한국산학기술학회논문지
Vol. 10, No. 11, pp. 3227-3233, 2009

부울 대입에 의한 논리식 최적화

권오형[1*]
[1]한서대학교 전자컴퓨터통신학부

Logic Optimization Using Boolean Resubstitution

Oh-Hyeong Kwon[1*]
[1]**Division of Electroinc, Computer, and Communication, Hanseo University**

요 약 본 논문에서는 나눗셈 행렬을 이용하여 부울 대입식을 산출하는 논리합성 방법을 제안한다. 최적화하고자 하는 2개의 논리식들로부터 대수 나눗셈에 의한 행렬을 만들고 부울 공리와 리터럴 추가를 통해 부울 나눗셈 행렬로 확장을 한다. 부울 나눗셈 행렬에 리터럴을 추가하여 확장된 부울 나눗셈 행렬을 만들고, 원소들을 커버링하여 부울 대입식을 산출한다. 실험결과 여러 벤치마크 회로에 대하여 제안한 방법이 기존 합성도구보다 리터럴 개수를 줄일 수 있음을 보였다.

Abstract A method for performing Boolean resubstitution is proposed. This method is efficiently implemented using division matrix. It begins by creating an algebraic division matrix from given two logic expressions. By introducing Boolean properties and adding literals into the algebraic division matrix, we make the Boolean division matrix. Using this extended division matrix, Boolean substituted expressions are found. Experimental results show the improvements in the literal counts over well-known logic synthesis tools for some benchmark circuits.

Key Words : Logic Synthesis, Logic Design, Boolean Division, Boolean Resubstitution

1. 서론

논리합성은 VHDL과 같은 하드웨어 표현 언어로 작성된 상위 계층의 동작을 실제 회로로 변환하기 위한 중간 단계로 논리합성 결과에 따라 산출되는 회로나 칩의 크기가 달라질 수 있다. 따라서 많은 연구자들이 최적 논리식 또는 논리회로를 산출하기 위한 논리합성 도구를 만들기 위해 연구를 하고 있다. 특히, 다단 논리회로가 2단 논리회로보다 더 간략화 될 수 있기 때문에 다단 논리회로에 대한 연구가 진행되고 있다. 이러한 다단 논리회로 최적화는 국부 최적화(local optimization)와 전역 최적화(global optimization)로 연구되어 왔다. 국부 최적화는 전체 부울 네트워크(Boolean network)의 형태에 영향을 주지 않고 단지 일부 논리식만의 최적화를 목적으로 한다. 반면에, 전역 최적화는 주어진 부울 네트워크의 변형까지 고려하면서 최적화를 수행하는 것을 목적으로 한다. 전역 최적화 방법 중의 하나인 대입(resubstitution) 기법은 2개의 논리식에서 하나의 논리식이 다른 논리식을 포함하는 경우, 포함된 부분을 간단히 변수로 대치하는 방법이다.

지금까지의 다단 논리식을 최적화하기 위한 대표적인 연구를 정리하면 다음과 같다. Brayton과 McMullen은 커널(kernel)을 이용한 공통 다항 큐브 제수를 찾는 방법[1]을 제시하였고, 후에 Brayton, Rudell, Sangiovanni-Vincentelli와 Wang은 커널의 부분집합을 이용한 다항 큐브 제수를 찾는 알고리즘과 조합회로를 위한 논리합성 도구 MIS[2]를 발표하였다. Sentovich 등은 MIS에 순차회로 최적화 능을 추가하여 SIS[3]를 만들었다. Rajski와 Vasudevamurthy는 다변수 출력을 갖는 논리회로에서 단지 2개의 리터럴만으로 된 단항 큐브와 그의 보수를 이용해서 공통식을 추출하는 방법[4]을 제안하였다. 이상의 [1-4]에서 제시한 방법들은 부울 공리인 등멱법칙($aa = a$)과 보수법칙($aa' = 0$)을 활용하지 않고, 단순히

