84

04568

MICROFILMED – 1984

Santirakul, Krayim

MULTI-VALUED LSI/VLSI LOGIC DESIGN

The University of Oklahoma PH.D. 1983

PLEASE NOTE:

In all cases this material has been filmed in the best possible way from the available copy. Problems encountered with this document have been identified here with a check mark ✓.

1. Glossy photographs or pages _____
2. Colored illustrations, paper or print _____
3. Photographs with dark background _____
4. Illustrations are poor copy _____
5. Pages with black marks, not original copy _____
6. Print shows through as there is text on both sides of page _____
7. Indistinct, broken or small print on several pages _____
8. Print exceeds margin requirements _____
9. Tightly bound copy with print lost in spine _____
10. Computer printout pages with indistinct print _____
11. Page(s) 10 lacking when material received, and not available from school or author.
12. Page(s) _____ seem to be missing in numbering only as text follows.
13. Two pages numbered _____. Text follows.
14. Curling and wrinkled pages _____
15. Other _____

THE UNIVERSITY OF OKLAHOMA

GRADUATE COLLEGE

MULTI-VALUED LSI/VLSI LOGIC DESIGN

A DISSERTATION

SUBMITTED TO THE GRADUATE FACULTY

in partial fulfillment of the requirements for the

degree of

DOCTOR OF PHILOSOPHY

By

KRAYIM SANTRAKUL

Norman, Oklahoma

1983

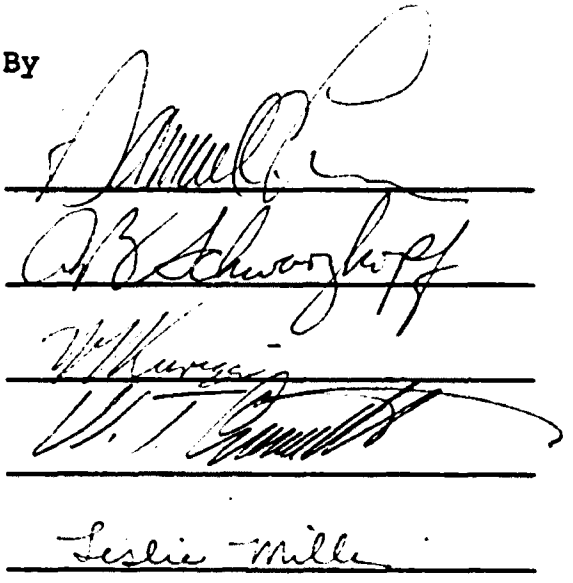
MULTI-VALUED LSI/VLSI LOGIC DESIGN

A DISSERTATION

APPROVED FOR THE DEPARTMENT OF ELECTRICAL ENGINEERING

AND COMPUTER SCIENCE

By



ACKNOWLEDGMENTS

I wish to express my sincere gratitude and appreciation to the many individuals who contributed to this research.

I especially want to thank Dr. Samuel C. Lee, my academic advisor, for his continuous advice, guidance and assistance throughout my entire graduate program. He has been a good friend who showed constant considerations in the years we worked together. Without him, I would not have been successful in completion of this degree.

I also wish to thank Dr. William T. Cronenwett, Dr. William L. Kuriger, Dr. Leslie L. Miller, and Dr. Albert B. Schwartzkopf for their valuable comments and willingness to serve in the supervisory committee.

I am sincerely grateful to my brother and my parents-in-law for their continuing moral support, understanding and encouragement over many years.

The warmest thanks and most affectionate appreciation are extended to my family, Chantra, my wife, who suffered the most throughout the several years of graduate school, and to Kayasith, my son, who will now be able to see more of his father.

And finally, I wish to thank my parents, Pan and Suparb Santrakul, who brought me up and made me what I am with

their endless love. With love and gratitude, I wish to dedicate this work to them.

TABLE OF CONTENTS

APPENDIX

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

This research describes a procedure for synthesizing any large complex logic system, such as LSI and VLSI integrated circuits. This scheme uses Multi-Valued MUltipleXers (MVMUX) as the basic building blocks and the tree as the structure of the circuit realization. Simple built-in test circuits included in the network (the main circuit), provide a thorough functional checking of the network at any time. Because the network can be partitioned into nearly identical subnetworks (Basic-Modular-Networks or BMNs) which, in turn, can be further partitioned into nearly identical sub-subnetworks, the testing of the entire network may be conducted in such a manner that all the sub-subnetworks and subnetworks are tested simultaneously by the built-in test circuits.

In brief, this dissertation has made the following four major contributions:

- developed a multi-valued Algorithmic State Machine (ASM) chart for describing an LSI/VLSI behavior (Sections 5.2 and 5.3)
- described a tree-structured multi-valued multiplexer network which can be obtained directly from an ASM chart (Sections 4.2 and 5.3)

- introduced a heuristic tree-structured synthesis method for realizing any combinational logic with minimal or nearly-minimal MVMUX (Section 4.5)

- presented a hierarchical design of LSI/VLSI with built-in parallel testing capability (Chapter 7)

The procedures and methods presented in this dissertation are completely general, systematic, and easy to apply to any m-valued ($m \geq 2$) combinational and sequential LSI/VLSI design.

CHAPTER I

INTRODUCTION

1.1 Advantages of LSI/VLSI Chips

The ongoing revolution in digital Large-Scale-Integration (LSI) spawned in the late 1960's is leaving a permanent imprint on all aspects of our lives. In the last few years, the system house has been shocked by the explosive increase in the number of components that can be integrated on a silicon chip. The level of integration has closely followed the trend predicted by Moore [153] as shown in Fig. 1.1: an increase by a factor of 2 every year. It is nowadays possible to integrate more than 100,000 components on a single chip. Digital circuits, such as memories and microprocessors show the highest integration level, while analog circuits exhibit a lower integration level. Because of the complexity, these digital circuits are better classified as subsystems than as circuits; mostly these subsystems include software, e.g. programmable circuits and microprocessors.

