Design and use of a universal logic circuit

Mahendrakumar Punjalal Shah

Follow this and additional works at: http://scholarsmine.mst.edu/masters_theses
Department: Electrical and Computer Engineering

Recommended Citation
DESIGN AND USE OF A UNIVERSAL LOGIC CIRCUIT

By

MAHENDRAKUMAR PUNJALAL SHAH, 1944-

A

THESIS

submitted to the faculty of

THE UNIVERSITY OF MISSOURI-ROLLA

in partial fulfillment of the requirements for the

Degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

Rolla, Missouri

1969

Approved by

(Advisor)
ABSTRACT

Yau and Tang have designed universal logic circuits (ULC's) of three variables with 8 I/O pins. They assumed that only one variable was available as a free input variable in its true and complementary form, the others were available as fixed input variables and the circuit generated true and complementary outputs. Their design required 22 I/O pins for a four-variable ULC, constructed from three variable ULCs. This paper described an algorithm for determining the input-pin connections, directly from the K-map of a given output function, for any n-variable ULC.

Forslund and Waxman have designed a three-variable ULC using the theory of equivalence classes. They assumed that all the variables and their complements were available as inputs, and that they generated true and complementary outputs. With this assumption they required 7 I/O pins for a ULC of three variables. Using the same theory and assumptions as stated above, this paper describes the design of a four-variable ULC from three-variable ULCs. This paper also describes the development of the circuit for realizing any n-variable function.
ACKNOWLEDGEMENTS

The author wishes to express his sincere thanks to Dr. Stephen A. Szygenda for help and guidance given. He also expresses his grateful thanks to Dr. James H. Tracey for suggestions and for the continuous guidance given throughout the preparation of this thesis.

Appreciation is also extended to Connie Hendrix for her typing efforts.
TABLE OF CONTENTS

Abstract ........................................... ii
Acknowledgements ................................. iii
Table of Contents ................................. iv
List of Figures .................................. v
List of Tables ................................. vi

I. Introduction .................................. 1

II. Use of the K-Map to Find Input Pin Connections for ULC's .......................... 3
   A) ULC of Three Variables .................. 3
   B) ULC of Four Variables Using Three-Variable ULCs .......................... 5
   C) An Algorithm for Input-Pin Connections Directly from the K-Map  .......... 8

III. Design of a Universal Logic Circuit Using the Theory of an Equivalence Classes 23
    A) Equivalence Classes of Three Variables .......................... 23
    B) Design of a Three-Variable ULC .......................... 24
    C) Design of a Four-Variable ULC Using Two Three-Variable ULCs  ......... 35

IV. Conclusion .................................. 43

Bibliography .................................... 46
Vita .............................................. 47
### LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Three-Variable ULC (6 input pins)</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>K-Map of Three-Variable Function</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>Four-Variable ULC Constructed From Three-Variable ULCs. (16 input pins)</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>K-Map of Four-Variable Function</td>
<td>7</td>
</tr>
<tr>
<td>5</td>
<td>Synthesis of Three-Variable Functions From Two-Variable Functions</td>
<td>25</td>
</tr>
<tr>
<td>6</td>
<td>Synthesis of Four-Variable Functions From Two-Variable Functions</td>
<td>26</td>
</tr>
<tr>
<td>7</td>
<td>K-Map of Four-Variable Function (nine out of ten equivalence classes)</td>
<td>27</td>
</tr>
<tr>
<td>8</td>
<td>K-Map of Five-Variable Function (ten out of ten equivalence classes)</td>
<td>28</td>
</tr>
<tr>
<td>9</td>
<td>Three-Variable ULC (5 input pins)</td>
<td>31</td>
</tr>
<tr>
<td>10</td>
<td>Representative Functions of the Ten Equivalence Classes</td>
<td>33</td>
</tr>
<tr>
<td>11</td>
<td>Block Diagram of Four-Variable ULC</td>
<td>36</td>
</tr>
<tr>
<td>12</td>
<td>Four-Variable ULC Constructed From Three-Variable ULCs (13 input pins)</td>
<td>38</td>
</tr>
<tr>
<td>13</td>
<td>Implementation of Five-Variable ULC</td>
<td>41</td>
</tr>
</tbody>
</table>
### LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Ten Equivalence Classes of Three-Variable Functions</td>
<td>24</td>
</tr>
<tr>
<td>2</td>
<td>Pin Connections for Three-Variable (5 input pin) ULC</td>
<td>32</td>
</tr>
</tbody>
</table>
A logic element is said to be a universal logic circuit if it can realize any function of a fixed number of variables by a mere variation of input-pin connections. Logic design with universal elements requires only one element for any function of a fixed number of variables for which that element is designed. Thus, using a universal element for the design of any function, one needs to change only the input connection pattern of the element, without making any changes to the logic circuit itself.

A single ULC of a specific number of variables can be used to implement any function of that particular number of variables, and less, but not more. If, for example, a ULC is designed for 'N' variables, it will not only realize n-variable functions, but also, realize any function of less than n variables.

It has been assumed that one input is available as a free input, in its true and complementary form, and the rest are available as fixed inputs. It will be assumed in this paper that both true and complementary outputs of the function are available. With these assumptions the design of a three-variable ULC with 8 I/O pins is described in Chapter II. In the same chapter, techniques are also given for the use of three-variable ULCs to design four-variable ULC. It may be stated here that 22 I/O pins are
required for this purpose. In addition, a new technique is
given to determine the input-pin connections directly from
the K-map of a given output function, for any n-variable
ULCs.

