

Multi-Level Logic Synthesis

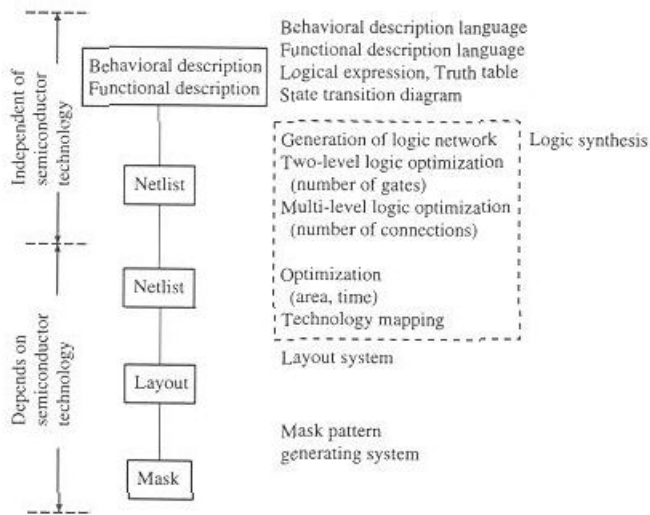
Gerhard Dueck

March 30, 2010

Outline

- 1 Logic Synthesis Systems
- 2 Factoring using Product Terms
- 3 Two-Variable Function Generator
- 4 Algebraic Divisions of Logic Expressions
- 5 Functional Decomposition
- 6 Transformation of Networks
- 7 Simplification using don't Cares

Structure of LSI Design



Factoring using Product Terms

- find common products in SOP terms
- yields multi-level expression

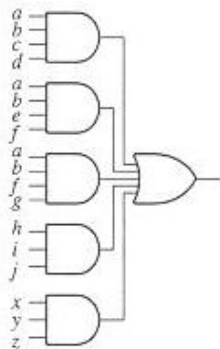
Example

$$F = abcd \vee abef \vee abfg \vee hij \vee xyz$$

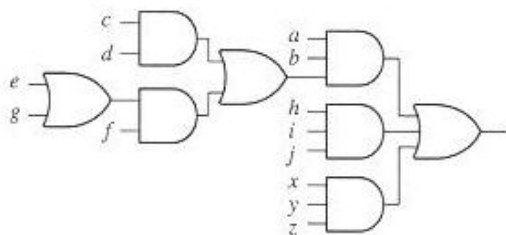
$$F = ab(cd \vee ef \vee fg) \vee hij \vee xyz$$

$$= ab(cd \vee (e \vee g)f) \vee hij \vee xyz$$

Sample Factoring



(a) Two-level logic network.



(b) Factored network.

Factoring using Product Terms

- factoring increases number of gates
- reduces fan-in

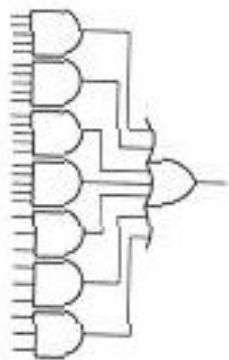
Algorithm (Factoring)

- 1 Enumerate common products
- 2 Select the product term that maximally reduces the number of literals
- 3 Reconstruct the expression, and repeat steps 1 and 2

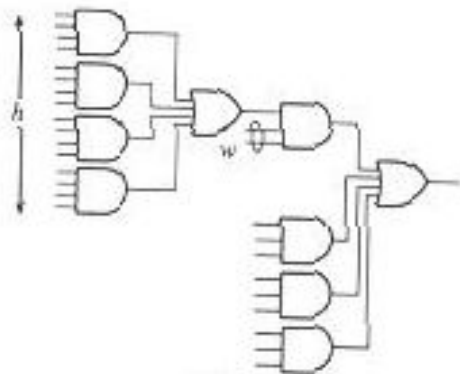
Optimal Factoring

- Assume AND-OR network (see next slide)
- h AND gates contain w common literals
- how many inputs can be saved?
- if gates have input restrictions, gates can be saved
- with **branch-and-bound** optimal factoring can be obtained

