Journal of the Korea Academia-Industrial cooperation Society
Vol. 12, No. 7 pp. 3251-3257, 2011

DOI : 10.5762/KAIS.2011.12.7.3251

커널-커널 쌍을 이용한 공통 논리식 산출

권오형[1*]
[1]한서대학교 전자컴퓨터통신학부

Common Expression Extraction Using Kernel-Kernel pairs

Oh-Hyeong Kwon[1*]
[1]**Division of Electroinc, Computer, and Communication, Hanseo University**

요 약 본 논문에서는 논리합성을 위한 공통식 추출 방법을 새롭게 제안한다. 제안하는 방법은 주어진 각 논리식들에서 커널/커널 쌍들과 코커널/커널 쌍을 추출한다. 커널/커널 쌍은 주어진 논리식을 부울 나눗셈에 의해 제수, 몫, 나머지로 논리식을 다시 표현하게 된다. 다음, 여러 논리식에서 산출된 제수, 몫들에서 공통식을 추출하는 커널 교집합에 의해 공통식을 구하는 방법을 제안한다. 실험 결과 기존의 공통식 산출 결과들과 비교했을 때 제안한 방법은 리터럴 개수를 줄일 수 있었다.

Abstract This paper presents a new Boolean extraction technique for logic synthesis. This method extracts kernel-kernel pairs as well as cokernel-kernel pairs. The given logic expressions can be translated into Boolean divisors and quotients with kernel-kernel pairs. Next, kernel intersection method provides the common sub-expressions for several logic expressions. Experimental results show the improvement in literal count over previous other extraction methods.

Key Words : Cokernel-kernel pair, Kernel-kernel pair, Boolean extraction

1. 서론

다단 논리회로(multi-level logic circuits)는 이단 논리회로(two-level logic circuits)보다 간결한 구조를 갖는다. 이단 논리회로 구현을 위한 방법은 카르노 맵(Karnaugh map)을 이용한 방법과 Espresso 등의 여러 방법이 제시되어 좋은 결과를 산출하고 있다. 그러나, 다단 논리회로 설계를 위한 다단 논리식 산출 방법에는 아직 알려진 최적 알고리즘이 없을 뿐만 아니라, 선험 알고리즘도 여전히 개선할 점이 많기 때문에 현재도 연구가 진행되고 있는 분야다. 일반적으로 다단 논리식을 산출하기 위해서는 다음과 같은 몇 가지 방법들로 구분하여 간략화를 수행하고 있다. 여러 출력들 사이에 존재하는 공통식 산출(extraction) 방법, 내부 무관항(internal don't care) 추출과 간략화, 논리식 별로 인수분해(factorization) 등으로 구분해서 주어진 논리식을 간략화하고, 궁극적으로 이러한 방법들을 혼합하여 간략화된 다단 논리회로를 산출한다. 본 논문은 다단 논리식을 산출하는 여러 방법 중에서 공통식을 산출하기 위한 방법을 제안한 것이다.

일반적으로 공통식 추출(extraction) 기법은 주어진 여러 논리식들을 리터럴 개수가 보다 적은 동일한 논리식들의 표현으로 변형하는 것이다. 이러한 공통식 추출 방법은 여러 논리식에서 공통으로 사용된 공통식을 찾고, 이 공통식을 새로운 변수로 대치하여 최적의 논리식들로 변형하게 된다. 보편적인 방법은 논리식들에서 다항 큐브(multiple-cube)로 구성된 제수(divisor)를 찾고, 이 제수를 새로운 변수로 치환하여 주어진 원 논리식들을 변형한다. 이러한 과정을 반복해서 다항 큐브의 제수를 찾아 치환하고, 더 이상 다항 큐브의 제수가 없으면 단항(single-cube)의 제수를 찾아 이 제수에 새로운 변수를 도입하고, 이 변수로 각 논리식을 변형하는 방법을 사용한다.

