
16	1 1 1 1	1

예를 들어 설명하여 보자. Unger의 논문과 비교하기 위하여 Unger가 사용한 예와 같은 함수인 $A'BC + AB'D = B'C' + CD$ 를 이용한다.

$$f(X)=\begin{bmatrix} 0&1&1&-\\ 1&0&-&1 \end{bmatrix},\ g(X)=\begin{bmatrix} -&0&0&-\\ -&-&1&1 \end{bmatrix}$$

테이블 연산을 적용하여 f (X)와 g(X)를 TXOR 한 결과를 테이블 TCD 라 할 때, (즉, TCD = f(X) $\oplus_T$ g(X)) TCD는 다음과 같이 하여 얻어진다.

$$T_{CD}=\begin{bmatrix} 0&1&1&-\\ 1&0&-&1 \end{bmatrix}\ \text{TXOR}\ \begin{bmatrix} -&0&0&-\\ -&-&1&1 \end{bmatrix}$$

$$=\begin{bmatrix} -&0&0&-\\ -&0&0&- \end{bmatrix}\cdot\begin{bmatrix} 0&0&0&-\\ -&0&0&0 \end{bmatrix}+\begin{bmatrix} 1&-&1&1\\ -&0&1&1 \end{bmatrix}\cdot\begin{bmatrix} 0&-&1&1\\ -&1&1&1 \end{bmatrix}$$

$$+[011-]\cdot[0110]+[1011]\cdot[1001]$$

$$=\begin{vmatrix} 1&0&0&0\\ 0&0&0&-\\ -&0&0&0\\ 1&1&1&1\\ 0&0&1&1\\ 0&1&1&0 \end{vmatrix}=\begin{vmatrix} 0&0&0&-\\ -&0&0&0\\ 1&1&1&1\\ 0&0&1&1\\ 0&1&1&0 \end{vmatrix}\Rightarrow\begin{vmatrix} 0&1&1&1\\ 1&0&-&1\\ -&1&0&-\\ 1&-&1&0\\ 0&0&1&0 \end{vmatrix}$$

의 보수가 된다.

이 결과는 표 3의 진리표에서 T_{CD}가 1 인 경우에 해당된다. Unger의 보수 기법에 의해 얻어진 결과인 식 (1)은 이 표에서 T_{CD} 가 0 인 경우에 해당되므로 이 두 결과가 보수 관계임을 알 수 있다.

위의 예는 제안 방법이 테이블 형의 배타적 논리합을 적용하여 동시 함수의 해를 직접 구할 수 있음을 보여 준다. 최종적인 함수의 해는 결과로 얻어진 테이블에 대한 보수를 취하여 얻을 수 있다.

Ⅴ. 결 론

부울 함수를 사용하는 대수적인 방법은 스위칭 함수의 해를 구하기가 쉽지 않고 복잡하며, 컴퓨터를 사용하기에는 적절하지 못한 단점이 있다. 그러나, 테이블 방식에서도 B^n의 n 값이 커질 때는 매우 큰 용량의 메모리를 요구하게 되므로 컴퓨터로 실현하는 것이 실용적이 아니다. 따라서, 스위칭 함수에 필요한 변수의 수를 최대한 줄이는 연구가 필요하다. 이 문제는 테이블 대수식 간의 배타적 논리합 (Exclusive OR) 연산을 통해 개선할 수 있다. 본 논문에서는 테이블 대수식의 배타적 논리합 연산을 위한 규칙들을 제안하였으며, 이를 Unger가 사용한 것과 같은 예에 적용하여 같은 결과를 얻었다. 이는 본 제안 규칙을 사용하는 경우 Unger가 제안한 보수 기법을 사용하는 것보다 $f(X) = g(X)$ 와 같은 유형의 스위칭 함수나 동시 함수의 해를 직접적이고 효과적인 방법으로 구할 수 있음을 입증하는 것이다.

참 고 문 헌

[1] A. L. Ruiz, P. P. Trabado, and J. O. Lopera, "Solution of switching equations based on a tabular a tabular algebra," *IEEE. Trans. Comput.*, vol.42, no.5, pp.591-596, May 1993.

[2] S. H.Unger, "Some additions to solution of switching equations based on a tabular agebra," *IEEE. Trans. Comput.*, vol.43, no.3, pp.365-367, March 1994.

[3] S. Rudeanu, "An algebraic approach to Boolean equations," *IEEE. Trans. Comput.*, pp.206-207, March 1974.
[4] W. Del Picchia, "A numerical algorithm for the resolution for Boolean equations," *IEEE. Trans. Comput.*, vol.C-23, pp.983-986, Sept. 1974.