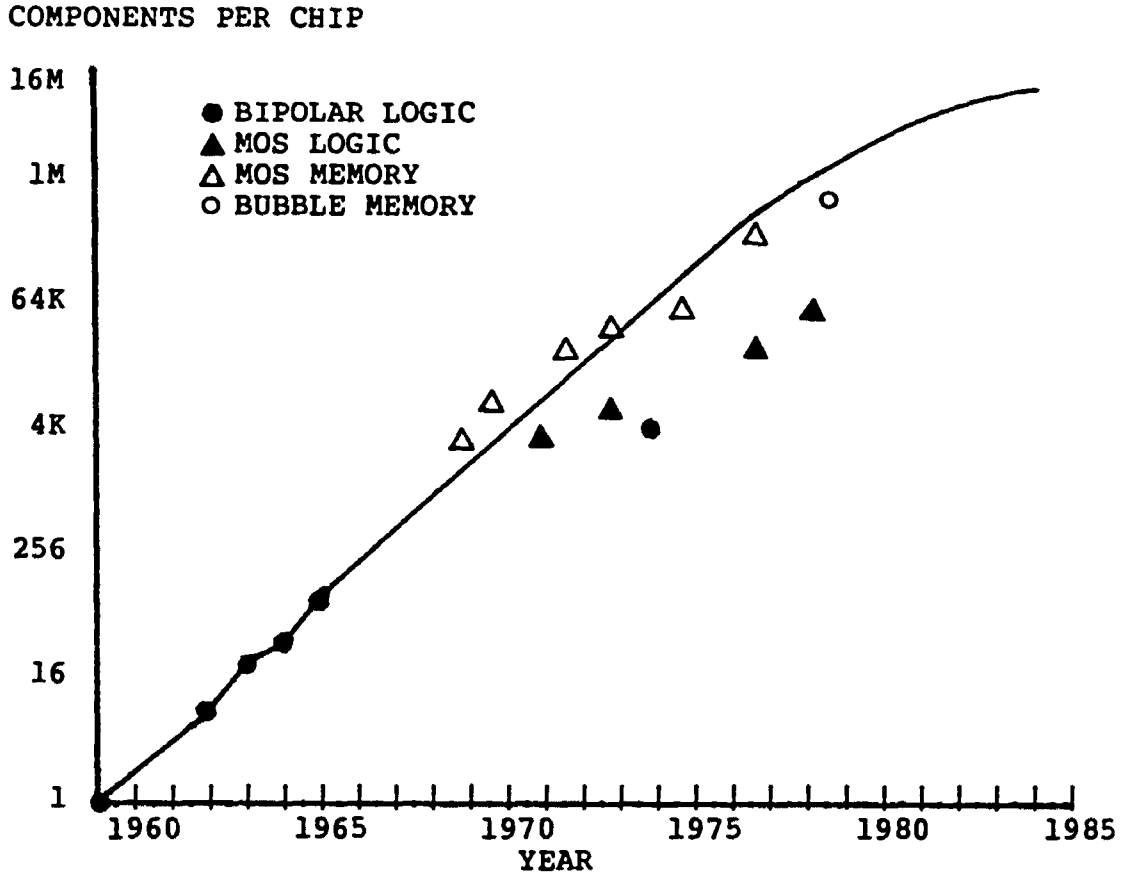


Fig. 1.1 Yearly Progress in Number of Components Per IC Chip.

ICs offer the fascinating advantages of smaller size, low power dissipation, and higher reliability; the most decisive advantage of ICs, however, is their cost advantage when mass produced. Due to their steadily improved cost effectiveness, ICs are forcing their way into different technical fields, and thereby replacing electromechanical functions and conventional electronic devices.

1.2 Advantages of the Use of MV Logic in LSI/VLSI Circuit

It is known [200] that the performance and cost per function of present LSI/VLSI circuit can be improved by the

MV logic system. The advantages of the MV Logic are:

1. increases the information per unit area, and

2. reduces the number of interconnections, and thereby reduces the complexity of the circuit.

Other advantages of MV LSI/VLSI over binary LSI/VLSI are:

3. IC size reduction.

4. More information per signal line.

5. IC pins reduction.

6. Interface complexity reduction.

7. Cost-per-function reduction.

8. Higher performance.

For these reasons, recently more and more designers have thought of taking these advantages of MV logic and using MV circuits in the design of digital systems. For example, Intel 8087 numeric processor and Intel 43203 iAPX-432 I/O interface chips both use MV ROMs. Each cell of these ROMs can store one of the four possible signal values [177].

Even though there are many advantages of the MV system over the binary system, so far the MV system is still not popular in practice. This may be due to the following reasons:

1. Most existing digital systems are binary systems and most logic designers are only familiar with binary logic design.

2. There is still room for technological improvements in binary LSI/VLSI circuits.

3. Most logic designers believe, rightly or wrongly, that the design of binary systems is easier than the design of an MV system.

4. The solving of the problems of noise immunity and tolerance of MV circuits is much more difficult than that of the binary system.

The advantages of MV logic over binary logic may be evident from the following example.

Example 1.1: Consider the 64-to-1 two-valued multiplexer and this 64-to-1 four-valued multiplexer of Figs. 1.2(a) and (b) respectively. They are constructed from single-data-select two-valued multiplexers and single-data-select four-valued multiplexers of Figs. 1.3(a) and (b) respectively. The cost of multiplexers is proportional to the number of transistors used and the chip density mostly depends on the transistors and resistors plus the silicon surface covered with metal signal lines interconnecting those components. A comparison between two multiplexers of Figs 1.2(a) and (b) is given in Table 1.1. It is seen that the four-valued multiplexer has all the advantages cited above.

