Chapter 8

8 Minimization Techniques

8.1 Introduction

The emphasis is on clean, irredundant, minimal designs has been dramatically affected by the evolution of LSI [VLSI] technology. There are instances where a minimal implementation costs more in board space [or die size] or design time without providing testing, cost, or execution speed advantages. An understanding of minimization techniques is still a requirement for the LSI [VLSI] designer, however, as much of the logical thought discipline is necessary for current micro-system designs.

For the chip designer, circuit cost can be shown to be linearly related to chip area, which in turn is a function (all other variables remaining fixed) of the number of gates and the number of connections. The emphasis of minimization would be gate count reduction for TTL, and ECL (bipolar LSI) since their gates occupy a larger chip area then connections; for MOS LSI, the emphasis of the minimization would be connection length or connection count (the connection length is related to the number of gates and to the number of connections).

Although highly simplified, expressing circuit cost as a function of the number of gates and the number of connections gives a reasonable approximation.[1]

For the circuit board designer, cost can be shown to be related to the number of packages and the number of connections. For high-speed designs, the emphasis of minimization would be connection length, which is a function of the number of gates and the number of connections. For densely packed boards, ,the emphasis of the minimization would be package count. It should also be noted that, for a fixed board size, as the cost of the IC packages drops, the cost of the connections becomes dominant and the emphasis of the minimization shifts accordingly.[2]

For testing purposes, redundant circuit are not as testable as their irredundant equivalents, since redundancy masks faults (see Chapter 12).

From the preceding, it is evident that judicious minimization remains an important part of any cost-effective design and implementation.

This and the following chapters will present some of the new techniques in minimization such as Svoboda's weight algorithm and the fundamental product concept, and some of the logical-instrument teaching aides that have been used to demonstrate the theorems behind these techniques.

8.2 Single-Output Minimization

For simplification, the minimization techniques will be presented using a single-output combinational circuit as shown in FIG 8-1. (A brief look at multiple-output minimization is included in Section 8.6).

8.2.1 Design Constraints

The selection of a minimization technique is a function of the overall objectives. Any design must be implemented under certain constraints similar to those called out in Table 8-1. A design is also broken up into units that are of a reasonable size for human comprehension and/or for the purpose of honoring the limits of design support systems

---

[1] Holds true for today's ASIC designs.

[2] This pattern continues to evolve. Die size and power are driving forces today.

will restrict the actual “cleanness” of the final design. By breaking a circuit up into modules, some opportunities for minimization will become forbidden. The further partitioning into sub-modules will also affect the minimization. The tradeoff is the design time. Each additional variable increases the problem complexity by a factor of two in the binary space. Therefore, it is more efficient if, after the initial module design is completed, it is reviewed for further reductions.

Sub-module size is a function of the techniques available for the design, If an APL program such as the package in the first seven chapters of this book is used, the restriction is thirteen to fifteen variables (limited by the workspace size).[3] The parallel Boolean processor, if built as discussed in a later chapter, could handle up to 22-variable modules.

The design constraints such as device tipe,, board space, etc., will determine the desired result of the minimization to be used. For example, NOR-NOR gate staging implements the minimal $\sum\prod$ form, while NAND-NAND gate staging implements the minimal $\prod\sum$ form. If multiplexors may be used, Marquand mapping and column-only minimization should be performed.

For any of these approaches, the underlying minimization techniques are the same.

8.2.2 Minimization by Inspection

There are many instances where the algebraic expression or the logical map (Venn, Veitch, Karnough, Marquand, Triadic) is simple enough for obvious reductions to be made without any particular technique being formally applied. This is the case for the simple expressions of six or fewer variables such as:

$$Y = A + AC + A\underline{B}$$

which may be immediately rewritten as

$$Y = A$$

By recognizing that

$$A\ (1 + C) = A\ (1) = A$$

Table 8-1 Design Constraints

Design Constraint	Comment
Cost	Affects everything
Fan-In, Fan-Out	Function of logic family to be used
Logic Type	Any constraint on NAND, NOR, etc., availability [base die process and wafer-fab line scheduling]
Timing Consideration	High-speed logic requirement
Board Size	Available “real-estate” (restricts package/gate count)
Reliability Requirements	Constrains redundancy allowed
Time	How much effort can be expended
Support	Computer assist; Manual Reductions;
Power Requirements	Further constraints on the implementation
Number of Variables	Pin-out limitations (primary variables); internal or secondary variables are a functions of the number of connections)

[3] Computer capacity at the time bears no resemblance to what we have today.

