해밍거리가 3인 큐브를 활용한 공통식 추출

권오형†

요 약

논리회로 심화학습에 사용할 수 있는 논리식 간략화 도구로 활용하고 더 나아가 반도체 부품 최적화를 위한 설계자동화 도구로 활용할 수 있는 도구를 제안한 것이다. 본 논문에서 제시하는 논리식 간략화 방법은 여러 논리식에 존재하는 공통부분을 찾아 반복 사용을 줄이는 것이다. 최종적으로 전체 논리식에 사용된 리터럴 개수를 최소화하는 것을 목표로 한다. 이 전의 연구들이 나눗셈 원리를 이용해서 공통식을 찾았기 때문에 논리식에 내재한 공통식을 산출하는 데는 실패하였다. 본 논문에서 제안하는 방법은 논리식들 사이에 내재된 공통식을 찾도록 해밍거리가 3인 큐브들을 이용하였다. 벤치마크 회로를 이용한 실험을 통해 타 방법들과 간략화 정도를 비교했을 때, 제안한 방법으로 최대 47% 정도의 리터럴 개수를 줄이는 효과를 보였다.

주제어 : 논리 합성, 공통식 추출, 해밍거리, 부울대수

Common Logic Extraction Using Hamming Distance 3 Cubes

Oh-Hyeong Kwon†

ABSTRACT

This paper proposes a tool that can be used as a logical expression simplification tool that can be used for deepening learning of logic circuits and further utilized as a design automation tool for optimizing semiconductor parts. The simplification method of logical expressions proposed in this paper is to find common subexpressions existing in various logical expressions and reduce the repetitive use. Finally, the goal is to minimize the number of literals used in all logical expressions. These previous studies failed to produce a common subexpression embedded in the logical expressions because they only use division principle. The proposed method uses cubes with a Hamming distance of 3 to find the common subexpression embedded between logical expressions. Experiments using benchmark circuits show that the proposed method reduces the number of literals by as much as 47% when comparing simplifications with other methods.

Keywords : Logic synthesis, Extraction, Hamming distance, Boolean Algebra

† 정 회 원: 한서대학교 항공컴퓨터전공 교수(주, 교신저자)
논문접수: 2017년 6월 19일, 심사완료: 2017년 7월 14일, 게재확정: 2017년 7월 28일

1. 서론

대학에서 논리회로에 대한 기초 학습은 카르노맵(Karnaugh Map)을 이용한 논리식 간략화에 초점을 맞추고 있다. 그러나 카르노맵을 이용한 방법은 입력 변수가 6개 이하인 경우 진리표를 입력으로 2단의 단순식을 수작업에 의해 산출하게 고안된 것이다. 본 논문은 논리회로 심화학습 과정에서 논리회로 간략화를 자동화하는 도구로 활용하고, 더 나아가 반도체 부품 개발 도구로 활용하기 위한 실용적 방법을 제안한 것이다.

간략화된 논리식을 산출하는 단계가 회로 설계 자동화 분야의 핵심 중의 하나로, 하드웨어 표현 언어(Hardware Description Language)로 작성된 동작 행위(behavior level)를 실제 회로로 변환하기 위한 중간 역할을 한다. 산출된 논리식이 최종 회로나 칩의 크기에 많은 영향을 미친다. 따라서 많은 연구자들이 최적 논리식 또는 논리회로를 산출하기 위한 논리합성 도구를 만들기 위해 연구를 진행하고 있다. 특히 여러 개의 출력들이 있는 논리회로를 설계하고자 할 경우, 여러 출력에서 공통으로 사용된 논리식을 한 번만 사용해서 전체 논리식을 간략화하는 것은 회로 간략화에 매우 중요한 과정이다. 이러한 과정을 공통식 추출(extraction)이라고 부른다. 더욱이 다단 논리회로가 2단 논리회로보다 더 간략화 될 수 있기 때문에 공통식 추출을 통해 간략화된 다단 논리회로를 산출하기 위한 연구들이 진행되고 있다.