Brayton 등은 커널(kernel)을 이용한 다항 큐브 제수를 찾는 방법[1-3]을 제시하였다. 이들의 알고리즘은 레벨-0

*교신저자 : 권오형(ohkwon@hanseo.ac.kr)
접수일 11년 06월 13일 수정일 11년 07월 06일 게재확정일 11년 07월 07일

에 대한 커널만을 산출하고, 이 레벨-0 커널들의 교집합을 산출하는 방법에 기반을 두고 있다. 이렇게 제시된 커널 기반 기술에 의한 다항 큐브 제수 산출 방법은 대수적인 방법으로 공통식을 찾는 것인 데 수행 속도는 부울 방법에 의한 공통식 산출보다 빠르나, 때때로 최적화된 결과를 산출할 수 없게 된다. Rajski와 Vasudevamurthy는 다변수 출력을 갖는 논리회로에서 대수적 방법으로 단항 큐브의 공통식과 이 공통식의 보수를 고려하여 여러 출력단에서 동일한 공통식을 산출하는 방법[4]을 제시하였다. Singh와 Diwan는 변수 치환을 이용한 공통식 산출 방법[5]을 소개하였다. 지금까지 서술한 방법들은 대수적 나눗셈에 기반을 둔 방법으로 부울 공리인 등멱법칙($a\ a = a$)과 보수법칙($a\ a' = 0$)을 활용하지 못하는 단점을 갖게된다. 한편, Hsu와 Shen은 부울공리를 활용한 coalgebraic division이라는 대수 나눗셈 방법[6]을 고안하여 resubstitution 기술에 적용하여 리터럴 개수를 줄이는 데 활용하였다. 주어진 논리회로를 Binary-Decision Diagram(BDD)로 표현하고 BDD로부터 공통식을 찾고자 하는 노력을 노력을 하였다[7-9]. 이러한 최근의 결과는 단지 일부 회로에서 대수적 나눗셈을 이용한 방법[1-3]과 비교해서 보다 리터럴 개수를 줄일 수 있었다. 이러한 관점에서 부울공리를 적용한 공통식 산출 방법(이하 부울 공통식이라 부름)은 단순히 대수적인 나눗셈을 이용한 최적화보다 리터럴 개수를 줄일 수 있음을 알 수 있으나, 여전히 어려운 문제로 남아있다.

본 논문에서는 선험방법(heuristic method)에 의한 부울 공통식 산출 방법을 제안한다. 여기서 제시하는 방법은 [10-13]에서 제시한 방법을 다변수 출력 논리회로(multiple-output logic circuit)로 확장한 것이다. 제안하는 핵심 기술은 주어진 논리식들에서 먼저 등멱법칙과 보수법칙을 고려한 부울 나눗셈을 적용한 커널/커널 쌍을 산출한다. 다음, 커널들의 교집합을 찾아서 여러 출력에서 공통으로 사용되는 공통식을 추출하는 방법을 제안한다.

논문의 구성은 다음과 같다. 2절에서는 본 논문 서술에 필요한 용어들의 정의를 소개하고, 3절에서 본 논문에서 제시하는 새로운 공통식 산출 방법인 커널/커널 쌍을 이용한 공통식 추출 방법을 소개한다. 4절에서 실험결과를 보이고, 5절에서 결론을 제시한다.

2. 용어

본 논문의 공통식 산출 방법을 서술하는데 필요한 용어들에 대하여 서술한다.

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 리터럴 a가 존재하면, 그의 보수 리터럴 a'을 포함하지 않는다. 단순식(expression 또는 sum-of-products(SOP) form)은 큐브들의 집합이다.

예 1: 문자 a는 변수이며, a와 a'은 리터럴이다. 리터럴 집합 $\{a,b\}$는 큐브이나 집합 $\{a,a'\}$은 큐브가 아니다. $\{\{a,b'\},\{b,c\}\}$는 단순식이다.

