論文 88-25-1-15

Microinstruction의 부호 할당에 관한 연구

(A Study on Code Assignment to Microinstructions)

吳 昌 埈*, 李 哲 東*, 柳 瑛 昱*

(Chang Jun Oh, Chul Dong Lee and Young Uk Yu)

要 約

VLSI 시스템의 제어부를 PLA로 실현할 때 microinstruction의 부호(code)를 효과적으로 할당함으로써 칩 면적을 최소화하는 프로그램을 개발하였다.

기존 알고리즘들이 one pass로 구성되어 있는 반면에, 본 논문에서 제안하는 알고리즘은 two pass로 구성하였으며, 본 알고리즘은 부호 할당 순서를 결정하기 위한 procedure, 부호 할당을 위한 procedure 및 부호길이(code length) 축소를 위한 procedure등 크게 세부분으로 나눌 수 있다. 특히, 부호 길이의 축소를 위하여 folding back이라는 새로운 방법을 제안하였다.

11개의 microinstruction, 6개의 microoperation의 예에 대해 4bit의 부호길이, 9개의 product term의 결과를 얻었으며, 프로세서의 제어부 설계에 유용하게 이용할 수 있음을 확인했다.

Abstract

We have developed a program that minimizes the chip area with effective code assignment of microinstructions when the control part of VLSI system is implemented in the PLA.

While conventional algorithms are composed of one pass, the algorithm proposed in this paper is composed of two passes and it consists of three principal procedures: a procedure for the determination of the code assignment order, a procedure for the code assignment and a procedure for the reduction of the code length. Especially, we have proposed a new method of folding back for the reduction of the code length.

We have obtained the result of 4 bit code length and 9 product terms when 11 microinstructions and 6 microoperations were given as inputs. Therefore, we have certified that it could be used efficiently for the control part of the processors.

I. 서 론

VLSI 시스템을 실현할 때 데이타 처리부와 제어부로 나누어 설계하는 경우가 많으며, 이때 제어부의 설계는 조정이 용이하도록 프로그램 가능한 성질을 지니면서 최적화(칩상에서의 면적의 최소화) 되어야 하기때문에 데이타 처리부 설계에 비해 어렵다. 제어부의 자동설계를 위해서는 구조의 규칙성이라는 관점에서 PLA가 유리하며 이의 최적화(논리 최소화나 folding)가 연구되어 왔으나,[1,2] 대부분이 이미 부호가 할당된 PLA에 대해 최적화를 행하는 것이기 때문에 그 부호에 따른 결과에는 자연히 한계가 생긴다.

이것을 해결하기 위해 주어진 조합(혹은 순서) 회로의 사양을 만족하는 동시에 그 회로를 실현하는 PLA의 면적이 최소가 되도록 입력 혹은 출력에 부호를 할당하는 문제를 생각할 수 있다. 이 문제에 대해 De Micheli 등이 주어진 순서회로의 사양으로 부터 심볼

*正會員, 電子通信研究所
(Electronics & Telecommunications Research Institute)

接受日字 : 1987年 10月 16日

릭으로 상태수를 최소화하고, 각 심볼릭에 대해 부호를 할당하는 발견적(heuristic) 방법을 제안하고 있다.[3,4] 그러나 그것은 cube의 분할이 최종 결과에 미치는 영향을 고려하지 않고 cube의 분할을 행하기 때문에 그 부호 할당의 결과가 반드시 최적이라고는 단언할 수 없다.[5]

본 연구에서는, De Micheli의 알고리즘을 개선하고, 조합회로의 입력에 대해 부호를 할당하는 새로운 방법을 제안한다.

이 방법은 프로세서 제어부의 각 microinstruction에 대해, 필요한 microoperation을 출력하는 회로의 설계를 목적으로 개발되었다. AND array를 이용해서 조합회로를 실현할 때, 각 microoperation에 대해 생성되는 cover에 포함되는 cube 갯수의 합이 최소가 되도록 하여, 각 microinstruction에 부호를 할당함으로써 AND array의 면적 최소화를 가능하게 한다.

