그래프 기법을 이용한 부울함수의 ALU 기능 해석에 관한 연구

우 광 방, 김 현 기, 박 인 규
(연세 대학교 대학원 전기공학과)

A Study of Function and Analysis of ALU for Graph-based Boolean Functions

Kwang-Bang Woo, Hyun-Ki Kim, In-Gyoo Bahk
(Department of Electrical Eng., Graduate School of Yonsei University)

ABSTRACT

This paper was aimed to, using a new data structure, develop a set of algorithms to execute the output function of Digital System. These functions were represented as directed, acyclic graphs. by applying many restrictions on vertices on graph, the efficient manipulation of boolean function was accomplished.

The results were as follows;

1. A canonical representation of a boolean function was created by the reduction algorithm.
2. The operation of two functions was accomplished using the apply algorithm, according to the binary operator.
3. The arguments having 1 as the value of function were enumerated using the satisfy algorithm.
4. Composing TTL 74181 4-bit ALU and 74182 look-ahead carry generator, the ALU having 4-bit and 16-bit as word size was implemented.

1 서 론

부울함수를 나타내고 처리하기위한 방법들이 오래전부터 개발되어 왔는데, 이러한 방법들로는 진리표, 카르노 맵, 정형의 곱의 합등이 있다.

이들은 n개의 아규먼트를 갖는 함수에 대해서 2^n개 이상의 크기를 갖으므로써, 비실용적이며, 최악의 경우에 있어서는 지수함수적으로 증가하는 계산시간이 필요 하므로, 본 논문에서는 이에대한 해결책의 일환으로써 함수를 방향성 비순환 그래프로써 나타내고, 그래프상에 존재하는 정점들에 많은 제약을 가함으로써 부울함수를 효율적으로 처리할 수있는 여러 알고리즘을 제시한다.

이러한 알고리즘에 의해서 한 함수의 고유한 정형(canonical representation)을 구하며, 여러가지 연산을 수행하며, 이러한 알고리즘을 이용하여 4 비트와 16 비트 워드 크기의 ALU 를 구현한다.

2 그래프 이론

1) 그래프의 정리

함수 그래프는 정점의 두가지 형태를 포함하는 정점집합 V 를 갖는 방향성 비순환 그래프로써 나타내고 비단말정점과 단말정점으로 구성된다.

비단말정점은 속성자(attribute)로써 index(v)$\in$ {1, ..., n} 과 단말정점은 역시 value(v)$\in$ {0, 1}를 갖고, 그래프상에는 비단말정점과 단말정점들은 각각 원과 사각형으로 표시된다.

* V 가 단말정점일 경우
  - value(v) = 1 에대해서 f_v = 1 이며,
  - value(v) = 0 에대해서 f_v = 0 이다.

* V 가 index(v) = i 를 갖는 비단말정점일 경우

∘ $f_v(x_1,\ldots,x_n) = \bar{x}_i f_{low(v)}(x_1,\ldots,x_n) + x_i f_{high(v)}(x_1,\ldots,x_n)$

또한 패스의 진행에 있어서 i 가 0 이면 로우 차일드(low child)로 이어지고, 1이면 하이차일드(high child)로 진행하여 결국 단말정점에서 함수의 값이 결정된다.

2) 함수 오더링의 원리

아래 그림 1 은 예제함수와 오더링의 원리를 보여준다.

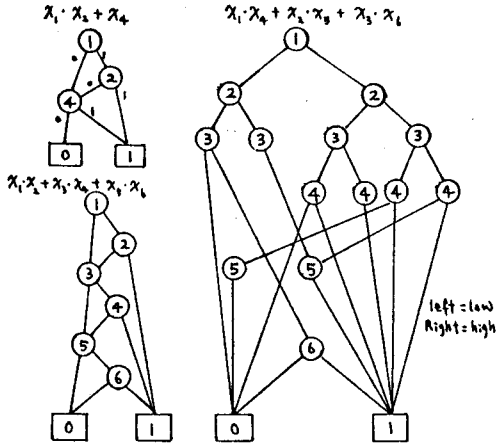

그림 1

한 함수를 나타내는 종속집합의 아규먼트들이 위의 그림에서처럼 간단한 그래프에서도 여러개의 서브 그래프를 갖고 있으며, 아규먼트의 오더링에 따라서 하나는 8 개의 정점 을 갖는 반면, 다른하나는 16 개의 정점을 갖으므로 해서 결국 입력 오더링의 선택이 중요시 된다.

3 함수의 처리 알고리즘

1) 자료구조와 그래프의 탐색

비단말정점과 단말정점들이 정점 V 에 대한 필드 값들은 다음과 같다.