Factoring

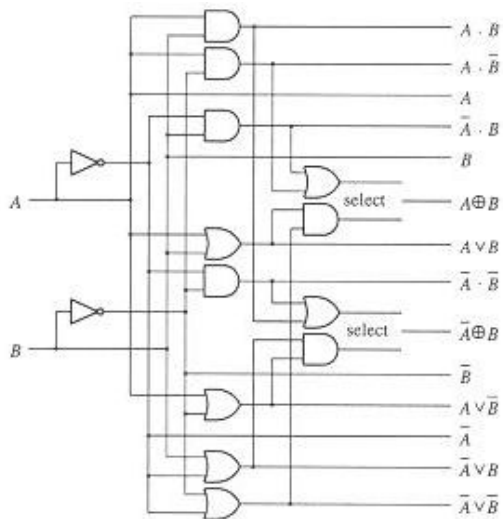


(a)



(b)

Two-Variable Function Generator



Two-Variable Function Generator

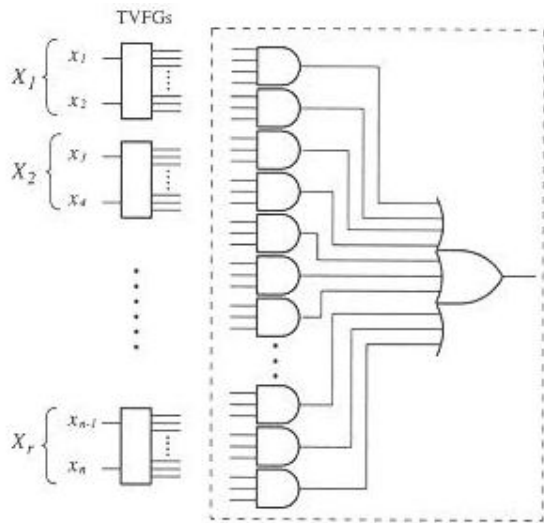
Theorem

An arbitrary logi function $f(x_1, x_2, \dots, x_n)$ ($n = 2r$) is represented by an SOP of 4-valued variables.

$$f(x_1, x_2, \dots, x_n) = \bigvee_{(S_1, S_2, \dots, S_r)} X_1^{S_1} X_2^{S_2} \dots X_r^{S_r}$$

where $X_1 = (x_1, x_2)$, $X_2 = (x_3, x_4)$, ... This function is realized by the figure in the next slide.

AND-OR two-level network with TVFGs



Example

Example

In the function shown in Fig. 11.7, let $X_1 = (x_1, x_2)$ and $X_2 = (x_3, x_4)$. Then, we have the expression:

$$f = X_1^{\{00\}} X_2^{\{00\}} \vee X_1^{\{00\}} X_2^{\{11\}} \vee X_1^{\{01\}} X_2^{\{01\}} \vee X_1^{\{01\}} X_2^{\{10\}} \vee \\ X_1^{\{01\}} X_2^{\{11\}} \vee X_1^{\{10\}} X_2^{\{01\}} \vee X_1^{\{10\}} X_2^{\{10\}} \vee X_1^{\{10\}} X_2^{\{11\}}$$

After simplification:

$$f = X_1^{\{00\}} X_2^{\{00,11\}} \vee X_1^{\{00,10\}} X_2^{\{01,1011\}}$$

Example

Example (continued)

Note that

$$\begin{aligned}X_1^{\{00\}} &= \bar{x}_1 \bar{x}_2 \\X_1^{\{01,10\}} &= x_1 \oplus x_2 \\X_2^{\{00,11\}} &= x_3 \oplus \bar{x}_4 \\X_2^{\{01,10,11\}} &= x_3 \vee x_4\end{aligned}$$

We could write f as

$$f = (\bar{x}_1 \bar{x}_2)(x_3 \oplus \bar{x}_4) \vee (x_1 \oplus x_2)(x_3 \vee x_4)$$

Example

		$X_1 = (x_1, x_2)$			
		00	01	11	10
	00	1			
$X_2 =$	01		1		1
(x_3, x_4)	11	1	1		1
	10		1		1

Figure 11.7 Example function.

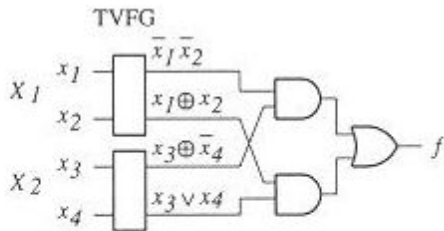


Figure 11.8 Realization of function in Fig. 11.7 using TVFGs.

