Logic Design using Modules

Gerhard Dueck

March 9, 2010
Outline

1. Logic Design using PLAs
2. PLAs with Input Decoders
PLA structure

Inputs

AND matrix

OR matrix

Outputs

Gerhard Dueck

Logic Design using Modules
- Direct relation to SOP
- Logic design is easy
- Layouts are easy—PLAs are structured
Networks can often be simplified by realizing $\bar{f}$, the complement of $f$, instead of the original function $f$.

**Definition (Achilles’ heel function)**

$$f = x_1 x_2 x_3 \lor x_4 x_5 x_6 \lor x_7 x_8 x_9 \lor \cdots \lor x_{n-2} x_{n-1} x_n \quad (n = 3r)$$

is an $n$-variable **Achilles’ heel function**.

**Lemma**

Let $f$ be an $n$-variable Achilles’ heel function ($n = 3r$). Let $t(f)$ be the number of product terms in a MSOP for $f$. Then, $t(f) = r$, and $t(\bar{f}) = 3^r$.

The complexity for $f$ and $\bar{f}$ can be quite different.
Some Properties

Property (Function with low density)

For n-variable functions whose densities are \( r \), if \( r \leq 2^{n-1} \), then the average number of products in the SOPs increases as \( r \) increases.

Property (Function with high density)

For n-variable functions whose densities are greater than \( 2^{n-1} \), the number of prime implicants is usually very large, and minimization is difficult. So, usually, the SOPs for the complements of the functions are simpler than the SOPs for the original functions.
**Theorem**

In a truth table for an n-input multiple-output function, if there are \( t \) input combinations that make all the output combinations 0, then the functions can be realized by a PLA with at most \( (2^n - t) \) products.

From the properties and the theorem we have the following: A good output phase has

- large number of input combinations that will make all outputs 0’s
- large number of 0’s in the truth table.
Example (Output phase assignment)

Design a 4-input **bit counting circuit** (WGT4) (see next slide)

- The straightforward PLA realization requires 15 products.
- The complemented function has no zero rows.
- The optimized output phase has 10 products.
### WGT4

<table>
<thead>
<tr>
<th>Input</th>
<th>Original</th>
<th>Optimized</th>
<th>Compl.</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$</td>
<td>$x_2$</td>
<td>$x_3$</td>
<td>$x_4$</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
The previous example leads to the following theorem:

**Theorem**

*The minimum PLA for the n-input bit-counting circuit (WGTn, n = 2r) requires \((2^n - 1)\) products when the output phase is original, and \(2^n - \binom{n}{n/2}\) products when the output phase is optimized.*
Algorithm (Near optimum output phase assignment)

1. For a given $m$-output function let PLA1 (with $2m$ outputs) be the PLA that realizes all $m$ outputs and their complements.

2. Let the output part of PLA1 be $G$. Attach the labels $P_1, P_2, \ldots, P_t$ to the rows of $G$. For each output $f_i (i = 0, \ldots, m - 1)$, make an SOP: $L_i = P_{a_1} P_{a_2} \cdots P_{a_r} \lor P_{b_1} P_{b_2} \cdots P_{b_s}$ where $P_{a_1} P_{a_2} \cdots P_{a_r}$, denote the rows whose $(i + 1)$th column of $G$ are 1’s (i.e. $f_i$). $P_{b_1} P_{b_2} \cdots P_{b_s}$, denote the rows whose $(i + m + 1)$th column of $G$ are 1’s (i.e. $\bar{f}_i$).

3. Expand the expression $Q(P_1, P_2, \ldots, P_t) = L_0 \cdot L_1 \cdots \cdots L_{m-1}$ into an SOP, and obtain the product with the fewest literals.

4. Obtain the output phase corresponding to the product obtained in 3.
**Example 12.2** Let us obtain the output phase for a 4-input bit-counting circuit (WGT4), using Algorithm 12.1.

1. Table 12.1 is the truth table for \( (f_2, f_1, f_0, \bar{f}_2, \bar{f}_1, \bar{f}_0) \). By simplifying this PLA by MINI2, we have a PLA shown in Table 12.2.