Design Constraint	Comment
Partitioning	Breaking a module up into manageable pieces
Module Definition	Module boundaries should be maintained
Classification of Design	Spacecraft (most minimal, most reliable); military (rugged); commercial

Design Constraints (Con't)

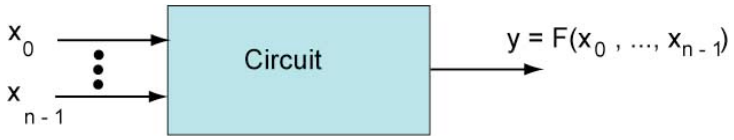


Figure 8-1 Combinational, Single-Output Circuit

An equivalent example for a map wpld be the appearance of an obvious structure as shown in Figure 8-2 for Karnough and Marquand maps of

$$Y = \underline{X_3}\underline{X_1} + \underline{X_3}X_2 + X_2X_1$$

8.2.3 Minimization and Mapping by Observation

For a small number of variables, a function to be minimized may be mapped by expanding the expression into a sum of products form and marking a "one" on the map at all points corresponding to a ***minterm*** of the function. Points which are logical distance one apart are connected. (On a Karnough 4-variable map, these would be adjacent points.) The resulting structures represent reduced terms. The terms of the largest structures form the prime implicants of the function.

This is a casual approach and as the number f variables and/or the number of minterms of the function increases, the reliability of this method decreases.

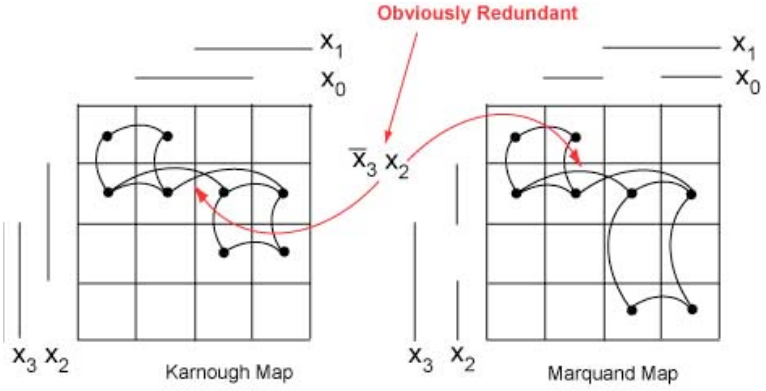


Figure 8-2 Reducing a Simple Function

8.2.4 Minimization by Algebraic Manipulation

Algebraic manipulation of an expression for a function to change it into a reduced minimal form is tedious and prone to human error. The same disadvantages may be cited for tabular reduction techniques.

Figure 8-3 demonstrates an algebraic reduction and presents a Marquand map of the sample function.

$$Y = BDE + \overline{BC}D + CDE + \overline{AB}CE + \overline{AB}C + \overline{BCDE}$$

$$Y = BDE + \overline{BC}D + CDE + \overline{AB}(CE + C) + \overline{BCDE}$$

$$Y = BDE + \overline{BC}D + CDE + \overline{AB}C + \overline{BCDE}$$

$$Y = BCDE + B\overline{C}DE + \overline{B}CDE + BCDE + \overline{BC}DE + \overline{BC}D\overline{E} + \overline{AB}C + \overline{BCDE}$$

$$Y = BCDE + B\overline{C}DE + \overline{B}CDE + BCDE + \overline{BC}DE + \overline{BC}E + \overline{AB}C$$

$$Y = BDE(C + \overline{C}) + \overline{B}DE(C + \overline{C}) + \overline{BC}E + \overline{AB}C$$

