論文93-30A-12-13

SHARE: 효율적인 출력 PHASE 할당을 이용한 PLA 최소화

(SHARE: PLA minimization based on an efficient output phase assignment)

李 永 浩*, 鄭 正 和*

(Young Ho Lee and Jong Wha Chong)

要 約

다출력 함수를 PLA로 구현할 때 정논리 또는 부논리로 각 출력을 구현할 수 있다. 이러한 자유를 이용하면 적항의 수를 더욱 줄일 수 있다. 본 논문은 효율적인 출력 phase 할당을 이용하여 PLA의 면적을 최소화하는 알고리듬을 제안한다. 이 알고리듬은 출력들 사이에 공유되는 적항 (product term)의 수가 최대화 되고, 특정 출력에만 사용되는 적항의 수가 최소화 되도록 출력의 phase를 결정한 후 결정된 출력 phase에 따라 PLA를 최소화한다. 이 알고리듬은 Sun4/330상에 C 언어로 구현되었으며, 56개의 예제 회로에 대하여 기존의 알고리듬과 비교되었다. 제안된 알고리듬은 18-32개의 PLA에 대하여 우수한 결과를, 1-5개의 PLA에 대하여 열등한 결과를, 23-33개의 PLA에 대하여 동일한 결과를 얻었다. 더구나, 계산 시간은 기존의 방법보다 훨씬 짧았다.

Abstract

When realizing a multiple-output function by a PLA, there is often the flexibility to choose either uncomplementary or complementary phase for each output. In this case, it is possible to significantly reduce the number of product terms by fully exploiting the freedom. This paper presents a PLA phase assignment algorithm called SHARE. The algorithm assigns an appropriate phase for each output of a multiple-output function so that the product terms for each output are maximally shared with the other outputs, and then minimizes the multiple-output function with the assigned output phase assignment. The algorithm has been implemented on Sun4/330 in C language and compared with the previous approaches for 56 example PLA's. The proposed algorithm has obtained better results for 18-32 PLA's, worse results for 1-5 PLA's, and the same results for 23-33 PLA's. Moreover, the computation time is much less than for the previous approaches.

*正會員, 漢陽大學校 CAD 및 通信 回路 硏究室
(CAD & Communication Circuit Lab., Hanyang Univ.)
接受日字 : 1992年 10月 15日

I. 서론

PLA는 규칙적 구조와 짧은 turnaround 시간 때문에 ASIC (Application Specific Integrated Circuit) 설계에 적합하여 VLSI 설계에 오래전부터

사용되어 왔으며, 최근에는 다단 (multi-level) 회로가 보편화되어 이에 대한 연구가 활발히 진행되고 있다. 이단 (two-level) 논리 회로의 최소화는 PLA로 논리회로를 효율적으로 구현하기 위하여 필요할 뿐만 아니라 다단 논리회로의 설계를 위한 준비단계로서 중요하다.

이단 (two-level) 논리회로 설계를 위한 툴은 ESPRESSO [1], MINI [2], 및 PRESTO [3]를 비롯하여 많은 툴이 개발되어 왔다. PLA 최소화 문제는 NP-complete 문제로 알려져 있으므로 대부분의 툴들은 효율적인 휴리스틱 알고리듬에 바탕을 두고 있다. 이러한 다양한 툴들 중에서 ESPRESSO가 가장 잘 알려져 있으며 그 기본 전략은 local optima solution을 구한 후 global optima solution을 얻기 위하여 local optima solution을 점차 개선하는 것이다.

다출력 함수를 PLA로 구현할 때 정논리 또는 부논리로 각 출력을 구현할 수 있다. 왜냐하면 MPLA (Mask Programmable Logic Array) [4]에서 부논리를 가진 출력이 필요한 경우 인버터를 이용하면 설계를 쉽게 변경할 수 있고, FPLA (Field Programmable Logic Array) [5]에는 퓨즈가 부가되어 있어서 정논리 또는 부논리로 출력을 구현할 수 있기 때문이다. 따라서, 이러한 정논리 또는 부논리로 출력을 구현할 수 있는 자유를 충분히 활용하면 PLA 최소화에서 적항 (product term)의 수를 효과적으로 줄일 수 있다. 이러한 자유를 PLA 최소화에 효과적으로 이용하기 위하여 효율적 알고리듬의 개발이 요구된다. 그러나, 불행히도 m개의 출력을 갖는 논리함수에 대하여 2^m 개의 서로 다른 출력 phase 할당이 있을 수 있어 모든 출력 phase 할당에 대하여 PLA를 최소화하는 것이 불가능할 뿐 아니라 PLA 최소화 프로그램을 이용하지 않고 최적의 출력 phase를 찾기는 어렵다.

논리회로의 출력 phase 할당 문제는 이단 논리회로를 위한 출력 phase 할당 문제와 다단 논리회로를 위한 출력 phase 할당 문제로 대별된다. 이단 논리회로를 위한 출력 phase 할당에서는 적항 (product term) 수가 최소화 되도록 논리회로의 출력 phase를 결정하는 것이 우선 과제이고, 다단 논리회로를 위한 출력 phase 할당에서는 리터럴 (literal) 수를 최소화하는데 그 목적이 있다. 본 논문에서는 이와같이 근본적으로 다른 문제를 동시에 해결하려고 시도하는 대신에 이단 논리회로를 위한 출력 phase 할당 문제에 국한시켜 설명하고자 한다.