In Chapter III it is assumed that all variables and
their complements are available as inputs. It is also
assumed that for any output function true and complementary
outputs are available. With these assumptions Forslund and
Waxman(2) partitioned 256 logic functions of three variables
into 10 equivalence classes. They utilized the above data
and designed a ULC of three variables with 7 I/O pins. A
design for a four-variable ULC using three-variable ULCs
with 19 I/O pins is presented in the same chapter. This
ULC of four variables is compared with the ULC of four
variables designed by Yau and Tang described in Chapter II.
The above new design shows an advantage over that designed
by Yau and Tang, in that it requires a lesser number of
I/O pins. But it does have the disadvantage of being more
complex than the one designed by Yau and Tang. In the same
chapter a new technique is given to develop a circuit which
can realize any function of n variables with two and three-
variable ULCs.
A) **ULC of Three Variables**

It is known that a logic function $F(X,Y,Z)$ of three variables $X,Y,Z$ can be expanded with respect to any two of the three variables $X,Y,Z$ as follows:

$$F(X,Y,Z) = \overline{X\overline{Y}} F(0,0,Z) + \overline{X\overline{Y}} F(0,1,Z)$$
$$+ X\overline{Y} F(1,0,Z) + X Y F(1,1,Z)$$

The implementation of Equation (1) is shown in Figure 1, where $I_0$, $I_1$, $I_2$ and $I_3$ are the functions $F(0,0,Z)$, $F(0,1,Z)$, $F(1,0,Z)$ and $F(1,1,Z)$, respectively.

![Diagram of a three-variable ULC with 6 input pins](image.png)

**Figure 1**: Three-Variable ULC (6 input pins)
The circuit shown in Figure 1 can realize any arbitrary three-variable logic function, \( F(X,Y,Z) \), if the terminals \( C_1 \) and \( C_2 \) are connected to fixed available inputs \( X \) and \( Y \), respectively, and the 4 terminals \( R_0, R_1, R_2, R_3 \) are connected to either \( Z, \overline{Z}, '0' \) or \( '1' \). The terminals which are connected to the fixed input variables are designed **side terminals** and those which are connected to available free inputs are designed **front terminals**. In Figure 1, 8 I/O pins are required for a ULC of three variables with the assumption that both true and complementary outputs of the functions are available. An example is given to show the method of determining the input connections of 4 front terminals, when the output function of three variables is given.

**Example**

Let \( F(X,Y,Z) = \text{Im}(2,4,5) \)

From the given output, the K-map of three variables can be drawn as shown in Figure 2.

![K-Map of Three-Variable Function](image)

**Figure 2**

K-Map of Three-Variable Function

Based on the above map (Figure 2) the output function can be written as: \( F = \overline{X} \overline{Y} \overline{Z} + X \overline{Y} \ldots \ldots \) (1A)
The proper connections of the front terminal of Figure 1 for the given output function can be found in the following way:

\[ F = \overline{X} Y \overline{Z} + X \overline{Y} \ldots \ldots \]  \hspace{1cm} (1A)

Parameter \( I_0 \) is defined as \( F(0,0,Z) \). Substituting \( X = 0 \) and \( Y = 0 \) in Equation 1A,

\[ I_0 = 1 \cdot 0 \cdot \overline{Z} + 0 \cdot 1 \]

Then

\[ I_0 = 0 \]

\[ \therefore I_0 \] is connected to logical 0. Similarly,

\[ I_1 = \overline{Z} \]

\[ I_2 = 1 \]

and

\[ I_3 = 0. \]

(B) \hspace{0.5cm} ULC of Four Variables Using Three-Variable ULCs

When only three-variable ULCs are available to realize a four-variable ULC, a slight modification becomes necessary as shown in Figure 3 and as explained below.

The logic function of four variables is expanded as follows:

\[ F(X,Y,Z,W) = \overline{X} F(0,Y,Z,W) + X F(1,Y,Z,W) \]  \hspace{1cm} (2)

It can easily be seen that the above equation can be realized by ULC-3, provided the side terminals \( C_1 \) and \( C_2 \) are both connected to the input variable \( X \), and the front terminals \( A_0 \) and \( A_3 \) are connected to the residue functions
Figure 3: Four Variable ULC Constructed From Three Variable ULCs (16 input pins)
F(O,Y,Z,W) and F(1,Y,Z,W), respectively. In the Figure it can be noted that it requires 8 I/O pins to realize the three-variable ULC and 22 I/O pins to realize the four-variable ULC, with the assumption that both true and complementary output functions are available in each case.

Example 2 explains the method of finding out the values of 8 front terminals for a ULC of four variables, if an output function of four variables is given.

Example 2

Let \( F(X,Y,Z,W) = \text{Im}(2,4,5,6,9,10,11,13) \)

The K-map of the above function is shown in Figure 4:

![K-Map of Four-Variable Function](image)

From the above K-map the function can be written as:

\[
F = \overline{X} Z \overline{W} + \overline{X} Y \overline{Z} + X \overline{Z} W + X \overline{Y} Z
\]  

(3)

Parameter \( I_0 \) is defined as \( F(0,0,0,W) \). Substituting \( X = 0, Y = 0 \) and \( Z = 0 \) in Equation 3,

\[
I_0 = I \cdot 0 \cdot \overline{W} + 1 \cdot 0 \cdot 1 + 0 \cdot 1 \cdot W + 0 \cdot 1 \cdot 0
\]

Then \( I_0 = 0 \)
\[ \text{I}_0 \text{ is connected to logical 0.} \]

Similarly,
\[
\begin{align*}
\text{I}_1 &= \overline{\text{W}} \\
\text{I}_2 &= 1 \\
\text{I}_3 &= \overline{\text{W}} \\
\text{I}_4 &= \text{W} \\
\text{I}_5 &= 1 \\
\text{I}_6 &= \text{W} \\
\text{I}_7 &= 0
\end{align*}
\]

Thus, from the above procedure the locations of the connections for the 8 front terminals of the ULC represented in Figure 3 are known.