$$Y = DE(B + \overline{B}) + \overline{BC}E + \overline{AB}C$$

$$Y = DE + \overline{BC}E + \overline{AB}C$$

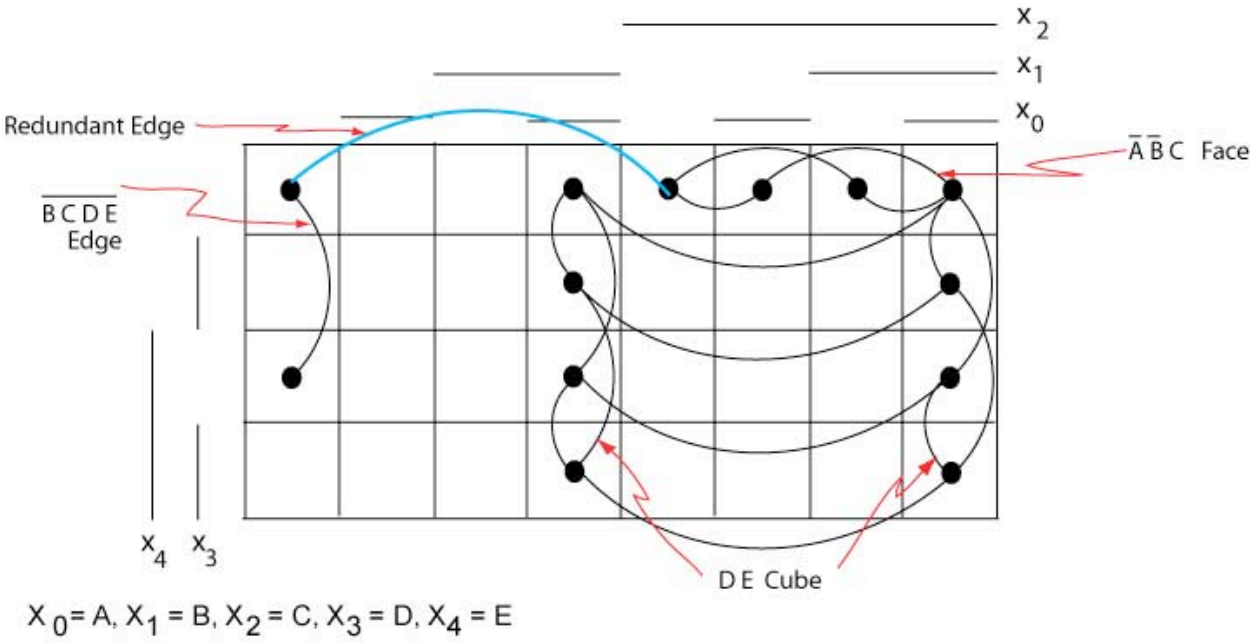


Figure 8-3 Algebraic and Map Minimization of a 5-VariableFunction

8.3 Svoboda's Weight Algorithm

A convenient algorithm for manual or programmed minimization of a function is the Weight Algorithm developed by Svoboda. It is readily applied to manual solutions of up to eight (8) variables, depending upon the complexity of the function. To perform the weight algorithm manually, proceed as follows:

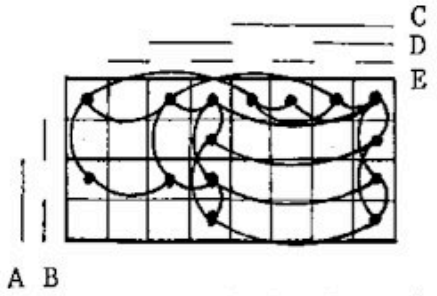
1. Map all terms where the function Y is true as "1" points on a Marquand Map. (A Karnough map may be used but it is inconvenient.) Include all "Don't Care" terms as "#" points. Unmarked points are those for which the function Y = 0.
2. Connect all pairs of points of logical distance one, where $p_i = 1$ or $p_i = \#$, where p_i is the label of point i. All such connections are referred to as "edges".
3. Using a second map (for clarity), fill in the squares corresponding to the minterms of the function Y (all $p_i = 1$) with the number of edges connected to that minterm. Include in the count edges between minterms where both $p_i = 1$, and between minterms where one $p_i = \#$.
4. Scanning the points sequentially from the origin (P0), find the minterm with the lowest edge count or ***weight***. The first search should be for points with weight w = 0.) This is a ***critical point*** of the function Y.

   In his paper, "***Ordering of Implicants***", Svoboda discusses the natural phenomena that, when coverage is made from the origin forward in sequence, the probability of the minimal function expression being obtained is increased. The ordering concept is the basis of the weight algorithm, fundamental product coverage, and the multiple-output minimization.
5. Select the term representing the largest structure (edge, face, and cube) which covers that minterm. [Not limited to 3 dimensions.] This is a ***prime implican***t of the function.
6. Record the prime implicant and mark all minterms of the prime implicant as ***covered*** by labeling the points as ***Don't Cares***.
7. Continue scanning from the last critical point to find the next uncovered minterm with the lowest weight. If there are none with the last selected weight value, increment the weight value by one and return to the origin to begin scanning again.
8. Repeat steps 5 through 7 until all minterms of the function are covered. The selected minterms form the critical set of minterms of the function.

