論文
8-4-3

확장된 2-큐브 행렬을 이용한 부울 분해식 산출
(A Boolean Factorization Using an Extended Two-cube Matrix)

권오형(Oh-Hyeong Kwon)[1], 오임걸(Im-Geol Oh)

요 약

분해식은 SOP 형태의 논리식들이 논리합과 논리곱으로 반복해서 표현된 논리식이다. 분해식을 산출하는 과정은 논리식 내에 있는 공통식을 찾아 인수분해를 반복하는 과정이다. 분해식의 형태에 따라 대수 분해식과 부울 분해식으로 구분되며, 리터럴 개수를 기준으로 부울 분해식이 대수 분해식보다 간략화된 형태를 갖는다. 본 논문은 부울 분해식 산출 방법을 제안한 것이다. 제안하는 방법은 주어진 논리식에서 2개의 큐브를 선택하여 제수/몫 쌍들을 산출한다. 이 때, 2개의 큐브로 구성된 몫에 공통인수를 남겨두어 확장 제수/몫 쌍들을 산출하고 후에 몫/몫 쌍들을 산출하도록 하였다. 산출된 제수/몫 쌍과 확장 제수/몫 쌍, 몫/몫 쌍들을 이용하여 부울 분해식 산출을 위한 행렬을 산출하고, 행렬 커버링을 통해 부울 분해식을 산출하는 방법을 제시한다.

ABSTRACT

A factored form is a sum of products of sums of products, ... , of arbitrary depth. Factoring is the process of deriving a parenthesized form with the smallest number of literals from a two-level form of a logic expression. The factored form is not unique and described as either algebraic or Boolean. 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 Boolean subexpressions from given two-level logic expression and to extract divisor/quotient pairs. Then, we derive extended divisor/quotient pairs, where their quotients are not cube-free, from the generated divisor/quotients pairs. We generate quotient/quotient pairs from divisor/quotient pairs and extended divisor/quotient pairs. Using the pairs, we make a matrix to generate Boolean factored form based on a technique of rectangle covering.

논문접수 : 2007. 9. 12.
심사완료 : 2007. 10. 4.

1) 정회원 : 한서대학교 인터넷공학과

1. 서론

논리합성은 반도체 설계 자동화를 위한 중간 단계로 논리식(또는 논리회로) 최적화를 목표로 한다. 논리합성의 결과는 목적에 따라 크게 이단 또는 다단의 논리식을 산출한다. 보통 이단의 논리식을 산출하는 경우는 논리곱항의 수를 줄이는 것을 목표로 한다. 반면에 다단 논리식을 산출하는 경우는 논리식을 구성하는 리터럴 개수를 최소화하기 위함이다. 특히, 리터럴 개수는 MOS 회로에 사용된 트랜지스터의 개수와 비례하는 특징을 갖기 때문에 회로설계 자동화(ECAD) 연구를 진행하는 연구자들이 최소 리터럴을 갖는 다단 논리식 산출을 위한 다각적인 연구를 진행하고 있다. 다단 논리식 산출을 위한 방법 중의 하나로, 주어진 논리식을 인수분해하는 방법은 오래 전부터 진행되어 왔던 연구 주체다. 본 논문에서는 인수분해에 의해 산출된 논리식을 분해식(factored form)이라 부른다. 최초로, 최적 분해식을 산출하기 위한 연구 결과가 Lawler에 의해 발표되었다[1]. 이 방법은 모든 분해식을 산출하고 그 중에서 가장 작은 개수의 리터럴을 갖는 분해식을 선택하는 방법이다. 이 방법은 모든 분해식을 산출하는 데 소요되는 시간이 무척 길기 때문에 매우 작은 논리식에 대한 분해식을 산출하는 경우를 제외하고는 사용하기에 어려움이 있다. 그래서, 분해식을 산출하기 위해서는 선험 방법(heuristic method)에 의한 산출 방법이 이용된다. 분해식을 위한 선험 방법은 크게 대수 분해 방법과 부울 분해 방법으로 구분할 수 있다. 대수 분해 방법은 수행 시간을 줄이는 장점을 갖는다. 반면에, 부울 분해 방법은 수행 시간은 대수 분해 방법보다 더 소요되나 리터럴 개수를 줄일 수 있는 장점을 갖는다.

