This paper deals with matrix modelling of trivial logic arrays (PLA, PAL, ROM) and the design of the above array as structural models of static and dynamic logic objects.

Keywords: cartesian product of matrices, PLA, PAL, ROM, canonical decomposition, states coding, states coding of Miller or Liu, substitute input variable.

1 Introduction

The trivial logic arrays dealt with here [1,2] (Fig. 1.) are:
- PLA (programmable logic array) with programmable input and output matrices,
- PAL (programmable array logic) with a programmable input matrix and a given output matrix,
- ROM (read only memory) a given input matrix (is given through an address decoder) and a programmable output matrix.

Fig. 1: Trivial logic arrays: a) block diagram, b) labelling; examples of logic arrays: c) PLA, d) PAL, e) ROM, where × and ⋄ mean the programmable and given positions, respectively.
The input matrix has either conjunctors (\& - AND) or inverse disjunctors (\(\lor - NAND\)). The output matrix has either disjunctors (\(\lor - OR\)) or inverse disjunctors (NOR).

The paper deals with matrix analysis and synthesis of trival logic arrays.

2 Cartesian product of Boolean matrices

The Cartesian product \(M(\bigvee_{\bigwedge} \mu) \in \mathbb{R}\) of Boolean matrices

\[
M : \{1, 2, \ldots, q\} \times \{1, 2, \ldots, p\} \rightarrow \{0, 1\} : (i, j) \mapsto m_{ij}
\]

\[
\mathbb{R} : \{1, 2, \ldots, \mu\} \times \{1, 2, \ldots, \nu\} \rightarrow \{0, 1\} : (i, j) \mapsto m_{ij}
\]

denotes a Boolean matrix

\[
M(\bigvee_{\bigwedge} \mu) \in \mathbb{R} : \{1, 2, \ldots, q\} \times \{1, 2, \ldots, p\} \rightarrow \{0, 1\} : (i, j) \mapsto m_{ij}
\]

where \(\mu_{ik} = \left(\bigwedge_{j} \bigvee_{i} \mu_{ij}\right)\) (and \(\bigvee_{m} \) and \(\bigwedge_{m}\) are Boolean operators.

Example 1: Let a system \(\{y_1, y_2\}\) of Boolean functions \(y_1, y_2\) be given

\[
\{y_1, y_2\} = \{0, 1\}\^{\{0,1\}^4}.
\]

\[
\begin{align*}
x_1 & x_2 & x_3 & x_4 & y_1 & y_2 \\
1 & 0 & 0 & 1 & 0 & 1 \\
0 & 1 & 0 & 0 & 1 & 1 \\
0 & 1 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 1 & 1 & 0
\end{align*}
\]

Write the system \(\{y_1, y_2\}\) as matrices \(\{\text{cnf}(y_1), \text{cnf}(y_2)\}\), where \(\text{cnf}\) denotes a canonical normal disjunctive formula. Note that \((x = 0) = \overline{x}\) and \((x = 1) = x\):

\[
[1 0 0 1] = [0 1 0 0 \bigvee_{\bigwedge} 1 1] = [0 0 0 0 \bigvee_{\bigwedge} 0 0] = [0 1 1 1 \bigvee_{\bigwedge} 1 1] = [1 1 0 1 \bigvee_{\bigwedge} 1 0] = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 \end{bmatrix} \bigvee_{\bigwedge} \begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{bmatrix}
\]

3 Structural model of a static logic object

Let \(y_j : \{0, 1\}^m \rightarrow \{0, 1\} : (x_1, x_2, \ldots, x_m) \mapsto y_j\) be a Boolean function and the literal \(x^o\) be \(x^o = x^o \lor x^o (\sigma \in \{0, 1\})\), where \(x^0 = \overline{x}\) and \(x^1 = x\). Note that the function \(y_j\) can be expressed as

\[
\text{cnf}(y_j) = \bigvee_{\bigwedge} x_1^0, x_2^0, \ldots, x_m^0 = \bigvee_{\bigwedge} K_{ij},
\]

or as \(\text{nfd}\) (a normal disjunctive formula)

\[
\text{nfd}(y_j) = \bigvee_{\bigwedge} \sigma_1^0, \sigma_2^0, \ldots, \sigma_m^0 = \bigvee_{\bigwedge} K_{ij},
\]

where \(K_{ij}\) is a normal conjunct on

\[
X = \{x_1, x_2, x_3, \ldots, x_m, \overline{x_m}\}
\]

or finally expressed as \(\text{mndf}(y_j)\) (minimal \(\text{nfd}\)). Let the symbol \(\Lambda\) denote a set of conjuncts \(K_{ij}\) from \(\text{nfd}(y_j)\).

