효율적인 면적의 제어부 실현을 위한 상태 할당 방법

○*박순규, *최성재, **조중휘, *정정화, *임인칠
*한양대학교 **인천대학교

State Assignment Method for Control Part Implementation of Effective-Area

**S.K.Park, *S.J.Choi, **J.W.Cho, *C.W.Jong, *I.C.Lim
*Hanyang University **Incheon University

ABSTRACT

In this paper, a new state assignment method is proposed for the implementation of the area-effective control part.

Introducing the concept of adjacency matrix to control table generated by SDL(Symbolic Description Language) hardware compiler, a state assignment method is proposed with which minimal number of flip flops and effective number of product terms can be obtained to accomplish the area-effective implementation.

Also, with substituting the assigned code to state transition table, boolean equations are obtained through 2-level logic minimization.

Proposed algorithm is programmed in C-language on VAX-750/UNIX and its efficiency is shown by the practical example.

Ⅰ. 서 론

VLSI 집적도가 증가함에 따라 복잡한 논리 회로를 짧은 시간내에 정확하게 설계하기 위해 설계 자동화 시스템이 요구되고 있다. 종래의 설계전문가에 의한 논리 설계는 칩 면적의 최소화 및 설계조건에 대한 최적화 목표를 달성할 수 있으나 과도한 설계시간의 소요와 설계오류를 포함할 수 있어, 설계 검증 및 재설계에 많은 시간이 소요되므로 설계비용이 증가하는 단점등이 지적되고 있다. 이를 위해 논리 설계 자동화 방법이 많이 연구되고 있다.

논리 설계 자동화는 설계조건의 동작 특성 또는 구조적 특성을 하드웨어 기술언어(HDL)로 기술한 후에 기술된 HDL을 컴퓨터에 의한, translation 과정을 거쳐 데이타 전송부와 제어부에 관한 정보를 갖고있는 테이블을 생성한 후 이로부터 논리 설계를 행한다. 일반적인 논리설계는 random logic 또는 array logic 방식으로 실현할 수 있다.[8,9] Random logic으로 실현할 경우 그의 불규칙성으로 인해 자동화하기가 어렵고 과다한 설계 시간의 소요로 인해 설계 비용이 증가한다. 반면에 array logic으로 실현하면 설계시간은 줄일 수 있으나 칩 면적의 증가로 인해 반드시 면적 최소화 과정이 요구된다.[1,3,5,6] 디지탈 시스템의 제어부를 FSM(Finite State Machine)[1,2,3,4,8]으로 모델화해서 array logic으로 실현할 경우 FSM의 출력 함수와 다음상태를 결정하는 boolean 함수식을 결정하기 위해 내부 상태(internal state)에 대한 상태 코드를 할당해야 한다. 그런데 각 상태에 코드를 어떻게 할당하느냐에 따라 좀더 간단한 함수식을 얻을 수 있어 조합 논리회로의 복잡성을 줄여 칩 면적을 감소시키고 시스템의 성능을 향상시킬 수 있다.[1,3,8] 상태할당 방법으로는 다음과 같은 것이 있다.

첫째, one-hot 코드를 이용하여 상태 할당을 행하는데, 이 방법은 각 상태에 하나의 플립플롭을 할당하여 FSM을 실현하므로 알고리즘이 간단하여 실현이 용이하나, 상태수가 증가하면 상태 변수의 증가 및 적항(product term)수의 증가로 인해 칩면적이 상당히 증가하는 단점이 있다.[4,6]

둘째, BCD 또는 gray 코드등을 인위적으로 할당하는 방법[7]은 각 상태에 하나의 플립플롭을 할당하는 방법에 비해서는 상태 변수 수를 줄일 수는 있으나, 이 방법 역시 칩 면적의 최소화를 달성할 수 없다. 세째, 휴리스틱한 방법에 의해 상태 할당하는 방법[3,5]은 앞의 어느 방법보다 칩 면적의 최소화를 달성할 수 있으나, 최소의 상태 변수 수($\lceil \log_2 N \rceil$, 단 N은 상태수)가 일부의 문제에 대해서는 보다 많은 상태 변수가 요구되어 칩 면적이 증가하는 경우가 있다.