본 논문은 선험 방법(heuristic method)으로 공통식을 찾아 논리식을 간략화하는 방법을 제안한 것이다. 논리식 또는 출력 별로 해밍거리가 3인 큐브들을 선택해서 공통식이 될 수 있는 후보군을 늘리는 것이 본 논문의 핵심이다. 각 논리식 별로 후보군이 많으면 공통식을 찾을 가능성이 높기 때문이다. 본 논문에서 제안하는 논리식 간략화의 목표는 AND, OR, NOT 연산자만을 사용하고 전체 논리식에 사용된 리터럴 개수를 최소화하는 것이다. 논리식을 CMOS 트랜지스터 회로로 전환했을 때 트랜지스터의 개수는 논리식에 사용된 리터럴 개수의 2배가 되기 때문에 리터럴 개수를 줄이는데 목표를 둔다.

논문의 구성은 다음과 같다. 2장에서는 본 논문 서술에 필요한 배경 지식인 정의와 관련 연구를 소개하고, 또한 커널을 이용한 공통식 추출 방법에 대하여 서술한다. 3장에서 본 논문에서 제시하는 해밍거리 3인 큐브들을 이용해서 공통식 추출 방안을 제시한다. 4장에서 실험결과를 보이고, 5장에서 결론을 제시한다.

2. 배경 지식

본 논문을 서술하는데 필요한 용어들과 관련 연구들에 대하여 기술한다. 관련 연구 중에서 대수 나눗셈 방법과 커널 방법에 의한 공통식 추출 방법은 본 연구 결과와 비교를 할 것이기 때문에 보다 별도의 절로 구분해서 자세히 서술한다. 정의와 예들은 [7][8]에서 소개한 내용들을 인용한 것이다.

2.1 용어 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 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 리터럴이 없으면 논리식이 큐브면제(cube-free) 되었다고 한다. 논리식이 어떤 큐브로부터 나누어졌을 때, 몫이 큐브면제라면 그 몫

을 커널이라 한다. 이 때 커널을 산출한 큐브를 코커널(co-kernel)이라 한다.

예 2: 논리식 $F=a+bc'$에 대하여 서포트 집합은 $sup(F)=\{a, b, c\}$가 된다. 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: 서포트 집합이 동일한 2개의 C_i와 C_j 큐브 사이에 해밍거리(Hamming Distance)는 2개의 큐브에서 차이가 나는 리터럴의 개수가 된다. 큐브 C_i와 C_j에 대한 해밍거리는 $HD(C_i,C_j)$로 표현한다.

예 3: $C_1=abcd$, $C_2=ab'c'd'$인 경우 $HD(C_1,C_2)=3$이 된다.

