마이크로 명령어의 코드 할당 알고리즘

°김학원, 김춘수, 홍인식, 임재윤, 임인칠
한 양 대 학 교

A Code Assignment Algorithm for Microinstructions

°H. R. Kim, C. S. Kim, I. S. Hong, J. Y. Lim, I. C. Lim
Hanyang University

ABSTRACT

In the case of VLSI computer system control unit design using PLA, optimal state code assignment algorithm to minimize the PLA area is proposed. An optimal state code assignment algorithm which considers output state and logic minimization simultaneously is proposed, and by means of this algorithm product term is minimized. Also, by means of this algorithm running time and memory capacitance is decreased as against heuristic state code assignment algorithm Which uses matrix calculation and considers the constraint relation only. This algorithm is implemented on VAX 11/750 (UNIX 4.3BSD). Through the various test example applied proposed algorithm, the efficiency of this algorithm is shown.

Ⅰ. 서 론

VLSI 시스템 설계의 복잡도가 증가함에 따라 자동설계 시스템이 부각하게 되었다. 이러한 자동설계 시스템은 그 기능에 알맞는 하드웨어 기술로부터 데이타부 및 제어부에 대한 상태를 추출한 후 이를 최적화 하며 특정 설계 기법에 의해 회로를 완성하게 된다. 이를 위해 각종 하드웨어 기술언어들이 개발되어 회로 기술에 따른 고유의 컴파일 알고리즘에 의해 특정 회로 형태의 데이타를 산출하고 이를 바탕으로 최종 마스크 패턴을 자동 생성한다. 그런데 이러한 자동설계 시스템으로 회로를 실현할 경우 제어부는 프로그램 가능하며 최소의 면적으로 회로를 실현해야 하는 문제점으로 인해 데이타부 설계에 비해 어려운 점이 많이 발생하고 있다. [1] 이러한 제어부의 설계는 그 규칙성 및 설계 시간의 잇점 등으로 인해 PLA(Programmable Logic Array)가 널리 사용되고 있으며 이의 최종 목적은 최소의 면적으로 해당 회로를 실현하는 것으로 함수최소화 및 접침(folding) 기법 등으로 면적의 최적화를 기하고 있다. [4] [5] 한편 제어부의 설계시 제어시킬 각 상태에 일정 코드를 할당함으로써 면적의 효율을 기하고 있다. 그런데 기존의 코드할당법으로는 그레이(gray) 코드 또는 휴리스틱한 상태할당법이 있는데 이러한 방법들은 출력의 상태를 고려하지 않고 상태들에 대한 제한 조건만을 고려 함으로 이를 함수 최소화 하여 회로로 실현하였을 경우 면적의 최소화를 기할 수 없는 경우가 많다. 또한 이 방법들은 상태할당시 행렬 계산을 반복 사용함으로서 데이타가 많아질 경우 과도한 수행시간 및 기억용량이 필요하게 된다.

따라서 본 논문에서는 마이크로 프로그램에 있어서 각 마이크로 인스트럭션에 대한 코드 할당시 회로의 출력 상태를 고려하고, 함수 최소화 기법에 알맞은 형태로 코드를 할당함으로서 최종 회로 실현시 면적의 최소화를 기하고 수행시간 및 기억용량을 감소시키기 위한 새로운 알고리즘을 제안한다.

본 알고리즘을 VAX 11/750(UNIX 4.3 BSD)에서 프로그램하여 여러 회로에 적용한 후 기존의 각종 방법과 비교 검토 함으로서 유효성을 보였다.

Ⅱ. 코드 할당 알고리즘

Ⅱ-1. 제반 정의 및 성질

할당하고자 하는 마이크로 인스트럭션을 s_i라 표시하고, 이때 수행되어야 할 각각의 마이크로 동작들을 f_j라 표시할 때 각 동작들 간의 관계를 표시하기 위해 동작간의 교차행열 및 연관행열을 정의한다.

(정의1) 각 마이크로 동작들 간의 공통된 1값의 갯수를 나타낸 행열을 교차행열 C라 한다.

이때 교차행열의 각 요소 C_{ij}는 f_i 및 f_j간의 공통된 1의 갯수를 나타내며 대각선에 대해 대칭이다. 이는 동일한 코드를 할당시 동시에 포함될 수 있는 값들을 나타내며 대각선상의 값은 해당 동작의 1의 갯수를 나타낸다.