본 논문에서는 이들 문제점을 해결하기 위해서 상태 변수 수를 최소로 하며 적항수를 효율적으로 줄일 수 있는 상태할당 방법을 제안한다. 우선 설계하고자 하는 디지탈시스템의 설계 요구조건을 하드웨어 기술언어인 SDL(Symbolic Description Language)로 기술하여 SDL 하드웨어 컴파일러에 입력시켜 제어부에 관한 정보를 갖고 있는 테이블을

얻는다. 제어부 테이블로부터 상태 천이 테이블을 얻어 각 상태에 할당된 코드에 관계없이 심볼 최소화〔4〕를 행하여 상태할당 과정에서 제약조건으로 사용될 인접 행렬(adjacency matrix)을 구한다. 그리고 인접 행렬에 대한 제약조건을 만족하는 상태 코드를 구한다. 각 상태에 할당된 코드를 상태 천이 테이블에 대치하여 논리 최소화를 행하면 효율적인 칩 면적으로 제어부를 실현할 수, 있다.

2. 준 비

설명의 간편화를 위해서 다음과 같은 구체적인 예로써 설명한다. 자동판매기에 들어온 동전의 금액을 판정한 신호를 받아 자동 판매기의 동작을 제어하는 Controller에 대한 설계요구 조건〔10〕을 하드웨어 기술언어로 기술한 후 제어부 테이블 및 상태 천이 테이블을 얻는다. 자동판매기에 대한 설계 요구조건을 하드웨어 기술언어인 SDL로 기술하면 그림 1과 같다. 이 SDL 기술을 하드웨어 컴파일러로 컴파일한 결과 그림 2와 같은 제어부 테이블을 얻는다. 제어부 테이블로부터 표 1과 같은 상태 천이 테이블을 얻는다.

```
SEQSDL:VENDING MACHINE.
INPUT:I1;
      I2;
      R.
OUTPUT:A;
       C.
SBEGIN:
        S1..
           C1.(R,!R)/(S1,C2)..
           C2.(!I1&I2,I1&I2,I1&!I2,!(I2|I1))/(S2,S3,S4,S1)..
        S2.A=1;
           C=1;
           ->S1..
        S3..
           C1.(R,!R)/(S1,C2)..
           C2.(I1,!I1)/(C3,S3)..
           C3.(I2,!I2)/(S5,S6)..
        S4..
           C1.(R,!R)/(S1,C2)..
           C2.(I1,!I1)/(C3,S4)..
           C3.(I2,!I2)/(S6,S3)..
        S5.A=1;
           ->S1..
        S6..
           C1.(R,!R)/(S1,C2)..
           C2.(!(I1|I2),I1|I2)/(S6,S5)..
SEND:
RESET:S1.
SYSTEMEND:
```

그림 1. 자동판매기 controller의 SDL 기술

present state	next state	condition
S1	S1	S1&!(I1|I2)&(!R)
S4	S1	S4&R
S2	S1	
S3	S1	S3&R
S5	S1	
S6	S1	S6&R
S1	S1	S1&R
S1	S2	S1&!I1&I2&(!R)
S3	S3	S3&!I1&(!R)
S1	S3	S1&I1&I2&(!R)
S4	S3	S4&!I2&(I1&(!R))
S1	S4	S1&I1&!I2&(!R)
S4	S4	S4&!I1&(!R)
S3	S5	S3&I2&(I1&(!R))
S6	S5	S6&(I1|I2)&(!R)
S4	S6	S4&I2&(I1&(!R))
S6	S6	S6&(!(I1|I2))&(!R)
S3	S6	S3&!I2&(I1&(!R))

그림 2. 제어부 테이블

I1I2R	present state	next state	A C
000	S1	S1	00
--1	S1	S1	00
010	S1	S2	00
110	S1	S3	00
100	S1	S4	00
---	S2	S1	11
0-0	S3	S3	00
110	S3	S5	00
100	S3	S6	00
--1	S3	S1	00
0-0	S4	S4	00
100	S4	S3	00
--1	S4	S1	00
110	S4	S6	00
102	S5	S1	10
000	S6	S6	00
--1	S6	S1	00
1-0	S6	S5	00
-10	S6	S5	00