2.2 관련 연구

간략화된 다단 논리식을 산출하기 위해 Brayton과 McMullen은 커널(kernel)을 이용한 공통 다항 큐브 제수를 찾는 방법[1]을 제시하였다. 그 후, 커널을 이용한 논리합성 도구인 MIS[2]가 Brayton, Rudell, Sangiovanni-Vincentelli, Wang에 의해 발표되었다. MIS 도구에 순차회로 합성 기능을 추가한 SIS[3]가 Sentovich 등에 의해 발표되었다. Rajski와 Vasudevamurthy는 여러 출력을 갖는 논리회로에서 단지 2개의 리터럴만으로 된 단항 큐브와 그의 보수를 이용해서 공통식을 추출하는 방법[4]을 제안하였다. 1986년에 Bryant에 의해 Binary-Decision Diagram(BDD) 알고리즘이 발표되면서 BDD를 이용해서 논리식을 간략화하기 위한 연구들이 시작되었다. 이 중에서 Yang과 Ciesielski는 XOR 논리연산을 이용한 논리식 최적화 방안인 BDS[5]를 발표하였는데, 이 때 이들이 BDD를 활용하였다. Wu와 Zhu는 BDS를 개선한 Folded BDD(FBDD)[6]라는 논리합성 방법을 발표하였다. 최근에는 2-큐브 논리 항들에 부울 공리를 적용해서 공통식을 찾는 방법[7]과 단일 논리식에 부울 공리를 적용해서 인수분해를 통한 논리식을 간략화는 방법[8]이 제시되었다. 이상의 방법들이 선험 방식에 기반을 두고 공통식을 산출하기 위한 노력을 기울였으나 여전히 논리식 내에 내재한 완벽한 공통식을 산출하지는 못하는 형편이다. 한편, 논리합성 도구로부터 산출된 결과를 원하는 회로로 대응(technology mapping)시키는 연구가 진행 되고 있다[9]. Amaru 등은 3개 입력을 갖는 majority 함수와 보수를 이용한 논리식 산출과 FPGA로의 대응 방법을 제시하였다[10]. Kagaris는 논리식을 간략화된 CMOS 트랜지스터 회로로 자동 변환되는 방법[11]을 제안하였다. 이들의 방법은 논리식 간략화보다는 이미 산출된 논리식을 회로로 대응시키지 위한 방법을 중점적으로 다루고 있다.

2.3 커널을 이용한 공통식 추출

대수 나눗셈을 이용한 공통식 산출 방법 중의 하나가 커널을 이용한 것이다. 커널을 이용한 방법이 널리 알려져 있고, 본 논문에서 제안하는 방법과 비교할 것이 때문에 본 절에서 자세히 소개한다.

여러 개의 논리식들이 주어졌을 때, 커널을 이용한 공통식 추출 방법은 3단계로 구성된다. 첫 단계는 각 논리식 별로 커널 집합을 산출하는 것이다. 다음, 두 번째 단계가 이렇게 산출된 커널 집합에서 교집합을 산출하여 공통식을 산출하는 것이다. 세 번째 단계로, 전체 리터럴 개수를 줄일 수 있는 공통식이 발견되면 새로운 변수를 도입하고 공통식을 포함하는 부분은 새로운 변수로 대치하여 전체 논리식들을 간략화 한다.

예 4: 다음 논리식 F_0 와 F_1 에 대하여 커널을 이용해서 공통식을 추출하고, 전체 논리식을 간략화 하자.

$$F_0=ace+bce+de+g$$
$$F_1=ad+bd+cde+ge$$

첫 번째 단계로 각 논리식에 대한 커널 집합을 산출한다. F_0의 커널 집합 $K(F_0)$ 은 $K(F_0)=\{(a+b),\ (ac+bc+d),\ (ace+bce+de+g)\}$. F_1

의 커널 집합은 $K(F_1) = \{(a+b+ce),\ (cd+g),\ (ad+bd+cde+ge)\}$. 두 번째 단계로, $(a+b) \in K(F_0)$와 $(a+b+ce) \in K(F_1)$ 사이에 커널 교집합으로 $a+b$ 를 얻게 된다. 다음, 세 번째 단계에서 논리식$a+b$ 를 W로 표현하고, F_0 와 F_1 의 $a+b$ 부분을 W로 치환한다. 그러면, F_0 와 F_1 는 다음과 같이 리터럴 개수가 줄어든 논리식들로 변형된다.

$$F_0 = Wce+de+g$$
$$F_1 = Wd+cde+ge$$
$$W = a+b.$$

3. 해밍거리 3인 큐브들에 의한 공통식 추출

해밍거리 3인 큐브들로부터 여러 출력에서 사용되는 공통식이 될 수 있는 후보식들을 찾는 방법과 이를 이용해서 간략화된 논리식을 산출하는 알고리즘에 대하여 제시한다.

3.1 해밍거리 3인 큐브 산출

본 논문에서 해밍거리가 3인 큐브들를 공통식 산출에 선정한 이유에 대해 먼저 설명하고 다음 해밍거리가 3인 큐브로부터 공통식을 산출하는 방법에 대하여 서술한다.