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

정의 2: 논리식 F의 서포트(support)는 논리식 F를 구성하는 변수들 집합으로 $sup(F)$로 표현한다. 논리식을 구성하는 모든 큐브들 간에 공통으로 사용되는 리터럴을 갖지 않은 경우 그 논리식은 큐브면제(cube-free) 되었다고 한다. 논리식이 어떤 큐브로부터 나누어졌을 때, 몫이 큐브면제라면 그 몫을 커널이라 한다. 이 때 커널을 산출한 큐브를 코커널(co-kernel)이라 한다.

예 2: 논리식 $F = a + bc'$의 경우, $sup(F)$= {a, b, c}. 논리식 $ab + c$ 는 큐브 면제된 경우이나, 논리식 $ab + ac$ 및 abc는 큐브 면제된 것이 아니다. $F = bc'd'e + ab'c + ab'e + ac'd'$은 다시 $F = bc'd'e + a(b'c + b'e + c'd')$으로 표현될 수 있으며, 이 때 $b'c + b'e + c'd'$은 코커널 a에 대한 커널이 된다.

정의 3: 논리회로는 비순환 방향 그래프(directed acyclic graph)로 표현되며, 각 노드 i는 변수 y_i를 나타내고, 논리식 F_{y_i}를 표현한다. 2개의 논리식 F와 G의 곱 FG는 $\{C_i \cup D_j | C_i \in F$와 $D_j \in G\}$를 의미한다. F와 G가 서로 독립 서포트(disjoint support)인 경우 FG는 대수 곱이고, 그 외의 경우 FG는 부울 곱이다. F/G는 가장 큰 큐브 집합인 몫 Q를 산출하여, 논리식 F를 $F = QG + R$로 표현 할 수 있다. 여기서, R은 나머지를 나타낸다. QG가 대수 곱인 경우, F/G는

대수 몫을, 그 외의 경우 F/G는 부울 몫이 된다. $F/G = Q \neq \varnothing$ 이고 Q가 대수 나눗셈에 의해 산출된 경우, G는 대수 제수(algebraic divisor)라 하고, 그 외의 경우 부울 제수(Boolean divisor)라 한다.

예 3: $(a+b)(c+d) = ac+ad+bc+bd$는 대수 곱이며, $(a+b)(a+c) = a+ab+ac+bc$는 부울 곱이다. $F = ad+abc+bcd$ 이고 $G = a+bc$ 가 주어진 경우를 보자. F/G의 결과로는 대수 몫 d와 나머지 abc가 산출된다. 이 때, $a+bc$는 대수 제수가 된다. $F = abg+acg+adf+aef+afg+bd+be+cd+ce$와 $G = ag+d+e$가 주어진 경우, $F = (af+b+c)(ag+d+e)$로 표현될 수 있다. 이 때, $af+b+c$는 부울 몫이며, $ag+d+e$는 부울 제수가 된다.

3. 공통 논리식 산출

3.1 커널 기반 공통식 산출

Brayton과 McMullen은 커널 집합을 이용해서 논리식들에서 공통식을 산출하는 과정을 2단계로 제안하였다[1]. 논리식들에서 커널들 산출이 첫 번째 단계이며, 커널 교집합으로부터 공통식을 선별하는 과정이 두 번째 단계다.

예 4: 다음과 같이 논리식 F_0와 F_1이 주어졌다고 하자. $F_0 = ace+bce+de+g$, $F_1 = ad+bd+cde+ge$. 첫 번째 단계에서, F_0의 커널 집합 $K(F_0)$은 $K(F_0) = \{(a+b), (ac+bc+d), (ace+bce+de+g)\}$이며, F_1의 커널 집합은 $K(F_1) = \{(a+b+ce), (cd+g), (ad+bd+cde+ge)\}$. 두 번째 단계에서, F_0와 F_1의 커널로부터 공통식을 찾는다. 즉, $(a+b) \in K(F_0)$와 $(a+b+ce) \in K(F_1)$의 교집합 $a+b$를 얻게 된다. 그러면, 간략화된 다음의 논리식들이 산출된다.

$$F_0 = Gce+de+g$$
$$F_1 = Gd+cde+ge$$
$$G = a+b$$

3.2 커널/커널 쌍 기반 공통식 산출

커널-커널 쌍 기반 공통식 산출 방법은 커널-커널 쌍을 찾는 과정과 커널-커널 쌍들 사이에 존재할 수 있는 공통식을 찾는 과정으로 되어 있다.