(정의 2) 임의 동작의 0값에 대해 다른 동작의 1의 갯수를 나타낸 행열을 연관행열 R이라 한다.

이때 연관행열의 각 요소 R_{ij}는 f_i에는 없으나 f_j에는 존재하는 값을 나타내며 이는 대각선에 대해 비대칭이다. 예로 임의의 상태 및 이에 대한 동작들의 C 및 R 행열을 표시하면 다음과 같다.

	f1	f2	f3	f4
S1:1	0	0	0	
S2:1	0	0	0	
S3:0	1	1	1	
S4:1	0	1	0	
S5:0	1	0	0	

(a) 상태 예

	f1	f2	f3	f4
f1	3	0	1	0
f2	0	2	1	1
f3	1	1	2	1
f4	0	1	1	1

(b) 교차행열 C

	f1	f2	f3	f4
f1	0	2	1	1
f2	3	0	1	0
f3	2	1	0	1
f4	3	1	1	0

(c) 연관행열 R

그림 1. 상태 및 동작에 대한 교차행열 및 연관행열

(정의 3) 교차행열 C 및 연관행열 R에서 $C_{ij}=0$이면 f_i와 f_j는 서로 독립관계, $C_{ij}>0$이고 $R_{ij}=0$이면 f_j는 f_i에 종속, $C_{ij}>0$ 및 $R_{ij}>0$이면 f_i와 f_j는 서로 연관 되었다고 한다.

위 예에서 f1과 f2, f1와 f4는 서로 독립관계에 있고 f4는 f2에 종속되고, f1과 f2, f2와 f3는 서로 연관관계에 있다.

(정의 4) 임의 동작 f_i에 대한 교차행열의 대각선상의 값을 n이라 할 때 $M_i=\lfloor \log_2 n \rfloor$을 f_i의 최대 큐브라 한다. 여기서 $\lfloor A \rfloor$는 A보다 크지 않은 최대의 정수이다.

위 예에서 $M1=1$, $M2=1$, $M3=1$, $M4=0$이다.

이상의 정의에 따라 다음과 같은 성질이 존재한다.

(정리 1) 서로 다른 상태수를 n이라 할 때 이 상태에 부여할 최소 코드수는 $Cn=\log_2 n$이다.

(정리 2) 동작수를 P라 하고 할당할 코드수를 Cn이라 할 때 서로 다른 상태 할당을 위해선 $Cn \leqq P$이다.

Ⅱ-2. 코드 할당 알고리즘

본 알고리즘은 마이크로 인스트럭션의 최적 상태 할당을 목적으로 개발된 것으로 각 마이크로 인스트럭션을 하나의 상태로 보고, 이에 해당되는 마이크로 동작을 함수집합으로 간주하여 각 함수들을 최소의 면적으로 실현하기 위한 각 마이크로 인스트럭션별 코드 할당을 수행한다.

(단계 1) 주어진 마이크로 인스트럭션의 갯수를 N이라 할 때, 이에 할당될 최소 코드길이 Cn 및 교차행열과 연관행열을 구한 후, 각 마이크로 동작의 최대 큐브를 계산하여 최대 크기 큐브별로 각 함수를 배열한다.

(단계 2) 최대 큐브인 함수가 하나일 경우 단계 3으로 간다. 2개 이상일 경우 이들간의 최대 큐브크기의 연관 및 독립 관계에 있는 함수쌍을 선정하며 이 쌍이 다수일 경우 최대 함수크기를 갖는 쌍을 우선으로 선정한다. 이것도 동일할 경우 이 쌍들에 의한 나머지 함수들의 연관도가 높은 순으로 선정하고 그 이외에는 임의로 쌍을 선정한다.

(단계 3) 큐브의 총 갯수가 $\lfloor N/2 \rfloor$일 때까지 최대 큐브별로 모아 주그룹을 형성하고, 최대 큐브가 없을 경우 나머지 항들에 대해서 다음 최대 큐브별로 단계 3을 반복 수행한다.

(단계 4) 단계 3의 결과로 생성된 마이크로 인스트럭션의 그룹을 주그룹이라 하고 나머지 마이크로 인스트럭션을 부그룹으로 한 후, 주그룹에 대해선 할당할 코드의 최상위 비트에 0을 부그룹에 대해선 1을 할당한다.

