한국산학기술학회논문지
Vol. 11, No. 11 pp. 4597-4603, 2010

2-큐브 비커널을 이용한 부울 분해식 산출

권오형[1*], 전병태[2]
[1]한서대학교 전자컴퓨터통신학부, [2]한경대학교 웹정보공학과

Boolean Factorization Using Two-cube Non-kernels

Oh-Hyeong Kwon[1*] and Byung Tae Chun[2]
[1]**Division of Electroinc, Computer, and Communication, Hanseo University**
[2]**Dept. of Web Information Engineering, Hankyong National University**

요 약 분해식 산출은 다단 논리식 산출에 매우 중요한 부분을 담당한다. 분해식의 리터럴 개수는 논리함수의 복잡도를 나타내는 기준이 되며, 또한 논리식을 회로로 구현할 경우 리터럴의 개수는 트랜지스터의 개수와 비례하게 된다. 분해식을 산출하는 수행시간과 최적화의 적정성을 맞추기 위해 분해식은 대수 분해식과 부울 분해식 산출로 구분하며, 부울 분해식이 대수 분해식보다 적은 리터럴 개수로 같은 논리식을 표현할 수 있다. 본 논문에서는 부울 분해식을 산출하기 위한 방법을 제시한다. 제안하는 핵심 방법은 2개의 2-큐브 비커널을 이용하여 이들의 곱을 구하여 부울 분해식을 산출하는 것이다. 벤치마크 회로를 통한 실험 결과 이전의 다른 분해식 산출 방법들보다 리터럴 개수를 줄일 수 있었다.

Abstract A factorization is a very important part of multi-level logic synthesis. The number of literals in a factored form is an estimate of the complexity of a logic function, and can be translated directly into the number of transistors required for implementation. Factored forms are described as either algebraic or Boolean, according to the trade-off between run-time and optimization. A Boolean factored form contains fewer number of literals than an algebraic factored form. In this paper, we present a new method for a Boolean factorization. The key idea is to identify two-cube nonkernel Boolean pairs from given expression. Experimental results on various benchmark circuits show the improvements in literal counts over previous other factorization methods.

Key Words : Boolean Factorization, 2-cube non-kernel

1. 서론

논리식 최적화는 논리회로를 간략화하고 궁극적으로는 반도체 칩의 크기를 최소화하기 때문에 논리식 최적화는 회로 설계 자동화의 중요한 단계임은 잘 알려진 사실이다. 이러한 논리식 최적화 방법은 크게 2단 논리식 최적화와 다단 논리식 최적화로 구분되어 개발되어 왔다. 일반적으로 2단 논리식 최적화는 항의 수를 최적화하는 것을 목표로 하고 있다. 반면에 다단 논리식의 최적화는 전체 논리식에 사용된 리터럴의 개수를 최적화하는 것을 목표로 한다. 특히, MOS 회로의 경우 트랜지스터의 개수는 논리식의 리터럴 개수와 비례하기 때문에, 최소 개수의 리터럴을 갖는 다단 논리식 산출을 목표로 많은 연구들이 수행되고 있다. 이러한 다단 논리식 산출 방법의 하나가 분해식(factored form)을 산출하는 것이다. 분해식 산출 방법은 주어진 논리식에서 반복 사용되는 공통 인수를 찾는 인수분해 방법을 적용한 것으로, 분해식은 단순식(sum-of-products form)보다 적은 리터럴 개수로 동일한 논리식을 표현할 수 있는 수단이다. 분해식 산출에 대한 연구로는 1964년에 Lawler[1]에 의해 최적 분해식을 산출할 수 있는 알고리즘이 발표되었다. 이 알고리즘은 전수 탐색(exhaustive search) 방법을 사용해서 가장 적은 리터럴을 갖는 분해식을 산출하는 방법이다. 그러나, 전수 탐색에 소요되는 시간이 매우 길기 때문에 실용성이 없다. 따라서, 분해식 산출에 대해서는 전수탐색 방법보다는 주로 선험 방법(heuristic method)이 이용된다.