Example

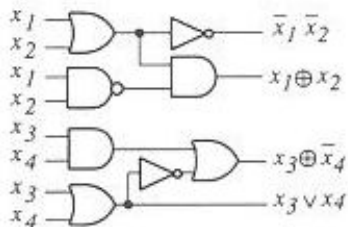


Figure 11.9 Macro expansion of a TVFG.

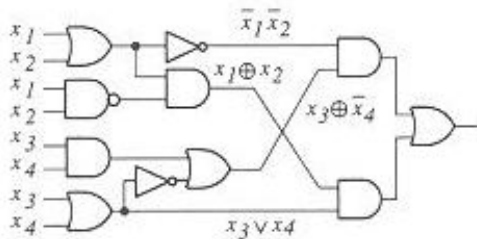


Figure 11.10 Final network.

This method can be subdivided into three independent sub-problems:

- 1 Assignment problem of the input variables to the TVFGs.
- 2 Simplification problem of the AND-OR two-level logic network.
- 3 The macro expansion problem of the TVFGs.

Algebraic Divisions of Logic Expressions

Theorem

Let $P(x) \neq 0$ be a polynomial with degree p , and let $S(x)$ be a polynomial with degree s . If $1 \leq p \leq s$, then there uniquely exists the polynomial $Q(x)$ with degree q , and the polynomial $R(x)$ with degree r satisfying the following conditions:

$$S(x) = P(x)Q(x) + R(x), q = s - p, \text{ and } 0 \leq r \leq p.$$

$Q(x)$ is the **quotient**, and $R(x)$ is the **remainder**.

Algebraic Divisions of Logic Expressions

Theorem

Let $P(x) \neq 0$ be a polynomial with degree p , and let $S(x)$ be a polynomial with degree s . If $1 \leq p \leq s$, then there uniquely exists the polynomial $Q(x)$ with degree q , and the polynomial $R(x)$ with degree r satisfying the following conditions:

$$S(x) = P(x)Q(x) + R(x), \quad q = s - p, \quad \text{and} \quad 0 \leq r < p.$$

$Q(x)$ is the **quotient**, and $R(x)$ is the **remainder**.

Example

Let $S(x) = x^3 - x + 1$ and $P(x) = x + 2$. Then, we have $Q(x) = x^2 - 2x + 3$ and $R(x) = -5$. Note that $Q(x)$ and $R(x)$ are unique.

Factoring SOPs

Similar factorization can be done with SOPs — however the problem is not so easy.

Example

Let $F = xy \vee z \vee w$, and $P = x \vee z$. Since $F = (x \vee z)(y \vee z) \vee w$, we have the solution $Q = y \vee z$ and $R = w$. However, F can also be represented as

$$F = (x \vee z)(y \vee z \vee w) \vee (z \vee w).$$

There exists another solution $Q = (y \vee z \vee w)$ and $R = (z \vee w)$.

The result of the division is not unique — we can not define a the quotient.

Theory of Division for SOPs

Definition

Let two SOPs be

$$F = \bigvee_{i=1}^p f_i \text{ and } G = \bigvee_{j=1}^q g_j$$

When F and G have no common variables, the **algebraic product** of F and G is defined as

$$F \cdot G = \bigvee_{i,j} (f_i, g_j).$$

Since, F and G have no common variables, the above operation does not produce any relation that is specific to the Boolean algebra.

Theory of Division for SOPs

Definition

Let F and P be two algebraic SOPs. F/P is the **quotient** of the algebraic SOP when F is divided by P . $Q = F/P$ is the maximum SOP that satisfies $F = Q \cdot P \vee R$. R is the **remainder**. $Q \cdot P$ is the **algebraic product**. When P is neither a constant 0 nor 1, P is a **divisor** of F . When the number of products in R is minimum, this division is a **weak division**.

Theory of Division for SOPs

Definition

Let F and P be two algebraic SOPs. F/P is the **quotient** of the algebraic SOP when F is divided by P . $Q = F/P$ is the maximum SOP that satisfies $F = Q \cdot P \vee R$. R is the **remainder**. $Q \cdot P$ is the **algebraic product**. When P is neither a constant 0 nor 1, P is a **divisor** of F . When the number of products in R is minimum, this division is a **weak division**.

For two given SOPs F and P , weak division produces the quotient Q and the remainder R uniquely.