Let

\[
\{0, 1\}^m : \{0, 1\}^n, \Lambda
\]

be a finite automaton model of a binary static logic object where \(\Lambda\) is the vector output function

\[
\Lambda : \{0, 1\}^m \times \{0, 1\}^n : (x_1, x_2, \ldots, x_m) \mapsto \{y_1, y_2, \ldots, y_n\}
\]

represented by a system \(\{\lambda_j\}_{j=1}^n\) of components, output functions

\[
\lambda_j : \{0, 1\}^m : \{0, 1\} : (x_1, x_2, \ldots, x_m) \mapsto y_j.
\]

Let \(M_m : \{0, 1\}^m \rightarrow \{0, 1\}^n\) be the input (output) matrix of the given static object, i.e.:

\[
M_m : \{1, 2, \ldots, \overline{\{0, 1\}}\} \times \{1, 2, \ldots, 2m\} \rightarrow \{0, 1\} : (r, s) \mapsto 0 \text{ for } x_i^s \notin K_{ij}, (r, s) \mapsto 1 \text{ for } x_i^s \in K_{ij}.
\]

\[
M_{\text{out}} : \{1, 2, \ldots, \overline{\{0, 1\}}\} \times \{1, 2, \ldots, n\} \rightarrow \{0, 1\} : (r, s) \mapsto 0 \text{ for } K_{ij} \notin \text{nfd}(\lambda_j), (r, s) \mapsto 1 \text{ for } K_{ij} \in \text{nfd}(\lambda_j).
\]
If $X^T: \{0, 2, ..., 2m\} \rightarrow X: s \mapsto x_s \text{ for } s = 2k + 1,$

$s \mapsto \bar{x}_s \text{ for } s = 2k (k = 0, 1, 2, \ldots, m - 1),$ then for the trivial logic array $M$ the following are valid

$[y_1, y_2, \ldots, y_n] = (M_{in} \& \& X)^T \lor \lor M_{out},$

$[\bar{y}_1, \bar{y}_2, \ldots, \bar{y}_n] = (M_{in} \& \& \bar{X})^T \lor \lor M_{out},$

where $X = [x_1, x_2, x_3, \ldots, x_m, \bar{x}_m].$

The total capacity (area) $C(M)$ (bit) of trivial logic array $M$ can be written as

$C(M) = C(M_{in}) + C(M_{out}) = (2m + n)[K].$

Example 2: Design a $PLA(M_{in}, M_{out})$ modeled system of output functions

$y_1 = (x_1 \oplus x_2)\bar{x}_3 \lor (x_1 = x_2) x_3 = \bar{x}_3 \bar{x}_3 \lor x_3$ $y_2 = x_1 x_2 \lor (x_1 \oplus x_2) x_3 = x_1 x_2 \lor \bar{x}_3 x_3$

and determine its capacity

$\xi = x_1 x_2 \lor \bar{x}_3 \bar{x}_2, \bar{\xi} = \bar{x}_1 x_2 \lor x_1 \bar{x}_2 \bigg) \quad \text{Hence}$

$M_{in} = \begin{bmatrix}
1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & x_1 & x_2 \\
0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & \bar{x}_1 & \bar{x}_2 \\
0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & \bar{x}_1 & \bar{x}_2 \\
0 & 0 & 0 & 0 & 1 & 0 & 1 & \bar{x}_1 & x_2 & x_3 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & \bar{x}_1 & \bar{x}_2 & x_3 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & \bar{x}_1 & \bar{x}_2 & x_3 \\
\end{bmatrix}$

$M_{out} = \begin{bmatrix}
y_1 & y_2 & \xi \\
0 & 1 & 1 & x_1 & x_2 \\
0 & 0 & 1 & \bar{x}_1 & \bar{x}_2 \\
0 & 0 & 0 & \bar{x}_1 & x_2 \\
1 & 0 & 0 & \bar{x}_3 & \bar{x}_4 \\
1 & 0 & 0 & \bar{x}_3 & \bar{x}_4 \\
0 & 1 & 0 & x_3 & x_3 \\
0 & 1 & 0 & \bar{x}_3 & \bar{x}_3 \\
\end{bmatrix}$

and Fig. 2 can be constructed.