Ⅱ. 본 론

1. 알고리즘의 개략적인 설명

조합회로의 사양은 각 microinstruction이 출력해야 할 microoperation의 대응표로써 주어진다.

〔예 1〕 그림 1에 사양의 예를 보였다. A-K의 11개의 microinstruction과 a-f의 6개의 microoperation이 있다. MicroinstructionA는 microoperation a b를, B는 a, b, d를 출력한다. De Micheli의 알고리즘에 따라 각 microinstruction에 부호를 할당한 결과를 그림 2(a)에, 그리고 그 결과에 따라 AND array상에 조합회로를 실현한 예를 그림 2(b)에 보인다. De Micheli의 알고리즘은 주어진 mnemonic들에 대해 하나씩 차례로 조합회로의 사양을 만족하는 부호를 할당해 나가며, 사양을 만족하는 부호가 존재하지 않을 경우는 각 논리함수의 cube를 분할하여 부호를 할당하는 방법으로서, one pass로 알고리즘이 종료된다.

PLA의 면적을 결정하는 중요한 요소는 AND array의 면적이다. 본 연구에 있어서는 folding이나 partition과 같은 물리적으로 면적을 줄이는 방법을 사용하지 않기 때문에 이 AND array의 면적은 단순히 입력 line수와 product term line수의 곱에 의해 결정된다. 입력 line수는 각 microinstruction에 할당되는 부호길이의 2배이다. Microinstruction 부호길이가 길어지게 되면 array의 면적을 증대시키기 때문에 부호길이는 최소로 해야한다. 그러므로 주어진 microinstruction의 갯수를 n이라 하면, 부호길이는$\log_2 n$보다 다 작지않은 최소의 정수 $\lceil \log_2 n \rceil$으로 제한되며 변경할 수 없다.

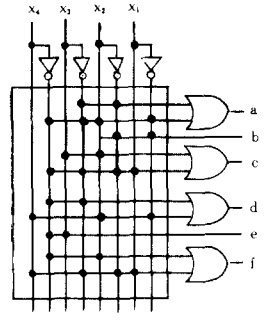
```
A : a, b;
B : a, b,  d;
C : a,     d,  f;
D : a,         f;
E :  b,    e;
F :  b;
G :        d,
H :     c, d;
I :     c,  ; e, f;
J :     c;
K : a,  c, d;
```

그림 1. 조합회로에 대한 사양의 예
(A, B, … : Microinstruction name)
(a, b, … : Microoperation name)

Fig. 1. An example of specification for a combinational logic.

MI	x_4	x_3	x_2	x_1
A	1	0	0	0
B	0	0	0	0
C	0	0	1	0
D	1	0	0	1
E	0	1	0	0
F	1	1	0	0
G	1	0	1	0
H	1	1	1	0
I	0	1	1	1
J	1	1	1	1
K	0	0	0	1

(a) 각 microinstruction에 할당된 부호



(b) AND array에 의한 실현

그림 2. 그림 1에 대해 De Micheli의 알고리즘에 따른 부호 할당의 결과

Fig. 2. Code assignment results based on De Micheli's algorithm for Fig. 1.

전체 product term line수와 각 microoperation에 대한 cube수의 관계 및 출력 line수와 microoperation의 갯수에 대한 관계를 수식으로 표현하면 다음

과 같다.

$$P_{total}=\sum_{i=1}^{m} C_i, \quad (1)$$

$$L_{out}=m \quad (2)$$

여기서,

P_{total} : total product term line수

C_i : microoperation i에 대한 cube수

m : microoperation의 갯수

L_{out} : 출력 line수 이다.

예를들면, 그림 2(a)의 부호에 대해, microoperation a에 대해 생성되는 cover는 두개의 cube $\bar{x}_4\bar{x}_3x_2\bar{x}_1$와 $\bar{x}_3\bar{x}_2$를 포함하기 때문에 두개의 product term line으로 되어 있다.