(C) An Algorithm for Input-Pin Connections Directly From the K-Map

Taking the case of a ULC of a fixed number of variables, as the side terminals of its circuit are always connected to fixed input variables, the locations of these connections do not present a problem. For example in the case of a three-variable ULC 2 side terminals \( C_1 \) and \( C_2 \) are usually connected to \( X \) and \( Y \), respectively, independent of the value of the output function, while the front terminals are always dependent on the value of the output functions. In the latter case the values are always any of 4 possibilities, that is, (1) logical 0, (2) logical 1, (3) assertion of free variable and (4) negation of free variable.
As shown in Examples 1 and 2, to find the appropriate value of the front terminals for any number of variable ULC, the following procedure was followed. First, the given output function was simplified. After that, the different fixed variables values were substituted in the simplified equation according to the required terminal connection value. This procedure, as can be seen, is very laborious and time consuming. A procedure has been devised for detecting the appropriate values of the front terminals directly from the K-map of a given output function \(^{(8)}\).

Logic function, \(F(X,Y,Z)\) of three variables \(X,Y,Z\), can always be expanded with respect to any two of the three variables as follows:

\[
F(X,Y,Z) = \overline{X} \overline{Y} F(0,0,Z) + \overline{X} Y F(0,1,Z) + X \overline{Y} F(1,0,Z) + X Y F(1,1,Z) \quad (4)
\]

where

\[
\begin{align*}
I_0 &= \overline{Z} F(0,0,0) + Z F(0,0,1) \\
I_1 &= \overline{Z} F(0,1,0) + Z F(0,1,1) \\
I_2 &= \overline{Z} F(1,0,0) + Z F(1,0,1) \\
I_3 &= \overline{Z} F(1,1,0) + Z F(1,1,1)
\end{align*}
\]

\(I_0, I_1, I_2 \) and \(I_3\) are functions of free variables only, (in this case \(Z\)), and each of these functions assume 1 of the 4 possibilities, \(0,1,Z\) and \(\overline{Z}\).
The K-map of three variable function is shown.

\[
\begin{array}{c|cccc}
  & 00 & 01 & 11 & 10 \\
\hline
  0 & 0 & 2 & 6 & 4 \\
  1 & 1 & 3 & 7 & 5 \\
\end{array}
\]

From Equation 6 and the K-map of a three-variable function shown above, it is seen that

\[
\begin{align*}
  I_0 &= \overline{Z} F(0) + Z F(1) \\
  I_1 &= \overline{Z} F(2) + Z F(3) \\
  I_2 &= \overline{Z} F(4) + Z F(5) \\
  I_3 &= \overline{Z} F(6) + Z F(7)
\end{align*}
\]

Equation 7 shows that if both the index number values of a particular connection are 0, the values of those functions are also 0. In this case the particular front terminal is considered to be "biased to logical 0". Accordingly, an identically similar situation arises when both the index number values are 1. Consider the case of a combination of index numbers 1 and 0. When this occurs either assertion of
the free variable i.e. \( Z \) or the negation of the free variable i.e. \( \overline{Z} \) is connected to the input front terminal. If an even index number of a connection has a value 1, the negation of the free variable implements the function. If, on the other hand, an odd index number has a value 1, the assertion of the free variable implements the function.

Consider the case of \( n \)-variable functions. Logic function \( F(X_1, X_2, \ldots, X_m, \ldots, X_n) \) of \( n \) variables \( X_1, X_2, \ldots, X_m, \ldots, X_n \) can always be expanded with respect to any \( n-1 \) variables of the above \( n \) variables as follows:

\[
F(X_1, X_2, \ldots, X_m, \ldots, X_n) = \overline{X}_1 \overline{X}_2 \ldots \overline{X}_m \ldots \overline{X}_{n-1} F(0, 0, \ldots, 0, x_n) + \overline{X}_1 \overline{X}_2 \ldots \overline{X}_m \ldots \overline{X}_{n-2} x_{n-1} \ldots F(0, 0, \ldots, 0, 0, \ldots, 0, 1, x_n) + \ldots + \overline{X}_1 \overline{X}_2 \ldots \overline{X}_m \ldots \overline{X}_{n-1} F(0, 0, \ldots, 0, 1, \ldots, 0, x_n) + x_1 x_2 \ldots x_m \ldots x_{n-1} F(1, 1, \ldots, 1, x_n) \tag{8}
\]

\[
F(X_1, X_2, \ldots, X_m, \ldots, X_n) = \overline{X}_1 \overline{X}_2 \ldots \overline{X}_m \ldots \overline{X}_{n-1} I_0 + \overline{X}_1 \overline{X}_2 \ldots \overline{X}_m \ldots \overline{X}_{n-2} x_{n-1} I_1 + \ldots + \overline{X}_1 \overline{X}_2 \ldots \overline{X}_m \ldots \overline{X}_{n-1} I_m + \ldots + x_1 x_2 \ldots x_m \ldots x_{n-1} I_{2^{n-1}-1} \tag{9}
\]
where

\[ I_0 = \overline{X}_n F(0, 0, \ldots, 0, \ldots, 0, 0) + X_n F(0, 0, \ldots, 0, \ldots, 0, 1) \]

\[ I_1 = \overline{X}_n F(0, 0, \ldots, 0, \ldots, 0, 1, 0) + X_n F(0, 0, \ldots, 0, \ldots, 0, 1, 1) \]

\[ \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \]

\[ I_m = \overline{X}_n F(0, 0, \ldots, 1, \ldots, 0, 0) + X_n F(0, 0, \ldots, 1, \ldots, 0, 1) \]

\[ \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \]

\[ I_{2^n-1-1} = \overline{X}_n F(1, 1, \ldots, 1, \ldots, 1, 0) + X_n F(1, 1, \ldots, 1, \ldots, 1, 1) \]

(10)

From Equation 10,

\[ I_0 = \overline{X}_n F(0) + X_n F(1) \]

\[ I_1 = \overline{X}_n F(2) + X_n F(3) \]

(11)

\[ \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \]

\[ I_m = \overline{X}_n F(2m) + X_n F(2m+1) \]

\[ \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \]

\[ I_{2^n-1-1} = \overline{X}_n F(2(2^n-1)) + X_n F(2(2^n-1) + 1) \]

Each front terminal connection is made up of 2 consecutive pairs of even and odd index numbers indicated in the K-map of that particular n-variable function, while still retaining its consecutive character in increasing order.

