論文
7 - 5 - 3

다변수 출력 함수에서 공통 논리식 추출 (A Boolean Logic Extraction for Multiple-level Logic Optimization)

권 오 형(Oh-Hyeong Kwon)[1)]

요 약

본 논문에서는 여러 개의 출력단을 갖는 논리회로에서 공통식을 찾는 방법을 제안하였다. 각각의 출력단위로 2개의 큐브로 구성된 묶을 찾고, 이 묶들 간의 쌍을 이용해서 부울 공통식을 찾는 방법을 보였다. 실험 결과로 2개의 큐브만을 이용한 공통식 산출만으로 전체 논리회로의 크기를 줄이는 데 효과가 있음을 SIS1.2 결과와 비교하여 보였다.

ABSTRACT

Extraction is the most important step in global minimization. Its approache is to identify and extract subexpressions, which are multiple-cubes or single-cubes, common to two or more expressions which can be used to reduce the total number of literals in a Boolean network. Extraction is described as either algebraic or Boolean, according to the trade-off between run-time and optimization. Boolean extraction is capable of providing better results, but difficulty in finding common Boolean divisors arises. In this paper, we present a new method for Boolean extraction to remove the difficulty. The key idea is to identify and extract two-cube Boolean subexpression pairs from each expression in a Boolean network. Experimental results show the improvements in the literal counts over the extraction in SIS for some benchmark circuits.

논문접수 : 2006. 9. 10.
심사완료 : 2006. 10. 1.

1) 정회원 : 한서대학교 인터넷공학과 교수

1. 서론

주어진 논리함수는 다양한 논리식으로 표현이 가능하다. 특히 다단논리회로를 표현하기 위해서는 AND와 OR 논리 연산자만을 이용하고 리터럴 개수를 최소화한 논리식이 많이 사용된다. 다단 논리식은 바로 MOS 회로로 전환이 가능하며 리터럴의 수가 MOS 회로에 사용되는 트랜지스터의 개수에 비례하는 특징을 갖는다. 따라서, 다단 논리식에 대한 최적화 도구는 가장 적은 개수의 리터럴을 갖는 논리식 산출을 목표로 한다. 다단논리회로(multiple-level logic circuit) 합성 설계 방법에 있어서 최적화는 국부적 최적화(local optimization)와 전역적 최적화(global optimization)로 개발되어 왔다. 국부적 최적화는 전체 부울 네트워크(Boolean network)의 형태에 영향을 주지 않고 단지 국부적인 논리식 최적화를 목적으로 한다. 반면에, 전역적 최적화는 주어진 부울 네트워크의 변형까지 고려하면서 최적화를 수행하는 것을 목적으로 한다. 전역적 최적화 방법 중의 하나인 공통식 추출(extraction) 기법은 주어진 여러 논리식(multiple-output Boolean expression)들을 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.

Brayton과 McMullen은 커널(kernel)을 이용한 다항 큐브 제수를 찾는 체계적인 방법[1]을 제시하였고, 후에 Brayton, Rudell, Sangiovanni-Vincentelli와 Wang은 커널의 부분집합(subset)을 이용한 다항 큐브 제수를 찾는 알고리즘[2]을 제시하였다. 이들의 알고리즘은 레벨-0에 대한 커널만을 산출하고, 이 레벨-0 커널들의 교집합을 산출하는 방법에 기반을 두고 있다. 이렇게 제시된 커널 기반 기술[1,2,3]에 의한 다항 큐브 제수 산출 방법은 대수적인 방법으로 공통식을 찾는 것인 데 수행속도는 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 변수치환을 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[7,8,9]. 이러한 최근의 결과는 단지 일부 회로에서 대수적 나눗셈을 이용한 방법[1,2,3]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 [10,11]에서 제시한 방법을 다변수 출력 논리회로(multiple-output logic circuit)로 확장한 것이다. 제안하는 핵심 기술

은 주어진 논리식들에서 2개의 큐브로 구성된 논리식 쌍(pair)을 산출하는 것으로 되어 있다. 일반적으로 실제 상황에서 부울식의 크기-항의 개수 또는 리터럴 개수-가 매우 크기 때문에 2개의 큐브로 표현되는 식만을 찾도록 제한하였다. 2개의 큐브로 표현된 논리식 쌍에 대한 공통식 추출을 하고, 다음 공통 단항 큐브를 찾는 과정으로 최적화를 진행한다.