대수 분해 방법과 부울 분해 방법에 대한 기존 연구를 정리하면 다음과 같다. 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, 항상 최소 개수의 리터럴을 갖는 분해식을 산출할 수 없는 단점을 갖는다. 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 큐브 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. 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. 최근 연구로 [12]의 방법은 주어진 논리식에서 2개의 큐브들 간에 공통인수를 추출하여 제수/몫 쌍을 산출하고 다시 몫과 몫 쌍들 사이에 부울 공리를 적용하여 부울 분해식을 산출하는 방법을 제안하였다. [12]의 방법은 커널을 산출하는 데 소요되는 시간을 줄일 수 있는 장점을 갖으면서 부울 분해식을 산출할 수 있는 장점도 갖는다. 그러나, 2개의 큐브 사이에 얻어진 몫과 몫 쌍 사이에 부울 공리를 적용할 수 없는 경우 대수 분해 방법보다 결과가 좋지 않을 가능성이 있다. 따라서, 본 논문에서는 [12]를 개선한 방법을 제시한다.

본 논문의 구성은 다음과 같다. 제 1절에서는 지금까지 서술한 서론이며, 제 2절에서는 본 논문에서 제시하는 부울 분해 방법을 서술한다. 제 3절에서는 벤치마크 회로들에 대하여 제안한 방법으로 실험한 결과를 보였다. 마지막으로 제 4절에서는 본 논문의 결론을 제시한다.

2. 부울 분해 방법

본 절에서는 제안하는 분해식 산출 방법에 대하여 서술한다. 제안하는 방법은 [12]의 방법을 확장한 것이기 때문에 [12]에서 제시한 제수/몫, 몫/몫 쌍 산출 방법을 다시 소개하고, 본 논문의 핵심인 확장 제수/몫 쌍 산출 방법에 대하여 설명한다. 제수/몫 쌍들과 확장 제수/몫 쌍들로부터 몫/몫 쌍들을 산출하고, 이 쌍들로부터 확장 2-큐브 행렬을 만드는 과정을 서술한다. 확장 2-큐브 행렬로부터 분해식을 산출하기 위한 행렬 커버링(covering) 알고리즘은 [12]에서와 같이 Kwon 등[8]의 알고리즘을 이용하였다. 따라서, 본 논문에서는 확장 2-큐브 행렬 커버링 알고리즘에 대한 소개는 제외하였다. 본 절에서는 본 논문 서술에 필수적인 용어를 소개하는 것으로 시작한다. 참고로, 여기에 소개되는 용어는 [12]에서 제시된 용어 설명을 그대로 인용하였음을 밝힌다.

2.1 관련 용어

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 수식 표기를 모두 사용한다. 따라서 큐브$\{a,b\}$는 ab와 동일

한 표현이며, 단순식{{a,b'},{b,c}}는 $ab'+bc$와 동일한 표현이다.

정의 2: 분해식(factored form)은 단순식들이 합과 곱으로 표현된 것이다. 구체적인 정의는 다음과 같다.

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

정의 3: 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 2: $F=bc'd'e+ab'c+ab'e+ac'd'$와 $F=a(b'(c+e)+c'd')+bc'd'e$ 모두 대수 분해식이다. $F=(a+be)(b'(c+e)+c'd')$는 부울 분해식이다.

2.2 공통인수 쌍 추출

주어진 단순식에서 2개의 큐브를 선택하고, 이 2개의 큐브들에서 공통 큐브를 찾아 제수/몫 쌍을 구한다. 그러면, 제수는 공통 큐브가 되며 몫은 2개의 큐브로 산출된다. D를 제수 집합, Q를 2개의 큐브로 구성된 몫 집합이라 하자. 본 논문에서는 제수/몫 쌍을 괄호를 이용하여 표현한다. 이 때, $d_i \in D$, $q_i \in Q$ 로 가정하면, (d_i,q_i)는 대수 나눗셈에 의한 제수/몫 쌍을 표현한 것이다. 또, 제수/몫 쌍 (d_i,q_i)에서 제수 d_i가 2개 이상의 리터럴로 구성되어 있다면 제수/몫 쌍을 다음과 같이 확장한다. 즉, 제수 d_i에 포함되는 리터럴 d_{i_j}에 대하여 제수 d_i에서 리터럴 d_{i_j}을 제거하고, 대신 리터럴 d_{i_j}을 q_i의 각 항에 곱한다. 본 논문에서는 이렇게 제수에 포함된 리터럴을 제거하면서 산출된 제수/몫 쌍을 확장 제수/몫 쌍이라 부르며, 확장 제수/몫 쌍들은 $(d_i/d_{i_j},\ q_i * d_{i_j})$와 같이 표현될 수 있다. 그러면, 제수/몫 쌍들과 확장 제수/몫 쌍들 모두를 (c_i,t_i), $i=1,\cdots,n$, 로 표기하자. 그러면, 서로 다른 2개의 쌍 (c_i,t_i)와 (c_j,t_j) 사이에 $c_i \in t_j$, $c_j \in t_i$ 이고 t_i와 t_j의 논리곱(AND) t_it_j 가 주어진 논리식에 포함되면, (t_i,t_j)를 몫/몫 쌍이라 한다.

