論文94-31A-10-17

명령어 해독기 설계를 위한 출력 부호화 방법

(Output encoding methods for the design of instruction decoder)

金 漢 興 *, 黃 承 浩**, 慶 宗 旻**

(Han Heung Kim, Seung Ho Hwang and Chong Min Kyung)

要約

본 논문에서는 마이크로프로그램된 프로세서의 명령어 해독기를 최소면적으로 구현하는 방법을 다룬다. 이 문제는 제한된 출력부호화 문제로 정의될 수 있음을 보였고, 시뮬레이티드 어닐링 기법에 기초하여 PLA나 다단계회로로 구현하기 위한 휴리스틱 방법들을 제안하였다. 제안된 방법들을 사용하여 임의부호화 방법보다 PLA 구현 시에는 10~40%, 다단계회로 구현 시에는 9.8~34.4% 정도의 면적을 줄일 수 있었다.

Abstract

In this paper, we consider the area-minimal implementation of the instruction decoder for microprogrammed processors such as modern CISC-type microprocessor. We formulate it as a constrained output encoding problem and, based on simulated annealing algorithm, propose efficient heuristic solution methods both for PLA and multi-level implementations of the decoder. Experimental results on various examples show that our methods produce, on the average, 10~40% reduction of the number of product terms for the PLA implementations and 9.8~34.4% reduction of the number of literals for the multi-level implementations compared to the results of random encoding method.

I. 서 론

마이크로프로그램된 프로세서에서 명령어 해독기는 각각의 기계언어를 이에 대응하는 마이크로코드 루틴의 시작번지로 맵핑시키는 일을 한다.[1,2] 명령어 해독기 구현에서 고려되어야 할 변수들은 입력인 기계언어 집합과 출력인 마이크로코드 루틴들의 시작번지 집합이 된다. 이 두가지 변수들은 상위단계 설계시에 기호로 표시되며, 논리설계 단계에서 2진수로 부호화된다.[3] 이와 같이 부호화된 명령어 해독기의 설계는 논리최소화 과정을 거쳐 PLA나 다단계회로로 구현된다.

최소면적의 명령어 해독기를 구현하기 위한 논리설계는 입력부호화와 출력부호화로 나누어진다.[4] 입력

*正會員, 現代電子産業(株) 半導體 第 2研究所
(R&D 2, Semiconductor Div., HEI Co. LTD)
**正會員, 韓國科學技術員 電氣 및 電子工學科
(Dept. of Elec. Eng., KAIST)
接受日字 : 1994年 1月 17日

부호화에는 기존의 방법을 사용할 수 있으므로, 본 논문에서는 출력부호화만을 다룬다. 출력부호화에서는 다음과 같은 제약조건들이 고려되어야 하며, 기존의 출력부호화 방법들로는 이와 같은 경우를 다룰 수 없다.

출력기호의 2진수 코드는 마이크로코드 루틴들을 저장하고 있는 On-chip 메모리의 주소공간으로 제한된다. 일반적으로 각 기계언어는 한개 이상 불특정한 갯수의 마이크로코드들로 나뉘어 해석된다. 수행성능을 향상시키기 위하여 이 마이크로코드들은 순차적으로 배치되어 연속된 메모리 공간을 차지하게 된다; 메모리 면적을 줄이기 위하여 점프 및 서브루틴 콜과 같은 마이크로프로그래밍 기법을 사용하는 경우에도 이와 같은 상황은 변함없다. 따라서 출력 기호의 코드는 메모리 주소공간중의 일부분으로 다시 제약된다. 또한 이 일부분도 마이크로코드 루틴들의 배치에 따라서 결정되는 것이므로 기존의 방법을 적용하기 어렵다.

본 논문의 구성은 먼저 II장에서 최소면적의 명령어 해독기를 구현하는 문제를 정의하고 III장에서는 PLA로 구현하는 경우에 대한 해결방법을 제안하며, IV장에서는 다단계회로로 구현하는 경우에 대한 해결방법을 제안한다. V장에서는 제안된 방법들을 사용한 실험결과를 정리하며 VI장에서는 결론과 제안된 방법들을 확장,보완하기 위한 방안을 간략히 기술한다.

II. 제한된 출력인코딩 문제

모든 기계언어는 n개의 그룹으로 분류되며 기계언어 그룹 $M_i(1\le i\le n)$는 기호 O_i를 출력한다. 기호 O_i의 2진수 코드를 $enc(O_i)$로 나타낸다. 그리고 I장에서 언급한 내용을 다음의 가정들을 도입하여 간략화한다:

가정 1. 기계언어 그룹 M_i에 대응하는 마이크로코드 루틴은 S_i개의 순차적 마이크로코드들로 이루어지며 이를 저장하는 메모리의 연속된 S_i개의 주소공간을 차지한다.

가정 2. 마이크로코드 메모리의 길이는 각 마이크로코드 루틴의 길이를 모두 더한 것, 즉, $(\Sigma_i S_i)$이다.

가정 1은 마이크로프로그래밍에서 점프나 서브루틴 콜을 제외하여 문제를 간략화 하려는 것이다. 이와 같은 마이크로프로그래밍 기법들을 포함하는 방법은 VI장에서 언급된다. **가정 2**는 마이크로코드 메모리 면적을 최소화하려는 것이다. 명령어 해독기의 면적을 줄이기 위하여 마이크로코드 루틴들 사이에 공간을 두는 것을 고려할 수 있는 데, 이는 반대급부로 마이크로코드 메모리 면적을 증가시키므로 칩 전체 면적을 줄이는 관점에서 고려되어야 한다.