FIELD	TERMINAL	NONTERMINAL
low	null	low(v)
high	null	high(v)
index	n + 1	index(v)
val	value(v)	X

그래프의 탐색에 있어서 정점들에 고유한 테이블을 할당하고, 각정점의 방문의 중복을 피하기 위해서 마크팔드를 이용한다. 이러한 탐색은 여러 알고리즘에서 부품으로써 효율성을 증진시킨다.

2) 축소 알고리즘

단말정점에서부터 루트정점까지 진행하면서 유일한 정수식별자(integer identifier)가 각각의 유일한 서브그래프 루트에 할당되어 진다. 그래프의 테이블 할당 방법은 다음과 같다.

* 두개의 단말정점들은 동일한 밸류속성자(value attribute)를 갖는다면 한개의 동일한 테이블을 갖는다.
* 인덱스 i 보다 큰 모든 단말과 비 단말정점들이 테이블 되어졌다고 가정할 경우에, 인덱스 i 를 갖는 정점들의 테이블링에따라서 한개의 정점 V 는 이미 테이블되어진 어 떤 정점의 테이블과 같다.
  - ∘ id(low(v)) = id(high(v)) 라면, 정점 V 는 리던던트 이므로 id(v) = id(low(v)) 를 수행한다.
  - ∘ id(low(v)) = id(high(u)) 와 id(high(v)) = id(high (u)) 를 갖는 index(v) = i 인 이미 테이블된 정점 u 가 있다면, id(v) = id(u) 를 수행한다.

위와같은 테이블링에 이어 키를 할당하게 되는데, 이는 다과 같다.

* 단말정점의 경우 key = (value) 를 수행하며,
* 비 단말정점의 경우 key = (lowid , highid) 를 수행한다

그래프상의 동일한 인덱스들에 따라서 여러개의 리스트들을 만들고, 각각의 정점들이 속해있는 키에 따라서 정점들이 정렬되어지며 이러한 정점들에 유일한 테이블을 할당하고 각각의 유일한 테이블에 대해서 정점레코드를 선택해서 그정점에 포인터를 저장하게 되는데, 결국 이러한 선택된정점들이 간소화된 그래프를 형성하게 된다. 아래 그림2 에서 처럼 인덱스 3 을 갖는 두개의 정점이 동일한 키이므로, 인덱스 2를 갖는 오른쪽의 정점이 리던던트로 작용해서 결국 간소화된 그래프가 만들어진다.

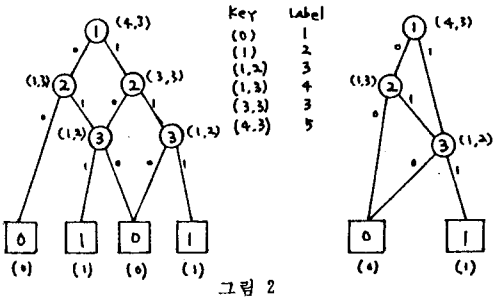

그림 2

3) 연산 알고리즘

이 프로시듀어는 두 함수에 관한 연산을 다음과 같이 정의하여 수행한다.

$[f_1 \langle op \rangle f_2](x_1,..,x_n) = f_1(x_1,..,x_n) \langle op \rangle f_2(x_1,..,x_n)$

이 연산 알고리즘의 기본원리는 다음과 같다.

* V_1 과 V_2가 단말정점이라고 가정하면 결과정점은 value(v_1) <op> value(v_2)의 값을 갖는 단말정점으로 구성된다.
* 적어도 두개의 정점중의 하나가 비단말정점이라고 가정하면 index(v_1) = index(v_2) = i에 대해서 index i를 갖는 정점 u를 만들고 이 알고리즘을 계속적으로 low(v_1)과 low(v_2)에 적용해서 루트가 low(u)인 서브그래프를 산출하며, 역시 high(v_1)과 high(v_2)에 대해서는 high(u)인 서브그래프가 만들어진다.
* index(v_1) = i이고 v_2가 단말정점이거나, index(v_2) > i라고 가정하면 루트 v_2를 갖는 그래프로 표현되는 함수는 x_i와 독립적이다. 즉, $f_2|_{x_i=0} = f_2|_{x_i=1} = f_2$ 이 성립한다.

결국, index i를 갖는 정점 u를 만든다. 그러나 계속적으로 이 알고리즘을 low(v_1)과 v_2에 적용해서 low(u)가 되는 서브그래프를 산출하고, high(v_1)과 v_2에 대해서도 역시 마찬가지로 루트가 high(u)가 되는 서브그래프를 만들게되는데, 두개의 정점의 역할이 반전되는 경우에도 비슷한 상황이 유지된다. 일반적으로 이러한 과정에 의해서 만들어지는 그래프는 간소화되지 않으므로, 정점들을 리턴하기 전에 이 알고리즘에 축소 알고리즘을 적용한다.

