Journal of the Korea Academia-Industrial cooperation Society
Vol. 22, No. 7 pp. 11-18, 2021

https://doi.org/10.5762/KAIS.2021.22.7.11
ISSN 1975-4701 / eISSN 2288-4688

스펙트럴 방식을 적용한 공통식 추출

권오형
한서대학교 항공컴퓨터전공

Common Expression Extraction Using Spectral Method

Oh-Hyeong Kwon
Dept. of Aeronautic Computer Engineering, Hanseo University

요 약 여러 논리식에 존재하는 공통식을 찾아 반복 사용을 줄여서 전체 논리식에 사용된 리터럴 개수를 줄이기 위한 공통식 추출 방법에 대한 방안을 제안한 것이다. 특히 XOR 연산자를 이용할 경우 논리식이 간략화 될 수 있으며 또한 2개 입력 XOR 게이트는 3개의 트랜지스터만으로 구현이 될 수 있는 장점이 있기 때문에 XOR 연산자를 활용한 논리식을 산출하도록 고안하였다. XOR 연산자를 포함한 논리식을 산출하는 데는 하다마드 행렬을 이용한 스펙트럴 방식을 이용하며, 산출된 논리식은 AND, OR, XOR, NOT 연산자들이 사용된 배타식을 산출한다. 공통식이 될 수 있는 배타식 후보군을 찾기 위한 선험 방법을 또한 제안하였으며, 이 선험 방법을 적용한 알고리즘을 제안하였다. 벤치마크 회로를 대상으로 수행 시간과 산출된 리터럴 개수를 대상으로 타 방법들과 제안한 방법을 비교하였다. 실험결과 여러 벤치마크 회로에 대하여 제안한 방법이 기존 논리식 방법들보다 리터럴 개수를 줄일 수 있음을 보였으며 특히 기존의 스펙트럴 방식보다 약 7% 정도의 리터럴 개수를 줄이는 효과를 얻었다.

Abstract This paper proposes a method for extracting common expressions to reduce the number of literals used in all logical expressions by finding common expressions in several logical expressions and reducing repetitive use. In particular, when using the XOR operator, the logical expression can be simplified. In the case of a two-input XOR gate, it can be implemented with only three transistors. The spectral method using the Hadamard matrix calculates the logical expression, including the XOR operator. The logical expressions produced, called exclusive expressions, are expressed using the AND, OR, XOR, and NOT operators. An a priori method for finding exclusive expressions candidate that can be common expressions is also proposed, and an algorithm applying this a priori method is proposed. The proposed method was compared with other methods for the execution time and number of literals for the benchmark circuits. The experimental results showed that the proposed method for several benchmark circuits could reduce the number of literals compared to the existing logical expression methods. In particular, it could reduce the number of literals by approximately 7%.

Keywords : Logic Simplification, Spectral Method, Hadamard Matrix, Common Expression Extraction, Exclusive Expression

*Corresponding Author : Oh-Hyeong Kwon(Hanseo Univ.)
email: ohkwon@hanseo.ac.kr
Received April 30, 2021 Revised June 10, 2021
Accepted July 2, 2021 Published July 31, 2021

1. 서론

논리합성 단계는 간략화된 논리회로 산출을 목표로 하며 목표를 달성하기 위해서 수행과정은 간략화된 논리식을 산출하고 이 결과를 논리회로로 변환하는 방법을 취하고 있다. 그래서 논리회로 간략화 연구는 논리식 간략화를 위한 연구로 진행 중에 있다. 특히 2단 이상으로 논리식을 산출하는 다단 논리식 간략화 방법에 대한 연구가 논리합성 단계에서 진행 중에 있다. 다단 논리식 산출 방법 중에는 여러 논리식들에 존재할 수 있는 공통 부분 즉, 공통식들을 찾아 반복 사용을 줄여 전체 논리식들을 간략화하는 공통식 추출 방법이 필요하게 된다. 본 논문은 공통식을 찾아 논리식 간략화를 성취하기 위한 방법을 제안한 것이다. 공통식을 산출하기 위해 본 논문에서는 하다마드(Hadamard) 행렬을 이용한 스펙트럴 방법(Spectral Method)을 활용한다.