저 자 소 개

鄭 華 子(正會員) 第 32卷 B編 第 2號 參照.

鄭 己 鉉(正會員) 第 32卷 B編 第 2號 參照.

論文95-32B-6-6

決定다이아그램에 의한 多値組合論理시스템 構成에 관한 硏究

(A Study on Constructing the Multiple-Valued Combinational Logic Systems by Decision Diagram)

朴 春 明 *, 金 興 壽 **

(Chun Myoung Park and Heung Soo Kim)

요 약

본 논문에서는 決定다이아그램에 의한 多値組合論理시스템構成 방법을 제시하였다. 우선, 多値組合論理시스템의 스위칭함수 진리치표를 literal에 의한 論理積의 合의 기본전개식으로 고친 다음, 이 전개식을 多値論理決定다이아그램으로 전환한다. 이때 變數順序選定이 대단히 중요하며, 變數順序에 따라 多値論理決定다이아그램은 서로 아주 달라진다. 때로는 變數順序選定이 적절치 못하면 방대한 多値論理決定다이아그램이 되어 최소화하기가 불가능할때도 있다. 多値論理決定다이아그램의 규모가 방대해지면 실제로 실현되는 회로도 방대해짐은 물론이다. 본 논문에서는 最小多値論理決定다이아그램을 얻기위한 變數順序選定을 결정하는 방법의 알고리듬을 제안하였으며, 예를 들어서 이 알고리듬의 타당성을 입증하였다. 그리고 最小多値論理決定다이아그램에 의하여 T-gate를 사용한 실제회로를 실현하였다.

Abstract

This paper presents a method of constructing the multiple-valued combinational logic systems(MVCLS) by decision diagram. The switching function truth table of MVCLS is transformed into canonical normal form of sum-of-products(SOP) with literals at first. Next, the canonical normal form of SOP is transfered into multiple-valued logic decision diagram(MVLDD). The selecting of variable ordering is very important in this stage. The MVLDDs are quite different from each other according to the variable ordering. Sometimes the inadequate variable ordering produces a very large size of MVLDD and it is very hard to minimize the MVLDD. The large size of MVLDD means the large size of circuit implementation. An algorithm for generating the prōper variable ordering produce minimal MVLDD and an example shows the verity of the algorithm. The circuits are realized with T-gate according to the minimal MVLDD.

*正會員, 馬山專門大學 電子制御科

(Dept. of Elec. Cont., Masan Junior College)

**正會員, 仁荷大學校 電子工學科

(Dept. of Elec. Eng., Inha Univ.)

接受日字: 1994年4月26日, 수정완료일: 1995年5月27日

I. 序 論

최근에 決定다이아그램에 기초한 組合論理시스템構成에 관한 연구가 활발히 진행중에 있다. R.E. Bryant[1]는 S.B.Akers[2]의 2値論理決定 다이아그램(Binary Decision Diagram : BDD)에 기초하여

부울함수를 방향이 있는 비사이클릭 그래프(Directed Acyclic Graph) 형태의 데이타 구조로 표현하여 부울함수를 간략화하는 방법을 최초로 제안하였으며, 그래프이론을 이용한 組合論理시스템構成 방법의 효시가 되고 있다. 한편, H.T.Liaw와 C.S.Lin[3]은 R.E.Bryant의 연구를 토대로 부울함수에 대한 순서화된 2値論理決定다이아그램(Ordered BDD : OBDD)의 동작에 대해 논하였으며 OBDD의 크기는 變數順序選定에 의존함을 보였고 합병규칙과 제거규칙을 제안하여 축소 순서화된 2値論理決定다이아그램(Reduced OBDD : ROBDD)을 구하는 방법을 논하였다.

그외 S.Chakravarty[4]는 OBDD의 여러가지 성질을 논하였고, A.H.Chan[5]은 多値論理 입력을 갖는 부울함수의 決定트리(Decision Tree)를 축소하는 알고리듬을 제안하였으며 이를 함수의 보수를 얻는데 이용하였다. 또한 T.Sasao[6]는 多位置決定다이아그램을 사용해 多値論理 입력/2値論理 출력의 pseudo-kronecker 표현에 대한 최적화 방법을 제안하였으며 M.A.Perkowski[7]는 多値論理 입력/2値論理 출력의 일반화된 정규직교확장을 GRM(Generalized Reed Muller), ESOP(Exclusive OR Sum of Product), KRM(Kronecker Reed Muller) 등의 표현을 사용하여 함수도출을 시도하였다.

그러나 이들 방법들은 비록 입력은 多値論理 형태이지만 출력은 모두 2値論理 형태로 국한하여 처리하였으므로 순수 多値論理 입출력 형태의 組合論理시스템을 다룬것은 아니다.