표 1. 명령어 해독기의 한 예.
Table 1. Truth table of an example decoder.

PLA input (Macroinstruction code)	PLA output symbol (Microcode start address)	Microcode size
M_1 : 000	O_1	$S_1=3$
110	O_1	
M_2 : 001	O_2	$S_2=5$
111	O_2	
M_3 : 010	O_3	$S_3=3$
M_4 : 011	O_4	$S_4=2$
M_5 : 100	O_5	$S_5=2$
M_6 : 101	O_6	$S_6=1$

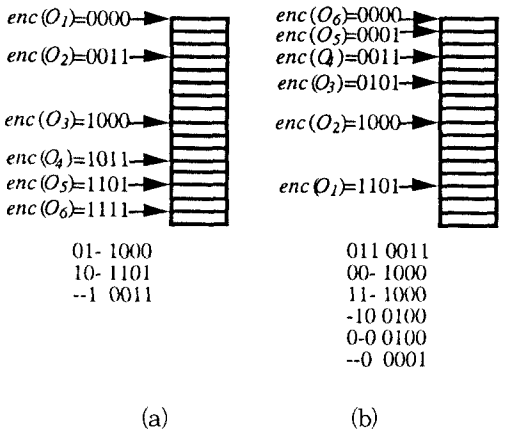


그림 1. 표 1의 예제에 대한 두가지 출력부호화 결과
Fig. 1. Two different output encoding results for the example shown in table 1, with the corresponding PLA personality matrix.

표 1에 위와 같이 정의된 간단한 명령어 해독기의 입력, 출력 및 마이크로코드 루틴의 길이가 나타나 있다. 그림 1에 표 1의 예에 대한 두가지 서로 다른 출력부호화 결과가 나타나 있는 바, 논리최소화 과정을 거친 후에 (a)의 경우는 3개의 프로덕트텀(Product term)을, (b)의 경우는 6개의 프로덕트텀을 갖는 2단계회로가 되는 것을 볼 수 있다. 이 예에서 살펴보면 출력부호화는 마이크로코드 루틴들의 선형배치에 대응되며, 최소면적의 명령어 해독기를 설계하기 위한 출력부호화 문제는 다음과 같이 정의됨을 알 수 있다.

- PLA로 구현하는 경우
  2단계 논리최소화 후에 프로덕트텀 수가 최소로 되도록 모든 출력기호 $O_i(1 \le i \le n)$의 2진수 코드, 즉, 모든 마이크로코드 루틴들의 선형배치 순서를 정한다.
- 다단계회로로 구현하는 경우
  다단계 논리최소화 후에 리터럴(Literal) 수가 최소로 되도록 모든 출력기호 $O_i(1 \le i \le n)$의 2진수 코드, 즉, 모든 마이크로코드 루틴들의 선형배치 순서를 정한다.

논리설계 단계에서 PLA 면적은 입력변수의 수, 출력변수의 수 및 프로덕트텀 수로 근사되는 함수[5]이나 이 경우에는 입력변수와 출력변수의 수가 정해져 있으므로 프로덕트텀 수에만 비례하게 되며, 다단계회로 면적은 리터럴 수로 근사되는데 하나의 리터럴은 궁극적으로 하나의 트랜지스터에 해당되기 때문이다.[6]

주어진 선형배치와 이의 비용 사이에는 복잡한 관계가 내재해 있으므로 최소비용을 갖는 선형배치를 찾기 위해서는 n!의 경우를 모두 살펴보아야 하는 어려움이 있다. 본 논문에서는 일부 경우만을 살펴보면서도 최적에 가까운 해를 찾아내는 시뮬레이티드 어닐링(Simulated annealing) 기법[7,8]을 이용하여 문제를 해결한다. 시뮬레이티드 어닐링 기법을 이용함에 있어 중요한 것은 주어진 선형배치의 비용을 빠르고 정확하게 계산하는 일이다. 이를 위하여 기존의 2단계 또는 다단계 논리최소화 프로그램을 사용할 수 있지만 이들의 빠른 형태들도(예를 들면 ESPRESSO-II with **fast** option[9] 또는 misII with **short** script[6]) 정확도 및(또는) 수행시간 면에서 부적합한 것으로 나타났다.

Ⅲ. 최소면적의 PLA로 구현하는 방법

본 논문에서 사용한 시뮬레이티드 어닐링 기법의 전체 흐름도는 그림 2와 같다. 일반적인 시뮬레이티드 어닐링 기법에 전처리 및 후처리 과정을 더한 것이다. 전처리 과정에서는 주어진 선형배치의 비용을 빠르고 정확하게 계산하기 위한 준비를 한다. 어닐링 루프 내에서는, 주어진 선형배치에서 서로 다른 두 마이크로코드 루틴들의 순서를 맞바꿀 때 새로이 정해지는 출력부호화 비용을 계산해 보고 새로운 배치의 수용여부를 판단하는 일을 반복하여 수행한다. 후처리 과정에서는 기존의 논리최소화 프로그램을 이용하여 최종 어닐링 결과에 대한 비용을 계산한다.

III.1절에서는 전처리 과정을 설명하며 III.2절에서는 비용 계산방법을 설명한다. 다단계회로에 대한 것은 IV장에서 설명한다.