현재까지 다단 논리식 산출을 위해 발표된 논리합성 연구들에 대하여 소개하면 다음과 같다. Brayton 등은 커널(kernel)을 이용한 공통 다항 큐브 제수를 찾는 방법[1-3]으로 논리합성 도구 SIS를 발표하였다. Kwon은 배타 논리합 산출 원리를 적용해서 AND, OR, NOT만으로 구성된 공통식을 산출하는 방법[4,5,6]을 적용해서 SIS보다 간략화된 논리식을 산출하는 방법을 제시하였다. Hurst 등은 스펙트럴 방식을 적용한 논리식 간략화 방법[7,8]을 제안하였고, 특히 스펙트럴 방식이 대칭 논리식(Symmetric Logic Expressions)에 적합함을 보이고 있다. 또 Diaz-Olavarrieta 등이 스펙트럴 방법을 적용한 논리식 간략화 방법[9]으로 주어진 진리표를 치환하는 방법을 통해 논리식 간략화를 시도하였다. 그러나 이들의 연구들은 공통식이 있음에도 불구하고 때때로 공통식을 산출하지 못하는 단점을 갖고 있다.

본 논문에서 산출되는 논리식은 AND, OR, NOT, XOR(Exclusive-OR) 연산자로 구성된다. Kim 등은 2개의 입력을 갖는 XOR 게이트를 단지 3개의 트랜지스터로 구현하였다[10]. 참고로, 2개의 입력을 갖는 AND와 OR 게이트는 각각 4개의 트랜지스터로 구현이 된다. 또한 AND 연산와 XOR 연산자를 이용할 경우 AND 연산자와 OR 연산자를 사용할 때보다 논리식을 구성하는 리터럴 개수를 줄일 수 있다. 따라서 XOR 연산자를 활용해서 간략화된 논리식을 산출하는 데는 논리회로 레벨뿐만 아니라 트랜지스터 레벨에서도 장점이 있다. 최근 연구로 XOR 게이트가 활용되는 분야로는 양자 컴퓨터와 관련된 리버서블 로직에 사용된다[11,12].

논문의 구성은 다음과 같다. 2장에서는 본 논문 서술에 필요한 배경 지식으로 용어 정의와 하다마드 행렬에 대해 서술하고, 3장에서 하다마드 행렬을 이용해서 논리식을 산출하는 기존 방법과 본 논문에서 제시하는 새로운 공통식 산출 방법을 소개한다. 4장에서 실험결과를 보이고, 5장에서 결론을 제시한다.

2. 배경 지식

본 장에서 서술하는 용어 정의들과 예들은 [1-3,6]에 서술된 것들을 인용한 것이다.

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. 항(term) 또는 큐브(cube)는 리터럴들의 집합으로 만일 리터럴 a가 존재하면, 그의 보수 리터럴 a'을 포함하지 않는다. 큐브를 카르노 맵(Karnaugh map)에서 표현했을 때 묶음이라고 한다. 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 몫을 커널이라 한다. 이 때 커널을 산출한 큐브를 코커널(co-kernel)이라 한다.

예 1: 문자 a는 변수이며, a와 a'은 리터럴이다. 리터럴 집합 $\{a,b\}$는 큐브이나 집합 $\{a,a'\}$은 큐브가 아니다. 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 2: 하다마드(Hadamard) 행렬 H의 각 원소는 Eq. (1)과 같이 재귀적 수식에 의해 산출된다. 하다마드 행렬의 원소값은 1과 -1로 표현된다. 즉, 논리값 0은 하다마드 행렬에서 1로, 논리값 1은 -1로 변경한다. 여기서 n은 입력 변수의 개수를 나타낸다.

$$H^n = \begin{bmatrix} H^{n-1} & H^{n-1} \\ H^{n-1} & -H^{n-1} \end{bmatrix} \quad (1)$$

$$H^0 = 1$$

예 2: $n=1$인 경우와 $n=2$에 대한 각 하다마드 행렬은 Eq. (2)와 같은 모습이 된다.

$$H^1 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \quad (2)$$

$$H^2 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}$$

정의 3: 하다마드 행렬의 각 행은 논리식에 대응된다. 논리식에 사용된 입력 변수를 x_i, (i=2^n-1,..., 0)라 하자. 그러면 하다마드 행렬의 각 행에 대응되는 논리식은 첫 번째 행은 0이 되고, 두 번째 행은 x_0, 세 번째 행은 x_1, 네 번째 행은 $x_0 \oplus x_1$, 다음 다섯 번째 행은 x_2, 다음 2개의 행은 각각 $x_0 \oplus x_2$, $x_1 \oplus x_2$와 같이 논리식이 대응된다. 나머지 행에 대해서도 동일한 방식으로 논리식이 대응된다.