표 1. 상태 천이 테이블

3. 상태 할당 알고리즘

3.1 심볼 최소화

상태 천이 테이블은(외부 입력, 현재 상태, 다음 상태, 제어 출력), 즉 (i, s, s′, o)의 4-tuple로 구성되는데, 외부 입력 변수와 제어 출력 변수값은 cube 형태로 표시되며, 현재 상태와 다음 상태는 심볼 형태로 표시된다. 상태 천이를 나타내는(i, s, s′, o)를 symbolic implicant 라고 하며, 이들의 집합을 symbolic cover라고 한다.〔3,4〕 심볼 최소화는 심볼 형태로 기술된 s, s′에 할당된 코드에 관계없이 최소의 symbolic implicant로 구성된 symbolic cover을 찾기 위한 것으로서 동일한 입력 변수값에 대하여 동일한 다음 상태로 천이하면서 동일한 출력 변수값을 내는 상태를 같은 그룹에 두어 각 그룹에 속한 상태에 할당된 코드가 최소의 subcube를 형성할 수 있도록 하는 상태 할당 코드를 얻기 위한 제약조건으로 이후의 상태 할당 알고리즘에서 사용된다. 그리고 이 상태 천이 테이블은 symbolic implicant의 현재 상태와 다음 상태에 대해 one-hot 코드를 사용하여 일치하는 위치에 1을 부여하고 나머지 위치에 0을 부여함으로서 다른 symbolic implicant로 구성된 상태 테이블을 얻는데, 이 테이블은 심볼 최소화를 위한 입력으로 사용되며, 심볼 최소화된 결과는 표 2와 같다. 심볼 최소화는 ESPRESSO-II〔11〕를 사용한다.

.p 16			
010	100000	010000	00
000	000001	000001	[illegible]
110	001000	000010	00
110	000100	000001	00
110	100000	001000	00
100	000100	001000	00
100	100000	000100	00
100	000010	100000	11
[illegible]	100000	100000	00
[illegible]	000100	000100	00
[illegible]	001000	001000	00
[illegible]	000001	000010	00
[illegible]	000001	000010	00
[illegible]	[illegible]	100000	00
[illegible]	010000	100000	11

표 2. 심볼 최소화된 결과

3.2 인접 행렬을 이용한 상태 할당

인접 행렬은 최소화된 symbolic implicant를 갖는 표 2의 현 상태 field에서 2개 이상의 상태를 포함하고 있는 field를 선택함으로서 얻어진다.

이 행렬의 row를 상태 그룹(state group)이라고 한다. 인접 행렬을 구하여 상태 그룹이 주어지면 각 그룹에 속하는 상태에 할당된 코드간에 서로 교차하지 않도록 하는 상태 할당 코드를 구한 다음에, 각 상태에 할당된 코드를 표 1에 대입한 뒤, 논리 최소화를 하면 심볼 최소화에 의해 얻어진 표 2의 symbolic implicant와 최소한 같은 수를 갖는 적항으로써 제어 출력 함수와 플립플롭의 다음 상태 함수를 실현할 수 있다. n_s를 상태수, n_b를 상태 할당된 코드의 길이, n_i를 인접 행렬 A의 row의 수, 즉 상태 그룹의 갯수라고 하면 가장 바람직한 상태 할당은 최소의 적항으로써 플립플롭의 다음 상태 함수와 제어 출력 함수를 실현 가능하게 하는 상태 할당을 말한다.

〈정의 1〉 (0, 1, −, *)을 원소로 갖는 Pseudo-boolean 변

수에 대한 AND와 OR 연산은 표3과 같이 정의된다.

여기서 "−"는 0 또는 1을 의미하는 don't care 를 의미하며, "*"는 0도 1도 아닌 공집합을 의미한다.

∧	0	1	−	*
0	0	*	0	*
1	*	1	1	*
−	0	1	−	*
*	*	*	*	*

∨	0	1	−	*
0	0	−	−	0
1	−	1	−	1
−	−	−	−	−
*	0	1	−	*

표3. Pseudo-boolean 변수에 대한 AND, OR 연산

〈정의2〉 인접 행렬 A는 다음과 같이 정의된다.

$$A = [a_{ij}]_{n_i \times n_s}$$

단 $a_{ij} = \begin{cases} 1 & \text{상태 } j\text{가 그룹 } i\text{에 속할 경우} \\ 0 & \text{그외 경우} \end{cases}$

인접 행렬의 임의의 행이 두개 이상의 상태 그룹의 AND로 이루어져 있을 경우 그 행을 *meet* 행이라고 말하며 meet행이 아닌 행을 prime행이라고 한다.

〈정의3〉 상태 코드 행렬 S는 다음과 같이 정의된다.

$$S = [s_{ij}]_{n_s \times n_b}$$

단 $s_{ij} \in (0,1)$

〈정의4〉 선택 규칙은 다음과 같이 정의된다.