3.2.1 커널/커널 쌍 산출

주어진 논리식 또는 논리회로의 각 출력별로 커널/커널 쌍을 산출한다. 커널/커널 쌍들은 코커널/커널들로부터 산출한다. 본 논문에서는 커널/커널 및 코커널/커널을 표시할 때 괄호를 이용한다. C를 코커널들의 집합이라 하고, K를 커널들의 집합이라 하자. 그리고, (c_i, k_i)와 (c_j, k_j)를 각각 코커널/커널 쌍이라 하고, $c_i \in C$, $c_j \in C$, $k_i \in K$, $k_j \in K$이고 $i \neq j$이라고 하자. 이 때 $c_i \in k_j$이고 $c_j \in k_i$인 경우 (k_i, k_j)를 커널/커널 쌍이라 한다.

예 5: 주어진 회로가 2개의 출력인 F_0과 F_1을 갖고 논리식은 다음과 같다고 가정하자.

$$F_0 = ab'c+ab'e+ac'd'+bc'd'e$$
$$F_1 = ab'c+ab'e+af+bef+b'cd+b'de+df$$

그러면, F_0에 대한 코커널/커널들은 다음과 같다.

코커널	커널
a	$b'c+b'e+c'd'$
ab'	$c+e$
$c'd'$	$a+be$
e	$ab'+bc'd'$

위의 코커널/커널 집합에서 두 개의 코커널/커널들 $(a, b'c+b'e+c'd')$과 $(c'd', a+be)$의 경우를 보자. $(a, b'c+b'e+c'd')$를 첫 번째 코커널/커널이라 하고, $(c'd', a+be)$를 두 번째 코커널/커널이라 하자. 그러면, 첫 번째 코커널/커널에 속하는 커널 $b'c+b'e+c'd'$는 두 번째 코커널/커널에 속하는 코커널과 동일한 큐브 $c'd'$를 포함한다. 또한 두 번째 코커널/커널에서 커널 $a+be$는 첫 번째 코커널/커널에 속하는 코커널 a와 동일한 큐브를 포함한다. 이러한 경우 첫 번째와 두 번째의 코커널/커널들에서 커널들만을 추출하여 커널/커널을 산출한다. 이 경우 $(b'c+b'e+c'd', a+be)$인 커널/커널이 얻어진다. 마찬가지로, 두 개의 코커널/커널 $(ab',$

$c+e$)와(e, $ab'+bc'd$)로부터 커널/커널 쌍 ($c+e$, $ab'+bc'd$)이 산출된다. 단순식 F_0에 대한 커널/커널 쌍들을 정리하면 다음과 같다.

커널	커널
$b'c+b'e+c'd'$	$a+be$
$c+e$	$ab'+bc'd'$

또한, F_1 에 대한 코커널/커널들은 다음과 같다.

코커널	커널
a	$b'c+b'e+f$
ab'	$c+e$
b'	$ac+ae+cd+de$
$b'd$	$c+e$
$b'c$	$a+d$
d	$b'c+b'e+f$
e	$ab'+bf+b'd$
f	$a+be+d$

F_0에서 커널/커널 쌍들을 구한 것과 같은 방법으로 F_1에 대한 커널/커널 쌍들을 구하면 다음과 같다.

