논리식 인수분해를 위한 코스웨어

권오형†

요 약

일반적으로 논리식은 수많은 인수분해식으로 표현이 가능하다. 논리식에 대한 보다 간략화된 인수분해식을 찾는 것이 논리합성의 기본 기능 중의 하나이며 본 논문에서 논리회로 수업의 교육용 도구로 부울 인수분해식을 산출하는 새로운 방법을 제안한다. 제안하는 방법은 서포트와 함께 2개의 항에 대한 나눗셈을 수행하는 것이다. 인수분해식의 리터럴 개수는 논리식의 간략화 정도를 판단하는 기준이 되는데, 제안하는 방법으로 실험한 결과, 기존의 타 방법들 보다 리터럴 개수를 줄이는 효과를 보였다.

주제어 : 논리합성, 부울 대수, 인수분해

Courseware for Factorization of Logic Expressions

Oh-Hyeong Kwon†

ABSTRACT

Generally, a logic function has many factored forms. The problem of finding more compact factored form is one of the basic operations in logic synthesis. In this paper, we present a new method for factoring Boolean functions to assist in educational logic designs. Our method for factorization is to implement two-cube Boolean division with supports of an expression. The number of literals in a factored form is a good estimate of the complexity of a logic function. Our empirical evaluation shows the improvements in literal counts over previous other factorization methods.

Keywords : Logic Synthesis, Boolean algebra, Factorization

† 정 회 원: 한서대학교 부교수(주, 교신저자)
논문접수: 2011년 10월 14일, 심사완료: 2011년 11월 09일, 게재확정: 2012년 01월 10일

1. 서론

논리회로 수업은 컴퓨터공학이나 전자공학 전공자를 위한 기초 과목으로 수업은 이론과 논리게이트들을 연결하여 회로를 만들어 보는 실습으로 교과 과정을 운영한다. 대학 수업에서 논리식 최적화는 주로 카르노맵(Karnaugh map)을 이용하고 있다. 그러나, 학교에서 보다 심도 있는 논리회로에 대한 수업으로 향후 디지털 회로 분야의 전문가 양성을 위해서는 최적화 알고리즘 측면에서 논리식을 간략화하기 위한 방법을 학습하는 것이 필요하다. 본 논문에서는 논리회로 분야의 코스웨어에 삽입될 한 가지 모듈로써 새로운 논리식 인수분해 방법을 제시한다.

