OPKFDD 최소화를 위한 노드의 확장형 결정

정 미 경[†] · 황 민[†] · 이 귀 상[††] · 김 영 철[†††]

요 약

OPKFDD (Ordered Pseudo-Kronecker Functional Decision Diagram)는 각 노드에서 다양한 확장방법(decomposition)을 취할 수 있는 Ordered-DD (Decision Diagram)의 한 종류로서 각 노드마다 Shannon, positive Davio, 그리고 negative Davio 확장중의 하나를 사용하도록 하며 다른 종류의 DD와 비교해서 작은 수의 노드로 함수를 표현할 수 있다. 그러나 각 노드마다 각기 다른 확장 방법을 선택할 수 있는 특징 때문에 입력 노드에 대한 확장 방법의 결정에 의해서 OPKFDD의 크기가 좌우되며 최소의 노드 수를 갖는 OPKFDD의 구성은 매우 어려운 문제로 알려져 있다. 본 논문에서는 DD 크기의 기준을 노드 수로 하여 기존의 OBDD (Ordered Binary Decision Diagram) 자료구조에서 각 노드의 확장방법을 결정하는 직관적(heuristic)인 방법을 제시하고, 주어진 입력변수 순서에 대해서 각 노드의 확장 방법을 결정하는 알고리즘을 제안하고 실험 결과를 제시한다.

Decision of the Node Decomposition Type for the Minimization of OPKFDDs

Migyoung Jung[†] · Min Hwang[†] · Gueesang Lee[††] · Youngchul Kim[†††]

ABSTRACT

OPKFDD (Ordered Pseudo-Kronecker Functional Decision Diagram) is one of ordered-DDs (Decision Diagrams) in which each node can take one of three decomposition types : Shannon, positive Davio and negative Davio decompositions. Whereas OBDD (Ordered Binary Decision Diagram) uses only the Shannon decomposition in each node, OPKFDD uses the three decompositions and generates representations of functions with smaller number of nodes than other DDs. However, this leads to the extreme difficulty of getting an optimal solution for the minimization of OPKFDD. Since an appropriate decomposition type has to be chosen for each node, the size of the representation is decided by the selection of the decomposition type. We propose a heuristic method to generate OPKFDD efficiently from the OBDD of the given function and the algorithm of the decision of decomposition type for a given variable ordering. Experimental results demonstrate the performance of the algorithm.

키워드 : DD, OBDD, ETDD, OPKFDD, 확장방법(Decomposition)

1. 서 론

최근에 불리안 함수로 대변되는 논리회로의 표현을 위해 그래프를 이용한 표현 방법인 DD (Decision Diagrams)가 제시되어 논리합성, 테스트 그리고 회로검증(verification)과 같은 여러 가지 설계자동화문제에 활발히 적용되어 왔다. 여러 가지 종류의 DD중에서 실제로 가장 많이 사용하고 있는 것은 OBDD (Ordered Binary Decision Diagrams)이다 [1]. 현재 OBDD의 응용범위는 더욱 넓어져가고 있으며 이는 불리안 함수 외에도 여러 가지 이산함수(discrete function) 즉, 다치함수, 큐브 집합(cube set), 산술함수(arithmetic function)등의 경우에도 적용되며 이와 같은 논리함수의 그래프 표현은 전산, 전자분야뿐 아니라 다른 많은 분야에서도 매우 유용하게 쓰이고 있다. 또한 장기적으로 이러한 그래프 표현은 그 특성인 정규적(canonic) 표현임과 동시에 사용의 편리함 등으로 인하여 더욱 그 사용이 늘어날 전망이다.

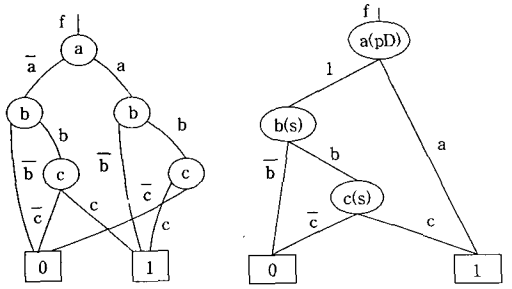
DD의 사용이 활발해짐에 따라 여러 가지 형태의 DD가 제시되었으며 그 중에서도 두드러진 것은 OFDD (Ordered Functional Decision Diagram)이며[2] 이는 같은 변수를 나타내는, 즉 같은 level의 노드에서 positive Davio와 negaive Davio중의 하나를 사용하도록 한다. OFDD로부터 OKFDD (Ordered Kronecker Functional Decision Diagram) [3-6]가 제안되었으며 이는 같은 레벨의 노드에서 세 가지 확장(decomposition)인 Shannon, positive Davio, negaive Davio중의 하나를 사용하도록 한다. 또한 이를 더욱 발전시킨 형태인 OPKFDD [7]는 각 노드마다 위의 세 확장 중에서 임의의

※ 이 논문은 2000년도 전남대학교 학술연구비 지원에 의하여 연구되었음.
† 정 회 원 : 전남대학교 대학원 전산통계학과
†† 종신회원 : 전남대학교 정보통신연구소 전산학과 교수
††† 정 회 원 : 전남대학교 전자공학과 교수
논문접수 : 2002년 3월 5일, 심사완료 : 2002년 6월 26일

하나를 선택하도록 한 것이다. 따라서 DD는 각 노드의 확장 방법을 어떤 타입을 선택하느냐에 따라서 DD의 종류가 결정된다.

