論文95-32A-6-12

함수복잡도를 이용한 큐브선택과 이단계 리드뮬러표현의 최소화

(Cube Selection using Function Complexity and Minimization of Two-level Reed-Muller Expressions)

李 貴 相 *

(Gueesang Lee)

요 약

본 논문에서는 함수복잡도를 고려한 큐브선택에 의한 이단계 리드뮬러표현의 최소화 방법을 제시한다. 리드뮬러표현을 최소화하기 위해 두개의 큐브(cube)를 묶는 기존의 Xlinking 방법과는 달리, 큐브선택 방법은 주어진 함수의 ON-set를 커버할 때까지 한번에 하나씩 큐브를 선택해 나가는 방법이다. 이 방법은 대개의 benchmark 함수에 대해 잘 동작하나 격자형 배치를 갖는 패리티 타입 함수에 대해서는 좋지 않은 성능을 보인다. 이를 해결하기 위해 큐브선택의 기준이 되는 비용함수로써 그 큐브에서의 ON-set 의 크기 대신에 함수의 복잡도를 사용한다. 따라서 단지 ON-set 의 크기만을 고려하는 대신 ON-set 을 구성하는 minterm 들이 얼마나 서로 가깝게 되어 하나의 큐브로 묶어질수 있는지 즉, 함수가 얼마나 다음의 최소화에 유리한 형태로 변화되는 지를 중요시하게 되며 이는 특히 격자형함수의 최소화에 잘 동작한다. 실험결과, 격자형함수뿐아니라 일반적인 경우에도 기존의 결과보다 개선된 결과를 보인다.

Abstract

In this paper, an effective method for the minimization of two-level Reed-muller expressions by cube selection which considers functional complexity is presented. In contrast to the previous methods which use Xlinking operations to join two cubes for minimization, the cube selection method tries to select cubes one at a time until they cover the ON-set of the given function. This method works for most benchmark circuits, but for parity-type functions it shows poor performance. To solve this problem, a cost function which computes the functional complexity instead of only the size of ON-set of the function is used. Therefore the optimization is performed considering how the true minterms are grouped together so that they can be realized by only a small number of cubes. In other words, it considers how the function is changed and how the change affects the next optimization step. Experimental results shows better performance in many cases including parity-type functions compared to previous results.

* 正會員, 全南大學校 電算學科

(Dept. of Computer Science, Chonnam Univ.)

接受日字: 1994年7月9日, 수정완료일: 1995年5月30日

I. 서 론

리드뮬러형식(Reed-Muller forms)은 다음과 같은 함수식으로 표현할 수 있다.

$$f\ (x_1, x_2, \ldots, x_n) = \{a_0 \oplus a_1 x_1 \oplus a_2 x_2 \oplus \ldots \oplus a_{2n-1} x_1 x_2 \ldots x_n\}, \quad (1)$$
$$x_i = x_i \ or\ x_i', a_i \in \{0,1\}$$

여기서 x_i 는 하나의 리터럴을 의미하며, x_i 또는 x_i' 로 나타낼 수 있다. 그러므로 리드뮬러 표현식은 임의의 논리함수를 EXOR 연산자를 이용하여 나타낸 것이라 할 수 있다. 여기서 임의의 리터럴이 항상 보수화되지 않은 상태로 나타나면 양극성, 보수화되어 나타나면 음극성이라 한다. 이처럼 리터럴이 갖는 극성에 따라 리드뮬러표현식은 다음과 같이 구분될 수 있다.[4]