예 3: 단순식 $F=abde+a'cd+cde$에 대하여 제수/몫 쌍과 확장 제수/몫 쌍을 구하자. 식의 각 큐브에 표 1과 같이 인덱스 번호를 부여한다. 다음, 주어진 논리식에서 모든 가능한 2개의 큐브를 선택하여 제수/몫 쌍을 산출하면 표 2와 같다. 표 2에서 행 2의 제수와 몫은 $de(ab+c)$인 분해식을, 행 3의 경우는 $cd(a'+e)$ 분해식을 산출한다. 여기에, 행 2과 행 3의 제수는 모두 2개의 리터럴로 구성되어 있기 때문에 이 제수들을 분해해서 표 3과 같은 확장 제수/몫 쌍들 산출한다. 표 3의 행 1과 2는 표 2에서 행 2의 제수 de를 분해해서 확장한 제수/몫 쌍들이며, 표 3의 행 3과 4는 표 2의 행 3의 제수 cd를 분해해서 확장한 제수/몫 쌍들이다. 최종적으로 표 2의 제수/몫 쌍들과 표 3의 확장 제수/몫 쌍들을 정리한 것이 표 4이다.

<표 1> 큐브 인덱스

큐브 C_i	$abde$	$a'cd$	cde
인덱스 $index(C_i)$	1	2	3

<표 2> 제수/몫 쌍

행	선택된 큐브	제수/몫 쌍(d_i/q_i)	
		제수 (d_i)	몫(q_i)
1	$C_1 \cap C_2$	d	$abe+a'c$
2	$C_1 \cap C_3$	de	$ab+c$
3	$C_2 \cap C_3$	cd	$a'+e$

<표 3> 확장 제수/몫 쌍

행	제수/몫 쌍 (d_i/q_i)		확장 제수/몫 쌍 $(d_i/d_{i_j},\ q_i * d_{i_j})$	
	제수	몫	제수	몫
1	de	$ab+c$	d	$abe+ce$
2	de	$ab+c$	e	$abd+cd$
3	cd	$a'+e$	c	$a'd+de$
4	cd	$a'+e$	d	$a'c+ce$

<표 4> 전체 제수/몫 쌍

행	선택된 큐브	전체 제수/몫 쌍(c_i, t_i)	
		제수 (c_i)	몫(t_i)
1	$C_1 \cap C_2$	d	$abe+a'c$
2	$C_1 \cap C_3$	de	$ab+c$
3	$C_1 \cap C_3$	d	$abe+ce$
4	$C_1 \cap C_3$	e	$abd+cd$
5	$C_2 \cap C_3$	cd	$a'+e$
6	$C_2 \cap C_3$	c	$a'd+de$
7	$C_2 \cap C_3$	d	$a'c+ce$

예 4: 논리식 $F=abde+a'cd+cde$와 예 3에서 산출한 표 4로부터 몫/몫 쌍을 산출한다. 표 4의 행 2와 행 6의 제수/몫 쌍을 보면 행 2의 제수가 행 6의 몫에 포함되고, 다시 행 6의 제수가 행 2의 몫에 포함된다. 또한 행 2와 행 6의 몫들의 논리곱은 주어진 논리식 F에 포함된다. 또 행 4과 행 5에서 같은 결과를 얻는다. 이를 정리하면 표 5와 같다. 표 5는 몫/몫 쌍을 몫과 몫 곱으로 표현한 것이다.