논문의 구성은 다음과 같다. 2절에서는 본 논문 서술에 필요한 정의와 커널 기반 공통식 산출방법[2,3]에 대하여 서술하고, 3절에서 본 논문에서 제시하는 새로운 공통식 산출 방법을 소개한다. 4절에서 실험결과를 보이고, 5절에서 결론을 제시한다.

2. 배경 지식

본 논문의 공통식 산출 방법을 서술하는 데 필요한 용어들과 본 논문의 비교 대상이 되는 커널 기반 공통식 산출 방법에 대하여 서술한다.

2.1 정의

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. 논리식을 구성하는 큐브들 간에 서로 포함 관계가 성립하지 않는 경우 논리식은 불필요한 항을 갖고 있지 않다고 한다.

정의 2: 논리식 F의 서포트(support)는 논리식 F를 구성하는 변수들 집합으로 $sup(F)$로 표현한다. 논리식을 구성하는 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: 논리함수 $F=a+bc'$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 $F=bc'd'e+a(b'c+b'e+c'd')$으로 표현될 수 있으며, 이 때 $b'c+b'e+c'd'$은 코커널 a에 대한 커널이 된다.

정의 3: 부울 네트워크는 directed acyclic graph (DAG)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 대수곱이고, 그 외의 경우 FG는 부울곱이다. F/G는 가장 큰 큐브 집합인 몫 Q를 산출하여, 논리식 F를 $F=QG+R$로 표현한다. 여기서, R은 나머지를 나타낸다. QG가 대수곱인 경우, F/G는 대수 몫을, 그 외의 경우 F/G는 부울 몫이 된다. $F/G=Q\neq\phi$ 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 대수곱이며, $(a+b)(a+c)=a+ab+ac+bc$는 부울곱이다. $F=ad+abc+bcd$ 이고 $G=a+bc$ 가 주어진 경우를 보자. F/G의 결과로는 대수 몫 d와 나머지 abc가 된다. 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.

2.2 커널 기반 공통식 산출

Brayton과 McMullen은 커널 집합과 다항 공통 논리식 관계에 대한 정리로부터 다음의 2가지 결론을 제시하였다[1]. 첫째로, 2개의 논리식으로부터 산출된 커널들의 교집합이 공집합 또는 단항 큐브인 경우는 논리식들이 다항의 공통식을 갖지 않는다는 결론을 제시하였다. 둘째로, 커널들의 교집합이 다항 큐브인 경우는 제수로 사용할 수 있다는 점을 제시하였다.

Brayton과 McMullen이 그들이 제시한 결과를 적용하여 다변수 출력 논리회로 또는 논리식들에서 공통식을 산출하는 과정은 2단계로 구분하여 구현하였다. 부울 네트워크에서 논리식들의 커널 집합 산출이 첫 번째 단계가 된다. 두 번째 단계가 커널 교집합으로부터 최적의 결과를 산출하도록 공통식을 선별하는 과정이다. 때때로, 커널을 산출하는 데 커널의 개수가 너무 많이 산출되어 장시간의 수행 시간이 소요될 수 있기 때문에 SIS(또는 MIS)의 경우 커널 산출에 레벨-0까지만 산출하는 선택 사양을 제공하고 있다[2].

예 4: 다음의 2개 논리식 F_0와 F_1이 주어졌다고 하자. $F_0=ace+bce+de+g$, $F_1=ad+bd+cde+ge$. 이 때, F_0의 커널 집합 $K(F_0)$은 $K(F_0)=\{(a+b),\ (ac+bc+d),\ (ace+bce+de+g)\}$. 또, F_1의 커널 집합은 $K(F_1)=\{(a+b+ce),\ (cd+g),\ (ad+bd+cde+ge)\}$. 그러면, F_0와 F_1로부터 다항 큐브인 공통식을 추출할 수 있다. 즉, $(a+b)\in K(F_0)$와 $(a+b+ce)\in K(F_1)$ 사이에 커널 교집합으로 $a+b$를 얻게 된다. 이로부터 주어진 원 논리식은 다음과 같이 변형된다.

$$F_0=wce+de+g$$
$$F_1=wd+cde+ge$$
$$F_w=a+b.$$

3. 부울 공통식 산출

부울 공통식 산출에는 2개의 큐브로 구성된 논리식 쌍을 이용한다. 본 절에서는 2개의 큐브로 구성된 논리식 쌍을 찾는 방법을 제안하고, 산출한 논리식 쌍으로부터 부울 공통식을 산출(Boolean extraction)하는 예를 제시한다.