The conditions here are similar to that of the three-variable
function. It can be seen from Equation 10 that $X_n$ is the free variable. The values of the free variable for each connection are in increasing order, i.e. of 2 values, of the free variable, 0 and 1, the value 0 should be in the first index number, and the value 1 should be in the second index number of each connection. Equation 11 shows that if both the index number values of a particular connection are 0, the values of those function are also 0. In a case like this that particular front terminal is considered to be biased to logical 0. Accordingly, an identically similar situation arises when both the index number values are 1.

Take the case of a combination of index numbers 1 and 0. When this occurs either the assertion or the negation of free variable is connected to input front terminal. If an even index number of a connection has a value 1 the negation of the free variable implements the function. If, on the other hand, an odd index number has a value 1, the assertion of the free variable implements the function. Based on the above discussion, a procedure is described below to detect front terminal connection values for any n-variable ULC.

First, the K-map of the n-variable function and the connection K-map involving n-1 variables are drawn. These maps are constructed as described below.

The number of columns in the connection K-map are equal to the number of columns in the function K-map and
the number of rows in the connection K-map are half the number of rows in the function K-map. For example, for a four-variable function, the function K-map of four variables and the connection K-map of three variables are drawn. The number of columns in the connection K-map and in the function K-map are four each. The number of rows in the function K-map is four while that in the connection K-map is two.

Second, one divides the even index numbers of the function K-map by two and arranges the resulting numbers in the corresponding blocks of the connection K-map. This gives the possible front terminal connections of the circuit. In the above example, the even index numbers of the function K-map are 0, 2, 4, 6, 8, 10, 12 and 14. Dividing these numbers by 2 the resulting numbers are 0, 1, 2, 3, 4, 5, 6, and 7, respectively. Arrangement of these resulting numbers in the corresponding blocks of the connection K-Map will give the front terminal connections of the circuit.

Third, in the function K-map, if both the index numbers, which constitute a front terminal connection, are
both 0 or both 1, then that connection is biased to logical 0 or logical 1, respectively. For example, in the four-variable function K-map shown as follows, if the index numbers 4 and 5 are 0 then input terminal \( I_2 \) is biased to logical 0. In the same example, if the index numbers 8 and 9 are 1 then input connection \( I_4 \) is biased to logical 1.

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

Fourth, if the 2 index numbers of the function K-map contains combination of 1 and 0 or 0 and 1, either the assertion or the negation of the free variable is connected to the input. If 1 occupies the block associated with the odd index number, the assertion of a free variable is connected to its associated input. If, on the other hand, 1 occupies the block associated with the even index number the negation of the free variable is connected to that input.

In the above example, the index number 12, of the function K-map has values 1 and that of 13, has value 0. Also, these index numbers constitute a front terminal connection \( I_6 \). Since 1 occupies the block associated with the even index number, the negation of the free variable is connected
to $I_6$. In the same example, blocks associated with the index numbers 2 and 3 contains 0 and 1, respectively. Since 1 is occupied by the block associated with the odd index number, the connection $I_1$ is connected to the assertion of the free variable.

Based on the above procedure a generalized algorithm is presented.

1) For a $n$-variable function the function $K$-map of $n$ variables and the connection $K$-map of $n-1$ variables are drawn.

2) Divide the even index numbers of the function $K$-map by 2. The resulting numbers are the numbered front terminal connections of the circuit.

3) If both the index numbers which constitute a front terminal connection are 0, or 1, then that connection is biased to logical 0, or 1, respectively.

4) If the index numbers which constitute a front terminal connection contain combinations 1 and 0, either the assertion or the negation of a free variable implements the function. If 1 occupies the block associated with the odd index number, the assertion of the free variable is connected to its associated input. If 1 occupies the block associated with the even index number, the negation of the free variable is connected to that input.
The Following Examples are Presented for Clarification

Two Variables

Function K-Map

Connection K-Map

(1) $F(X,Y) = \Sigma m(0,2,3)$

Solution:

Then, $I_0 = Y$

$I_1 = \bar{Y}$

(2) $F(X,Y) = \Sigma m(1,2)$

Solution:

Then, $I_0 = Y$

$I_1 = \bar{Y}$
### Three Variables

#### Function K-Map

<table>
<thead>
<tr>
<th>XY</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>00</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

#### Connection K-Map

- $I_0 = \overline{Z}$
- $I_1 = Z$
- $I_2 = 1$
- $I_3 = 0$

#### (3) $F(X,Y,Z) = \Sigma m(0,3,4,5)$

#### (4) $F(X,Y,Z) = \Sigma m(1,2,5,6)$
Four Variables

(5) \( F(X, Y, Z, W) = \Sigma m(0, 2, 3, 5, 6, 8, 11, 13, 14) \)

(6) \( F(X, Y, Z, W) = \Sigma m(1, 2, 4, 5, 7, 8, 12, 13, 15) \)
\[ \begin{array}{cccc}
X & Y \\
0 & 1 & 1 & 1 \\
1 & 1 & 1 & 0 \\
0 & 1 & 1 & 0 \\
1 & 0 & 0 & 0 \\
\end{array} \]

\[ I_0 = w \quad I_4 = \overline{w} \]
\[ I_1 = \overline{w} \quad I_5 = 0 \]
\[ I_2 = 1 \quad I_6 = 1 \]
\[ I_3 = w \quad I_7 = w \]

Five Variables

\[
\begin{array}{cccccccccccc}
000 & 001 & 011 & 010 & 110 & 111 & 101 & 100 \\
0 & 4 & 12 & 8 & 24 & 28 & 20 & 16 \\
01 & 1 & 5 & 13 & 9 & 25 & 29 & 21 & 17 \\
11 & 3 & 7 & 15 & 11 & 27 & 31 & 23 & 19 \\
10 & 2 & 6 & 14 & 10 & 26 & 30 & 22 & 18 \\
\end{array}
\]

