順序 集合 槪念을 利用한 多出力 論理函數 最少化 알고리즘

백 영 석* 김 태 헌* 이 성 봉* 정 정 화*
한양 대학교 전자 공학과*

Multi-Output Logic Minimization Algorithm Using the Concept of Ordering Set

Young-Suck Baek*, Tae-Hun Kim*, Seong-Bong Lee*, Jong-Wha Chong*
Department of Electronics Engineering, Hanyang University*

ABSTRACT

In this paper, a new multi-output logic minimization algorithm is presented. A base minterm is selected in the given function and the prime implicant is obtained by expanding it in the order of the expansion set that is decided by heuristic method. Input-oriented expansion procedure is used to reduce fan-in and fan-out number. To show the effectiveness of this algorithm, comparative run time with other minimization algorithm is given.

1. 서 론

LSI/VLSI의 집적도가 증가함에 따라 manual 설계 방법으로는 설계 면적에 대한 최적화 목표는 어느정도 달성할 수는 있으나 설계시간이 증가하는 단점이 있다. 따라서, 디지탈 설계에 대한 정확성의 보장과 설계 시간을 단축하기 위한 논리 설계 자동화 에 대한 연구가 크게 요구되고 있다. 그러나, 논리 설계 자동화에의한 설계 면적은 많은 redundancy를 포함하게 되어 manual 에 의한 설계보다 크게 되는 단점이 있으며, 따라서, 이를 해결하기 위해 논리 함수 최소화에 대한 연구가 많이 진행 되고 있다.

논리최소화에 대한 방법은 크게 두가지로 구분할 수 있다. 첫째 방법은 주어진 논리 함수의 모든 prime implicant 항들을 생성하여 이것으로 부터 minimal cube를 간추려 내는 방법으로 QM방법,MINI[1] 등이 있으나 QM 방법은 많은 memory와 많은 시간을 필요로 하며, MINI는 보수항을 만들어야하는 단점이 있다. 또한, 최근에는 MINI를 개선한 ESPRESSOII[2]와 ESPRESSOII를 개선한 McBOOL[3] 이 발표되어 좋은 효과를 보고 있다. 그러나, 일반적으로 이상의 방법은 생성된 모든 prime implicant 들에 비해 minimal cube 가 매우 적기 때문에 minimal cube 가 아닌 prime implicant 들을 생성하는데 대부분의 시간이 소요되어 매우 비효율적이며 [8], 변수와 vertices 의 수에 대해 memory 와 시간이 지수적으로 증가하는 단점이 있다.

이를 해결하기위해 제시된 두번째 방법은 휴리스틱 (heuristic)함수를 이용하여 최종적인 prime implicant 들만을 구하는 방법으로 DSA 방법[4], Zosimo Arevalo 의 방법[5], 그리고, Biawas의 CAMP[6] 와 CAMP 를 다출력으로 확장 시킨 MOM[7] 등이 있다. 그러나 이 방법은 전자의 방법보다는 효과적이기는하나 DSA 방법은 cycle 을 해결해야하는 단점이 있으며, Arevalo 의 방법은 하나의 prime implicant 를 생성하기위해 기준 최소항과 같은 거리에 있는 최소항들을 모두 취해 product항을 만들어야 하므로 비효율적인 단점이 있다. 또 CAMP 와 MOM은 selective prime implicant 를 생성할 때 여러번 CPT(Candidate Product Term) 을 생성해야하므로 시간이 많이 걸리는 단점이 있다.

따라서, 본 논문에서는 순서 집합 개념을 도입하여 좀더 효과적으로 논리함수를 최소화 하는 방법을 제시한다.

2. 다출력 함수의 논리 최소화

(1) 용어 정의

[1] 거리 d(m,n) : 2진수 표현의 두 cube m, n 에서 서로 다른 bit의 갯수.

[2] 인접집합 A(m) : $A(m) = \{n \mid d(m,n)=1,\ m,n \in F\}$

[3] 인접도 : $\mid A(m) \mid$.

[4] 후보집합 C(m) : 기준 최소항 m의 인접 집합에서, 최소항 m의 cover 되지 않은 출력을 모두 가지고 있는 원소들의 집합.