*교신저자 : 권오형(ohkwon@hanseo.ac.kr)
접수일 09년 10월 10일 수정일 (1차 09년 11월 02일, 2차 09년 11월 10일) 게재확정일 09년 11월 12일

대수적인 방법으로 논리식 최적화를 수행하기 때문에 대체로 수행 속도는 빠르나, 때때로 최적화된 결과를 산출할 수 없게 된다. 최근에는 주어진 논리회로를 Binary-Decision Diagram(BDD)으로 표현하고 BDD를 이용한 논리식 최적화를 위한 연구들이 있었다. Yang과 Ciesielski는 XOR 게이트를 포함하는 논리식의 최적화를 위해 BDD를 이용한 방법인 BDS[5]를 제안하였다. Wu와 Zhu는 BDS 방법에서 영감을 얻어 Folded BDD[6]를 발표하였다. BDD 자체에 대한 연구로는 BDD를 확장한 Edge-valued Multi-valued Decision Diagram (EVMDD)[7]을 이용한 논리식 표현에 대한 연구가 최근 발표되었다. 한편으로는 논리합성 도구로부터 산출된 결과를 원하는 회로로 대응시키는 연구[8, 9]가 진행 되고 있다. 또한 반도체의 직접도 향상으로 하나의 칩에 완전한 시스템을 구현하기 위한 System on Chip (SoC)의 연구와 테스팅[10, 11]에 대한 연구가 진행되고 있다.

본 논문과 관련된 대입에 의한 논리식 최적화 연구로는 MIS(또는 SIS)에 적용된 대수 나눗셈에 의한 방법[2,3]과 Chang과 Cheng에 의한 방법[12]이 있다. [2,3]의 방법은 주어진 부울 공리를 적용하지 않고 하나의 논리식이 다른 논리식을 나눌 경우, 몫과 나머지를 찾아 논리식을 최적화하는 방법이다. Chang과 Cheng의 방법은 주어진 논리회로에 논리동작이 같도록 유지하면서 여분의 연결선이나 게이트를 추가하고 고착결함(stuck-at fault) 검사 기법을 응용하여 회로를 최적화하는 것이다. Chang과 Cheng의 실험 결과를 통해 많은 벤치마크회로에 대하여 MIS보다 좋은 결과를 산출하였으나, 여전히 일부 벤치마크회로에 대하여는 MIS가 보다 좋은 결과를 산출하였다.

본 논문에서는 선험 부울 대입 방법(heuristic Boolean resubstitution)을 제안한다. 제안하는 핵심 기술은 주어진 논리식들로부터 대입식 산출을 위한 행렬을 만든다. 다음 이 행렬에 부울 공리인 등멱법칙과 보수법칙을 적용하고, 행렬의 빈 공간을 채워나가는 방식으로 행렬을 확장하여 부울 대입식을 산출한다. 논문의 구성은 다음과 같다. 2장에서는 본 논문 서술에 필요한 정의와 대수 나눗셈에 의한 대입 방법에 대하여 서술하고, 3장에서 본 논문에서 제시하는 새로운 부울 대입 방법을 소개한다. 4장에서 실험결과를 보이고, 5장에서 결론을 제시한다.

2. 배경 지식

본 논문을 서술하는데 필요한 용어들과 대수 나눗셈 방법과 대입식 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 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 리터럴을 갖지 않은 경우 그 논리식은 큐브면제(cube-free) 되었다고 한다.

예 2: 논리식 $F=a+bc'$의 경우, $sup(F)=\{a, b, c\}$. 논리식 $ab+c$는 큐브 면제된 경우이나, 논리식 $ab+ac$ 및 abc는 큐브 면제된 것이 아니다.