Since

$\begin{bmatrix}
y_1 & y_2 & \xi \\
0 & 1 & 1 \\
0 & 0 & 1 \\
0 & 0 & 0 \\
1 & 0 & 0 & \bar{x}_3 \bar{x}_3 \lor x_3 \bar{x}_3, x_1 x_2 \lor x_3 \bar{x}_3, x_1 x_2 \lor \bar{x}_3 \bar{x}_3 \\
1 & 0 & 0 & \bar{x}_3 \bar{x}_3 \lor x_3 \bar{x}_3, x_1 x_2 \lor x_3 \bar{x}_3, x_1 x_2 \lor \bar{x}_3 \bar{x}_3 \\
0 & 1 & 0 & \bar{x}_3 \bar{x}_3 \lor x_3 \bar{x}_3, x_1 x_2 \lor x_3 \bar{x}_3, x_1 x_2 \lor \bar{x}_3 \bar{x}_3 \\
0 & 0 & 0 & \bar{x}_3 \bar{x}_3 \lor x_3 \bar{x}_3, x_1 x_2 \lor x_3 \bar{x}_3, x_1 x_2 \lor \bar{x}_3 \bar{x}_3 \\
\end{bmatrix}$

$\lor \lor M_{out}.$

and $C(M) = 2 \times 4 \times 6 + 3 \times 6 = 66.$

The starting point for design $M_{in}$ and $M_{out}$ trivial logic array matrices is a system of tables $\{\lambda_j\}$ (ROM) or a compressed form $[2] (PLA, PAL)$ of it, including the corresponding record of the $cnfd(\{\lambda_j\})$ of the group function or the $nfd(\{\lambda_j\})$ of the component functions of $\lambda_j.$

Example 3: The following system of functions

$\begin{array}{cccccc}
x_1 & x_2 & y_1 & y_2 & y_3 \\
0 & 0 & 0 & 1 & 1 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 \\
1 & 0 & 0 & - & - & - \\
1 & 0 & 1 & - & 0 & 1 \\
1 & 1 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & - & 0 & 1 \\
\end{array}$

and $\text{cnfd(}y_1, y_2, y_3) = \overline{x}_1 \overline{x}_2 \overline{x}_3 (y_1 y_2) \lor \overline{x}_1 \overline{x}_2 x_3 (y_1) \lor \overline{x}_1 x_2 \overline{x}_3 (y_1 y_2) \lor \overline{x}_1 x_2 x_3 (y_1) \lor x_1 \overline{x}_2 x_3 (y_3) \lor x_1 x_2 x_3 (y_3)$

including the group $cnfd$ or $nfd$:

$\text{cnfd}$

$(y_1, y_2, y_3) = \overline{x}_1 \overline{x}_2 \overline{x}_3 (y_1 y_2) \lor \overline{x}_1 \overline{x}_2 x_3 (y_1) \lor \overline{x}_1 x_2 \overline{x}_3 (y_1 y_2) \lor \overline{x}_1 x_2 x_3 (y_1) \lor x_1 \overline{x}_2 x_3 (y_3) \lor x_1 x_2 x_3 (y_3)$

including the group $cnfd$ or $nfd$:
or \(ndf\)

\[
(y_1, y_2, y_3) = \bar{x}_1 x_3 (y_1 y_2) \lor \bar{x}_1 x_5 (y_1) \lor x_1 x_2 (y_3).
\]

\(*\) Note that the undetermined values of function arguments need to be interpreted as both zeroes and ones, whereas the undetermined values of functions are very difficult, uninteresting, or even impossible, to determine and their delivery is motivated by the maximal rate of their utility.

If the system \(\{\lambda_j\}_{j=1}^n\) is also given by the system \(\{cndf(y_j)\}_{j=1}^n\), then it is to be decided whether to realise ROM by means of the \(cndf(y_j)\) array or the \(cndf(\bar{y}_j)\) array. If, therefore, for the Hamming weight \(w_H(y_j)\) of the function \(\lambda_j\), – \(w_H(y_j) \leq 2^n/2\) holds, then naturally we work with use \(cndf(y_j)\) or otherwise we do so with \(cndf(\bar{y}_j)\).

Since \(cndf(y_j)\) or \(cndf(\bar{y}_j)\) are very complicated systems in practice – the systems of \(\lambda_j\) functions are systems consisting of tens or hundreds of functions depending on tens or hundreds of arguments – the classic minimisation procedures are not applicable on \(cndf(y_j)\) or \(cndf(\bar{y}_j)\). With advantage, however, the Quine and McCluske method of minimising \(\{cndf(y_j)\}_{j=1}^n\) systems can be used, under the condition that the definition of the undetermined values of functions \(\lambda_j\) of system \(\{\lambda_j\}_{j=1}^n\) will be suitably defined and the covering table \([2]\) will be not constructed. In this way, we will obtain a subminimal group \(ndf(y_1, y_2, \ldots, y_n)\) – see Example 4.