The problem with the algorithm is that, where a choice of structures exists, the algorithm fails. In some cases a solution may be obtained by choosing the structure which covers the most 1s or ***uncovered*** minterms. Where there are equal choices, there is more than one solution and the algorithm fails to provide a choice of one best solution. In many cases, there is little practical advantage in pursuing more than one of the minimal solutions.

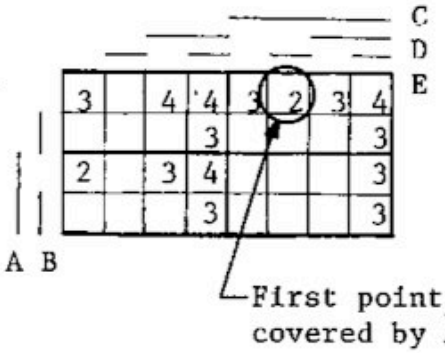
The algorithm is presented here without proof. Theorems for this and other procedures are presented in Chapter 9.

Figure 8-6 presents a step-by-step solution of a 5-variable function using Marquand Maps. Figure 8-5 presents an interesting 6-variable, incompletely specified function. The latter example appears throughout the text.

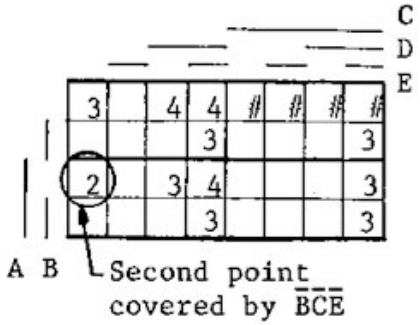


a. Edges of the function

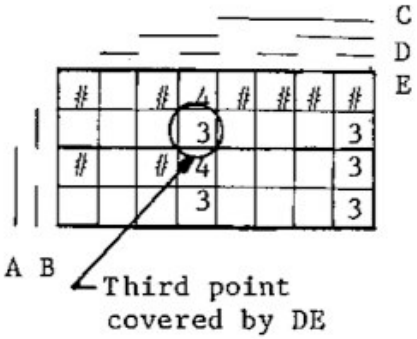
$$y = BDE + \bar{B}\bar{C}D + CDE + \bar{A}\bar{B}CE + \bar{A}\bar{B}C + \bar{B}C\bar{D}\bar{E}$$



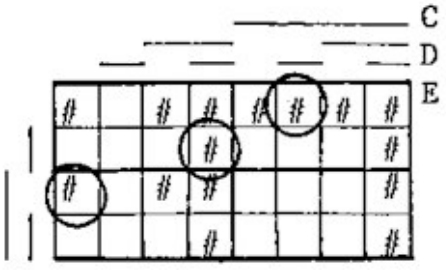
b. Weights of the function, showing the first critical point with W = 2.



c. The second critical point with W = 2.