만일 2개의 큐브 사이에 해밍거리가 1인 경우 즉, $F = ab + ab'$의 경우 $HD(ab, ab') = 1$이고 부울공리를 적용하면 $F = ab + ab' = a$가 된다. 1개의 큐브로 간략화되는 장점이 있으나, 큐브 수가 줄어들기 때문에 다수 개의 논리식들에서 공통으로 사용되는 큐브들인 공통식 산출에 필요한 후보군 수가 줄어드는 단점이 있다. 해밍거리가 2인 경우는 배타논리합(Exclusive OR)으로 논리식이 표현된다. 즉 ab'와 $a'b$의 경우 $HD(ab', a'b) = 2$이고 논리식은 $a \oplus b$로 표현된다. 따라서 논리식에 사용된 리터럴 개수를 줄일 수 있으나, 배타논리합이라는 연산자 또는 논리게이트를 이용해야 하는 단점이 있다. 반면에 2개의 큐브 사이가 해밍거리가 4 이상이 되기 위해서는 큐브들이 적어도 4개의 변수로 구성되어야 한다. 따라서 해밍거리가 4이상의 경우만을 고려한 경우, 변수가 3이하의 큐브들로부터 산출될 수 있는 공통식을 찾을 수 없는 경우가 발생한다. 반면에 임의의 해밍거리 n의 큐브들로부터 후보식들을 찾고, 다시 해밍거리 $n-1$의 큐브들로부터 후보식을 찾는 방법으로 해밍거리를 줄여가면서 후보식들을 찾는 경우에는 수행시간이 늘어나게 된다. 따라서 본 논문은 큐브들 사이에 해밍거리가 3인 경우로 한정하고 공통식을 산출하는 방법을 제안한다.

큐브 2개의 해밍거리가 3이고 서포트 집합이 동일한 경우 이 2개의 큐브를 분해해서 새로운 논리식들을 산출하는 방법에 대하여 기술한다. 3개의 리터럴로 구성된 2개의 큐브 M과 N 이 각각 $M = m_1m_2m_3$, $N = n_1n_2n_3$ 으로 표현되고 $sup(M) = sup(N)$ 이라고 하자. 그러면 M과 N은 식(1)과 같이 표현될 수 있다. 이 때, $m_km_l+n_pn_r$, c_1c_2는 공통식을 추출하기 위한 후보식이라고 본 논문에서 부른다.

$$M+N = (m_km_l+n_pn_r)(c_1c_2)' \quad (1)$$
여기서, $k,l,p,r \in \{1,2,3\}$, $k \neq l$, $p \neq r$
$c_1' \in M$, $c_2' \in N$ 또는 $c_1' \in N$, $c_2' \in M$

예 5: $F = a'bc' + ab'c$ 에 대하여, $M = a'bc'$, $N = ab'c$로 하자. 그러면 $HD(M, N) = 3$ 이고 $sup(M) = sup(N)$이 된다. 그러면 논리식 F에 식(1)을 적용할 수 있게 되며, F는 다음과 같은 3 종류의 논리식들로 변형될 수 있다.

$$F = (bc' + ac)(ab)' \text{ 또는}$$
$$F = (a'b + b'c)(ac)' \text{ 또는}$$
$$F = (a'b + ac)(bc)'$$

여기서, $bc' + ac$, $a'b + b'c$, $a'b + ac$, ab, ac, bc가 공통식 추출에 사용되는 후보식들이 된다.

식(1)과 예 5에서 보인 바와 같이 산출된 논리식들이 후보식이 되며, 이 후보식들을 사용해서 여러 논리식 또는 출력에서 동시에 사용된 공통식을 추출한다. 즉, 식(1)을 적용함으로써 공통식이 될 수 있는 후보군이 많아진다는 것이다. 예5에서 m_km_l, n_pn_r, c_1c_2가 다양한 큐브로 표현됨을 보였는데, 이는 공통식을 찾을 가능성이 높음을 의미한다.