이상의 사실로 부터, AND array의 면적을 최소로 하기 위해서는 각 microoperation에 대한 cover에 포함되는 cube 갯수의 합이 최소로 되도록 각 microinstruction에 부호를 할당하면 된다는 것을 알 수 있다. 여기서, cube는 어떤 논리함수에 포함되는 minterm을 의미하며, cover는 그 cube들로 구성된 집합을 의미한다.

De Micheli에 의해 제안된 부호 할당 알고리즘〔3〕의 개략을 그림 3에 보인다. 여기서 constraint relation이라는 것은 하나의 mnemonic(본 논문에서는 microinstruction에 해당)에 대해 할당 가능한 부호인가의 여부를 판정하는 조건을 의미한다. 즉, 어떤 mnemonic에 할당 가능한 부호란 것은, 그 mnemonic을 포함하는 집합에 대해 생성되는 cover에는 포함될 수 있지만 mnemonic을 포함하지 않는 집합에 대해 생성되는 cover에는 포함되지 않는 것이어야 한다.

그림 3의 〔Step 2〕에서는 주어진 mnemonic에 대해 할당 가능한 부호를 구하는데, 가능한 한 각 출력에 대해 생성되는 cover가 작게 되도록 하는 부호를 택하도록 조작을 행한다. 그와같은 부호가 존재하지 않을 경우에는 〔Step3〕에서 부호길이가 제한치 이내이면 부호길이를 증가시키고, 제한치에 도달한 경우에는 각 microoperation에 대해 생성되는 cover에 포함되는 cube 중에서 적당한 것을 택하여 그것을 분할한 후, 다시 할당 가능한 부호를 찾기위해 〔Step2〕로 되돌아간다. 〔Step3〕에서 어느 cube를 어떻게 분할할 것인가의 문제는, 그 분할에 의해 최종 결과가 어떻게 변할 것인지를 예측할 수 없기 때문에 그 시점에서 가장 적절한 cube를 선택한다는 것은 매우 어려운 문제이다.

그러므로 본 연구에서는 부호길이의 제한을 없애고, 할당 가능한 부호가 존재하지 않을 경우에는 일단 부호길이를 확장하여 우선 전체의 microinstruction에

〔Step 1〕 Select an uncoded mnemonic.
〔Step 2〕 Determine a code for the mnemonic satisfying the constraint relation.
〔Step 3〕 IF (no code exists)
IF (code length < given bound)
Increase code length.
Go to Step2.
IF (code length = given bound)
Relax a constraint.
Go to Step2.
〔Step 4〕 Assign the code to the selected mnemonic.
〔Step 5〕 IF (all mngmonic have been encoded)
Stop.
ELSE
Go to Step1.

그림 3. De Micheli가 제시한 부호 할당 알고리즘
Fig. 3. Code assignment algorithm proposed by De Micheli.

부호를 할당할 수 있도록 한다. 이것은 각 microoperation에 대해 생성되는 cover가 단 하나의 cube로 이루어 지도록 하는 부호 할당을 의미한다. 그러므로 당연히 부호길이는 제한치 $\lceil \log_2 n \rceil$ 보다 커지게 된다.

그 다음에 부호길이가 제한치에 이를때까지 microoperation의 cube를 분할하면서 부호길이를 축소해나간다. 이 조작을 위해 본 연구에서는 folding back 이라는 새로운 방법을 도입하였다. 이와같이 알고리즘을 two pass로 함에 따라, 보다 최적의 해에 접근하기 위한 cube의 효과적인 분할이 행해질 수 있다.

이하에 본 방법의 개략을 보인다.

〔단계 1〕 주어진 사양으로부터 부호를 할당할 microinstruction의 순서를 결정한다.

〔단계 2〕 그 순서에 따라, 그림 3의 알고리즘과 같은 방법으로 각 microinstruction에 부호를 할당해 나간다. 단, 이때 부호길이의 제한치는 없는 것으로 한다.

〔단계 3〕 Microoperation에 대해 생성되는 cover 에 포함되는 cube를 분할해 가면서 부호길이를 제한치 ($=\lceil \log_2 n \rceil$, n : microinstruction 수)까지 축소시킨다. 이때, $2^{\lceil \log_2 n \rceil} \geq n$ 이므로 모든 경우에 대해 부호길이는 제한치까지 축소가 가능하다.