논리식 인수분해에 대한 연구로는 Lawler가 1964년에 전수 탐색에 의한 최적의 결과를 산출할 수 있는 알고리즘[1]을 발표하였다. 이 방법은 전수 탐색으로 장시간의 수행 시간이 필요하기 때문에 아주 작은 회로 최적화 정도에나 사용된다. 따라서, 인수분해에는 전수탐색 방법보다는 선험 방법이 이용된다. 인수분해식 산출의 선험 방법은 크게 대수 인수분해 방법과 부울 인수분해 방법으로 구분한다. 대수 인수분해 방법은 논리식을 대수 다항식으로 간주하고 인수분해를 하여 논리식을 간략화하는 방법이다. Brayton 등[2][3][4][5]은 코커널/커널을 이용하여 대수 인수분해식을 산출하는 커널 기반 방법을 제안하였다. 이 인수분해 방법은 인수분해식 산출 속도를 빠르게 향상시켰으나, 때때로 최소 개수의 리터럴을 갖는 인수분해식을 산출할 수 없는 단점을 갖는다. 반면에, 부울 인수분해 방법은 대수 인수분해 방법과 비교하면 보다 적은 개수의 리터럴을 갖는 인수분해식을 산출할 수 있는 장점을 갖는다. 최근의 부울 인수분해식 산출 방법들을 정리하면 다음과 같다. Liao 등[6]은 다치 함수와 분지 및 한계 커버 방법을 이용한 부울 인수분해 방법을 제안하였다. Stanion 등[7]은 Binary Decision Diagram(BDD)를 이용한 부울 나눗셈과 부울 인수분해 방법을 제안하였다. Kwon 등[8]은 코커널 큐브 행렬을 확장하여 부울 인수분해식 산출 방법을 제시하였다. 이 방법은 대수 인수분해식을 산출하기 위한 SIS에서 사용하는 행렬을 포함하는 수퍼행렬을 만들어 부울 인수분해식을 산출하도록 고안하였다. Yang 등[9]은 BDD로 논리함수를 표현하고, 다시 BDD를 나누는 BDD 분리 엔진을 제안하였으며, 이 엔진으로부터 인수분해를 위한 트리(tree) 만드는 방법을 제시하였다. Modi 등[10]은 2개의 리터럴만을 갖는 제수를 구해 논리식을 인수분해하는 방법을 제안하였다. Mintz 등[11]은 그래프 나누기 방법을 이용해서 인수분해식을 산출하는 방법을 제안하였다. 또, 최근에 Kwon 등[12]은 2-큐브 비커널(non-kernel)을 이용한 부울 인수분해식 산출 방법을 발표하였다. 지금까지 소개한 부울 인수분해식 산출 방법들은 논리식의 항들에서 서포트(support)는 같으나 리터럴이 서로 다를 경우 마치 다른 변수와 같이 취급을 하기 때문에 보다 간결한 부울 인수분해식을 산출하지 못하는 단점을 갖고 있다. 예로 논리식 $F=ab'c+a'bde'+b'cde$에 대하여 리터럴을 구분하여 인수분해 했을 경우와 구분하지 않을 경우 차이점을 관찰하면 문제점이 분명해 진다. 첫째로, 논리식에서 리터럴을 구분했을 경우, 즉 b와 b' 또 e와 e'을 구분하여 인수분해를 하면 10개의 리터럴을 갖는 $F=d(a'be'+b'ce)+ab'c$로 표현된다. 둘째로, 리터럴을 구분하지 않으면 먼저 $F=(a+db+de)(b'c+a'e')$로 인수분해가 되고, 다시 $db+de$를 인수분해 하면 최종적으로 8개 리터럴의 $F=(a+d(b+e))(b'c+a'e')$가 산출된다. 이러한 관찰 결과에 따라 본 논문에서는 서포트를 추가하여 공통식 또는 제수를 찾아 인수분해를 하는 인수분해식 산출 방법을 제시한다.

2. 서포트를 이용한 인수분해

간략화된 논리식을 산출하기 위한 수단 중의 하나가 논리식을 인수분해하는 것이다. 인수분해에 의해 다단 구조(multi-level)의 논리식이 산출된다. 다단 구조가 2단 구조의 논리식(Sum-of-products)보다 간략화되는 것으로 알려져 있다. 특히, 다단 논리식의 간략화 정도는 논리식에 사용된 리터럴 개수로 판단을 하게 되는데, 이유는 논리식을 바로 MOS 회로로 구현할 경우 리터럴 개수에 비례해서 트랜지스터의 수가

증가하기 때문이다. 본 절에서는 논리식 인수분해를 위한 새로운 방법을 제안하며 논문 기술에 필요한 기본 용어들을 정의와 제안한 인수분해 방법에 대하여 소개한다.

정의 1: 변수(variable)는 부울 공간(Boolean space)에서 한 좌표를 나타내는 문자다. 리터럴(literal)은 변수 그 자체 또는 그의 보수(complement)다. 큐브(cube)는 리터럴들의 집합으로 만일 리터럴a가 존재하면, 그의 보수 리터럴 a'을 포함하지 않는다. 단순식(expression 또는 sum-of-products(SOP) form)은 큐브들의 집합이다.

예 1: 문자a는 변수다. a와 a'은 리터럴이다. 리터럴 집합$\{a,b\}$는 큐브, 그러나 $\{a,a'\}$은 큐브가 아니다. $\{\{a,b'\},\{b,c\}\}$는 단순식이다.

본 논문에서는 큐브와 단순식을 표현하는 경우 집합 표기와 보편적으로 사용되는 수식 표기를 모두 사용한다. 따라서 큐브$\{a,b\}$는 ab와 동일한 표현이며, 단순식$\{\{a,b'\},\{b,c\}\}$는 $ab'+bc$와 동일한 표현이다.