식(1)의 모든 리터럴이 양극성만으로 이루어지면 양극성을 갖는 리드뮬러형식(PRM: Positive- polarity Reed-Muller forms), 모든 리터럴이 음극성만으로 이루어지면 음극성을 갖는 리드뮬러 형식(NRM : Negative-polarity Reed-Muller forms), 모든 리터럴이 두가지 극성중 하나만으로 이루어지면 고정극성을 갖는 리드뮬러 형식(FRM: Fixed- polarity Reed-Muller forms), 그리고 각각의 리터럴이 동시에 양극성과 음극성을 가지면 혼합극성을 갖는 리드뮬러형식(MRM : Mixed-polarity Reed- Muller forms)이라 한다[4]. 이 때, 고정극성을 갖는 리드뮬러형식은 각각의 리터럴이 임의의 극성을 가지므로 일반화된 리드뮬러형식(GRM : Generalized Reed Muller forms) 라고도 한다. 일반적으로 혼합극성을 갖는 리드뮬러형식(MRM)이 최소의 면적으로 회로 구현이 가능하므로 더 효율적인 것으로 평가되고 있다[4]. 따라서, 본 논문에서는 MRM형식으로 표현된 리드뮬러회로를 고려하기로 한다.

최근 들어, 이러한 리드뮬러형식을 이용하여 논리함수를 설계하고 구현하는데 많은 관심이 기울여지고 있다[1,2,3,4]. 예를 들어 최근까지의 PLA 구성방법은 ESPRESSO 와 같은 도구를 사용하여 SOP 형태로 최소화하는 과정을 거치는 것이 일반적이었다[6]. 그러나, 격자형 함수(parity-type functions)의 경우에 SOP형태의 구현은 사실상 불가능하며 만일 XOR 게이트들을 사용하면 매우 간단하게 최소화되는 것은 이미 알려진 사실이다. 또한 일반적인 회로의 합성에서도 XOR 게이트들을 사용할 때 SOP 형태로 최소화하는 것과 비교하여 비교될 수 있는 결과들이 제시되고 있다[1,2,3,4]. XOR 게이트를 사용하는 회로의 합성방법의 유일한 문제점은 XOR 게이트가 OR 게이트에 비하여 그 구현비용이 많다는 점이었으나 최근들어 동등한 비용에 의한 구현이 가능해졌다[3].

뿐만 아니라, XOP 형태(Xor of Products form)로 합성한 회로는 그 결함검출력(testability) 에 있어서 SOP 형태의 회로에 비하여 월등함이 연구결과 발표되고 있다[3,9,10]. 따라서 임의의 논리함수를 XOP 형태(Xor of Products form)로 합성하는 것은 매우 매력적인 합성방법의 하나로써 집중적인 연구의 대상이 되고 있다[1,2,3,4].

SOP형식의 최소화 방법으로는 카르노맵을 이용하는 방법이나 퀸-맥클러스키 방법이 많이 이용되었다[11]. 이 방법들을 리드뮬러형식에 대한 최소화 방법으로 확장하여 사용하는 것은 쉽지 않다. 그러나, 부울대수의 XOR성질에 의하여 함수의 OFF-set을 짝수번 커버하면($A \oplus A = 0$) 원래의 함수와 같아진다는 성질을 이용하여 함수의 ON-set을 실현하는 큐브의 집합을 만들 때, 인접된 OFF-set까지 확장하여 큐브를 만들 수 있다. 최근의 리드뮬러형식 최소화 방법에 관한 연구중에서 Perkowski[11]와 Saul[2]의 연구가 대표적이라 할 수 있다. Perkowski[11]가 제안한 Xlinking 방법은 두개의 큐브(cube)를 묶는 연산 과정이 자주 반복됨으로 복잡할 뿐만 아니라 처음에 어떤 큐브를 선택하는가에 따라서 결과가 달라지는 문제점을 내포하고 있다. Saul[2]은 Xlinking 방법의 큐브선택의 문제점을 개선하기 위해 Unlink방법을 제안하였는데, Unlink방법은 큐브의 수를 늘어나게 하는 민텀(minterms)들을 원상태로 복귀시키는 방법이다. 이러한 방법들은 처리해야 할 항(terms)들의 갯수가 증가함에 따라서 그 탐색범위(search space) 가 기하급수적으로 증가하는 단점으로 인하여 소규모 회로에서는 효과적이나 그렇지 않은 경우에서는 적절한 해를 찾기 어려운 단점을 갖고 있다.