2. 부호 할당 순서의 결정

각 microinstruction에 부호를 할당할 순서를 결정하

기 위해서, 주어진 그림 1과 같은 사양으로 부터 2부 그래프(bipartite graph)를 만든다. 이 그래프의 각 정점은 microinstruction 및 microoperation들의 이름이며, 한 microoperation이 어떤 mic roinstruction 에 의해 출력 될때에 한해, 그들 정점 사이에 가지가 형성되게 된다.

이 2부 그래프를 이용해서 부호 할당의 순서(초기 순서)를 결정하는데, 이 순서에 따라 부호할 당의 결과가 크게 좌우된다. 그러나 이 순서가 절대적인 것은 아니며, 3절에 기술하는 바와같이 어떤 같이 어떤 시점에서 할당 가능한 부호가 존재하지 않을 경우는 순서를 바꾼다.

순서를 결정할 때 중요한 것은 2부 그래프 상의 각 정점의 degree이다. 이때 각 정점의 degree란, 2부 그래프 상의 각 정점에 연결되어 있는 가지(branch)의 수를 의미한다. 많은 microoperation 을 출력하는 microinstruction, 즉, 2부 그래프 상에서 degree가 큰 microinstruction은 앞서 기술한 constraint relation에 의해, 그만큼 많은 microoperation들의 cover에 공통으로 포함되는 부호이어야 한다. 본 알고리즘의 첫번째 pass에서는 microoperation 에 대해 생성되는 cube의 분할을 행하지 않기 때문에, 순서가 늦은 쪽에서 degree가 큰 microinstruction에 부호를 할당하려고 하면 부호가 구해지지 않거나 무의미한 cube의 확장이 일어날 가능성이 있다. 그러므로 부호 할당은 가능한 한 degree가 큰 microinstruction부터 행할 필요가 있다.

본 연구에서 사용한 개략적인 순서결정 방법은 다음과 같다

〔단계 1〕 가장 먼저 부호가 할당될 microinstruction 은 2부 그래프에서 degree가 최대인 정점으로 한다. 같은 degree를 가진 microinstruction 정점이 2개 이상 존재할 경우에는 그 정점들에 인접하는 각 microoperation 정점들의 degree의 합이 최대인 것을 선택한다.

〔단계 2〕 그 다음에 부호를 할당할 microinstruction은 아직 순서가 결정되지 않은 microinstruction중에서, 이미 순서가 결정된 microinstruction에 인접하는 microoperation에 접속하는 가지의 갯수가 최대인 것으로 하며, 이와 같은 가지의 갯수가 같을 경우에는 degree가 큰 microinstruction을 선택한다.

〔예 2〕 그림 1의 사양으로부터 만들어진 2부 그래프를 그림 4에 나타내었고, 이를 이용하여 결정한 부호 할당 순서를 그림 5에 나타내었다.

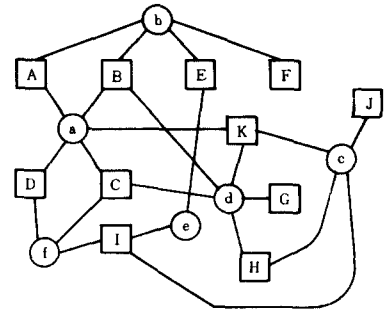


그림 4. 그림 1에 대한 2부 그래프
Fig. 4. Bipartite graph for Fig. 1.

순 서	1	2	3	4	5	6	7	8	9	10	11
MI명	B	C	K	I	A	D	E	H	F	G	J

그림 5. 그림 1에 대한 부호 할당의 초기 순서
Fig. 5. Initial order for code assignment for Fig. 1.

본 연구의 부호 할당 알고리즘에 있어서, 부호나 logic product을 표현하기 위해 cube notation을 이용한다. 즉, 부호길이를 n, 각 literal을 x_i ($1 \leq i \leq n$) 라 하면, microinstruction에 할당된 부호W 및 microoperation에 대해 생성되는 cover의 cube C를 다음과 같이 나타낸다.