정의 2: 서포트(support)는 단순식 F에 사용된 모든 변수들이다. 이 때, 단순식 F의 서포트 변수 집합 $sup(F)$는 다음과 같다.

$$sup(F)=\{x|\text{큐브 } C\in F\text{에 대하여, } x\in C \text{ 또는 } x'\in C\}$$

예 2: 단순식 $a+bc'$의 서포트는 $sup(a+bc')$ = { a, b, c }이다.

정의 3: 단순식을 구성하는 모든 큐브들에 대하여, 공통으로 쓰인 리터럴이 없다면 그 단순식은 큐브면제(cube-free) 되었다고 한다. 단순식이 어떤 큐브로부터 나누어졌을 때 몫이 큐브면제라면, 그 몫을 커널(kernel)이라 한다. 이 때 커널을 산출한 큐브를 코커널(co-kernel)이라 한다.

예 3: 단순식 $ab+c$는 큐브면제, 그러나 $ab+ac$와 abc는 큐브면제가 아니다. 단순식 $F=abfg+a'cdef+cdfg$ 에 대하여, F의 커널과 코커널의 예를 보이기 위해서 주어진 논리식을 다음과 같이 표현하자. 즉, $F=abfg+cdf(a'e+g)$로 표현된 경우, $a'e+g$는 커널이 되며, 이 때 코커널은 cdf이다.

정의 4: 인수분해식(factored form)은 단순식들이 합과 곱으로 표현된 것이다. 구체적인 정의는 다음과 같다.

1) 리터럴은 인수분해식이다.
2) 인수분해식들의 곱은 인수분해식이다.
3) 인수분해식들의 합은 인수분해식이다.

인수분해식은 단순식들이 합과 곱으로 반복해서 표현된 논리식이다.

정의 5: 등멱법칙($a\cdot a=a$)과 보수법칙($a\cdot a'=0$)을 적용할 필요없이 인수분해식을 구성하는 단순식들을 곱할 수 있는 경우 대수 인수분해식(algebraic factored form)이라 한다. 대수 인수분해식 이외의 경우 부울 인수분해식(Boolean factored form)이라 한다.

예 4: $F=bc'd'e+ab'c+ab'e+ac'd'$와$F=a(b'(c+e)+c'd')+bc'd'e$ 모두 대수 분해식이다. 반면에, $F=(a+be)(b'(c+e)+c'd')$는 부울 분해식이다.

2.1 인수분해를 위한 제수

주어진 단순식에서 2개의 큐브를 선택하고 이 2개의 큐브들에서 공통 인수, 즉 공통 큐브를 찾는다. 이 때, 공통 큐브가 제수고 이 제수로 2개의 큐브를 나눈 것이 몫이 된다. C를 제수 집합, Q를 2개의 큐브로 구성된 몫 집합이라 하자. 표기상 제수/몫 쌍을 괄호를 이용하여 표현하고, $c_i\in C$, $q_i\in Q$,라 하자. 그러면, (c_i,q_i)는 대수 나눗셈에 의한 제수/몫 쌍을 표현한 것이다.

또 q_i에서 공통으로 사용된 서포트 x가 있다면 c_ix를 제수로 하고 q_i에서 x를 포함하는 항에서는 x를 제거하고, x'을 포함하는 항은 x'은 그대로 두고 나머지 리터럴을 제거한 항을 만든다. 즉,

항에 보수법칙이 적용되도록 한다.

예 5: $d(a'be'+b'ce)$와 같은 인수분해식이 있다고 하자. 그러면, 제수는 d이고, 2-큐브 몫은 $a'be'+b'ce$가 된다. 2-큐브 몫에서 공통의 서포트 b와 e가 발견되기 때문에 새로운 부분 분해식은 다음과 같이 산출된다.