2. Let \( G \) be the output PLA1. Then,

\[

table{|c|c|c|c|c|c|c|c|c|c|\hline
f_2 & f_1 & f_0 & \bar{f}_2 & \bar{f}_1 & \bar{f}_0 & P_1 & P_2 & P_3 & P_4 & P_5 & P_6 & P_7 & P_8 & P_9 & P_{10} & P_{11} & P_{12} & P_{13} & P_{14} & P_{15} & P_{16} & P_{17} & P_{18} \\
\hline
1 & 0 & 0 & 0 & 1 & 1 & P_1 & P_2 & P_3 & P_4 & P_5 & P_6 & P_7 & P_8 & P_9 & P_{10} & P_{11} & P_{12} & P_{13} & P_{14} & P_{15} & P_{16} & P_{17} & P_{18} \\
0 & 0 & 1 & 0 & 1 & 0 & P_1 & P_2 & P_3 & P_4 & P_5 & P_6 & P_7 & P_8 & P_9 & P_{10} & P_{11} & P_{12} & P_{13} & P_{14} & P_{15} & P_{16} & P_{17} & P_{18} \\
0 & 0 & 1 & 0 & 1 & 0 & P_1 & P_2 & P_3 & P_4 & P_5 & P_6 & P_7 & P_8 & P_9 & P_{10} & P_{11} & P_{12} & P_{13} & P_{14} & P_{15} & P_{16} & P_{17} & P_{18} \\
0 & 0 & 0 & 0 & 1 & 1 & P_1 & P_2 & P_3 & P_4 & P_5 & P_6 & P_7 & P_8 & P_9 & P_{10} & P_{11} & P_{12} & P_{13} & P_{14} & P_{15} & P_{16} & P_{17} & P_{18} \\
0 & 0 & 1 & 0 & 1 & 0 & P_1 & P_2 & P_3 & P_4 & P_5 & P_6 & P_7 & P_8 & P_9 & P_{10} & P_{11} & P_{12} & P_{13} & P_{14} & P_{15} & P_{16} & P_{17} & P_{18} \\
0 & 1 & 1 & 0 & 0 & 0 & P_1 & P_2 & P_3 & P_4 & P_5 & P_6 & P_7 & P_8 & P_9 & P_{10} & P_{11} & P_{12} & P_{13} & P_{14} & P_{15} & P_{16} & P_{17} & P_{18} \\
0 & 1 & 0 & 0 & 0 & 1 & P_1 & P_2 & P_3 & P_4 & P_5 & P_6 & P_7 & P_8 & P_9 & P_{10} & P_{11} & P_{12} & P_{13} & P_{14} & P_{15} & P_{16} & P_{17} & P_{18} \\
0 & 1 & 0 & 0 & 0 & 1 & P_1 & P_2 & P_3 & P_4 & P_5 & P_6 & P_7 & P_8 & P_9 & P_{10} & P_{11} & P_{12} & P_{13} & P_{14} & P_{15} & P_{16} & P_{17} & P_{18} \\
0 & 1 & 1 & 0 & 0 & 0 & P_1 & P_2 & P_3 & P_4 & P_5 & P_6 & P_7 & P_8 & P_9 & P_{10} & P_{11} & P_{12} & P_{13} & P_{14} & P_{15} & P_{16} & P_{17} & P_{18} \\
0 & 1 & 0 & 0 & 0 & 1 & P_1 & P_2 & P_3 & P_4 & P_5 & P_6 & P_7 & P_8 & P_9 & P_{10} & P_{11} & P_{12} & P_{13} & P_{14} & P_{15} & P_{16} & P_{17} & P_{18} \\
0 & 1 & 1 & 0 & 0 & 0 & P_1 & P_2 & P_3 & P_4 & P_5 & P_6 & P_7 & P_8 & P_9 & P_{10} & P_{11} & P_{12} & P_{13} & P_{14} & P_{15} & P_{16} & P_{17} & P_{18} \\
0 & 0 & 0 & 1 & 0 & 0 & P_1 & P_2 & P_3 & P_4 & P_5 & P_6 & P_7 & P_8 & P_9 & P_{10} & P_{11} & P_{12} & P_{13} & P_{14} & P_{15} & P_{16} & P_{17} & P_{18} \\
0 & 0 & 0 & 1 & 0 & 0 & P_1 & P_2 & P_3 & P_4 & P_5 & P_6 & P_7 & P_8 & P_9 & P_{10} & P_{11} & P_{12} & P_{13} & P_{14} & P_{15} & P_{16} & P_{17} & P_{18} \\
\hline
\end{array}
\]