$W = [w_n, w_{n-1}, \cdots, w_1]$

$C = [c_n, c_{n-1}, \cdots, c_1]$

여기서, 각 W와 C의 각 요소는 literal $\bar{x}_i$가 존재하면 0, literal $\bar{x}_i$가 존재하면 1, literal $\bar{x}_i$와 x_i둘다 존재하지 않으면 2로 나타낸다.

3. 부호 할당 알고리즘

할당순서 결정 알고리즘에 의해 구한 부호 할당순서에 따라 각 microinstruction에 부호를 할당하는 알고리즘을 이하에 기술한다.

〔단계 0〕 부호길이를 1로하여 처음의 두 microinstruction의 각각에 부호$W_1 = [0]$와, 부호 $W_2 = [1]$을 할당한다.

〔단계 1〕 그 다음에 부호를 할당할 microinstruction을 newMI라 하고, 그것이 출력하는 microoperation의 집합을 S라 한다.

〔단계 2〕 S에 속하는 각 microoperation에 대해, new MI에 할당 가능한 부호를 전부 구한다.

〔단계 3〕 〔단계 2〕에서 구한 부호 중 S에 속하는 모든 microoperation에 공통인 부호를 구한다.

〔단계4〕 공통의 부호가 존재할 경우는 그 중 하나를 선택하여 newMI에 할당한다. 이때, 각 microoperation에 대해 생성되는 cover에 포함되는 cube를 불필요하게 크게하지 않고, 또, 2부 graph에 있어서 S에 속하는 microoperation 정점에 인접하고, 이미 부호가 할당된 microinstruction과 newMI와의 hamming distance의 총합이 최소가 되도록 부호를 선택한다.

〔단계5〕 공통의 부호가 존재하지 않을 경우는 부호길이를 확장(이미 할당된 부호에 대해서는 0을 MSB로 추가)하여 〔단계2〕로 되돌아간다. 그러나 이미 newMI에 대해 부호길이를 확장해버린 경우에는 더이상 확장하더라도 의미가 없기 때문에 다음에 기술할 알고리즘에 의해 부호 할당 순서를 바꾼다.

〔단계6〕 전체의 microinstruction에 부호가 할당되어지면 모든 조작이 종료되며 그렇지 않은 경우는 〔단계1〕로 되돌아 간다.

부호할당 알고리즘〔단계5〕에 있어서 부호 할당 순서를 바꾸는 방법에 대해 다음에 기술한다.

〔단계1〕 newMI 바로 전에 부호가 할당된 microinstruction을 preMI라 하고, 이것과 newMI의 degree(2부 graph상에서)를 비교한다.

〔단계2〕 이들이 같거나 preMI의 degree가 작을 경우는 preMI에 할당된 부호를 무효로 하고 newMI와 preMI의 할당순서를 바꾸어 종료한다. 그러나 preMI의 초기 순서가 newMI보다 늦은 경우에는 그 두 microinstruction의 순서를 바꾸어 다시 〔단계1〕로 되돌아간다.

〔단계3〕 preMI의 degree가 큰 경우에는 newMI 이후의 부호 할당 순서를 초기 순서로 환원시키고, preMI에 할당된 순서를 무효로 한 후 preMI를 newMI로 하여 〔단계1〕로 되돌아 간다.

〔단계4〕 할당 순서를 바꾼 결과가 원래의 초기 순서로 환원된 경우는, 앞서 기술한 방법에 의해 결정된 할당 순서로는 주어진 사양에 대한 부호 할당이 불가능 함을 의미한다. 그 원인의 하나로서, 최초에 부호를 할당한 두 microinstruction에 할당된 부호의 hamming distance가 너무 가까왔기(최초에는 1)때문에 그 이후의 부호 할당에 장애를 초래했다고 생각할 수 있다. 그러므로 최초의 두 microinstruction에 할당하는 부호의 hamming distance를 크게 하기 위해서, 부호 할당 알고리즘의 〔단계0〕에 있어서 부호길이를 1bit 증가 켜 최초의 두 microinstruction에는 각각 부호 $W_1=\{0,\cdots,0\}$ 및 부호 $W_2=\{1,\cdots,1\}$을 할당하여 다시 부호 할당 조작을 행한다.