한편, D.M.Miller[8]는 R.E.Brayant의 순서화된 2値論理決定다이아그램에 대한 연구를 최초로 多値論理 입출력인 경우로 확장 적용하는 알고리듬과 축소하는 방법의 한가지를 제안하였다.

이상의 決定다이아그램에 기초를 둔 組合論理시스템構成 방법은 變數順序選定에 의해 축소된 決定다이아그램의 추출이 좌우되는 단점이 있다. 이러한 점을 고려하여 본 논문에서는 決定다이아그램에 기초를 둔 多値組合論理시스템構成의 한가지 방법을 제안한다.

본 논문의 서술 과정은 다음과 같다. Ⅱ장에서는 본 논문을 전개하는데 필요한 數學的 背景으로 多値論理 함수를 literal에 의한 論理積의 合의 형식인 Shannon의 전개식을 소개한 다음 多値論理決定다이아그램의 構成에 대해서 설명하였다. Ⅲ장에서는 2値論理決定다이아그램에 관한 기존 논문의 간략화 방법을 소개한 다음 多値論理決定다이아그램 사용시의 적절한 變數順序選定의 필요성을 지적하였다. 그리고 Ⅳ장에서는 變數順序選定에 관한 알고리듬을 제안하고 예를 들어 알고리듬의 적용 절차를 설명하였다. Ⅴ장에서는 Ⅳ장의 알고리듬에 의해 얻어진 最小多値論理決定다이아그램을 토대로 T-gate를 사용하여 실제회로를 실현하는 방법에 대해 논하였으며 기존 논문들과의 比較,檢討를 하였다. 마지막 Ⅵ장에서는 본 논문의 結論을 기술하였다.

Ⅱ. 數學的 背景

1. 多値論理函數의 數學的 표현

일반적으로 多値論理函數의 數學的 표현은 대별해서 3가지로 분류된다. 그 하나는 literal에 의한 論理積의 合(canonical Sum of-Products : SOP)의 형식인 Shannon의 전개식이고, 또 하나는 Post 代數에 근거를 둔 전개식이며 또 하나는 Galois體에 근거를 둔 일반 Reed-Muller 전개식이다.[9-12] 이들 3가지 표현식이 나름대로 다 특색이 있지만 본 논문에서는 literal에 의한 論理積의 合의 표현을 근거로 하여 多値組合論理시스템을 구성하고자 한다. 먼저, Shannon의 전개식을 사용하기 전에 다음과 같은 定義를 한다.

(定義 1)

X_i를 집합 $R_i=\{0,1,2,\ \ldots\ ,(r-1)\}$ 중의 임의의 값을 취하는 多値論理變數라고한다. 임의의 부분집합 $B_i \subseteq R_i$에 대해서, $X_i^{B_i}$를 다음 내용을 표현하는 literal이라고 한다.

$$X_i^{B_i}=\begin{cases} 1 & \text{if } X_i \in B_i \\ 0 & \text{if } X_i \notin B_i \end{cases}$$

(定義 2)

literal의 積 $X_1^{B1}X_2^{B2}\ldots X_n^{Bn}$ 을 積項(product)이라고 한다.

(定義 3)

$F(X)=\sum_{Q=0}^{r-1} Q\cdot X_1^{I1}X_2^{I2}\ldots X_n^{In}$ 와 같은 積項의 合으로 표현되는 전개식을 Disjunctive Normal Form(DNF)이라고 한다.

위의 定義에 따라 표1과 같은 2變數 4値組合論理시스템의 스위칭 함수에 대한 DNF 표현을 구하면 다음식(1)과 같다.

표 1. 2變數 4値組合論理시스템의 스위칭 함수에 대한 진리치표

Table 1. Truth table for switching function of a 2-variable 4-valued combinational logic system

X_2 \ X_1	01	1	2	3
0	1	1	0	3
1	2	2	0	3
2	0	0	0	0
3	0	2	2	0

$$F(X_1, X_2) = X_1^0X_2^0 + X_1^1X_2^0 + 3X_1^3X_2^0 + 2X_1^0X_2^1 + 2X_1^1X_2^1 + 3X_1^3X_2^1 + 2X_1^1X_2^3 + 2X_1^2X_2^3 \quad (1)$$

또한, 위의 식은 다음 식(2)와 같이 축소할 수 있다.

$$F(X_1, X_2) = X_1^{(0,1)}X_2^0 + 2X_1^{(1,2)}X_2^3 + 2X_1^{(0,1)}X_2^1 + 3X_1^3X_2^{(0,1)} \quad (2)$$

이와 같은 전개식을 2値論理인 경우에는 Shannon의 SOP 형식이라고 칭하는데 多値論理에 대한 전개식의 경우에도 이 명칭을 그대로 사용한다.