이단 논리 회로를 위한 출력 phase 할당 문제를 해결하기 위한 방법으로서 대표적인 두 방법 즉, Sasao의 방법 [6]과 PLAYGROUND [7]을 살펴보면 다음과 같다. Sasao의 방법 [6]은 m개의 출력을 가진 논리함수에 적당한 출력 phase를 할당하기 위하여 2m개의 출력, 즉 $f_1, f_2, \ldots, f_m, \overline{f_1}, \overline{f_2}, \ldots, \overline{f_m}$를 가진 이중 phase 특성 함수를 최소화한 후 최소화된 PLA의 출력들을 조사한다. PLAYGROUND [7]는 m개의 출력을 가진 논리함수에 적당한 출력 phase를 할당하기 위하여 2m개의 출력, 즉 $f_1, f_2, \ldots, f_m, \overline{f_1}, \overline{f_2}, \ldots, \overline{f_m}$를 가진 병합 PLA를 구하여 최소화한 후 최소화된 PLA의 출력들을 조사하여 출력 phase를 결정하는 과정을, PLA 면적이 감소되지 않을 때까지 반복 수행한다. 이러한 알고리듬들은 m개의 출력을 갖는 논리함수에 적절한 출력 phase를 할당하기 위하여 2m개의 출력을 갖는 논리 함수를 최소화해야 하므로 많은 입출력을 가지는 논리함수에 적절한 출력 phase를 할당하는데 과대한 시간이 걸린다.

본 논문은 효율적인 출력 phase 할당을 이용하여 PLA의 면적을 최소화하는 알고리듬을 제안한다. 이 알고리듬은 출력들 사이에 공유되는 적항의 수가 최대화되고, 특정 출력에만 사용되는 적항의 수가 최소화 되도록 출력의 phase를 결정한 후, 결정된 phase에 따라 PLA를 최소화한다. 이 알고리듬은 Sun4/330상에 C 언어로 구현되었으며, 56개의 예제 회로에 대하여 Sasao의 방법 [6] 및 PLAYGROUND [7]와 비교되었다. 제안된 알고리듬은 18-32개의 PLA에 대하여 우수한 결과를, 1-5개의 PLA에 대하여 열등한 결과를, 23-33개의 PLA에 대하여 동일한 결과를 얻었다. 더구나, 계산 시간은 기존의 방법보다 훨씬 짧았다.

본 논문은 다음과 같이 구성되어 있다. Ⅱ절에서는 용어 정의 및 표현법에 대하여 언급하고, Ⅲ절에서는 출력 phase 할당에 관하여 논한 후, 출력 phase 할당을 위한 효율적인 휴리스틱 알고리듬을 기술한다. Ⅳ절에서 제안된 알고리듬의 실험결과를 제시하고 기존의 방법들과 비교한 후 Ⅴ절에서 결론을 내리고 앞으로의 연구과제를 제시한다.

Ⅱ. 용어 정의 및 표현법

이 절에서는 차후에 사용될 용어 정의 및 표현방법에 대하여 기술한다. PLA로 m개의 출력을 가지는 논리함수 ff를 구현할 때 각 출력을 정논리 또는 부논리로 구현하는 것이 가능하다.

논리함수 ff의 각 출력을 구현하는데 사용한 논리를 나타내는 출력 phase는 phase minterm p =

$[p_1, \cdots, p_m] \in \{0,1\}^m$를 이용하여 표현할 수 있는데 $p^j = 1$이면 ff_j는 정논리, $p_j = 0$이면 ff_j는 부논리로 실현된다. 여기서, m은 논리함수 ff의 출력의 수, ff_j는 ff의 j번째 출력, p_j는 ff의 j번째 출력인 ff_j의 phase를 나타낸다. phase cube $p = [p_1, \cdots, p_m] \in \{0,1,2\}^m$는 몇개의 phase minterm으로 구성된다. phase minterm이나 phase cube에서 0은 부논리, 1은 정논리를 나타내고, 2는 정논리와 부논리를 동시에 나타낸다. 논리함수 ff의 구현에 사용할 ff의 출력 phase를 선택하면, 이 출력 phase는 위에서 정의된 phase minterm p를 이용하여 표현가능하고, phase minterm p에 의해 일의적으로 결정되는 출력 phase를 가지고 최소화된 PLA는 ff^p로 표현할 수 있다.

예를들면, 그림 1(a)로 주어진 논리함수 ff의 모든 출력을 정논리로 실현한 것은 그림 1(b)처럼 나타내는데, $ff^{[1,1,1]}$의 윗첨자 [1,1,1] 은 논리함수 ff의 각 출력을 정논리로 구현했음을 의미한다. 같은 방법으로 그림 1(a)로 주어진 논리함수 ff의 모든 출력을 부논리로 실현한 것은 그림 1(c)처럼 나타내는데, $ff^{[0,0,0]}$의 윗첨자 [0,0,0] 은 논리함수 ff의 각 출력이 부논리로 구현했음을 의미한다. 또, 그림 1(d)는 논리함수 ff의 첫번째 출력은 부논리, 두번째 출력은 정논리, 세번째 출력은 부논리로 실현한 경우를 나타내며, $ff^{[0,1,0]}$의 윗첨자인 [0,1,0] 가 이와같은 사실을 표현하고 있다.

최소화된 PLA ff^p에 대하여, $n(ff^p)$는 ff^p의 적항의 수, $n(ff^{pj})$는 ff^p의 j번째 출력을 구현하는데 사용한 적항 (product term)의 수, $m(ff^{pj})$는 ff^p의 j번째 출력을 구현하는데만 사용되고 그 외의 출력인 i ($i \neq j$)번째 출력을 구현하는데는 사용되지 않는 적항 (product term)의 수이다.

$$ff = \begin{bmatrix} \text{--1 100} \\ \text{00- 100} \\ \text{00- 010} \\ \text{10- 101} \\ \text{-1- 001} \\ \text{11- 010} \end{bmatrix} \qquad ff^{[1,1,1]} = \begin{bmatrix} \text{--1 100} \\ \text{00- 110} \\ \text{10- 101} \\ \text{-1- 001} \\ \text{11- 010} \end{bmatrix}$$

(a) (b)

$$ff^{[0,0,0]} = \begin{bmatrix} \text{-10 100} \\ \text{00- 001} \\ \text{10- 010} \\ \text{01- 010} \end{bmatrix} \qquad ff^{[0,1,0]} = \begin{bmatrix} \text{-10 100} \\ \text{00- 011} \\ \text{11- 010} \end{bmatrix}$$

(c) (d)

그림 1. 다양한 출력 phase를 이용하여 구현 가능한 PLA들

Fig. 1. Possible PLA implementations with several output phase assignments.