[5] 후보 인접도 : $\mid C(m) \mid$

[6] 순서 집합 : 집합에서 원소뿐 아니라 원소의 순서에 의해서도 결정이 되는 집합. Aord, Bord ... 와 같이 집합 기호에 ord 첨자 를 붙여 표시.

[7] 순서 후보 집합 Cord(m) : 최소항 m 의 후보집합 C(m)에 주어진 휴리스틱 방법을 이용하여 C(m) 원소의 순서를 결정한 순서 집합.

[8] 빈도수 : 다출력 함수에서 최소항이 cover 되어야 할 출력의 수.

[9] 집합 연산자 +,- : A + k (또는 -)는 집합 A의 각 원소에 k 의 값을 + (-)한 값을 원소로하는 집합이다.

(2) 기준 최소항 선택

최소항들의 상태를 표시하는 mode는 세가지가 있다. 첫째, EPI mode 는 그 최소항을 기준 최소항으로 선택하여 확장 했을 때 essential prime implicant를 생성할 가능성이 있는 상태를 표시하며, 둘째, SPI mode는 그 최소항을 기준 최소항으로 선택하여 확장하였을 때 essential prime implicant 를 생성하지 못한 상태를 표시하며, 셋째, COV mode는 그 최소항이 이미 생성된 prime implicant 들에의해 cover된 상태, 즉 빈도수가 0 인 상태를 나타낸다. 입력으로 들어온 최소항들의

처음 상태는 모두 EPI mode 로 놓는다. Essential prime implicant는 반드시 최종적인 해에 포함이 되므로, 각 빈도수에서 먼저 essential prime implicant를 생성하도록 최소항을 선택한다. 여기에서 기준 최소항 선택 방법은 EPI mode 와 SPI mode에서 기준 최소항을 선택 비교하여 최종의 기준 최소항을 결정한다.

1) EPI mode 에서의 기준 최소항
EPI mode 에서의 기준 최소항은 EPI mode에 있는 최소항 가운데 빈도수가 최소인 최소항으로 선택한다.
2) SPI mode 에서의 기준 최소항
SPI mode 에서의 기준 최소항은 다음 순서에의해 선택한다.
가) SPI mode에 있는 최소항중 최소의 빈도수를 갖는 최소항.
나) 가)의 조건을 만족하는 최소항이 다수일때, 각 최소항의 인접집합에 COV mode에 있는 최소항의 수가 가장 많은 최소항.
다) 나)의 조건을 만족하는 최소항이 다수일때, 임의로 선택한다.

EPI mode 와 SPI mode에서 선택된 기준 최소항 들로부터 최종적인 기준 최소항을 선택한다.

```
기준_최소항_선택()
{
    i = EPI_기준_최소항();
    j = SPI_기준_최소항();
    if (i의 빈도수 <= j의 빈도수)
        return(i);
    else
        return(j);
}
```

(3) 휴리스틱(heuristic) 방법과 확장 집합

휴리스틱 방법에의해 생성된 순서 후보 집합의 원소의 나열순은 기준 최소항에의해 생성될 수 있는 prime implicant들 가운데서, cover되지 않은 최소항들을 가장 많이 cover하며, 또 가장 큰 prime implicant를 생성 시킬 확률이 높은 순서이다. 휴리스틱 방법은 선택된 기준 최소항의 후보집합을 대상으로 한다. 이 휴리스틱 방법은 후보 집합을 순서 후보 집합으로 변화 시키는 것으로 다음과 같은 순서로 순서 집합을 얻는다.

여기서, >는 순서집합의 원소나열 순서를 나타낸다. 즉, a > b 는 a가 b보다 앞선 순서임을 나타낸다.

1) 기준 최소항이 cover하고자 하는 출력에 대하여, cover되지 않은 최소항 > cover된 최소항.
2) 1) 의 조건을 만족하는 최소항이 다수일 때,후보 인접도가 큰 최소항 > 후보 인접도가 작은 최소항.
3) 2) 의 조건을 만족하는 최소항이 다수일 때, 그 최소항의 후보집합 중에서 기준 최소항의 cover되지 않은 출력들이 cover 되지 않은 최소항의 수가 많은 것 > cover 되지 않은 수가 적은 것.
4) 3) 까지 같은 순위의 최소항이 여러개 있으면 임의로 나열한다.