분해식 산출의 선험 방법은 분해식 산출 수행 시간과 리터럴 개수의 최적화 중에 어디에 더 중점을 두느냐에

*교신저자 : 권오형(ohkwon@hanseo.ac.kr)
접수일 10년 10월 29일 수정일 10년 11월 18일 게재확정일 10년 11월 19일

따라 대수 분해 방법과 부울 분해 방법으로 구분한다. 대수 분해 방법에 있어서는 논리함수를 대수 다항식으로 간주한다. Brayton 등[2-5]은 코커널/커널을 이용하여 대수 분해식을 산출하는 방법 즉, 커널 기반 방법(kernel-based method)을 제안하였다. 이 분해 방법은 분해식 산출 속도를 빠르게 향상시켰으나, 때때로 최소 개수의 리터럴을 갖는 분해식을 산출할 수 없는 단점을 갖는다. 반면에, 부울 분해 방법은 대수 분해 방법과 비교하면 보다 적은 개수의 리터럴을 갖는 분해식을 산출할 수 있는 장점을 갖는다. 최근의 부울 분해식 산출 방법들을 정리하면 다음과 같다. Liao 등[6]은 다치 함수(multi-valued function)와 분지 및 한계 커버 방법(branch-and-bound covering)을 이용한 부울 분해 방법을 제안하였다. Stanion 등[7]은 Liao의 방법이 대수 분해 방법에 비해 리터럴 개수를 줄이는 데 효과가 매우 작다고 지적하였다. 그래서, 효과를 높이기 위해서 Stanion 등은 부울 분해식을 산출하는 데 Binary Decision Diagram(BDD)를 이용하였다. 또한, 이들은 부울 나눗셈과 부울 분해를 수행하기 위해서 BDD를 활용하는 방법을 제시하였다. Kwon 등[8]은 코커널 규브 행렬을 확장하여 부울 분해식 산출 방법을 제시하였다. 이 방법은 대수 분해식을 산출하기 위한 SIS에서 사용하는 행렬을 포함하는 수퍼행렬을 만들어 부울 분해식을 산출하도록 고안하였다. Yang 등[9]은 BDD로 논리함수를 표현하고, 다시 BDD를 나누는 BDD 분리 엔진(BDD decomposition engine)을 제안하였으며, 이 엔진으로부터 분리 트리(factoring tree)를 만드는 방법을 제시하였다. Modi 등[10]은 2개의 리터럴만을 갖는 제수를 구해 논리식을 분해하는 방법을 제안하였다. 최근 연구로 Mintz 등[11]은 그래프 나누기(graph partitioning) 방법을 이용해서 분해식을 산출하는 방법을 제안하였다. 그러나, 이러한 부울 분해식 산출 방법 모두 최적의 결과를 산출하지 못하는 문제점과 때때로 대수 분해식보다 리터럴 개수가 많은 분해식을 산출하는 단점을 갖고 있다. 따라서, 부울 분해식 산출 방법에 대한 연구가 여전히 진행 중에 있는 실정이다. 이상의 연구들이 각 출력별로 논리 최적화를 위한 방법들인 반면에 여러 개의 출력을 갖는 회로에서 하나의 출력이 다른 출력의 일부분에 해당하는 경우 회로를 간략화 하는 부울 대입방법이 제시되었다[12]. 한편, 최근에는 VHDL 등과 같이 하드웨어로 설계된 회로 표현을 최적화하기 위한 연구가 진행되고 있다[13]. 이러한 상위 단계에서 표현된 회로 설계를 최적화하기 위해서는 본 연구의 주제인 논리식 최적화 단계를 이용해야 하기 때문에 논리식 최적화가 중요한 모듈로 사용되고 있다. 따라서, 본 논문에서는 부울 분해 방법으로 논리식을 최적화하는 방안을 제시한다.

2. 부울 분해 방법

본 절에서는 제안하는 분해 방법을 서술하는 데 기본이 되는 용어에 대하여 먼저 서술한다. 다음, 제안하는 분해식 산출 행렬 방법과 분해식 산출 행렬로부터 분해식을 산출하기 위한 행렬 커버링(covering) 알고리즘을 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 서포트를 $sup(F)$로 표기한다. 그러면, $sup(F)=\{x|$큐브 $C\in F$에 대하여, $x\in C$ 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. 큐브로부터 나누어진 몫이 큐브면제가 아닌 경우 그 몫은 비커널(non-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 cdf 이다. 반면에 $F=abfg+cd(a'ef+fg)$로 표현한 경우 $a'ef+fg$ 는 비커널이 된다. 이와 같이 비커널이 2개의 항으로 구성되었기 때문에 2큐브 비커널이 된다.