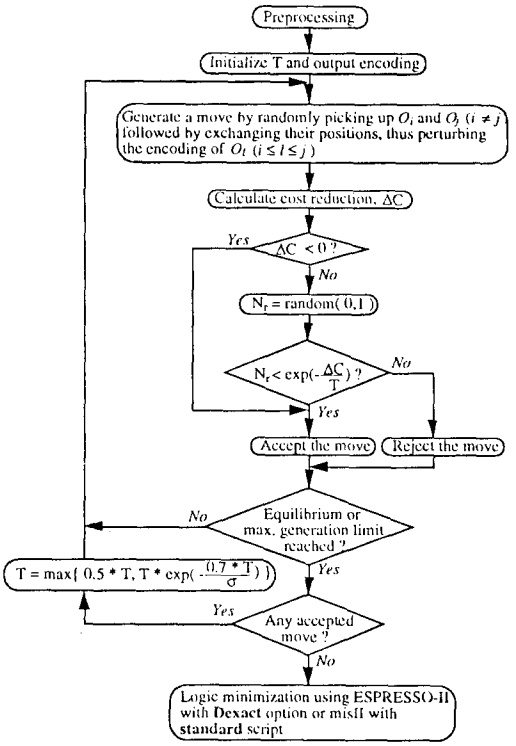


그림 2. 시뮬레이티드 어닐링 기법의 전체 흐름도; T는 온도, △C는 비용의 변화량, 그리고 σ는 비용의 표준편차임

Fig.2. An overall simulated annealing procedure, where T is temperature, σ is standard deviation of the cost distribution of accepted moves and △C is the cost change between new encoding and last accepted encoding.

1. PLA구현 비용계산을 위한 전처리

부호화 분야에서 연구된 결과에 따르면, 프로덕트텀 수를 줄이기 위해서는 출력기호들 사이에 특정한 관계가 성립해야 한다. 이와 같은 관계는 **지배**(Dominance)관계와 이접(Disjunction)관계로 대별된다.[4]

먼저 $enc(O_i)$가 $enc(O_j)$를 지배(Dominate)한다고 함은 $enc(O_j)$가 "1"의 논리값을 갖는 비트위치에서 $enc(O_i)$의 논리값은 항상 "1"임을 뜻하며 $\{enc(O_i) > enc(O_j)\}$로 표시된다. 따라서 $\{enc(O_i)=11, enc(O_j)=01\}$인 경우, $enc(O_i)$는 $enc(O_j)$를 지배하지

만 역은 성립하지 않는다. 지배관계에 의하여 프로덕트텀 수가 줄어드는 것을 그림 3에 보였다. 그림 3(b)에서 {enc(O_2)=001, enc(O_4)=011, enc(O_5)=101}이므로 {enc(O_4)>enc(O_2), enc(O_5)>enc(O_2)}가 성립하고 그림 3(c)에 나타난 바와 같이 두개의 프로덕트텀 {001 O_2,111 O_2}가 하나의 프로덕트텀 {--1 O_2}로 줄어든다. 이는 O_2의 on-set을 최소화하는데 O_4및 O_5의 on-set들이 don't-care set으로 작용했기 때문임을 알 수 있다. 이와 같이 O_2의 on-set을 최소화 하는데 필요되는 don't-care set은 **슈퍼큐브**(Supercube)란 용어를 도입하여 정의할 수 있다. 큐브들의 집합 C={Ci}의 슈퍼큐브는 C의 모든 큐브를 포함하는 최소의 큐브로 정의되며 sup(C)로 나타낸다. 그림 3(a)에서 O_2의 두 입력큐브 {001, 111}을 이들의 슈퍼큐브 {--1}로 줄이기 위해서는 이에 포함되는 큐브들의 집합, 즉, {011,101}이 don't-care로 필요하며, 따라서 그림 3(b)와 같은 지배관계가 성립하도록 부호화 해야한다.

001	O_2	001	001	--1	001
111	O_2	111	001	011	011
011	O_4	011	011	101	101
101	O_5	101	101		
	(a)		(b)		(c)

그림 3. 지배관계의 한 예
Fig. 3. Illustrating dominance relations.

2진수 코드 enc(O_i)가 enc(O_j)와 enc(O_k)의 비트별 논리합(Bitwise-OR)으로 표시될 때 **이접적 일치**(Disjunctive equality)관계가 성립하며 {enc(O_i)=enc(O_j) |enc(O_k)}로 나타낸다. 예를 들어 {enc(O_i)=111, enc(O_j)=011, enc(O_k)=110}인 경우 {enc(O_i)=enc(O_j) |enc(O_k)}가 성립한다. 이접적 일치관계에 의하여 프로덕트텀 수가 줄어드는 것을 그림 4에 보였다. 그림 4(b)의 지배관계로는 프로덕트텀 수를 줄일 수 없는 반면에 그림 4(c)의 이접적 일치관계는 프로덕트텀 {101 O_6}을 줄이는 결과를 가져온다. 이는 {101 O_6}가 {101 O_2,101 O_5}로 분리되어 O_2및 O_5의 입력큐브를 확장(Expand)시키는데 쓰였기 때문이다. 이 작용이 일어나기 위해서는 O_6의 입력큐브 {101}과 O_2의 입력큐브 {111} 사이의 햄밍거리(Hamming distance)가 1이어야 하고 O_6의 입력큐브 {101}과 O_5의 입력큐브 {100} 사이의 햄밍거리도 1이어야 한다.

101	O_6	101	00	101	11	10-	01
100	O_5	100	01	100	01	1-1	10
111	O_2	111	11	111	10		
		(a)		(b)		(c)	(d)

그림 4. 지배관계와 이접적 일치관계의 차이
Fig. 4. Dominance relation versus Disjunctive equality relation.