Ⅲ. 출력 PHASE 할당

이 절에서는 출력 phase 할당 문제에 관하여 논한 후, 효율적인 출력 phase 할당 알고리듬을 제안한다. 제안되는 알고리듬의 아이디어는 다음 예제를 사용하여 설명된다.

예제 1: 그림 1(a)에 있는 3 입력과 3 출력을 가진 논리함수 ff는 임의의 출력 phase, 즉 phase minterm p를 가지고 구현될 수 있다. 여러 phase minterm들인 $p_1 = [1,1,1]$, $p_2 = [0,0,0]$, 및 $p_3 = [0,1,0]$ 를 가지고 구현된 PLA들이 그림 1(b), 그림 1(c), 그림 1(d)에 각각 표시되어 있다. 상기의 세 가지 PLA들 중에서 가장 적은 적항을 가지는 PLA는 phase minterm이 p_3인 PLA이다.

위의 예제에서 알 수 있는 바와 같이 논리함수의 모든 출력을 정논리로 구현하기 보다는 적절한 출력 phase를 할당하여 PLA로 구현함으로써 적항의 수를 더 줄일 수 있다. 따라서, 다출력 함수에 대하여 최적의 출력 phase를 결정하는 문제는 매우 중요하다. 출력 phase 할당 문제는 다음과 같이 정의될 수 있다.

문제 1: 적항의 수가 최소인 PLA가 얻어지도록 논리함수에 출력 phase를 할당하라.

출력 phase 할당 문제를 해결하기 위한 가장 간단한 방법은 모든 가능한 출력 phase에 대하여 PLA 최소화 툴을 적용하는 것이다. 그러나, 이 방법은 극히 적은 수의 출력을 가진 논리함수에 대해서만 유효할 뿐 많은 출력을 갖는 논리함수에 대해서는 적절한 해결책이 될 수 없다. 왜냐하면, m개의 출력을 갖는 논리함수에 대하여 서로 다른 2^m개의 출력 phase 할당이 가능하기 때문이다. 따라서, 효율적인 출력 phase 할당 방법이 필요하다. 본 논문의 목적은 SHARE라 불리우는 효율적인 출력 phase 할당 알고리듬을 제안하는 것이다. 이 알고리듬의 아이디어는 출력들 사이에 공유되는 적항의 수가 최대화되고, 특정 출력에만 사용되는 적항의 수가 최소화 되도록 각 출력의 phase를 결정하는 것이다. 이러한 아이디

어에 근거를 두고 $ff^{[0,0,\cdots,0]}$와 $ff^{[1,1,\cdots,1]}$의 출력들을 조사함으로써 다출력 논리함수의 각 출력을 위한 적절한 논리를 정할 수 있다. 회로를 최소화하는데 있어서 논리함수에 존재하는 don't care를 효과적으로 이용하느냐 못하느냐에 따라 회로의 크기가 줄어들기도 하고 늘어나기도 한다. 회로의 면적을 더욱 줄이기 위해서 필요한 출력 phase 할당에 있어서도 논리함수의 don't care를 고려하여 출력 phase를 할당하느냐 고려하지 않고 할당하느냐에 따라서 회로의 최소화에 미치는 영향이 달라지고, 최종적으로는 회로의 크기가 달라진다. 따라서, 본 논문에서는 출력 phase를 할당함에 있어서 논리함수의 don't care를 고려하고자 했으며, 본 논문에서 제안하는 알고리듬에서는 $ff^{[0,0,\cdots,0]}$와 $ff^{[1,1,\cdots,1]}$를 구할때, don't care를 고려하는 이단 논리 최소화기[1]를 이용하기 때문에 논리함수의 don't care가 고려된다. 따라서, 본 논문에서 제안하는 알고리듬은 논리함수에 don't care가 있는 경우와 없는 경우 모두에 효과적으로 사용될 수 있다.

예제 2: 그림 1(a)에 있는 3 입력과 3 출력을 가진 논리함수 ff에 대하여 출력 phase minterm [0,0,0] 과 [1,1,1] 을 가지고 논리함수를 최소화하면 그림 1(b)와 그림 1(c)에 있는 $ff^{[1,1,1]}$와 $ff^{[0,0,0]}$이 얻어진다. $ff^{[1,1,1]}$와 $ff^{[0,0,0]}$로부터 논리함수 ff의 j번째 출력을 구현할 논리 즉, j번째 출력의 phase를 결정하기 위하여 아래표를 만든다. 이 표에서 j열에 기록한 수는 몇 번째 출력인가를 나타내는 번호이고, $n(ff_j^{[1,1,1]})$ 아래에 기록한 수는 논리함수 ff의 모든 출력을 정논리로 구현했을 때, ff의 j번째 출력을 구현하기 위하여 필요한 적항 (product term)의 수이다. $n(ff_j^{[0,0,0]})$ 아래에 기록한 수는 논리함수 ff의 모든 출력을 부논리로 구현했을 때, ff의 j번째 출력을 구현하기 위하여 필요한 적항 (product term)의 수이다. 또, $m(ff_j^{[1,1,1]})$ 아래에 기록한 수는 논리함수 ff의 모든 출력을 정논리로 구현했을 때, ff의 j번째 출력을 구현하기 위해서만 사용되고, 그외의 출력을 구현하는데는 사용되지 않는 적항 (product term)의 수이다. $m(ff_j^{[0,0,0]})$ 아래에 기록한 수는 논리함수 ff의 모든 출력을 부논리로 구현했을 때, ff의 j번째 출력을 구현하기 위해서만 사용되고, 그외의 출력을 구현하는데는 사용되지 않는 적항 (product term)의 수이다.