$A \in (0,1)^{p \times b}$, $B \in (0,1,*,0)^{q \times r}$ 일 때

$$A \cdot B = [C_{ij}]^{p \times r}$$

여기서 $C_{ij} = \vee_{k=1}^{q} s_{ik} \cdot b_{kj}$ 이고,

$$s_{ik} \cdot b_{kj} = \begin{cases} b & \text{if } a = 1 \\ * & \text{otherwise} \end{cases}$$

〈정의5〉 face 행렬 F는 다음과 같이 정의된다.

$$F = [f_{ij}]_{n_f \times n_b}$$

단 $f_{ij} \in (0,1,-,*)$

$$F = A \cdot S$$

즉, 인접 행렬에 의한 상태 코드의 선택 규칙에 의해서 결정된다. 만약 인접 행렬 A가 다음과 같고, 구하고자 하는 상태 코드 행렬 S가 다음과 같을 때,

$$A = \begin{bmatrix} 0110001 \\ 1001000 \\ 0001001 \end{bmatrix} \quad S = \begin{bmatrix} 0110001 \\ 1100010 \\ 0010110 \end{bmatrix}^T$$

face 행렬 F는 아래와 같다.

$$F = A \cdot S = \begin{bmatrix} 0110001 \\ 1001001 \\ 0001001 \end{bmatrix} \bullet \begin{bmatrix} 010 \\ 110 \\ 101 \\ 000 \\ 001 \\ 011 \\ 100 \end{bmatrix} = \begin{bmatrix} 1-- \\ 0-0 \\ -00 \end{bmatrix}$$

여기서 F의 i번째 row인 f_i는 같은 상태 그룹에 있는 상태에 할당될 상태 코드의 bitwise OR를 나타낸다. 즉 상태 그룹 1의 $f_1 = 1--$는 100, 110, 101, 111의 bitwise OR이므로 그룹 1에 속한 s2, s3, s7은 100, 110, 101, 111중에서 하나씩 할당되어야만 인접 행렬에 대한 제약 조건을 만족하는 상태 할당 코드를 얻을 수가 있다.

$\overline{A}$가 인접 행렬 A의 complement이고 $\overline{F_i} = a_{i1} \cdot s_1$. 이라 할때

$$\overline{F_i} \wedge F = *, \quad 1 \le i \le n_s$$

상태 코드 행렬이 이 조건을 만족 할때 주어진 인접 행렬 A에 대한 제약 조건을 만족한다고 한다. 제약 조건을 만족하면서 *최소의 column*, 즉 최소 코드 길이를 갖는 상태 코드 행렬을 구하는 것이 상태 할당 문제의 목적이라 하겠다.

$$A = \begin{matrix} A_P \\ A_M \end{matrix}$$

단, A_P는 prime row

A_M는 meet row

라고 할때 S가 A에 대해 제약 조건을 만족하면 S는 A_P에 대한 제약 조건을 만족한다. 따라서 A의 모든 prime row만 고려함으로써 문제의 크기를 줄여 보다 빠른 시간에 상태 코드 행렬을 구하는 것이 가능하다.

3.3 알고리즘 기술

최적의 상태 코드 할당을 찾는 것은 NP-compl-

```
Input: adjacency matrix
Output:state code matrix

state_encode()
{
   do{
        state_select();
        do{
             candicodegen();
             constcheck();
             code_select();
             if(the selected code is not)
                  bit_increase();
         while(the selected code is not);
             A<- code_assign();
    while(all state is not assigned)
    B<-BCDcode_assign();
    C<-graycode_assign();
    compare(A,B,C);
```

ete 문제이므로 인접 행렬에 대한 제약 조건을 만족하는 상태 코드를 할당하는 휴리스틱 알고리즘을 제안한다. 상태 코드를 할당하는 알고리즘은 위와 같다.

4. 실험 및 결과

이상에서 기술된 알고리즘은 VAX/750 UNIX상에서 C 언어로 수행하였다. 표 1의 상태 천이 테이블에서 각 상태에 인접 행렬 개념을 이용하여 각 상태에 할당된 코드는 아래와 같다.

```
********** state code **********
 S1 = 000
 S2 = 001
 S3 = 100
 S4 = 110
 S5 = 101
 S6 = 010
```

