

Heuristic Optimization of Two-Level Networks

Gerhard Dueck

March 2, 2010

Outline

- 1 Simplification of SOPs with many Inputs
- 2 Merge, Expand, and Delete
- 3 Reduce and Reshape
- 4 Detection of Essential Prime Implicants
- 5 Encoding Methods for Combinational Networks

Simplification of SOPs with many Inputs

- Quine-McCluskey Method
 - needs list of minterms
 - expansion to multiple-output function possible
 - limited number of variables (≤ 20)
- Heuristic Methods
 - minimal results are not guaranteed
 - in practical applications “near minimal” results may be sufficient
 - heuristics are extensively used: PRESTO, MINI, ESPRESSO

Merge

Definition

Let two product terms be

$$c_1 = X_1^{S_1} X_2^{S_2} \dots X_n^{S_n} \text{ and } c_2 = X_1^{T_1} X_2^{T_2} \dots X_n^{T_n}$$

Then, **L-distance** $L\text{-dist}(c_1, c_2)$ is defined as follows:

$$L\text{-dist}(c_1, c_2) = [\text{the number of } i\text{'s such that } (S_i \neq T_i)]$$

That is, the number of different literals in the two products.

Merge

Algorithm (Simplification of SOPs)

- 1 Merge products whose L-distances are 1 (pre-processing).
- 2 Expand each product into a prime implicant.
- 3 Delete redundant products.

Merge is a simplification of SOPs by using the relation:

$$X^A \cdot c \vee X^B \cdot c = X^{A \cup B} \cdot c$$

Expand

Expand is an operation of increasing the volume of cubes without changing the function.

Theorem (Expansion)

Let the given SOP be represented as $F = c \vee G$, where $c = X_i^{S_i} \cdot e$. Let $c(a) = X_i^{\{a\}} \cdot e$, where $a \in P_i - S_i$.

- 1 If $c(a) \leq F$, then $F = X_i^{S_i \cup \{a\}} \cdot e \vee G$.
- 2 If $c(a) \not\leq F$, for all i ($i = 1, \dots, n$) and for all a ($a \in P_i - S_i$), then c is a prime implicant of F . Note that \leq denotes the inclusion relation of functions.

Expand

Example (Expansion)

Consider the SOP

$$F_2 = X_1^{\{1\}} X_2^{\{2\}} \vee X_1^{\{1,2\}} X_2^{\{0\}} \vee X_1^{\{0\}} X_2^{\{2\}} \vee X_1^{\{2\}} X_2^{\{2,3\}}$$

Note that this SOP can be represented as: $F_2 = c_1 \vee G$ where

$$c_1 = X_1^{\{1\}} X_2^{\{2\}}$$

$$G = X_1^{\{1,2\}} X_2^{\{0\}} \vee X_1^{\{0\}} X_2^{\{2\}} \vee X_1^{\{2\}} X_2^{\{2,3\}}$$

Show all expansions of c_1 .

Delete

- The **delete** operation remove redundant products.
- Let $F = c \cdot G$, where c is a product. If $c \leq G$, then c is redundant.
- An SOP of prime implicants with no redundant terms is an **irredundant sum-of-products expression (ISOP)**.

ISOP

Theorem (Generation of an ISOP)

- 1 Let an SOP be represented as $F = c \vee G$. if $c \leq G$, then the product c can be deleted from F , i.e., $F = G$.
- 2 Let the SOP be represented as a sum of prime implicants: $F = c_1 \vee c_2 \vee \dots \vee c_p$. Let G_i be the sum of products in F other than c_i . If $c_i \not\leq G_i$ for all i ($i = 1, \dots, n$), then F is an ISOP.

ISOP

Theorem (Generation of an ISOP)

- 1 Let an SOP be represented as $F = c \vee G$. if $c \leq G$, then the product c can be deleted from F , i.e., $F = G$.
- 2 Let the SOP be represented as a sum of prime implicants: $F = c_1 \vee c_2 \vee \dots \vee c_p$. Let G_i be the sum of products in F other than c_i . If $c_i \not\leq G_i$ for all i ($i = 1, \dots, p$), then F is an ISOP.

Questions

- Is the following true “Any MSOP is also an ISOP?”
- Is the following true “Any ISOP is also a MSOP?”
- Is an ISOP always unique?