각 노드의 확장 방법은 세 가지 확장인 Shannon, positive Davio, 그리고 negaive Davio 확장 중 한가지를 선택한다. 각 노드의 선택의 폭이 넓어짐에 따라 표현공간 최소화의 가능성은 더욱 높아지나[6] OPKFDD의 경우 n 개의 입력변수에 대해서 3^{2^n-1} 개의 서로 다른 확장을 할 수 있으므로 이를 최소화하는 연구는 상대적으로 더 복잡성을 띄게 된다[7].

예를 들어 만약 주어진 함수 f가 $ab' + ac' + a'bc$라 할 때 (그림 1)과 같이 같은 함수를 표현하는데 있어서 DD의 종류에 따라서 필요한 노드의 수가 다르다. (그림 1)의 (a)와 (b)를 비교해 보면 같은 함수를 각 노드의 확장 방법을 어떻게 선택하느냐에 따라 DD의 크기가 달라진다. (a)는 모든 노드를 shannon으로 확장하는 OBDD형태로 표현한 경우이고 (b)는 각 노드마다 각기 다른 확장 방법을 사용하여 OPKFDD를 생성한 경우이다. OPKFDD에 의한 불리안 함수 표현은 각 노드마다 각기 다른 확장 방법을 선택할 수 있으므로 BDD보다 더 적은 노드 수로 함수를 표현할 수 있다. 따라서 같은 함수를 표현하지만 DD 종류에 따라서 즉, 각 노드의 확장형에 따라서 노드 수가 다름을 알 수 있다.



OBDD　　OPKFDD

(그림 1) 각 노드의 확장방법에 따른 DD의 크기

본 논문에서는 주어진 불리안 함수를 표현하기 위해서 OPKFDD를 사용한다. OPKFDD를 최소화하기 위해서는 위와 마찬가지로 두 가지 관점에서의 연구가 필요하다. 첫째, 입력 변수가 n개 일 때, $n!$ 가지의 입력 변수 순서 중에서 DD의 크기를 작게 하는 입력변수 순서를 결정하는 것이다. 둘째, OBDD, OFDD, 그리고 OKFDD와는 다르게 OPKFDD에 대한 연구는 각 노드에서의 확장방법을 선택하는 문제가 새로이 고려되어야 한다. 이러한 특성 때문에 OPKFDD의 최소화 문제는 매우 어려운 문제로 알려져 있으며 아직 본격적인 연구가 이루어지지 못하고 있다. 현재까지는 OKFDD와 Pseudo Kronecker tree에 관한 연구가 주로 진행되어 왔으나, 최근에 들어 OPKFDD에 대한 관심이 증가하고 있다[4-6, 9-13]. 본 논문에서는 특히 두 번째의 문제, 즉 노드의 확장방법을 결정하는 문제에 초점을 맞추어 효율적인 방법을 제시하고자 한다. 입력변수순서를 결정하는 방법은 또 다른 연구의 주제이므로 본 논문에서는 다루지 않는다.

먼저 OPKFDD를 추출하기 위한 ETDD를 구성하기 위하여 OBDD의 각 노드의 두 간선을 exclusive-OR하여 $f_{v=0} \oplus f_{v=1}$을 가리키는 XOR-간선을 갖도록 수정한다. 이러한 형태의 DD를 ETDD(EXOR Ternary Decision Diagram)라 한다. 즉, OBDD로부터 각 노드마다 간선을 추가하여 ETDD를 만들어 내고, 최소한의 노드를 갖는 OPKFDD를 유도하는 방법을 제안한다. 각 노드의 확장방법은 3개의 간선 중에서 2개를 필요로 한다. 따라서 ETDD의 THEN-간선, ELSE-간선 그리고 XOR-간선 중에서 2개의 간선을 선택하여 각 노드에서의 확장형을 결정하여야 하며 OPKFDD의 최소화문제는 어떻게 이와 같이 3개의 간선에서 2개를 선택하면 최소의 노드로 DD를 구성할 수 있는가의 문제인 것이다. 어떤 간선들을 선택할 것인지를 결정하기 위해 ETDD의 각 노드들의 비용함수를 계산한다. 본 논문에서는 이 비용함수로 ETDD의 3가지 간선 중에서 가장 작은 노드를 갖는 한 간선을 선택하기 위해서 각 간선에 연결된 부트리(sub-tree)에 포함된 노드 수를 계산하는 함수이다. 그러나, 이 방법은 서로 다른 노드들에 의하여 공유된 노드들의 경우를 고려하지 못하므로 본 논문에서는 비용함수의 효율성을 높이고 중복성을 고려하기 위해 간단한 휴리스틱 방법을 제시한다. 마지막으로 이 논문에서 제안한 방법으로 구성한 OPKFDD와 OBDD를 비교한 실험결과를 제시한다.

본 논문의 구성은 다음과 같다. 제 2장에서는 OPKFDD의 정의를 제시하고, 제 3장에서는 기존의 OBDD 자료에서 OPKFDD를 유도해내는 방법과 주어진 입력변수 순서에 대해서 각 노드의 확장 방법을 결정하는 알고리즘을 제안한다. 그리고 제 4장에서는 제 3장에서 제안한 알고리즘의 실험결과를 제시하며, 마지막으로 결론과 향후 연구 방향을 언급한다.