Example 4: Let \(y_1 = 00010111\) and \(y_2 = 00010110\), that is

\[
y_1 = (x_1 \lor x_2) x_3 \lor x_2 x_2 and y_2 = \bar{x}_1 x_2 x_3 \lor x_1 (\bar{x}_2 x_3 \lor x_2 \bar{x}_3)
\]

or also \(y_2 = y_1 x_1 x_2 x_3\) since:

<table>
<thead>
<tr>
<th>(x_1)</th>
<th>(y_1)</th>
<th>(x_2)</th>
<th>(x_3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

\[
\begin{array}{ccc|c}
1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
3 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
4 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\
5 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
7 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
8 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
10 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
11 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\]

Let the rows of the input (output) matrix be compatible if each of them contains at least one common input literal (one common output function). Since the relation of compatibility of rows is their relation of tolerance, on the set of rows of the matrix it defines the covering \(C_{\{i\}_{j=1}}^{max}(\{i\}_{j=1})\) with all maximal classes:

\[
C_{\{i\}_{j=1}}^{max}(\{i\}_{j=1}) = \{1, 2, 4, 5, 7, 8, 9, 10\} \cup C_{\{i\}_{j=1}}^{max}(\{i\}_{j=1})
\]

Since the rows within the input (output) matrix correspond to the sets of input literals (output functions):

\[
\begin{array}{ccc|c}
1 & 2 & 3 & 4 & 5 & 6 & 7 \\
1 & 0 & 1 & 0 & 0 & 0 & 0 \\
2 & 0 & 0 & 1 & 0 & 0 & 0 \\
3 & 0 & 0 & 1 & 1 & 0 & 0 \\
4 & 1 & 0 & 0 & 0 & 0 & 0 \\
5 & 1 & 1 & 1 & 0 & 0 & 0 \\
6 & 0 & 0 & 0 & 0 & 0 & 0 \\
7 & 0 & 1 & 0 & 0 & 0 & 0 \\
8 & 1 & 0 & 1 & 0 & 0 & 0 \\
9 & 0 & 0 & 0 & 1 & 1 & 0 \\
10 & 1 & 0 & 1 & 1 & 0 & 0 \\
11 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\]
According to $C_{\max}(\{i\})$ we get to the covering on the set of input variables (output functions) of all maximum classes:

$$C_{\max}(\{x_j\}_{j=1}^{3}) = \{\{x_1, x_2, x_3\}, \{x_3, x_4, x_5\}\}$$

$$C_{\max}(\{x_k\}_{k=1}^{7}) = \{\{y_1, y_2, y_3\}, \{y_1, y_4, y_5, y_6, y_7\}\}$$

i.e., we obviously obtain sparse matrices

<table>
<thead>
<tr>
<th>Rows</th>
<th>Stimuli</th>
<th>Responses</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>${x_1, x_2}$</td>
<td>${y_2}$</td>
</tr>
<tr>
<td>2</td>
<td>${x_1}$</td>
<td>${y_2}$</td>
</tr>
<tr>
<td>3</td>
<td>${x_3, x_5}$</td>
<td>${y_5, y_4}$</td>
</tr>
<tr>
<td>4</td>
<td>${x_1}$</td>
<td>${y_1}$</td>
</tr>
<tr>
<td>5</td>
<td>${x_3, x_5}$</td>
<td>${y_1, y_2, y_3}$</td>
</tr>
<tr>
<td>6</td>
<td>${x_5}$</td>
<td>${y_5, y_7}$</td>
</tr>
<tr>
<td>7</td>
<td>${x_1, x_3}$</td>
<td>${y_2}$</td>
</tr>
<tr>
<td>8</td>
<td>${x_1, x_2}$</td>
<td>${y_1, y_3}$</td>
</tr>
<tr>
<td>9</td>
<td>${x_5}$</td>
<td>${y_4, y_5}$</td>
</tr>
<tr>
<td>10</td>
<td>${x_5, x_4, x_1}$</td>
<td>${y_1, y_2, y_3}$</td>
</tr>
<tr>
<td>11</td>
<td>${x_4}$</td>
<td>${y_4, y_6}$</td>
</tr>
</tbody>
</table>

The matrices $M_{in}$ ($M_{out}$) are represented by submatrices

$$M_{in1} = \begin{bmatrix}
2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
4 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
5 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\
7 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\
8 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\
1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\
3 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\
10 & 0 & 1 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}$$

$$M_{out} = \begin{bmatrix}
2 & 0 & 0 & 0 & 1 \\
4 & 1 & 0 & 0 \\
7 & 0 & 1 & 0 \\
5 & 1 & 1 & 1 \\
1 & 0 & 1 & 0 \\
3 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
6 & 0 & 0 & 0 & 0 & 0 & 1 & 0
\end{bmatrix}$$

where $y_1 = y_1' \lor y_1''$ and $y_3 = y_3' \lor y_3''$ which leads to saving the capacity of matrices $M_{in1}, M_{in2}$ ($M_{out1}, M_{out2}$) compared to the capacity of $M_{in}$ ($M_{out}$).