2. 決定다이아그램(Decision Diagram)

決定다이아그램은 접점(node)과 지로(branch)로 구성되는 방향성 그래프(directed graph)이며 접점은 論理 變數에, 지로는 입력 變數에 각각 대응된다. 여기서 한 論理 變數에 대응하는 접점은 여러개 존재할 수 있으며 각 접점에서 부터 나가는 지로는 입력 變數의 갯수와 일치한다.

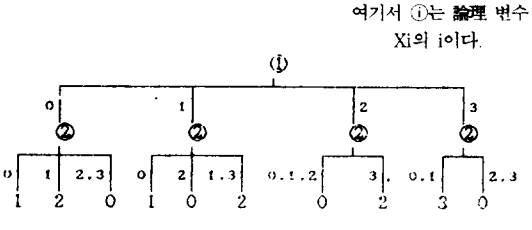


그림 1. 식(1)에 대한 決定다이아그램

Fig. 1. Decision diagram for expression (1)

決定다이아그램은 성격상 트리(Tree) 형태를 취하며 첫째 단 접점수는 반드시 한개이고 여기서 부터 시작하여 각 접점마다 입력 變數 갯수 만큼의 부트리가 발생한다. 決定다이아그램은 접점에 대응하는 變數가 나타나는 順序에 따라 달라지며 한 論理 變數에 대해 여러개의 決定다이아그램이 존재한다. 따라서 論理 變數를 취해가는 順序를 어떻게 할것인가가 문제로 제기된다. 예를 들어 앞의 식(1)에서 첫번째 접점을 論理 變數 X_1으로 취하고 두번째 접점을 論理 變數 X_2로 하여 決定다이아그램으로 도시하면 그림1과 같다.

Ⅲ. 決定다이아그램에 의한 해석

본 장에서는 決定다이아그램의 構成 및 기존 논문에서의 決定다이아그램의 축소 과정을 소개한 다음, 多値論理決定다이아그램에 대한 해석을 한다.

1. 決定다이아그램의 構成

원칙적으로 決定다이아그램은 진리치표로 부터 작성되는 원시적 決定다이아그램(primitive decision diagram)이 그 근본이 된다. 그렇지만 이와같은 원시적 決定다이아그램은 그 규모가 일반적으로 방대해짐으로 Ⅱ장에서 설명한 論理積의 合의 형식의 전개식으로 부터 작성하는 것이 유리하다. 그림1에서 접점수는 5개인데 만일 어떤 방법이 있어서 접점수를 감소시킬 수 있다면 실제회로를 설계할 때 줄어든 접점수 만큼의 게이트가 감소되어 회로가 간단화 될 수 있다. 일반적으로 n變數함수에 있어서 첫째 단 變數를 취하는 방법은 n개, 두번째 단 變數를 취하는 방법은 (n-1)개, 세째 단 變數를 취하는 방법은 (n-2)개 등과 같으므로 동일한 논리함수에 대해 n!개의 서로 다른 決定다이아그램이 존재한다. 이것도 각 부트리에 대한 접점 順序가 같다는 전제조건하에서 이며, 각 부트리에 있어서의 變數 順序가 다를때는 이보다 훨씬 더 많은 決定다이아그램이 존재한다. 따라서 만일 무작위로 變數 順序를 취해서 決定다이아그램을 작성한 다음 이 決定다이아그램을 축소하여 간략화하는 작업은 무의미한 것이된다. 그러므로 처음부터 變數 順序를 決定해 가면서 한번의 시도에 의하여 最小決定다이아그램을 얻을 수 있다면 가장 이상적이다. 본 장에서는 이 문제의 해결 방법에 대해 논한다. 본 논문에서 취급하는 決定다이아그램은 자유 決定다이아그램이며 비순서 決定다이아그램이다.[4]

2. 多値論理函數에 대한 決定다이아그램

본 절에서는 Ⅱ장에서 설명한 literal을 사용한 Shannon의 전개식을 근거로 多値論理決定다이아그램의 構成에 대해서 논하기로 한다. 표1의 진리치표에 대한 Shannon의 전개식은 식(1) 또는 식(2)와 같으며 變數 X_1을 첫째 단의 접점으로 취할때의 決定다이

아그램은 그림1과 같다. 만일 變數 X_2를 첫째 단의 접점으로 취하면 그림2의 決定다이아그램을 얻는다.