2. 관련 연구

2.1 OPKFDDs (Ordered Pseudo-Kronecker Functional Decision Diagrams)

DD와 OBDD, ETDD 그리고 OPKFDD의 정의는 다음과 같다[7, 14]. OPKFDD는 확장된 DD의 개념을 갖는다.

[정의 1] Decision Diagram

입력변수 $X_n := \{x_1, x_2, \cdots, x_n\}$에 대한 DD는 루트가 있는 방향성 비 순환 그래프(directed acyclic graph)이다. 이 그래프는 2가지 타입의 노드를 갖는데 비-단말(non-terminal)과 단말(terminal) 노드로 구성된다. 비-단말 노드 v는 입력 변수 x_i로 분류되고, $low(v)$와 $high(v)$ 두 개의 자 노드를 갖는다. 그리고 단말 노드 v는 0 또는 1의 값을 갖고 자 노드가 없다.

[정의 2] Ordered Decision Diagram

DD의 각 경로에서 각 입력 변수가 최소한 한번 있고 각

경로에서 입력변수의 순서가 동일하다면 DD를 ordered하다고 한다.

DD는 아래와 같은 확장 타입을 사용하여 불리안 함수를 표현할 수 있다.

$$\begin{aligned} f &= \overline{x_i} f_i^0 \oplus x_i f_i^1 && \textit{shannon } (S) \\ f &= f_i^0 \oplus x_i f_i^2 && \textit{positive Davio } (pD) \\ f &= f_i^1 \oplus \overline{x_i} f_i^2 && \textit{negative Davio } (nD) \end{aligned}$$

여기서 f_i^0와 f_i^1는 각각 $x_i=0$와 $x_i=1$에 대한 f의 *cofactor*이고, $f_i^2 = f_i^0 \oplus f_i^1$이다. 그리고 $\oplus$은 exclusive OR 연산자이다.

예를 들면 임의의 함수 $f=a+b$를 고려해 보자. 변수 a에 대하여 $f_i^0=b$, $f_i^1=1$이므로 $f_i^2=f_i^0 \oplus f_i^1 = b \oplus 1 = \overline{b}$이다. 그러므로, 3가지 확장방법은 각각 다음과 같은 형태로 나타낼 수 있다.

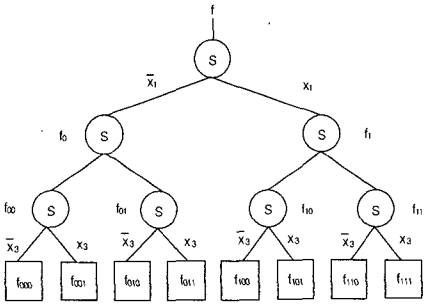
$$\begin{aligned} &\textit{Shannon expansion}: f = \overline{a} \cdot b \oplus a \cdot 1 \\ &\textit{positive Davio expansion}: f = b \oplus a \cdot \overline{b} \\ &\textit{negative Daviuo expansion}: f = 1 \oplus \overline{a} \cdot \overline{b} \end{aligned}$$

즉, 임의의 함수는 위의 3가지 확장방법 중 하나를 사용하여 표현할 수 있으므로 임의의 함수표현에 필요한 것은 3개의 cofactor f_i^0, f_i^1, f_i^2중에서 2개로 충분함을 알 수 있다.

[정의 3] OBDD

OBDD는 입력변수 x_n에 대한 ordered DDs이고 각 노드의 확장 타입은 Shannon으로 구성된다.

예를 들면 OBDD는 (그림 2)와 같이 확장 방법이 S로만 확장되기 때문에 확장 방법은 고려 대상이 되지 않으며 입력변수가 n개일 때 $n!$개의 입력변수의 순서가 만들어진다.



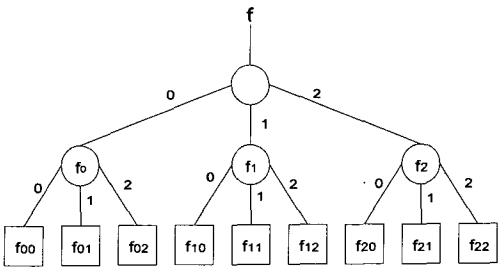
(그림 2) n변수 함수에 대한 OBDDs의 확장방법 선택

[정의 4] ETDD

ETDD는 다음과 같이 정의된 방향성 비순환 그래프 G로서 재귀적으로 정의된다.

1. 만약 G가 0(혹은 1)값을 갖는 단말노드이면, G는 상수 0(혹은 1) 함수를 나타내는 ETDD이다.
2. 만약 G가 함수 f의 입력변수 v에 대한 근 노드를 갖고 있다면, 그 노드는 부함 수 $f_0 = f(v=0)$, $f_1 = f(v=1)$ and $f_2 = f(v=0) \oplus f(v=1)$을 나타내는 3개의 간선을 갖는다.
3. 함수에서 각 노드의 확장 변수는 입력변수 순서로 확장된다.

ETDD는 OBDD와 유사하지만 (그림 3)과 같이 ETDD는 각 노드마다 2개의 간선을 exclusive-OR한 간선을 추가로 가지고 있다. 각 노드마다 3개의 outgoing 간선을 갖는 함수를 TDDs (Ternary Decision Diagrams) [14]라고 한다.