커널	커널
$a+be+d$	$b'c+b'e+f$
$a+d$	$b'c+b'e+f$

3.2.2 공통식 산출

산출된 커널들을 이용해서 공통식을 찾고 공통식을 나타내는 새로운 변수를 도입한다. 다음 공통식을 포함하는 논리식은 새로운 변수를 이용해서 논리식을 간략화한다. 커널들에서 공통식 부분을 찾는 방법은 SIS(또는 MIS)에서 제시한 방법을 이용하였다. 이를 정리하면 다음과 같다. k_i, $(i=1,\cdots,n)$를 커널이라 하고, $K=\{k_1, k_2, \cdots, k_n\}$는 커널들의 집합이라 하자. 그러면, 커널 교집합 $I(K)$는 다음과 같은 방법으로 산출된다. 커널 집합 K에 대응하는 새로운 식 $IF(K)$를 만들고, $IF(K)$에서 제수들을 찾으면 바로 이것들 커널의 교집합이 된다. $IF(K)$를 만들 때, 커널들을 구성하는 큐브들을 새로운 변수로 치환하고, 커널을 새로운 변수들의 곱으로 나타낸다. 예로, 산출된 커널 중에 하나가 $k_1=b'c+b'e+c'd'$, $k_2=b'c+b'e+f$라 하자. 그러면, $t_1=b'c$, $t_2=b'e$, $t_3=c'd'$, $t_4=f$로 각 큐브를 치환하면, 커널 k_1은 $t_1t_2t_3$, 커널 k_2는 $t_1t_2t_4$가 되어 $IF(K)=t_1t_2t_3+t_1t_2t_4$로 표현한다. 이 때, $IF(K)$에서 제수 $Divisor(IF(K))$를 찾을 때는 논리식의 공리를 적용하지 않고 단지 대수식으로 가정하고 제수들을 찾는다. 이 제수들이 바로 커널 교집합 $I(K)$이다. 커널 교집합을 찾는 구체적인 예를 다음에 소개한다.

예 6: 다음과 같이 커널 집합 K가 산출되었을 때, 커널 교집합 $I(K)$를 산출하자.

$$K=\{k_1, k_2, k_3, k_4\}$$
$$k_1=b'c+b'e+c'd'$$
$$k_2=a+be$$
$$k_3=b'c+b'e+f$$
$$k_4=a+be+d$$

그러면, 커널을 구성하는 각 큐브에 새로운 변수를 배당한다.

$$t_1=b'c$$
$$t_2=b'e$$
$$t_3=c'd'$$
$$t_4=a$$
$$t_5=be$$
$$t_6=f$$
$$t_7=d$$

그러면, $IF(K)$는 새로운 변수들을 이용해서 $IF(K)=t_1t_2t_3+t_4t_5+t_1t_2t_6+t_4t_5t_7$가 된다. 이 $IF(K)$에서 제수들의 집합 $Divisor(IF(K))$를 산출하면 $Divisor(IF(K))=\{t_1t_2, t_4t_5\}$가 된다. 그러면, 커널 교집합 $I(K)$는 $IF(K)$의 제수로 해석할 수 있다. 그래서 $I(K)=\{t_1+t_2, t_4+t_5\}=\{b'c+b'e, a+be\}$가 된다.

3.2.3 가중치 계산

커널 교집합에 의해 다수 개의 공통식이 발견된 경우 주어진 여러 논리식들에서 가장 많은 리터럴들을 줄일 수 있는 공통식을 선택하는 것이 중요하다. 본 논문에서는 공통식 선택을 위해 가중치 부여 방법을 사용하였다. 공통식 q_i에 대한 가중치를 $weight(q_i)$로 나타내면, 식 (1)과 같다. 이 식 (1)은 SIS에서 제시한 것과 동일하다.

$$weight(q_i)=(NF(q_i)-1)(L(q_i)-1)-1 \quad (1)$$

여기서, $NF(q_i)$는 공통식q_i를 포함하는 논리식들의 개수이고, $L(q_i)$는 공통식q_i 자체의 리터럴 개수다.