여기서, 단일 출력을 대상으로 한다면, 후보 인접도는 그 최소항의 인접도와 같아지며, 후보집합은 인접집합과 같게 된다.

이 휴리스틱 방법에의해 생성된 순서 후보 집합의 원소의 나열 순은 기준 최소항에의해 생성될 수 있는 prime implicant 들 가운데서, cover되지 않은 최소항들을 가장 많이 cover 하며, 또 가장 큰 prime implicant를 생성 시킬 확률이 높은 순서이다.

휴리스틱 방법에의해 얻어진 기준 최소항 m 의 순서 후보 집합 Cord(m)에서 확장 집합 Eord(m) 을 얻는 방법은 다음과 같다.

```
확장_집합_생성(m)
{
    if (m의 mode == EPI)
        Eord(m) = C(m) - m;
    else
        Eord(m) = Cord(m) - m;
}
```

EPI mode 에서는 확장 순서가 필요없으므로 시간을 줄이기 위해 휴리스틱 방법에의해 순서를 결정하지 않고 후보 집합 C(m)의 원소의 순서를 임의로 취해 확장 집합을 생성한다.

정리 1 : 확장집합의 각 원소들의 값의 절대값은 모두 2 의 지수승이다.
(증명) 인접집합의 각원소들은 기준 최소항과 인접한, 즉 거리가 1 (값의 차이가 2의 지수승)인 원소들의 집합이므로 2진수 표현에서 1bit 차이가 난다. 또한, 순서 후보 집합은 인접 집합의 부분 집합인 후보 집합의 각 원소의 순서만 정한 것이고 값의 변화는 없으며, 확장 집합은 이 순서후보 집합의 각 원소에 기준 최소항의 값을 뺀 값이므로 확장집합의 원소들의 절대값은 2 의 지수승이다.

(4) 확장 프로시듀어

확장 프로시듀어는 선택된 기준 최소항과 휴리스틱 방법을 적용해 얻어진 확장 집합에의해 prime implicant를 생성하는 부분으로 최적에 가까운 해를 산출하게 한다. 여기에서 제안하는 확장 프로시듀어는 아래와 같다.

```
확장_프로시듀어(m, Eord(m))
{
    T = { m }
    for ( ; Eord(m)의 각 원소 k에 대하여 ; ) {
        S = T + k;
        if ( S에 주어진 함수를 만족하지 않는
             최소항이 있으면)
        {
            switch (m의 mode)
              case EPI :
                   m의 mode = SPI;
                   return;
              case SPI :
                   goto LABEL;
        }
        T = T U S;
        cpt = cpt  OR  abs(k);
        LABEL :
            continue;
    }
    출력_확장_프로시듀어(T);
    PRINT_OUT(m,cpt);
    COVER(T);
}
```

확장_프로시듀어에서 essential prime implicant 항은 다음 정리 2의 성질을 가지고 있다.

정리 2 : 선택된 기준 최소항이 확장 집합의 모든 방향으로 확장된다면, 그 기준 최소항으로 생성된 prime implicant는 essential prime implicant이다.
(증명) 이 정리는 참고문헌 [11]에 증명되어 있으므로 증명을 생략한다.

정리 3 : 선택된 기준 최소항의 후보 인접도가 0 이거나 1 이면 이 최소항으로 생성된 prime implicant 는 essential prime implicant 이다.
(증명) 후보 인접도가 0 이라는 것은 자기 자신이며, 후보 인접도가 0이 아니면 최소항 한번은 확장이 되므로 후보 인접도가 1 인 경우는 확장 집합에 있는 모든 원소와 확장이 된다. 따라서, 정리 2 에의해 항상 essential prime implicant 가 된다.

변수 cpt 는 기준 최소항을 확장 시켰을 때, 확장이 성공한 방향을 나타내는 것으로 해당하는 bit 를 1 의 값으로 표시한다. cpt 를 계산하는데 사용하는 연산자 OR 은 bit OR 를 나타낸다.