.p 12	.p 17	.p 14
100-00 01000	000110 01000	-10010 01000
0100-0 00100	110100 11000	0-0110 10000
11010- 10100	100101 00110	-00110 01000
000-1- 01000	100100 01100	100-01 11000
-1011- 01000	0100-1 01000	-10-01 01000
1-00-0 10000	110001 01100	100-11 00110
100-1- 10000	1000-1 10000	110-10 10100
1-001- 00100	110011 10100	0-0-10 01000
-1001- 10000	-00011 01000	1-0010 10100
---0-1 00001	0--01- 00100	--1-01 00100
0-01-- 10000	0-01-0 10000	0--0-1 00100
-----1 00010	00-0-1 00100	--0101 10100
	1-0110 10100	--1-10 00100
	--11-0 00100	---011 00111
	-1--10 00100	
	--10-1 00100	
	---010 00111	

표4. 본 알고리즘의 경우

표5. BCD 할당

표6. gray 할당

이렇게 할당된 코드를 표 1에 대입하여 얻은 테이블에 대한 논리 함수 최소화를 행하면 표 4와 같은 테이블을 얻는다. 표 4를 보면 테이블의 row는 적항(product term)을 표시한다. PLA로 실현할 경우 플립플롭의 입력 함수와 제어 출력 함수는 12개의 적항으로 실현할 수 있음을 알 수 있다. 표 5, 6은 각 상태에 BCD, gray code로 할당했을때 표 1에 대입한 후 논리 함수 최소화를 행한뒤 얻은 테이블로 인접 행렬을 고려해서 상태 할당을 행한 경우 보다 적항수가 증가함을 알수 있다.

위에서 보는 바와 같이 인접 행렬을 고려하여 상태 할당을 행한뒤 논리함수에 대한 논리 최소화를 취하여 디지탈 시스템의 제어부를 보다 적은 적항수로 실현할 수 있다.

5. 결 론

본 논문에서는 디지탈 시스템의 제어부를 array logic으로 실현하는데 있어서 칩 면적을 줄이기 위한 새로운 상태 할당 방법을 제안했다. 시스템에 대한 설계 요구 조건을 하드웨어 기술 언어 SDL(Symbolic Description Language)로 기술한 후 SDL 하드웨어 컴파일러를 이용하여 제어부 테이블을 생성하고, 이 테이블로부터 상태천이 테이블을 생성하였다. 심볼 형태로 기술된 상태천이 테이블에 심볼 최소화 과정을 거쳐 인접 행렬을 구하고 인접 행렬에 대한 제약 조건을 만족하는 상태 코드를 할당함으로써 상태 변수 수와 적항수를 효율적으로 줄여 제어부를 효율적인 칩 면적으로 실현할 수 있었다. 제안된 알고리즘은 VAX-750 UNIX상에서 C 언어로 프로그램하여 실제 예에 적용해 봄으로써 그의 효용성을 입증했다.

참 고 문 헌

1. Bill Tell, A. Sangiovanni-Vincentelli, G. De Micheli, "A Finite State Machine Synthesis System", Proc. of ISCAS 85, pp. 647-650, 1985.
2. Douglas W. Brown, "A State-Machine Synthesizer-SMS", 18th Design Automation Conf., pp. 301-305, 1981.
3. G. De Micheli, Robert K. Brayton, A. Sangiovanni-Vincentelli, "Optimal State Assignment for Finite State Macine", IEEE Trans. on CAD, Vol. CAD-4, No. 3, pp. 269-285, July 1985.
4. G. De Micheli, "Symbolic Minimization of Logic Functions", ICCAD, pp. 293-295, 1985.
5. G. De Micheli, R. Brayton, "KISS : A Program for Optimal State Assignment of Finite State Machine", ICCAD, pp. 209-211, 1984.
6. A. R. Newton, "Techniques for Logic Synthesis", VISI 85 International Conference, pp. 27-42, Aug. 1985.
7. S. Kang, W. M. VanCleemput, "Automatic PLA Synthesis from a DDL-P Desoription", 19th Design Automation Conference, pp. 391-397.
8. Jung-Hwee Cho, "A Study on The Symbolic Description Language for VLSI Logic Design Automation", PH.D. dissertation, Univ. Han Yang, 1986.
9. Parker, "Automated Synthesis of Digital Systems", IEEE Trans. Design & Test, Vol. 1, no. 4, pp. 75-81, Nov. 1984.
10. F. J. Hill, G. R. Peterson, "Digital Logic and Microprocessors", John Wiley & Sons, 1984.
11. R. K. Brayton, etc, "Logic Minimization Algorithms for VLSI Synthesis", Kluwer Academic Publishers, 1984.