전처리는 명령어 해독기의 진리표에서 위와 같은 관계들을 찾아내는 것으로 다음의 3단계로 구성된다:

단계 1. 주어진 진리표를 출력별로 최소화(Out-put disjoint minimization)한다.

단계 2. 지배관계 그룹들을 찾아낸다.

단계 3. 이접적 일치관계 그룹들을 찾아낸다.

단계 1은 출력기호들 사이의 어떠한 관계도 배제한 상태(일반적으로 원핫(One-hot) 부호화를 사용함)에서 2단계 논리최소화하는 것이다.[3] 이때 얻어지는 프로덕트텀 수 NPT_{dis}는 최악의 부호화에 따른 프로덕트텀 수로서 임의의 부호화가 얼마나 좋은가를 판단하는 기본이 된다.

단계 2는 둘 이상의 기계언어로 구성된 기계언어 그룹에 적용된다. 기계언어 그룹 M_i의 입력큐브 집합을 $C_i=\{C_i^j\}$라 한다. 이의 두 큐브 C_i^j와 C_i^k ($j\neq k$)를 $\sup(C_i^j, C_i^k)$로 줄이기 위해 필요한 don't-care set $D_{ij}^{j,k}$는 다음과 같다:

$$D_i^{j,k}=\left\{C_p^l \middle| \left(C_p^l \cap \sup\left(C_i^j, C_i^k\right)\right)\neq\phi, p\neq i, \forall l\right\}$$

이 don't-care set을 확보하려면 $\{C_p^l \in D_i^{j,k}\}$인 각각의 큐브 C_p^l이 출력하는 O_p와 O_i사이에 {enc(O_p)>enc(O_i)}의 지배관계가 모든 p에 대하여 성립해야 한다. 이러한 지배관계들을 하나의 그룹으로 하여 $DRG_i^{j,k}$로 나타낸다. DRG의 저장모델이 그림 5(a)에 나타나 있다. 줄이려하는 두 입력큐브가 각각 3-1)항과 3-2)항에, 이들이 출력하는 기호가 1)항에 저장된다. 필요한 don't-care들이 출력기호과 함께 5)항에 저장된다. 2)항에 저장되는 값은 CRC(Cost Reduction Credit)로 필요한 don't-care가 모두 확보되는 경우에 줄어드는 프로덕트텀 수를 나타낸다. 이 값은 4)항에 저장되는 것이 없는 경우에는 1이고 그렇지 않은 경우에는 4)항에 저장되는 입력큐브의 갯수에 1을 더한 값을 갖는다. 4)항에는 [{sup(C_i^j, C_i^k)}$\supset C_i^l$] 이고 ($l\neq j,k$)인 입력큐브들이 저장된다. 그림 3(a)의 예를 DRG형태로 저장한 것이 그림 5(b)에 나타나 있다. 이와 같은 일을 서로 다른 두 입력큐

브의 모든 쌍에 대하여 수행하고 계속하여 다음 기계언어 그룹에 대하여 같은 일을 반복하여 수행한다.

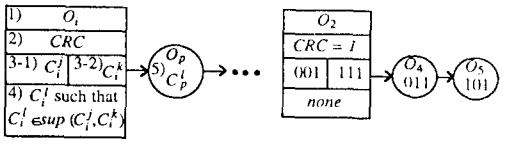


그림 5. 지배관계 그룹 저장모델

Fig. 5. Data model for storing dominance relation group.

단계 3에서는 각 입력큐브 C_i와 햄밍거리가 1인 모든 입력큐브들을 찾아 이접적 일치관계 그룹 $DERG_i^j$를 형성한다. 이의 모델이 그림 6(a)에 나타나 있다.

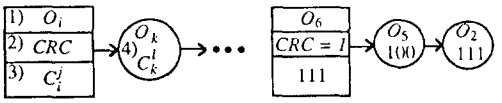


그림 6. 이접적 일치관계 그룹 저장모델.

Fig. 6. Data model for storing disjunctive equality relation group.

줄이려하는 프로덕트텀의 출력기호가 1)항에, 입력큐브가 3)항에 각각 저장된다. 2)항의 CRC값은 항상 1이 된다. 4)항에는 3)항의 입력큐브와 햄밍거리가 1인 입력큐브및 이의 출력심볼이 나열된다. 일반적으로 enc(O_i)가 m비트 2진수 코드라면 (m-1)개의 입력큐브가 4)항에 나열될 수 있는데 이들중 두개 이상의 입력큐브에 대응하는 출력기호들의 코드와 1)항의 출력코드간에 이접적 일치관계가 성립하면 1),3)항의 프로덕트텀을 줄일 수 있게된다. 그림 4(a)의 예를 DERG형태로 저장한 것은 그림 6(b)와 같다.

2. PLA 구현 비용계산 방법

마이크로코드 루틴의 주어진 선형배치에서 i번째와 j번째(i≠j)의 루틴을 맞바꾸면, 이들의 길이는 일반적으로 $S_i \neq S_j$이므로, 이들 사이에 위치한 루틴들의 시작번지 enc(O_l) (i<l<j)는 $|enc(O_l)+(S_j-S_i)|$로 바뀌며 새로이 j번째가 되는 루틴의 시작번지는 원래 j번째 루틴의 시작번지에 (S_j-S_i)를 더한 2진수 코드로 바뀐다.