정의 3: 논리회로는 비순환 방향 그래프(directed acyclic graph)로 표현되며, 각 노드i는 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 부울 곱이다. 또 2개의 논리식 F와 D에 대하여, F/D는 가장 큰 큐브 집합인 몫 Q를 산출하여, 논리식 F를 $F=QD+R$로 표현 할 수 있다. 여기서, R은 나머지를 나타낸다. QD가 대수 곱인 경우, F/D는 대수 몫을, 그 외의 경우 F/D는 부울 몫이 된다. $F/D=Q \neq \varnothing$ 이고 Q가 대수 나눗셈에 의해 산출된 경우, D는 대수 제수(algebraic divisor)라 하고, 그 외의 경우 부울 제수(Boolean divisor)라 한다. 주어진 2개의 논리식 F와 D에 대하여, 논리식 $F=QD+R$로 표현 가능하면, 논리식 F는 논리식 D의 대입으로 표현

가능하다고 한다.

예 3: $(a+b)(c+d)=ac+ad+bc+bd$는 대수 곱이며, $(a+b)(a+c)=a+bc$는 부울 곱이다. $F_1=ad+abc+bcd$ 이고 $D_1=a+bc$ 가 주어진 경우를 보자. F_1/D_1의 결과로는 대수 몫 d와 나머지 abc가 산출된다. 이 때, $a+bc$는 대수 제수가 된다. 또, $F_2=abg+acg+adf+aef+afg+bd+be+cd+ce$와 $D_2=ag+d+e$인 경우, $F_2=(af+b+c)(ag+d+e)$로 표현될 수 있다. 이 때, $af+b+c$는 부울 몫이며, $ag+d+e$는 부울 제수가 된다. $F_3=ac+ad+bc+bd+e$와 $D_3=c+d$인 경우 $F_3=(c+d)D_3+e$와 같이 논리식이 변형 가능할 경우 논리식 F_3가 논리식 D_3의 대입으로 표현된 경우이다.

2.2 대수 대입 방법(Algebraic Resubstitution)

주어진 2개의 논리식들의 대수 나눗셈을 수행하여 논리식들의 대입 관계를 찾기 위해 Brayton 등이 MIS에서 제시한 방법[2, 3]을 소개한다. 다음은 Brayton 등의 대수 나눗셈에 의한 대입식 산출 알고리즘을 보인 것이다.

Algorithm : 대수 나눗셈
입력 : 주어진 논리식 F와 G
출력 : F를 G로 나눈 후의 몫과 나머지
begin
  U = 논리식 F에서 논리식 G의
    리터럴을 포함하는 부분만 추출;
  V = 논리식 F에서 논리식 G의
    리터럴을 포함하지 않는 부분만 추출;
  /* F의 j번째 항이 u_jv_j 라고 하자 */
  $V_i=\{v_j\in V \mid u_j=g_i\}$;
  $Q=\cap V_i$;
  $R=F-GQ$;
  return (Q, R);
end

예 4: 다음과 같이 2개의 논리식 F와 G가 주어졌다고 하자. $F=ac+ad+bc+bd+e$, $G=a+b$. 논리식 F가 G를 포함하는 논리식으로 변형 가능한지는 다음과 같은 대수 나눗셈 방법으로 산출된다. 논리식 F의 각 항에서 논리식 G의 리터럴을 포함하는 부분만 추출하여 $U=a+a+b+b+1$를 산출한다. 논리식 F의 각 항에서 논리식 G의 리터럴을 포함하지 않는 부분만 추출하여 $V=c+d+c+d+e$를 산출한다. U의 각 항을 $u_1=a,\ u_2=a,\ u_3=b,\ u_4=b,\ u_5=1$으로, 또 V의 각 항을 $v_1=c,\ v_2=d,\ v_3=c,\ v_4=d,\ v_5=e$로 나타내자. 또, G의 항을 각각 $g_1=a,\ g_2=b$로 표시하자. 그러면, $u_1=a$이고 $g_1=a$가 되기 때문에 $v_1=c$가, $u_2=a$이고 $g_1=a$ 임으로 $v_2=d$를 산출하여, 이를 $V_1=c+d$ 라 하자. 또, $g_2=b$에 대하여 동일한 방법으로 $V_2=c+d$를 얻는다. 마지막으로 $Q=V_1\cap V_2$를 산출하면 $Q=c+d$가 산출되고, $R=F-GQ$를 산출하면 $R=e$를 얻게 된다. 이 결과를 활용하면 주어진 논리식은 다음과 같은 대수 대입식으로 변형된다.