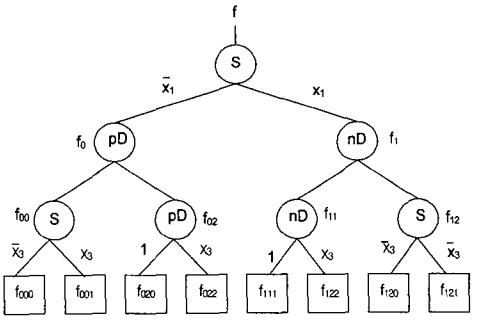


(그림 3) 2변수 함수에 대한 ETDD

[정의 5] OPKFDD

OPKFDD는 입력변수 x_n에 대한 ordered DD이고 각 노드마다 Shannon, positive Davio, 그리고 negative Davio 확장 중에서 1가지를 선택하여 각기 다른 확장 타입으로 구성된다.

예를 들면 입력변수가 n개인 경우 함수를 (그림 4)와 같이 OPKFDD로 구성할 수 있다. OPKFDDs에서 각 노드는 서로 다른 확장 방법을 선택할 수 있으므로 OPKFDDs는 $3^{2^{n-1}}$개의 서로 다른 확장 방법이 있을 수 있다. 그러므로 OPKFDDs에서는 임의의 노드에 대하여 3가지 확장방법 중에서 하나를 사용하므로 3개의 서브노드(또는 간선)중에서 2개만 사용하여도 함수를 구성할 수 있다. 본 논문의 초점은 바로 전체의 노드수가 최소화되도록 3개의 서브노드(또는 간선)에서 2개만을 선택하는 방법을 찾는 것이다.



(그림 4) n변수 함수에 대한 OPKFDD의 확장방법 선택

[정의 6] OPKFDD의 노드

OPKFDD에서 어떤 노드가 Shannon expansion으로 확장

되면 Shannon 노드라고 하고 Davio expansion 중에서 positive Davio expansion으로 확장된다면 그 노드를 positive Davio 노드라고 하며, negative Davio expansion으로 확장된다면 그 노드를 negative Davio 노드라고 한다.

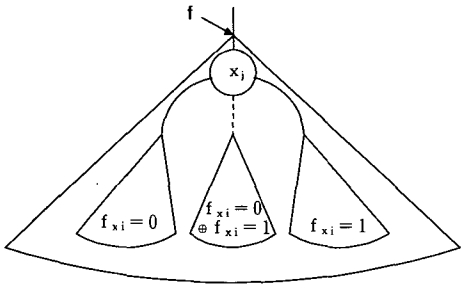
[정의 7] DD 크기

어떤 불리안 함수 F의 DD 크기를 $|F|$라 할 때 이 크기는 F의 단말노드를 제외한 노드 수이다.

3. 최적의 OPKFDD 구성을 위한 각 노드의 확장형 결정 방법

이 장에서는 주어진 함수의 OBDD로부터 OPKFDD를 생성하는 방법을 설명한다. OBDD의 각 노드는 Shannon expansion에 의하여 확장된 2개의 부분함수를 가리키는 THEN-간선과 ELSE-간선을 가진다. 이 때, 노드함수를 f라 하고 노드에 해당하는 변수를 v라 하면 THEN-간선은 $f_{v=1}$을 가리키고 ELSE-간선은 $f_{v=0}$을 가리킨다. 먼저 ETDD를 생성하기 위하여 (그림 5)와 같이 주어진 함수의 OBDD를 구성하는 모든 노드에 XOR-간선을 추가한다. 즉, 주어진 함수를 표현하고 있는 OBDD의 각 노드의 두 간선을 exclusive-OR하여 $f_{v=0} \oplus f_{v=1}$을 가리키는 XOR-간선을 갖도록 수정한다.

일단 ETDD가 생성되면 이제는 각 노드에 연결된 간선 중에서 2개만 선택함으로써 OPKFDD를 생성해 나가는 것이다. 본 논문에서는 이 수정된 OBDD, 즉 ETDD의 THEN-간선, ELSE-간선 그리고 XOR-간선 중에서 2개의 간선을 선택하되 가능하면 최소의 노드가 되도록 각 노드에서의 확장형을 결정하여 OPKFDD를 구성한다.



(그림 5) 각 노드의 XOR-간선 생성(ETDD의 생성)

ETDD로부터 최소의 노드 수를 갖는 OPKFDD를 구성하기 의해서 각 노드의 확장형을 결정하기 위해서 다음과 같은 비용함수를 사용한다. 즉, (그림 6)과 같이 ETDD의 3가지 간선 중에서 가장 작은 노드를 갖는 한 간선을 선택하기 위해서 각 간선에 연결된 노드 수를 계산하는 함수이다. 그러나, 이 비용함수는 서로 중복된 노드들을 고려하지 않으므로 이 비용함수만을 사용하는 경우에 매우 비효율적인 결과를 갖게 된다. 그러므로, 이러한 비효율성을 제거하기 위하여 순차적으로 노드들을 선택하여 나가되, 이미 선택된 노드들은 그 비용함수를 0으로 하는 방법을 선택하면 비용함수의 결과는 달라지게 된다. 수정된 비용함수는 (그림 7)과 같이 노드에 '*selected*'라고 표기된 경우, 이미 선택된 노드이므로 더 이상 추가할 필요가 없으므로 그 비용을 0으로 계산하는 것이다. 이 수정된 비용함수는 처음의 비용함수에 비하여 좀 더 정확한 비용을 계산해준다. 이러한 방식으로 노드를 순차적으로 선택하되 선택된 노드들에 대하여 비용함수가 항상 변화되게 하는 것이 본 논문에서 제시하는 방법의 주된 아이디어이다.

```
/* OPKFDD의 각 간선에 연결된 노드 수 계산하는 함수 */
OPKFDD_GetCost(m) /* m : an ETDD node */
{
 If( m is a terminal node) return 1 ;
 l = OPKFDD_GetCost(else_node(m)) ; /* ELSE-간선의 cost 계산 */
 r = OPKFDD_GetCost(then_node(m)) ; /* THEN-간선의 cost 계산 */
 x = OPKFDD_GetCost(xor_node(m)) ; /* XOR-간선의 cost 계산 */
 return l + r + x - max(l, r, x) ;
}
```