j	$n(ff_j^{[1,1,1]})$		$n(ff_j^{[0,0,0]})$	$m(ff_j^{[1,1,1]})$		$m(ff_j^{[0,0,0]})$
1	3	>	1	1	=	1
2	2	=	2	1	<	2
3	2	>	1	1	=	1

출력들 사이에 공유되는 적항의 수를 최대화하기 위해서 $m(ff_2^{[1,1,1]})$이 $m(ff_2^{[0,0,0]})$보다 작기 때문에 ff_2는 정논리로 구현하는 것이 바람직하다. 각 출력의 적항의 수를 최소화하기 위해서 $n(ff_1^{[1,1,1]})$는 $n(ff_1^{[0,0,0]})$보다 크고, $n(ff_3^{[1,1,1]})$는 $n(ff_3^{[0,0,0]})$보다 크기 때문에 ff_1와 ff_3는 부논리로 구현하는 것이 좋다. 즉, 출력들 사이에 공유되는 적항의 수를 최대화함과 동시에 각 출력의 적항의 수를 최소화하기 위해서 ff_1와 ff_3는 부논리, ff_2는 정논리로 구현하는 것이 바람직하다.

위와 같은 관찰에 기초하여, 다음과 같이 C^1_j와 C^0_j를 정의하고, 이를 이용하여 출력들 사이에 공유되는 적항의 수를 최대화함과 동시에 각 출력의 적항의 수를 최소화할 출력 phase를 선택한다. 그 구체적 방법은 알고리듬 1과 2에서 자세히 설명한다.

$$C^1_j = n(ff_j^{[1,\cdots,1]}) + m(ff_j^{[1,\cdots,1]}),$$
$$C^0_j = n(ff_j^{[0,\cdots,0]}) + m(ff_j^{[0,\cdots,0]}).$$

알고리듬 1은 효율적 출력 phase 할당 알고리듬을 이용한 PLA 최소화 알고리듬이다. 알고리듬 1에서는 SHARE라 불리우는 알고리듬 2를 이용하여 하나의 phase cube p를 결정한 후, 이 phase cube p에 속하는 각 phase minterm c을 출력 phase로하여 논리함수 ff를 최소화하여 ffc를 얻는다. 최소 면적의 PLA를 얻는다.

알고리듬 1:

/* $C^1_j = n(ff_j^{[1,\cdots,1]}) + m(ff_j^{[1,\cdots,1]})$,
$C^0_j = n(ff_j^{[0,\cdots,0]}) + m(ff_j^{[0,\cdots,0]})$.
*/

1. 논리함수 ff에 대하여, $ff^{[1,\cdots,1]}$와 $ff^{[0,\cdots,0]}$를 구하라.
2. 논리함수 ff의 각 출력에 대하여, C^1_j와 C^0_j를 계산하라.
3. SHARE를 이용하여 phase cube p를 구하라.
4. $c \subseteq p$인 각 phase minterm c에 대하여, 최소화된 PLA ff^c를 구하라.
5. $ff^{[1,\cdots,1]}$, $ff^{[0,\cdots,0]}$, 및 ff^c, $c \subseteq p$ 인 PLA들 중에서 적항의 수가 최소인 PLA를 선택하라.

알고리듬 2는 본 논문의 핵심 알고리듬으로서 이 알고리듬의 기술에 사용된 기호는 Ⅱ절에서 정의된 바 있다. 알고리듬 2의 자세한 설명은 다음과 같다.

스텝 2에서는 논리함수 ff의 모든 출력을 정논리로

실현했을 때와 부논리로 실현했을 때, 각 출력을 구현하는데 필요한 적항 (product term)의 수를 비교하여 출력 phase를 결정한다. 논리함수 ff의 j번째 출력을 정논리로 구현했을 때의 적항 (product term)의 수 ($n(ff_j^{[1,\cdots,1]})$)가 부논리로 구현했을 때의 적항의 수 ($n(ff_j^{[0,\cdots,0]})$)보다 작으면 $c_j = 1$, 크면 $c_j = 0$, 같으면 $c_j = 2$로 결정한다. 여기서, 0, 1, 2의 의미는 Ⅱ절에서 정의한 것과 같다. 이와같은 방법으로 j번째 출력의 phase cj를 모으면 논리함수 ff의 출력 phase cube p_n을 얻는다.