$F=(c+d)G+e$
$G=a+b$

3. 제안한 부울 대입식 산출 방법

주어진 2개의 논리식을 행렬로 표현하고 부울 공리를 적용하여 부울 대입식을 찾는 방법을 제시한다.

3.1 부울 대입 행렬

2개의 논리식을 2차원 행렬 T로 표현하자. 이 행렬은 부울 대입식을 산출하기 위한 용도로 활용된다. 주어진 논리식이 F와 G라고 하고, 논리식 F가 논리식 G의 대입식으로 표현되는지 검사한다고 하자. 그러면, 행렬 T의 행은 G의 각 항과 대응되고, 열들은 F의 각 항들을 G의 각 항들로 나눌 경우 산출되는 몫에 대응하도록 만든다. 그리고, 행렬의 원소는 식 (1)에 따라 값이 산출된다. 수식 (1)에서 *index*( )는 주어진 논리식 F의 각 항에 대응하는 양의 정수 값을 산출하는 함수이며, *는 무관항(don't care)을 나타낸다.

$$T(g_i,k_j)=\begin{cases} index(g_ik_j) & \text{if } g_i\cdot k_j\in F \\ * & \text{if } g_i\cap k_j=\phi \end{cases} \quad (1)$$

예 5: 다음 2개의 논리식들이 주어졌을 경우 대입 방법에 의한 결과를 산출해 보자.

$$F=abcd+abce+ac'd+bc'd+abde'+bde'$$
$$G=ad+bd+ace$$

표1과 같이 논리식 F의 각 항에 0 보다 큰 정수 값을 배정한다. 다음 2 개의 논리식으로부터 부울 나눗셈을 위한 행렬을 만든다. 행렬의 각 행은 논리식 G의 각 항과

대응되도록 한다. 그러면 ad, bd, ace 에 대응되는 3개의 행이 산출된다. 다음 G의 각 항으로 F의 각 항을 나눌 경우 얻어지는 몫에 대응되는 열을 만든다. 따라서, G의 항 ad로부터 산출되는 몫들인 bc, c', be'에 대응되는 열을 추가하고 행렬에는 행과 열에 대응하는 식들의 논리곱(AND)과 일치하는 인덱스를 표1에서 찾아 입력한다. 유사하게 bd로 나눌 경우 산출되는 몫들인 ac, c', ae', e' 그리고 ace로 나눌 경우 산출되는 몫인 b로부터 행렬을 산출한다. 이렇게 산출된 행렬이 표2와 같은 대수 나눗셈 행렬이다. 다음 부울 나눗셈 행렬을 만든다. 표2의 각 열에 적어도 하나의 값이 배당되어 있으면 부울 공리 적용한다. 따라서, 표2의 2행 3열의 경우 $(bd)(be') = bde'$ 이 되기 때문에 2행 3열에는 $6 = index(bde')$, 3행 1열의 경우 $2 = index(abce)$가 입력된다. 반면에 3행 2열의 경우 $(ace)(c') = \phi$가 되기 때문에 3행 2열에는 *가 입력된다. 마찬가지로, 3행 3열, 3행 5열, 3행 6열의 경우도 *가 입력된다. 이렇게 완성된 것이 표3의 부울 나눗셈 행렬이다.

[표 1] 논리식 F의 인덱스(index) 배정

F의 항(f_i)	$abcd$	$abce$	$ac'd$	$bc'd$	$abde'$	bde'
$index(f_i)$	1	2	3	4	5	6

[표 2] 대수 나눗셈 행렬

G의 항(g_i) \ 몫(k_j)	bc	c'	be'	ac	ae'	e'	b
ad	1	3	5				
bd		4		1	5	6	
ace							2

[표 3] 부울 나눗셈 행렬

G의 항(g_i) \ 몫(k_j)	bc	c'	be'	ac	ae'	e'	b
ad	1	3	5				
bd		4	6	1	5	6	
ace	2	*	*		*	*	2