그러나, 최근에 제안된 큐브선택에 의한 방법[11]은 이러한 문제를 해결하는 매우 간단한 방법으로써 좋은 결과를 보여주었다. 이 방법은 기존의 Xlinking 방법과는 달리 한번에 하나씩 큐브를 선택하며 이러한 과정을 함수의 ON-set 이 전부 커버될 때까지 반복한다. 이 때, 큐브의 선택은 가장 빨리 ON-set 이 커버되는 방향으로 진행되며, 이를 위하여 효과적인 휴리스틱을 적용한다. 그러나 이 방법은 격자형으로 배치된 함수에 대해서는 ON-term들이 분산되어 있어 좋은 결과를 내지 못하는 단점을 안고 있다. 이를 해결하기 위해 큐브선택의 기준이 되는 비용함수(또는 이득함수)로써 그 큐브에서의 ON-set 의 크기 대신에 함수의 복잡도를 사용한다. 함수의 복잡도란 각각의 minterm들이 이웃하는(Hamming distance 가 1 인) minterm들과 같은 값을 갖는지를 수치로 표현한 것이다. 따라서 함수복잡도가 크면 클수록 그 함수를 더 작은 수의 큐브로 표현할 수 있다. 다시 말하면, 단지 ON-set 의 크기만을 고려하는 대신 ON-set 을 구성

하는 minterm 들이 얼마나 서로 가깝게 되어 하나의 큐브로 묶어질수 있는지 즉, 함수가 얼마나 다음의 최소화에 유리한 형태로 변화되는 지를 중요시하게 되며 이는 특히 격자형함수의 최소화에 잘 동작한다.

2장에서는 리드뮬러회로의 최소화 접근 방법인 Cube selection 방법에 대해 간단히 요약, 기술한다[11]. 3장에서는 함수복잡도의 정의와 간단한 예제회로들을 통해서 본 논문에 제시한 방법에 대해 설명하고, 4장에서는 benchmark회로를 이용한 실험 결과를 제시하며, 마지막으로 결론을 제시한다.

Ⅱ. CUBE-SELECTION 방법[11]

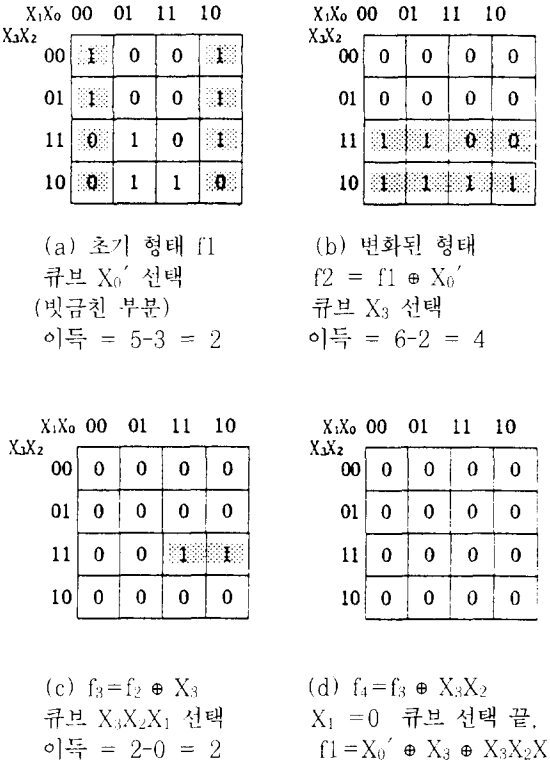
본 논문에서는 곱항의 리터럴이 양극성과 음극성을 동시에 갖는 혼합극성 리드뮬러형식으로 구성된 논리함수를 고려한다. 여기서는 기존의 합성방법인 Cube Selection 의 방법을 간단히 설명한다. 자세한 내용은 [11]을 참조하기 바란다.