Function K-Map

Connection K-Map
Function K-Map

Connection K-Map

<table>
<thead>
<tr>
<th>XYZ</th>
<th>Wv</th>
<th>1</th>
<th>1</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- \( I_0 = \overline{v} \)
- \( I_1 = v \)
- \( I_2 = \overline{v} \)
- \( I_3 = \overline{v} \)
- \( I_4 = \overline{v} \)
- \( I_5 = 1 \)
- \( I_6 = v \)
- \( I_7 = 0 \)
- \( I_8 = \overline{v} \)
- \( I_9 = v \)
- \( I_{10} = v \)
- \( I_{11} = v \)
- \( I_{12} = v \)
- \( I_{13} = v \)
- \( I_{14} = v \)
- \( I_{15} = v \)
(8) \( F(X, Y, Z, W, V) = \Sigma m(0, 1, 4, 5, 6, 11, 12, 14, 16, 20, 22, 28, 30, 31) \)

\[
\begin{array}{c|cccc}
Y & Z & W & V & X = 0 \\
\hline
1 & 1 & 1 & 0 & 1 \\
1 & 1 & 0 & 0 & 1 \\
0 & 0 & 0 & 1 & 0 \\
0 & 1 & 1 & 0 & 0 \\
\end{array}
\]

\[
\begin{array}{c|cccc}
Y & Z & W & V & X = 1 \\
\hline
0 & 1 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 0 \\
\end{array}
\]

\[I_8 = 1, \quad I_1 = 0, \quad I_2 = 1, \quad I_3 = \overline{V}, \quad I_4 = 0, \quad I_5 = V, \quad I_6 = \overline{V}, \quad I_7 = \overline{V}\]

\[I_8 = \overline{V}, \quad I_9 = 0, \quad I_{10} = \overline{V}, \quad I_{11} = \overline{V}, \quad I_{12} = 0, \quad I_{13} = 0, \quad I_{14} = \overline{V}, \quad I_{15} = 1\]
A) Equivalence Classes of Three Variables

Two Boolean functions are of the same equivalence class if one of them can be obtained from the other by a process or permutation and combination of input variables. Hellerman partitioned 256 three-variable logic functions into 80 equivalence classes. He assumed that only true variables are available as inputs and that they generate only true outputs. To achieve a reduction of the number of equivalence classes, Forslund and Waxman\(^{(2)}\) assumed that both true and complement variables are available at the input, and that true and complementary logic functions are generated at the output. Biasing (to a logical 1 or 0) and duplication of input variables also enhance this reduction. The equivalence classes, defined in this way, get reduced from 80 to 10 for three-variable logic functions.

Table 1 shows 10 equivalence classes of three-variable functions. To realize each class it requires 3 input pins and 2 output pins, one each for true and complementary output functions.
TABLE 1

TEN EQUIVALENCE CLASSES OF THREE-VARIABLE FUNCTIONS

<table>
<thead>
<tr>
<th>TRUE</th>
<th>COMPLEMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1) $XYZ$</td>
<td>$X + \overline{Y} + \overline{Z}$</td>
</tr>
<tr>
<td>2) $X\overline{Y}Z + X\overline{YZ}$</td>
<td>$X + \overline{Y} \overline{Z} + YZ$</td>
</tr>
<tr>
<td>3) $\overline{XY}Z + XY\overline{Z}$</td>
<td>$XY + \overline{X} \overline{Z} + \overline{Y} \overline{Z}$</td>
</tr>
<tr>
<td>4) $XY + YZ$</td>
<td>$\overline{Y} + \overline{X} \overline{Z}$</td>
</tr>
<tr>
<td>5) $XY\overline{Z} + \overline{XYZ} + X\overline{YZ}$</td>
<td>$XY + \overline{Y} \overline{Z} + \overline{X} \overline{Z} + XYZ$</td>
</tr>
<tr>
<td>6) $XY\overline{Z} + \overline{YZ}$</td>
<td>$XY + YZ + \overline{YZ}$</td>
</tr>
<tr>
<td>7) $XY + YZ + ZX$</td>
<td>$XY + \overline{X} \overline{Z} + \overline{Y} \overline{Z}$</td>
</tr>
<tr>
<td>8) $XY\overline{Z} + \overline{XZ} + \overline{YZ}$</td>
<td>$\overline{YZ} + \overline{X} \overline{Z} + XYZ$</td>
</tr>
<tr>
<td>9) $\overline{XY} + Z\overline{Y}$</td>
<td>$XY + \overline{Y} \overline{Z}$</td>
</tr>
<tr>
<td>10) $XYZ + X\overline{YZ} + \overline{XY}Z + \overline{X}Y\overline{Z}$</td>
<td>$XY\overline{Z} + \overline{XYZ} + X\overline{YZ} + XYZ$</td>
</tr>
</tbody>
</table>

B) Design of a Three-Variable ULC

As seen in Table 1, it requires 3 input pins and 2 output pins in each circuit to realize 1 out of 10 equivalence classes. Accordingly, 10 different logic circuits would be required to realize all the 10 equivalence classes of the three-variable functions mentioned above. But a universal logic circuit can be realized only if the 3 input pin restriction was removed. The 10 functions can then be implemented with 1 logic circuit. In Figure 5, 4 three-variable functions, each having different index numbers are shown, with their 2 possible decompositions into two, two-variable functions.
SYNTHESIS OF THREE-VARIABLE FUNCTIONS FROM TWO-VARIABLE FUNCTIONS

Figure 5

Of the ensemble of two-variable function, 1 of each index number 0, 2, 3 is chosen as one capable of synthesizing many three-variable functions. These are assembled
into a four-variable function. This can be done in 96 ways. This is shown in Figure 6.