위와같은 방법의 실행은 이 알고리즘의 타임 콤플렉시티가 증가하기 때문에 다음과 같은 대안을 이용한다.

* 정점의 중복 방문을 할 필요가 없고, 결과의 정점을 알려주는 (V_1, V_2, U)형태의 엔트리를 포함하는 테이블을 유지한다.
* V_1이 단말정점이고 특정의 연산자에 대해서 value(v_1) 이 제어값일 경우에, 1 이 OR 연산자에 대한 제어값이고, 0 은 AND 연산자에 대한 제어값이 되므로, 이러한 경우에는 이상 계산을 할 필요가 없으며 적절한 값을 갖는 단말정점을 만든다.

아래 그림 3 는 OR 연산을 적용한 예를 보여준다.

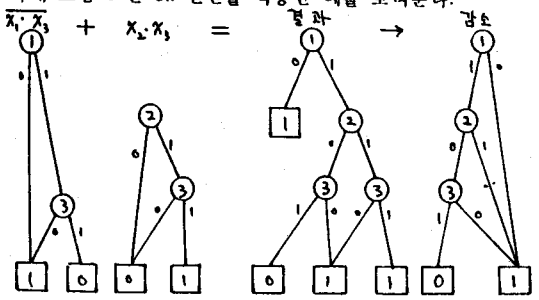

그림 3

4) 만족 알고리즘

이 알고리즘은 하나의 함수가 갖는 만족집합S 에 대해서 아규먼트의 수, 아규먼트의 리스팅을 알기 위한 알고리즘으로서, 한개의 원소를 택하기 위해서 단일 만족 프로시듀어를 이용하는데, 이는 함수의 $S_f = 0$ 이라면 value 를 false 로 리턴하고 $s_f = 1$ 이라면 true로 되돌리며 value i 을 갖는 그래프의 단말정점을 찾기 위해서 깊이우선 탐색(depth-first search) 을 이용해서 S_f의 모든 원소를 열거하므로써, value i을 갖는 원소를 프린트 하게된다.

결국 S_f 의 크기를 계산을 하기위한 value 를 할당한다. 이 그래프에 존재하는 각각의 정점에 대해서 value α_v를 할당한다

* V 가 단말정점이라면 α_v = value(v) 이며,
* V 가 비 단말정점이라면 다음과 같다.

$$\alpha_r = \alpha_{low(v)} \cdot 2^{index(low(v)) - index(v)} + \alpha_{high(v)} \cdot 2^{index(high(v)) - index(v)}$$

$$|S_f| = \alpha_v \cdot 2^{index(v) - 1}$$

4. ALU 에 대한 적용

본 절에서는 TTL 74181 산술논리 연산장치와 74182 look-ahead carry generator 를 결합하여 4 비트와 16 비트 워드의 연산을 수행할 수 있는 ALU를 구성하여 위에서 기술한 리즘을 적용 하였다. 아래의 그림 4 에는 두 칩의 연결도가 타나 있다.

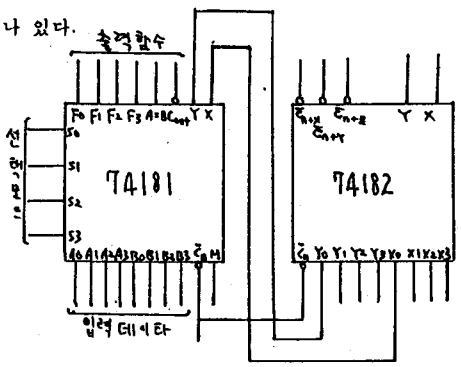

그림 4

시뮬레이션을 위해서 게이트 레벨 표현으로부터 두 칩의 출력함수를 유도하여 이러한 함수들을 합성하여 ALU에 대한 연산을 수행한다.

5. 시뮬레이션과 결과고찰

n 비트의 워드를 가지는 ALU 는 6+2n 개의 입력을 가지고 있다.

* 5 개의 제어입력즉 M, S_0, S_1, S_2, S_3 (이 는 ALU 함수를

선택)

* 캐리 입력 즉 C_{in}
* n 비트의 두 데이타 워드 즉 $A_0, B_0, \ldots, A_{n-1}, B_{n-1}$

또한 n+2개의 출력을 가지고 있다.

* n 개의 함수출력 즉 $F_0, \ldots, F_{n-1}$
* 캐리 출력 즉 C_{out}
* 비교 출력 즉 A = B

4 비트와 16 비트에 대한 파스칼 프로그램의 성취도가 다음과 같이 표 1 에 나타나 있다.