논리함수가 주어지면, 함수를 카르노맵(Karnaugh map)으로 표현한다. 논리함수의 크기, 즉 논리함수의 ON-set의 크기는 논리함수의 ON-set 민텀의 갯수이다. 논리함수 f에 관한 큐브 C의 이득은 함수 f를 커버하기 위해 선택된 큐브의 ON-set의 수에서 OFF-set의 수를 감산한 수이다. 이것을 식으로 표현하면 다음과 같다.

$$\text{이득}(C) = \text{크기}(f) - \text{크기}(f \oplus C)$$

이 방법은 함수의 ON-set을 커버할 때까지 한번에 하나의 큐브를 선택해 가는 방법이다. 함수의 ON-set을 커버하는 큐브는 반드시 존재한다. 큐브가 선택되면 남아있는 민텀은 다른 ON-set을 구성한다. 그러므로, 남아있는 민텀에 의해 형성된 ON-set은 새로운 큐브에 따라 계속적으로 변한다. Cube selection 방법에 의한 최소화 알고리즘의 주된 방법은 남아있는 항의 수가 가장 빨리 감소하도록 적절하게 큐브를 선택하는 것이다. 그림 1.에 큐브선택과 함수의 변화과정의 예가 주어져 있다.

여기서 적절한 큐브를 선택하기 위한 간단한 휴리스틱이 제시되었다. 첫번째 휴리스틱은 시도되지 않는 가장 큰 큐브부터 가장 작은 큐브까지 차례대로 시도하는 간단한 방법이다. 똑같은 크기를 가진 큐브는 임의로 선택한다. 만일 선택한 큐브의 이득이 양수이면 (항의 수가 줄어들면) 이를 선택하게 된다. 이 큐브 선택 전략은 큰 큐브가 작은 큐브보다 좀 더 좋은 결과를 얻을 가능성이 있다는 휴리스틱에 기초를 두고 있다. 두번째의 경우 우선 가장 이득이 큰 큐브를 선택한다. 두번째 방법이 첫번째 방법에 비교해서 비교적 더 나은 성능을 보였다[11]. 그러나 이 방법들은 격자형 배치를 갖는 회로에 대해서는 결과가 좋지 못했다. 그것은 극단적으로 parity 함수에 대해서는 어떤 큐브에 대해서도 이득이 0 이기 때문이다. 그림 2(a) 를 보면 이 함수에서 큐브 X_3' 를 선택한 경우를 보이고 있다.



(a) 초기 형태 f1
큐브 X_0' 선택
(빗금친 부분)
이득 = 5-3 = 2

(b) 변화된 형태
$f2 = f1 \oplus X_0'$
큐브 X_3 선택
이득 = 6-2 = 4

(c) $f_3 = f_2 \oplus X_3$
큐브 $X_3X_2X_1$ 선택
이득 = 2-0 = 2

(d) $f_4 = f_3 \oplus X_3X_2$
$X_1 = 0$ 큐브 선택 끝,
$f1 = X_0' \oplus X_3 \oplus X_3X_2X_1$

그림 1. 큐브의 선택의 예
Fig. 1. An example of cube selection.

이 때, 값이 1 인 minterm 과 0 인 minterm 들의 갯수가 같으므로 이득이 0 임을 알 수 있다. 임의의 어떤 큐브에 대해서도 모두 이득이 0 임을 확인해 보라. 이러한 문제점을 해결하기 위하여 spectral 기법이 사용되었다[14]. 이 방법은 본 논문의 주제가 아니므로 자세한 설명은 생략한다. spectral 기법은 선형부분이 회로에 추가됨으로써 회로구현이 더욱 복잡하게 되는 단점을 갖고 있다. 본 논문에서는 선형부분의 회로가 추가되지 않고서도 이 문제를 해결할 뿐 아니라 격자형 함수가 아닌 경우에도 성능의 향상을 가져올 수 있는 함수복잡도에 근거한 비용함수를 제안하고자 한다.

Ⅲ. 함수복잡도(Function Complexity) 의 사용

주어진 논리함수를 표현하는 데 있어서 그 함수의 값뿐만 아니라 그 함수가 갖고 있는 정보를 같이 표현하고자 하는 노력들이 있어왔다. 그 대표적인 경우로 Spectral technique[13] 을 들수 있는데 이는 n 입력함수의 경우 2n 개의 숫자로써 함수의 값과 그 함수의 형태, 예를 들면 ON-set 의 크기나 함수가 어떤 큐브와 얼마나 근사한 형태를 갖는가 하는 것등을 표현한다. 이 방법은 실제 회로의 설계에 많이 이용되고 있으며[13] 본 논문에 제시된 문제를 해결하기 위하여 격자형함수의 최소화에 spectral 기법을 이용하기도 하였다.[14]

이와 같은 맥락에서, 함수의 특성을 하나의 숫자로 표현할 수 있는 방법이 연구되었으며, 이를 일반적으로 그 함수의 함수복잡도라 부른다. 대체로 사용되는 함수복잡도의 정의는 다음과 같다[13].