Four groups can be arranged as follows:

\[
\begin{array}{c|c|c|c|c}
0 & 1 & 0 & 1 & 0 & 2 \\
3 & 2 & 2 & 3 & 3 & 1 \\
\end{array}
\]

where 0,1,2,3 indicates group number. Each group can be arranged, as shown below.

SYNTHESIS OF FOUR-VARIABLE FUNCTIONS FROM TWO-VARIABLE FUNCTIONS

Figure 6

Thus, total combination could be \(3 \times 1 \times 4 \times 2 \times 4 = 96\)

For example:
The 96 four-variable functions assembled (Figure 6) were studied to see the possible number of equivalence classes that could be available from them. It was seen that a minimum of 6 and a maximum of 9 equivalence classes were available\(^{(2)}\). The missing class in every case was the exclusive-or function. However, as shown in the discussion later, this missing class can be realized by the addition of an input variable in the function shown in Figure 7.

The above description, therefore, clearly indicates that 4 input pins are required to realize 9 out of 10 equivalence classes of three-variable functions. One of the 96 combinations of four-variable functions is shown in the accompanying K-map (Figure 7) where 9 out of 10 equivalence classes of three-variable functions are realized.

Figure 7: (nine out of ten equivalence classes)
By simplifying the above K-map, the following function can be obtained:

\[ F = T \overline{U} + ST \overline{R} + ST \overline{U} + RST \overline{U} + RUST \]  \hspace{1cm} (12)

It was stated in the above paragraph that only a maximum of 9 out of the 10 equivalence classes could be realized by 96 combinations as against the 10 equivalence classes of three variables shown in Table 1. Therefore, Figure 7 stands incomplete due to the missing exclusive-or function. A scrutiny of Figure 7 (the K-map in question) will show that the generation of an additional class can be achieved by the addition of a minterm to the function in Equation 12. This resolves itself into the fact that it requires 1 more input pin, to add the missing class. The resultant K-map, which could realize all the 10 equivalence classes of three variables including an exclusive-or functions is shown in Figure 8.
By simplifying the above K-map (Figure 8) \( F = T \overline{R} + ST \overline{R} + STU + RSTU + RUST + RSTUV \) (13)

The five-variable function, shown in Equation 13, can realize all the 10 equivalence classes of three-variables as shown.

(1) \( F = XYZ \)

By substituting
\( R = X, S = Y, T = X, U = Z \) and \( V = 0 \)

In Equation 13, \( F = X Y Z \).

Similarly,

(2) \( F = X(Y \overline{Z} + \overline{Y} Z) \)
\( R = X, S = \overline{Y}, T = 0, U = \overline{Z} \) and \( V = 0 \)
\( F = X(Y \overline{Z} + \overline{Y} Z) \)

(3) \( F = \overline{X} Y \overline{Z} + X \overline{Y} Z \)
\( R = Y, S = Z, T = X, U = \overline{X} \) and \( V = 0 \)
\( F = \overline{X} Y \overline{Z} + X \overline{Y} Z \)

(4) \( F = X Y + Y Z \)
\( R = 0, S = X, T = Y, U = Z, \) and \( V = 0 \)
\( F = X Y + Y Z \)

(5) \( F = X Y \overline{Z} + X Y Z + X \overline{Y} Z \)
\( R = 1, S = X, T = \overline{Y}, U = Z \) and \( V = 0 \)
\( F = X Y \overline{Z} + X Y Z + X \overline{Y} Z \)

(6) \( F = X Y \overline{Z} + \overline{Y} Z \)
\( R = Y, S = \overline{Y}, T = Z, U = X, \) and \( V = 0 \)
\( F = X Y \overline{Z} + \overline{Y} Z \)
(7) \[ F = XY + YZ + ZX \]
\[ R = \overline{X}, S = Y, T = 1, U = Z \text{ and } V = 0 \]
\[ F = XY + YZ + ZX \]

(8) \[ F = XY \overline{Z} + \overline{X}Z + \overline{Y}Z \]
\[ R = X, S = \overline{Y}, T = Z, U = 1 \text{ and } V = 0 \]
\[ F = XY \overline{Z} + \overline{X}Z + \overline{Y}Z \]

(9) \[ F = \overline{X}Y + Z \overline{Y} \]
\[ R = Y, S = \overline{X}, T = Z, U = Z \text{ and } V = 0 \]
\[ F = \overline{X}Y + Z \overline{Y} \]

(10) \[ F = XYZ + XY \overline{Z} + \overline{X}Y \overline{Z} + \overline{X} \overline{Y}Z \]
\[ R = 1, S = X, T = Y, U = Z \text{ and } V = 1 \]
\[ F = XYZ + XY \overline{Z} + \overline{X}Y \overline{Z} + X \overline{Y}Z \]

Thus, it can be seen that Equation 13 realizes all the 10 equivalence classes of three variables. The implementation of Equation 13 is shown in Figure 9.

In Figure 9 it can be seen that the circuit requires 5 input pins and 2 output pins, the latter representing true and complementary output functions, to realize all the 10 equivalence classes of three variables. In its present state the circuit becomes a universal logic circuit of three variables.

A list of the pin connections for the circuit shown in Figure 9, for the 10 equivalence classes shown in Table 1, is given in Table 2. 1 or 0 represents a logical one
Figure 9: Three-Variable ULC (5 input pins)
or logical zero tied to an associated pin. This table can be used to obtain all the 10 equivalence classes.