$$bd(a'e'+b'c)$$
$$bd(a'e'+b'e)$$
$$de(a'e'+b'c)$$
$$de(be'+b'c)$$

다음 대수 나눗셈에 의한 것과 서포트를 이용한 제수와 2-큐브 쌍 2개를 (c_i, q_i), (c_j, q_j)로 표현하자. 그러면, $c_i \in C$, $c_j \in C$, $q_i \in Q$, $q_i \in Q$ 이고 $i \neq j$가 된다. 2-큐브만으로 구성된 부분 부울 인수분해식이 다음과 같은 조건에서 산출된다. 만일 $c_i \in q_j$, $c_j \in q_i$ 이고 $q_i q_j$ 가 주어진 단순식에 포함되면, 2-큐브 쌍 (q_i, q_j)는 부분 부울 인수분해식이 된다. 다음 예는 주어진 단순식으로부터 2개의 항을 선택해서 산출한 2-큐브 대수 인수분해식, 서포트를 이용한 2-큐브 인수분해식, 2-큐브 쌍의 부분 부울 인수분해식을 산출하는 방법을 보인 것이다.

예 6: 논리식 $F = ab'c + a'bde' + b'cde$ 로부터 표 1과 같이 각 큐브에 양의 값을 갖는 인덱스를 부여한다. 다음 2개의 큐브들 사이에서 공통인수를 추출한 대수 인수분해식을 구한다. 표 2는 대수 인수분해식들 이다. 표 2에서는 인수분해식을 제수와 몫을 열(column)로 구분해서 표시한다. 표 2의 제 1열은 설명을 쉽게 하기 위해 부여한 행 번호다. 표 3은 2개의 항들 사이에서 산출된 서포트를 추가한 경우를 보인 것이다. 표 2와 표 3의 인수분해식들로부터 2-큐브 인수분해식 쌍을 찾으면 $(a+de)(a'e'+b'c)$가 된다. 표 2의 행 1과 표 3의 행 5의 인수분해식들을 보면 표 2의 행 1의 제수가 표 3의 행 5의 몫에 포함되고, 다시 표 3의 행 5의 제수가 표 2의 행 1의 몫에 포함된다. 또한 표 2의 행 1과 표 3의 행 5의 몫들의 논리곱(AND)을 구하면 주어진 논리식에 포함되기 때문에 이는 부분 부울 인수분해식 $(a+de)(a'e'+b'c)$이 된다. 이를 정리한 것이 표 4이다.

<표 1> 큐브 인덱스

큐브 C_i	$ab'c$	$a'bde'$	$b'cde$
인덱스 $index(C_i)$	1	2	3

<표 2> 2-큐브 인수분해식

행	선택된 큐브	2-큐브 대수 부분 인수분해		
		분해식	제수	몫
1	$C_1 \cap C_3$	$b'c(a+de)$	$b'c$	$a+de$
2	$C_2 \cap C_3$	$d(a'be'+b'ce)$	d	$a'be'+b'ce$

<표 3> 서포트를 이용한 인수분해식

행	선택된 큐브	서포트 큐브에 의한 인수분해		
		분해식	제수	몫
1	$C_2 \cap C_3$	$bd(a'e'+b'ce)$	bd	$a'e'+b'ce$
2	$C_2 \cap C_3$	$bd(a'e'+b'c)$	bd	$a'e'+b'c$
3	$C_2 \cap C_3$	$bd(a'e'+b'e)$	bd	$a'e'+b'e$
4	$C_2 \cap C_3$	$de(a'be'+b'c)$	de	$a'be'+b'c$
5	$C_2 \cap C_3$	$de(a'e'+b'c)$	de	$a'e'+b'c$
6	$C_2 \cap C_3$	$de(be'+b'c)$	de	$be'+b'c$

<표 4> 부분 부울 인수분해식

선택된 2개 행	인수분해식
표2의 행1과 표3의 행5	$(a+de)(a'e'+b'c)$

2.2 분해식 산출 행렬 M과 부울 분해식

단순식 F가 주어졌을 때, 인수분해식 산출 행렬 M은 2개의 큐브로부터 표 2와 같이 제수와 몫으로 구성된 부분 인수분해식, 표 3과 같은 서포트를 이용한 인수분해식과 표 4와 같은 2-큐브 쌍만으로 구성된 부분 인수분해식을 이용한다. 행렬 M의 행은 제수와 몫과 몫의 쌍에 포함되는 큐브에 대응하여 구성된다. 행렬 M의 열은 몫을 구성하는 각 큐브당 하나의 열이 배당된다. C를 행에 대응되는 큐브들의 집합, CQ를 행렬 M의 열에 해당하는 큐브들의 집합으로 표기하고자 한다.