예 3: 입력 변수가 c, b, a 경우 8x8 하다마드 행렬과 각 행에 대응되는 논리식을 행렬의 왼쪽에 나타내면 Eq. (3)과 같다.

$$\begin{array}{r} 0 \\ a \\ b \\ a\oplus b \\ c \\ a\oplus c \\ b\oplus c \\ a\oplus b\oplus c \end{array} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix} \quad (3)$$

정의 4: 논리식 F를 배타 연산자 $\oplus$(Exclusive-OR 또는 간략히 XOR)를 포함하는 식 $F=(S\oplus M)+R$로 표현했을 때 배타식이라 부른다. 여기서, S를 선택식, M를 보정식, R은 나머지라 부르며, S. M, R은 다시 배타식으로 표현이 될 수 있다. 이 때 $R=\phi$가 가능하다.

예 4: Table 1의 진리표는 논리식 $F=(a\oplus abc)+bc'$으로 표현 가능하다. 여기서 선택식 $S=a$, 보정식 $M=abc$, 나머지 $R=bc'$가 된다.

Table 1. Truth Table w.r.t. Ex. 4

Inputs			Output
c	b	a	F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

3. 제안한 논리식 간략화 방법

본 논문의 핵심 부분인 하다마드 행렬을 이용해서 논리식을 간략화하는 방법에 대해 서술한다. 산출되는 논리식은 배타식으로 특히 출력이 여러 개가 있을 경우 공통식 추출에 의한 간략화 방법을 제안한다. 3.1절에서는 단일 출력 회로에 대한 배타식 산출 과정을 보이고, 3.2절과 3.3절에서는 출력이 여러 개인 경우 공통식을 산출해서 전체 논리식을 간략화 하는 방법을 소개한다. 3.2절에서는 기존 연구 결과를 3.3절에서는 제안하는 방법을 기술한다.

3.1 하다마드 행렬을 이용한 배타식 산출

정의 2에서 제시한 하다마드 행렬과 출력에 대응되는 행렬(이하 출력 행렬이라 부른다)을 만들고 이 2개의 행렬 곱셈 결과를 산출한다. 주어진 출력값 0은 1로, 1은 -1로 대치해서 출력 행렬이 1과 -1로 구성되도록 한다. 하다마드 행렬과 출력 행렬의 곱셈 결과 산출된 행렬에서 절대값이 가장 큰 행 위치를 찾는다. 그러면 하다마드 행렬에서 이 행 위치에 대응하는 논리식이 바로 선택식이 된다.

$$H^n F = Z \quad (4)$$

Eq. (4)는 선택식을 산출하기 위해 하다마드 행렬과 출력 행렬의 곱셈을 표현한 수식이다. Eq. (1)에서 H^n는 입력 변수의 개수가 n인 경우 2^n x 2^n 크기의 하다마드 행렬을 나타낸 것이며, F는 2^n x 1 출력 행렬을 표현한 것이다. Z는 2^n x 1 행렬로 곱셈 결과를 나타낸 것이다. 예 5는 하다마드 행렬을 이용해서 논리식 즉, 선택식 S를 산출하는 것을 보인 것이다.

예 5: 주어진 진리표가 Table 2와 같다고 하자.

Table 2. Truth Table w.r.t. Ex. 5

Inputs			Output
c	b	a	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

그러면 입력 변수가 3개이기 때문에 하다마드 행렬은 8x8의 크기가 되고, 출력 행렬은 Eq. (5) 왼쪽 변에서 두 번째 8x1 크기의 행렬이다.

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ -1 \\ 1 \\ 1 \\ 1 \\ -1 \\ -1 \\ 1 \end{bmatrix} = \begin{bmatrix} 2 \\ 2 \\ -2 \\ 6 \\ 2 \\ 2 \\ -2 \\ -2 \end{bmatrix} \quad (5)$$

Eq. (5)의 오른쪽 변에 있는 행렬에서 6이 가장 큰 절대값이 되며, 이는 네 번째 행이 된다. 따라서 예 3에서 보인 바와 같이 하다마드 행렬의 네 번째에 대응되는 $a \oplus b$가 선택식 S가 된다.