(단계 5) 주그룹에 대해서 최대 큐브 그룹의 갯수가 2의 n승이면 각 그룹에 대해 다음 코드 차순으로 n차 gray 코드를 할당한다. 만일 2^n보다 크고 2^{n+1} 미만이면 이중 함수의 1의 갯수를 가장 많이 포함, 이것과 연관되는 함수의 1의 갯수가 최대인 순으로 2만큼 선정하여 다음 코드 차순으로 n차 gray 코드를 할당한다. 차수가 낮은 마이크로 인스트럭션에 대해서도 동일하게 반복한다.

(단계 6) 주그룹의 마이크로 인스트럭션에 대해 코드할당시 동일 함수가 최대로 포함되거나, 두 인스트럭션이 서로 동일 함수에 대해 포함관계에 있는 인스트럭션에 동일한 코드를 할당한다. 또한 기존의 할당된 코드와 포함관계에 있는 인스트럭션은 이미 할당된 코드와 다르고 한 비트 차이가 날 수 있도록 gray 코드를 산출하여 할당한다. 만일 주그룹 내의 인스트럭션들 간의 아무 포함관계가 존재하지 않을 경우 인스트럭션의 1의 갯수가 큰 순으로 gray 코드를 할당한다. 모든 주그룹내의 인스트럭션에 대한 코드할당이 끝날때까지 단계 5를 반복 수행한다.

(단계 7) 주그룹내의 각 인스트럭션에 대한 코드할당을 마친 후, 할당된 주그룹내의 각 인스트럭션에 대해 부그룹내의 낮은 차수에 대한 큐브 관계를 갖는 쌍을 발견하여 이것이 존재하면 주항의 해당 인스트럭션과 나머지 코드에 대해 동일한 코드를 할당한다.

(단계 8) 주그룹과는 관계가 없으나 부그룹내에서 인스트럭션간 인접 관계에 있는 인스트럭션이 존재할 경우 주그룹내의 각 인스트럭션에 대해 코드할당

한 것과 동일한 방법으로 코드 할당을 수행한다.

(단계 9) 미 할당된 부그룹내의 각 인스트럭션에 대해 주그룹내의 특정 인스트럭션과 인접 관계에 있으며 특정 함수를 커버할 수 있으면 우선적으로 주그룹내의 인스트럭션과 나머지 코드에 대해 동일한 값을 부여한다. 또한 부그룹내의 이미 할당된 인스트럭션과 인접관계에 있는 인스트럭션을 선정하여 gray 코드를 할당한다. 그렇지 않으면 임의의 gray 코드를 할당한다. gray코드 할당 순서는 이미 할당된 코드가 존재하면 증가 gray코드, 존재치 않으면 감소 gray코드를 할당한다.

(단계 10) 모든 인스트럭션에 대한 코드 할당이 완료되면 할당에 참여하지 않은 코드를 don't care 항으로 하고 이미 할당된 코드와 더불어 입력을 형성하고 마이크로 동작에 해당되는 각 함수를 출력으로 하여 다 입력, 다 출력 함수 최소화기인 PLAMIN의 입력으로 한 후 코드할당을 완료한다. 그렇지 않으면 미 할당된 인스트럭션에 대해 단계 9를 반복 수행한다.

Ⅲ. 알고리즘 적용예

본 알고리즘의 효율성을 보이기 위해 [6] 에서 제시한 마이크로 인스트럭션을 예로 들어 본 알고리즘에 따라 상태 할당을 수행하면 다음과 같다. 우선 수행할 마이크로 인스트럭션의 상태는 표1과 같다.

	f1	f2	f3	f4	f5	f6
S1 :	1	1	0	0	0	0
S2 :	1	1	0	1	0	0
S3 :	1	0	0	1	0	1
S4 :	1	0	0	0	0	1
S5 :	0	1	0	0	1	0
S6 :	0	1	0	0	0	0
S7 :	0	0	0	1	0	0
S8 :	0	0	1	1	0	0
S9 :	0	0	1	0	1	1
S10 :	0	0	1	0	0	0
S11 :	1	0	1	1	0	0

표1. [6]에서 제시한 마이크로 프로그램 예

이에 대해 프로그램 수행과정을 나타내면 다음과 같다.

(과정 1) 최소 상태수를 4로 선정하며 각 동작별 최대 큐브는 각각 4,4,4,4,2,2 이다. 또한 최대 항은 f1 및 f4이다.