이와 같이 정해지는 출력부호화 비용은 III.1절에서 찾아놓은 DRG및 DERG들을 얼마나 만족시키는가로써 계산된다. 이는 다음의 4단계로 나뉘어 수행된다:

단계 1. 두개의 출력기호로 된 모든 쌍에 대하여 지배관계 성립여부를 조사하여 저장하고 아울러 성립하는 경우를 세어 NUM_{dom}에 저장한다.

단계 2. 첫번째 마이크로코드 루틴을 사용하는 기계언어 그룹의 기계언어 수를 NPT^{red}에 저장한다.

단계 3. DRG들을 차례로 조사하면서 주어진 부호화가 이를 만족시키는 경우 이의 CRC값을 NPT_{red}에 더한다.

단계 4. DERG들을 차례로 조사하면서 주어진 부호화가 이를 만족시키는 경우 이의 CRC값을 NPT_{red}에 더한다.

단계 1에서 출력 쌍에서 성립하는 지배관계를 정리하는 이유는 이들이 DRG를 구성하는 기본이며 또한 DERG가 만족되기 위한 필요조건이 되기 때문이다. 단계 2에서는 PLA의 한시(Default)출력인 0코드로 부호화 됨으로써 구현할 필요가 없어지는 프로덕트텀 수를 찾아 줄어든 갯수를 저장하는 변수 NPT_{red}에 할당한다. 단계 3과 4를 거치면서 NPT_{red}값은 만족되는 모델의 CRC값 만큼 증가되어 최종적으로 줄어든 프로덕트텀 수를 나타내게 된다.

주어진 부호화의 비용은 $(NPT_{dis}-NPT_{red})$로 나타낼 수 있다. 실제로 실험에 사용한 비용계산 식은 다음과 같다:

$$(NPT_{dis} - NPT_{red}) + \alpha^* \frac{1}{NUM_{dom}} \qquad (1)$$

두번째 항을 첨가한 이유는 각개의 지배관계가 발생시키는 don't-care가 III.1절에서 모델링하지 못한 방식으로 프로덕트텀의 감소를 초래할 수 있는 경우를 고려하기 위한 것이다. 그렇지만 이러한 경우는 희소함으로 α값을 조정하여 비중치를 결정토록 하였다.

Ⅳ. 최소면적의 다단계회로로 구현하는 방법

본 장에서는 다단계 논리최소화 프로그램의 주요한 기교인 단일큐브(Single cube) 추출[6]을 모사하는 것과 최적의 2단계회로는 좋은 다단계회로로 변환된다는 실험결과[10]에 착안하여 다단계회로 구현 비용을 근사계산하는 방법을 설명한다. 후자를 위하여는 III장에서 설명한 방법을 사용할 수 있으므로 여기서는 단일큐브 추출을 모사하는 방법만을 설명한다.

1. 다단계회로 구현 비용계산을 위한 전처리

출력부호화에서 단일큐브 추출을 모사하는 방법으로 햄밍거리가 가까운 입력큐브들에 대응하는 출력기호들을 햄밍거리가 가까운 2진수 코드들로 부호화하는 것을 생각할 수 있다.[4] 여기서 입력큐브들 사이의 햄밍거리는 추출되는 단일큐브의 크기를 결정하고 출력코드 사이의 햄밍거리는 추출된 단일큐브가 공통으로 쓰이는 빈도 수를 결정한다. 이와 같은 방법은 가장 빈번히 나타나는 가장 큰 단일큐브를 우선 추출, 치환하는 다단계 논리최소화 프로그램의 기교[6]를 잘 모사하는 것이 된다.

전처리 과정에서는 출력기호들을 정점(Vertex)집합 V로 하는 완전가중그래프(Weighted complete graph) G(V,E,W(E))를 만들고 각 에지(Edge) ei의 가중치 W(ei)를 계산한다. $W(e_i)$값은 e_i의 두 정점을 출력하는 입력큐브들 사이의 최소 햄밍거리로 한다. 이 완전가중그래프를 이용하여 주어진 부호화의 비용을 계산하는 식은 다음과 같다:

$$f_I = \sum_i \frac{1}{W(e_i)} * dis\left(enc(v_j), enc(v_k)\right) \qquad (2)$$

여기서 dis()함수는 enc(v_j)와 enc(v_k)간의 햄밍거리를 계산한다. 표 1의 명령어 해독기에 대한 전처리 결과는 그림 7과 같다.

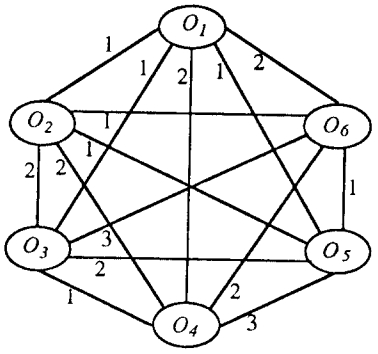


그림 7. 완전가중그래프의 일 예

Fig. 7. A weighted complete graph constructed for the instruction decoder shown in table 1.

이를 이용하여 그림 8에 나타난 두가지 부호화의 비용을 (2)식으로 계산하면 그림 8(a)의 경우 12.17이고 그림 8(b)의 경우는 14.67이 된다. 이 결과는 그림 8(a)의 결과가 8개의 리터럴을 갖는 다단계회로로 변환되며 그림 8(b)의 결과가 9개의 리터럴을 갖는 다단계회로로 변환되는 차이를 잘 반영하고 있다.