TABLE 2

PIN CONNECTIONS FOR THREE-VARIABLE (5 input pin) ULC

<table>
<thead>
<tr>
<th>Equivalence Class Number</th>
<th>R</th>
<th>S</th>
<th>T</th>
<th>U</th>
<th>V</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>X</td>
<td>Y</td>
<td>X</td>
<td>Z</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>X</td>
<td>Y</td>
<td>0</td>
<td>Z</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>Y</td>
<td>Z</td>
<td>X</td>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>X</td>
<td>Y</td>
<td>Z</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>X</td>
<td>Y</td>
<td>Z</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>Y</td>
<td>Y</td>
<td>Z</td>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>X</td>
<td>Y</td>
<td>1</td>
<td>Z</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>X</td>
<td>Y</td>
<td>Z</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>Y</td>
<td>X</td>
<td>Z</td>
<td>Z</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>X</td>
<td>Y</td>
<td>Z</td>
<td>1</td>
</tr>
</tbody>
</table>

The ULC shown in Figure 9 can be used to realize any output function of three variables with the help of the ten representative functions of three variables. Figure 10 shows the K-map of 10 representative functions of three variables.
REPRESENTATIVE FUNCTIONS OF THE TEN EQUIVALENCE CLASSES

Figure 10
To realize any output function, first, the given function is simplified, and the equivalence class found from Table 1 and Figure 10. Then from Table 2, the related term(s) of Equation 13 is/are obtained. Then by connecting the input pin with proper available input variables (or their complement), the required output function is realized. In addition, Table 1 and Figure 10 can be utilized to locate the output pin, which will give the desired output function. An example is given below to illustrate the above procedure.

Example 1:

\[ F(X,Y,Z) = \Sigma m(0,2,5) \]

Solution:

\[
\begin{array}{ccc}
X & \bar{Y} & Z \\
1 & 1 & 1 \\
\end{array}
\]

\[ F = \bar{X} \bar{Y} \bar{Z} + X \bar{Y} Z + \bar{X} Y Z \quad (14) \]

From Table 1 and Figure 10 it is seen that the output function obtained is of the equivalence class number 5. Further from Table 2 it is found that the value of R is 1 and V is 0. Substituting these values in Equation 13 it is seen that the terms \( S T U + S \bar{T} \bar{U} + U \bar{S} \bar{T} \ldots \) \( (14A) \) give the desired output function, provided the input variables are substituted in the manner shown below (this is done by trial and error method). Substituting \( Y \) for \( T \), \( \bar{X} \) for \( S \) and \( Z \) for \( U \), Equation 14A can be written as follows:
\[ F = X \bar{Z} \bar{Y} + \bar{X} \bar{Y} \bar{Z} + \bar{X} Y Z \]

This is the desired output function. Further from Table 1 and Figure 10 it can be seen that the desired output function is located on the true output pin (F of Figure 9).

C) Design of a Four-Variable ULC Using 2 Three-Variable ULCs.

In the previous section the design of a three-variable ULC with 7 I/O pins was discussed (2). It is now proposed to consider the design of a ULC of four variables, using 2 three-variable ULCs.

Lemma - 1:

Any n-variable function \( F \) can be expanded as

\[
F(X_1, X_2, ..., X_n) = \sum_{i_1=0}^{1} \sum_{i_2=0}^{1} ... \sum_{i_{n-k}=0}^{1} (X_1^{i_1} X_2^{i_2} ... X_{n-k}^{i_{n-k}}) \cdot \ldots \cdot X_{n-k}^{i_{n-k}})
\]

Using Lemma 1, four variable functions can be written as:

\[
F(W, X, Y, Z) = \overline{W} F(0, X, Y, Z) + W F(1, X, Y, Z)
\]

Equation 15 presents a decomposition of four-variable functions into 2, three-variable functions, \( g \) and \( h \).

The block diagram of Equation 15 is shown in Figure 11.
Figure 11: Block Diagram of Four-Variable ULC

Thus, as shown in Figure 11, it requires 2 three-variable ULCs and a two-variable ULC. One of the three-variable ULC may be used for realizing all the 10 equivalence of three variables for part g of Equation 15, and the other for realizing part h of the equation. A two variable ULC is used to connect the proper outputs of part g and h with the fourth input variable as shown in Figure 11. A complete implementation of Figure 11 is shown in Figure 12. The
circuit shown here could be used to realize two, three, or four-variable functions. In Figure (12) it can be seen that 5 I/0 pins are required to realize a two-variable function, 7 I/0 pins to realize a three-variable function and 19 I/0 pins to realize a four-variable function. In all the cases the generated output function appears in true and complemented form. A brief description of the procedure for using Figure 12 to realize a given four-variable function is given.

The given function of four variables should be decomposed into two three-variable function. Each of the three-variable functions has to be simplified. The three-variable function with \( \bar{W} \) is designated as \( g \) while that with \( W \) is designated as \( h \). From Figure 10 and Table 1, the number of the equivalence class of part \( g \) and part \( h \) are found. Then from Table 2 the related term(s) of Equation 13 for part \( g \) and \( h \) is /are found. Then by manipulating the input terminals with available inputs, the required output can be determined. From Table 1 and Figure 10 the type of output (i.e. whether it is true or complementary) can be found. The corresponding output pins of modules \( g \) and \( h \) are connected to \( I_0 \) and \( I_1 \), respectively. The output terminals of module 3 will give the desired output of a four-variable function. The following example is given to illustrate the procedure described above.
Figure 12: Four Variable ULC Constructed From Three Variable ULCs (13 input pins)
Example 1:

$$F(X, Y, Z, W) = \Sigma m(0, 1, 2, 3, 6, 8, 11, 13, 14)$$

Solution:

$$\begin{array}{ccc}
WX & YZ & \\
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1 \\
\end{array}$$

The function is decomposed into 2 three-variable functions as $$F = \overline{W}g + Wh$$. By simplifying $$g$$ and $$h$$, $$F$$ can be written as $$F = \overline{W}(\overline{X} + Y \overline{Z}) + W(X \overline{Y} Z + X \overline{Y} \overline{Z} + \overline{X} Y Z + \overline{X} \overline{Y} \overline{Z})$$.