d. The third critical point with W = 3



Coverage is complete

e. Final function is

$$y = \bar{A}\bar{B}C + \bar{B}\bar{C}\bar{E} + DE$$

○ indicates critical point

Figure 8-4 The Weight Algorithm for a Five-Variable Function

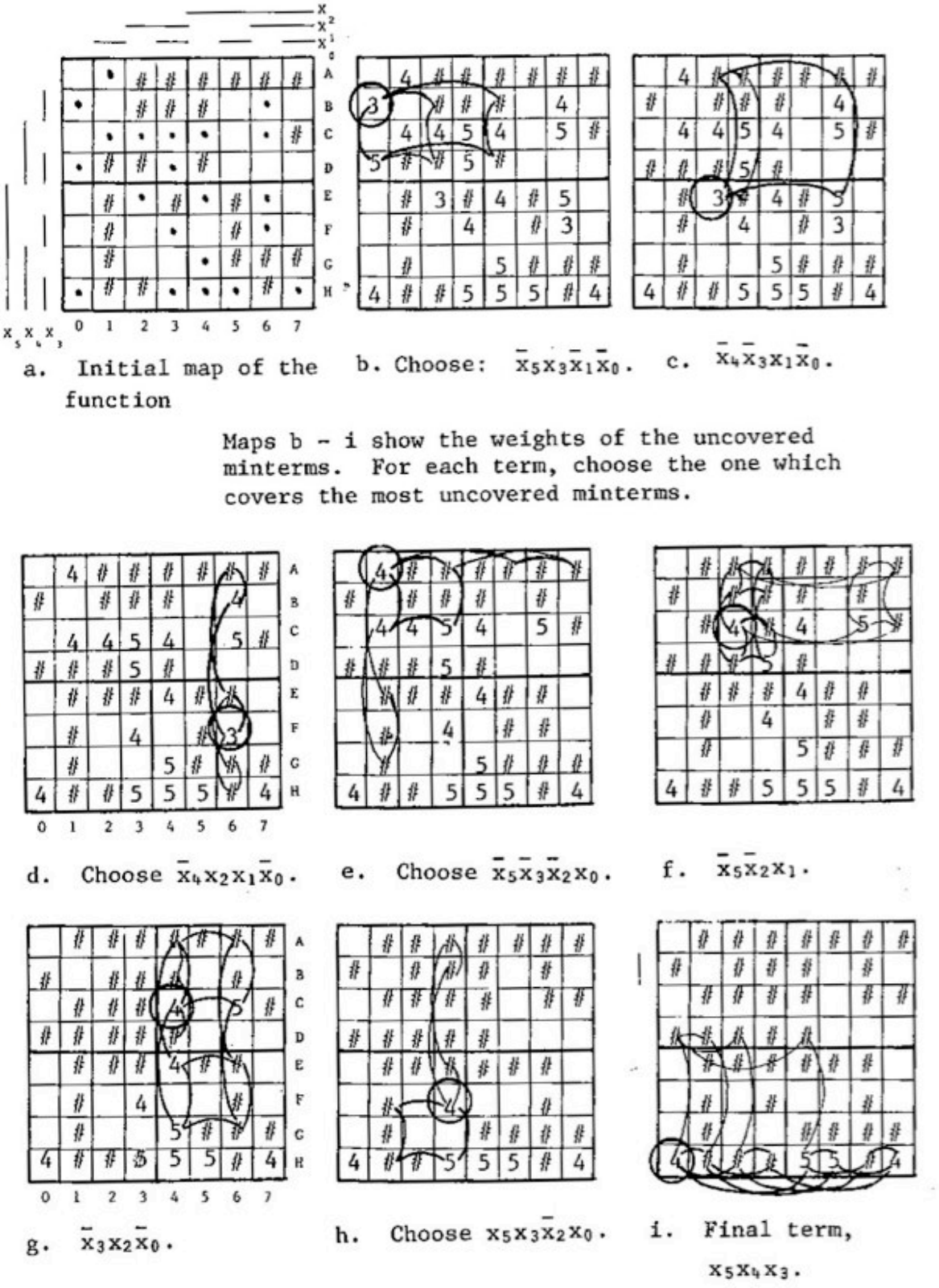


Figure 8-5 The Weight Algorithm for a Six-Variable Function

8.4 Logical Instruments: The Weight Deck (Original 80-Hole Punched Card Deck)

8.4.1 Description of the Cards

As a teaching aide[4] for the weight algorithms, Svoboda developed a deck referred to as the ***weight deck***, a set of 80-column punched cards where each card represents a point in the 6-variable binary space.

Each card is indexed in the upper left-hand corner by the row-column short-hand index in the form xn (letter-digit).Each card is also indexed in the upper right corner by the decimal point identifier.

The card is punched with a group of six vertically grouped punches for each point in the binary space. Holding the card so that the eighty columns run vertically, from top to bottom and so that the punch rows zero through seven run from let to right, the groups are seen to be arranged in rows and columns corresponding to the Marquand Map. (See Figure 8-6.)