확장 집합 Eord(m) 의 각 원소가 입력부분으로 확장시키는 작업을 하는 반면, 출력_확장_프로시듀어는 prime implicant 를 출력방향으로 확장하는 작업을 한다. 입력 부분으로의 확장이 모두 끝난 후, 생성된 최소항들이 모두 기준 최소항이 cover 되지 않는 출력들 이외의 다른 공통의 출력을 갖는다면 그출력 방향으로 확장을 하여 더 큰 출력을 생성한다. 그러나 출력 부분으로 확장 부분이 모두 cover된 경우, 그 출력의 방향으로 확장하면 최종적인 회로에서 불필요한 fan-out 만증가하게 되므로 그방향으로 확장을 행하자 않는다. COVER(T) 프로시듀어에서는 집합 T 에 있는 모든 원소들의 출력 가운데 출력_확장_프로시듀어에의해 생성된 출력 부분을 cover하고 변동된 빈도수를 재조정한다.

(5) 다출력 함수의 논리 최소화 알고리즘

앞에서 기술한 프로시듀어들을 이용하여 본 논문의 알고리즘 LOMIO (LOgic MInimizer using Odering set) 를 기술하면 다음과 같다.

```
LOMIO( )
{
    최소항_생성();
    준비_프로시듀어();
    for ( ; 모든 최소항이 cover될 동안; ) {
       m = 기준_최소항_선택();
       Eord(m) = 확장_집합_생성(m);
       확장_프로시듀어(m, Eord(m));
    }
}
```

LOMIO는 최소항을 처리대상으로 하므로 입력이 cube로 들어올때 이것을 최소항으로 변화시키기 위한 작업이 필요하다. 이 작업을 수행하는 프로시듀어가 최소항_생성() 프로시듀어이며, 준비_프로시듀어 에서는 본 알고리즘을 수행하기 위한 준비 작업을 행하는 프로시듀어로서 인접도 계산, 인접 집합 생성, 빈도수 계산 등을 행한다.

Don't care 항은 본 알고리즘의 처음부분에서 don't care항이 갖는 출력부분을 cover된 출력으로 간주함으로 don't care항을 쉽게 고려할수 있다. 이것은 don't care항이 prime implicant 항을 크게 생성 시키는데만 유효하게 쓰이고 cover할 필요가 없기 때문이다.

(6) 실험 결과

본 알고리즘의 유효성을 보이기 위해 Arevalo 의 알고리즘[5], Biswas의 MOM[7] 과의 비교 결과를 그림 1. 에 제시한다.

생성된 prime implicant들의 수는 거의 차이가 없으나, 그림에서 보인 것과 같이 다른 알고리즘들에 비해 본논문에서 제안된 알고리즘의 실행 시간이 매우 빠른 것을 알 수 있다.

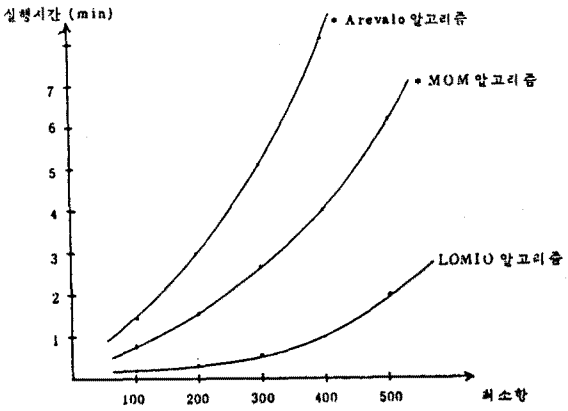


그림 1. 최소항과 실행 시간과의 관계
Fig 1. The relation between the number of minterm and run-time

*논문에 나온 알고리즘에따라 자체적으로 실현한 것임.

또, 그림 2.는 주어진 함수를 MINI[1]와 ESPRESSOII [2]와 본 알고리즘으로 실행한 결과를 비교한 것이다. Cost를 fan-in과 fan-out의 합이라고 하였을 때 MINI는 8 개의 prime implicant 에 cost가 37 이고 ESPRESSOII는 8 개의 prime implicant에 cost가 29 인 반면에, 본 알고리즘은 같은 8 개의 prime implicant에 cost 가 27임을 알 수 있다. 따라서, random logic 의 실현에도 유효함을 알 수 있다.