3.1 2개의 큐브로 구성된 논리식 쌍

2개의 큐브로 구성된 부울식 쌍은 2개의 제수/몫 쌍으로부터 산출된다. 주어진 논리식에서 2개의 큐브를 선택하여 이 2개의 큐브에 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 $q_i q_j$ 가 주어진 논리식에 포함되면, (q_i, q_j)는 2개의 큐브로 구성된 부울식이라 한다.

예 5: 다음의 3 가지 논리식이 주어졌다고 하자. $F_0 = wx'y + wxz + yz$

$F_1 = wx'y + xz + yz$

$F_2 = wxy + xz + yz$

그러면, 제수/몫 쌍은 다음과 같이 산출된다. 논리식 F_0로부터 $\{(w, x'y+xz), (y, wx'+z), (z, wx+y)\}$. F_1으로부터 $\{(y, wx'+z), (z, x+y)\}$. F_2로부터 $\{(x, wy+z), (y, wx+z), (z, x+y)\}$. F_0의 제수/몫 쌍인 $(y, wx'+z)$와 $(z, wx+y)$를 보자. 전자의 제수/몫에서 몫의 z는 후자의 제수/몫의 제수와 동일하고, 후자의 제수/몫에서 몫의 y는 전자의 제수/몫의 제수와 동일하다. 그리고, 전자와 후자의 몫들의 곱 $(wx'+z)(wx+y)$는 F_0에 속한다. 따라서, $(wx'+z)(wx+y)$는 F_0에 속하는 2개의 큐브로 구성된 부울식 쌍이 된다. 동일한 방법으로 F_1의 $(y, wx'+z)$와 $(z, x+y)$로부터 $(wx'+z)(x+y)$를 얻게 된다.

3.2 2-큐브 행렬

다수 개의 출력을 갖는 논리식이 주어진 경우, 제수들의 집합 C와 2개의 큐브로 구성된 몫들의 집합 Q를 3.1절에서 서술한 바와 같이 산출할 수 있다. 그러면 산출된 집합 Q의 원소 q 에 0보다 큰 양의 정수 값을 배정하며, 이 때 값을 배정하는 함수를 $index(q)$로 표기한다.

공통식을 산출하기 위해서 2개의 큐브로 구성된 몫으로 2-큐브 행렬 T를 만든다. 행렬 T의 행들은 각 논리함수의 이름이며, 열은 2개의 큐브로 구성된 몫이다. 이 때, 행이 F_k이고 열이 q_i일 경우 행렬의 원소 $T(F_i, q_i)$에 대한 정의는 다음과 같다.

$$T(F_k, q_i) = \begin{cases} c_i & \text{if } (c_i, q_i) \in F_k, c_i \in C, \\ & \text{and } q_i \in Q \\ c_i, index(q_j) & \text{if } (c_i, q_i) \in F_k, (q_i, q_j) \in F_k, \\ & q_i \in Q, \text{ and } q_j \in Q \end{cases}$$

예 6: 예5의 3개의 논리함수에 대하여 2-큐브 행렬을 산출하는 예를 보인다. 먼저 표1에 보인 것처럼 Q의 각 원소에 양의 정수를 배당한다. 그리고, 표 2에 2-큐브 행렬의 예를 보였다. 논리함수 F_0에서 제수/몫은 $(w, x'y+xz)$이기 때문에 행렬에서 F_0를 나타내는 첫 번째 행과 몫 $x'y+xz$을 나타내는 첫 번째 열에 해당하는 행렬의 원소에는 w가 입력된다. 또, 예 5에서 F_0로부터 2개의 큐브로 구성된 논리식 쌍 $(wx'+z, wx+y)$를 산출하였고, 표 1에서 $index(wx+y)=3$으로 값을 배정하였기 때문에, $T(F_0, wx'+z)=y,3$이 된다. 그 외의 행렬의 원소 내용도 같은 방법으로 원소의 내용이 결정된다.