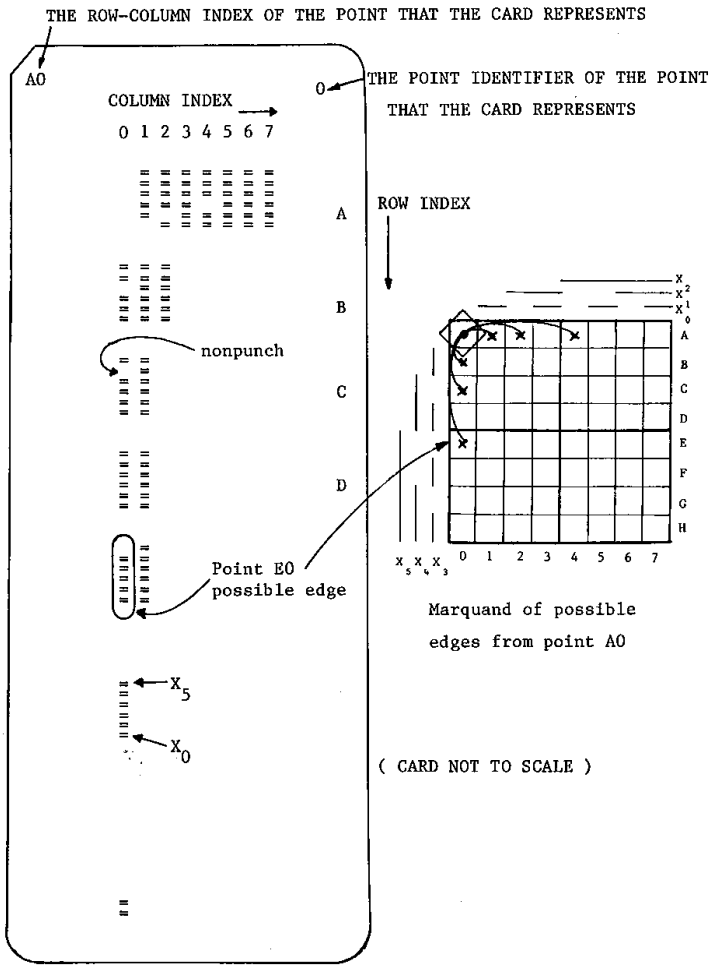


Fig. 8.6. Weight deck card A0.

Figure 8-6 Weight Deck Card A0

[4] 1960s, 1970s

A ***sextet*** is non-punched at the point corresponding to the point the card represents. One punch is a sextet is non-punched if the point in that sextet represents is at logical distance one from the point represented by the card. The particular punch position corresponds to the variable which changes (the variable missing from the term representing the edge formed by the two points).

8.4.2 Finding the Weights for the Six-Variable Example

Referring to Figure 8-5a, to obtain the weights or edge counts for the minterms of the function, from the set of 64 cards representing th binary space, remove those cards which represent the zeros of the function.

The set of cards representing the zeros, when aligned and held to a light (hold up in front of a lightbulb), will show one punched position in each group or punch positions for each edge which may be formed between "1" and "Don't Care" points of the function.

By reading a count of the punch positions punched in each sextet, the weight of the point that sextet represents is obtained.

In addition, the punch position contains the information describing which edge of the six possible for any point exists. The lowest punch position of the sextet corresponds to x_0 and the highest to x_5.

For our example, use the cards representing the points: A0, B1, B5, B7, C0, C5, D5, D6, D7, E0, E7, F0, F2, F4, F7, G0, and G2. These cards will produce the weights for the "1" and for the "Don't Care" points. The maps shown earlier did not record the weights for the "Don't Care" points. (See Figure 8-5b.)

8.5 Svoboda's Fundamental Product Procedure

8.5.1 Introduction

For more difficult problems, the structures or terms which cover a given minterm may not be "visible", or more than one structure of seemingly identical properties may cover the minterm. In these cases, the choice of the proper term for the best coverage becomes non-trivial. The fundamental product and the theorems of mutual term exclusivity were developed for these situations.

The ***fundamental product*** of a minterm is defined as the product of those literals that occur in every edge that can be connected to the minterm or point under examination. In other words, it is the product of those literals that are present in every term that covers the minterm.

An ***effective literal*** is a literal that, when added to the literals of a fundamental product, forms a product that causes ones to be included in the term and no zeros to be included.

Any minterm, which is also covered by all of the prime implicants, which are found for the minterm under examination, is called ***mutually term exclusive***.

8.5.2 The Procedure for Finding the Fundamental Product and the Effective Mask

To manually perform minimization via the fundamental product modification to the weight algorithm, proceed as follows (note that this is a programmable procedure):

1. Perform the first steps of the weight algorithm to find the weights of all minterms of the function to be reduced.
2. Begin at the origin and scan in decimal index order to find the first critical point (as per the weight algorithm).
3. Find the fundamental product for that minterm by listing all of the edges that can be formed from that minterm to points that are minterms or points that are Don't Cares. The fundamental product will be those literals which appear in each and every edge.
4. Examine (using a scratch map) only those points represented by the fundamental product. Are there any literals that may be added to the fundamental product to form an effective mask such that all ones visible on the scratch map are covered, but no zeros are covered? These literals form the effective mask.