α_i, β_j를 각각 M의 i번째 행, j번째 열을 나타낸다고 하면 $\alpha_i \in C$, $\beta_j \in CQ$ 이다. 이 때, 행렬 M의 원소 $M(\alpha_i, \beta_j)$는 다음과 같은 값을 갖는다. 다음 식에서 *는 무관(don't care)을 의미한다.

$$M(\alpha_i,\beta_j)=\begin{cases} index(\alpha_i\beta_j) & \text{if } \alpha_i \in C, \\ & \beta_j\text{는}\alpha_i\text{로나눈몫에 속하는큐브} \\ index(c) & \text{if } \alpha_i\text{와}\beta_j\text{는몫과몫쌍에 포함되는} \\ & \text{큐브, } c=\alpha_i\beta_j \neq 0 \\ * & \text{if } \alpha_i\beta_j=0, \\ & \alpha_i\text{와 } \beta_j\text{는 몫과 몫 쌍의큐브} \\ 0 & \text{그 외의 경우} \end{cases}$$

행렬 M을 만든 다음, 행렬에서 프라임 사각형을 찾는다. 이 프라임 사각형을 찾는 알고리즘은 선행 연구로 발표했던 참고자료[8]의 방법을 이용했기 때문에 본 논문에서는 프라임 사각형을 찾는 알고리즘에 대한 소개는 생략하고 사각형에 대한 정의만을 다시 소개한다[8][12][13]. 그리고, 예 7에서 행렬을 만드는 과정과 행렬에서 산출한 프라임 사각형을 보인다.

정의 6: 부울 인수분해식 산출 행렬 M에서 사각형은 행과 열의 부분 집합 (R,C)이며 다음 조건을 갖는다. $\alpha, \gamma \in R$ 및 $\beta, \delta \in C$에 대하여 $M(\alpha,\beta) \neq 0$이며 $\alpha \neq \gamma$이고 $\beta \neq \delta$인 경우 $M(\alpha,\beta) > 0$ 이고 $M(\gamma,\delta) > 0$ 이면 $M(\alpha,\beta) \neq M(\gamma,\delta)$. 프라임 사각형은 다른 사각형에 포함되지 않는 사각형이다. 사각형(R,C)의 사각형 비용은 행 R과 열 C에 포함되는 리터럴들의 개수로 정의한다.

사각형 또는 프라임 사각형에 포함되는 원소들은 서로 다른 양의 정수 값과 다수의 *를 포함할 수 있으며, 또한 사각형에 포함되는 행들이나 열들은 서로 인접할 필요는 없다.

예 7: 예 5의 $F=ab'c+a'bde'+b'cde$에 대한 표 1, 표 2, 표 3과 표 4을 이용해서 인수분해식 산출 행렬M을 만든다. 행렬의 행들은 표 2와 표 3의 제수에 해당하는 큐브와 표 4의 2-큐브 쌍에 포함되는 큐브들에 대응하도록 한다. 행렬의 열은 표 2와 표 3의 몫에 포함되는 큐브들에 대응되도록 한다. 그러면, 행렬의 행은 집합 $\{b'c,\ d,\ bd,\ de,\ a,\ a'e'\}$의 각 원소에 대응된다. 행렬의 열은 집합 $\{a,\ de,\ a'be',\ b'ce,\ a'e',\ b'c,\ b'e,\ a'be',\ be'\}$의 각 원소에 대응된다. 산출된 인수분해식 산출 행렬은 표 5와 같다. 표 5의 행렬에서 첫 번째 행과 첫 번째 열에 해당하는 $M(1,1)$은 표 2의 행 1로부터 산출된 것으로 $M(1,1)=M(b'c,a)$의 원소 값은 $index(ab'c)=1$가 된다. 또한 세 번째 행과 여섯 번째 열에 해당하는 $M(3,6)$은 표 3의 행 1로부터 산출된 것으로 $M(3,6)=M(bd,b'c)$의 원소 값은 $bd \cdot b'c=0$이 되기 때문에 $M(3,6)=*$ 가 된다. 나머지 원소들에 대해서도 같은 방법으로 행렬에 값이 할당된다. 표 5로부터 산출된 프라임 사각형은 회색으로 표시된 부분이다. 이 프라임 사각형으로부터 산출된 인수분해식은 $(a+bd+de)(a'e'+b'c)$가 된다. 다시 $a+bd+de$에 대하여 인수분해식을 산출하면 최종적으로 $F=(a+d(b+e))(a'e'+b'c)$가 되며, 리터럴 수는 8개가 된다.