Reduce and Reshape

- The **reduce** operation reduces the size of cubes without changing the function.
- The **reshape** operation replaces a pair of adjacent cubes with an other pair of cubes without changing the function.

Reshape

Definition (Reshape)

Let the L-distance of the two products c_1 and c_2 be two. Let

$$c_1 = X_1^{S_1} X_2^{S_2} \dots X_i^{S_i} \dots X_j^{S_j} \dots X_n^{S_n}, \text{ and}$$

$$c_2 = X_1^{T_1} X_2^{T_2} \dots X_i^{T_i} \dots X_j^{T_j} \dots X_n^{T_n},$$

where $S_i \cap T_i = \emptyset$ and $S_j \subseteq T_j$ and $S_k = T_k$ ($k = 1, 2, \dots, n, k \neq i, k \neq j$). Then, the reshape operation replaces the pair (c_1, c_2) with (c_3, c_4) , where

$$c_3 = X_1^{S_1} X_2^{S_2} \dots X_i^{S_i \cup T_i} \dots X_j^{S_j} \dots X_n^{S_n}, \text{ and}$$

$$c_4 = X_1^{T_1} X_2^{T_2} \dots X_i^{T_i} \dots X_j^{T_j - S_j} \dots X_n^{T_n},$$

Reshape

Example

Let $c_1 = X_1^{\{2\}} X_2^{\{0\}}$, $c_2 = X_1^{\{0\}} X_2^{\{0,1\}}$, we have the reshaped products: $c_3 = X_1^{\{0,2\}} X_2^{\{0\}}$ and $c_1 = X_1^{\{0\}} X_2^{\{1\}}$.

Detection of Essential Prime Implicants

Theorem

If a logic function F has a distinguished minterm m , then an MSOP of F has an essential prime implicant that contains m .

Example

	0	1	2
0		1	1
1			
2	1	1	1
3			1

Consensus

Definition (Consensus)

Let two products be

$$c_1 = X_1^{S_1} X_2^{S_2} \dots X_n^{S_n}, \text{ and } c_2 = X_1^{T_1} X_2^{T_2} \dots X_n^{T_n}.$$

The **consensus with respect to variable** x_i of c_1 and c_2 is defined as

$$\text{cons}(i : c_1, c_2) = X_1^{S_1 \cap T_1} X_2^{S_2 \cap T_2} \dots X_i^{S_i \cup T_i} \dots X_n^{S_n \cap T_n}.$$

The **consensus** of c_1 and c_2 is defined as

$$\text{cons}(c_1, c_2) = \bigcup_{i=1}^n \text{cons}(i : c_1, c_2).$$

Detection of Essential Prime Implicants

Algorithm (Detection of essential prime implicants)

- 1 Let F be the set of prime implicants of a function. Let c be the prime implicant which we want to determine is essential or not. Let G be the set of prime implicants other than c .
- 2 Partition the products in G into three classes:
 G_1 : Set of products that have common elements with c .
 G_2 : Set of products that are adjacent to c .
 G_3 : Set of other products.
- 3 $T = (G_1 \# c) \cup G_2$.
 $H = \text{cons}(c, T)$.
- 4 $c \not\in H$ iff c is an essential prime implicant. This decision can be done by the tautology decision of $H(|c)$.

Encoding for inputs and outputs

Inputs and outputs are often represented by multi-valued symbols for which the circuit is two-valued. We have to represent these symbols by two-valued codes.

Example

Given the following problem:

Input	Output
I_0	U_2
I_1	U_1
I_2	U_0
I_3	U_2
I_4	U_2
I_5	U_3

Encoding for inputs and outputs

Example (continued)

Input	Output
000	10
001	01
010	00
011	10
100	10
101	11

Input	Output	
l_0	U_2	→ 00
l_1	U_1	→ 01
l_2	U_0	→ 10
l_3	U_2	→ 00
l_4	U_2	→ 00
l_5	U_3	→ 11

Encoding of outputs

- In the case of a PLA, no line is required for the output $(0, 0, \dots, 0)$
- Assign most frequent output to pattern $(0, 0, \dots, 0)$

Encoding of input

Example (Input encoding)

		Input					
		l_0	l_1	l_2	l_3	l_4	l_5
z_0				1			1
z_1			1				1
		011	001	010	100	101	000

- Minterms should be adjacent
- Hamming distance between l_2 and l_5 should be one
- Hamming distance between l_1 and l_5 should be one