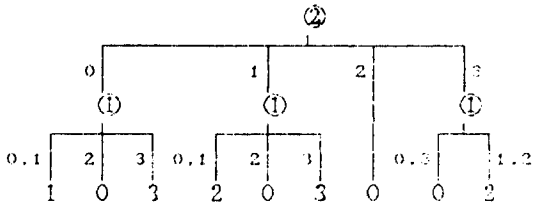


그림 2. 식(1)에 대한 決定다이아그램
Fig. 2. Decision diagram for expression (1).

이 두 決定다이아그램을 비교하면 그림1에서는 접점수가 5개인데 그림2에서는 접점수가 4개이다. 즉, 부트리가 하나 감소한 셈이며 회로를 실현시킬 때 多値論理게이트 수가 하나 절약됨을 알 수 있다. 이와같이 동일 함수에 대해서도 각 단의 접점에 대응하는 變數의 順序를 달리함으로써 多値論理決定다이아그램의 축소가 각각 달라질 수 있다. 기존의 決定다이아그램의 축소 방법들은 소규모의 決定다이아그램이나 또는 대규모 決定다이아그램의 부분트리에 적용하여 효과를 볼 수 있지만 대규모의 決定다이아그램 전체에 대한 종합적인 축소나 多變數 多値論理決定다이아그램에 대해서는 큰 효과를 볼 수 없다. 따라서 多變數 多値論理函數의 構成에 관해서는 별도의 방법이 제안되어야 할 것이다. 일반적으로 n變數 函數에 대해서는 서로 다른 多値論理決定다이아그램의 수가 최소한 n!개 이상이므로 이와같은 방대한 수의 多値論理決定다이아그램을 일일이 검토하여 最小多値論理決定다이아그램을 구하는 것은 불가능한 일이다. 따라서 最小多値論理決定다이아그램을 구하기 위한 합리적인 방법이 제안되어야 할것이다. 다음 장에서는 이를 위한 알고리듬을 제안한다.

Ⅳ. 알고리듬 및 적용 예

본 장에서는 Shannon의 전개식으로 부터 직접 最小多値論理決定다이아그램을 구하는 알고리듬을 제시하고, 이 알고리듬에 의하여 最小多値論理決定다이아그램을 구하는 과정을 보인다.

1. 最小多値論理決定다이아그램을 구하기 위한 알고리듬

(절차 1) 각 積項에 포함되어 있는 X_i^{Bi}에 대하여 $|\cup B_i|$가 최소인 變數를 취한다. 만일 해당되는 變數가 없고 단일 變數項이 존재하면 그 變數를 취한다. 여기서 $|\cup B_i|$는 $\cup B_i$에 포함되는 입력수를 뜻하며 또한, X_i를 포함하지 않는 積項에 대한 $|B_i|$는 R_i로 간주한다. 해당 사항이 없을때는 절차2로 옮긴다.

(절차 2) 2變數 積項 $X_i^{Bi}X_j^{Bj}$ 가 존재할때는 $|B_i|$, $|B_j|$ 중에서 작은 쪽의 變數를 취한다. 만일 2變數 積項이 여러개 있을때는 이들 사이에 절차1을 적용하여 $|\cup B_i|$가 최소인 變數 X_i를 취한다. 해당 사항이 없을때는 절차3으로 옮긴다.

(절차 3) 3變數 積項까지 확대하여 $|\cup B_i|$가 최소인 變數를 취한다. 그래도 해당 사항이 없을때는 순차적으로 4變數 積項, 5變數 積項 등으로 확대해서 $|\cup B_i|$가 최소인 變數를 취한다. 해당 사항이 없을때는 절차4로 옮긴다.

(절차 4) 가장 많은 積項에 포함되는 變數 X_i를 취한다.

(절차 5) 이상의 절차가 다 무효일때는 임의로 變數를 취한다.

2. 알고리듬의 적용 예

본 절에서는 앞의 最小多値論理決定다이아그램을 구하기 위한 알고리듬이 어떻게 적용되는지를 예를 들어 설명한다.

예) 다음 표2와 같은 3變數 3値組合論理시스템의 스위칭 함수에 대해 알고리듬을 적용하면 다음과 같다. 먼저, 표2를 論理積의 合의 形式인 Shannon의 전개식으로 표현하면 식(3)과 같다.

표 2. 3變數 3値組合論理시스템의 스위칭 함수에 대한 진리치표
Table 2. Truth table for a switching function of 3-variable 3-valued combinational logic systems.

	X_1X_2	00	01	02	10	11	12	20	21	22
X_3	0	0	1	1	1	0	0	0	0	0
	1	1	1	1	0	1	1	0	1	1
	2	0	0	0	1	1	1	1	1	0

$$F(X_1,X_2,X_3)= X_1^0X_3^1+X_1^1X_3^2+X_2^{(1,2)}X_3^1+ X_1^{(1,2)}X_2^{(0,1)}X_3^2+X_1^1X_2^0X_3^{(0,2)}+ X_1^0X_2^{(1,2)}X_3^{(0,1)} \quad (3)$$

(1) 첫째 단 變數의 選定

(절차 1) $|\cup B_1|=3$, $|\cup B_2|=3$, $|\cup B_3|=3$이며

단일적항도 없으므로 해당 사항이 없다.