3.2 확장된 부울 나눗셈 행렬

부울 나눗셈 행렬에 리터럴을 추가하여 확장된 부울 나눗셈 행렬을 만든다. 확장된 행렬의 행은 부울 나눗셈 행렬의 것과 동일하나 열의 경우는 변경이 발생한다. 즉, 확장 전 부울 나눗셈 행렬의 열에 대응하는 항에 리터럴을 추가하고, 식(2)를 이용해서 행렬 XT의 원소 값을 산출한다.

$$XT(g_i,\ l_j) = \begin{cases} T(g_i,\ k_j) & \text{if } k_j = l_j \\ index(g_i \bullet l_j) & \text{if } k_j \subset l_j \end{cases} \quad (2)$$

예 6: 예 5에서 산출한 표3의 부울 나눗셈 행렬로부터 확장된 부울 나눗셈 행렬을 만든다. 표3의 열1을 보자. G의 항을 구성하는 리터럴은 $\{a,b,c,d,e\}$ 인데, 열 1의 리터럴들은 $\{b,c\}$ 이기 때문에 $\{a,b,c,d,e\} - \{b,c\} = \{a,d,e\}$의 원소 중에서 ad와 ace에 공통으로 사용된 리터럴은 a이므로 bc에 a를 곱한 abc로 수정한다. 그러면, 2행 1열은 $abcd = (bd)(abc)$가 되고$1 = index(abcd)$ 임으로, 2행 1열에 1을 삽입한다. 이렇게 확장된 것이 표4 이다.

[표 4] 확장된 부울 나눗셈 행렬

G의 항(g_i) \ 몫(l_j)	abc	c'	be'	ac	ae'	e'	b
ad	1	3	5				
bd	1	4	6	1	5	6	
ace	2	*	*		*	*	2

3.3 부울 대입식 산출

확장된 부울 나눗셈 행렬로부터 부울 대입식을 만든다. 주어진 2개의 논리식이 F와 G라고 하고, 논리식 F가 논리식 G를 포함한다고 가정하자. 그러면, 확장된 행렬의 행은 G의 각 항에 대응할 것이다. F를 G로 나누어 산출되는 몫은 행렬의 열이 0보다 큰 값이거나 *로 모두 채워져 있는 열에 대응되는 항들의 논리합 Q가 된다. 그리고, $F - GQ$를 산출하여 나머지 논리식 R을 구한다.

예 7: 표4를 이용해서 논리식 F와 논리식 G로부터 산출될 부울 대입식을 구한다. 표4에서 열이 모두 채워진 부분만을 선택한다. 첫째, 둘째, 셋째 열이 모두 채워져 있기 때문에 이 열들에 대응하는 항들의 논리합은 $(abc + c' + be')$이 되고, 논리식F의 각 항에 대응되는 인덱스 값 1부터 6이 모두 커버 된다. 따라서, F에 G를 대입하여 얻어지는 논리식은 다음과 같다.