(그림 6) 노드 수를 계산하는 비용함수

```
/* OPKFDD의 각 간선에 연결된 노드 수 계산하는 함수 */
OPKFDD_GetCost2(m) /* m : an ETDD node */
{
 If(m is marked as 'selected') return 0 ;
 If(m is a terminal node) return 1 ;
 l = OPKFDD_GetCost(else_node(m)) ; /* ELSE-간선의 cost 계산 */
 r = OPKFDD_GetCost(then_node(m)) ; /* THEN-간선의 cost 계산 */
 x = OPKFDD_GetCost(xor_node(m)) ; /* XOR-간선의 cost 계산 */
 return l + r + x - max(l, r, x) ;
}
```

(그림 7) 노드 수를 계산하는 수정된 비용함수

위와 같이 두 종류의 비용 함수를 이용하여 각 노드의 확장형을 결정하는 방법은 (그림 8)에 나타난 알고리즘을 이용한다. 이 방법은 다음과 같이 3단계로 구성된다. 첫 단계는 (그림 6)의 비용함수를 이용하여 ETDD의 근(root) 노드에 연결된 각 간선에 연결된 노드 수를 계산하여 3 간선 중 가장 작은 노드 수를 갖는 한 간선을 선택하고 그 간선의 부트리에 포함된 서브 노드들에 대해서만 노드들을 선택하여 선택된 노드들에 대해서는 '*selected*'라고 표기한다. 서브노드들에 대해서는 비용함수를 계산하여 그 비용이 작은 2개의 간선을 선택하는 재귀적인 방법을 취한다. 이렇게 노드들은 선택하여 '*selected*'라고 표기하는 것은 노드선택이 최종확정 되는 것을 말한다. 두 번째 단계에서는 1단계에서 선택되지 않은 근노드의 나머지 2개의 간선에 연결된 노드 수를 (그림 7)의 수정된 비용함수를 이용하여 계산하

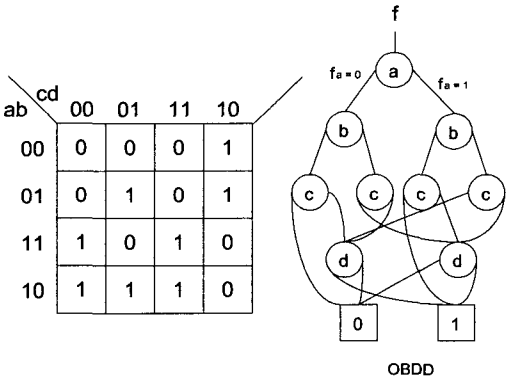
고 둘 중에서 작은 노드 수를 갖는 간선을 선택하여 1단계와 마찬가지로 선택된 간선의 서브 노드들을 선택한다.

마지막 단계에서는 중복된 노드 선택을 제거한다. 위의 노드 선택과정들을 거치는 경우에도 많은 노드들이 그 서브 노드들 중에서 2개만을 선택하는 것이 아니라 3개를 모두 선택하는 경우가 있게 된다. 이는 하나의 노드를 여러 다른 부분함수들이 사용할 수 있기 때문이다. 따라서 이러한 경우에는 3 간선을 모두 다 선택한 노드에 대해서 각 간선의 참조수(reference number)를 계산하여 참조수가 1인 노드를 제거하는 방법을 취한다. 참조수가 1인 경우는 공유가 없는 부분이므로 나머지 2개의 서브노드(2개의 간선)만 사용하여도 하나의 확장방법이 선택되므로 함수를 구성할 수 있게 된다.

```
/* Decomposition 방법 선택 */
Decomposition_Type_Choice(m) /* m : an ETDD node */
{
  If( m is a terminal node) return ;
  For each subnode n of m, compute cost by OPKFDD_GetCost(n) ;

  Let n = node with minimum cost ;
  Fix_subnodes(n) ; /* 노드 n 이하의 노드를 선택하여 이를 mark 함 */
  For each subnode n of m, recompute cost by OPKFDD_GetCost2(n) ;
  Select another node n' with smaller cost ;
  Fix_subnodes(n') ;
  Remove_redundancy(m) ; /* 중복된 노드 선택을 제거함 */
}

Fix_subnodes(n)
{
  Mark n as 'selected' ;
  If (n is a terminal node) return ;
  l = OPKFDD_GetCost2(else_node(n)) ;
  r = OPKFDD_GetCost2(then_node(n)) ;
  x = OPKFDD_GetCost2(xor_node(n)) ;
  Select smaller two nodes n1, n2
     from {else_node (n), then_node (n), xor_node (n)} ;
  Fix_subnodes (n1) ;
  Fix_subnodes (n2) ;
}

Remove_redundancy (m)
{
  Find a list l of nodes in the subtree of m,
     where all of its subnodes are marked as 'selected' ;
  For each node n of l {
     Find reference count of the subnodes of n ;
     If there is a subnode with reference_count = 1, unselect the node ;
  }
}
```