스텝 3에서는 논리함수 ff의 모든 출력을 정논리로 실현했을 때와 부논리로 실현했을 때, 특정 출력 (논리함수 ff의 j번째 출력)을 구현하는데만 필요한 적항 (product term)의 수를 비교한다. 여기서 주의할 것은, 스텝 2에서 특정 출력 (논리함수 ff의 j번째 출력)을 구현하는데 필요한 모든 적항 (product term)의 수를 비교하는 것과는 달리, 스텝 3에서는 특정 출력 (논리함수 ff의 j번째 출력)을 구현하는데만 사용되고

알고리듬 2: (SHARE)

/* m = 논리함수 ff의 출력의 수,
$C^1_j = n(ff_j^{[1,\cdots,1]}) + m(ff_j^{[1,\cdots,1]})$,
$C^0_j = n(ff_j^{[0,\cdots,0]}) + m(ff_j^{[0,\cdots,0]})$,
$NP = (n(ff^{[1,\cdots,1]}) + n(ff^{[0,\cdots,0]})) / 2$,
n(p) = phase cube p에 있는 2의 수,
R = 0.06,
N = 4.
*/

1. Tolerance = R * NP
2. phase cube $p_n = [c_1, \cdots, c_m]$ 를 구하라.
여기서 $WF_6 + SiH_4 + C \xrightarrow{\Delta E} W + (Si_X H_y F_Z)_{products} + C$

3. phase cube $p_m = [c_1, \cdots, c_m]$ 를 구하라.
여기서,
$$c_j = \begin{cases} 1 & \text{if } m(ff_j^{[1,\cdots,1]}) < m(ff_j^{[0,\cdots,0]}), \\ 0 & \text{if } m(ff_j^{[1,\cdots,1]}) > m(ff_j^{[0,\cdots,0]}), \\ 2 & \text{otherwise.} \end{cases}$$

4. phase cube $p_c = [c_1, \cdots, c_m]$ 를 구하라.
여기서,
$$c_j = \begin{cases} 1 & \text{if } \|(C^1_j - C^0_j)\| > \text{Tolerance and } C^1_j < C^0_j, \\ 0 & \text{if } \|(C^1_j - C^0_j)\| > \text{Tolerance and } C^1_j < C^0_j, \\ 2 & \text{otherwise.} \end{cases}$$

5. If($n(p_c) \le N$)$p = p_c$
else if($n(p_c) > N$ && Tolerance > 0)
Tolerance = Tolerance − 1, goto Step 4.
else if($n(p_c) > N$ && Tolerance = 0)
if($n(p_n) \le N$)$p = p_n$
else if($n(p_m) \le N$)$p = p_m$
else $p = \phi$

나머지 출력을 구현하는데는 사용되지 않는 적항 (product term)의 수만을 비교한다 것이다. 논리함수 ff의 j번째 출력을 정논리로 구현했을 때 j번째 출력에만 사용되는 적항 (product term)의 수 ($m(ff_j^{[1,\cdots,1]})$)가 부논리로 구현했을 때 j번째 출력에만 사용되는 적항의 수 ($m(ff_j^{[0,\cdots,0]})$)보다 작으면 $c_j = 1$, 크면 $c_j = 0$, 같으면 $c_j = 2$로 결정한다. 이와같은 방법으로 j번째 출력의 phase cj를 모으면 논리함수 ff의 출력 phase cube pm을 얻는다.

스텝 4에서는 논리함수의 출력 phase를 결정하기 위하여 스텝 2의 평가기준 ($n(ff_j^{[1,\cdots,1]})$, $n(ff_j^{[0,\cdots,0]})$)과 스텝 3의 평가기준 ($m(ff_j^{[1,\cdots,1]})$, $m(ff_j^{[0,\cdots,0]})$)을 병용한다. 즉, $C^1_j = n(ff_j^{[1,\cdots,1]}) + m(ff_j^{[1,\cdots,1]})$, $C^0_j = n(ff_j^{[0,\cdots,0]}) + m(ff_j^{[0,\cdots,0]})$를 사용한다. 논리함수 ff의 j번째 출력 phase를 결정함에 있어서 C^1_j와 C^0_j의 차이가 별로 없을때, 즉 C^1_j와 C^0_j의 차이가 Tolerance보다 작을때는 그 해당 출력을 정논리로 구현해야할지 부논리로 구현해야할지 애매모호하므로 본 논문에서는 허용오차인 Tolerance를 도입했다. 또, Tolerance의 값은 여러차례의 실험을 통하여 스텝 1의 계산식에 의해 주어지는 값으로 정한바, 허용 오차 Tolerance는 $n(ff^{[1,\cdots,1]})$와 $n(ff^{[0,\cdots,0]})$의 평균값인 NP에 R (현재는 0.06)을 곱한 것이다. 하나의 값으로 정하지 않은 이유는 회로에 존재하는 적항 (product term)의 수에 따라 Tolerance의 조정이 필요한 것으로 생각되었기 때문이다. 이와같이 결정된 Tolerance를 이용하여 스텝 4에서와 같은 방법으로 논리함수 ff의 출력 phase p_c를 결정함으로서 특정함수에 사용되는 적항의 수와 특정함수에만 사용되는 적항의 수를 효과적으로 고려하고자 했다.

스텝 5에서는 스텝 2, 3, 4에서 결정된 phase cube p_n, p_m, p_c들중에서 논리함수 ff의 출력 phase를 선택한다. phase cube p_c에 있는 2의 수가 N개 (현재는 4개) 이하이면 논리함수 ff의 출력 phase로 p_c를 선택한다. 출력 phase cube에 있는 2의 수를 N개 이하로 제약한 이유는 다음과 같다. phase cube에 있는 2는 해당 출력을 어떤 논리로 실현할 것인가를 결정하지 못했음을 의미하므로 면적이 최소인 PLA를 얻기 위해서는 2N개의 출력 phase에 대

하여 논리 최소화를 행한 후 면적이 최소인 PLA를 선택할 필요가 있다. 따라서, PLA 최소화에 소모되는 계산 시간을 줄이기 위해서는 N의 크기를 제한할 필요가 있다. 이와같은 배경에서 본 알고리듬은 N을 4로 정하여 어떠한 경우에도 2^4 (16)번 이하의 PLA 최소화를 보장한다. phase cube p_c에 있는 2의 수가 N (현재는 4)보다 크면, Tolerance를 1씩 감소시킨 후 스텝 4를 다시 수행하여 새로운 phase cube p_c를 얻고 p_c에 있는 2의 수를 재조사한다. 이와같은 과정을 p_c에 있는 2의 수가 N 이하일 때까지 반복하여 2의 수가 N개 이하가 되면 p_c를 논리함수 ff의 출력 phase로 선택한다. 이러한 반복 과정중에 Tolerance가 0이 되어도 p_c에 있는 2의 수가 N개 이하로 되지 않으면 p_n, p_m 중에서 p_n에 우선 순위를 두고 논리함수 ff의 출력 phase를선택한다. 이때 p_n이나 p_m에 있는 2의 수는 N개 이하이어야 한다. 만약 상기의 조건을 만족하는 phase cube가 존재하지 않으면 출력 phase cube p = ø 된다.