Let \( L_2, L_1, \) and \( L_0 \) be the expressions for outputs \( f_2, f_1, \) and \( f_0, \) respectively. Then, we have

\[
L_2 = P_1 \vee P_{14} P_{15} P_{16} P_{17} P_{18},
\]
\[
L_1 = P_7 P_8 P_9 P_{10} P_{11} P_{12} P_{13} P_{14} P_{15} P_{16} \vee P_1 P_3 P_4 P_5 P_6, \text{ and}
\]
\[
L_0 = P_2 P_3 P_4 P_6 P_7 P_{12} P_{15} P_{16} \vee P_1 P_5 P_8 P_9 P_{10} P_{11} P_{13} P_{14}.
\]
Example: output phase assignment

<table>
<thead>
<tr>
<th>Input part $x_4$ $x_3$ $x_2$ $x_1$</th>
<th>Output part</th>
</tr>
</thead>
<tbody>
<tr>
<td>01 01 01 01</td>
<td>100011</td>
</tr>
<tr>
<td>01 10 10 10</td>
<td>001010</td>
</tr>
<tr>
<td>10 10 10 01</td>
<td>001010</td>
</tr>
<tr>
<td>10 10 01 10</td>
<td>001010</td>
</tr>
<tr>
<td>10 10 10 10</td>
<td>000011</td>
</tr>
<tr>
<td>10 01 10 10</td>
<td>001010</td>
</tr>
<tr>
<td>01 10 01 01</td>
<td>011000</td>
</tr>
<tr>
<td>01 10 10 01</td>
<td>010001</td>
</tr>
<tr>
<td>01 10 01 10</td>
<td>010001</td>
</tr>
<tr>
<td>10 10 01 01</td>
<td>010001</td>
</tr>
<tr>
<td>10 01 01 10</td>
<td>010001</td>
</tr>
<tr>
<td>10 01 01 01</td>
<td>011000</td>
</tr>
<tr>
<td>10 01 10 01</td>
<td>010001</td>
</tr>
<tr>
<td>01 01 10 10</td>
<td>010101</td>
</tr>
<tr>
<td>01 01 01 10</td>
<td>011100</td>
</tr>
<tr>
<td>01 01 10 01</td>
<td>011100</td>
</tr>
<tr>
<td>10 11 11 11</td>
<td>000100</td>
</tr>
<tr>
<td>11 10 11 11</td>
<td>000100</td>
</tr>
</tbody>
</table>
Example: output phase assignment

3. By expanding the logical expression $Q(P_1, P_2, \ldots, P_{18}) = L_0 \cdot L_1 \cdot L_2$, we have the following expression:

\[ P_1P_2P_3P_4P_5P_6P_8P_9P_{10}P_{11}P_{13}P_{14}P_{15}P_{16}P_{17}P_{18} \]
\[ \lor P_1P_2P_3P_4P_5P_6P_7P_{12}P_{14}P_{15}P_{16}P_{17}P_{18} \]
\[ \lor P_1P_5P_7P_8P_9P_{10}P_{11}P_{12}P_{13}P_{14}P_{15}P_{16}P_{17}P_{18} \]
\[ \lor P_2P_3P_4P_6P_7P_8P_9P_{10}P_{11}P_{12}P_{13}P_{14}P_{15}P_{16}P_{17}P_{18} \]
\[ \lor P_1P_2P_3P_4P_5P_6P_8P_9P_{10}P_{11}P_{13}P_{14} \]
\[ \lor P_1P_2P_3P_4P_5P_6P_7P_{12}P_{15}P_{16} \]
\[ \lor P_1P_5P_7P_8P_9P_{10}P_{11}P_{12}P_{13}P_{14}P_{15}P_{16} \]
\[ \lor P_1P_2P_3P_4P_6P_7P_8P_9P_{10}P_{11}P_{12}P_{13}P_{14}P_{15}P_{16} \]

Note that the first product corresponds to $(\bar{f}_2, \bar{f}_1, f_0)$, the second one corresponds to $(\bar{f}_2, \bar{f}_1, f_0), \ldots$, and the last term corresponds to $(f_2, f_1, f_0)$. The product with the fewest literals is the 6-th, and it corresponds to the output $(f_2, \bar{f}_1, f_0)$.
Standard PLAs have 1-bit input decoders.