Table 1.1 The Comparison of Four-Valued and Two-Valued MUX.

information	four-valued	two-valued
Total transistors	256	378
Total pins	68	71
Information per line	4	2
Delay time	3	6
Interconnection	less	more
Circuit complexity	less	more
Circuit performance	higher	lower
Cost per function	lower	higher
IC size	smaller	bigger

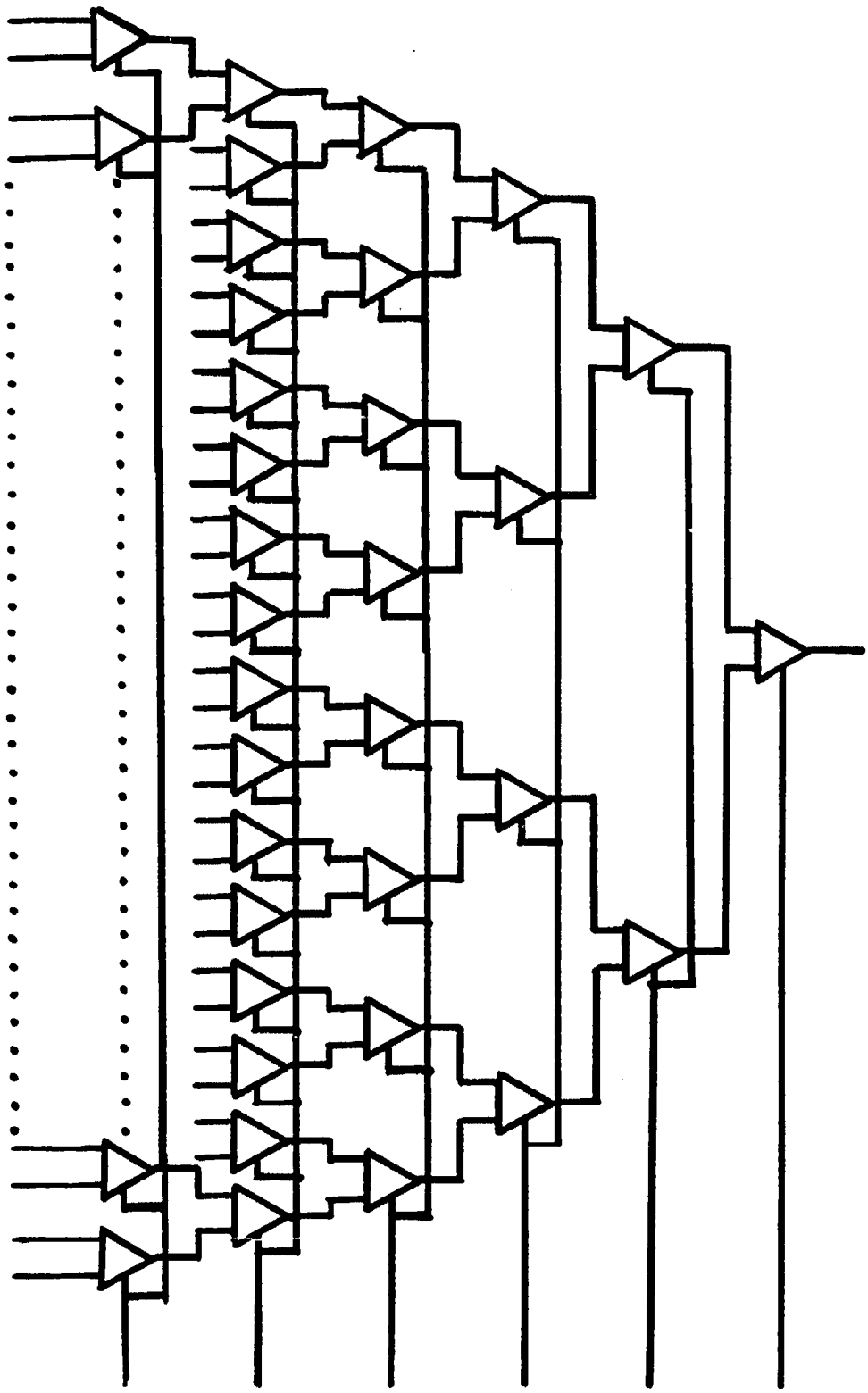


Fig. 1.2 (a) 64-to-1 Two-Valued Multiplexer.