<표 5> 부울 인수분해식 산출 행렬

행 \ 열	a	de	$a'be'$	$b'ce$	$a'e'$	$b'c$	$b'e$	$a'be'$	be'
$b'c$	1	3							
d			2	3					
bd					2	*	*		
de					*	3		*	*
a					*	1			
$a'e'$	*	*							

2.3 인수분해식 산출 알고리즘

주어진 단순식으로부터 최종 부울 인수분해식을 산출하는 알고리즘은 재귀 호출에 의한 구조로 만들어 진다. 전체적인 알고리즘은 그림 1과 같다. 주어진 단순식 F에 대하여 인수분해식 산출 행렬을 만들고, 프라임 사각형을 찾아 인수분해식을 산출하면 $F=D \cdot Q+R$ 형태의 논리식으로 표현된다. 다시, D, Q와 R에 대하여 반복해서 각각 인수분해식 산출 행렬을 만들어 프라임 사각형을 찾아 인수분해식을 만든다. 그림 1의 알고리즘에서 단순식이 인수분해가 가능한지 먼저 확인한다.

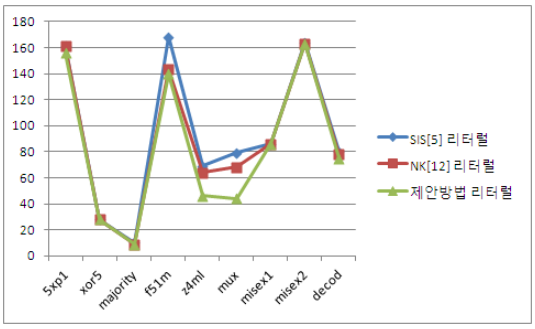
인수분해가 가능하면 행렬 만드는 과정과 인수분해식 산출하는 과정을 반복하고, 인수분해가 불가능 할 경우 종료한다. 또한 나머지는 프라임 사각형으로부터 산출된 D와 Q로부터 $R = F - QD$에 의해 산출한다. 알고리즘의 마지막 부분에서 $R = \phi$인 경우는 F가 D와 Q만으로 분해되는 경우다. $R \neq \phi$인 경우 다시 R로부터 부울 인수분해식을 찾는다.

```
Algorithm 4: 부울 인수분해식 산출.
입력: 주어진 단순식 F.
출력: 부울 인수분해식.
방법:
FACTOR( F )
    if (( F의 인수분해 가능성) == true)
      then write( F);
      else return;
    서포트를 이용한 부분 인수분해식을 이용해서
    부울 인수분해식 산출 행렬 M 산출;
    M에서 프라임 사각형 산출;
    가장 리터럴 수를 줄이는 프라임 사각형 선택;
    선택한 프라임 사각형의 행들을 식 Q로 전환;
    FACTOR( Q );
    선택한 프라임 사각형의 열들을 식 D로 전환;
    FACTOR( D );
    R = F - QD ;
    if ( R≠Φ)       then write("+ ");
                    else FACTOR( R );
```