Weak Division

Algorithm (Weak Division)

WEAK_DIV(F, P)

Let, $F = \{f_1, f_2, \dots, f_t\}$, and $P = \{p_1, p_2, \dots, p_s\}$

partition the literals of all products as follows:

$U = \{u_1, u_2, \dots, u_t\}$, and $V = \{v_1, v_2, \dots, v_s\}$

where $f_j = u_j \cdot v_j$, and u_j denotes a product of literals that are in p_i among the literals in f_j . Also, assume that if all the literals are deleted from u_j or v_j , then the result is 1. L

Weak Division

Algorithm (Weak Division) continued

Let,

$$V(p_i) = \bigcup_{u_j=p_i} v_j$$
$$Q = \bigcap_{i=1}^s V(p_i)$$
$$R = F - P \cdot Q$$

Return (Q, R)

Weak Division — Example

Example

Let $F = ac \vee ad \vee ae \vee bc \vee bd \vee be \vee \bar{a}b$, $P = a \vee b$ Then

$$U = (a, a, a, b, b, b, b),$$

$$V = (c, d, e, c, d, e, \bar{a}),$$

$$V(a) = \{c, d, e\},$$

$$V(b) = \{c, d, e, \bar{a}\},$$

$$Q = \{c, d, e\} = F/P, \text{ and}$$

$$R = \bar{a}b.$$

Therefore, $F = (a \vee b)(c \vee d \vee e) \vee \bar{a}b$.

Weak Division — Exercise

Exercise

Given

$$f = ab \vee ac \vee ad \vee bc \vee bd,$$

and

$$P = a \vee b.$$

Find Q and R

Definition

In an SOP with more than one product, if all the products have no common literals, then the SOP is **cube-free**. An Sop with only one product is not cube-free.

Cube-free

Definition

In an SOP with more than one product, if all the products have no common literals, then the SOP is **cube-free**. An Sop with only one product is not cube-free.

Exercise

Which of the following are cube-free?

- 1 $abc \vee abd$
- 2 abc
- 3 $abc \vee abd \vee bde \vee ae$

Definition

Let F be an SOP, and let c be a product term. The cube-free quotient F/c is a **kernel** of F . The set of all kernels of F is denoted by $K(F)$. If the original expression is cube-free, then the original expression is also a kernel (divided by the constant $c = 1$).

Definition

Let F be an SOP, and let c be a product term. The cube-free quotient F/c is a **kernel** of F . The set of all kernels of F is denoted by $K(F)$. If the original expression is cube-free, then the original expression is also a kernel (divided by the constant $c = 1$).

Kernels are derived from SOPs by factoring the common products. For example, consider $F = ab \vee ac \vee ad \vee bc \vee bd$. Since, $ab \vee ac \vee ad = a(b \vee c \vee d)$, we have a kernel $b \vee c \vee d$. Also, since $ac \vee bc = b(a \vee c)$, we have a kernel $a \vee c$.

Exercise

Let

$$F = ae \vee be \vee cde$$

$$G = ad \vee ae \vee bd \vee be \vee bf$$

$$H = ab\bar{c}$$

Find

$$K(F) =$$

$$K(G) =$$

$$K(H) =$$

Common Divisors

Theorem

Let F and G be SOPs. If F and G have a cube-free common divisor, then the cube-free common divisor for two SOPs, $H \in K(F)$ and $M \in K(G)$, consists of sum-of-products that are common to H and M .

Therefore, common divisors for two or more logical expressions are derived from the common product terms of the kernels.

Common Divisors

Example

$$F = ae \vee be \vee cde$$

$$G = ad \vee ae \vee bd \vee be \vee bf$$

$$K(F) = \{a \vee b \vee cd\}$$

$$K(G) = \{a \vee b, d \vee e, d \vee e \vee f, ad \vee ae \vee bd \vee be\}$$

Let $H = a \vee b \vee cd$ and $M = a \vee b$, then the divisor that is common to F and G is $a \vee b$.

Common Divisors

Example

Let us design the multi-level logic network for three functions F , G , and H .

$$F = ade \vee bde \vee cde \vee f$$

$$G = bg \vee cg \vee dg \vee aef$$

$$H = aeg \vee bc$$

Factoring using Product Terms

Example (continued ...)