3.2.4 알고리즘

알고리즘은 코커널/커널 쌍 추출과 코커널/커널 쌍으로부터 커널/커널 쌍들을 찾는 과정부터 시작된다, 커널/커널 쌍들을 구성하는 커널들 사이에 공통식들이 산출되면, 가중치가 가장 큰 공통식을 선택하여 새로운 변수를 부여한다. 다음, 선택된 공통식을 포함하는 주어진 논리식들은 새로운 변수를 이용하여 논리식들을 간략화 한다. 전체적인 알고리즘은 다음과 같다.

알고리즘 : 부울 공통식 추출
입력 : 단순식 형태의 다변수 출력 논리식
출력 : 공통식이 추출된 다변수 출력 논리식
방법 :
단계 1: 각 출력별로 코커널/커널 집합 *CK*을 산출;
단계 2: 커널/커널 집합 *KK* 산출;
단계 3: *KK*에 속하는 커널들의 집합 *K*에 대하여 제수들의 집합 *I(K)*를 산출;
단계 4: 가장 가중치가 큰 2개의 q_i와 q_j를 선택하고 각각에 새로운 변수 Q_i와 Q_j를 부여;
단계 5: 각 출력에서 q_i와 q_j를 포함하는 부분을 새로운 변수 Q_i와 Q_j로 대치;
단계 6: 나머지 부분은 코커널/커널 쌍과 단일항 공통식 추출에 의해 간략화;

예 7: 예 5의 2개의 주어진 논리식 F_0와 F_1에 대하여 위 알고리즘을 적용한다. 예 5는 알고리즘의 단계 1-2의 결과를 보인 것이다. 예 6은 단계 3의 결과를 보인 것이다. 단계 4는 F_0에 대한 커널/커널 쌍 $(b'c+b'e+c'd')(a+be)$, $(c+e)(ab'+bc'd')$와 출력 F_1에 대한 커널/커널 쌍 $(a+be+d)(b'c+b'e+f)$, $(a+d)(b'c+b'e+f)$로부터 커널들의 교집합 2개를 산출한다. 그러면, $q_1=b'c+b'e$, $q_2=a+be$를 얻는다. 즉, q_1은 2개의 식 F_0와 F_1에 포함하고, q_1은 4개의 리터럴로 구성되기 때문에 $weight(q_1)=(2-1)*(4-1)-1=2$ 또한 $weight(q_2)=1$이 산출되어, q_1과 q_2를 선택하고, $Q_1=b'c+b'e$, $Q_2=a+be$로 새로운 변수 Q_1, Q_2를 도입한다. 단계 5에서 F_0의 $b'c+b'e$에 해당 하는 부분은 Q_1으로, $a+be$에 해당

[표 1] 실험 결과
[Table 1] Experimental results

회로	입력 수	출력 수	SIS		two-cube		제안방법	
			리터럴 수	시간(초)	리터럴 수	시간(초)	리터럴 수	시간(초)
b12	15	9	124	0.1	119	0.1	118	0.1
rd53	5	3	77	0.2	71	0.1	71	0.2
rd73	7	3	176	0.4	169	0.4	172	0.5
rd84	8	4	243	0.2	236	0.2	234	0.3
con1	7	2	23	0.1	23	0.1	23	0.1
z4ml	7	4	70	0.2	61	0.3	63	0.2
cmb	16	4	70	0.1	73	0.1	70	0.1
vg2	25	8	107	0.1	112	0.1	105	0.2
decod	5	16	64	0.1	51	0.1	54	0.2
misex1	8	7	80	0.1	80	0.1	80	0.2
alu4	14	8	1755	2.0	1557	1.8	1554	2.3
sao2	10	4	203	0.3	192	0.4	192	0.5
e64	65	65	254	0.1	254	0.1	254	0.2
apex6	135	99	904	0.1	902	0.1	901	0.3
C880	60	26	702	0.1	628	0.2	640	0.3
C1355	41	32	1032	0.1	989	0.3	990	0.3
C1908	33	25	1469	0.1	982	0.3	980	0.3
C2670	233	140	1995	0.2	1509	0.4	1511	0.3
C5315	178	123	4355	0.2	3268	0.5	3274	0.5
C6288	32	32	4800	0.1	4705	0.7	4702	0.5
C7552	207	108	5968	0.1	4510	0.8	4510	0.7