3.2 공통식 추출 알고리즘

주어진 논리식들에 식(1)을 적용하면 예 5와 예 6처럼 1개 또는 2개의 큐브(또는 항)로 구성된 논리식으로 변형이 될 수 있다. 이들 중에서 전체 논리식의 리터럴 개수를 가장 많이 줄일 수 있는 공통식을 선택하는 것이 중요하다. 본 절에서는 3개 이상 큐브로 구성된 후보식 산출 방법, 후보식들에서 공통식을 추출하는 방법, 그리고, 여러 공통식 중에서 리터럴 개수를 가장 많이 줄일 수 있는 공통식을 선택하는 방법에 대하여 기술한다.

식(1)은 단지 2개 또는 1개의 큐브로 구성된 후보식들만을 산출한다. 따라서 3개 이상의 큐브로 구성된 후보식을 산출하기 위해서 이미 산출한 후보식에서 $(c_1c_2)'$가 동일할 경우 나머지 식들의 논리합을 구한다. 예로, 식(1)에 의해 $(bc'+ac)(ab)'$와 $(bd'+ad)(ab)'$가 산출되었다면 $(bc'+ac)$와$(bd'+ad)$의 논리합을 산출해서 새로운 후보식 $(bc'+ac+bd'+ad)$를 산출한다. 각 논리식 별로 방금 서술한 원리에 따라 후보식을 산출한다.

후보식들에서 여러 논리식에 공통으로 사용되는 공통 논리식을 찾는 방법을 정리하면 다음과 같다. l_i, $(i=1,\cdots,n)$를 후보식이라 하고, $L=\{l_1, l_2, \cdots, l_n\}$는 후보식들의 집합이라 하자. 그리고, $IF(L)$을 후보식 집합 L에 대응하는 식이라고 하고, $I(L)$을 후보식들의 교집합이라 하자. 교집합 산출 방법은 다음과 같다.

알고리즘 1: 공통식 추출
입력 : 후보식 집합 $L=\{l_1, l_2, \cdots, l_n\}$
출력 : 공통식 $I(L)$
방법 :
단계 1: $IF(L)\leftarrow\varnothing$
for each $l_i\in L$ **do** {
  $t_j\leftarrow\varnothing$
  $j\leftarrow1$
  for each cube $c_i^j \in l_i$ **do** {
    $t_j^i\leftarrow c_i^j$
    $t_j\leftarrow t_j\cup t_i^j$
    $j\leftarrow j+1$
  }
  $IF(L)\leftarrow IF(L)\cup t_j$
}

단계 2: Calculate cokernels with $IF(L)$ and save the results to $I(L)$
return $(I(L))$

알고리즘 1은 후보식 집합 L에 대응하는 새로운 식 $IF(L)$를 만들고, $IF(L)$에서 제수들을 찾으면 바로 이것이 후보식들의 교집합이다. 교집합 산출 방법은 MIS에서 제안한 방법을 그대로 사용한다. $IF(L)$를 만들 때, 후보식들을 구성하는 큐브들을 새로운 변수로 치환하고, 후보식을 새로운 변수들의 곱으로 나타낸 것이 단계 1이다. 다음 $IF(L)$로 표현된 한 개의 식에 대해서 코커널 집합을 산출하는 것이 단계 2로 예 2에서 보인 바와 같이 결과를 산출한다. 본 연구에서는 코커널을 산출하는데 Brayton 등이 제시한 알고리즘[2]을 사용하였다. Brayton 등이 제시한 방법을 간단히 소개하면 논리식에서 사용한 서포트 변수의 조합을 만들어 가면서 2개 이상의 항을 동시에 나누는 제수(divisor)를 찾는데, 이 때 산출된 제수가 바로 코커널이 된다. 이 코커널들이 바로 공통식이 된다.