<표 1> 2-큐브 몫에 대한 *index* 배정

<Table 1> Indexes of Two-cube Quotients

2-큐브 몫 q	x'y + xz	wx' + z	wx + y	x + y	wy + z	wx + z
index(q)	1	2	3	4	5	6

<표 2> 2-큐브 행렬 T

<Table 2> Two-cube Array T

F \ q	x'y + xz	wx' + z	wx + y	x + y	wy + z	wx + z
F_0	w	y,3	z,2			
F_1		y,4		z,2		
F_2				z	x	y

3.4 가중치 계산

주어진 논리식들을 가장 적은 개수의 리터럴을 갖는 최적 논리식들로 전환하기 위해서 가능한 모든 공통식들을 찾고, 이 중에서 최적의 결과를 산출하게 될 공통식을 선택하여야 한다. 본 논문에서는 최적 결과를 산출할 수 있는 공통식을 선택하기 위해 가중치 부여 방법을 사용하였다. 즉, 2개의 큐브로 구성된 부울식 쌍에 다음과 같은 방법으로 가중치를 부가하도록 고안하였다.

$$weight(k) = \sum_{i=0}^{1} \{(NF(k_i)-1)(L(k_i)-1)-1\}$$

여기서, $k = k_0 k_1$이고 k_0와 k_1은 2개의 큐브로 구성된 부울식이다. $NF(k_i)$는 논리식 k_i를 포함하는 논리식들의 개수이고, $L(k_i)$는 논리식 k_i의 리터럴 개수를 표현한 것이다.

3.5 알고리즘

2개의 큐브로 구성된 부울식 쌍들을 찾고, 이로부터 논리식들(또는 출력함수)에서 부울 공통식을 산출하여 원 논리식들을 최적의 논리식들로 전환하는 알고리즘을 제안한다.

Algorithm : Boolean extraction.
Input : Set F of Boolean expressions.
Output : Set F of Boolean expressions.
Method :

```
begin
  for each Boolean expression in F do
  begin
  generate algebraic divisor/quotient pairs S;
   find two-cube Boolean pairs P from S;
   end
   for i=1 to |P| do
   begin
      x = argmax_{k∈P} { weight(k) };
     if ( weight(k) ≤ 0) then break;
  /* x is the multiplication of x0 and x1. */
      for each expression f∈F
        for which x0 may be a divisor do
      begin
         substitute f with x0;
      end
      F = F ∪ {x0};
      for each expression f∈F
        for which x1 may be a divisor do
      begin
         substitute f with x1;
      end
      F = F ∪ {x1};
   end
   extract single-cube divisors for F;
end
```

알고리즘은 논리식들 집합 F 를 입력으로 하여, 2개의 큐브로 구성된 모든 제수/몫 쌍을 산출한다. 그리고, 각 제수/몫 쌍에 가중치를 부여하고 가장 큰 값을 갖는 제수/몫 쌍을 선택하여 새로운 변수를 할당한다. 이 제수/몫 쌍을 포함하는 주어진 논리식을 새로운 변수로 대치하는 과정으로 알고리즘은 구성되어 있다. 마지막으로, 단일 항으로 구성된 공통식을 찾고 공통식을 포함하는 주어진 논리식 중에서 공통식을 새로운 변수로 대치한다.

예 7: 예 2의 논리식들에 대하여 위 알고리즘을 적용한다. 첫 번째 for-loop에서 F_0로부터 $(wx'+z)(wx+y)$, F_1으로부터 $(wx'+z)(x+y)$를 얻게 된다. 가중치를 부여한 결과 F_1의 $(wx'+z)(x+y)$가 가장 큰 가중치를 갖기 때문에 $(wx'+z)(x+y)$를 선택한다. 여기서, $wx'+z$는 F_0와 F_1에서 공통으로 사용되기 때문에, F_0와 F_1에서 $wx'+z$를 새

로운 변수를 도입하여 대치한다. 또한, F_1의 $x+y$는 F_1와 F_2에서 새로운 변수를 도입하여 대치할 수 있다. 따라서, 알고리즘은 새로운 변수를 도입한 다음의 논리식 집합을 산출한다.

$$F_0 = t\,(wx+y)$$
$$F_1 = t\,s$$
$$F_2 = wxy+zs$$
$$F_t = wx'+z$$
$$F_s = x+y$$

알고리즘은 16개의 리터럴을 갖고 있는 논리식들이 산출되었다. 그러나, 대수적인 방법에 의한 공통식을 추출할 경우 전체 17개의 리터럴을 갖는 다음의 논리식들을 산출하게 된다.