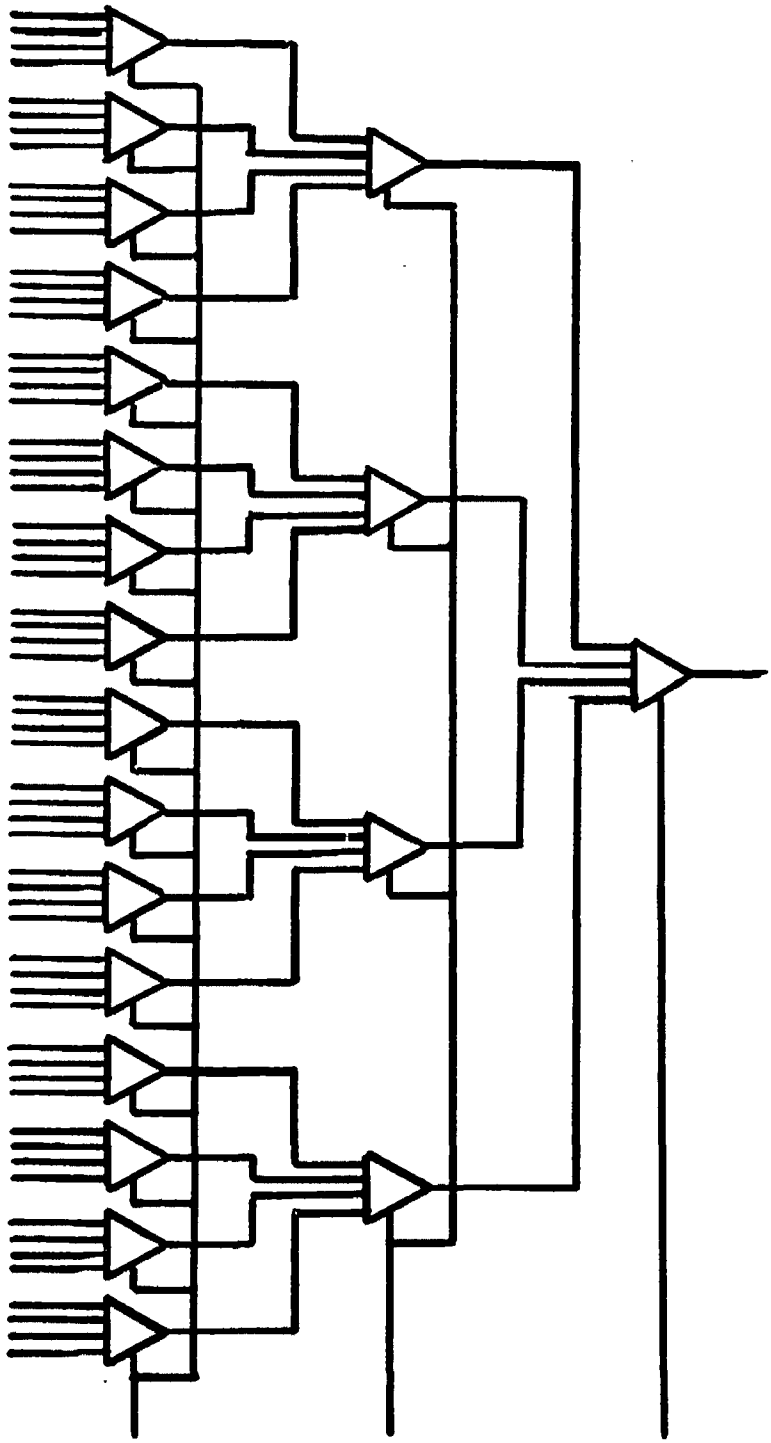
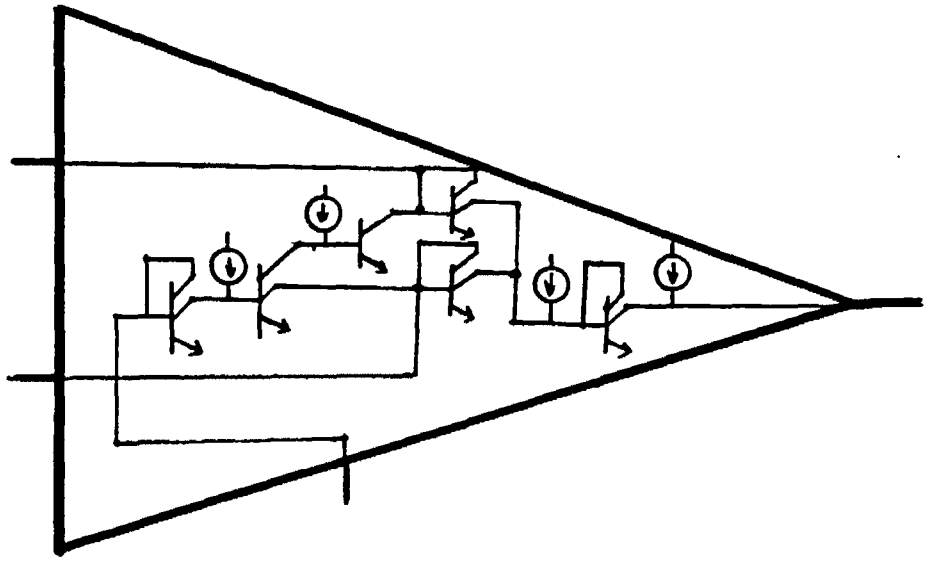
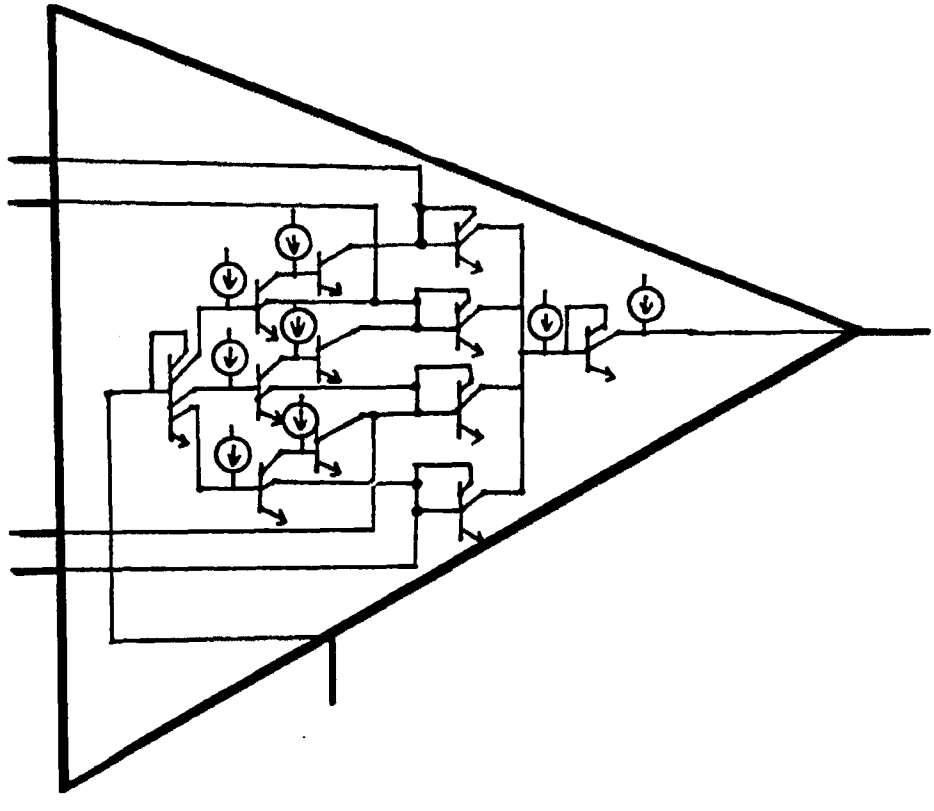


Fig. 1.2 (b) 64-to-1 Four-Valued Multiplexer.



(a)



(b)

Fig. 1.3 I^2L Circuit Single-Data-Selector Multiplexers
(a) two-valued multiplexer
(b) four-valued multiplexer.