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

1) 리터럴은 분해식이다.
2) 분해식들의 곱은 분해식이다.
3) 분해식들의 합은 분해식이다.

분해식은 단순식들이 합과 곱으로 반복해서 표현된 논리식이다.

정의 5: 등멱법칙($a\cdot a=a$)과 보수법칙($a\cdot a'=0$)을 적용할 필요없이 분해식을 구성하는 단순식들을 곱할 수 있는 경우 대수 분해식(algebraic factored form)이라 한다. 대수 분해식 이외의 경우 부울 분해식(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 2큐브 비커널 공통인수 쌍 추출

주어진 단순형의 논리식에서 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, $c_i\in C$, $c_j\in C$, $q_i\in Q$, $q_i\in Q$ 이고 $i\neq j$ 라 하자. 그러면, (c_i, q_i), (c_j, q_j)는 대수 나눗셈에 의한 제수/몫 쌍을 표현한 것이다. 만일 $c_i\in q_j$, $c_j\in q_i$ 이고 q_iq_j 가 주어진 논리식에 포함되면, (q_i,q_j)는 주어진 논리식에 포함되는 부분 부울 부울식이라 한다. 여기서, q_i 와 q_j가 큐브 면제일 필요는 없다.

예 5: 논리식 $F=abfg+a'cdef+cdfg$ 로부터 표 1과 같이 각 큐브에 양의 값을 갖는 인덱스를 부여한다. 다음 2개의 큐브들 사이의 공통인수를 추출하여 산출한 대수 분해식을 표 2에 나열한다. 표 2에서는 분해식을 제수와 몫을 열(column)로 구분해서 표시한다. 표 2의 제 1열은 설명을 쉽게 하기 위해 부여한 행 번호다. 표 3은 표 2의 행 3에서 몫이 2-큐브 비커널이 되도록 수정한 것이다.

[표 1] 큐브 인덱스

큐브 C_i	$abfg$	$a'cdef$	$cdfg$
인덱스 *index*(C_i)	1	2	3

[표 2] 2규브 커널과 제수

행	선택된 큐브	분해식	
		제수	2-큐브 커널(몫)
1	$C_1\cap C_2$	f	$abg+a'cde$
2	$C_1\cap C_3$	fg	$ab+cd$
3	$C_2\cap C_3$	cdf	$a'e+g$

[표 3] 2규브 비커널과 제수

행	선택된 큐브	분해식	
		제수	2-큐브 비커널
1	$C_1\cap C_2$	f	$abg+a'cde$
2	$C_1\cap C_3$	fg	$ab+cd$
3	$C_2\cap C_3$	cd	$a'ef+fg$

표 3으로부터 행2와 행3의 분해식을 보면 행2의 제수가 행3의 몫에 포함되고, 다시 행3의 제수가 행2의 비커널에 포함된다. 또한 행2와 행3의 분해식의 논리곱(AND)을 구하면 주어진 논리식에 포함되며, 이는 부울 분해식이 된다. 이를 정리하면 표 4와 같다.

[표 4] 부분 부울 분해식

선택된 2개 행	비커널과 비커널 곱
행2, 행3	$(ab+cd)(a'ef+fg)$

2.2 분해식 산출 행렬 M

단순식 F가 주어졌을 때, 분해식 산출 행렬 M은 2개의 큐브로부터 표 2와 같이 제수와 몫으로 구성된 쌍과 표 4와 같은 비커널과 비커널의 쌍을 이용한다. 행렬 M의 행은 제수와 몫과 몫의 쌍에 포함되는 큐브에 대응하여 각 행이 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.

$$M(\alpha_i,\beta_j)=\begin{cases} index(\alpha_i\beta_j) & \text{if } \alpha_i \in C, \\ & \beta_j \text{는} \alpha_i \text{로나눈 몫에 속하는 큐브} \\ index(c) & \text{if } \alpha_i \text{와} \beta_j \text{는 몫과 몫 쌍에 포함되는} \\ & \text{큐브, } c=\alpha_i\beta_j \neq 0 \\ * & \text{if } \alpha_i\beta_j = 0, \\ & \alpha_i \text{와} \beta_j \text{는 몫과 몫 쌍의 큐브} \\ 0 & \text{그 외의 경우} \end{cases}$$