WORD SIZE	GATES	PATTERNS	A=B GRAPH
4	52	1.6×10^4	197
16	227	2.7×10^{11}	737

표 1

위의 테이블에서 보는 바와 같이 게이트의 수는 두칩의 로직 게이트의 수에 사용되는 각 칩의 수의 곱이고, 패턴의 수는 서로 다른 입력의 수이며, A=B 는 간소화 된 그래프의 크기이다. 그림 5 는 입력 오더링의 차이에 따른 변화를 보여준다.

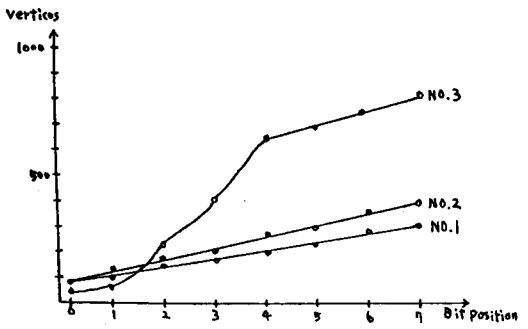

그림 5

* NO.1 = $M, S_0, S_1, S_2, S_3, C_{in}, A_0, B_0, \ldots, A_{n-1}, B_{n-1}$
* NO.2 = $M, S_0, S_1, S_2, S_3, C_{in}, A_{n-1}, B_{n-1}, \ldots, A_0, B_0$
* NO.3 = $C_{in}, A_0, B_0, \ldots, A_{n-1}, B_{n-1}, M, S_0, S_1, S_2, S_3$

NO.1 에서보면 가장 양호한 오더링으로써 먼저 계산되어질 함수를 나타내는 비트를 읽고, 이어 캐리입력을 읽으며 마지막으로 최하위 비트부터 시작하는 두개의 데이타워드의 연속적인 비트를 읽게된다. NO.2 에서는 NO.1 과 비슷하지만 데이타 워드의 오더링에서 보면 출력 워드의 i 번째 비트는 임의의 낮은 순서의 비트보다 입력워드의 i 번째 비트에 더많이 의존함을 알수있다. 마지막으로 NO.3 에서는 출력이 주로 제어입력에 더많이 의존하기 때문에 그러한 결과를 초래하게 되며 데이타의 워드에 대한 덧셈과 논리적 연산들을 나타내는 함수들에 대해서 이러한 오더링이 잘 작용 함으로서 종래의 방법보다 효율성이 있다는 점을 알았다.

6. 결 론

본 논문에서는 자료구조에 의하여 표현되어지는 부울함수에 대한 여러가지 연산을 수행하기 위한 일련의 알고리즘을 제시하였다.

이러한 알고리즘을 실현하기 위해서 그래프 기법을 이용하였으며 프로그램에서는 파스칼 언어의 포인터 기능을 이용하였다. 부울함수의 그래프에 의한 표현법을 취해서 정점테이블에 제약을 가함으로서 간소화된 정형그래프가 산출됨을 보였고 함수를 표현하는데 필요한 스토리지의 양을 감소시킬뿐만 아니라 계산시간의 감소를 간소화 알고리즘에 의해서 구현시킬수 있었다. 함수의 연산에서도 $(v_1, v_2, \langle op \rangle, u)$형태의 엔트리를 포함하는 이차원 배열의 테이블을 이용함으로서 효율적인 연산이 가능하였다. 이와같은 알고리즘을 8비트 이상의 워드를 갖는 산술논리 연산장치에 적용함으로서 부울함수를 표현하고 처리하는 종래의 표현법들이 갖는 한계성을 극복할수 있음을 보였다.

참고문헌

1. A.V. Aho, J.E. Hopcroft, J.D. Ullman, Data Structures and Algorithms, Bell telephone Laboratories, Inc., 1983

2. N.Dale, S.C.Lilly, Pascal plus data structures, heath, 1985

3. C.Y.Lee, "Representation of switching circuits by binary-decision programs," Bell. Syst. tech. J., Vol. 38, pp. 985-999, July 1959

4. S.B.Akers, "Binary decision diagrams," IEEE Trans. Compu., Vol. c-27, pp. 509-516, June 1978

5. F.J.Hill and G.R.Peterson, Introduction to switching theory and Logical Design. New York: Wiley, 1974

6. C.S.Wllace, "A suggestion for a fast multiplier," IEEE Trans. Electron. Comput., Vol. EC-13, pp. 14-17, Jan. 1964

7. A.V.Aho, J.E.Hopcroft, J.D.Ullman, The design and analysis of computer algorithms. Reading, MA: Addison - Wesley, 1974

8. TTL Data Book, Texas Instruments, Dallas, TX. 1976