1.3 MV LSI/VLSI System Design Process

Several methods have been applied to the design of relatively small MV digital systems [9,77,116,208], but to this date, no method for designing a large and complex MV system (LSI/VLSI) has been published. However, the design of MV LSI/VLSI may use approaches similar to those used in the two-valued LSI/VLSI system [39,46,144,152,197,224]. The purpose of this section is to suggest a process for designing a large and complex MV digital system.

Similar to the binary LSI/VLSI, the design process of an MV LSI/VLSI system may be divided into three major levels [39,224], i.e. behavioral level, structural level, and physical level. Each of these levels are further subdivided into many levels. The design actions are referred to as transformations which generate the intermediate design description of one level to its next level. Figure 1.4 shows the hierarchy of MV LSI/VLSI design process. This process may be carried out by a sequence of transformations performed in either of the two directions: the "bottom-up" and the "top-down".

down large unit into smaller ones, and should be independent of the specific technology as much as possible. It also seems to be more adequate to support design constraints when different parts of a system have to be constructed at different times or by different design groups. Since this design strategy is opposite that of the bottom-up strategy, the main disadvantage of the top-down strategy is that by breaking down large units into smaller ones under technical constraints, the strategy poses a combinatorial problem with an exponential growth function and also the transformation sequence yields an excessive number of different solutions.

Since a design using strictly a top-down or bottom-up strategy very rarely occurs in practice, real developments normally make an alternate use of both design strategies or refer to the information extracted from the other branch to properly and efficiently guide the process.

The hierarchical design approach will be used in developing the design process for an MV LSI/VLSI system. This approach is used with appropriate merging of levels to accomplish the design of LSI/VLSI systems. It proceeds in a top-down sequence with bottom-up detailed implementation and addresses both functional and physical problems at each level. The hierarchical design approach also contains those three major levels as shown in Fig. 1.4.

1. The behavioral level, which is the top level of the design hierarchy, contains elements for a board

functional system description, requirements for interfacing units specifying performance and compatibility, and methods for partitioning the system into major functional blocks such as processors, memory, and I/O.

2. The structural level, where the system functions are defined as interconnections of blocks or modules.

3. The physical level, where the physical implementation and process technology may be considered in order to construct circuit elements and determine their behavior.

In most cases the behavioral and structural design processes go on concurrently, while the physical design process is done separately. However, it often occurs that the physical design process can have a tremendous influence on the behavioral or structural design of a system, necessitating many iterations during a design.

In this dissertation interest will be focused on structural level design, especially on the search of a new network structure and its synthesis method to achieve a reliable and design-economical MV LSI/VLSI system. The hierarchical structured design process starts from each subsystem which has been partitioned by the behavioral design process. It then makes a decision on which semiconductor technology, component module, and type of network structure will be used in the design of this subsystem which will

easily be implemented in the physical process. The process proceeds by selecting a design tool (a design automation tool is usually used in this process to reduce the designing time) that is suitable for the technology and the structure that have been chosen. This design tool should give all the facilities required by the design of MV LSI/VLSI logic and the synthesis of the subsystem which can be implemented in the physical design process. In order to ensure that the MV LSI/VLSI system is fault-free, the system must be testable. To eliminate external testing equipment and to reduce testing time, the technique of inserting built-in test circuits in the various parts of the partitioned subunits of the subsystem is recommended. The hierarchical structured design process presented in this dissertation is shown in Fig. 1.5 and explained below:

This design process is to be applied to each of the subsystems of the partition at the behavioral level. I^2L and single-data-selector Multi-Valued MUltipleXer (MVMUX) are chosen (Chapter 3) to be the semiconductor technology and the component module of the design, respectively. The tree-structured MVMUX is to be used in the synthesis of design functions (Chapter 3). As a design tool, the Multi-Valued Algorithmic State Machine (MVASM) chart is chosen (Chapter 5) for the reason that it is simple to "translate" the system design description directly into a tree-structured MVMUX network in a systematic way. For synthesizing the subsystem function, a heuristic MVMUX modular synthesis method is used

(Chapter 4), since this method provides a minimal network realization and is easy to apply manually as well as by a computer. Because the subsystem realization is a tree-structured MVMUX network, it can be partitioned into nearly identical subunits, which, in turn, can be further partitioned into nearly identical sub-subunits, etc. This allows this design method to have a special desirable feature: the built-in parallel testing capability (Chapter 7).