Before minimization		After minimization		Multi-level expression
Inputs	Outputs	Inputs	Outputs	
$x\,y\,z$	$f_1\,f_2\,f_3\,f_4$	$x\,y\,z$	$f_1\,f_2\,f_3\,f_4$	
0 0 0	0 0 0 0	0 1 -	1 0 0 0	$f_1 = x'y + f_2$
1 1 0	0 0 0 0	1 0 -	1 1 0 1	$f_2 = xy'$
0 0 1	0 0 1 1	- - 1	0 0 1 1	$f_3 = z$
1 1 1	0 0 1 1			$f_4 = f_2 + z$
0 1 0	1 0 0 0			
0 1 1	1 0 1 1			
1 0 0	1 1 0 1			
1 0 1				

(a)

Before minimization		After minimization		Multi-level expression
Inputs	Outputs	Inputs	Outputs	
$x\,y\,z$	$f_1\,f_2\,f_3\,f_4$	$x\,y\,z$	$f_1\,f_2\,f_3\,f_4$	
0 0 0	0 0 0 0	1 0 1	0 1 0 0	$f_1 = z$
1 1 0	0 0 0 0	0 1 -	0 1 0 1	$f_2 = f_3 z + x'y$
0 0 1	1 0 0 0	- - 1	1 0 0 0	$f_3 = xy'$
1 1 1	1 0 0 0	1 0 -	0 0 1 1	$f_4 = f_2 + f_3$
0 1 0	0 1 0 1			
0 1 1	1 1 0 1			
1 0 0	0 0 1 1			
1 0 1	1 1 1 1			

(b)

그림 8. 표 1의 명령어 해독기에 대한 두가지 출력부호화 결과

Fig. 8. Two feasible encoding results of the instruction decoder shown in table 1.

2. 다단계회로 구현 비용계산 방법

주어진 부호화의 비용계산 식은 다음과 같다:

$$\alpha^* f_I + \beta^* f_p \qquad (3)$$

첫째항 f_I은 (2)식으로 계산되는 비용으로 단일큐브 추출을 모사하며 두번째 항의 f_p는 (1)식의 첫항 (NPT_{dis}-NPT_{red})로 계산되는 비용으로 2단계회로 비용으로 다단계회로 비용을 근사하는 것이다. (3)식으로는 다단계 논리최소화 기법중 커널(Kernel)추출을 모사할 수 없는 한계가 있으나 종래의 다단계회로 구현을 위한 부호화 방법에서 쓰여온 단일큐브 추출 모사에 더하여 f_p항을 더한 것이 새롭다. 상수 a와 b를 도입하여 f_I과 f_p의 비중을 조절한다.

V. 실험 결과

앞에서 설명한 비용계산 방법을 사용하여 시뮬레이티드 어닐링에 근거하는 출력부호화 프로그램을 C언어로 작성하였다. 어닐링 루프를 제어하는 일반적인 기법은 기존 연구결과[8]를 따랐고 임의의 두 마이크로코드 루

턴의 순서를 맞바꾸는 것만을 무브(Move)로 삼았다. 각 온도단계에서 무브의 횟수는 300으로 한정하였고 이 횟수에 이르도록 평형(Equilibrium)에 도달하지 못하면 이의 3배만큼 더 확장되도록 하였다. 빠른 어닐링을 위하여 윈도우를 설정하고 임의로 선택된 마이크로코드 루틴으로 부터 앞뒤로 윈도우크기의 순서안에 드는 하나의 마이크로코드 루틴을 선택하여 무브하도록 하였다. 초기온도에서 윈도우크기는 출력기호의 갯수와 같고 온도가 단계별로 감소할 때마다 0.9배 크기로 감소되도록 하였다. 어닐링과정에서 최소비용을 갖는 부호화 결과을 저장하였다가 최종 부호화로 삼아 후처리단계에서 논리최소화 하였다.

표 2. 명령어 해독기의 예제
Table 2. Summary of examples.

	No. of input [bit]	No. of output [bit]	No. of output symbols	No. of prod. terms after output disjoint minimization
EX1	3	4	6	8
EX2	4	5	8	13
EX3	8	8	81	109
EX4	8	9	81	109
EX5	8	10	91	115
EX6	10	10	102	153

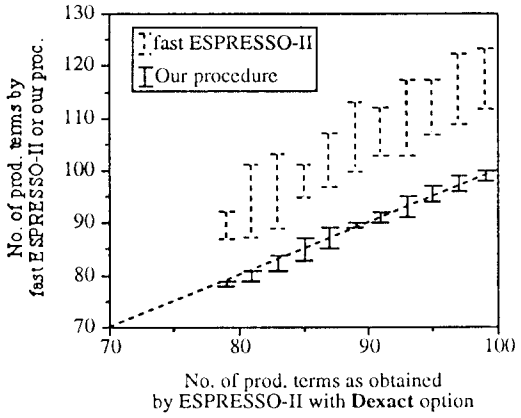


그림 9. 제안된 비용계산 방법과 기존 방법간의 정확도 비교 예
Fig. 9. A comparison of estimation accuracy between our cost estimation procedure and ESPRESSO-II with fast option for the case of EX4.

표 2에 실험에 사용된 명령어 해독기 예제들을 정리하였다. 그림 1에 보인 EX1과 EX2는 간단한 예제들이며 EX3부터 EX6은 CISC형 마이크로프로세서[11]의 1바이트 옵코드(Opcode)에 대한 명령어 해독기를 여러가지 형태로 변형시킨 것이다. 입력 비트 수, 출력 비트 수와 기계언어 그룹 수가 각각 첫째, 둘째 및 세째 단에 나타나 있다. 마지막 단은 출력별 2단계 논리최소화 후에 얻어지는 프로덕트텀 수 이다.