### 4 Structural model of a dynamic logic object

Let us consider a trivial block diagram of canonic composition of a dynamic logic object – like that in Fig. 3, where the substitute $\Sigma$ [5] is a parallel register consisting of “memory” modules $M_k$ (Fig. 3) modeled with finite automata $\{0, 1\}^k, \delta_k$ where $\delta_k$ are the transition functions $\delta_k : \{0, 1\}^k \times \{0, 1\}^k \rightarrow \{0, 1\}^k , (q_k, \epsilon_k \in \Sigma_k) \rightarrow q_k'$. where $q_k$ and $q_k'$ are the affiliate state and the substitute state, respectively.

![Fig. 3: “Memory” module $M_k$](image)

Let $[M] = M$ or $[\emptyset] = [m] = m \lor \emptyset$, where $M$ is a set and $m$ its element ($m \in M$). The finite automaton model or the given binary dynamic object is to be an ordered sextuple $\{0, 1\}^m \times \{0, 1\}^h, \delta, \Delta, \Lambda\}$ where the vector of transitional function $\Delta \in \{0, 1\}^m \times \{0, 1\}^h$ and the output function are...
Let $M_{in} (M_{out} = [M_{ext}, M_{int}])$ be the input (output, output external, output internal) matrix, (respectively) of the given dynamic object, i.e.,

$$M_{in} : \{1, \ldots, |K|\} \times \{1, \ldots, 2m\} \times \{1, \ldots, 2p\} \rightarrow \{0, 1\}^k : \begin{cases} (r, s, t) \mapsto 0 & \text{for } x_0^s, q_{0^t}^t \notin K_{ij}, \\ (r, s, t) \mapsto 1 & \text{for } x_0^s, q_{0^t}^t \in K_{ij} \end{cases}$$

$$M_{ext} : \{1, \ldots, |K|\} \times \{1, \ldots, 2n\} \rightarrow \{1, \ldots, 2n\} \rightarrow \{0, 1\}^k : \begin{cases} (r, s) \mapsto 0 & \text{for } K_{ij} \notin ndf(\lambda_j), \\ (r, s) \mapsto 1 & \text{for } K_{ij} \notin ndf(\lambda_j) \end{cases}$$

$$M_{int} : \{1, \ldots, |K|\} \times \{1, \ldots, 2p\} \rightarrow \{0, 1\}^k : \begin{cases} (r, t) \mapsto 0 & \text{for } K_{ij} \notin ndf(E_{12k}), \\ (r, t) \mapsto 1 & \text{for } K_{ij} \notin ndf(E_{12k}) \end{cases}$$

If in addition, $Q^T : \{1, \ldots, 2p\} \rightarrow Q : t \rightarrow q_t$ for $t = 2k + 1$, $t \mapsto \tilde{q}_t$ for $t = 2k(k = 0, 1, \ldots, p - 1)$ for the logic array $M$ of the given dynamic object there holds

$$Q_{11} = \left[ M_{in} \& \& \frac{X^T}{Q} \right] \lor \& M_{out}^{ext},$$

$$\tilde{Q}_{11} = \left[ M_{in} \& \& \frac{X^T}{Q} \right] \lor \& M_{out}^{ext}.$$
Table 1: a) the operating table, b) the modified operating table from Example 7

<table>
<thead>
<tr>
<th>qk</th>
<th>q′k</th>
<th>Dk</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Rk Sk Dk Lk Jk Kk Tk
| 1  | 1   | 1  | 1  | 1 |
| 0  | 1   | 0  | 1  | 1 |

Table 2: a) flow table, b) operating table, c) table of substitution variables from Example 8

<table>
<thead>
<tr>
<th>q1 q2</th>
<th>y</th>
<th>Rk</th>
<th>Sk</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0</td>
<td>0</td>
<td>S2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 1</td>
<td>0</td>
<td>S1, R2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 0</td>
<td>0</td>
<td>R1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

According to Table 1b) we obtain

\[ y = \overline{q1} \overline{q2} x \lor \overline{q1} q2 x \lor q1 \overline{q2} x \]

as well as

\[ R_1 = q1 \overline{q2} \]
\[ R_2 = \overline{q1} q2 x \]
\[ S_1 = \overline{q1} q2 x \]
\[ S_2 = \overline{q1} \overline{q2} x \lor q1 q2 x \]

Hence

\[ x \overline{x} q1 \overline{q1} q2 \overline{q2} \]
\[ 0 1 0 1 0 1 \overline{q1} \overline{q2} x \]
\[ 0 1 0 1 1 0 \overline{q1} q2 x \]
\[ M_{im} = 1 0 0 1 1 0 \overline{q1} q2 x \]
\[ 1 0 1 0 0 0 q1 \overline{q2} x \]
\[ 0 0 1 0 0 1 q1 q2 \]

Table 3: Determining parameters of a PLA

<table>
<thead>
<tr>
<th>y</th>
<th>R1</th>
<th>S1</th>
<th>R2</th>
<th>S2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>S2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>S1, R2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>R1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Since the number of logical array input ports is usually markedly higher than the number of rows of input as well as exciting columns, the area of the input matrix \(M_{im}\) is unnecessarily large. Replace, therefore, the input variables by suitable substitutes.