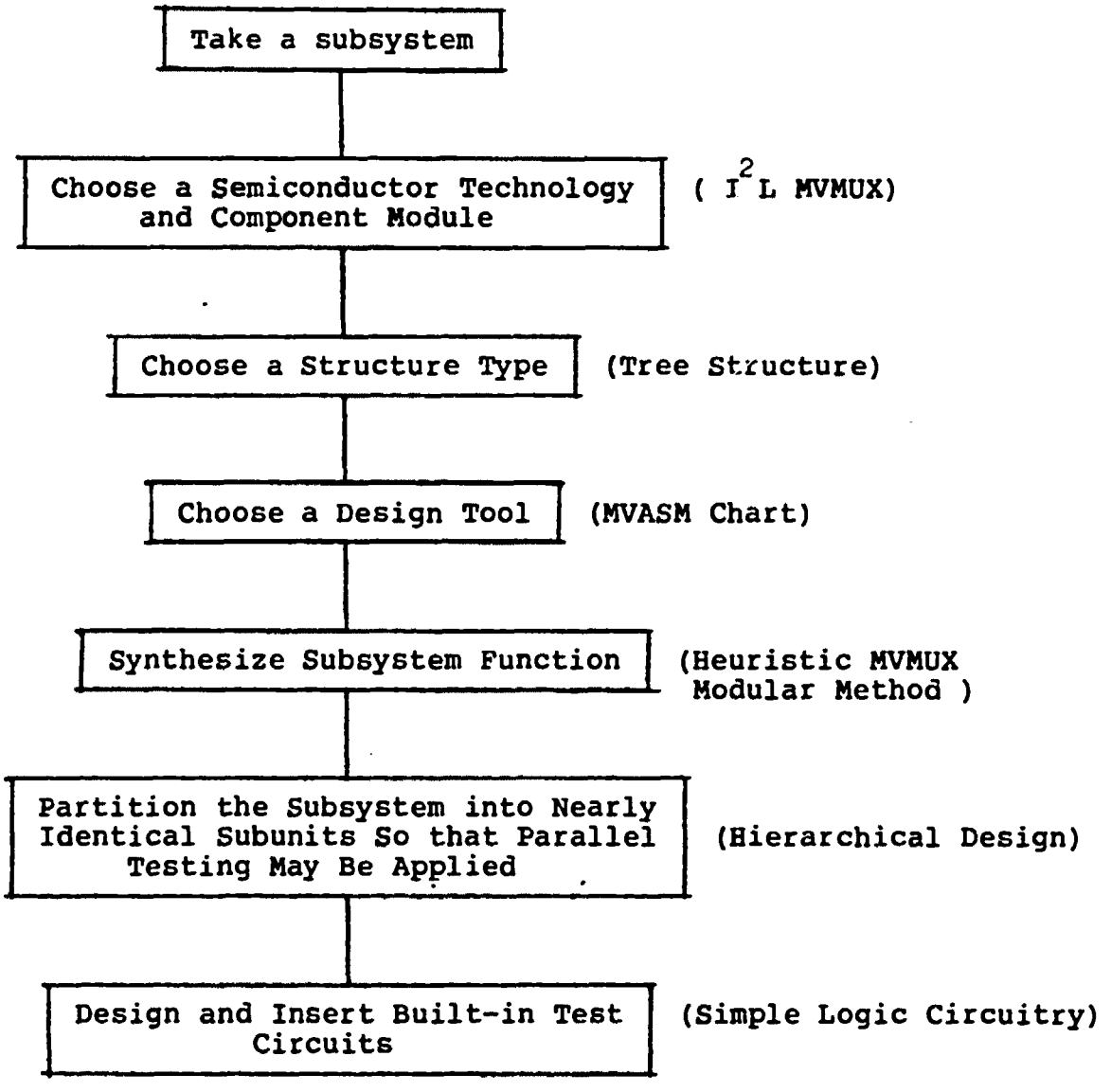


Fig. 1.5 The Hierarchical Structured Design Process.

1.4 Summary by Chapter

The dissertation consists of a total of eight chapters and two appendices. Chapter 2 discusses how the chip density of LSI/VLSI can be increased and how the complexity of LSI/VLSI can be reduced through the use of MV logic. Both conventional and MVMUX combinational logic designs are given

in Chapter 3. The synthesis of tree-structured MVMUX network is discussed in Chapter 4. Chapter 5 presents a design of multi-valued memory component and a new technique called multi-valued algorithmic state machine which is used as a multi-valued logic design tool. Chapter 6 reviews several techniques used in structured design for testability of binary LSI/VLSI circuits. A new proposed hierarchical design of MV LSI/VLSI with parallel built-in testing capability is introduced in Chapter 7. Chapter 8 concludes all the discussions in this dissertation. Two appendices are included: Appendix A gives a summary of past relevant work related to the MV logic, and Appendix B summarizes circuit technologies in MV logic circuits.

CHAPTER II

IMPROVING THE CHIP DENSITY AND COMPLEXITY OF LSI/VLSI CIRCUITS USING MULTI-VALUED LOGIC

Recently, the rapid evolution of semiconductor technology has allowed logic designers to design a complex digital system and put it in a single Integrated Circuit (IC) chip. In general, the IC chip may be divided into four types- Small Scale Integrated (SSI) circuit (gates less than 10), Medium Scale Integrated (MSI) circuit (10 < gates < 100), Large Scale Integrated (LSI) circuit (100 < gates < 1000) , and Very Large Scale Integrated (VLSI) circuit (gates > 1000). The LSI and VLSI (LSI/VLSI) circuits are considered as a complicated digital circuit here.

Practically, to this date two classes of semiconductor technology are used in integrated circuits: bipolar and MOS technology. Recently, the size of individual MOS transistors has been eliminated as a theoretically limiting factor in the production of LSI/VLSI circuits - nor is there even a nearby limit to the reduction in geometries of bipolar devices [26]. Both in principle can be made 10 or 20 times smaller, and both in theory have the same speed for a given active region size (base for bipolar, gate for MOS), since

the switching speed of either is a measure of the propagation time of carriers through the active region. However, the major limitation of device size right now is the fabrication technology which is still in a development stage. In addition, reducing the size of the device causes the increasing complexity of the circuit; therefore, it is quite difficult to manage the complexity and thus, design methodology is needed.

2.1 Limits of LSI/VLSI Circuits

In order to design the LSI/VLSI chip more efficiently, the physical, technological and complexity constraints have to be improved. However, there are some limitations [153] of these constraints.

(1) Physical Limits

This type is the fundamental limitation which is determined by the laws of physics, such as speed of light, entropy (irreversibility), uncertainty principle and thermal energy. This fundamental phenomena presents barriers to switching speed and power dissipation that cannot be surpassed.

(2) Technological Limits

This type of limitation depends on material constants, fabrication techniques and electrical parameters. The constraints imposed by these considerations can often be cir-

cumvented by using new materials, lower operating temperatures, structure changes, better cooling techniques, and other forms of device and circuit cleverness.

(a) Material constants include electrical and thermal conductivity, mobility, dielectric constants, saturation velocity, and dopant solubility.

(b) Limits associated with fabrication techniques involve doping fluctuations, processing radiation, defects, layer thickness uniformity and pattern edge roughness, bias and tolerance, and reduction of high temperature processing cycle (i.e., the diffusion coefficient-time product).

(c) Constraints relating to electrical parameters include oxide and junction breakdown, tunneling, hot electron injection, avalanche multiplication punch through conduction, small geometry effects, and nuclear radiation effects.