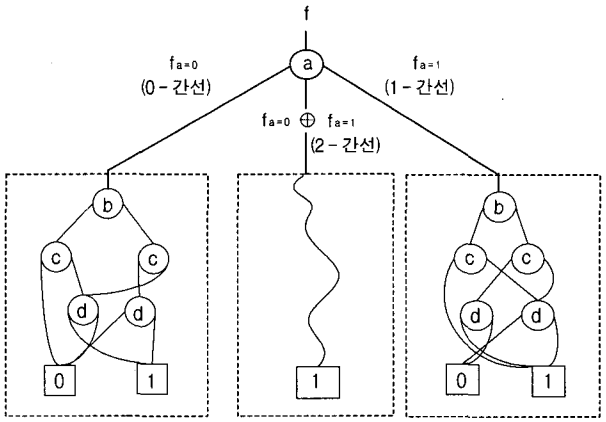
(그림 8) 확장 방법 선택 알고리즘

위의 방법으로 주어진 블리안 함수에 대한 OBDD로부터 각 노드의 확장형을 결정하여 OPKFDD를 생성하는 과정을 다음과 같은 예제를 통해서 설명한다. 예를 들어 함수 f가 $a'cd'+ac'd'+acd+ab'd+a'bc'd$이고 입력변수 순서가 a, b, c, d 라 할 때 OBDD는 (그림 9)와 같다.



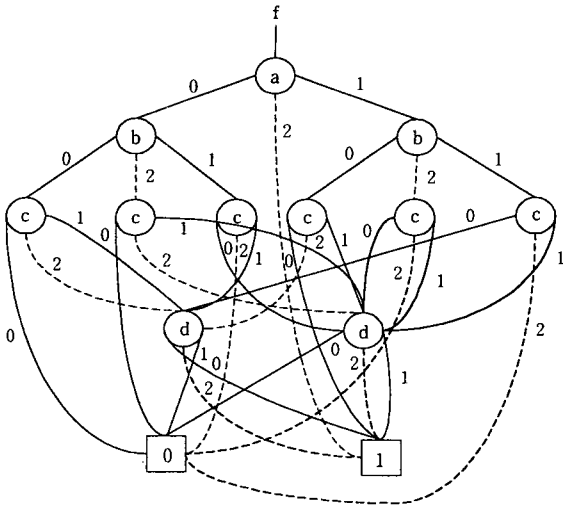
(그림 9) 함수 f의 OBDD

함수 f의 OBDD의 각 노드는 Shannon expansion에 의하여 확장된 2개의 부분함수를 가리키는 2개의 간선을 가진다. 이 때, 노드함수를 f라하고 노드에 해당하는 변수를 a라 하면 THEN-간선은 $f_{a=1}$을 가리키고 ELSE-간선은 $f_{a=0}$을 가리킨다. OPKFDD는 주어진 함수의 BDD를 구성하는 모든 노드에 XOR-간선을 추가함으로써 쉽게 구성해 낼 수 있다. 즉, (그림 10)과 같이 OPKFDD를 유도하기 위해 주어진 함수를 표현하고 있는 BDD의 각 노드의 두 간선을 XOR하여 $f_{a=0} \oplus f_{a=1}$을 가리키는 XOR-간선을 갖도록 수정한다.



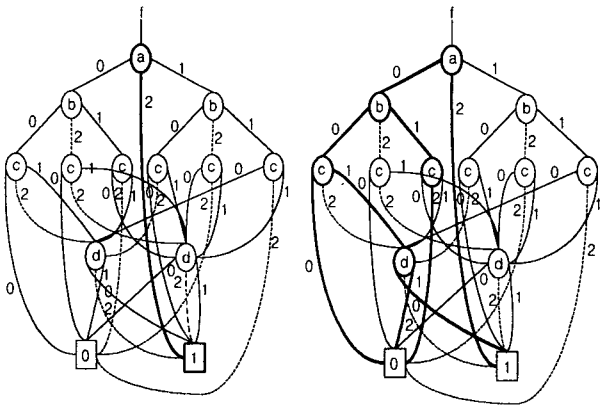
(그림 10) 노드 a에 대한 XOR-간선 생성

위와 같은 과정을 나머지 입력변수 순서 b, c, 그리고 d에 대한 노드에 대해서도 XOR-간선을 생성하여 (그림 11)과 같이 수정된 OBDD로부터 ETDD를 만든다. 먼저 ETDD의 각 노드에 대한 3간선에 연결된 노드 수를 (그림 5)에서 주어진 비용함수를 이용하여 구하고 3가지 간선 중에서 가장 작은 노드 수를 갖는 간선을 선택한다. 예를 들어 (그림 11)의 경우 입력변수 a에 대한 3가지 간선 중에서 2-간선(XOR-간선)에 연결된 노드 수가 가장 작으므로 (그림 12)의 (a)와 같이 1단계에서 2-간선이 선택된다. 이 경우에는 2-간선이 단말노드로 끝남으로써 더 이상 서브 노드를 확장하지 않