Example 8: Given a synchronous logic object (Table 2a) by an operating table (Table 2b), create a table with a minimal number of columns denoted by substitution variables of the alphabet state. The alphabet state coding [7] transferred on synchronous logic objects [2] is used, and if the elementary substitute will be a PLA, we obtain a(1) = 0 for the state and the state is unnecessary large. Replace, therefore, the input variables by suitable substitutes.

Example 8: Given a synchronous logic object (Table 2a) by an operating table (Table 2b), create a table with a minimal number of columns denoted by substitution variables of the alphabet state. The alphabet state coding [7] transferred on synchronous logic objects [2] is used, and if the elementary substitute will be a PLA, we obtain a(1) = 0 for the state and the state is unnecessary large. Replace, therefore, the input variables by suitable substitutes.

\[ s_1 = a(b(1) \lor b(2)) \lor d( (p(3) \lor p(5)) \lor e \lor b \lor b(2), s_3 = e, \]

where \( p: \{1\}^{10} \rightarrow \{0,1\}: q \rightarrow 0 \) if the object is not in the state \( q \), in the opposite case \( q \rightarrow 1 \). “Substitution” variables define on the alphabet state \( \{1, 2, 3, 4, 5, 6\} \) partition \( \{1, 2\}, \{3, 5\}, \{4, 6\} \) (the permutations of row elements of Table 2c can also helpful). If Miller’s state coding [7] transferred on synchronous logic objects [2] is used, and if the elementary substitute will be a PLA, we obtain a(1) = 0 for the state.

Table 2: a) flow table, b) operating table, c) table of substitution variables from Example 8

<table>
<thead>
<tr>
<th>q</th>
<th>x</th>
<th>y</th>
<th>q′</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>a</td>
<td>a</td>
<td>2</td>
</tr>
<tr>
<td>c</td>
<td>b</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>e</td>
<td>γ</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>a</td>
<td>β</td>
<td>2</td>
</tr>
<tr>
<td>b</td>
<td>δ</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>d</td>
<td>α</td>
<td>–</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>b</td>
<td>α</td>
<td>–</td>
</tr>
<tr>
<td>d</td>
<td>β</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>c</td>
<td>α</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>a</td>
<td>β</td>
<td>–</td>
</tr>
<tr>
<td>d</td>
<td>β</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>c</td>
<td>γ</td>
<td>5</td>
</tr>
<tr>
<td>e</td>
<td>α</td>
<td>–</td>
<td></td>
</tr>
</tbody>
</table>
b)  

<table>
<thead>
<tr>
<th>( q_1 )</th>
<th>( q_2 )</th>
<th>( q_3 )</th>
<th>( s )</th>
<th>( s_1 )</th>
<th>( s_2 )</th>
<th>( s_3 )</th>
<th>( D_k )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>( D_3 )</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>( D_3 )</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>( D_2, D_3 )</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>( D_3 )</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>( D_2 )</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>( 0 )</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>( D_3 )</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>( D_2 )</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>( 0 )</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>( D_3 )</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>( D_2 )</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>( 0 )</td>
</tr>
</tbody>
</table>

Weights \( w(1) = 0, w(3) = w(6) = 1, w(2) = w(4) = 2, w(5) = 3 \)
\( w(\bar{q}) = \chi(\bar{q}) \), where \( \chi(\bar{q}) \) is the characteristic function of the
set \( \{\bar{q} \backslash \chi(\bar{q})\} \) of the state binary code
\( \{1, 2\} \to \{0, 1\}^3: 5 \mapsto 000, 3 \mapsto 100, 6 \mapsto 011, 4 \mapsto 010, \)
\( 2 \mapsto 001, 1 \mapsto 101, \)
which is a reasonable compromise:

\[
\begin{array}{c|cccc}
q_1 & 5 & 4 & 6 & 2 \\
\hline
q_2 & & & & \\
q_3 & 3 & - & - & 1
\end{array}
\]