Ⅳ. 실험결과

이 절에서는 제안된 알고리듬 (SHARE)의 유용성을 입증하기 위하여 실험결과를 제시한다. SHARE 프로그램은 Sun4/330상에서 C언어로 구현되고 실행되었으며, PLA 최소화 툴로써는 ESPRESSO를 이용하였다. 원칙적으로, 본 논문에서 제안한 SHARE 알고리듬의 개념은 ESPRESSO 뿐만 아니라 어떠한 PLA 최소화 툴과도 결합되어 사용될 수 있다.

Sasao 방법의 결과는 ESPRESSO에 옵션 "-do opo"를 주고 Sun4/330상에서 실행시켜 얻은 것이다. PLAYGROUND 프로그램은 구할 수 없어서 실험결과는 [7] 에서 인용한 것이다. [7] 에서의 실험은 VAX8600에서 행해졌다.

표 Ⅰ- Ⅴ는 [1] 에 있는 56개의 PLA에 대한 실험 결과이다. 1열은 PLA명, 2열은 입력 수, 3열은 출력 수, 4열은 Sasao 방법 [6] 으로 얻은 PLA의 적항 수, 5 열은 PLAYGROUND [7] 로 얻은 PLA의 적항의 수, 6열은 본 논문에서 제안한 SHARE로 얻은 PLA의 적항수, 7열은 모든 가능한 출력 phase에 대하여 ESPRESSO를 적용하여 얻은 PLA의 적항의 수이다. 별표 "*"는 해당하는 열의 프로그램으로 5시간의 Sun4/330 또는 VAX8600 CPU 시간내에 최소화된 PLA를 얻을 수 없었음을 나타낸다. 대시 "-"는 해당하는 행의 PLA에 대하여 실험을 하지 않았음을 나타낸다.

표 Ⅰ은 SHARE 프로그램이 12개의 예제 PLA에 대하여 최소면적의 PLA를 생성함을 보여준다. 표 Ⅱ는 SHARE 프로그램이 20개의 PLA에 대하여 Sasao의 방법보다 적은 면적의 PLA를 생성함을 보여준다. 표 Ⅲ은 SHARE 프로그램이 6개의 PLA에 대하여 PLAYGROUND보다 적은 면적의 PLA를 생성함을 보여준다. 표 Ⅳ는 Sasao의 방법, PLAY-

표 1. SHARE가 가장 우수한 결과를 갖는 회로

Table 1. Circuits for which SHARE has the best results.

Name	ni	no	espresso -do opo	PLAY GROUND	SHARE	exhaustive solution
alu3	10	8	42	47	37	37
chkn	29	7	173	138	136	136
spa	24	109	*	*	143	-
dk48	15	17	18	14	13	-
exep	30	63	*	66	67	-
in2	19	10	141	125	121	114
in7	26	10	46	43	42	33
mish	94	43	*	*	52	-
opa	17	69	79	76	74	-
rdoc	8	5	53	48	48	48
x1dn	27	6	*	110	103	103
x9dn	27	7	*	107	103	103

* : 5시간의 CPU 시간으로 결과를 얻을 수 없었음.
- : 실험을 행하지 않았음.

표 2. SHARE가 Sasao의 방법보다 우수한 결과를 갖는 회로

Table 2. Circuits for which SHARE has the better results than Sasao's method.

Name	ni	no	PLAY GROUND	espresso -do opo	SHARE	exhaustive solution
bc0	26	11	179	185	179	179
[illegible]	26	46	180	*	180	-
[illegible]	49	25	59	*	59	-
[illegible]	29	32	137	*	137	-
[illegible]	26	38	117	120	117	-
[illegible]	8	5	35	37	35	35
alu4	14	8	104	102	103	103
[illegible]17	10	11	15	17	16	14
[illegible]y	12	7	107	117	107	107
[illegible]	12	20	35	39	34	-
[illegible]	32	20	212	*	212	-
[illegible]	24	14	62	149	62	-
[illegible]	24	8	54	*	54	-
[illegible]	36	57	116	*	115	-
[illegible]p4	8	8	112	123	112	111
[illegible]	56	23	34	*	34	-
rac	8	12	26	26	25	-
sao2	10	12	38	39	20	39
x2	25	8	110	140	110	110
xfm	4	7	8	9	8	8

* : 5시간의 CPU 시간으로 결과를 얻을 수 없었음.
- : 실험을 행하지 않았음.

표 3. SHARE가 PLAYGROUND보다 우수한 결과를 갖는 회로

Table 3. Circuits for which SHARE has the better results than PLAYGROUND.

Name	ni	no	espresso -do opo	PLAY GROUND	SHARE	exhaustive solution
Z5xp1	7	10	59	60	59	58
alu2	10	10	40	43	40	40
apla	10	12	22	25	24	22
in1	16	17	104	106	104	-
dk27	9	9	8	9	8	8
x6dn	39	5	81	82	81	81

* : 5시간의 CPU 시간으로 결과를 얻을 수 없었음.
- : 실험을 행하지 않았음.