(그림 11) 수정된 OBDD로부터 ETDD구성

지만 2-간선에 서브 노드가 있다면 서브 노드의 3가지 간선 중에서 작은 노드 수를 갖는 2개의 간선을 선택하여 확장형을 결정한다. 2단계에서는 (그림 12)의 (b)와 같이 (그림 7)의 수정된 비용함수를 이용하여 선택되지 않은 1-간선(THEN-간선)과 0-간선(ELSE-간선)에 연결된 노드 수를 결정하여 노드 수가 작은 간선을 선택하는데 이 경우에는 0-간선이 선택된다. 그리고 1단계와 마찬가지로 선택된 간선의 서브 노드에 대한 3가지 간선 중에서 작은 노드 수를 갖는 2개의 간선을 선택하여 확장형을 결정한다.

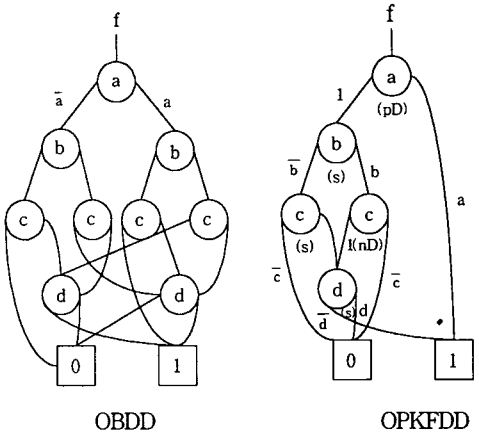


(a) 1단계 : 노드 a에서 2-간선 선택

(b) 2단계 : 노드 a에서 0-간선 선택 및 0-간선의 서브노드에 대한 간선선택

(그림 12) 각 노드의 확장형 선택 과정

위의 결과로 각 노드에 연결된 3가지 간선 중에서 최소의 노드 수를 갖는 2개를 선택하여 확장형을 결정하여 (그림 13)와 같이 OPKFDD를 생성한다. (그림 13)의 OPKFDD는 노드 a는 positive Davio로 노드 b는 Shannon으로 확장된다. 노드 b의 1-간선, 즉 노드 c는 Shannon 그리고 노드 b의 b-간선, 즉 노드 c는 negative Davio로 확장한다. 또한 노드 d는 Shannon으로 확장하여 OPKDFF를 구성한다. OBDD와 비교해서 주어진 함수를 DD로 표현하는 경우 OPKFDD가 더 적은 노드 수로 함수를 표현할 수 있다.



OBDD OPKFDD

(그림 13) 함수 f의 OPKFDD 생성

4. 구현 및 실험결과

본 논문에서 제시한 방법은 C언어로 구현했고, PC 펜티엄에서 실행했다. 실험 결과는 주어진 벤치마크(benchmark) 회로에 대하여 본 논문에서 제안한 방법과 OBDD를 노드 수로 비교하였다.

〈표 1〉 OBDD와 OPKFDD의 노드 수 비교

Benchmark 회로	OBDD (Initial order)	OBDD (sifting)	OPKFDD (Initial order)	OPKFDD (sifting)
	# node	# node	# node	# node
5xp 1	73	41	66	35
card 4	73	27	40	21
clip	225	91	124	73
clog 8	154	199	141	158
cmlp 4	146	153	113	101
cnrm	159	122	146	118
cu	58	42	59	42
f5lm	41	41	37	37
inc	76	72	74	67
mlp 3	45	42	33	33
rd 53	16	16	13	13
rd 73	30	30	21	21
sao 2	154	89	106	86
t481	20	20	16	16
vg2	1043	229	596	215
Total	2313	1214	1585	1036

<표 1>의 결과는 각 노드의 확장 방법에 따른 OBDD와 OPKFDD의 노드 수를 비교한 것이다. 단 제시한 결과는 초기 입력변수 순서와 sifting 방법[15]에 따라서 미리 구해

진 입력순서를 사용하여 본 논문에서 제안한 OPKFDD 구성을 위한 노드 확장 방법을 적용한 결과이다.

<표 1>에서 알 수 있는 것은 입력변수 순서와 확장 방법에 따라서 DD의 크기가 다름을 알 수 있다. OPKFDD (sifting)는 OBDD (Initial order)과 OBDD (sifting)와 비교해서 노드 수가 작음을 알 수 있다. 특히 OBDD (Initial order)와 비교해서 55%정도 감소함을 알 수 있다. 즉, 입력변수 순서와 각 노드의 확장 방법을 잘 선택함으로써 DD의 크기를 최소화할 수 있다. 그리고 OBDD (sifting)과 비교해서 15%정도 감소함을 알 수 있다. 즉, 같은 입력변수 순서 하에서 각 노드의 확장 방법을 잘 선택함으로써 DD의 크기를 최소화할 수 있다.

5. 결론 및 연구방향

본 논문에서는 OPKFDD을 이용한 불리안 함수의 최적화 표현을 위해서 각 노드의 확장 형을 선택하는 직관적(heuristic)인 방법을 제안하였다. 각 노드의 확장 방법은 OBDD의 ELSE 간선과 THEN 간선을 exclusive-OR 시킨 XOR 간선을 추가한 하여 ETDD를 만들고 ETDD의 각 간선에 연결되어있는 노드 수를 계산하는 비용함수를 사용하여 최소의 노드 수를 갖는 2개의 간선만을 선택함으로써 결국에는 최소의 크기를 갖는 OPKFDD를 생성한다. 그리고 OBDD와 비교해 볼 때 각 노드의 확장형 선택에 따라서 DD의 크기가 달라지며 개선된 결과를 보인다.