(절차 2) 두 變數項만 생각할때 $|UB_1|=|UB_2|=3$, $|UB_3|=2$이므로 X_3을 취한다.

(2) 두째 단 變數의 選定

$$F(X_1,X_2,0)=X_1^1X_2^0+X_1^0X_2^{(1,2)} \quad (4)$$

$$F(X_1,X_2,1)=X_1^0+X_2^{(1,2)}+X_1^0X_2^{(1,2)} \quad (5)$$

$$F(X_1,X_2,2)=X_1^1+X_1^{(1,2)}X_2^{(0,1)} \quad (6)$$

ⓐ 式(4)에 알고리듬을 적용하면 절차1에 의하여 X_1을 選定한다.

ⓑ 式(5)에 알고리듬을 적용하면 절차5에 의하여 임의로 X_2를 選定한다.

ⓒ 式(6)에 알고리듬을 적용하면 절차1에 의하여 X_1이 選定된다.

이상의 결과에 의하여 決定다이아그램을 구하면 그림3과 같다.

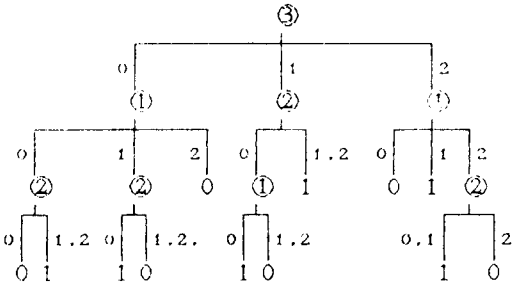


그림 3. 식(3)에 대한 最小決定다이아그램
Fig. 3. Minimal decision diagram for expression (3).

위 예는 A.H.Chan[5]의 논문에서의 예를 인용한 것이며 동일한 결과를 얻었다.

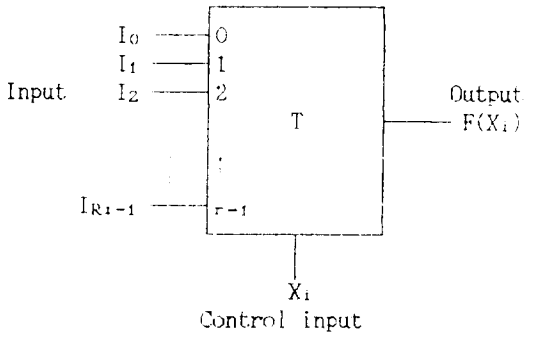
V. 多値論理回路 實現 및 檢討

본 장에서는 앞에서 구한 最小多値論理決定다이아그램을 근거로 多値論理 T-gate를 사용하여 실제로 多値組合論理의 스위칭 함수를 회로로 실현시키는 문제에 대해서 논한다.

그리고 본 논문에서 제안한 多値論理決定다이아그램에 의한 多値組合論理시스템 構成 방법을 기존의 방법들과 比較 및 檢討하였다.

1. 多値論理 T-gate의 블럭다이아그램[13,14]

多値論理 T-gate의 블럭다이아그램은 그림4와 같이 P개의 입력단자와 한개의 제어단자로 구성된다. 그림4에서 각 입력단자에는 외부로 부터 $R_i=\{0,1,2,\ldots,(r-1)\}$ 중의 한 입력이 들어오며 어느 입력단자의 입력을 출력시키는가의 선택은 제어입력에 따라 결정된다. 가령 k단자의 입력을 출력시킬려면 제어단자에 $k(k\in R_i)$를 입력시키면 된다.



where, $I_k(k=0,1,\ldots,R_i-1)\in R_i=\{0,1,2,\ldots,(r-1)\}$

그림 4. 多値論理 T-gate의 블럭다이아그램
Fig. 4. Block diagram of multiple-valued logic T-gate.

2. 回路 實現

본 절에서는 最小多値論理決定다이아그램을 근거로 T-gate를 사용한 多値論理 회로의 실현에 대해 고찰한다. 多値論理決定다이아그램을 회로로 실현하려면 출력 단의 gate는 多値論理決定다이아그램의 첫째 단 變數를 제어입력으로하고, 출력 단 gate에 연결되는 T-gate는 多値論理決定다이아그램의 두째 단 變數를 제어입력으로 한다. 즉, 多値論理決定다이아그램은 첫째 단부터 밑으로, T-gate는 출력 단 부터 역으로 대응시키면 된다.