(정의) 함수복잡도 C(f)

어떤 n 입력 논리함수 f 에 대하여 그 함수를 구성하는 각각의 minterm 에 대하여 그 minterm 과 Hamming distance 가 1 인 minterm 이 같은 값을 갖는 경우의 수를 그 함수의 함수복잡도라 한다.

위 정의를 수식으로 표현하면 다음과 같다.

$$C(f) = \sum_{i=0}^{2^n} \sum_{j=1}^{n} g(m_i, p_{ij}) \qquad (2)$$

여기서 $m_i = i$번째 minterm, $p_{ij} = m_i$ 와 j번째 변수에서 Hamming distance 1 인 minterm 이며, 함수 g 는 두개의 minterm들이 같은 값을 가지면 1, 그렇지 않으면 0 이다. 예를 들어, $m_7 = X_3'X_2X_1X_0 =$ [0111] 이라면 $p_{71} = X_3'X_2X_1X_0' =$ [0110] 이며 $p_{74} = X_3X_2X_1X_0 =$ [1111] 이다.

다음의 두가지 함수의 함수복잡도를 보자. 첫번째는 격자형함수이고 두번째 예는 하나의 큐브로 묶어질 수 있는 함수이다.

두가지 함수의 함수복잡도를 계산하여 보면, 첫번째 함수는 모든 minterm 들이 인접한 (Hamming Distance = 1) minterm 들과 다른 값을 가지므로 함수복잡도는 0 이고, 두번째의 경우는 각각의 minterm 들이 모두 3 방향으로 인접해 있으므로 함수복잡도는 3 x 16 = 48 이 된다. 이 예에서 알 수 있듯이 함수복잡도가 증가할수록 작은 수 의 큐브로 함수를 구현하기가 쉬어 짐을 알 수 있다. 그러나 위의 함수복잡도가 모든 경우에 적용할 수 있는 절대적인 척도는 아니다. 예를 들면 [12] 에서는 위의 함수복잡도를 변화시켜 다음과 같은 형태를 사용하였다.

$$C'(f) = \sum_{i=0}^{2^n} \sum_{j=0}^{n} g'(m_i, p_{ij})$$

$$Size(ON-set(f)) + \sum_{i=0}^{2^n} \sum_{j=1}^{n} g'(m_i, p_{ij})$$

X_3X_2 \ X_1X_0	00	01	11	10
00	0	1	0	1
01	1	0	1	0
11	0	1	0	1
10	1	0	1	0

(a) 격자형 함수 $C(f_1) = 0$

X_3X_2 \ X_1X_0	00	01	11	10
00	0	1	1	0
01	0	1	1	0
11	0	1	1	0
10	0	1	1	0

(b) Group 으로 형성된 함수 $C(f_2) = 3 \times 16 = 48$

그림 2. 두가지 함수의 함수복잡도 C(f) 의 비교
Fig. 2. Comparison of two function complexities.

여기서 함수 g′ 는 두개의 minterm들이 모두 1 의 값을 가지면 1, 그렇지 않으면 0 이다. 따라서 0 값을 갖는 minterm 들은 고려하지 않는다. 또한 식 (2) 과 비교해서 j 값은 0 부터 시작함을 알 수 있다. 즉, 자기자신이 1 인 상태, 쉽게 말하면 함수의 ON-set 의 크기를 고려함을 알 수 있다.