The decoders for PLAs with 2-bit input decoders are shown on the left.

How many 2 input decoders are possible?
2-bit input decoders realize all maxterms of two variables.

An two input function can be realized by a logical product of these maxterms.

This is the canonical expression

\[ f(x_1, x_2) = (c_0 \lor x_1 \lor x_2)(c_1 \lor x_1 \lor \overline{x_2})(c_2 \lor \overline{x_1} \lor x_2)(c_3 \lor \overline{x_1} \lor \overline{x_2}) \]

What is the worst case for 2-variable function using a standard PLA?
Consider the \( f = f_1(x_1, x_2)f_2(x_3, x_4)f_3(x_5, x_6) \), where

\[
\begin{align*}
  f_1(x_1, x_2) &= x_1x_2 \lor \overline{x_1}\overline{x_2} \\
  f_2(x_3, x_4) &= x_3x_4 \lor \overline{x_3}\overline{x_4}, \text{ and} \\
  f_3(x_5, x_6) &= x_5x_6 \lor \overline{x_5}\overline{x_6}
\end{align*}
\]

Show the PLA realization (see Fig. 12.7 in the text).
### Representation for PLAs with input decoders

- 2 variables can be combined into one 4-valued variable.
- In the coincidence example, let $X_1 = (x_1, x_2)$, $X_2 = (x_3, x_4)$, and $X_3 = (x_5, x_6)$.
- The function is represented as

<table>
<thead>
<tr>
<th>$X_1$</th>
<th>$X_2$</th>
<th>$X_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>00</td>
<td>00</td>
</tr>
<tr>
<td>01</td>
<td>01</td>
<td>01</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>01</td>
</tr>
<tr>
<td>11</td>
<td>11</td>
<td>11</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>00</th>
<th>01</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

#### Theorem

1. **Each product line in a PLA with 2-bit input decoders realizes a product**
   
   $$f_1(x_1, x_2)f_2(x_3, x_4) \cdots f_r(x_{n-1}, x_n)$$

2. **The function realized by the product lines is represented by**
   
   $$c_0^1c_1^1c_2^1c_3^1 - c_0^2c_2^2c_2^2c_3^2 - \cdots - c_0^rc_r^rc_r^rc_r^r(n = 2r)$$
Algorithm (Design with input decoders)

Step 1  Transform the logical expression into a bit representation
  1  Partition input variables into pairs.
  2  Derived bit representation for the function with 4-valued variables.

Step 2  Transform the bit representation into a PLA.
  1  Input part: For the part 0, make an AND connection
  2  Output part: For the part 1, make an OR connection
Example (2-bit adder)

Find the bit representation for the 2 bit adder.

\((x_1, x_0) + (y_1, y_0) = (z_2, z_1, z_0)\). We have:

\[
\begin{align*}
    z_0 &= x_0 \oplus y_0 = x_0 \overline{y_0} \lor \overline{x_0} y_0, \\
    z_1 &= x_0 y_0 \oplus x_1 \oplus y_1 \\
    &= (\overline{x_0} \lor \overline{y_0})(x_1 \overline{y_1} \lor \overline{x_1} y_1) \lor (x_0 y_0)(x_1 y_1 \lor \overline{x_1} \overline{y_1}), \text{ and} \\
    z_2 &= x_1 y_1 \lor (x_0 y_0)(x_1 \lor y_1).
\end{align*}
\]
Example (2-bit adder) cont.

Let \( X_1 = (x_0, y_0) \), \( X_2 = (x_1, y_1) \), and \( X_3 = (z_0, z_1, z_2) \). Then we have

\[
\begin{array}{cccc}
X_1 & X_2 & X_3 \\
0110 & 1111 & 100 \\
1110 & 0110 & 010 \\
0001 & 1001 & 010 \\
1111 & 0001 & 001 \\
0001 & 0111 & 001 \\
\end{array}
\]

The PLA realization is shown in Fig. 12.8 (text)
Bound for an arbitrary function

**Theorem**

An arbitrary $n$-variable function ($n = 2r$) is realized by a PLA with 2-bit input decoders using at most $2^{n-2}$ products.