보정식을 산출하는 과정은 예 5를 이용해서 설명한다. 주어진 출력에 대응되는 진리표를 카르노 맵으로 표현하고, 여기에 산출된 선택식 $S = a \oplus b$에 대응되는 부분에 묶음으로 표현한 것이 Fig. 1과 같다.

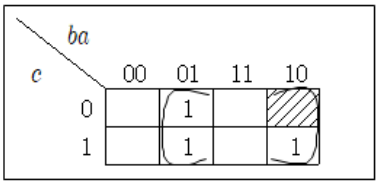

Fig. 1. Cover of $S = a \oplus b$

Fig. 1의 묶음은 $a'bc'$에 해당하는 빗금친 부분을 포함하기 때문에 다시 이 부분을 제거하는 위한 논리식이 필요하게 되는데, 이 식이 보정식 $M = a'bc'$이 된다. 그러면 예 5의 진리표에 일치하는 논리식은 $F = S \oplus M = (a \oplus b) \oplus a'bc'$이 된다. 이처럼 보정식은 선택식에 포함된 부분 중에서 0에 해당하는 부분을 제거하기 위한 논리식이 된다.

3.2 공통식 산출 행렬

출력이 2개 이상인 다출력 논리회로를 간략화 하고자 할 때, 여러 출력에 존재할 수 있는 공통식들을 찾아서 전체 논리식들에서 이들이 한 번만 사용되도록 해서 논리식을 간략화 한다.

본 절에서 서술하는 내용은 기존의 연구 결과를 보인 것이다. 공통식 산출 방법으로 출력 행렬들의 각 행들을 더하고 이를 하다마드 행렬과 곱셈을 하고, 절대값이 가장 큰 행에 대응하는 식을 공통식으로 산출하는 방법이다 예 6은 3개의 출력을 갖는 회로를 간략화하기 위해서 공통식을 찾는 기존 방법을 보인 것이다. 기존 방법의 문제점과 제안한 방법으로 간략화한 결과만을 예 6에서 보이고, 제안하는 공통식 산출 방법은 다음 절에서 서술한다.

예 6: 주어진 진리표가 다음과 같을 때 공통식을 산출하자.

Table 3. Truth Table w.r.t. Ex. 6

Inputs			Outputs			Outputs (after conversion)		
c	b	a	F_0	F_1	F_2	F_0	F_1	F_2
0	0	0	0	0	0	1	1	1
0	0	1	0	1	0	1	-1	1
0	1	0	1	1	0	-1	-1	1
0	1	1	1	1	0	-1	-1	1
1	0	0	0	0	1	1	1	-1
1	0	1	1	1	1	-1	-1	-1
1	1	0	1	1	1	-1	-1	-1
1	1	1	0	0	0	1	1	1

먼저 위의 각 출력단 즉, F_0, F_1, F_2 출력값을 1과 -1로 변경한다. 위의 표에서 오른쪽 3개의 열에 해당하는 부분(표에서 "변경 후"라고 명시한 부분)으로 변경한다. 다음 오른쪽 3개의 열들을 행 별로 더한다. 한 가지 경우를 보이면, 진리표의 첫 번째 행에 해당하는 $F_0 = 0$, $F_1 = 0$, $F_2 = 0$의 경우는 1+1+1=3이 된다.

진리표의 나머지 행들도 같은 방법으로 값이 산출되며, 이렇게 산출된 출력 행렬과 하다마드 행렬의 곱을 산출한다. 즉, Eq. (6)과 같은 행렬 곱셈을 한다. 그러면 네 번째 행에서 12라는 값이 가장 큰 절대값이 되고, 이는 하다마드 행렬의 네 번째 행에 대응되는 $a \oplus b$가 공통식이 됨을 의미한다.

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix} \begin{bmatrix} 3 \\ 1 \\ -1 \\ -1 \\ 1 \\ -3 \\ -3 \\ 3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 4 \\ 12 \\ 4 \\ 4 \\ 8 \\ -8 \end{bmatrix} \quad (6)$$

위의 방법은 완벽하게 공통식을 산출할 수 없게 되는데, 이유는 $R = \phi$인 배타식 $F = S \oplus M$ 형태의 논리식만을 산출하면서 공통식은 선택식 S 부분에서만 찾기 때문이다.