표 3. 單一變數 3値組合論理시스템 스위칭 함수의 진리치표
Table 3. Truth table for single variable 3-valued combinational logic system switching function.

X_1	0	1	2
$F(X_1)$	2	0	1

예를 들어 위 표3과 같은 單一變數 3値組合論理

스위칭 함수를 DNF로 표현하면 다음 식(7)과 같다. 그리고 식(7)을 多値論理決定다이아그램으로 나타내면 그림5와 같고 이를 그림4의 T-gate로 회로 실현하면 그림6과 같다. 또한, 동작특성은 다음 식(8)과 같다.

$$F(X_1) = 2X_1^0 + X_1^2 \tag{7}$$

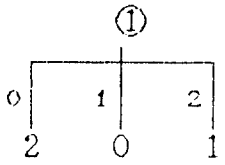


그림 5. 식 (7)에 대한 決定다이아그램
Fig. 5. Decision diagram for expression (7).

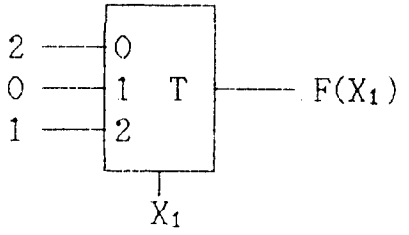


그림 6. 그림5에 대한 회로 실현
Fig. 6. Circuit realization for Fig. 5.

이에따라 앞장의 그림3을 T-gate로 회로 실현하면 다음 그림7과 같다.

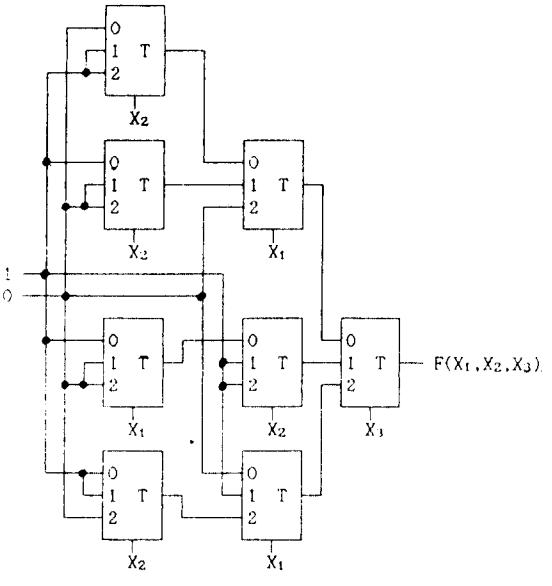


그림 7. 그림3에 대한 회로 실현
Fig. 7. Circuit realization for Fig.3.

$$F(X_1) = \begin{cases} 2 & \text{if } X_1 = 0 \\ 0 & \text{if } X_1 = 1 \\ 1 & \text{if } X_1 = 2 \end{cases} \tag{8}$$

본 논문에서 제안한 多値論理決定다이아그램에 의한 多値組合論理시스템 構成 방법을 기존의 방법들과 比較 및 檢討하면 다음 표4와 같다.

표 4. 比較 표
Table 4. Comparison table.

	比較項目		
	최소화알고리듬	다이아그램 형태	목 적
D.M.Miller[4]	*R.E.Bryant의 알고리듬을 多値論理인 경우로 확장 *처리과정: 실(thread)이라는 연장을 사용하여 동형관계 추출 *처리시간: 김	순서화다이아그램	동형관계를 갖는 부트리를 줄여 최소화시킴
A.H.Chan[5]	*Shannon의 전개식 이용 *처리과정: 비교적 복잡 $L(X_i)$와 $\delta(X_i)$로 부터 지로함수 $b(X_i)$를 구한 후 발생빈도 freq(i, j)를 추출하여 루트선정 *처리시간: 비교적 김	결정트리	함수의 보수 추출
본 논문	*Shannon의 전개식 이용 *처리과정 : 비교적 간결 각 積項에 포함된 X_iB_i에 대해 $\|UB_i\|$가 최소인 變數의 추출을 單一變數에서 부터 시작하여 多變數로 확장해 가며 變數를 선정함 *처리시간 : 비교적 짧음	자유 決定다이아그램 비순서화다이아그램	最小化 決定다이아그램 추출

위 표4에서 2値論理인 경우의 기존 논문은 比較 대상에서 제외함.