하는 부분은 Q_2로 대치하여, $F_0 = (Q_1 + c'd')Q_2$로 같은 방법으로$F_1 = (Q_2 + d)(Q_1 + f)$로 표현한다. 그러면 단계 5까지의 결과는 다음과 같게 된다.

$$F_0 = (Q_1 + c'd')Q_2$$
$$F_1 = (Q_2 + d)(Q_1 + f)$$
$$Q_1 = b'c + b'e$$
$$Q_2 = a + be$$

단계 6에서는 더 이상의 논리식들 사이에 공통식이 없기 때문에 결과의 변동이 없다. 따라서 최종적으로 15개의 리터럴을 갖는 논리식을 산출하게 된다. 만일 $Q_1 = b'c + b'e$에 대하여 인수분해를 한 결과 $Q_1 = b'(c + e)$ 까지 고려하면 최종 결과는 14개의 리터럴로 논리식을 간략화하게 된다.

4. 실험 결과

제시한 알고리즘은 Pentium IV 1.4GHz CPU PC의 Linux 환경에서 실험하였다. 실험결과를 비교하기 위해서 SIS1.2와 최근에 발표된 two-cube에 의한 공통식 산출 방법들과 제안한 방법을 리터럴 수와 수행 시간을 대상으로 비교하였다. SIS1.2에서는 공통식을 추출하기 위한 명령 "gkx -b"를 먼저 수행하고, 다음 "gcx -b"를 수행한 결과를 보인 것이다. 참고로, SIS1.2의 "gkx"는 다항 공통식을 산출하여 최적화를 수행하는 명령이고, "gcx"는 단항 공통식을 찾아 최적화를 수행하는 명령이다. 실험 결과를 표 1에 정리하였다. 표 1의 첫 번째 열은 벤치마크 회로[14]의 이름이고, 다음 2개의 열은 벤치마크 회로의 입력 수와 출력 수를 나타낸 것이다. 다음 2개의 열은 SIS1.2가 수행한 결과이고, 그 다음 2개의 열은 two-cube 방식에 의한 결과를 보인 것이다. 마지막 2개의 열은 본 논문에서 제안한 방법으로 수행한 결과를 보인 것이다. 실험 결과를 보면 제안한 방법이 SIS1.2 보다 적은 수의 리터럴을 갖는 논리식을 산출하였다. 이유는 SIS1.2는 코커널/커널 쌍만으로 공통식을 산출하는데 반해, 제안한 방법은 코커널/커널 쌍과 커널/커널 쌍까지 고려하였기 때문이다. 따라서, 리터럴 수를 줄일 수는 있었지만 커널/커널 쌍을 산출하는데 소요되는 수행 시간이 다소 늘어났다. 또한, two-cube 방식과 비교했을 때 대부분의 벤치마크 회로에서 리터럴 개수를 줄일 수 있었다. Two-cube 방식에서는 공통식을 산출하는데 보수를 고려하였기 때문에, 일부 회로에서 제안한 방법보다 리터럴 개수를 줄일 수 있는 것으로 판단된다.