〔예3〕 그림1의 사양에 대해, 그림5의 순서에 따라 본 알고리즘을 이용하여 부호장의 축소를 행하기 전까지 수행한 결과, 즉, pass one의 결과를 그림6에 나타내었다. 또한 그 결과를 Karnaugh map상에 나타낸 것을 그림7에 보였다.

4. 부호길이 축소 알고리즘

Pass one의 결과에 대해, 부호길이가 제한치에 이를때까지 단축하는 알고리즘은 다음과 같다.

〔단계1〕 Folding back을 행할 bit를 결정하고, 그 bit에 대해 folding back을 행한다.

〔단계2〕 Folding back의 결과, 부호의 재할당을 필요로 하는 모든 microinstruction에 대해 부호의 재할당을 행한다.

〔단계3〕 부호길이가 제한치와 같게되면 알고리즘이 종료되며, 그렇지 않으면 〔단계1〕로 되돌아 간다.

1) Folding Back

어떤 cube의 한 bit(x_i)의 값이 1 또는 0이며, 다른 bit의 값은 2(don't care)인 cube〔2, ⋯, 2, 1, 2, ⋯, 2〕 또는 〔2, ⋯, 2, 0, 2, ⋯, 2〕를 각각 literal x_i의 반평면(half plane), $\bar{x}_i$의 반평면이라 정의한다. 어떤 cube의 한 bit(x_i)에 대해, literal x_i의 반평면 상에

MI	w_5	w_4	w_3	w_2	w_1
A	0	1	0	0	0
B	0	0	0	0	0
C	0	0	0	0	1
D	0	1	0	0	1
E	0	0	1	0	0
F	0	1	1	0	0
G	1	0	0	0	0
H	1	0	0	1	0
I	0	0	1	1	1
J	1	0	1	1	0
K	0	0	0	1	0

(a) 각 microinstruction에 할당된 부호

MO	c_5	c_4	c_3	c_2	c_1
a	0	2	0	2	2
b	0	2	2	0	0
c	2	0	2	1	2
d	2	0	0	2	2
e	0	0	1	2	2
f	0	2	2	2	1

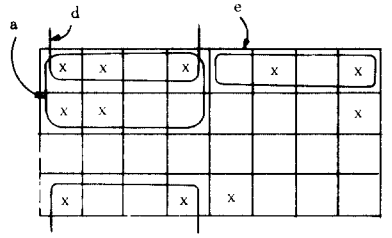
(b) 각 microoperation의 cover 상태

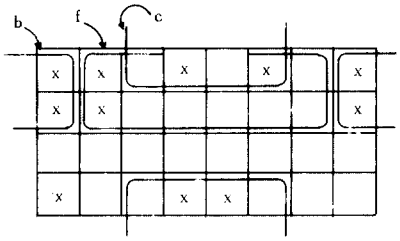
그림 6. 그림1의 사양에 대한 본 알고리즘의 pass one까지의 결과

Fig. 6. The results of pass one in this algorithm for the specification of Fig. 1.

	000	001	011	010	110	111	101	100
00	B	C		K		I		E
01	A	D						F
11								
10	G			H	J			

(a) 각 microinstruction에 대한 Karnaugh map





(b) 각 microoperation의 cover에 대한 Karnaugh map.

그림 7. 그림 6에 대한 karnaugh map
Fig. 7. Karnaugh map for Fig. 6.