예 6: 예 5의 $F=abfg+a'cdef+cdfg$에 대한 표 1, 표 2와 표 4을 이용해서 분해식 산출 행렬 M을 만든다. 행렬의 행들은 표 3의 제수에 해당하는 큐브와 표 4의 2-큐브 비커널과 2-큐브 비커널의 쌍에 포함되는 큐브들에 대응하도록 한다. 다음 열은 표 3의 몫에 포함되는 큐브들에 대응되도록 한다. 그러면, 행렬의 행은 집합 $\{f, fg, cdf, cd, ab, a'ef\}$의 각 원소에 대응된다. 행렬의 열은 집합 $\{abg, a'cde, ab, cd, a'e, g, a'ef, fg\}$의 각 원소에 대응된다. 산출된 분해식 산출 행렬은 표 5와 같다. 표 5의 행렬에서 첫 번째 행과 첫 번째 열에 해당하는 $M(1,1)=M(f,abg)$의 원소 값은 $index(abfg)=1$가 된다. 또한 여섯 번째 행과 세 번째 열에 해당하는 $M(6,3)=M(a'ef,ab)$의 원소 값은 $a'ef \cdot ab=0$가 되기 때문에 $M(6,3)=*$ 가 된다. 나머지 원소들에 대해서도 같은 방법으로 행렬에 값이 할당된다.

[표 5] 부울 분해식 산출 행렬

행 \ 열	abg	$a'cde$	ab	cd	$a'e$	g	$a'ef$	fg
f	1	2						
fg			1	3				
cdf					2	3		
cd							2	3
ab							*	1
$a'ef$			*	2				

2.3 분해식 산출 행렬 커버와 분해식

본 절에서는 사각형에 대한 정의를 하고 커버링 방법에 대하여 설명한다.

2.3.1 사각형 커버

정의 6: 부울 분해식 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.

위의 정의는 사각형 또는 프라임 사각형에 포함되는 원소들은 서로 다른 양의 정수 값과 다수의 *(don't-care)를 포함할 수 있음을 의미한다. 또한 사각형에 포함되는 행들이나 열들은 서로 인접할 필요는 없다.

사각형 커버 문제를 풀기 위해서 부울 분해식 산출 행렬에서 모든 양의 정수 원소를 커버하는 프라임 사각형들을 찾는다. 그리고, 최소 커버 비용으로 행렬을 커버하는 프라임 사각형들의 부분 집합을 산출한다. 그림 1의 알고리즘은 행렬에서 프라임 사각형들을 산출한다. 이 때, 사각형 정의에 따라 프라임 사각형은 다수의 *를 포함할 수 있으나, 다수의 동일한 인덱스를 포함할 수 없다. 그림 1의 알고리즘에서 PRIME_RECT는 각 $M(\alpha, \beta)>0$인 행 α와 β에 대하여 EXPAND를 호출한다. EXPAND는 $M(\alpha, \beta)>0$인 원소에 대하여, 행 α에서 열 β를 포함하는 열들의 부분 집합을 찾는다. 다음, EXPAND는 열의 부분 집합인 C_k에 대응하는 행들의 부분 집합 R_k를 찾는다. 그러면, 이 행과 열의 부분 집합이 프라임 사각형 (R_k, C_k)이다.

Algorithm 1: 프라임 사각형 산출.
입력: 부울 분해식 산출 행렬 M.
출력: 프라임 사각형 집합 P.
방법:

PRIME_RECT(M)
  $P = \phi$;
  for α in set of rows in M **do**
    for β in set of columns in M **do**
      if $(M(\alpha, \beta) > 0)$
      then $P = P \cup$ EXPAND (α, β, M);
  return(P);