그림 9는 EX4에 대하여 f_p만을 사용하여 계산한 비용과 ESPRESSO-II(with fast option)을 사용하여 계산한 비용의 정확도를 비교한 것이다. 본 논문에서 제안한 방법이 보다 정확한 것을 알 수 있으며 이러한 결과는 EX3 및 EX5,6에서도 마찬가지 였다.

표 3에 PLA 구현에 대한 실험결과를 정리하였다. 'Random encoding'란에 나타난 자료는 EX1 및 EX2의 경우에는 모든 경우를 살핀 것이고 그외의 예제들에 대하여는 100개 부호화를 임의선택하여 ESPRESSO-II(with **Dexact** option)으로 비용을 계산한 것이다. 'E-FAST'란의 자료는 ESPRESSO-II(with **fast** option)을 비용계산에 사용한 시뮬레이티드 어닐링으로 EX1 및 EX2에 대하여는 10번, EX3부터 EX6에 대하여는, 많은 수행시간이 걸리는 이유로, 3번 수행한 결과이다. 'Our method'란의 자료는, (1)식에서 각 항에 의한 비용값의 비율을 조정하기 위하여

$$CR = (NPT_{dis} - NPT_{red}) / \{\alpha' (1 / NUM_{dom})\} \quad (4)$$

로 비용값 비율(CR)을 정의하고, CR=30 이 되도록 α를 조정하여 비용계산에 사용한 시뮬레이티드 어닐링으로 10번 수행한 결과이다. 각 란의 CPU시간은 총 소요시간을 수행횟수로 나눈 것이다. 'Our method'의 결과는 평균적으로 'Random encoding'보다 10~40%정도, 'E-FAST'보다 11%정도 좋았다. 또한 'E-FAST'보다 100배정도 적은 CPU 시간이 소요되었다.

표 3. PLA 구현에 대한 실험결과
Table 3. Experimental results for PLA implementation.

	Random encoding					E_FAST					Our method				
	No. of trials	No. of prod. terms			CPU time [sec]	No. of trials	No. of prod. terms			CPU time [sec]	No. of trials	No. of prod. terms			CPU time [sec]
		Min	Max	Avg			Min	Max	Avg			Min	Max	Avg	
EX1	6'	3	7	6.4	0.003	10	3	5	4.1	7.9	10	3	5	3.5	0.2
EX2	8'	7	12	10.9	0.05	10	7	8	7.6	19.4	10	7	9	7.7	0.3
EX3	100	96	106	101.7	4.2	3	74	80	77.3	22993.0	10	75	79	76.8	310.6
EX4	100	102	108	105.6	7.8	3	92	98	94.0	38110.9	10	79	85	81.3	322.8
EX5	100	107	114	111.0	8.7	3	99	101	99.7	32502.6	10	86	90	87.5	443.7
EX6	100	142	152	147.5	18.3	3	123	126	124.3	90397.9	10	107	119	110.7	645.4

Note)'CPU time' denotes average CPU seconds per trial on SPARCstation-2, which was obtained by dividing the total elapsed time by the number of trials.

표 4에 다단계회로 구현에 대한 실험결과를 정리하였다. 'Random encding'란의 자료는 100개 부호화를 임의선택하여 misII(with standard script)로 비용을 계산한 것이다. 'm-heuristic'란의 자료는 (3)식을 $\alpha=1, \beta=0$으로 하여 비용계산에 사용한 시뮬레이티드 어닐링을 10번 수행한 결과이다. 'mt-heuristic'란의 자료는, (3)식에서

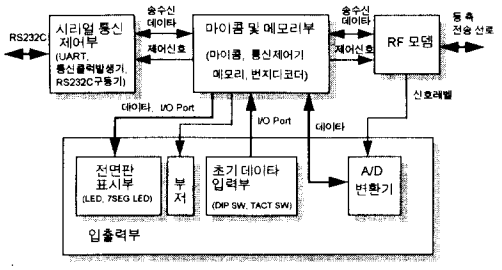
$$\alpha=\left\{f_{p,init}/\left(10^{*}f_{p,init}\right)\right\}^{*}t^{2}$$

(여기서 t = |(출력기호 수)-(윈도우크기)|/(출력기호 수) 이고$f_{l,init}$ 및 $f_{p,init}$는 초기 온도단계에서 계산된 f_l 및 f_p의 평균값임)

로 하고 $\beta=1$로 하여 비용계산에 사용한 시뮬레이티드 어닐링을 10번 수행한 결과이다. 2단계회로 비용으로 다단계회로 비용을 근사하는 것을 더하여 생각한 'mt-heuristic'이 'm-heuristic'보다 좋은 결과를 나타냈다. 'mt-heuristic'의 결과는 'Random encoding'보다 평균 10~34% 좋았다. 여기에 사용된 비용계산 방법은 misII(with **standard** script)보다 1000배 이상 빠른 것으로 관찰되었다. 전체적인 수행시간은 'Our method'의 1.5배 정도가 소요되었다.

표 4. 다단계회로 구현에 대한 실험결과

Table 4. Experimental results for multi-level implementation.



Ⅵ. 결론 및 토론

CISC형 마이크로프로세서와 같은 마이크로프로그램된 프로세서의 명령어 해독기를 최소면적의 PLA나 다단계회로로 구현하는 문제를 제한된 출력부호화 문제로 정의하였다. 시뮬레이티드 어닐링에 근거한 문제해결 방법들을 제안하였고 실험적으로 이들의 효율성을 증명하였다. 제안된 비용계산 방법들은 기존 논리최소화 프로그램들의 빠른 형태들보다 빠르고 정확하였다. 또한 다단계회로 비용을 예측함에 있어 2단계회로 비용으로 이의 다단계회로 비용을 근사하는 방법을 더하여 생각하는 것이 효과적임을 보였다.