5. 결론

본 논문에서 제시한 방법은 공통식을 산출하는데 부울 공리를 적용한 공통식을 추출하여 대수 나눗셈에 의한 공통식 산출 방법보다 리터럴 개수를 줄일 수 있었다. 제안한 방법은 코커널과 커널만으로 공통식을 산출하는 대수 나눗셈에 의한 SIS1.2의 기능에 커널/커널 쌍까지 포함하도록 하였다. 또, two-cube 방식은 공통식이 2개의 큐브로 구성된 경우만을 찾는데 비해, 제안한 방법은 3개 이상의 큐브로 구성된 공통식까지 찾을 수 있도록 하였다. 그 결과 SIS보다 모든 회로에서 제안한 방법은 리터럴 개수를 줄일 수 있었다. 또한 2-큐브을 이용한 공통식 산출 방법보다 대부분의 회로에서 리터럴 개수를 줄일 수 있었다. 제안한 방법은 선험 방식에 의한 알고리즘으로 실험에서 보인 것처럼 대체로 좋은 결과를 보이나, 항상 우수한 결과를 보인다고 단정하기 어렵다. 따라서, 논리합성을 수행할 경우는 타 방법들과 함께 사용하면서 제안한 방법이 가장 우수한 결과를 산출하면 활용하는 방식으로, 즉 여러 논리 합성 방법 중의 하나로 활용하는 것이 좋을 것으로 생각된다.

References

[1] R. K. Brayton and C. McMullen, “The Decomposition and Factorization of Boolean Epressions.” *Proc. ISCAS*, pp. 49-54, 1982.

[2] R. K. Brayton, R. Rudell, A. Sangiovanni-Vincentelli, and A. R. Wang, “ MIS: A Multiple-Level Logic Optimization System.” *IEEE Trans. CAD*, Vol. 6, No. 6, pp. 1062-1081, 1987.

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

[4] J. Rajski and J. Vasudevamurthy, "The testability-preserving concurrent decomposition and factorization of Boolean expressions," *IEEE Trans. CAD*, Vol. 11, No. 6, pp. 778-79, 1992.

[5] V. K. Singh and A. A. Diwan, "Heuristic for decomposition in multilevel logic optimization," *IEEE Trans. VLSI*, Vol. 1, No. 4, pp. 441-445, 1993.

[6] W.-J. Hsu and W.-Z. Shen, "Coalgebraic division for multilevel logic synthesis," *Proc. of DAC*, pp. 438-442, 1992.

[7] C. Yang and M. Ciesielski, "BDS: A Boolean BDD-

Based Logic Optimization System," *IEEE Trans. CAD,* Vol. 21, No. 7, pp. 866-876, 2002

[8] D. Wu and J. Zhu, "FBDD: A Folded Logic Synthesis System," *Proc. of International Conference on ASIC(ASICON),* pp. 746-751, 2005.

[9] D. Wu and J. Zhu, "BDD-based Two Variable Sharing Extraction," *Proc. of Asia and South Pacific Design Automation Conference (ASPDAC),* pp. 1031-1034, 2005.

[10] O.-H. Kwon, S. J. Hong, and J. Kim, "A Boolean Factorization Using and Extended Boolean Matrix," *IEICE Trans. Inf. and Sys.,* Vol. E81-D, No. 12, pp. 1466-1472, 1998.

[11] O.-H. Kwon, I.-G. Oh, "Boolean Extraction Technique Using Two-cube Divsors and Complements," Journal of Korea Information Processing Society, Vol. 15-A, No. 1, pp. 9-16, 2008.

[12] O.-H. Kwon, "Logic Optimization Using Boolean Resubstitution" Journal of the Korea Academia-industrial cooperation Society, Vol. 10, No. 11, pp. 3227-3233, 2009.

[13] O.-H. Kwon, B. T. Chun, "Boolean Factorization Using Two-cube Non-kernels," Journal of the Korea Academia-industrial cooperation Society, Vol. 11, No. 11, pp. 4597-4603, 2010.

[14] IWLS 2005 Benchmarks, http://iwls.org/iwls2005/benchmarks.html

권 오 형(Oh-Hyeong Kwon) **[정회원]**

- 1999년 2월 : 포항공과대학교 컴퓨터공학과 (컴퓨터공학박사)
- 1989년 7월 ~ 1993년 2월 : 전자통신연구원 연구원
- 1999년 3월 ~ 2003년 2월 : 위덕대학교 컴퓨터공학과 교수
- 2003년 3월 ~ 현재 : 한서대학교 전자컴퓨터통신학부 교수

<관심분야>
회로설계자동화, 논리합성