$F = (abc + c' + be')G$

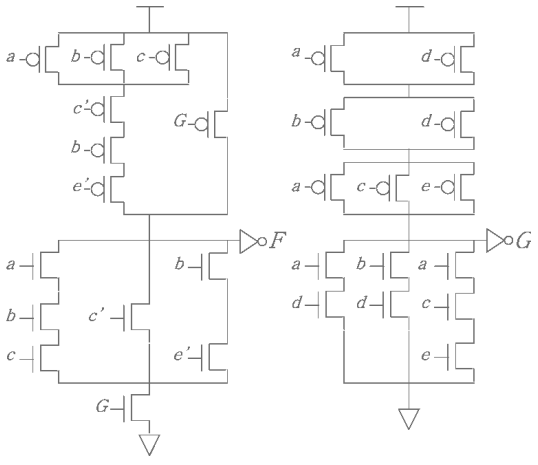
$G = ad + bd + ace$

따라서, 28개 리터럴로 구성된 논리식들은 부울 대입

[표 5] 대입 논리식 산출 비교 결과

회로	SIS		Chang과 Cheng (기본 나눗셈)		Chang과 Cheng (확장 나눗셈)		제안방법	
	리터럴 수	시간 (초)	리터럴수	시간 (초)	리터럴수	시간 (초)	리터럴 수	시간 (초)
alu2	446	18.6	412	5.2	417	5.3	367	4.9
alu4	829	78.8	779	15.8	784	15.0	775	16.6
apex6	807	1.2	796	3.4	795	3.7	795	2.5
apex7	278	0.2	263	0.6	263	0.7	263	0.5
dalu	2003	18.2	1753	52.9	1768	51.9	1748	23.4
des	4048	79.8	5712	167.9	5697	189.8	4023	156.3
i8	1817	12.1	905	63.3	915	64.3	1012	58.2
i9	750	2.1	729	14.7	729	16.8	731	10.6
rot	751	1.4	701	1.6	698	1.6	691	1.6
t481	2608	77.3	2009	31.9	1958	27.9	2004	40.2
C880	416	0.9	412	0.7	413	0.7	412	0.7
C1355	562	1.1	558	1.8	558	1.9	559	1.6
C1908	631	2.1	597	1.5	597	1.5	597	2.7
C2670	843	2.9	831	2.9	829	3.2	825	3.1
C5315	1961	6.9	1910	8.1	1914	8.3	1901	8.6
C6288	4212	35.8	3747	20.1	3747	21.2	3689	45.2
C7552	2522	15.2	2413	14.2	2410	15.3	2420	15.1

방법에 따라 14개의 리터럴의 논리식으로 된다. 산출된 논리식은 nMOS 14개, pMOS 14개로 구성된 그림 1의 CMOS 회로가 된다. 즉, 최적화 전의 논리식을 CMOS 회로로 구현하면 nMOS 28개, pMOS 28개가 필요하지만, 제안하는 방법을 적용할 경우는 nMOS 14개와 pMOS 14개를 줄이는 효과를 얻게 된다.



[그림 1] CMOS 회로

3.4 부울 대입식 산출 알고리즘

알고리즘은 주어진 논리식들로부터 확장된 부울 나눗셈 행렬을 만들고, 이를 이용해서 부울 대입 논리식을 산출한다. 알고리즘의 순서는 지금까지 서술한 단계와 같다. 다음 알고리즘에서 주어진 논리식이 F와 G인 경우, F가 G를 포함하는 경우와 G가 F를 포함하는 경우를 각각 구하고 이 중에서 리터럴 개수가 적은 부울 대입식을 선택한다.

Algorithm : 부울 대입식 산출
입력: 주어진 논리식 집합 ***F***
출력: 부울대입에 의한 간략화된 논리식 집합 ***F***
begin
  for each expression $F \in$ ***F*** **do**
    for each expression $G \in$ ***F*** **do**
    begin
      /* F를 피제수, G를 제수로 선택 */
      대수 나눗셈 행렬 T산출;
      행렬 T에 등멱, 보수 공리적용;
      확장된 부울 나눗셈 행렬 XT 산출;
      부울 대입식 E_1 산출;
      /* G를 피제수, F를 제수로 선택 */
      대수 나눗셈 행렬 T산출;
      행렬 T에 부울 공리 적용;
      확장된 부울 나눗셈 행렬 XT 산출;
      부울 대입식 E_2 산출;
      E_1과 E_2 중에서 리터럴 개수가

```
            적은 것 선택;
        end
    end
```