By using Table 1 and Figure 10 it is seen that $$g$$ belongs to the equivalence class number 4, while $$h$$ belongs to equivalence class number 10. Further from Table 1 and Figure 10 it can be shown that

$$F = \overline{W} \overline{g} + W \overline{h}.$$  \hspace{1cm} (17)

From Table 2 it is found that $$R = 0$$ and $$V = 0$$. Substituting these values in Equation 13 the required $$g$$ part can be obtained provided the input terminals are manipulated as $$S = Z$$, $$T = X$$ and $$U = \overline{Y}$$. It is known from Equation 17 that the desired output of $$g$$ part is on the complementary pin. Further from Table 2 it is seen that the terms $$STU + RST \overline{U} + RUS \overline{T} + RT \overline{S} \overline{U} V$$ of Equation 13 give the required $$h$$ part of the output function, if $$S$$ is connected with $$X$$, $$T$$ with $$Y$$, $$U$$ with $$Z$$ and $$V$$ with 1. Also, from Equation 17 it is known that the desired output function of $$h$$ part is on the
complementary pin. The 2 outpins of module g and h are connected to I₀ and I₁ of module 3, respectively. Thus, the desired output function is obtained on the output pins of module 3.

The development of a circuit using two and three-variable ULC, which can realize a function of n variables is given. First, decompose the given function by reducing one variable at a time until it attains the form of three-variable functions. Then the outputs of the concerned number of three-variable ULCs are connected to the corresponding number of two-variable ULCs. The desired function is obtained on the final output pins. This is illustrated by the following example:

\[ F(x_1, x_2, x_3, x_4, x_5) = \overline{x}_1 F(0, x_2, x_3, x_4, x_5) + x_1 F(1, x_2, x_3, x_4, x_5) \]

\[ = \overline{x}_1 g_1 + x_1 h_1 \ldots \]

On further decomposition

\[ g_1 = \overline{x}_2 F(0, 0, x_3, x_4, x_5) + x_2 F(0, 1, x_3, x_4, x_5) \]

\[ = \overline{x}_2 g_2 + x_2 h_2 \]

\[ h_1 = \overline{x}_2 F(1, 0, x_3, x_4, x_5) + x_2 F(1, 1, x_3, x_4, x_5) \]

\[ = \overline{x}_2 g_3 + x_2 h_2 \]

In Equation 17 g and h parts are now functions of three variables. The implementation of which is partly
shown in Figure 13 which is the total implementation of the
given five-variable function.

In Figure 13 the output terminals of modules $g_2$ and
are connected to \( I_2 \) and \( I_3 \) of the corresponding two-variable ULC which will result in module \( g_1 \). Similarly, the output terminals of modules realizing \( g_3 \) and \( h_3 \) are connected to the input terminals \( I_4 \) and \( I_5 \) of the corresponding two-variable ULC which will in turn result in module \( h_1 \). Thus, the resultant modules \( g_1 \) and \( h_1 \) are but an implementation of Equation 17. The output terminals of modules \( g_1 \) and \( h_1 \) are then connected to \( I_0 \) and \( I_1 \) of the corresponding ULC of two-variable. The final output pins will give the desired output function.
CHAPTER IV
CONCLUSION

In Chapter II, the ULC of three variables, designed by Yau and Tang, has been presented. They have assumed that any one variable and its complement are available as a free input variable and that the remaining two variables are available as fixed input variables. In the same chapter, the design of a four-variable ULC using three-variable ULCs has also been described. The existing procedure for finding the proper value of each front terminal connection for the three and four-variable ULCs, has been described. A new method for determining the value of front terminal connections for Yau and Tang's ULC, directly from the K-map of a given output function, has been devised and explained in detail.

In Chapter III the design of a ULC of three-variables, using the theory of equivalence classes, with the assumption that all variables and their complements are available as inputs, has been described\(^2\). In the same chapter the design of a four-variable ULC using two three-variable ULCs, has also been presented. A method for developing a circuit to realize n-variable functions has been added.

It is proposed in this chapter to draw a comparison between the ULC of four variables designed by Yau and Tang, and that designed by using the theory of equivalence classes. To effectively accomplish this, it is assumed that both true and complementary inputs are available in each case.
Yau and Tang, in their design of the four-variable ULC, have used three-variable ULCs applying the theory of decomposition and using 22 I/O pins. In this paper it has been found possible to achieve the same result by combining the theories of decomposition and equivalence classes and using 19 I/O pins. This alternative procedure suggested above involves simplification, correlation and the consequent use of a lesser number of input pins. A detailed comparison is shown.

Figure 12

1. 2 modules of three-variable ULCs and 1 module of two-variable ULC are required to realize a four-variable function.

2. 13 input pins are needed to realize a four-variable function.

3. 6 output pins are required to realize a four-variable function, with the assumption that both, true and complementary output functions are generated in each module.

4. For every output function of four variables, there is need to compute 10 input connections.

5. The appropriate value of front terminals connection cannot be readily determined.

Figure 3

1. 3 modules of three-variable ULCs are required to realize a four-variable function.

2. 16 input pins are needed to realize a four-variable function.

3. 6 output pins are needed to realize a four-variable function, with the assumption that both true and complementary output functions are generated in each module.

4. For every output function of four-variables, there is need to compute 8 input connections.

5. The appropriate value of front terminals connection can be readily determined.
(6) Assumption is made that all variables and their complements are available as inputs.

(7) 7 I/O pins are required for a ULC of three variables.

(8) A greater degree of universality is achieved since it could realize all the functions of two, three, and four variables.

Thus, in conclusion, it can be stated that an alternative procedure is possible with greater flexibility and universality and involving a lesser number of I/O pins.
BIBLIOGRAPHY


VITA

Mahendrakumar Punjalal Shah was born on August 9, 1944 in Modasa, Gujarat, India. He received his primary and secondary education in Modasa. He received a Bachelor of Science Degree in Electrical Engineering from the Birla Engineering College, affiliated to the Sadar Patel University, in June 1967.

He has been enrolled in the Graduate School of the University of Missouri-Rolla since September, 1968.

He was a graduate assistant during Spring 1969.