(1) Factoring using product terms

The product term that is common to F , G , and H is ae . Let $X = ae$. Then we have:

$$F = Xd \vee bde \vee cde \vee f$$

$$G = bg \vee cg \vee dg \vee Xf$$

$$H = Xg \vee bc$$

$$X = ae$$

Kernel Extraction

Example (continued ... Kernel Extraction)

Kernel for F is $a \vee b \vee c$, and for G is $b \vee c \vee d$. We have:

$$F = (a \vee b \vee c)de \vee f$$

$$G = (b \vee c \vee d)g \vee aef$$

$$H = aeg \vee bc$$

The common Kernel for F and G is $b \vee c$. Let $X = b \vee c$. Then:

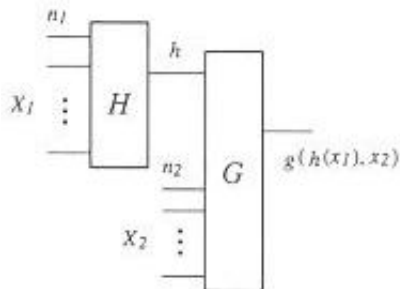
$$F = (a \vee X)de \vee f$$

$$G = (X \vee d)g \vee aef$$

$$H = aeg \vee bc$$

Functional Decomposition

- An n -variable function requires about $2^n/n$ gates.
- H requires about $2^{n_1}/n_1$ gates.
- G requires about $2^{n_2}/n_2$ gates.
- $2^n/n \gg 2^{n_1}/n_1 + 2^{n_2}/n_2$ for large n .



Bipartition

Definition

Let the input variables be $X = \{x_1, x_2, \dots, x_n\}$. Let $\{X\}$ denote the set of variables in X . If $\{X_1\} \cup \{X_2\} = \{X\}$ and $\{X_1\} \cap \{X_2\} = \phi$, then $X = (X_1, X_2)$ is a **bipartition** of X . Let the number of variables in X be denoted by $d(X)$.

Decomposition Chart

Definition

Given $f(X)$ where $X = (X_1, X_2)$. The **decomposition chart** of f is a table with 2^{n_1} columns and 2^{n_2} rows, and each row and column has a label with a binary number, and the corresponding element denotes the value of f , where $n_1 = d(X_1)$ and $n_2 = d(X_2)$, and each row and column has all the patterns of n_1 -bit and n_2 -bit, respectively.

Decomposition Chart - Example

					X_1					
		0	0	0	0	1	1	1	1	x_1
		0	0	1	1	0	0	1	1	x_2
$x_4 x_5$		0	1	0	1	0	1	0	1	x_3
	00	0	1	0	0	0	1	1	0	
	01	1	1	1	1	1	1	1	1	
X_2	10	1	0	1	1	1	0	0	1	
	11	0	1	0	0	0	1	1	0	

Gain of the Decomposition

Definition

The number of different column patterns of the decomposition chart is the **column multiplicity** and it is denoted by μ .

$\gamma = \min(2^{n_1}, 2^{n_2})/\mu$ is the **gain of the decomposition**.

Decomposition Chart

Definition

In a decomposition chart, let the logic function represented by distinct column patterns be $\Phi_i(X_2)$ ($i = 0, 1, \dots, \mu - 1$). Let $\Psi_i(X_1)$ be the function showing set of the input variables X_1 that produces $\Phi_i(X_2)$.

By definition of the decomposition chart, Ψ_i have the following property: $\Phi_i \cdot \Psi_j = 0 (i \neq j)$, $\bigvee_{i=0}^{\mu-1} \Psi_i = 1$. Also, Φ_i is represented as $\Phi_i = f(|X_1 = \Psi_i)$.

Gain of the Decomposition - Example

Example

In the decomposition chart, $X_1 = (x_1, x_2)$ and $X_2 = (x_3, x_4, x_5)$. Also, $n_1 = 2$, $n_2 = 3$, $\mu = 3$, and $\gamma = \min(2^2, 2^3)/3 = 4/3 = 1.33$.

			X_1				
			0	0	1	1	x_1
x_3	x_4	x_5	0	1	0	1	x_2
	000		0	0	0	1	
	001		1	1	1	1	
	010		1	1	1	0	
X_2	011		0	0	0	1	
	100		1	0	1	0	
	101		1	1	1	1	
	110		0	1	0	1	
	111		1	0	1	0	
			Φ_0	Φ_1	Φ_0	Φ_2	

Gain of the Decomposition - Example

Example (continued ...)