<표 5> 몫/몫 쌍

선택된 2개 행	몫과 몫 쌍(t_i, t_j)
행2, 행6	$(ab+c)(a'd+de)$
행4, 행5	$(abd+cd)(a'+e)$

2.3 부울 분해식 산출 확장 2-큐브 행렬

제수/몫 쌍들과 몫/몫 쌍들을 이용해서 분해식 산출 행렬 M을 만든다. 행렬 M의 행은 제수/몫 쌍들의 제수들과 몫/몫 쌍에 포함되는 큐브에 대응하여 각 행을 하나씩 만든다. 행렬 M의 열은 몫을 구성하는 각 큐브 당 하나씩 만든다. 여기서, C를 행에 대응되는 큐브들의 집합, CQ를 행렬 M의 열에 해당하는 큐브들의 집합으로 표기하자. 그리고, 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.

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

예 5: 다시 $F=abde+a'cd+cde$에 대하여, 예 3에서 산출한 표 1, 표 4와 예 4에서 산출한 표 5를 이용해서 분해식 산출 행렬 M을 만든다. 행렬의 행들은 표 4의 제수에 해당하는 큐브들과 표 5의 몫/몫 쌍에 포함되는 큐브들에 대응하여 하나씩 만든다. 그러면, 행렬의 행에 대응되는 큐브 집합은 {*d, de, cd, e, c, ab, a'd, abd, a'*}가 되며, 열에 대응되는 큐브 집합은 {*abe, a'c, ab, c, a', e, ce, abd, cd, a'd, de*}가 된다. 산출된 분해식 산출 행렬은 표 6과 같다. 표 6의 행렬에서 첫 번째 행과 첫 번째 열에 해당하는 $M(1,1)=M(d,abe)$의 원소 값은 $index(abde)=1$가 된다. 또한 여섯 번째 행과 다섯 번째 열에 해당하는 $M(7,3)=M(a'd,ab)$의 원소 값은 $a'd \bullet ad=0$가 되기 때문에 $M(7,3)=*$ 가 된다. 나머지 원소들에 대해서도 같은 방법으로 행렬에 값이 할당된다. 표 6에서 진하게 표시된 부분은 [12]에서 제시한 분해식 산출 행렬에 별도로 추가된 부분이다.

<표 6> 분해식 산출 확장 2-큐브 행렬

열 / 행	*abe*	*a'c*	*ab*	*c*	*a'*	*e*	*ce*	*abd*	*cd*	*a'd*	*de*
d	1	2	0	0	0	0	3	0	0	0	0
de	0	0	1	3	0	0	0	0	0	0	0
cd	0	0	0	0	2	3	0	0	0	0	0
e	0	0	0	0	0	0	0	1	3	0	0
c	0	0	0	0	0	0	0	0	0	2	3
ab	0	0	0	0	0	0	0	0	0	*	1
a'd	0	0	*	2	0	0	0	0	0	0	0
abd	0	0	0	0	*	1	0	0	0	0	0
a'	0	0	0	0	0	0	0	*	2	0	0

2.4 분해식 산출 행렬 커버와 분해식

행렬 커버는 Brayton 등[2-4]과 Kwon 등[8]이 제시한 사각형 커버링 방법과 유사하다. 먼저 사각형에 대한 정의를 하고 커버링 방법에 대하여 설명한다.

정의 4: 행렬 M에서 **사각형**은 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 $M(\gamma,\delta)>0$이 $M(\alpha,\beta)\neq M(\gamma,\delta)$

. **프라임 사각형**은 다른 사각형에 포함되지 않는 사각형이다. 사각형(R,C)의 **사각형 비용**은 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.

예 6: 표 6와 같은 분해식 산출 행렬 M이 주어졌다고 하자. $(\{de\},\{ab\})$는 사각형이다. 그러나,$(\{de\},\{ab\})$은 $(\{de,a'd\},\{ab,c\})$에 포함되기 때문에 프라임 사각형은 아니다. 반면에, $(\{de,a'd\},\{ab,c\})$을 포함하는 사각형이 없기 때문에 프라임 사각형이라 한다. $(\{d\},\{abe,a'c,ab\})$는 $M(d,ab)=0$를 포함하기 때문에 사각형이 아니다. 사각형 $(\{de,a'd\},\{ab,c\})$의 사각형 비용은 행과 열을 나타내는 리터럴의 개수 합으로 7이 된다.