본 논문에서는 다음과 같은 또 다른 형태의 함수복잡도를 사용한다.

$$D(f) = W*Size(OFF-set(f)) + \sum_{i=0}^{2^n} h(m_i, p_{ij}) \qquad (3)$$

여기서 W 는 경우에 따라 달라질 수 있는 임의의 상수이며, 함수 h 는 두개의 minterm들이 모두 0 의 값을 가지면 1, 그렇지 않으면 0 이다. 첫번째 항은 함수의 ON-set 에 어떤 상수 weight 값 W 를 곱한 것이다. 또한 식 (2) 에서 사용한 함수 g 대신 h 를 사용하였다. 이는 큐브선택에 의한 최소화의 방법이 일련의 큐브들을 추출해 냄으로써 값이 1 인 minterm 들을 없애나가면서 궁극적으로 함수를 0 로 만들어 나가기 때문에 값이 0 인 minterm 들로부터 함수복잡도를 취한 것이다. 다음의 예를 보면 식(2) 의 함수복잡도보다는 식(3) 의 함수복잡도가 큐브선택방법에서 적합함을 알 수 있다.

일반적으로 함수복잡도가 증가하면 함수의 형태가 최소화에 더욱 바람직한 방향으로 변화함을 알 수 있다. 그림 3. 에서 함수복잡도로써 식 (2) 에 주어진 C(f)를 구하면 그 값이 38 에서 52 로 증가함으로써

더 단순한 형태로 변했음을 알 수 있다. 최적화된 형태는 $f_1 = X_3 \oplus X_3X_2X_1$, $f_2 = 1 \oplus X_3X_2X_1$이므로 f_2 가 더 단순한 함수임을 알 수 있다. 그러나 그림 3.(b) 와 같이 큐브를 선택한다면 결국 $f_1 = f_2 \oplus X_3' = 1 \oplus X_3X_2X_1 \oplus X_3'$ 이므로 더 많은 갯수의 항으로 구성되므로 바람직하지 못한 선택이었음을 알 수 있다. 다시 말하면, X_3' 보다는 X_3를 선택했었어야 하므로 위의 큐브선택은 잘못된 것이라 할 수 있다. 반면 본논문에서 사용하는 함수복잡도(식 3) D(f)를 이용하면 그 값이 감소하므로 그림 3(b) 형태로의 변화는 일어나지 않는다.

X_3X_2 \ X_1X_0	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	1	1	0	0
10	1	1	1	1

(a) 초기 형태 f_1
$C(f_1) = 38$
$D(f_1) = 37(W=1)$

X_3X_2 \ X_1X_0	00	01	11	10
00	1	1	1	1
01	1	1	1	1
11	1	1	0	0
10	1	1	1	1

(b) 변화된 형태 $f_2 = f_1 \oplus X_3'$
$C(f_2) = 52$
$D(f_2) = 4(W=1)$

그림 3. 큐브의 선택과 함수복잡도의 변화
Fig. 3. Cube selection and the change of function complexity.

X_3X_2 \ X_1X_0	00	01	11	10
00	0	1	0	1
01	1	0	1	0
11	0	1	0	1
10	1	0	1	0

(a) 초기 형태 f_1

X_3X_2 \ X_1X_0	00	01	11	10
00	1	1	0	0
01	0	0	1	1
11	1	1	0	0
10	0	0	1	1

(b) $f_2 = f_1 \oplus X_0'$
$D(f_2) - D(f_1) = 8(W=1)$

X_3X_2 \ X_1X_0	00	01	11	10
00	0	0	0	0
01	1	1	1	1
11	0	0	0	0
10	1	1	1	1

(c) $f_3 = f_2 \oplus X_1'$
$D(f_3)-D(f_2)=8(W=1)$

X_3X_2 \ X_1X_0	00	01	11	10
00	1	1	1	1
01	1	1	1	1
11	0	0	0	0
10	0	0	0	0

(d) $f_4 = f_3 \oplus X_2'$
$D(f_4)-D(f_3)=8(W=1)$

X_3X_2 \ X_1X_0	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	0	0	0	0
10	0	0	0	0

$f_1 = X_0' \oplus X_1' \oplus X_2' \oplus X_3' =$
$[---0] \oplus [--0-] \oplus [-0--] \oplus [0---]$

(e) $f_5 = f_4 \oplus X_3' = 0$ (f) 선택된 큐브와 최종결과
$D(f_5) - D(f_4) = 48(W=1)$

그림 4. parity 함수의 함수복잡도에 의한 큐브선택과 함수 변화
Fig. 4. Change of the parity function by cube selection using function complexity.