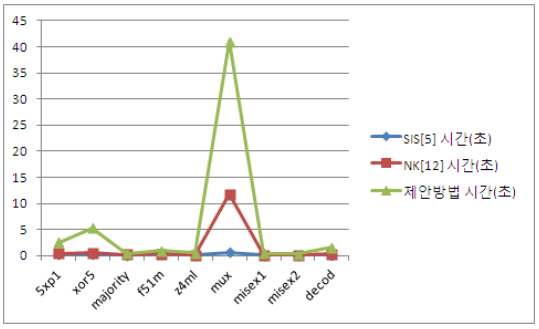
<그림 1> 부울 인수분해식 산출 알고리즘

3. 실험 결과

제안한 방법을 MCNC 벤치마크 회로에 대하여 테스트하였다. 참고로, 본 실험에서 사용한 MCNC 벤치마크 회로에는 추첨회로(voter)인 majority, 대수연산 회로인 f51m, 2-bit 덧셈 회로인 z4ml, 디코더 회로인 decod 등 다양한 논리 회로가 포함되어 있기 때문에 MCNC 벤치마크 회로에 대하여 성능을 비교하는 것은 실제 상황에 적용하는 것과 동등한 효과를 갖게 된다. 제안한 부울 인수분해 방법의 성능을 판단하기 위해서 대수 인수분해 방법과 부울 인수분해를 수행하는 기존의 방법들을 선정하여 결과를 비교하였다. 대수 인수분해 방법은 수행속도가 상대적으로 빠르기 때문에 제안한 방법과 수행시간을 비교하기 위한 대상으로 적합하다. 비교 대상이 될 대수 인수분해 방법에는 가장 널리 사용되는 SIS[5]를 선정하였다. 또한 다른 부울 인수분해 방법과 비교함으로써 제안한 방법이 얼마나 리터럴 개수를 줄일 수 있는지 판단하기 위해 제안한 방법과 유사한 방법으로 부울 인수분해식을 산출하며 또한 가장 최근에 발표된 비커널에 의한 부울 인수분해 산출 방법[12]을 선정하여 결과를 비교하였다. 그림 2는 제안한 방법과 타 방법들을 리터럴 개수를 대상으로 비교한 결과이며, 그림 3은 각 회로를 최적화하는데 소요되는 시간을 초단위로 산출한 결과를 비교한 것이다. 실험은 Pentinum IV 1.4GHz CPU로 Linux 환경에서 진행하였다. 그림 2와 3의 x 축에는 각각의 벤치마크 회로를 표시하였다. 그림 2에서 보이는 바와 같이 제안한 방법에 의해 산출한 회로의 리터럴을 나타내는 그래프는 타 방법으로 산출한 그래프보다 아래쪽에 위치하고 있어 대부분의 벤치마크 회로에서 리터럴 개수를 줄일 수 있음을 보이고 있다. 특히, mux 회로의 경우 리터럴 개수를 많이 줄일 수 있었다. 반면에, 그림 3의 수행 시간을 비교한 경우를 보면 mux 회로의 경우 제안한 방법으로는 수행시간이 다소 늘어났으나 나머지 회로에 대하여 수행시간이 타 방법들과 유사한 결과를 보이고 있다. 그림 2와 3에서 그래프가 아래쪽에 위치할수록 우수한 것임을 나타낸다. 실험결과를 정리하면 제안한 방법이 다른 방법들 보다 수행시간은 다소 늘어났지만, 리터럴 개수를 줄이는 효과가 있음을 보이고 있다. 보통 CPU 등의 회로에는 본 실험에 사용된 벤치마크 회로와 유사한 종류의 회로가 반복해서 사용되는 경향이 많기 때문에, 반복되는 회로를 최적화하여 리터럴 개수를 줄이게 되면 전체 회로를 구성하는 총 리터럴 개수를 줄이는 효과가 매우 커지게 된다. 실험에 사용한 벤치마크 회로들은 실제 디지털 부품으로 사용되기 때문에 적어도 한 개의 리터럴 수를 줄이는 효과는 이 디지털 부품이 반복 사용 될수록 리터럴 개수를 줄이는 효과가 더욱 커질 것이다.