예 6: 산출된 후보식 중에 하나가 $l_1=bc'+ac+bd'+ad$, $l_2=ac+bd'+dc'+b'e$라 하자. 단계 1에서 $t_1=bc'$, $t_2=ac$, $t_3=bd'$, $t_4=ad$, $t_5=dc'$, $t_5=b'e$로 치환한다. 다음 후보식 l_1은 $t_1t_2t_3t_4$, 후보식 l_2는 $t_2t_3t_5t_6$가 되어 $IF(L)=t_1t_2t_3t_4+t_2t_3t_5t_6$로 표현한다. $IF(L)$로부터 단계 2에서 코커널 집합을 산출하면 $I(L)=\{t_2t_3\}$가 된다. 즉, 공통식은 $ac+bd'$이 된다.

만일 알고리즘 1에서 다수 개의 공통식이 산출된 경우 전체 리터럴 개수를 줄이는 공통식을 선택한다. 선택 방법은 각 공통식에 가중치를 부여한 식(2)를 사용한다. 공통식이 q_i인 경우 가중치는 $weight(q_i)$로 표기한다.

$$weight(q_i)=(NF(q_i)-1)(L(q_i)-1)-1 \quad (2)$$

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

전체 논리식을 간략화하는 방법은 알고리즘 2와 같이 표현된다.

<표 1> 논리식 산출 비교 결과

회로	입력수	출력수	초기 리터럴수	SIS[3]		FBDD[6]		2-cube[7]		제안방법	
				리터럴수	시간 (초)	리터럴수	시간 (초)	리터럴수	시간 (초)	리터럴 수	시간 (초)
alu4	14	8	6610	1099	50.8	5206	4.2	255	12.1	243	60.2
apex6	135	99	1422	854	0.1	1441	0.1	799	0.2	781	3.1
C880	60	26	703	473	0.1	634	0.1	465	0.7	445	2.2
C1355	41	32	1032	670	0.1	610	0.1	554	1.4	484	2.5
C1908	33	25	1497	564	0.1	605	0.1	552	1.7	522	3.4
C2670	233	140	2043	840	0.2	1352	0.3	902	2.2	832	7.2
C5315	178	123	4369	2008	0.9	2688	0.6	1824	4.1	1807	6.1
C6288	32	32	4800	3787	1.3	4800	0.1	3350	6.8	3341	7.9
C7552	207	108	6098	2584	2.2	3213	1.7	2450	14.2	2426	23.5
합	-	-	28,574	12,879	-	20,549	-	11,151	-	10,881	-

알고리즘 2: 해밍거리 3을 이용한 논리식 간략화
입력 : 단순식 형태의 다변수 출력 논리식
출력 : 공통식이 추출된 다변수 출력 논리식
방법 :
단계 1: 각 출력별로 커널/코커널 집합과 해밍거리 3인 논리항들을 고려한 후보식들 산출;
단계 2: 알고리즘 1을 적용해서 공통식을 산출;
단계 3: 식(2)를 적용해서 가중치가 가장 큰 공통식을 선택;
단계 4: 단계 3에서 선택한 공통식을 새로운 변수 W로 대치하고, 공통식을 포함하는 모든 논리식에서 공통식 부분을 W로 변경;

예 7: 다음 3개의 논리식에 대하여 알고리즘 2를 적용한 결과를 산출해 보자.

$$F_1 = a'bc' + ab'c + a'bd' + ab'd$$
$$F_2 = acef + bd'ef + adef$$
$$F_3 = ace + bd'e$$