**Proof.**

An arbitrary $n$-variable function is expanded as

$$f = \bigvee_{a} f(x_1, x_2, a) \cdot (x_3^{a_3} x_4^{a_4}) \cdot (x_5^{a_5} x_6^{a_6}) \cdots (x_{n-1}^{a_{n-1}} x_n^{a_n}),$$

where $a = (a_3, a_4, \ldots, a_n)$, $a_i \in \{0, 1\}$. The number of product terms is $2^{n-2}$.
Theorem
An arbitrary $n$-variable symmetric function ($n = 2r$) is realized by a PLA with 2-bit input decoders using at most $3^{r-1}$ products.

Proof.
An arbitrary $n$-variable symmetric function is expanded as

$$f = \bigvee_{b} S(x_1, x_2, b) \cdot S_{b_2}(x_3, x_4) \cdot S_{b_3}(x_5, x_6) \cdots \cdot S_{b_r}(x_{n-1}, x_n),$$

where $b = (b_2, b_3, \ldots, b_r)$, $b_i \in \{0, 1, 2\}$, and

$$S_j(x_i, x_{i+1}) = \begin{cases} 1, & \text{when } x_i + x_{i+1} = j \\ 0, & \text{otherwise} \end{cases}$$

Note that $S(x_1, x_2, b)$ is symmetric with respect to $x_1$ and $x_2$. 

Gerhard Dueck
Theorem

An $n$-variable parity function ($n = 2r$) is realized by a PLA with 2-bit input decoders using at most $2^{r-1}$ products.

Proof.

An $n$-variable parity function is expanded as

$$f = \bigvee_{c} P(x_1, x_2, c) \cdot P_{c_2}(x_3, x_4) \cdot P_{c_3}(x_5, x_6) \cdots \cdot P_{c_r}(x_{n-1}, x_n),$$

where $c = (c_2, c_3, \ldots, c_r)$, $c_i \in \{0, 1\}$, and

$$P_{c_j}(x_i, x_{i+1}) = x_i \oplus x_{i+1} \oplus c_j$$

Note that $P(x_1, x_2, c)$ is also a parity function.
**Table:** Number of products to realize $n$-variable functions.

<table>
<thead>
<tr>
<th></th>
<th>1-bit decoders</th>
<th>2-bit decoders</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arbitrary function (worst case)</td>
<td>$2^{n-1}$</td>
<td>$2^{n-2}$</td>
</tr>
<tr>
<td>Symmetric function (worst case)</td>
<td>$2^{n-1}$</td>
<td>$(\sqrt{3})^{n-2}$</td>
</tr>
<tr>
<td>Parity function</td>
<td>$2^{n-1}$</td>
<td>$(\sqrt{2})^{n-2}$</td>
</tr>
<tr>
<td>Adder</td>
<td>$6 \cdot 2^n - 4n - 5$</td>
<td>$n^2 + 1$</td>
</tr>
<tr>
<td>10-variable random function (average)</td>
<td>163</td>
<td>120</td>
</tr>
</tbody>
</table>
Assignment of input variables

Example

Given a 2-bit adder

- 5 products are needed with assignment $X_1 = (x_0, y_0)$ and $X_2 = (x_1, y_1)$.
- 9 products are needed with assignment $X_1 = (x_0, x_1)$ and $X_2 = (y_0, y_1)$.

- Number of product terms greatly depends on the variable assignment

- **Optimum assignment of input variables**
  - exhaustive (how many possibilities?)
  - for experimental results see Table 12.3
  - heuristics
Definition (Partition)

Let $I = \{1, 2, \ldots, n\}$ be a set of subscripts for the input variables $X$. Let $\Pi$ be a partition of $I$. Let $t(f : \Pi)$ be the number of products in a MSOP for $f$, under the partition $\Pi$. Let $F$ be an SOP for the function $f$. Let $q(i, j)$ be the number of different products in the SOP that are obtained from $F$ by deleting literals for $x_i$ and $x_j$. Let $t(f : \Pi_{ij})$ be the number of products in a minimum SOP for $f$, when $x_i$ and $x_j$ are paired.
Optimum assignment

Example

Let \( F \) be

\[
F = \overline{x_1}x_2x_3x_4 + \overline{x_1}x_2x_3x_4 + x_1\overline{x_2}x_3x_4 + \\
x_1\overline{x_2}x_3x_4 + x_1\overline{x_2}x_3\overline{x_4} + x_1x_2\overline{x_3}\overline{x_4}
\]

\[
q(1, 2) = \\
q(1, 3) = \\
q(1, 4) = \\
q(2, 3) = \\
q(2, 4) = \\
q(3, 4) =
\]
Lemma

Let $\Pi_{ij} = \{[1], [2], \ldots, [i, j], \ldots, [n]\}$. Then, $t(f : \Pi_{ij}) \leq q(i, j)$. 
Proof.