제안된 방법들의 성능을 여러가지 측면에서 개선시킬 수 있을 것으로 생각한다. 어닐링 루프를 제어하는 기법이나 무브 종류를 설정하는 방법 및 낮은 온도에서 평형상태를 판정하는 방법등을 개선, 보완할 수 있을 것이다. 무엇보다도 II장에서 설정한 가정들을 완화시킴으로써 적용범위를 확대하려는 노력과 출력 상할당(Output phase assignment)기법[5] 등을 고려하여 보다 나은 해를 찾고자 하는 연구, 출력심볼의 수가 100개를 훨씬 상회하는 커다란 문제를 적정한 시간내에 해결하기 위한 기법을 찾는 것이 요구된다. 이에 대한 연구사례가 참고문헌[12] 에 나타나 있다.

參 考 文 獻

[1] B.E. Cline, *Microprogramming concepts and techniques*, New York/Princeton: Petroccelli Books, 1981.

[2] K. El-Ayat and R. Agarwal, "The Intel 80386 architecture and implementation", *IEEE MICRO*, Vol. 5, No. 6, Dec. 1985, pp. 4-22.

[3] G. De Micheli, "Computer-aided design and optimization of control units for VLSI processors", *Int. J. Circuit Theory and application*, Vol. 16, 1988, pp. 347-369.

[4] P. Asher, S. Devadas and A.R. Newton, *Sequential logic synthesis*, Boston, MA: Kluwer Academic, 1992.

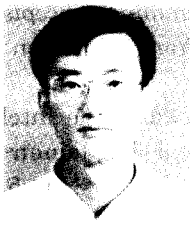
[5] T. Sasao, "Input variable assignment and output phase optimization of PLA's", *IEEE Trans. Computer*, Vol. c-33, No. 10, Oct. 1984, pp. 879-894.

[6] R.K. Brayton, R. Rudell, A. Sangiovanni-Vincentelli and A. Wang, "MIS: Multiple-level interactive logic optimization systems", *IEEE Trans. Computer-Aided Design*, Vol. CAD-6, Nov. 1987, pp. 1062-1081.

[7] S. Kirkpatrick, C.D. Gelatt Jr. and M.P. Vecchi, "Optimization by simulated annealing", *Science*, Vol. 220,

No. 4598, May 1983, pp. 671-680.
[8] M.D. Huang, F. Romeo and A. Sangiovanni-Vincentelli, "An efficient general cooling schedule for simulated annealing", *in Proc. Int. Conf. Computer-Aided Design*, 1986, pp.381-384.
[9] R.K. Brayton, G.D. Hachtel, C.T. McMullen and A. Sangiovanni-Vincentelli, *Logic minimization algorithms for VLSI synthesis*", Boston, MA:Kluwer Academic, 1984.
[10] W. Wolf, K. Keutzer and J. Akella, "Addendum to "kernel-finding state assigment algorithm for multi-level logic"", *IEEE Trans. Computer-Aided Design*, Vol. 8, No. 8, Aug. 1989, pp. 925-927.
[11] 386 DX microprocessor programmer's reference mamual, Intel Inc., 1989.
[12] Han-Heung Kim, *A study on the output encoding method for the practical design of the instruction decoder for microprogrammed controllers*, Ph.D. thesis, DEE 84083, KAIST, Taejon, Feb. 1994.

著 者 紹 介



金 漢 興(正會員)
1984年 서울대학교 전자공학과 학사. 1986年 한국과학기술원 전기 및 전자공학과 석사. 1986年 ~ 1988年 현대전자산업(주) 근무. 1994年 한국과학기술원 전기 및 전자공학과 박사. 1994年 ~ 현재 현대전자산업(주) 근무. 주관심 분야는 논리합성, Testing and Simulation 등임.

慶 宗 旻(正會員)
1975年 서울대학교 전자공학과 학사. 1977年 한국과학기술원 전기 및 전자공학과 석사. 1981年 한국과학기술원 전기 및 전자공학과 박사. 1981年 ~ 1983年 AT&T Bell Lab., Murray Hill, NJ. 근무. 1983年 한국과학기술원 전기 및 전자공학과 조교수. 1985年 Univ. of Tokyo 객원 교수. 1986年 한국과학기술원 전기 및 전자공학과 부교수. 1989年 독일 Karlsruhe 대학교 객원 교수. 1990年 ~ 현재 한국과학기술원 전기 및 전자공학과 교수. 주관심 분야는 CAD Algorithms for VLSI, 컴퓨터 그래픽스, VLSI Architecture for DSP and Computation 등임.

黃 承 浩(正會員)
1979年 서울대학교 전자공학과 학사. 1981年 한국과학기술원 전기 및 전자공학과 석사. 1981年 ~ 1984年 대우중공업 R&D 연구원 1984年 (주) 일성 연구개발 실장. 1989年 미국 U.C. Berkeley 공학박사. 1989年 ~ 1990年 미국 Schlumberger Tech. Inc. 소프트웨어 연구원. 1990年 한국과학기술원 전기 및 전자공학과 조교수. 1994年 ~ 현재 한국과학기술원 전기 및 전자공학과 부교수. 주관심 분야는 Simulation, Behavioral and Logic Synthesis, Verification, and Testing of VLSI Systems 등임.