있는 모든 부호 $[c_n, \cdots, c_{i+1}, 0, c_{i-1}, \cdots, c_1]$을 $[c_n, \cdots, c_{i+1}, 1, c_{i-1}, \cdots, c_1]$으로 변환 하는것을 literal $\bar{x}_i$의 반평면을 literal x_i의 반평면으로 folding back을 행하였다고 정의한다. 역으로, literal x_i의 반평면을 folding literal $\bar{x}_i$의 반평면에 folding back 행한다고 하는 것은 literal x_i의 반평면 상에 있는 전체의 부호에 대해 x_i의 값을 1에서 0로 변경하는 것을 말한다. 이때 부호가 중복되는 microinstruction이 반드시 존재하게 되는데, 그 microinstruction들에 대해서는 후술하는 바와 같이 부호 재할당을 행하게 된다. 이와 같이 folding back을 행함으로써 전체 부호의 bit x_i의 값은 0 또는 1로 통일되기 때문에 이 bit x_i를 무시할 수 있게 되며, 따라서 부호길이의 한 bit 축소가 가능하게 된다. 그 결과는 어느 bit에 대해 folding back을 행하는가에 따라 좌우되는데, 본 연구에서는 다음의 3가지 기준에 의해 folding back을 행할 bit를 결정한다.

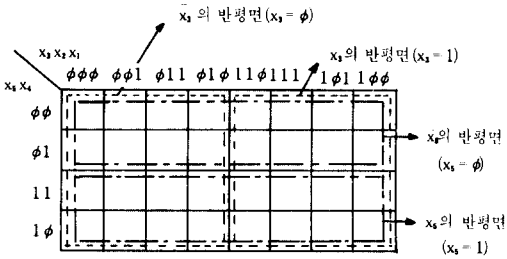


그림 8. 반평면의 예
Fig. 8. An example of half plane.

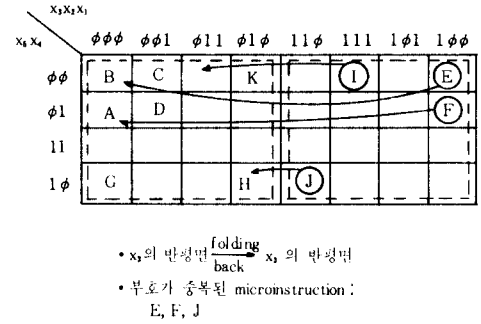


그림 9. Folding back의 예
Fig. 9. An example of folding back.

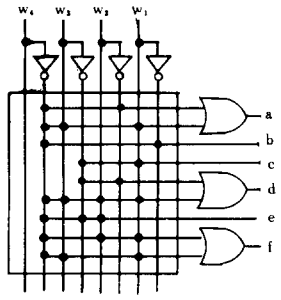
- Folding back 되는 반평면 상에 할당되어 있는 microinstruction의 수가 최소인 bit, 즉 그림 8에 반평면의 예를 나타내었는데, 이 반평면 상에 할당되어 있는 microinstruction의 수가 최소인 bit를 선택한다.
- 그림 9에 보는 바와 같이 folding back으로 인해 부호가 중복되는 microinstruction의 수가 최소인 bit.
- Folding back의 결과 각 microoperation의 cover에 포함되는 cube의 총합이 최소인 bit.

2) 부호의 재할당

부호길이 단축 알고리즘의 [단계 2]에서는 folding back에 의해 부호의 재할당이 필요하게된 microinstruction에 대해 부호 할당 조작을 행한다. 이 조작이 앞서 기술한 부호 할당 알고리즘과 다른점은 부호길이를 확장하지 않기 때문에 microoperation에 대해 생성되는 cover를 구성하는 cube가 분할된다는 점이다. 이때 microinstruction에 재할당 하고자 하는 부호는 Quine McCluskey법[6]을 이용하여 각 microoperation에 대해 생성되는 cover를 구성하는 cube의 총합이 최소가 되도록 하는 것이다.

MI	w_4	w_3	w_2	w_1
A	0	1	0	0
B	0	0	0	0
C	0	1	1	1
D	0	1	0	1
E	0	0	1	0
F	0	1	1	0
G	1	0	0	0
H	1	0	0	1
J	0	0	1	1
K	1	0	1	1
	0	0	0	1

(a) 각 microinstruction에 할당된 부호



(b) AND array에 의한 실현

그림10. 그림 6 의 부호에 대해 부호단축을 행한 결과
Fig. 10. The results of the reduction of code length for the code of Fig. 6.