EXPAND (α, β, M)
  $LP = \phi$;
  $RM =$ set of rows in M;
  $CM =$ set of columns in M;
  find all subset of columns, $C = C_k$, such that

$$C_k = \{\beta\} \cup \begin{array}{l}\{l | M(\alpha,l) \neq 0, \forall\, l \in CM, \text{and if } M(\alpha,l) > 0 \\ \quad \text{then } M(\alpha,l) \neq M(\alpha,c), \forall\, c \in C_k\};\end{array}$$

  for each $C_k \in C$ **do**
    find subset of rows, R_k, such that

$$R_k = \{\alpha\} \cup \begin{array}{l}\{l | M(l,m) \neq 0, \forall\, l \in RM, \forall\, m \in C_k, \text{ and if } M(l,m) > 0 \\ \quad \text{then } M(l,m) \neq M(\gamma,c), \forall\, \gamma \in R_k, \forall\, c \in C_k\};\end{array}$$

    $LP = LP \cup (R_k, C_k)$;
  return(LP);

[그림 1] 프라임 사각형 산출 알고리즘

Algorithm 4: 부울 분해식 산출.
입력: 주어진 단순식 F.
출력: 부울 분해식.
방법:

FACTOR(F)
  if (there is not any common literal to some cubes in F)
    then write(F);
    else return;
  Construct an extended co-kernel cube matrix M;
  $P =$ PRIME_RECT(M);
  select a prime rectangle, $best\_rect$, with best cost;
  $Q =$ expression corresponding to rows of $best\_rect$;
  FACTOR(Q);
  $D =$ expression corresponding to columns of $best\_rect$;
  FACTOR(D);
  $R = F - QD$;
  if ($R \neq \phi$) **then** write("+");
    else FACTOR(R);

[그림 2] 부울 분해식 산출 알고리즘

[표 6] 실험 결과

회로	입력수	출력수	SIS 1.2[5]		KK[8]		XF[11]		제안 방법	
			리터럴 수	시간 (초)	리터럴 수	시간 (초)	리터럴 수	시간 (초)	리터럴 수	시간 (초)
5xp1	7	10	161	0.3	161	1.2	161	0.8	161	0.4
xor5	5	1	28	0.2	28	1.4	28	5.5	28	0.5
majority	5	1	10	0.1	10	0.1	9	0.3	9	0.2
f51m	8	8	168	0.3	164	2.9	143	0.8	143	0.3
z4m1	7	4	69	0.1	67	5.3	48	0.2	64	0.1
mux	5	1	79	0.6	71	16.3	47	20.2	68	11.7
misex1	8	7	86	0.1	86	0.1	88	0.4	86	0.1
misex2	25	18	164	0.1	163	0.1	163	0.2	163	0.1
decid	5	16	80	0.1	80	0.1	80	0.3	78	0.3

예 7: 표 5와 같은 분해식 산출 행렬 M이 주어졌다고 하자. $(\{fg\},\{ab,\ cd\})$는 사각형이다. 그러나, $(\{fg\},\{ab,cd\})$는 $(\{fg,a'ef\},\{ab,\ cd\})$ 에 포함되기 때문에 프라임 사각형은 아니다. 반면에, $(\{fg,\ a'ef\},\{ab,\ cd\})$을 포함하는 사각형이 없기 때문에 프라임 사각형이라 한다. 사각형 $(\{fg,\ a'ef\},\{ab,cd\})$의 사각형 비용은 행과 열을 나타내는 리터럴의 개수 합으로 9가 된다.

최소 비용을 갖는 사각형의 집합을 산출한 후, 사각형에 대응되는 분해식을 산출한다. 그림 2는 프라임 사각형들의 집합으로부터 부울 분해식을 산출하는 알고리즘이다. 그림 2의 분해식 산출 알고리즘은 재귀 호출에 의해 나눗셈에 의한 몫, 제수, 나머지에 대하여 다시 부울 분해식 산출을 하도록 호출하면 한다. 이렇게 반복 호출를 하면 다단의 분해식이 산출된다. 예로, 예 7의 프라임 사각형 $(\{fg,a'ef\},\{ab,cd\})$에 대응되는 분해식으로 $F=(fg+a'ef)(ab+cd)$가 산출되고, $fg+a'ef$에 대하여 다시 분해식 산출 행렬을 만들고 사각형 커버를 찾으면, 최종적으로 리터럴 개수가 8개인 분해식 $F=(f(g+a'e))(ab+cd)$가 산출된다. 반면에 대수적 분해식 방법으로는 $F=f(abg+\ cd(a'e+g))$가 산출되며 리터럴 개수는 9개가 된다.