<그림 2> 리터럴 개수 비교 실험결과



<그림 3> 수행 시간 비교 실험결과

4. 결론

본 논문은 논리회로 수업에 이용하기 위한 코스웨어의 한 모듈로 서포트를 이용한 제수를 찾아 부울 인수분해식을 산출하는 방법을 제시하였다. 서포트를 추가할 경우 인수분해식 산출을 위한 제수의 종류 즉, 인수분해와 부울 공리를 적용하여 간략화 가능성이 있는 제수 개수가 증가하여 최종적으로 리터럴 수가 적은 부울 분해식 산출을 위한 선택의 폭이 커지게 된다. 제수의 개수가 늘어나기 때문에 수행 시간이 다른 방법보다 증가하는 단점이 있으나, 논리식 간략화를 위한 수행시간 보다는 리터럴 수를 줄이는 것이 중요할 경우 제안한 방법을 활용하는 것이 좋을 것으로 판단된다. 특히, 기존의 교육용 논리회로 합성 도구들과 함께 본 논문에서 제안하는 방법을 적용하면, 논리회로 수업에 있어 대수 인수분해식과 부울 인수분해식의 차이를 확인할 수 있을 것이다. 향후 본 연구 결과를 교육용뿐만 아니라 산업용으로 활용 가능하도록 발전시키기 위해서 리터럴 개수와 함께 수행시간도 줄이기 위한 방법에 대한 연구가 필요할 것으로 본다.

참 고 문 헌

[1] Lawler, E. (1964). An Approach to Multilevel Boolean Minimization. *Journal of ACM*, 11(3), 283-295.

[2] Brayton, R. K. and McMullen, C. (1982). The Decomposition and Factorization of Boolean Epressions. *Proc. ISCAS*, 49-54.

[3] Brayton, R. K., Rudell, R., Sangiovanni-Vincentelli, A. and Wang, A. R. (1987). MIS: A Multiple-Level Logic Optimization System, *IEEE Trans. CAD*, 6(6), 1062-1081.

[4] Brayton, R. K., Rudell, R., Sangiovanni-Vincentelli, A. and Wang, A. (1987). Multi-Level Logic Optimization and the Rectangle Covering Problem. *Proc. ICCAD*, 66-69.

[5] Sentovich, E. M., Singh, K. J., Moon, C., Savoj, H., Brayton, R. K., and Sangiovanni-Vincentelli, A. (1992). Sequential Circuit Design Using Synthesis and Optimization. *Proc. ICCD*, 328-333.

[6] Liao, S., Devadas, S., and Ghosh, A. (1993). Boolean Factoring Using Multiple-Valued Minimization. *Proc. ICCAD*, 606-611.

[7] Stanion, T. and Sechen, C. (1994). Boolean Division and Factorization Using Binary Decision Diagrams. *IEEE Trans. CAD*, 13(9), 1179-1184.

[8] Kwon, O.-H., Hong, S. J., and Kim, J. (1998). A Boolean Factorization Using and Extended Boolean Matrix. *IEICE Trans. Inf. and Sys.*, E81-D(12), 1466-1472.

[9] Yang, C. and Ciesielski, M. (2002). BDS: A Boolean BDD-Based Logic Optimization System. *IEEE Trans. CAD*, 21(7), 866-876.

[10] Modi, N. and Cortadella, J. (2004). Boolean Decomposition Using Two-literal Divisors. *Proc. of 17th International Conference on*

VLSI Design, 765-768.

[11] Mintz, A. and Golumbic, M. C. (2005). Factoring Boolean Functions Using Graph Partitioning. *Discrete Applied Mathematics,* 149, 131-153.

[12] Kwon, O.-H. and Chun, B. T. (2010). Boolean Factorization Using Two-cube Non-kernels. *Journal of the Korea Academia-industrial cooperation Society*, 11(11), 4597-4603.

[13] Kwon, O.-H. (2011). Common Expression Extraction Using Two-cube Qoutient Matrices. *Journal of the Korea Academia-industrial cooperation Society*, 12(8), 3715-3722.

권 오 형

1999 포항공과대학교
컴퓨터공학과(공학박사)
1990～1993 한국전자통신연구원
1999～2003 위덕대학교 컴퓨터공학과
2003～현재 한서대학교 전자컴퓨터통신학부
관심분야: 컴퓨터교육, 설계자동화
E-Mail: ohkwon@hanseo.ac.kr