표 4. 세가지 방법이 동일한 결과를 갖는 회로
Table 4. Circuits for which three methods have the same results.

Name	ni	no	espresso -do opo	PLAY GROUND	SHARE	exhaustive solution
Z9sym	9	1	72	72	72	72
add6	12	7	293	293	293	293
adr4	8	5	61	61	61	61
alu1	12	8	15	15	15	15
co14	14	1	14	14	14	14
dc1	4	7	9	9	9	9
f51m	8	8	76	76	76	76
in0	15	11	107	107	107	107
radd	8	5	61	61	61	61
rck1	32	7	32	32	32	32
rd53	5	3	22	22	22	22
rd73	7	3	93	93	93	93
sqn	7	3	33	33	33	33
ti	47	72	*	*	*	-
tial	14	8	359	359	359	359
x2dn	82	56	*	*	*	-
x7dn	66	15	*	*	*	-
z4	7	4	45	45	45	45

* : 5시간의 CPU 시간으로 결과를 얻을 수 없었음.
- : 실험을 행하지 않았음.

GROUND, 및 SHARE가 18개의 PLA에 대하여 동일한 면적의 PLA를 생성함을 보여준다. 또한, 5 시간의 CPU 시간내에 Sasao의 방법은 15개의 PLA, PLAYGROUND는 5 개의 PLA, SHARE는 3개의 PLA에 대하여 최소화된 결과를 얻을 수 없음을 보여준다.

SHARE 프로그램의 유용성을 입증하기 위하여 모든 가능한 출력 phase에 대하여 ESPRESSO를 실행하였다. 이와같은 과정으로 얻어진 결과는 ESPRESSO가 휴리스틱 알고리듬에 바탕을 두고 있기 때문에 최적의 PLA는 아니지만 출력 phase 할당 프로그램의 유용성을 평가하는데 사용될 수 있다. 표 Ⅰ-Ⅳ의 7열은 36개의 PLA에 대해 실험하여 얻은 PLA의 적항의 수인데, SHARE 프로그램은 36개의 PLA 중에서 29개의 PLA에 대하여 동일한 결과를 생성함을 알 수 있다. 따라서, SHARE 프로그램은 대부분의 PLA에서 최적 또는 최적에 가까운 해를 얻을 수 있을 것으로 기대된다.

표 Ⅴ는 Sasao의 방법, PLAYGROUND, 및 SHARE의 CPU 시간을 비교한 것이다. 56개 PLA의 대부분에 대하여 SHARE가 훨씬 적은 CPU 시간을 요구함을 알 수 있다. bca, bcb, bcc, 및 bcd와 같은 PLA에 대하여 걸리는 계산시간은 다른 PLA에 비하여 굉장히 많지만 CPU 시간의 대부분은 논리함수의 모든 출력을 부논리로 구현하는데 소모된 것이다. SHARE에서 CPU시간이 감소된 원인은 이중 phase 특성함수 [6] 또는 병합 PLA [7] 를 최소화 하는 과정이 SHARE 알고리듬에는 없기 때문이다. 따라서, 입출력의 수와 적항의 수가 적은 PLA에 대해서는 큰 차이가 없지만 입출력의 수와 적항의 수가 많아지면 SHARE 알고리듬은 다른 방법에 비하여 훨씬 적은 계산 시간을 필요로 한다.

표 5. Sasao의 방법, PLAYGROUND, SHARE의 성능 비교
Table 5. Performance comparison of Sasao's method, PLAYGROUND, and SHARE.

Name	ni	no	espresso + -do opo	PLAY + + GROUND	SHARE +
Z5xp1	7	10	7.4	70.2	13.2
Z9sym	9	1	5.8	57.7	2.5
add6	12	7	77.5	864.6	47.9
adr4	8	5	6.5	38.1	4.5
alu1	12	8	0.9	10.4	0.5
alu2	10	10	8.8	30.5	7.5
alu3	10	8	9.5	21.3	7.4
ex5p	8	63	5.1	70.7	18.2
pcle	19	9	99.4	448.1	431.2
pcle	19	9	*	-	7433.9
pcle	19	9	*	-	4281.6
pcle	19	9	*	-	3786.6
pcle	19	9	*	-	2459.0
bcc	26	38	2137.8	2001.3	98.8
chkn	29	7	483.9	-	9.8
co14	14	14	2.3	-	1246.0
cps	24	109	*	-	-
dc1	4	7	0.1	-	13.3
dc2	8	7	2.8	-	39.6
dist	8	5	14.7	148.5	7.7
dk17	10	11	3.7	16.1	0.8
dk27	9	9	0.2	-	20.4
dk48	15	17	*	78.4	941.7
ex5p	30	63	*	4144.6	7.0
f51m	8	8	11.2	-	87.2
gary	15	11	26.2	-	81.2
in0	15	11	41.2	-	202.6
in1	16	17	5.7	-	47.1
in2	19	10	283.4	239.1	64.3
in3	35	29	96.5	487.7	584.3
in4	32	20	*	2151.1	58.0
in5	24	14	268.7	-	104.6
in6	33	23	*	-	8.8
in7	26	10	12.9	-	287.0
jbp	36	57	*	9537.4	53.8
misg	56	23	*	287.4	1508.8
mlp4	8	8	34.8	283.4	130.6
opa	17	69	166.0	595.0	109.6
radd	8	5	3.9	-	2.4
rck1	32	7	15.0	-	121.3
rd53	5	3	0.8	-	1.7
rd73	7	3	2.8	-	10.9
risc	8	31	7.8	23.8	17.4
root	8	5	6.1	-	1.0
sqn	7	3	1.1	-	9.4
adr6	6	12	5.0	72.0	*
ti	47	72	*	-	571.8
tial	14	8	536.0	-	164.6
vg2	25	8	1586.1	-	1.8
wim	4	7	0.1	-	151.1
x1dn	27	6	*	-	*
x2dn	84	56	*	-	25.3
x6dn	39	5	884.5	-	214.6
x7dn	66	15	*	1759.2	1.9
z4	7	4	2.0	-	

+ : Sun4/330 CPU time (seconds)
++: VAX 8600 CPU time (seconds)
* : 5시간의 CPU 시간으로 결과를 얻을 수 없었음.
- : 실험을 행하지 않았음.