3. 실험 결과

주어진 2개의 논리식을 행렬로 표현하고 부울 공리를 적용하여 부울 대입식을 찾는 방법을 제시한다.본 논문에서 제안한 방법을 PLA 형의 여러 MCNC (Microelectronics Center of North Carolina) 벤치마크 회로(benchmark circuit)에 대하여 테스트하였고, 그 결과를 SIS 1.2[5], KK[8], XF[11]들과 비교하였다. SIS는 코커널과 커널 기반으로 대수 분해식을 산출하는 대표적인 방법이며, KK[8]은 SIS의 커널 기반 방식을 개선하여 부울 분해식을 산출하도록 제안한 방법이다. 또한 XF[11]은 가장 최근에 발표된 부울 분해식 산출 방법이기 때문에 본 논문에서 제안한 방법과 비교 대상으로 선정하였다. 표 6은 실험 결과를 정리한 것으로 왼쪽부터 3개의 열은 벤치마크 회로의 이름과 입력 수, 출력 수를 정리한 것이다. 성능은 최종 리터럴의 개수와 수행 시간을 대상으로 비교하였다. 수행 시간은 각 회로를 최적화하는데 소요되는 시간을 초단위로 산출하였다. 제안한 방법으로 얻은 결과는 오른쪽의 2 개의 열에 나열되어 있다. 실험은 Pentinum IV 1.4GHz CPU로 Linux 환경에서 진행하였다. 표 6의 벤치마크 회로에 대하여 제안한 방법이 대부분에서 타 방법들 보다 우수함을 보이고 있다.

4. 결론

본 논문은 비커널을 고려하여 부울 분해식을 산출할 수 방법을 제시하였다. 주어진 논리식에서 2 개의 큐브를 선택하고 2 개의 큐브로 구성된 2-큐브 비커널 분해식을 산출하는 방법이 커널 산출 방법보다 간단하면서 부울 분해식 산출도 가능하게 하는 행렬을 만들 수 있는 기반을 제공한다. 제안한 방법은 선험 방식에 의해 논리식 방법이기 때문에 f51m이나 misex2와 같이 큰 회로에 대하여도 수행 시간이 크게 늘어나지 않고 리터럴 개수를 줄일 수 있었다.

참고문헌

[1] E. Lawler, "An Approach to Multilevel Boolean Minimization," *Journal of ACM*, Vol. 11, No. 3, pp. 283-295, 1964.

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

[3] 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.

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

[5] 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.

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

[7] T. Stanion and C. Sechen, "Boolean Division and Factorization Using Binary Decision Diagrams," *IEEE Trans. CAD*, Vol. 13, No. 9, pp. 1179-1184, 1994.

[8] 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.

[9] C. Yang and M. Ciesielski, "BDS: A Boolean BDD-Based Logic Optimization System," *IEEE Trans. CAD,* Vol. 21, No. 7, pp. 866-876. 2002.

[10] N. Modi and J. Cortadella, "Boolean Decomposition Using Two-literal Divisors," *Proc. of 17th International Conference on VLSI Design,* pp. 765-768, 2004.

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

[12] 권오형, “부울 대입에 의한 논리식 최적화,” 한국산학기술학회 논문지, 제10권, 제11호, pp. 3227-3233, 2009.

[13] 정준모, “SoC 내의 효율적인 Test Wrapper 설계,” 한국산학기술학회 논문지, 제10권, 제6호, pp. 1191-1195, 2009.

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

전 병 태(Byung-Tae Chun) **[정회원]**

- 2001년 2월 : 고려대학교 컴퓨터과학과 (박사)
- 1989년 ~ 1996년 : 한국과학기술연구원(KIST)시스템공학연구소 연구원
- 1996년 ~ 2004년 : 한국전자통신연구원 선임연구원
- 2004년 2월 ~ 현재 : 한경대학교 웹정보공학과 교수
- 1992년 5월 : IR52 장영실상 수상 (과기부 장관상)
- 2003년 8월 ~ 현재 : 한국저작권위원회 감정전문위원
- 2007년~ 현재 : 한국정보기술학회 이사
- 2007년 ~ 2009년 : 한국전자통신연구원 초빙연구원

<관심분야>
영상처리, 멀티미디어 영상처리