(과정 2) 이 큐브들 중에서 서로 최대 큐브 연관, 독립 관계에 있는 동작을 선정하면 f1<—>f3 및 f2 <—>f3 등 두 쌍이 존재하는데 최대항을 포함하는 f1<—>f3 쌍을 선정한다. 이 두 쌍에 해당되는 인스트럭션을 주그룹, 나머지를 부그룹으로 분류한 후 주그룹은 코드의 최상위 비트에 0을 할당하고, 부그룹은 1을 할당한다.

(과정 3) 주그룹내의 두 독립 항들은 서로 다르게 할당되어야 하므로 한 항들에 대해서는 0, 다른 항들에 대해서는 1을 할당한다. 만일 이러한 항들이 여럿이라면 이 항들에 대해서는 gray 코드 식으로 할당한다.

(과정 4) 다음으로 주그룹들 중에서 각 코드 할당시 동일 코드에 의해 가장 많은 항들이 포함될 수 있고 기존에 할당된 코드와 서로 인접관계에 있는 쌍들을 우선적으로 할당하여 cover율을 높이는 방향으로 할당해 나간다. 이에 따라 주그룹의 각항에 코드를 할당하면 다음과 같다.

	C4	C3	C2	C1
S1 :	0	0	1	1
S2 :	0	0	1	0
S3 :	0	0	0	0
S4 :	0	0	0	1
S5 :	1	-	-	-
S6 :	1	-	-	-
S7 :	1	-	-	-
S8 :	0	1	1	0
S9 :	0	1	0	1
S10 :	0	1	1	1
S11 :	0	1	0	0

우선 한 비트 할당으로 공통 항이 가장 많이 포함되는 상태는 S3 및 S11이다. 따라서 C2 및 C1에 00을 할당한다. 이때 f1 및 f4의 항이 동시에 포함된다. 다음으로 S4 및 S9도 f6에 의해서 포함되므로 다음 상태인 01을 할당한다. 한편 S2는 S3와 f4에 대해 서로 인접하므로 S2와 한 비트 차이나게 할당한다. 이미 01 이 할당 되었으므로 한 비트 차이나는 10할당하고 S8도 S2와 f4에 의해 서로 인접하므로 동일한 값을 할당한다. 이때 나머지 항은 하나 밖에 없으므로 11을 할당함으로서 주그룹에 대한 할당을 마친다.

(과정 5) 마지막으로 이미 할당된 상태와 부그룹내의 미할당 상태 간에 포함 및 인접관계를 고려하여 나머지 상태 할당을 수행한다. 여기서 f5에 대해 S9와 S5는 인접관계를 가지므로 한 비트 차이나게 할당해야 되나 이미 한 비트가 차이나므로 나머지는 동일한 코드로 할당한다. 또한 S6와 S5도 서로 f2에 대해 인접하므로 다음 gray코드를 생성하여 S6에 할당한다. 마지막으로 S7은 S8과 인접하고, 이미 한 비트 차이가 나므로 S8의 다른 비트와 동일한 코드를 부여한다. 상태 할당 완료.

이상의 과정에서 할당된 코드를 나타내면 표2와 같다.

이때 don't care 항은 각각 1000, 1001, 1010, 1011, 1100 등 5개항이다. 표2의 결과와 이들 5개의 don't care항들에 의해 다입력, 다출력 함수 최소화기인 PLAMIN의 수

	C4	C3	C2	C1
S1 :	0	0	1	1
S2 :	0	0	1	0
S3 :	0	0	0	0
S4 :	0	0	0	1
S5 :	1	1	0	1
S6 :	1	1	1	1
S7 :	1	1	1	0
S8 :	0	1	1	0
S9 :	0	1	0	1
S10:	0	1	1	1
S11:	0	1	0	0

표2. 표1의 각 상태에 코드를 할당한 결과

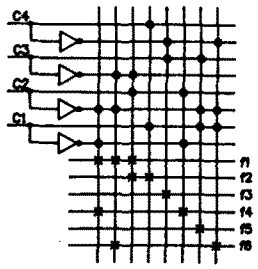

그림2. 표2의 코드에 대한 표1의 PLA형태

행 결과를 PLA로 나타내면 그림2와 같다. 여기서 적항선의 갯수는 총 8개이다. 한편 표1에 대한 예를 [2]에서 제시한 알고리즘에 의해 할당된 코드를 표3에 나타내었으며, 이를 최소화 한 후 그 결과를 PLA로 실현한 것을 그림3에 나타내었다.