(3) Complexity limits

This type of limitation relates to the designer's inability to design circuitry involving very large numbers of components. This could also be thought of as the limit of the human conceptual ability. Complexity includes product definition, design time, engineering changes, testing, on-chip redundancy, computer-assisted design, and packaging.

To date, there is still some way to go before the "absolute" physical limits of LSI/VLSI are reached. In order to exploit fully the advantages of semiconductor technology and push the current physical limits set by today's technology closer to the absolute physical limits set by the law of physics, one should find means to improve the existing semiconductor technology and find better ways of handling the complex logic design problem of LSI/VLSI. The major advantages of LSI/VLSI are high performance and low cost. From previous discussion, the low cost can be achieved through high production volumes and by increasing the number of components per chip (chip density). The performance of the circuit can be improved by reducing the device capacity and by shortening propagation delay time. Therefore, it can be seen that by increasing the levels of performance and decreasing the production cost of semiconductor devices and circuits, the technology is improved and the complexity can be easily managed.

2.2. Technological Improvement

In order to increase the performance of the circuit, the smaller dimension (submicron) of the device must be improved. It should be mentioned that the device size reduction, which is termed "scaling", relates to the type of lithographic tool needed to accomplish dimensional shrinkage. Clearly, no single technological achievement will be sufficient to meet this goal. For example, advanced litho-

graphic technology is not sufficent by itself to fabricate the devices with smaller features. Other technologies, such as solid-state physics, device modeling, processing techniques, materials growth, analytical techniques, device design, and circuit architecture will also be required to improve. These technologies are developed by different groups of designers as follows [104]:

1. Solid-state physicists and device modelers developed two and three-dimensional models of the semiconductor devices and models of corresponding fabrication process for these devices.

2. Device designers developed layout styles to minimize the interdevice capacitance and maximize the output current; they also developed device test structures.

3. Lithographers fine tuned their electron-beam and optical techniques and developed new electron-beam, X-ray, ion-beam, and optical equipment to meet the eventual production requirements of the new devices.

4. Processing groups developed both a low-temperature device fabrication process and high-accuracy pattern transfer techniques needed to fabricate submicron devices.

5. Materials groups developed high-reliability, high-conductivity interconnection materials, while also studying the semiconductor-device failure modes.

Establishing an experimental device-processing line insured that the new device designs, new materials, and new processes actually work.

6. Analytical groups worked with the materials groups to improve their own understanding of the new materials and processes using different techniques, such as Auger electron spectroscopy, scanning-electron microscopy, transmission-electron microscopy, etc..

Recently, there are several technologies that try to improve the submicrometer (submicron) LSI/VLSI circuits fabrication. These technologies include X-ray, electron-beam and ion-beam lithography, dry etching, resists and process, materials growth, etc.[45,104].

(a) X-ray lithography technology

X-ray lithography appeared to have promise as a submicron pattern transfer technology; it offers a strong economic advantage over electron-beam lithography for high-volume applications. X-ray lithography is a proximity lithography pattern technique. It was expected to have advantages over both the industry standard optical lithography and high-resolution electron-beam pattern generators in the following areas [104]:

1. Increased throughput by parallel exposure.

2. Improved control of diffraction effects.

3. Minimized feature broadening due to electron scattering.

4. Relative simplicity of X-ray equipment.

5. Source wavelength and brightness.

6. Automated wafer handling.

7. High-quality mask and resist characteristics.

However, the major problem has been in devising a sensitive enough resist combined with a reliable bright source to give exposures of less than a minute. Also, small uncontrolled distortions in the mask substrate during and after patterning have caused significant distortions in the pattern [64]. X-ray systems have been used to make many devices with 2-μm features and a few in the 1-μm range. As the resolution and registration tolerances have tightened to keep ahead of the growing optical technology, the X-ray mask distortions for full field exposure have improved, but have continued to be a problem. With the current registration tolerances in the 0.1-0.2 μm range, smaller x-ray masks with the step-and-repeat exposures of large wafers are being studied. The use of step and repeat will complicate the x-ray systems and increase their cost compared to that of other lithographic systems.

(b) Electron-Beam Lithography Technology

Electron-beam lithography is extensively used for mask production today and the direct-write electron-beam systems are being readied for the submicron device fabrication development and low-volume fabrication. These are due to electron lithography which offers higher resolution because of the small wave length of the 10-25 keV electrons used [12] and because it is not limited by diffraction. Recently, several companies have begun developing electron direct writing systems. The primary advantages of electron-beam over contact-printing and projection-printing techniques for direct slice writing are [197]

1. the elimination of masks and mask defects

2. the alignment accuracy achievement

3. the fast turnaround of computer-controlled imaging

4. the superior resolution capability

5. The geometry size compensation.

However, the major disadvantage to date has been the low throughput and the requirement for different resists and processes. In addition, the resolution is limited by the electron scattering and by the resists.

(c) Ion-Beam Lithography Technology

Ion-beam lithography is in a very early stage of

development. It can be expected to have at least three advantages over electron-beam lithography [170]: 50-100X greater resist sensitivity, no proximity effects, and maskless processing. This means that very-high-resolution configurations could be structured by the direct-write ion-beam lithography using single resist layers on thick substrates or with no resists at all with the proper ion sources.

The ion-beam exposure of resists is a promising technique for the replication of patterns having submicron features. Proton ion-beam exposure of resists was stimulated by the possibility of having a high-resolution lithography technique that would not require the long exposure times which is necessary for the X-ray. Protons do not diffract significantly, they produce low energy secondary electrons when interacting with the matter, and can be performed into highly collimated beams. In addition, simple, commercial available sources can provide sufficient current to expose even the most insensitive resists in a fraction of a second [45].

(d) Dry Etching Technology

In the process of submicron device fabrication, conventional wet-etching technology and isoplasma etching technology are not applicable to the process because of large undercutting and poor controllability in pattern width [69]. Therefore, a new technology known as dry-etching techniques, such as plasma etching and reaction-ion etching have been

developed to produce anisotropic profiles and good dimension control in the process of submicron device fabrication. These properties are especially desired for transferring high-resolution patterns created with X-ray, electro-beam or ion-beam lithography [212].