단계 1이 각 논리식에 대해 산출한 해밍거리 3인 후보식들이 $F_1 = (bc' + ac + bd' + ad)(ab)'$, $F_2 = (ac + bd' + ad)ef$, $F_3 = (ac + bd')e$가 된다. 단계 2에서 공통식 $q_1 = ac + bd'$가 산출되고, 단계 3에서 q_1이 F_1, F_2, F_3에 포함되기 때문에 $NF(q_1) = 3$, 또 q_1은 4개의 리터럴로 구성되었기 때문에 $L(q_1) = 4$가 된다. 단계 4에서 F_1, F_2, F_3에서 q_1에 해당하는 부분에 새로운 변수 W를 도입해서 전체 논리식을 간략화 하면 다음과 같이 18개의 리터럴을 갖는 간략화된 논리식들로 표현된다.

$$F_1 = (W + bc' + ad)(ab)'$$
$$F_2 = (W + ad)ef$$
$$F_3 = We$$
$$W = ac + bd'$$

4. 실험 결과

Linux 운영체제하의 3.3GHz i3 CPU가 장착된 PC에서 제안한 방법을 C언어로 작성하였다. 실험은 Microelectronics Center of North Carolina(MCNC) 벤치마크 회로[12] 일부를 대상으로 제안한 방법과 타 방법들의 수행 결과인 리터럴 개수와 수행시간을 기준으로 비교하였다. 이유는 본 논문에서 제안한 방법이 선험 알고리즘이기 때문에 시간 복잡도 산출대신 벤치마크 회로를 대상으로 알고리즘 수행결과를 비교한 것이다. 성능 비교 대상으로는 커널 기반의 알고리즘을 사용한 MIS 또는 SIS와 FBDD를 이용한 논리식 산출 방법, 그리고 부울 공리를 적용한 2-큐브 제수를 산출해서 공통식을 산출하는 방법이다. 이 방법들을 선정한 이유는 SIS는 연구자들 비교 대상으로 널리 사용하기 때문이며, 다른 방법들은 최근에 발표된 것으로 제안하는 방법과 같이 공통식 추출을 목표로 하기 때문이다. <표 1>에 실험 한 자료를 나열하였다. <표 1>의 첫 번째 열은 벤치마크회로의 이름을 나타낸 것이다. 두 번

째와 세 번째 열에 벤치마크 회로의 입력과 출력 개수를 각각 표시하였다. 네 번째 열은 초기상태에서 벤치마크 회로를 구성하는 리터럴 개수를 표시하였다. <표 1>에서 마지막 2개의 열들이 제안한 방법에 의해 산출된 결과를 보인 것이다. <표 1>의 마지막 행은 표에 리터럴 개수의 합을 표시한 것이다. 실험 결과 리터럴 개수를 비교하면, 모든 벤치마크 회로에 대하여 리터럴 개수를 줄이는 효과를 발휘하였다. 제안한 방법은 SIS보다 16%, FBDD보다 47%, 2-cube 보다 3% 정도의 리터럴 개수를 줄이는 효과를 얻었다. 반면에 해밍거리가 3인 큐브들 산출에 소요되는 시간이 요구되고, 또 타 방법보다 공통식을 추출하는데 사용되는 후보식들이 많아 상대적으로 긴 수행시간이 요구되었다. 특히 alu4 벤치마크 회로의 경우 수행시간이 많이 소요된 이유는 제안한 방법이 SIS에서 사용된 커널산출을 통한 공통식 추출 방법을 포함하고 있기 때문으로 풀이된다. <표 2>는 제안한 방법에 의해 산출된 리터럴 개수와 해밍거리가 3인 경우 큐브들을 이용해서 산출된 후보식 개수와 후보식들에서 2개 이상의 출력에서 공통으로 사용된 공통식 개수를 나타낸 것이다. <표 2>는 제안한 방법과 타 방법이 산출한 후보식과 공통식 개수를 비교한 것이다. 대체로 제안한 방법이 후보식의 개수가 가장 많이 산출되었고, 그 결과 공통식의 산출 개수도 상대적으로 많음을 보이고 있다. 제안 방법의 경우 C7552의 경우 <표 2>에서 후보식에 비해 산출된 공통식 개수가 적다. 따라서 <표 1>에서 C7552가 타 방법에 비해 장시간의 수행 시간 소요에 비해 리터럴 개수를 줄이는 효과가 상대적으로 적음을 보이고 있다.