실험 결과를 요약하면 다음과 같다. Sasao의 방법과 비교하여 SHARE 프로그램은 32개의 PLA에 대하여 우수한 결과, 1개의 PLA에 대하여 열등한 결과, 23개의 PLA에 대하여 동일한 결과를 얻는다. 또한, PLAYGROUND 프로그램과 비교하여 SHARE 프로그램은 18개의 PLA에 대하여 우수한 결과, 5개의 PLA에 대하여 열등한 결과, 33개의 PLA에 대하여 동일한 결과를 얻는다.

Ⅴ. 결론

본 논문에서는 효율적인 출력 phase 할당을 이용한 PLA 최소화 알고리듬을 제안하였다. 이 알고리듬은 출력들 사이에 공유되는 적항의 수가 최대화 되고, 특정 출력에만 사용되는 적항의 수가 최소화 되도록 출력의 phase를 결정한 후 결정된 출력 phase

에 따라 PLA를 최소화한다. 이 알고리듬은 Sun4/330상에 C 언어로 구현되었으며, 56개의 예제 회로에 대하여 기존의 알고리듬과 비교되었다. Sasao의 방법과 비교하여 SHARE 프로그램은 32개의 PLA에 대하여 우수한 결과, 1개의 PLA에 대하여 열등한 결과, 23개의 PLA에 대하여 동일한 결과를 얻었다. 또한, PLAYGROUND 프로그램과 비교하여 SHARE 프로그램은 18개의 PLA에 대하여 우수한 결과, 5개의 PLA에 대하여 열등한 결과, 33개의 PLA에 대하여 동일한 결과를 얻었다. 더구나, SHARE 프로그램은 Sasao의 방법과 PLAYGROUND에 비해 계산 시간이 훨씬 짧았다. 이러한 계산 시간의 단축은 m개의 출력을 가진 논리함수에 최적의 출력 phase를 할당하기 위하여 Sasao의 방법과 PLAYGROUND가 2m개의 출력을 갖는 이중 phase 특성 함수 또는 병합 PLA를 최소화하는데 비해 SHARE는 이러한 최소화 과정을 필요로 하지 않는데 기인한다. 따라서, 본 논문에서 제안된 방법은 많은 입출력과 적항을 갖는 논리함수의 PLA 구현에 이용될 수 있을 것으로 기대된다. SHARE 프로그램에서 CPU 시간의 대부분은, 논리함수에 적절한 출력 phase를 할당하는 과정에서 모든 출력을 부논리로 구현하는데 소비 되었다. 따라서, 앞으로는 모든 출력을 부논리로 구현하지 않고도 적절한 출력 phase를 할당할 수 있는 알고리듬이 개발되어야 한다.

감사의 글

본 논문의 알고리듬을 개발하는데 많은 도움을 주신 대전 공업 대학교 이재홍 교수님과 실험을 수행하는데 성의를 다해 도와준 김태수씨께 깊은 감사를 드립니다.

參考文獻

[1] R. K. Brayton, G. D. Hactel, C. T. McMullen, and A. L. Sangiovanni-Vincentelli, *Logic Minimization Algorithms for VLSI Synthesis.* Hingham, MA: Kluwer Academic, 1985.

[2] S. J. Hong, R. G. Cain, and D. L. Ostapki, "MINI: A heuristic approach for logic minimization," *IBM J. Res. Develop.*, pp. 443-458, Sept. 1974.

[3] D. W. Brown, "A state-machnine synthesizer-SMS,", in *Proc. 18th Design Automation Conf.*, pp. 301-304, Nashiville, June 1981.

[4] C. A. Mead and L. A. Conway, *Introduction to VLSI Systems.* Reading, MA: Addison-Wesley, 1980.

[5] N. Cavlan and S. J. Durhan, "Field-programmable arrays: Powerful alternatives to random logic," *Electron.*, pp. 89-94, July 5, 1979.

[6] T. Sasao, "Input variable assignment and output phase optimization of PLA 's," *IEEE Trans. Comput.*, vol. C-33, pp. 879-894, Oct. 1984.

[7] C. L. Wey and T. Y. Chang, "An Efficient Output Phase Assignment for PLA Minimization," *IEEE Trans. Computer-Aided Design.* vol. 9, pp. 1-7, Jan. 1990.

著 者 紹 介

李 永 浩(正會員)

1965年 8月 7日生. 1989年 2月 한양대 전자공학과 졸업. 1991年 2月 한양대 전자공학과 석사학위 취득. 1993年 2月 제 2 회 젊은 공학도를 위한 반도체 workshop 에서 장려상 수상. 현재 한양대 전자공학과 박사과정 재학. 주관심분야는 VLSI 설계 및 CAD 특히. Logic Synthesis 및 Testing 등임.

鄭 正 和(正會員)

1950年 3月 10日生. 1975年 2月 한양대 전자공학과 졸업. 1977年 2月 한양대 전자공학과 석사학위 취득. 1981年 3月 와세다대학 박사학위 취득. 현재 한양대 전자공학과 교수. CAD 및 VLSI 설계 연구회 위원장. 주관심분야는 VLSI 설계 및 CAD 특히. High-level Synthesis. Logic Synthesis. Layout Synthesis 등임.