Let the input binary code be
\( \{a, b, c, d\} \to \{0, 1\}^3: a \mapsto 000, b \mapsto 001, c \mapsto 010, \)
\( d \mapsto 011, \)
\( \bar{e} \mapsto 100 \)
i.e.:
\( s_1 = \bar{x}_1 \bar{x}_2 \bar{x}_3 q_2 q_3 \vee \bar{x}_1 x_2 x_3 \bar{q}_2 \bar{q}_3 \)
\( s_2 = \bar{x}_1 x_2 \bar{x}_3 q_2 q_3 \vee \bar{x}_1 x_2 x_3 q_2 \bar{q}_3 \vee \bar{x}_1 \bar{x}_2 x_3 \bar{q}_1 q_2 q_3; \)

\[
\begin{pmatrix}
  x_1 & x_2 & x_3 & \bar{x}_1 & \bar{x}_2 & \bar{x}_3 & q_1 & \bar{q}_1 & q_2 & \bar{q}_2 & q_3 & \bar{q}_3 \\
\end{pmatrix}
\]

\[
\begin{pmatrix}
  \bar{x}_1 & \bar{x}_2 & \bar{x}_3 & q_2 & q_3 \vee \bar{x}_1 x_2 x_3 \bar{q}_2 \bar{q}_3 \vee \bar{x}_1 \bar{x}_2 \bar{x}_3 q_1 q_2 q_3 \vee \bar{x}_1 \bar{x}_2 x_3 \bar{q}_1 q_2 q_3 \\
\end{pmatrix}
\]

\[
\begin{pmatrix}
  \bar{x}_1 & \bar{x}_2 & \bar{x}_3 & q_2 & q_3 \vee \bar{x}_1 x_2 x_3 \bar{q}_2 \bar{q}_3 \vee \bar{x}_1 \bar{x}_2 \bar{x}_3 q_1 q_2 q_3 \vee \bar{x}_1 \bar{x}_2 x_3 \bar{q}_1 q_2 q_3 \\
\end{pmatrix}
\]

hence
\[
[ y_1, y_2 ] = \begin{pmatrix}
  0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\
  0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\
  0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\
  1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\
  0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\
\end{pmatrix}
\]
Let us code the responses according to Miller again. This can also be used with advantage in coding reactions of asynchronous objects:

\[
\{\alpha, \beta, \gamma, \delta\} \rightarrow \{0,1\}^2:
\]

\[
: \beta \mapsto 00, \alpha \mapsto 01, \gamma \mapsto 10, \delta \mapsto 11,
\]

since the response weights are \(w(\beta) = 5, w(\alpha) = 4, w(\gamma) = 3\) and \(w(\delta) = 1\). For excitation and reactions we can write:

\[
D_1 = \overline{q_1} q_2 \overline{q}_3 s_2
\]

\[
D_2 = (q_1 \overline{q}_2 q_3 \lor \bar{q}_1 q_2 q_3)s_2 \lor \bar{q}_1 \overline{q}_2 \overline{q}_3 s_1 = \bar{q}_2 q_3 s_2 \lor \bar{q}_1 \overline{q}_2 \overline{q}_3 s_1
\]

\[
D_3 = q_1 \overline{q}_2 q_3 s_1 \lor q_1 \overline{q}_2 q_3 s_2 \lor \bar{q}_1 \overline{q}_2 \overline{q}_3 s_1 = \bar{q}_2 q_3 s_1 \lor q_1 \overline{q}_2 q_3 s_2
\]

\[
\eta_1 = (q_1 \bar{q}_2 q_3 \lor \bar{q}_1 \overline{q}_2 q_3 \lor \bar{q}_1 q_2 \overline{q}_3 \lor \bar{q}_1 \bar{q}_2 q_3)s_2 = (\bar{q}_2 q_3 \lor \bar{q}_1 \bar{q}_2 q_3)s_2
\]

\[
\eta_2 = q_1 \overline{q}_2 q_3 s_1 \lor (\bar{q}_1 \overline{q}_2 q_3 \lor \bar{q}_1 \bar{q}_2 \overline{q}_3 \lor \bar{q}_1 q_2 q_3)s_2 = q_1 \overline{q}_2 q_3 s_1 \lor (\bar{q}_1 q_3 \lor \bar{q}_1 \overline{q}_2 \overline{q}_3)s_2
\]

Hence

\[
\begin{bmatrix}
\eta_1, \eta_2, D_1, D_2, D_3
\end{bmatrix} = \begin{bmatrix}
q_1 & \bar{q}_1 & q_2 & \bar{q}_2 & q_3 & \bar{q}_3 & s_1 & \bar{s}_1 & s_2 & \bar{s}_2 \\
0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\
1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\
1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \\
0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \\
0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0
\end{bmatrix}
\]

\[
\begin{bmatrix}
\eta_1 & \eta_2 & D_1 & D_2 & D_3 \\
0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 \\
1 & 0 & 0 & 1 & 0 \\
0 & 1 & 0 & 1 & 0 \\
1 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 1
\end{bmatrix}
\]