3.3 선험 법칙에 의한 배타식 산출과 공통식 추출

본 논문는 다출력 회로에서 하다마드 행렬을 이용한 스펙트럴 방식에 의해 산출된 배타식의 선택식 부분들, 보정식 부분들과 나머지 부분에서 공통식을 찾는 방법을 제안한 것이다. 먼저 예 7를 통해 제안하는 방법으로 논리식을 간략화하는 과정을 보인다.

예 7: 주어진 진리표가 예 6과 같을 때 제안하는 방법에 따라 간략화하면 다음과 같다. 진리표를 Fig. 2에서 카르노 맵으로 표현하였고, 또한 산출된 배타식도 함께 제시하였다. Fig. 2에서 각 출력별로 선택식 부분은 묶음으로, 보정식 부분은 빗금으로, 묶음에 속하지 않는 부분은 나머지 부분으로 배타식을 구성하게 된다. 다음 산출된 세 개 배타식에서 공통부분을 다음과 같이 새로운 변수로 대치하면 최종적인 간략화된 논리식을 산출하게 된다. 그러면 Eq. (7)에 나열된 것과 같이 최종 논리식은 14개의 리터럴로 구성된 식들이 산출된다.

$$\left.\begin{aligned} F_0 &= X + Zc \\ F_1 &= X + Z \\ F_2 &= c \oplus Y \\ X &= b \oplus Y \\ Y &= abc \\ Z &= ab' \end{aligned}\right\} \quad (7)$$

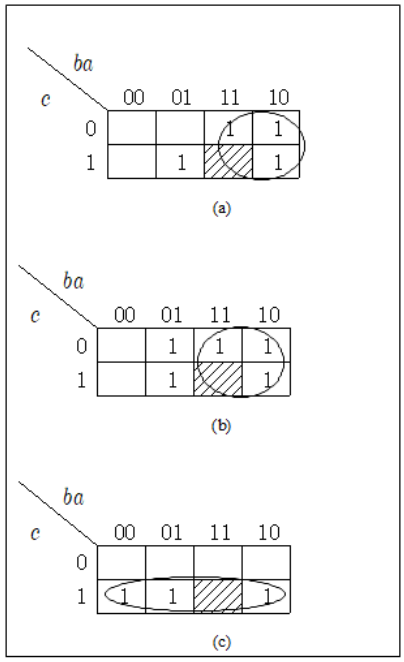

Fig. 2. Exclusive Expressions
(a) $F_0 = (b \oplus abc) + ab'c$
(b) $F_1 = (b \oplus abc) + ab'$
(c) $F_2 = c \oplus abc$

3.3.1 선험 방법

수행시간을 줄이면서 여러 공통식 후보군을 포함하게 될 배타식을 산출하기 위해 다음 2가지 선험 방법을 제안한다.

선험 방법 1:
행렬 곱셈 결과 절대값이 가장 큰 값을 산출하는 행과 바로 다음으로 절대값이 큰 값을 갖는 행을 선택한다.

선험 방법 2:
행렬 곱셈 결과 여러 행들이 동일한 절대값을 갖는 경우, 그 행에 대응되는 논리식이 다른 출력들의 논리식들과 일치하는 리터럴 개수가 가장 많은 것, 즉 행을 선택한다.

선험 방법 1은 공통식이 될 수 있는 2개의 논리식을 산출하기 위한 것이다. 선험 방법 2는 행렬 곱셈 결과 2개 이상의 값이 동일하면서 가장 큰 절대값이 되는 경우, 이 중에서 1개만을 선택하기 위한 것이다. 예 8은 선험

방법 1과 2를 적용해서 공통식이 될 수 있는 후보식을 산출하는 것을 보인 것이다.

예 8: 예 7에서 F_1의 경우 하다마드 행렬과의 곱셈 결과는 $[-2\ 2\ 2\ 6\ -2\ 2\ 2\ -2]^T$가 된다. 여기서 T는 전치 행렬임을 표시한 것이다. 그러면 예 3에서 보인 것과 같이 행렬의 원소값이 6인 네 번째 행에 대응되는 $a \oplus b$ 와 선험 방법 1에 따라 또 하나의 공통식 후보가 될 수 있는 행을 선택한다. 이 때 절대값이 모두 2가 되기 때문에 이 중에서 하나의 행을 선택하는 데 선험 방법 2를 적용한다. 그러면 선험 방법 2에 따라 세 번째 행이 선택된다. 즉 $b \oplus abc$에 대응되는 묶음이 F_0과 F_1에서 동시에 발견되기 때문이다.