최소 비용을 갖는 사각형의 집합을 산출한 후, 사각형에 대응되는 분해식을 산

출한다. 예 6의 프라임 사각형 $(\{de,a'd\},\{ab,c\})$에 대응되는 분해식으로 $F=(de+a'd)(ab+c)$가 산출되고, $de+a'd$에 대하여 다시 분해식 산출 확장 2-큐브 행렬을 만들고 사각형 커버를 찾으면, 최종적으로 리터럴 개수가 6개인 분해식 $F=(d(e+a'))(ab+c)$가 산출된다. 반면에 SIS의 대수 분해식 방법으로는 7개의 리터럴을 갖는 분해식 $F=d(e(ab+c)+a'c)$가 산출되며, [12]의 방법을 적용할 경우 리터럴 8개의 분해식 $F=de(ab+c)+abd$가 산출된다.

3. 실험 결과

본 논문에서 제안한 방법을 PLA 형의 여러 MCNC (Microelectronics Center of North Carolina) 벤치마크 회로(benchmark circuit)에 대하여 분해식을 산출하였다. 제안한 방법의 실효성 비교를 위해 [5]와 [12]의 분해식 결과들과 비교하였다. SIS로 알려진 [5]에서 분해식 산출하기 위해서 "factor -g" 명령을 각 벤치마크 회로에 적용하였다. 참고로, "factor -g"는 가장 작은 수의 리터럴을 갖는 분해식을 산출하는 SIS 명령이다. 또, [12]는 2-큐브에서 제수와 몫을 산출하여 부울 분해식을 산출하는 방법이며, 본 논문은 [12]에서 제시한 방법을 확장한 것인 만큼 [12]에서 산출한 부울 분해식 산출 결과와도 비교하였다. 표 7은 본 논문에서 제시한 방법으로 실험한 실험결과를 [5]와 [12]의 결과와 비교한 것이다. 실험 결과는 각 회로를 최적화하는데 소요되는 시간을 초단위로 산출하고 분해식의 리터럴 개수를 구한 것이다. 실험은 Pentinum IV 1.4GHz CPU가 장착된 Linux 환경에서 실시하였으며, 본 논문에서 제시한 알고리즘은 C언어를 이용하여 프로그램 하였다. 제안하는 방법은 [5]와 [12]의 방법과 비교해서 리터럴 개수를 줄일 수 있음을 표 7을 통해 보였다. 본 논문에서 제시한 방법은 [12]의 방법을 확장한 것으로 [12]의 2-큐브 행렬을 포함하는 수퍼행렬을 산출하기 때문에 수행시간이 다소 늘어나는 단점을 갖는다. 따라서, 논리합성 목표가 최소 리터럴을 갖는 논리식을 산출하고자 하는 경우에는 본 논문에서 제안하는 방법이 유용하게 사용될 것으로 생각된다.

<표 7> 실험 결과

회로	입력수	출력수	[5] 결과		[12] 결과		제안방법	
			리터럴수	시간(초)	리터럴수	시간(초)	리터럴수	시간(초)
rd53	5	3	71	0.2	69	0.2	65	03
squar5	5	8	127	0.3	91	0.4	89	0.6
f51m	8	8	168	0.3	165	0.3	162	0.4
z4ml	7	4	69	0.1	66	0.1	64	0.3
mux	5	1	79	0.6	72	0.7	70	0.8
sct	19	15	138	0.1	137	0.1	137	0.2
tcon	17	16	48	0.1	40	0.1	38	0.3
cmb	16	4	98	0.1	82	0.2	80	0.3
inc	7	9	246	0.7	247	0.5	243	0.8
decod	16	4	98	0.1	97	0.1	82	0.2
squar5	5	8	127	0.2	124	0.1	92	0.4

4. 결론

본 논문은 커널 산출 없이 부울 분해식

을 산출할 수 방법을 제시하였다. 특히, 주어진 단순 식에서 2개의 큐브를 선택하고 제수와 몫의 쌍을 구하는 과정에서, 몫에 공통인수를 남겨두어 나중에 몫과 몫 쌍을 산출할 수 있는 가능성을 높였다. 이렇게 산출된 제수와 몫 쌍들과 몫과 몫 쌍들은 확장된 2-큐브 행렬을 산출하게 되며, 이 행렬을 커버하는 사각형을 산출하게 되면 리터럴 개수가 상대적으로 작은 부울 분해식을 산출하게 된다.

※ 참고문헌

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

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

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

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

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

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

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

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

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

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

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

[12] 권오형.(2006). 2개의 곱항에서 공통인수를 이용한 논리 분해식 산출. *한국컴퓨터산업교육학회논문지*, 7(4), pp. 293-298.