3. 결 론

본 논문은 휴리스틱 방법에의해 결정된 순서 집합 개념을 이용한 다출력 논리 함수 최소화 알고리즘을 제안 하였다. 제안된 알고리즘은 논리 함수 최소화의 최종적인 prime implicant 들만을 생성하는 방법으로, 기준 최소항을 선택하고 이 최소항의 순서 집합을 생성하여 확장 프로시듀어를 통해 빠른 시간내에 논리함수 최소화를 행한다. 또한 다른 알고리즘과 실행 시간을 비교함으로서 본 알고리즘의 유효성을 보였으며 fan-in과 fan-out이 다른 알고리즘에 비해 부분적으로 더 줄어들므로 random logic 에서의 실현에도 유용함을 보였다. 본 알고리즘은 SSM-16의 UNIX V7 상에서 C 언어로 실현 하였다.

[참고 문헌]

[1] S.J.Hong, et. al., "MINI : A Heuristc Approach for Logic Minimization," IBM Journal of Res. and Dev. Vol. 18, pp.443-459, September 1974.

[2] R.K. Brayton, et. al., "ESPRESSOII:A New Logic Minimizer for PLA," in IEEE Proc. Custom Integrated Circuits Conf., p.370, May 1984.

[3] M.R. Dagenais, et. al., " McBOOLE : A New Procedure for Exact Logic Minimization," IEEE Trans. on Comput. Aided Design, Vol.CAD-5,No.1, January 1986.

[4] V.T. Rhyne, et. al., " A New Technique for the Fast Minimization of Switching Functions," IEEE Trans. on Computers, Vol.C-26, pp.757-764, August 1977.

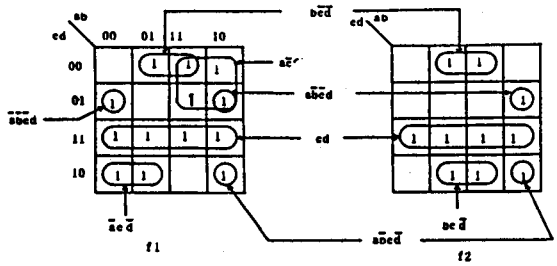
[5] Z. Arevalo, et. al., " A Method to Simplify a Boolean Function into a Near Minimal Sum-of-Products for Programmable Logic Arrays," IEEE Trans. on Computers, Vol. C-27, pp. 1028-1039, November 1978.

[6] N.N.Biswas,"Computer Aided Minimization Precedure for Boolean Functions," Proc. 21st Design Automation Conference, pp.669-702, June 1984.

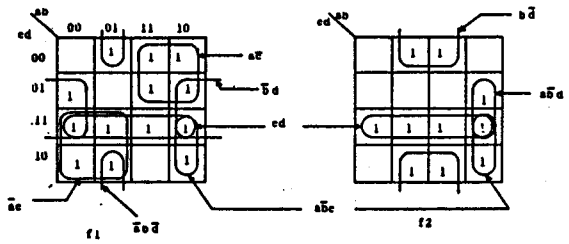
[7] N.N. Biswas, " Multiple Output Minimization," Proc. 22nd Design Automation Conference, pp. 674-680, June 1985.

[8] V. Bubenik, " Weighting Method for the Determination of the Irredundant Set of Prime Implicants," IEEE Trans. Computers, Vol. C-21, pp. 1449-1451, December 1972.

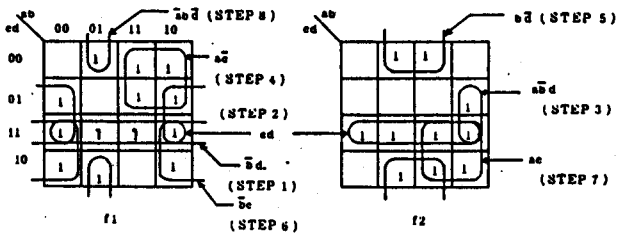
[9] 백영석 등, "순서집합 개념을 이용한 새로운 논리 최소화 알고리즘,"대한 전자 공학회 CAD, 반도체, 재료 및 부품 연구회 합동 학술발표 대회 논문집, pp.24-27, 5, 1986.



a) MINI의 결과



b) ESPRESSO II의 결과



c) LOMIO의 결과

$f1 = \Sigma\{1,2,3,4,6,7,8,9,10,11,12,13,15\}$

$f2 = \Sigma\{3,4,6,7,9,10,11,12,14,15\}$

그림 2. 다른 알고리즘과의 비교 예.

Fig 2. The example of the comparison with other algorithms.