Let $F$ be an SOP for $f$. Assume $i = 1$ and $j = 2$ (WLOG).

$$F = \bigvee_{S} x_1^{S_1} x_2^{S_2} \cdots x_n^{S_n}$$

where $S = (S_1, S_2, \ldots, S_n)$, and $S_i \subseteq \{0, 1\}$. Note that

$$x_i^{S_i} = \begin{cases} 
1, & \text{when } S_i = \{0, 1\} \\
 x_i, & \text{when } S_i = \{1\} \\
\overline{x_i}, & \text{when } S_i = \{0\}
\end{cases}$$

Factoring $F$ by $x_1^{S_1} x_2^{S_2} \cdots x_n^{S_n}$ we have

$$F = \bigvee_{S^*} G(x_1, x_2, S^*) x_3^{S_3} x_4^{S_4} \cdots x_n^{S_n},$$
where \( S^* = (S_3, S_4, \ldots, S_n) \), and \( S_i \subseteq \{0, 1\} \)

- \( t(F_1) \) is equal to the distinct number of patterns in \( x_3^{S_3} x_4^{S_4} \cdots x_n^{S_n} \).
- These are obtained from \( F \) by deleting literals \( x_1 \) and \( x_2 \)
- \( t(F_1) = q(1, 2) \) (by definition)
- Let \( X_1 = (x_1, x_2) \). Replace \( G(x_1, x_2, S^*) \) with literal \( X_1^{T_1} \) (where \( T_1 \subseteq \{00, 01, 10, 11\} \)) we have \( F_1 \) which is an SOP under partition \( \Pi_{ij} \).
- \( t(f : \Pi_{ij}) \leq t(F_1) \)
- Thus we have \( t(f : \Pi_{ij}) \leq q(i, j) \)
Definition (Variable assignment graph)

A **variable assignment graph** $G$ of an $n$-variable function $f(x_1, x_2, \ldots, x_n)$ is a complete graph with weights satisfying the following conditions:

1. $G$ has $n$ nodes.
2. The weight of the edge $(i, j)$ is $q(i, j)$. 

$G$ is a **complete graph** 

- **Complete graph**: A graph in which every pair of distinct nodes is connected by a unique edge.
Variable assignment

**Algorithm (Assignment of variables for a PLA with 2-bit input decoders)**

1. Obtain a (near) minimal SOP for $f$.
2. Obtain a variable assignment graph $G$ for $f$.
3. Cover all nodes of $G$ by a set of edges that have no common elements. Find a set such that the sum of the weights are minimum.
4. Obtain the partition of the variables corresponding to the edges.

- The algorithm is heuristic. Minimal results are not guaranteed.
- Experimental results show that it obtains optimal solutions for many functions.
## Benchmarks

<table>
<thead>
<tr>
<th>func.</th>
<th>products</th>
<th>Stand. PLA</th>
<th>PLA inp. decod.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Output phase</td>
<td>Output phase</td>
</tr>
<tr>
<td>addr4</td>
<td>255</td>
<td>75</td>
<td>61</td>
</tr>
<tr>
<td>inc8</td>
<td>255</td>
<td>37</td>
<td>36</td>
</tr>
<tr>
<td>log8</td>
<td>255</td>
<td>123</td>
<td>111</td>
</tr>
<tr>
<td>mul8</td>
<td>225</td>
<td>119</td>
<td>108</td>
</tr>
<tr>
<td>rdm8</td>
<td>255</td>
<td>76</td>
<td>76</td>
</tr>
<tr>
<td></td>
<td></td>
<td>17</td>
<td>14</td>
</tr>
<tr>
<td></td>
<td></td>
<td>21</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td></td>
<td>93</td>
<td>89</td>
</tr>
<tr>
<td></td>
<td></td>
<td>85</td>
<td>74</td>
</tr>
</tbody>
</table>