향후 연구 방향으로는 불리안 함수의 최적화 표현을 위해서 노드 수를 감소시킬 수 있도록 입력 변수의 순서와 각 노드의 확장 방법을 찾는 알고리즘에 대한 연구와 본 논문에서 제안한 방법들이 최적의 해에 얼마나 근사한지 평가, 보완하는 연구가 이루어져야한다.

참 고 문 헌

[1] R. E. Bryant, "Graph-based algorithms for Boolean function manipulation," *IEEE Trans. on Computer*, pp.677-691, 1986.

[2] B. Becker, R. Drechsler, and M. Theobald. "On the Implementation of a Package for Efficient Representation and Manipulation of Functional Decision Diagrams," *IFIP Workshop on the Applications on the Reed Muller Expansions*, pp.162-169, 1993.

[3] B. R. Becker and R. Drechsler, "OKFDD's versus OBDD's and OFDD's," *In Proceedings of ICALP, LNCS 944*, pp. 475-486, 1995.

[4] R. Drechsler, B. Becker, A. Sarabi, M. Theobald and M. A. Perkowski, "Efficient Representation and Manipulation of Switching functions Based on Ordered Kronecker Functional Decision Diagrams," *In Proceeding of DAC*, pp.415-419, 1994.

[5] R. Drechsler, B. Becker and N. Drechsler, "Minimization of OKFDDs by Genetic Algorithms," *International Symposium on Soft Computing, Reading*, Vol.B pp.528-263, 1996.

[6] R. Drechsler, and B. Becker, "On Variable Ordering and Decomposition Type Choice in OKFDDs," *IEEE Tran. on Computer*, pp.1398-1403, Nov., 1998.

[7] T. Sasao and M. Fujita, Representations of Discrete Functions, *Kluwer Academic Publishers*, 1996.

[8] R. Drechsler, personal communication, 1999.

[9] T. Sasao and J. T. Butler, "A Design Method for Look-up Table Type FPGA by Pseudo-Kronecker Expansion," *In Proceedings International Symposium on Multiple-Valued logic*, pp.97-106, 1994.

[10] M. A. Perkowski, M. Chrzanowska-Jeske, A. Sarabi, and I. Schafer, "Multi-Level Logic Synthesis Based on Kronecker and Boolean Ternary Decision Diagrams for Incompletely Specified Functions," *In VLSI Design*, 3(3), pp.301-313, 1995.

[11] R. Drechsler, "Pseudo Kronecker Expressions for Symmetric Functions," *International Conference on VLSI Design*, pp.511-513, 1997.

[12] P. Lindgren, R. Drechsler, B. Becker, "Improved Minimization Methods of Pseudo Kronecker Expressions for Multiple Output Functions," *ISCAS Proceedings of the IEEE International Symposium*, Vol.6, pp.187-190, 1998.

[13] P. Lindgren, R. Drechsler, B. Becker, "Look-up Table FPGA Synthesis from Minimized Multi-Valued Pseudo Kronecker Expressions," IEEE International Symposium on Multiple-Valued Logic, pp.95-100, 1998.

[14] T. Sasao, AND-EXOR Expression and Their Optimization, *Kluwer Academic Publisher*, 1993.

[15] R. Rudell, "Dynamic Variable Ordering for Ordered Binary Decision Diagrams," *In IEEE International Conference on Computer-Aided Design*, pp.42-47, 1993.

정 미 경

e-mail : mgjung@chonnam.chonnam.ac.kr

1987년 전남대학교 전산통계학과 졸업 (학사)

1989년 전남대학교 대학원 전산통계학과 석사

1995년~현재 전남대학교 대학원 전산통계학과 박사수료

관심분야 : VLSI/CAD, 논리합성, 인공지능

황 민

e-mail : hwang@cs.chonnam.ac.kr

1986년 전남대학교 전산통계학과 (학사)

1988년 중앙대학교 대학원 전자계산학과 석사

2002년 전남대학교 대학원 전산통계학과 박사

관심분야 : 논리합성, 멀티미디어통신

이 귀 상

e-mail : gslee@chonnam.chonnam.ac.kr

1980년 서울대학교 공대 전기공학과 졸업 (학사)

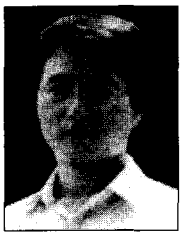
1982년 서울대학교 대학원 전자계산기공학과 석사

1982년 금성통신 연구소 근무

1991년 Pennsyivania 주립대학 박사

1984년~현재 전남대학교 정보통신연구소 전산학과 교수

관심분야 : VLSI/CAD, 멀티미디어 시스템, 테스팅, 논리합성



김 영 철

e-mail : yckim@chonnam.chonnam.ac.kr

1981년 한양대학교 전자공학과 공학사

1987년 University of Detroit, EE, 공학석사

1993년 Michigan State University, EE, 공학박사

1993~현재 전남대학교 전자공학과 부교수

2000~현재 전남대학교 반도체설계교육센터 소장

관심분야 : 초고속통신망, 인터넷 응용, 회로설계, 보안 칩 개발