$$F_0 = wxz+t$$
$$F_1 = xz+t$$
$$F_2 = x(wy+z)+yz$$
$$F_t = (wx'+z)y$$

4. 실험 결과

제시한 알고리즘은 Pentium IV 1.4GHz CPU PC의 Linux 환경에서 실험하였다. 실험 결과를 비교하기 위해서, 대부분의 연구자들이 비교 대상으로 사용하는 커널 기반 대수 공통식 산출에 의한 다출력 최적화 도구인 SIS1.2 (또는 MIS)와 수행한 결과를 리터럴 수와 수행을 대상으로 비교하였다. SIS1.2로 공통식을 추출하기 위해서 SIS1.2 명령어 "gkx -b"를 먼저 수행하고, 다음 "gcx -b"를 수행한 후 최적 논리식들의 리터럴 개수를 산출하도록 하였다. SIS1.2의 "gkx"는 다항 공통식을 산출하여 최적화를 수행하는 명령이고, "gcx"는 단항 공통식을 찾아 최적화를 수행하는 명령이다. 실험 결과를 표 3에 정리하였다. 표 3의 첫 번째 열은 벤치마크 회로의 이름이고, 다음 2개의 열은 벤치마크 회로의 입력 수와 출력 수를 나타낸 것이다. 다음 열은 SIS1.2가 수행하고 산출한 리터럴 개수를 나타낸 것이고, 마지막 열은 본 논문에서 제안한 방법으로 벤치마크 회로에 대한 최적화 작업 후 얻은 리터럴 개수를 정리한 것이다. 표 3으로부터 SIS1.2는 대칭 논리식(symmetric expression)인 rd53과 rd73에서 제안한 방법보다 적은 수의 리터럴을 갖는 회로로 합성되었다. 그러나, 나머지 벤치마크 회로에 대해서는 제안한 방법이 SIS1.2 보다 적은 수의 리터럴을 갖는 회로로 합성되었다.

<표 3> 실험 결과
<Table 3> Experimental Results

회로	입력 수	출력 수	SIS1.2		제안방법	
			리터럴 수	시간 (초)	리터럴 수	시간 (초)
Example1	4	3	17	0.1	16	0.1
apex1	45	45	2627	2.0	1101	1.9
rd53	5	3	71	0.2	73	0.1
rd73	7	3	164	0.4	169	0.4
con1	7	2	23	0.1	23	0.1
z4ml	7	4	70	0.2	61	0.3
cmb	16	4	70	0.1	57	0.1
C17	5	2	10	0.1	10	0.1
vg2	25	8	105	0.1	102	0.1
decod	5	16	52	0.1	51	0.1

5. 결론

논리회로의 출력수가 2개 이상의 경우 최적화된 논리회로를 얻기 위해서는 각 출력에 동일하게 사용되는 공통회로를 찾아 각 출력에 반복 사용되지 못하도록 수정하는 것이 중요한 일이다. 이러한 작업에 부울 공리인 등멱법칙과 보수법칙까지 고려하고 공통 논리식을 산출하면 장시간의 수행 시간이 요구되었다. 따라서, 단지 SIS1.2처럼 대수적인 방법으로 전체 회로를 간략화 작업을 수행하나, 항상 최적의 결과를 얻을 수 없게 된다. 이러한 문제를 해결하기 위해서, 본 논문에서는 2개의 큐브로만

구성된 부울식 쌍을 이용하여 수행시간을 줄이고, 등멱법칙과 보수법칙을 고려하여 전체 리터럴 개수를 최대로 줄일 수 있는 공통식을 찾도록 하였다. 대수적인 방법과 비교하여 실험한 벤치마크 회로의 대부분에서 리터럴 개수를 줄일 수 있었다. 실험결과로부터 제안한 방법을 SIS의 보조 수단으로 활용하면 보다 리터럴 개수를 줄일 수 있을 것이다.

참고문헌

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

[2] 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.

[3] 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.

[4] J. Rajski and J. Vasudevamurthy. The testability-preserving concurrent decomposition and factorization of Boolean expressions(1992). *IEEE Trans. CAD*, 11(6), pp. 778-79.

[5] V. K. Singh and A. A. Diwan(1993). A heuristic for decomposition in multilevel logic optimization. *IEEE Trans. VLSI*, 1(4), pp. 441-445.

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

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

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

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

[10] 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.

[11] 권오형(2006), 2개의 곱합에서 공통인수를 이용한 논리 분해식 산출. *한국컴퓨터산업교육학회 논문지*, 7(4).