$$\Phi(X_2) = \bar{x}_3 \bar{x}_4 x_5 \vee \bar{x}_3 x_4 \bar{x}_5 \vee x_3 \bar{x}_4 \bar{x}_5 \vee x_3 \bar{x}_4 x_5 \vee x_3 x_4 x_5,$$

$$\Phi(X_2) = \bar{x}_3 \bar{x}_4 x_5 \vee \bar{x}_3 x_4 x_5 \vee x_3 x_4 \bar{x}_5 \vee x_3 \bar{x}_4 x_5,$$

$$\Phi(X_2) = \bar{x}_3 \bar{x}_4 \bar{x}_5 \vee \bar{x}_3 \bar{x}_4 x_5 \vee \bar{x}_3 x_4 x_5 \vee x_3 \bar{x}_4 x_5 \vee x_3 x_4 \bar{x}_5$$

$$\Psi(X_1) = \bar{x}_1 \bar{x}_2 \vee x_1 \bar{x}_2,$$

$$\Psi(X_1) = \bar{x}_1 x_2, \text{ and}$$

$$\Psi(X_1) = x_1 x_2.$$

Decomposable Functions

Definition

Let a p -valued output function $h : B^{n_1} \rightarrow P$, where $B = \{0, 1\}$, and $P = \{0, 1, \dots, p-1\}$, be defined as follows:
 $h(X_1) = i \Leftrightarrow \Psi_i(X_j) = 1 (i = 0, 1, \dots, p-1)$.

Definition

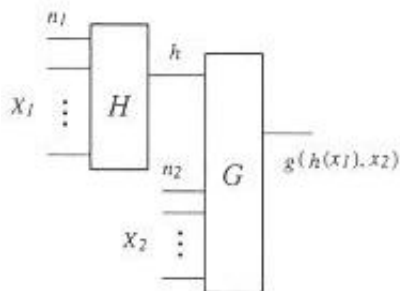
Let a function $g(Y, X_2) : P \times B^{n_2} \rightarrow B$, $B = \{0, 1\}$, $P = \{0, 1, \dots, p-1\}$ be defined as follows:
 $h(X_1) = i \Leftrightarrow g(i, X_2) = f(X_1, X_2)$.

Definition

For the functions f , g , and h in the previous definitions, $f(X) = g(h(X_1), X_2)$ is a decomposition of the function f . When $\gamma > 1$, this decomposition is **non-trivial**. When a function f has a non-trivial decomposition, f is **decomposable**.

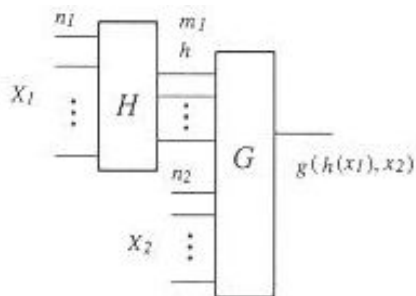
Simple Disjoint Decomposition

The **Simple Disjoint Decomposition** only considers functions of type in diagram when $\mu = 2$.



Disjoint Decomposition

Decomposable function with
 $\gamma > 1$



Theorem

Theorem

When the column multiplicity of the decomposition $f(X) = g(h(X_1), X_2)$ is μ , f is realizable by the network in the previous slide, where m_1 is the minimum integer such that $\mu \leq 2^{m_1}$.

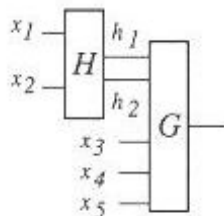
Proof.

From the definitions we have functions $g : M \times B^{n_2} \rightarrow B$, and $h : B^{n_1} \rightarrow M$, such that $f(X) = g(h(X_1), X_2)$, where $B = \{0, 1\}$, $M = \{0, 1, \dots, \mu - 1\}$, $n_1 = d(X_1)$, $n_2 = d(X_2)$, and μ denotes the column multiplicity. The the network H realizes m_1 functions h_1, h_2, \dots, h_{m_1} , where, binary vector $(h_1, h_2, \dots, h_{m_1})$ represents the value of M . Also the network G realizes the function g . Therefore, the network in the previous slide realizes the function f . □

Example

x_1	x_2	h	h_1	h_2
0	0	0	0	0
0	1	1	0	1
1	0	0	0	0
1	1	2	1	0

	$h_1 h_2$			
	00	01	11	10
$x_3 x_4 x_5$				
000	0	0	×	1
001	1	1	×	1
011	0	0	×	1
010	1	1	×	0
110	0	1	×	1
111	1	0	×	0
101	1	1	×	1
100	1	0	×	0



Partially Symmetric

Definition

Let $\{X_1\}$ be a subset of the input variables of a function f . If f is invariant under any permutation of the variables in X_1 , then f is **partially symmetric** with respect to $\{X_1\}$.