	C4	C3	C2	C1
S1 :	1	0	0	0
S2 :	0	0	0	0
S3 :	0	0	1	0
S4 :	1	0	0	1
S5 :	0	1	0	0
S6 :	1	1	0	0
S7 :	1	0	1	0
S8 :	1	1	1	0
S9 :	0	1	1	1
S10:	1	1	1	1
S11:	0	0	0	1

표3. [2]의 알고리즘에 의한 코드할당

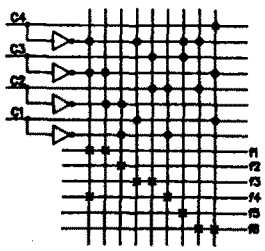

그림3. [2]에 의한 PLA 형태

여기서 [2]에 의한 상태 할당은 출력 상태를 고려하지 않고 상태들 간의 제한조건에 의한 코드할당을 수행하므로 이를 PLA로 실현시 최적설계를 보장할 수 없게 된다. 예에서 보듯이 본 알고리즘에 의한 결과는 [2]의 결과에 비해 약 11%의 면적 축소 효과를 가져 왔으며 [2]에서의 행렬 계산의 결과에 의한 코드 할당에 비해 수행 시간도 적게 소모되며, 함수 최소화기법을 염두에 두고 코드할당을 수행하므로 보다 효율적인 코드할당을 기대할 수 있다.

Ⅳ. 결 론

VLSI 시스템의 설계 중 마이크로 인스트럭션을 PLA로 실현할 경우 PLA의 면적의 최소화를 위해, 함수 최소화기법 및 출력값에 대한 상태를 종합적으로 고려한 최적 코드할당법을 제안하였다. 제안된 알고리즘은 컴퓨터 시스템 제어부를 PLA로 실현할 경우 면적의 최소화를 기할 수 있도록 최적 코드 할당을 수행하였다. 종래의 제한 조건만을 고려한 휴리스틱한 코드 할당에 비해 설계 효율이 증가하였으며, 기억용량 및 수행시간이 감소하였고, gray 코드 할당기법을 이용하므로서 코드 할당의 간소화를 도모하였으며 이를 함수 최소화기 및 PLA 자동생성기의 입력으로 직접 사용하여 설계할 수 있게 함으로써 마이크로 프로그램 제어부에 대한 자동 설계 시스템을 구성하였다.

앞으로의 연구과제로는 본 알고리즘을 확장하여 일반적 순서 회로의 제어부 설계를 자동화 할 수 있게 확장하고, 데이타부와 효율적으로 결합할 수 있는 하드웨어 기술언어의 확장 및 이를 자동적으로 실현키 위해 적절한 설계기법을 채용하여 설계의 전과정을 자동화하려는 실리콘 컴파일러의 제작이다.

참 고 문 헌

[1] M.J.Meyer, P.Agrawal, R.G.Pfister, "A VLSI FSM Design System," 21th DAC, pp.194-200.

[2] G.De Micheli, Robert K.Brayton, A Sangiovanni-Vincentelli, "Optimal State Assignment for Finite State Machine," IEEE Trans. on CAD.vol CAD-4. No.3, pp.269-285, July 1985.

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

[4] S.Kang, W.M.Vancleemput, "Automatic PLA Synthesis from a DDL-p Description," 19th DA Conf, pp.391-397.

[5] G.De Micheli, A Sangiovanni-Vincentelli, and T.Villa, "Computer aided Synthesis of PLA based finite state machine," in Int. Conf. on Comp. Aid. Des. Santa Clara, CA pp.154-157, Sept 1983.

[6] Bill Tell, A. Sangiovanni-Vincentelli, G. De Micheli, "A Finite State Machine Synthesis System," Proc of ISCAS 85, pp.647-650, 1985.

[7] 오창준, 이철동, 유영옥, "Microinstruction의 부호 할당에 관한 연구," 대한 전자공학회지 vol 25, No 1, pp.108-114, 1988.1.

[8] 김학림, 임재윤, 이기희, 홍인식, 임인칠, "하드웨어 기술언어 DASL의 설계와 VLSI FSM의 자동합성," 한국 정보 과학회, 봄 학술발표집, pp.295-298, 1988.