3.3.2 공통 부분 추출

산출된 여러 배타식들에서 공통 부분을 추출하기 위한 방법에 대해서 기술한다. 배타식을 2 종류의 논리식 집합으로 구분한다. 이유는 배타식은 항들을 OR로 연결하는 부분들과 XOR로 연결하는 부분들로 구성되기 때문이다. 즉, AND 항들을 OR 연산자로 연결하는 AND와 OR 연산자로 구성되는 SOP(Sum of Products) 형태의 논리식들 집합과 AND 항들을 XOR 연산자로 연결하는 AND와 XOR 연산자로 구성된 EOP(Exclusive-Sum of Products) 형태의 논리식들 집합으로 구분한다. 본 논문에서는 이 2 부류의 논리식 집합을 각각 SOP와 EOP라 부르겠다. 공통식 산출에는 [1-3]에서 제시한 방법을 활용한다. 예 9는 SOP 논리식 집합과 EOP 논리식 집합에서 공통식을 찾는 과정을 보인 것이다.

예 9: 논리식 $Y_1 = (e \oplus f \oplus g) + bc' + ac + bd' + ad$, $Y_2 = (e \oplus g) + ac + bd' + dc' + b'e$에 대해서 공통식을 산출하자. 그러면, SOP 형태의 논리식 집합에는 Y_1으로부터 $l_1 = bc' + ac + bd' + ad$를 추출할 수 있고, Y_2로부터 $l_2 = ac + bd' + dc' + b'e$를 추출할 수 있다. 추출한 논리식들의 각 항을 다음과 같이 $t_1 = bc'$, $t_2 = ac$, $t_3 = bd'$, $t_4 = ad$, $t_5 = dc'$, $t_5 = b'e$로 치환한다. 다음, l_1은 $t_1t_2t_3t_4$로, l_2는 $t_2t_3t_5t_6$로 바꾸고 이들을 OR 연산자를 이용해서 $IF(L) = t_1t_2t_3t_4 + t_2t_3t_5t_6$로 표현한다. $IF(L)$에 대해서 코커널 집합을 산출하면 $I(L) = \{t_2t_3\}$가 된다. 즉, 공통식은 $ac + bd'$이 된다. 마찬가지로 EOP 집합의 논리식들은 Y_1로부터 $z_1 = e \oplus f \oplus g$와 Y_2로부터 $z_2 = e \oplus g$가 추출되고, 동일한 방법으로 공통식을 산출하게 되면 $e \oplus g$가 얻어진다.

본 논문에서는 추출된 공통식을 새로운 변수로 대치하기 위해서 [2]에서 제시한 공통식에 가중치를 부여하는 Eq. (8)을 이용한다. 이 때 가중치가 1보다 큰 경우 배타식들에 존재하는 공통부분을 새로운 변수로 대치해서 전체 논리식을 구성하는 리터럴 개수를 줄인다. Eq. (8)은 공통식이 q_i인 경우 가중치 $weight(q_i)$를 산출하는 것으로 다음과 같은 값들로 식이 구성된다.

$$weight(q_i) = (NF(q_i) - 1)(L(q_i) - 1) - 1 \quad (8)$$

Where $NF(q_i)$ is the number of logical expressions including the common expression q_i, and $L(q_i)$ is the literal number of the common expression q_i itself.

3.3.3 알고리즘

선험 방법들과 공통식 추출 방법을 적용해서 간략화된 논리식을 산출하기 위해 다음 알고리즘을 제안한다. 입력 자료는 진리표이며, 출력 결과는 공통식이 추출된 배타식들이다. 단계 1은 출력 별로 하다마드 행렬과 곱셈을 수행하는 과정이며, 단계 2는 곱셈 결과 논리식을 간략화 할 것으로 생각되는 배타식을 산출한다. 단계 3-6은 예 9에서 보인 것과 같은 동작을 수행해서 최종적으로 간략화된 논리식을 구하도록 고안한 것이다.