이 예제에서 보듯이 식 (2) 의 함수복잡도는 본 논문에서 취하는 큐브선택방법에는 적합치 않으며 식(3) 의 함수복잡도가 적절함을 알 수 있다. 그림4는 parity 함수에 대한 함수복잡도에 의한 함수 선택의 결과로써 각각의 큐브선택의 단계마다의 함수의 변화를 보여준다.

표 1. 실험결과
Table 1. Experimental results.

Benchmark회로			SAUL [2,4]		Cube Selection			함수복잡도에 의한 방법		
종류	PI	PO	term	literal	term	literal	time (sec)	term	literal	time (sec)
rd53	4	1	15	66	23	106	0.0	16	39	0.1
rd73	7	3	68	237	96	630	1.0	51	223	7.7
rd84	8	3	101	411	191	1457	6.5	95	603	41.0
5xp1	7	10	61	223	42	146	4.3	35	126	10.9
9sym	9	1	94	507	71	596	3.6	59	451	54.5
con1	7	2	7	39	14	46	0.4	11	37	2.4
clip	8	7	123	812	119	720	71.0	102	661	258.
misex 1	5	3	35	99	29	117	9.4	18	68	29.0
sao2	10	4	103	752	55	480	48.4	44	356	170.
f51m	8	8	36	232	42	157	11.5	35	133	39.0
bw	5	28	92	404	61	202	1.4	49	184	1.8

Ⅳ. 실 험 결 과

본 논문에서 제안한 함수복잡도를 이용한 큐브선택방법은 IBM PC UNIX SYSTEM 에서 C 언어로 작성되었으며, Saul 의 결과, 기존의 큐브선택방법 과 비교하였다. 나열된 benchmark 회로들중 rdxx 는 parity 함수를 포함하고 있으며 9sym 은 함수내부의

각 부분에 대칭성을 갖고 있어 Miller 가 제안한 함수 복잡도[13]는 적용했을 때 가장 나쁜 결과를 보여주었다. 기존의 큐브선택방법과 새로 제안한 함수복잡도에 의한 큐브선택방법 각각에 대하여 II 절에 제시한 두번째 휴리스틱에 의한 결과를 제시하였다. 실행시간은 함수복잡도의 계산시간이 단순한 ON-set 의 크기를 계산하는 것보다 많아지기 때문에 기존의 큐브선택방법보다 더 길어지나, 이는 앞으로 충분히 개선될 수 있을 것으로 보인다[15].

V. 결 론

본 논문에서는 일반화된 리드뮬러회로를 최소화하기 위해 Cube selection 방법을 위한 함수복잡도에 근거한 새로운 비용함수(이득함수)를 제시하였으며 실험결과를 제시하였다. 이 방법은 기존의 결과에 비교하여 크게 개선된 것으로 나타났다.

Cube selection 방법은 남아 있는 민텀의 수가 매우 빠르게 감소되도록 적절한 일련의 큐브들을 하나씩 찾아가는 방법이다. 이 방법은 최적화문제의 전형적인 해결방법인 Hill Climbing 의 기법을 채택하고 있으며 Hill 의 경사도의 기준치로써 비용함수를 사용하고 있다. 따라서 비용함수의 선택은 이 방법의 성능에 절대적인 영향을 가지며 더 나은 비용함수의 선택은 앞으로도 더 연구되어야 할 부분이다.