   If there is a failure to find an effective mask, mark this point as “failed” and proceed to the next critical point and repeat the process.

   If there is success, the term so formed belongs to the minimal form. Mark all minterms that have been covered by this term as “Don’t Cares” and proceed to the next critical point.
5. Periodically retry “failed” points.

6. Repeat steps 3 through 5 until all minterm points are covered.

Part of the solution of the six-variable problem using the fundamental product approach is shown in Figure 8-7. The partial solution should be compared to that of Figure 8-5. **Note**: it is still possible to have multiple solutions.

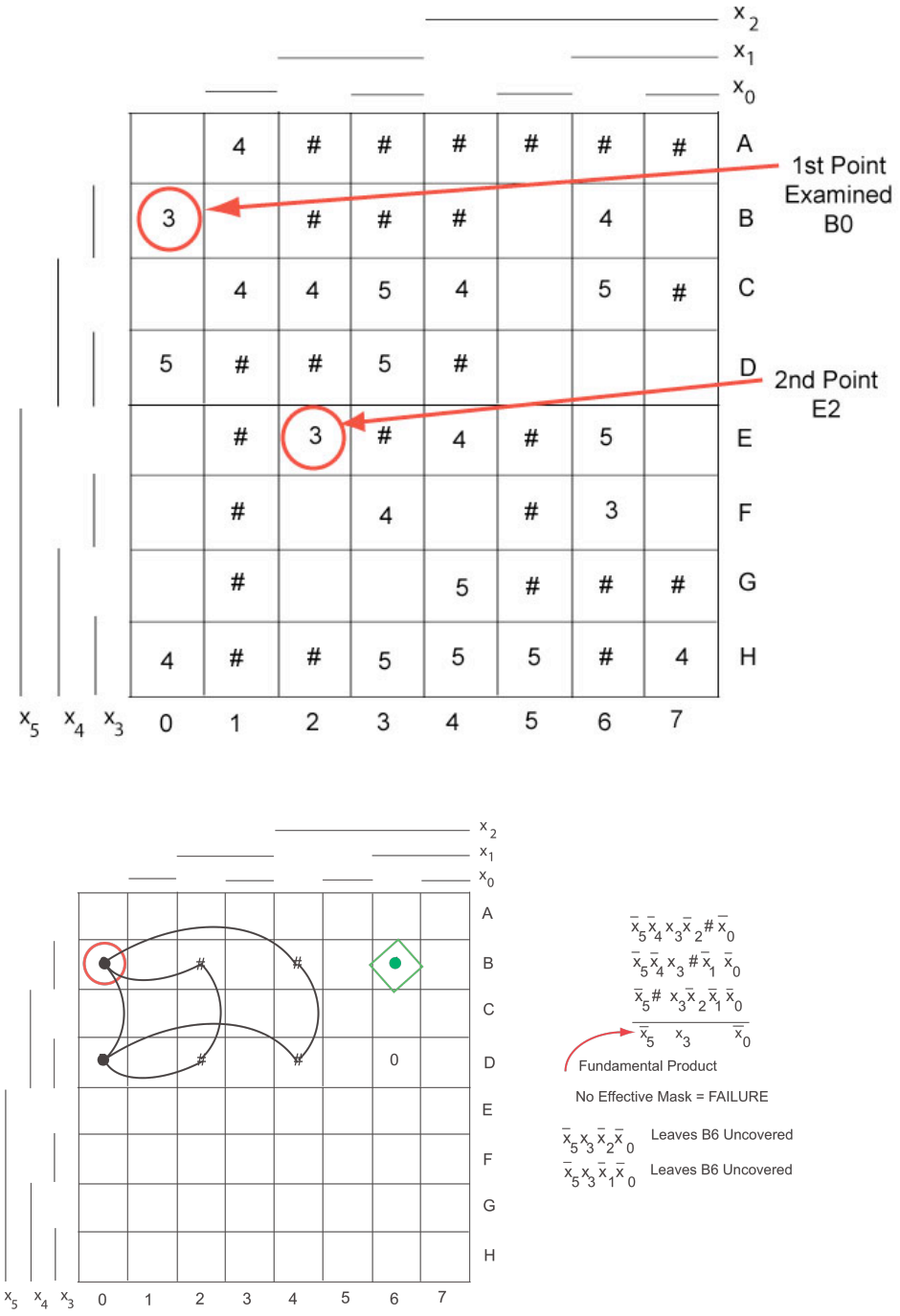
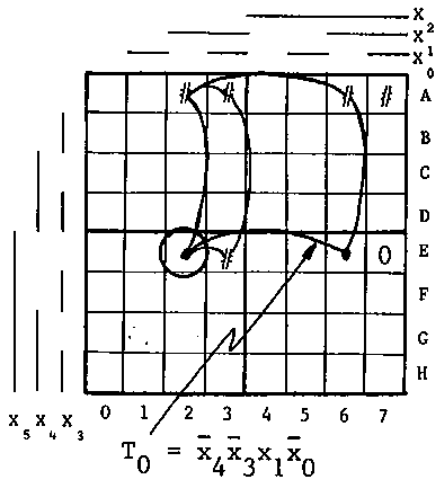