Ⅲ. 결과 검토 및 결론

그림 6 에 보인 부호 할당의 결과(pass one의 결과)에 대해 부호길이의 단축을 행한 결과(pass two의 결과)를 그림10(a)에 나타내었다. 그리고 그 결과에 따라 AND array상에 실현한 것을 그림10(b)에 나타내었다.

그림 1 의 사양에 대해, De Micheli의 알고리즘을 이용하여 각 microinstruction에 부호 할당을 행한 결과 그림 2 (b)와 본 알고리즘에 따라 구한 결과(그림10(b))를 비교하였다. 전자의 경우는 각 microoperation 의 cover에 포함되는 cube의 총합이 10개, 후자의 경우는 9 개 이므로, 후자 즉, 본 알고리즘에 의해 부호를 할당 함으로써, De Micheli의 알고리즘에 따라 구한 결과에 비해 10% 개선할 수 있었다. microinstruction 의 수가 많아지게 되면 각 microoperation의 cover상태가 더욱 복잡해지기 때문에, 분할 해야할 cube 수가 자연히 증가하게 된다. 그런데 De Micheli의 알고리즘은 각 microinstruction의 부호할당 작업 도중에 cube의 분할을 행할때 그 분할이 최종 결과에 미치는 영향을 고려하지 않고 분할해 버리기 때문에 cube의 수가 많아 질수록, 즉 microinstruction의 수가 많아 질수록 비효과적으로 부호가 할당될 가능성이 높아지게 된다. 그러나, two pass로 구성되어 있는 본 알고리즘은 일단 pass one에서 부호길이의 제한을 두지 않고 부호할당을 행하게 되며, 그 결과에 대해 부호길이가 제한치에 이를때까지 전술한 바와같이 folding back을 행하여 부호길이를 축소해 나간다. Folding back을 행하면 cube의 분할이 자연히 일어나게 되는데, 이때 folding back은 그 시점에서 부호의 모든 bit 중에서 cube의 분할이 가장 적게 일어나는 bit에 대해 행해지기 때문에 각 Microoperation의 cover에 포함되는 cube의 수를 최소로 할 수 있다. 이와같이 cube 수를 최소로 하기 위한 cube 분할의 어려움을 지닌 종래 알고리즘의 단점을 본 알고리즘에서는 folding back이라는 새로운 부호길이 축소방법을 이용한 two pass의 알고리즘으로 개량하여 해결하였으며, microinstruction의 수가 많아질수록 De Micheli의 알고리즘에 비해 본 알고리즘의 개선의 폭은 더욱 커질 것으로 예상된다.

參 考 文 獻

[1] R. Brayton, G.D. Hachtel, C. McMullen and A.L. Sangiovanni-Vincentelli, LOGIC MINIMIZATION ALGORITHMS FOR VLSI SYNTHESIS, *Kluwer Academic Publishers*, 1984.

[2] G.D. Hachtel, A.R. Newton and A.L. Sangiovanni-Vincentelli, "An algirithm for optimal PLA folding", *IEEE Trans. CAD.*, vol. CAD-1, pp. 63-77, Apr. 1982.

[3] G. De Micheli, "Optimal encoding of control logic", *ICCD NY*, pp. 16-22, Sept. 1984.

[4] G. De Micheli, R.K. Brayton and A.L. Sangiovanni-Vincentelli, "Optimal state assignment for finite state machines", *IEEE Trans. CAD*, vol. CAD-4, pp. 269-285, Jul. 1985.

[5] 高山 浩一郎, 吳昌埈, 二宮喜一郎, "マトクロ命令の符號化に關する一考察," 信學技報, vol. 86. no. 326, pp. 63-69, Jan. 1987.

[6] 樹下行三, 浅田邦博, 唐津修, VLSIの 設計Ⅱ (論理と テスト), 岩波書店, pp. 18-32, 1985. *