그러나, 큐브선택방법은 해결해야 할 문제들이 남아 있다. 그 문제들은 두가지로 지적될 수 있는데, 기억공간 과 실행시간이다. 입력이 많은 함수에 대해서는 그 함수의 표현이 현재와 같이 진리표형태를 취한다면 그 기억용량이 입력수에 대해 지수적으로 증가하기 때문에 간결한 함수 표현이 요구된다. 이 문제에 대해서는 BDD 나 Reduced Form[13] 을 사용하면 해결 될 수 있을 것으로 보인다.(함수 traversal 시간이 짧아짐) 둘째로, 실행시간에 있어서 다음에 선택할 큐브들을 찾아 나가는 데 필요한 탐색대상이 너무 많다는 점을 지적할 수 있다. 이에 대해서는 exhaustive 한 방법이 아닌 함수의 ON-set 으로부터 다음의 큐브를 선택할 수 있는 방법들이 연구되고 있다.

본 논문을 통해서 대답하고자 하는 물음은 "큐브선택방법은 모든 논리함수의 이단계 리드뮬러 최적화에 적합한 방법인가? 즉, 격자형 함수를 포함한 모든 함수에서 기존의 결과와 비교할 때 만족할 만한 결과를 생성하는가?" 이다. 위에 제시한 단점들에도 불구하고 본 논문의 함수복잡도를 이용한 방법은 이 물음에 해답을 준다는 점에 의의가 있다. 앞으로의 연구는 위에 제시한 문제점들을 해결할 수 있어야 할 것이다[15].

* 본 연구는 '95년도 한국과학재단 연구비지원(과제번호 951-0917-119-1)에 의한 결과임.

참 고 문 헌

[1] M. Helliwell and M. Perkowski, "A Fast Algorithm to Minimized Multi-output Mixed-Polarity Generalized Reed-Muller Forms," Design Automation Conference, pp.427-432, 1988.

[2] J. Saul, "An Improved Algorithm for the Minimization of Mixed Polarity Reed_Muller Representation," ICCD, pp.372-375, 1990.

[3] A. Sarabi and M. Perkowski, "Fast Exact and Quasi-Minimal Minimization of Highly Testable Fixe-Polarity AND/XOR Canonical Networks," Design Automation Conference, pp.30-35, 1992.

[4] J. Saul, " An Algorithm for the Multi-level Minimization of Reed-Muller Representations," ICCD 1991.

[5] T.Sasao and P. besslich, " On the complexity of Mod-2 Sum PLA's," IEEE transaction on Computers, Vol.39 NO.2, pp. 262-265, Feb. 1990.

[6] R.k Brayton et al., "Logic Minimization Algorithms for VLSI synthesis," Kluwer Academic Publishers.

[7] S.Even,IKohavi, A.Paz, "On minimal module-2 sums of products for switching functions," IEEE transaction on Electronic Computers, Vol. EC-16 , pp. 671-674, Oct. 1967.

[8] 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 Jan. 1982.

[9] S.M. Reddy, "Easily Testable Realization for Logic Functions", IEEE trans. on Computers, vol C-21, No. 11, pp. 1183-1188, 1972.

[10] D.K.Pradhan, "Universal Test Sets for Multiple Fault Detection in AND_EXOR

Arrays", IEEE trans. on Computers, Vol. C-27, No. 2, PP. 181-187, 1978.

[11] 장준영, 이귀상, "리드뮬러회로 합성을 위한 새로운 방법", 전자공학회 논문지 ,제30권 B편, 제9호, pp.813-820, 1993년 9월

[12] Devadas Verma, E. A. Trachtenberg, "Design Automation Tools for Efficient Implementation of Logic Functions by Decomposition", IEEE Transation on Computer-Aided Design Vol. 8, No. 8, August 1989.

[13] S.L.Hurst et.al, "Spectral Techniques in Digital Logic," Academic Press, 1985, pp. 146-147.

[14] 이 화, "큐브선택을 이용한 이단계 리드뮬러 합성도구의 설계 및 구현," 석사 학위논문, 전남대학교, 1994년 2 월

[15] 이귀상, 천승환, "BDD 를 이용한 이단계 리드뮬러 회로의 합성," 대한전기학회 논문지 1995, 5월호

저 자 소 개

李 貴 相(正會員)

1980년 서울대학교 전기공학과 졸업(B.S). 1982년 서울대학교 전산기공학과 졸업(M.S.). 19 91년 Pennsylvania State University 전산학과(Ph.D.). 현 전남대학교 전산학과 조교수.