Figure 8-7 Start



E2 Edges: $x_5\bar{x}_4\bar{x}_3\bar{x}_2x_1\#$

$x_5x_4x_3\#x_1\bar{x}_0$

$\#\bar{x}_4x_3x_2x_1\bar{x}_0$

$\bar{x}_4\bar{x}_3 \quad x_1$

Fundamental product

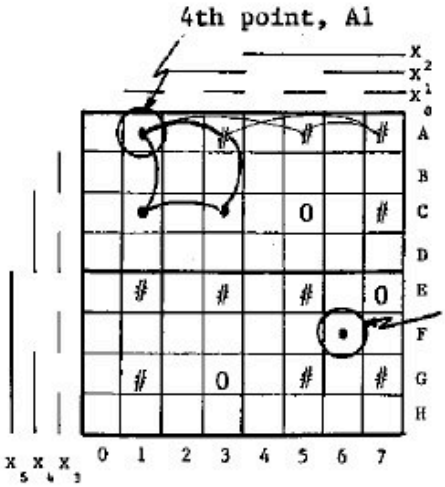
Effective mask: $\bar{x}_4\bar{x}_3x_1\bar{x}_0$

covers all ones, no zeros

($\bar{x}_4\bar{x}_3\bar{x}_2x_1$ leaves E6 uncovered)

SUCCESS

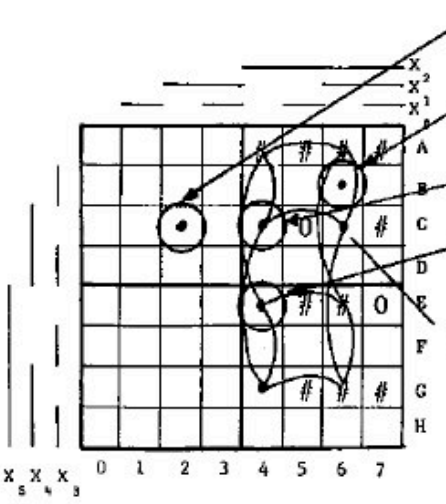
Figure 8-8 Continued



A1: Fundamental product: $\bar{x}_3 x_0$

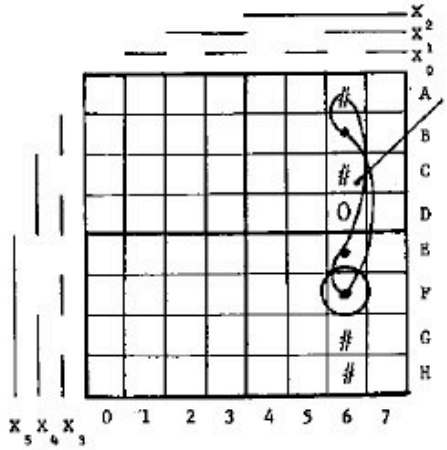
Effective mask: $\bar{x}_5 \bar{x}_3 \bar{x}_2 x_0$

SUCCESS



E4: Effective mask: $\bar{x}_3 x_2 \bar{x}_0$

NOTE



Retry F6 (3rd point)

Fundamental product

$x_2 x_1 \bar{x}_0$

Effective mask

$\bar{x}_4 x_2 x_1 \bar{x}_0$

SUCCESS

$T_3 = \bar{x}_4 x_2 x_1 \bar{x}_0$