Lemma

If a function f is partially symmetric with respect to $\{X_1\}$, then the column multiplicity of the decomposition $f(X) = g(h(X_1), X_2)$ is at most $n_1 + 1$, where $n_1 = d(X_1)$.

Transformation of Networks

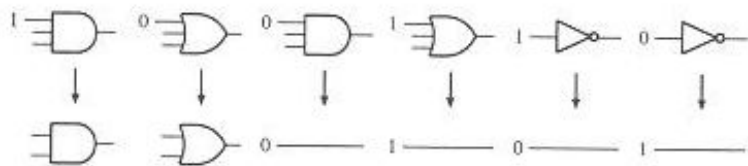


Figure 11.15 Reduction of constants.

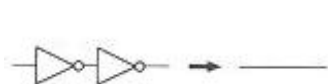


Figure 11.16 Reduction of inverters.

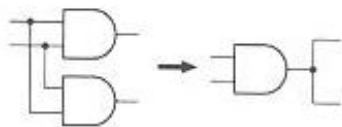


Figure 11.17 Reduction of duplicated gates.

Transformation of Networks

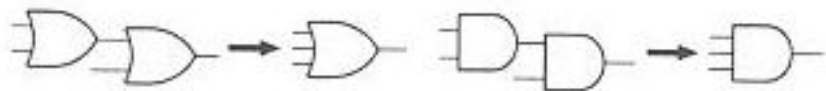


Figure 11.18 Gate merging.

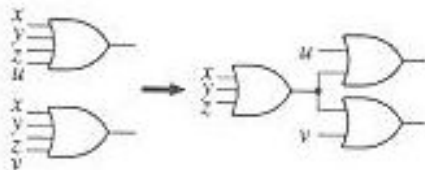


Figure 11.19 Sharing of factors.

Transformation of Networks

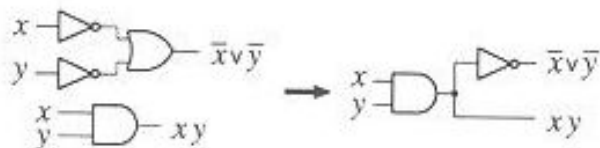


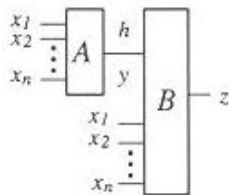
Figure 11.20 Simplification of networks using inverters.



Figure 11.21 Deletion of redundant connections.

Satisfiability don't Care

- Suppose network A is available and network B is under design.
- There are combinations that will never occur for network B.
- The set of such combinations is called the **Satisfiability don't Care** (SDC).
- $SDC = h \oplus y$



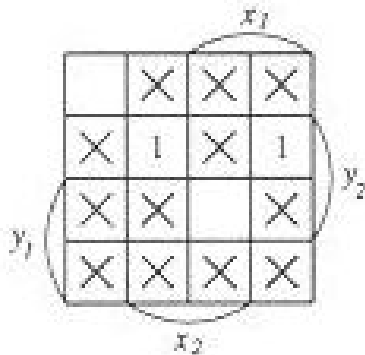
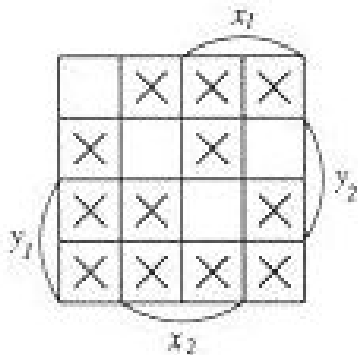
Satisfiability don't Care - Example

Example

Let the network A in the previous slide realize the function $h_1 = x_1 x_2$ and $h_2 = x_1 \vee x_2$, and the network B realize the function $z = x_1 \oplus x_2$.

$$\begin{aligned} \text{SDC} &= (h_1 \oplus y_1)(h_2 \oplus y_2) \\ &= (x_1 x_2) \bar{y}_1 \vee \overline{(x_1 x_2)} y_1 \vee (x_1 \vee x_2) \bar{y}_2 \vee \overline{(x_1 \vee x_2)} y_2. \end{aligned}$$

Karnaugh Maps



$z = x_1 \oplus x_2$ can be simplified to $z = y_2 \bar{y}_1$.