4. 실험 결과

Linux 운영체제하의 1.4GHz Pentium IV PC에서 제안한 방법을 C언어로 작성하였다. 실험결과에 대한 비교를 하기 위해서 ALU, 곱셈기(multiplier), 랜덤로직(random logic) 등 다양한 논리 회로로 구성된 Microelectronics Center of North Carolina(MCNC) 벤치마크 회로[13]와 IWLS2005의 벤치마크 회로[14]에 대하여 리터럴 개수와 수행시간을 기준으로 비교하였다. 참고로, 본 논문에서 제안한 방법은 선험 알고리즘이기 때문에 벤치마크 입력에 따른 리터럴 개수와 수행 시간을 측정한 결과로 알고리즘의 성능을 나타내었다. 성능 비교를 가장 널리 사용되는 합성도구인 SIS의 대입식 산출 결과와 비교하였다. 현재까지 부울 대입방법으로 가장 최근에 발표된 연구가 Chang과 Cheng의 연구[12]이기 때문에 본 실험 결과를 또한 이들의 방법과도 비교하였다. SIS에서 대입식을 산출하기 위해 "resub -d" 명령을 수행하여, 리터럴 개수와 수행시간을 산출하였다. 또한 Chang과 Cheng이 제시한 기본 나눗셈(basic division)에 의한 방법과 확장된 나눗셈(extended division)에 의한 방법으로 산출된 결과들과 비교하였다. 표5는 대입식 산출 결과를 정리한 것이다. 표5의 첫 번째 열은 벤치마크회로의 이름을 나타낸 것이다. 두 번째와 세 번째 열은 SIS의 대수 나눗셈에 의한 대입식 산출 결과를 보인 것이다. 네 번째에서 일곱 번째 열은 Chang과 Cheng의 기본 나눗셈과 확장 나눗셈에 의한 대입식 산출결과이다. 본 논문에서 제시한 방법으로 산출된 대입식 결과는 여덟 번째와 아홉 번째 열에 나타내었다. 본 논문에서 제시한 방법으로 SIS 보다 좋은 결과를 산출하였다. 또한 Chang과 Cheng의 방법과 비교했을 경우도 리터럴 개수나 수행 시간에서 많은 경우 좋은 결과를 산출할 수 있었다. 벤치마크회로 i8에서 Chang과 Cheng의 방법이 보다 좋은 결과를 산출했는데, 이유는 여분의 연결선(redundant wire)을 추가함으로써 부울 나눗셈이 되는 경우가 발생하기 때문인 것으로 판단된다.

5. 결론

논리회로의 출력수가 2개 이상의 경우 부울 대입 방법에 의한 논리회로를 최적화 방법을 제시하였다. 제시한 방법은 대수 나눗셈을 이용한 행렬에 부울 공리를 적용하고, 또한 행렬의 열 단위로 빈 공간을 채우도록 리터럴을 추가하는 방법을 제시하였다. 실험결과에서 보인 바와 같이 제시한 방법으로 우수한 결과를 얻었다. 결론적으로 논리회로 최적화를 위해서 Chang과 Cheng 방법과 제안한 방법을 함께 활용할 경우 실용성이 매우 높을 것으로 판단된다.

참고문헌

[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.

[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] J. Rajski and J. Vasudevamurthy, "The Testability-Preserving Concurrent Decomposition and Factorization of Boolean Expressions," *IEEE Trans. CAD*, Vol. 11, No. 6, pp. 778-79, 1992.

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

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

[7] S. Nagayama and T. Sasao, "Representation of Elementary Functions Using Edge-Valued MDDs," *Proc. of the 37th International Symposium on Multiple-Valued Logic(ISMVL '07)*, pp. 5-11, 2007.

[8] J. Cong and K. Minkovich, "Optimality Study of Logic Synthesis for LUT-Based FPGAs," *IEEE Trans. CAD,* Vol. 26, No. 2, pp. 230-239, 2007.

[9] A. C. Ling, P. Singh, and S. D. Brown, "FPGA PLB Architecture Evaluation and Area Optimization Techniques Using Boolean Satisfiablity," *IEEE Trans. CAD,* Vol. 26, No. 7, pp. 1196-1210, 2007.

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

[11] 정준모 “고속 SoC 검증의 위한 자동 가상 플랫폼 생

성”, 한국산학기술학회 논문지, 제9권, 제5호, pp. 1139-1144, 2008.
[12] S.-C. Chang and D. I. Cheng, "Efficient Boolean Division and Substitution Using Redundancy Addition and Removing," *IEEE Trans. CAD*, Vol. 18, No. 8, pp. 1096-1106, 1999.
[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 전자컴퓨터통신학부 교수

<관심분야>
회로설계자동화, 논리합성