**Example 9:** Given an asynchronous logic object (Table 3a) by an operating table (Table 3b), create a table with a minimal number of columns denoted by “substitute” variable \(\sigma_i\) (\(i = 1, 2, 3\)) – (Table 3c). Since \(\sigma_1 = a, \sigma_2 = b\) and \(\sigma_3 = c\), let us discuss just the parallel coding of states \(\{q\}_{q=1}^6 \rightarrow \{0,1\}^5\) according to Liu [2]. In the case of single stimuli \(a, b, c\), the noncritical pairs of state transitions on the state alphabet \(\{q\}_{q=1}^6\) define

\[
\begin{bmatrix}
\xi_1, \xi_2, x_2, x_3, \bar{x}_3 \\
0 & 1 & 1 & 0 & 0 \\
1 & 0 & 0 & 1 & 0 \\
0 & 1 & 0 & 1 & 0 \\
1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0
\end{bmatrix}
\]

\[
\begin{bmatrix}
\xi_1, \xi_2, x_2, x_3, \bar{x}_3 \\
1 & 0 & 1 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 1
\end{bmatrix}
\]

Hence for the separation state components [2] we obtain a state code (Table 3d), again with an obvious effort to achieve the least possible number (Table 3e) of state components \(q\) \((j = 1, 2, 3)\) and the least possible \(w_H(D_j)\) exciting weight \(D_j\).
Table 3: a) transition table, b) operating table, c) table of input "substitute" variables, d) state code table, e) state code table with a minimum state code words from Example 9

### a) Transition Table

<table>
<thead>
<tr>
<th>q</th>
<th>a</th>
<th>b</th>
<th>c</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

### b) Operating Table

<table>
<thead>
<tr>
<th>(q_1 q_2 q_3)</th>
<th>(x_1 x_2)</th>
<th>(q_1 q_2 q_3)</th>
<th>(D_1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>-</td>
</tr>
<tr>
<td>0 1 0</td>
<td>1 0 1 0</td>
<td>0 1 0</td>
<td>(D_3)</td>
</tr>
<tr>
<td>0 0 1</td>
<td>1 0 1 1</td>
<td>0 0 1</td>
<td>0 0 1</td>
</tr>
<tr>
<td>0 1 1</td>
<td>1 0 1 1</td>
<td>0 0 1</td>
<td>0 0 1</td>
</tr>
<tr>
<td>1 0 0</td>
<td>1 0 0 0</td>
<td>0 0 1</td>
<td>0 0 1</td>
</tr>
<tr>
<td>1 0 1</td>
<td>1 0 0 1</td>
<td>0 1 0</td>
<td>0 0 1</td>
</tr>
</tbody>
</table>

### c) Table of Input "Substitute" Variables

<table>
<thead>
<tr>
<th>q</th>
<th>(\sigma_1)</th>
<th>(\sigma_2)</th>
<th>(\sigma_3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>a</td>
<td>b</td>
<td>c</td>
</tr>
<tr>
<td>2</td>
<td>a</td>
<td>b</td>
<td>c</td>
</tr>
<tr>
<td>3</td>
<td>a</td>
<td>b</td>
<td>c</td>
</tr>
<tr>
<td>4</td>
<td>a</td>
<td>b</td>
<td>c</td>
</tr>
<tr>
<td>5</td>
<td>a</td>
<td>b</td>
<td>c</td>
</tr>
<tr>
<td>6</td>
<td>a</td>
<td>b</td>
<td>c</td>
</tr>
</tbody>
</table>

### d) State Code Table

<table>
<thead>
<tr>
<th>x</th>
<th>a</th>
<th>b</th>
<th>c</th>
</tr>
</thead>
<tbody>
<tr>
<td>q</td>
<td>(q_1)</td>
<td>(q_2)</td>
<td>(q_3)</td>
</tr>
<tr>
<td>1</td>
<td>1 0 0 0 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0 0 1 0 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0 0 1 0 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0 0 1 0 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>1 1 1 1 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>1 1 1 1 1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### e) State Code Table with Minimum State Code Words

<table>
<thead>
<tr>
<th>q</th>
<th>(q_1)</th>
<th>(q_2)</th>
<th>(q_3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1 0 0 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0 0 1 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0 0 1 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0 1 0 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>0 1 0 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>0 1 0 0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
5 Conclusions

In [8], we find exact, theoretically demanding and quite elaborate optimization algorithms both according to state coding (Miller’s economic coding and Liu’s parallel coding) and by the number rows of matrices of the matrix structural model of the dynamic logic object.

If the number of states of the dynamic object is large, it is suitable to apply a simple block decomposition [2, 9] and design the structural model of the matrix as a composition of matrix structural models of individual blocks.

References


© Czech Technical University in Prague

http://ctn.cvut.cz/ap/