Algorithm : 논리식 간략화
입력: 주어진 2개 이상의 출력을 갖는 진리표
출력: 공통식이 추출된 배타식 집합
단계 1. 각 출력에 대해 하다마드 행렬과 곱셈을 산출
단계 2. 선험 방법 1과 2를 적용한 행들을 선택하고, 배타식을 산출
단계 3. 배타식들에서 SOP 집합과 EOP 집합을 산출
단계 4. SOP 집합에서 공통식 산출
단계 5. EOP 집합에서 공통식 산출
단계 6. Eq. (4)를 적용해서 공통식 부분을 선택하고 간략화된 논리식을 산출

Table 4. Experimental Results

Circuit Name	# of inputs	# of outputs	SIS		Hurst		Kwon		Proposed method	
			# of literals	time (sec)	# of literals	time (sec)	# of literals	time (sec)	# of literals	time (sec)
rd53	5	3	71	0.2	67	0.2	65	0.2	62	0.3
rd73	7	3	164	0.4	147	0.1	145	0.5	138	0.2
con1	7	2	23	0.1	19	0.1	19	0.2	17	0.1
z4ml	7	4	70	0.2	63	0.1	65	0.2	58	0.2
cmb	16	4	70	0.1	66	0.1	68	0.1	59	0.2
C17	5	2	10	0.1	8	0.1	10	0.2	8	0.1
decod	5	16	52	0.1	48	0.1	48	0.3	47	0.2
alu4	14	8	1752	1.2	1520	0.3	1491	0.8	1491	0.5
sao2	10	4	198	0.2	180	0.1	182	0.2	165	0.4
5xp1	7	10	158	0.2	113	0.1	124	0.3	107	0.6
misex1	8	7	83	0.1	68	0.1	75	0.2	66	0.5
Total Sum	-	-	2651	-	2299	-	2292	-	2218	-
Average	-	-	241	-	209	-	208	-	202	-

4. 실험 결과

3.3GHz i3 CPU가 장착된 Linux 운영체제의 컴퓨터에서 C언어로 제안한 방법을 구현하였다. 제안한 방법과 기존 방법들의 비교는Microelectronics Center of North Carolina (MCNC) 벤치마크 회로[12,13]를 대상으로 논리식 간략화에 소요된 수행시간과 리터럴 개수를 측정하였다. 실험 결과를 정리한 것이 Table 4다. Table 4의 첫 번째 열은 벤치마크회로의 이름이고, 두 번째와 세 번째 열은 각 벤치마크 회로의 입력 수와 출력 수를 표시하였다. 다음 2개의 열은 SIS에서 공통식 산출 방법을 적용해서 얻은 결과를 보인 것이다. 참고로, SIS에서 산출한 회로는 AND OR, NOT 게이트만을 사용한다. 다음 2개의 열은 하다마드 행렬을 이용해서 Hurst 등이 제시한 방법으로 산출한 결과를 보인 것이다. 다음 2개의 열은 Kwon의 해밍거리 3인 큐브들을 이용해서 공통식을 산출한 결과를 보인 것[6]으로 SIS와 같이 AND, OR, NOT 게이트만을 사용한 결과다. 마지막 2개의 열이 본 논문에서 제시한 스펙트럴 방식에 의해 공통식을 산출한 결과를 제시한 것이다. Fig. 3은 산출된 리터럴의 평균 개수를 막대 그래프로 표현한 것으로 막대 크기가 작을수록 우수성을 보인다. 제안한 방법에 의해 산출된 막대가 가장 작게 나타나 있음을 보이고 있다.

실험 결과를 보면 입력이 n개, 출력이 m개인 경우 $2^n * 2^n$ 행렬 곱셈을 m번 수행하는 것으로 공통식을 산출하기 때문에 공통식 산출에 소요되는 시간은 Hurst 방법에 비해 다소 늘어났지만 리터럴 개수를 줄이는 효과를 얻을 수 있었다. Hurst 등의 방법은 출력의 개수에 무관하게 한 번의 행렬 곱셈을 하기 때문에 수행시간은 상대적으로 빠른 방면 본 논문에서 제시하는 방법보다 리터럴 개수가 늘어나는 단점을 갖는다.