5. 결론

논리회로 심화학습을 위한 설계자동화 도구를 제안한 것으로 여러 논리식에 존재하는 공통부분을 찾고, 공통부분이 반복 사용되지 않도록 하여 전체 논리식을 간략화하는 방법을 보인 것이다. 간략화 결과는 전체 논리식에 사용된 리터럴 개수를 최소화하도록 고안하였다. 출력수가 2개 이상의 논리회로에서 공통식을 산출해서 각 출력에서 공통으로 사용된 부분을 새로운 변수로 대치하여 논리식을 간략화하는 공통식 추출방법을 제안하였다. 제안한 방법은 각 논리식 또는 출력별로 해밍거리가 3인 큐브들을 이용해서 주어진 논리식 변형을 통해 보다 많은 공통식이 될 수 있는 후보식을 산출하는 것이다. 실험 결과 수행시간은 타 방법들에 비해 상대적으로 늘어났지만, 리터럴 개수를 줄이는 데 효과적임을 보였다. 보통 설계자동화 도구는 2~3일 정도의 수행시간을 허용할 정도로 수행시간 보다는 간략화 정도를 더 중요시 한다. 이런 측면에서 제안한 방법이 타 방법에 비해 우수함을 보인다. 그러나 시간 제약이라는 조건을 만족해야 하는 경우를 고려하면 타 방법과 제안하는 방법을 함께 사용해서 서로 보완하는 방법도 좋을 것으로 본다.

<표 2> 후보식과 공통식 개수 비교

회로	SIS[3]		FBDD[6]		2-cube[7]		제안방법	
	후보식 수	공통식 수	후보식 수	공통식 수	후보식 수	공통식 수	후보식 수	공통식 수
alu4	231	42	189	32	273	69	503	78
apex6	186	51	78	28	582	56	849	86
C880	468	27	452	19	870	39	1329	53
C1355	624	40	578	37	882	48	1689	61
C1908	825	42	726	41	1127	45	2527	49
C2670	472	77	424	61	1056	75	3489	83
C5315	1121	52	947	45	2895	88	5369	101
C6288	1102	83	1032	78	5895	91	6423	92
C7552	1689	23	1502	12	6458	25	8934	29

참 고 문 헌

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

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

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

[4] J. Rajski, J. Vasudevamurthy. (1992). The Testability-Preserving Concurrent Decomposition and Factorization of Boolean Expressions. *IEEE Trans. CAD*, 11(6), 778-79.

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

[6] D. Wu, J. Zhu. (2005). FBDD: A Folded Logic Synthesis System. Technical Report TR-07-01-05, University of Toronto.

[7] 권오형, 오임걸 (2008). 2-큐브 제수와 보수에 의한 공통 논리식 산출. **정보처리학회 논문지 A, 15**(1), 9-16.

[8] 권오형 (2012). 논리식 인수분해를 위한 코스웨어. **컴퓨터교육학회논문지, 15**(1), 65-72.

[9] J. Cong, K. Minkovich. (2007) Optimality Study of Logic Synthesis for LUT-Based FPGAs. *IEEE Trans. CAD,* 26(2). 230-239.

[10] L Amaru, P.-E. Gaillardon, G. De Micheli. (2016). Majority-Inverter Graph: A New Paradigm for Logic Optimization. *IEEE Trans. CAD,* 35(5), 806-819.

[11] D. Kagaris. (2016). MOTO-X: A Multiple-Output Transistor-Level Synthesis CAD Tool. *IEEE Trans. CAD,* 35(1), 114-127.

[12] S. Yang. (1991). Logic Synthesis and Optimization Benchmarks User Guide Version 3.0. Technical Report, Microelectronics Center of North Carolina.

권 오 형

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