VI. 結 論

본 논문에서는 多値組合論理시스템을 構成하는데 있어서 literal에 의한 論理積의 合의 표현식으로부터 출발하여 多値論理決定다이아그램을 최소화하여 T-gate를 사용한 회로 실현에 이르는 일련의 과정을 논하였다. 본래 決定다이아그램은 2値論理決定다이아그램이 기본을 이루고 있으며, 2値組合論理시스템을 해석하는데 이를 이용한 연구는 상당수에 이루고 있으나 多値組合論理시스템에 있어서의 決定다이아그램에 대해서는 별로 취급되고 있지 않다. 2値論理決定다이아그램

에 관한 연구는 原始決定다이아그램(原型決定다이아그램)을 어떻게 축소하여 최소화 하는가에 촛점을 맞추고 있는 것이 많으며, 이와같은 접근방법은 多値組合論理시스템에는 적합하지 않다. 그 이유는 多値論理決定다이아그램에 있어서는 각 접점에 대응하는 變數를 취하는 順序에 따라 多値論理決定다이아그램의 규모가 엄청나게 달라지기 때문이다. 따라서 어떤 順序로 變數를 정해가는냐를 고찰할 필요가 생긴다. 본 논문에서는 이러한 관점에서 처음부터 最小多値論理決定다이아그램을 얻기위한 알고리듬을 설정하여 多値組合論理시스템을 構成하는 방향으로 이론을 전개하였으며 다수의 多値論理函數에 알고리듬을 적용하여 보았다. 그 결과 전부가 最小多値論理決定다이아그램임이 확인되었고 설사 最小多値論理決定다이아그램을 얻지 못하는 경우라도 最適多値論理決定다이아그램 (optimum MDD)을 얻을 수 있었으며 제안한 알고리듬은 기존의 알고리듬에 비해 보다 간결하며 처리시간이 짧다는 이점이 있다. 또한, 多値組合論理시스템 構成에 있어서 진리치표에서 부터 시작하여 실제회로 실현에 이르는 전과정을 일관성있게 취급한 논문은 거의 없다는 점을 감안할때 본 논문에서는 이에 대한 한가지 해석 방법을 제시하였다고 사료된다.

참 고 문 헌

[1] R.E.Bryant,"Graph-Based Algorithms for Boolean Function manipulation," IEEE Trans. Comput., vol.C-35, no.8, pp.677- 691, Aug. 1986.

[2] S. B.Aker,"Binary Decision Diagrams," IEEE Trans. Comput., vol.C-27, no.6, pp. 509-516, June. 1978.

[3] H.T.Liaw and C.S.Lin,"On the OBDD-Representation of General Boolean Functions," IEEE Trans. Comput., vol. 41, no.6, pp.661- 664, Jun. 1992.

[4] S.Chakravarty,"A Characterization of Binary Decision Diagrams," IEEE Trans. Comput., vol.42, no.2, pp.129-137, Feb. 1993.

[5] A.H.Chan,"Using decision trees to derive the complement of a binary function with multiple-valued inputs," IEEE Trans. Comput., vol.C-36, no.2, pp.212-214, Feb. 1987.

[6] 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.

[7] M.A.Perkowski, "The Generalized Orthonormal Expansion of Functions with Multiple-Valued Inputs and Some of Its Applications," IEEE Proc. of Symposium on Multiple-Valued Logic, Sendai Japan, pp.442-450, May. 1992.

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

[9] G.Birkhoff and T.C.Bartee, *Modern Applied Algebra*, McGraw-Hill book company, N.Y., 1970.

[10] E.Artin, *Galois Theory*, NAPCO Graphic arts," Inc., Wisconsin. 1971.

[11] J.B.Fraleigh, *A First course in abstract algebra*, Addison-Wesley Publishing Comp., California, 1982.

[12] R.Lidi and G.Pilz, *Applied abstract algebra*, Spring-Verlag, Inc., N.Y., 1984.

[13] K.Y.Fang and A.S.Wojcik, "Modular Decomposition of combinational Multiple-Valued Circuits," IEEE Trans. Comput., vol.37, no.10, pp.1293-1301, Oct. 1988.

[14] D.Green, *Modern Logic Design*, Addison-Wesley publishing Company, 1988.

저 자 소 개

朴 春 明(正會員)

1955년 12월 4일생 1983년 2월 인하대학교 전자공학과 졸업(공학사) 1986년 2월 인하대학교 대학원 전자공학과 졸업(공학석사) 1994년 2월 인하대학교 대학원 전자공학과 졸업(공학박사) 1990년 3월 - 현재 : 마산전문대학 전자계산기과, 전자제어과 조교수 주관심 분야 : 다치논리이론구성 및 회로설계, 디지탈논리시스템 및 컴퓨터시스템 구조, coding theory, 마이크로프로세서 응용 등임.

金 興 壽(正會員) 第31卷 B編 第1號 參照
현재 인하대학교 전자공학과 교수