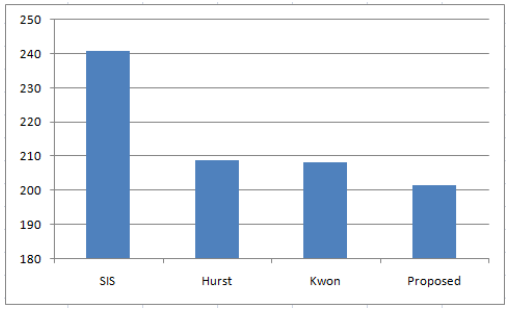

Fig. 3. Average Number of Literals Produced by Each Method

5. 결론

논리회로의 출력수가 다수 개인 경우 스펙트럴 방식 특히, 하다마드 행렬을 이용해서 공통식을 산출하는 방법에 대해 제시하였다. 제안한 방법은 각 출력별로 논리식을 산출하고 다음 산출된 논리식들에서 공통식을 찾도

록 하였고, 또한 공통식이 될 수 있는 후보식을 상대적으로 많이 찾기 위해서 2가지 선험 방법을 제안하였다. 실험 결과에서 보인 바와 같이 제안하는 방법으로 리터럴 개수를 줄이는 효과를 얻었다. 그러나 입력 변수와 출력 개수가 많을 경우 매우 큰 하다마드 행렬을 여러 번 사용하게 되는 단점을 갖기 때문에 과도한 메모리를 사용으로 인해 프로그램 수행에 문제를 초래할 수 있다. 따라서 대형 회로 간략화를 위해 주어진 회로를 나누어 각각 간략화하고 다시 이들의 결과들을 합하는 방법(Divide-and-Conquer)을 적용한 향후 연구가 필요할 것으로 본다.

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.
DOI: https://doi.org/10.1109/TCAD.1987.1270347

[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] O.-H. Kwon, “Logic Optimization Using Boolean Resubstitution", *Journal of Korean Academia-industrial cooperation Society*, Vol. 10, No. 11, pp. 3227-3233, 2009.
DOI: https://doi.org/10.5762/KAIS.2009.10.11.3227

[5] O.-H. Kwon and B.T. Chun, “Boolean Factorization Using Two-cube Non-kernels", *Journal of Korean Academia-industrial cooperation Society*, Vol. 11, No. 11, pp. 4597-4603, 2010.
DOI: https://doi.org/10.5762/KAIS.2009.10.11.3227

[6] O.-H. Kwon, “Common Logic Extraction Using Hamming Distance 3 Cubes", *The Journal of Korean Association of Computer Education*, Vol. 20, No. 4, pp. 77-84, 2017.
DOI: https://doi.org/10.32431/kace.2017.20.4.008

[7] S. L. Hurst, D. M. Miller, and J. C. Muzio, *Spectral Techniques in Digital Logic*.. San Diego, CA: Academic Press, 1985.

[8] C. R. Edwards and S. L. Hurst, "A Digital Synthesis Procedure Under Function Symmetries and Mapping Methods," *IEEE Trans. Computer*, Vol. c-27, No. 11, pp. 985-997, 1978.
DOI: https://doi.org/10.1109/TC.1978.1674988

[9] L. Diaz-Olavarrieta, K. Illanko, and S. G. Zaky, "Goal-Oriented Decomposition of Switching Functions," *IEEE Trans. CAD*, Vol. 12, No. 5, pp. 655-665, 1993.
DOI: http://dx.doi.org/10.1109/43.277610

[10] J. B. Kim, S. J. Hong, and J. Kim, "New Circuits for XOR and XNOR Functions," *International Jonrnal of Electronics,* Vol. 82, No. 2, pp. 131-144, 1997.
DOI: http://dx.doi.org/10.1080/002072197136138

[11] D. M. Miller and G. W. Dueck, “Spectral Techniques for Reversible Logic Synthesis,” *6th International Symposium on Representations and Methodology of Future Computing Technologies*, pp. 56-62, 2003.

[12] A. Zulehner and R. Wille, “One-Pass Design of Reversible Circuits: Combining Embedding and Synthesis for Reversible Logic,” *IEEE Trans. CAD,* Vol. 37, No. 5, pp. 996-1008, 2018.
DOI: http://dx.doi.org/10.1109/TCAD.2017.2729468

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

[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 항공컴퓨터전공 교수

〈관심분야〉
회로 설계자동화