(e) Resists and Process Technology

Resists are temporary layers applied onto the workpiece only for imaging purposes. After pattern transferring onto the active layer (insulator or semiconductor material), the resist is removed (stripped) in a solvent or an oxidizing solution. The most common resists are an organic solution applied on workpiece (wafer) by spinning, and dried by baking at a suitable temperature referred as "prebake temperature." The thickness of the dried resist layer depends largely on the concentration of solids in the solution and on the spinning speed. After baking the resist-coated wafer, the desired pattern is exposed in the lithographic tool, and the resist is developed. The most important distinction between resists concerns the pattern polarity after development [61].

In fact, the resist materials and processing will, by necessity, follow the trends of lithographic tool development for improvements in resolution and throughput. The improvement of resists and the process technology are discussed in Hatzakis' paper [61].

(f) Materials Growth Technology

One of the most important parameters which limit the performance of the semiconductor technology is the properties of the materials. New device structures or scaled-down versions of existing devices often require improvements in the state of the art of materials growth. Further reading on materials growth can be found in Economou's paper [45].

2.3 Chip Density and Management of the Complexity Improvements

In general, an LSI/VLSI chip may be regarded as an assemblage of three types of components [40]:

1. Active devices (transistor, MOS, MESFET) which occuppy about 10% of the chip area.

2. Passive isolation (oxide, dielectrics) which covered about 20% of the chip area.

3. Passive wiring (metal, polysilicon) which occupied about 70% of the chip area.

One way to increase the chip density is by scaling down the dimension of the components. The other way is by reducing the passive wiring area; the information content of each connection must be increased. The scaling down dimension strongly depends on the semiconductor technology which has already been discussed. The reduction of the passive wiring will be considered here. In order to increase the

information content in each connection, the time multiplexing method and the level multiplexing method may be used.

1. The time multiplexing method: this method is mainly limited to interchip pin connections, for example, the data terminals and address terminals of Intel 8085 microprocessor [110] use this method.

2. The level multiplexing method: this method reduces the interchip circuitry, therefore, interchip pin connections are also reduced, for example, to carry four information data i.e. 0, 1, 2, 3, if a binary logic system is used, at least 2 wires are required; but only one wire is needed when using the four levels multiplexing method. Therefore, about 50% of interchip connections and 50% for interchip pin connections are reduced.

It can be concluded that the latter method is the best choice to be considered for the IC chip area reduction. As of this date, the multi-valued logic system is the only technique that can be applied in the level multiplexing method. Therefore, a multi-valued logic system provides a solution which not only reduces the complexity of the system, but also increases the chip density when compared with the two-valued logic system.

As the technology in the device dimension is improved, the complexity level of circuit design for LSI/VLSI requires

a new type of design methodology. This new type of methodology must be able to manage the complexity of the LSI/VLSI circuit which includes the function definition, the architectural description, the logic interpretation, the circuit design, the physical layout design, the wafer fabrication, and the testing with verification and validation at each of these levels. This requires proper partitioning of the design technique into subcircuits of manageable size. Furthermore, in designing any fault-free complex circuit, it is necessary to have an efficient and economical testing and a verification method with reasonable testing time. This may be achieved by using built-in test and verification circuits which will be discussed in Chapter 7.

To conclude, two solutions for the improvement of the LSI/VLSI chip are presented in this dissertation: the chip density and the management of the complexity. The hierarchical structured design approach could be used to manage the complexity of the system and the multi-valued logic system could be used to improve the chip density.

CHAPTER III

MULTI-VALUED COMBINATIONAL LOGIC DESIGN

3.1 Multi-Valued Algebra

It is well known that binary switching theory and logical design have been based on Boolean Algebra as the mathematical model [168]. Multi-valued switching theory and logical design has been based on Post algebras [149] and their extension [47,95].

Definition 3.1: Binary operations:

Let $x, y \in Q = \{0,1,2,\ldots,p\}$. Where $p = m-1$.

(a) MAX operation: $x + y = MAX(x,y)$

i.e. $x + y = x$ for $x \geq y$

$= y$ for $x \leq y$

Fig. 3.1 Illustrates the Table of this MAX Operation.

x \ y	0	1	2	3	4	.	.	.	.	p
0	0	1	2	3	4	.	.	.	.	p
1	1	1	2	3	4	.	.	.	.	p
2	2	2	2	3	4	.	.	.	.	p
.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.
p	p	p	p	p	p	.	.	.	.	p

Fig. 3.1 Truth Table of MAX(x,y).

(b) MIN operation: $x \cdot y = MIN(x,y)$

i.e. $x \cdot y = x$ for $x \leq y$

$= y$ for $x \geq y$

Fig. 3.2 Illustrates the Table of this MIN Operation.

x \ y	0	1	2	3	4	.	.	.	.	p
0	0	0	0	0	0	.	.	.	.	0
1	0	1	1	1	1	.	.	.	.	1
2	0	1	2	2	2	.	.	.	.	2
.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.
p	0	1	2	3	4	.	.	.	.	p

Fig. 3.2 Truth Table of MIN(x,y).

From Definition 3.1, the following properties are satisfied.

Idempotent: $x + x = x$ $\qquad x \cdot x = x$

Commutation: $x + y = y + x$ $\qquad x \cdot y = y \cdot x$

Association: $(x + y) + x = x + (y + z)$

$(x \cdot y) \cdot z = x \cdot (y \cdot z)$

Absorption: $x + x \cdot y = x$ $\qquad x \cdot (x + y) = x$

Distribution: $x + y \cdot z = (x + y) \cdot (x + z)$

$x \cdot (y + z) = x \cdot y + x \cdot z$

Null element: $x + 0 = x$ $\qquad x \cdot 0 = 0$

Universal element: $x + p = p$ $\qquad x \cdot p = x$

Where $x, y, z \in Q = \{0, 1, 2, \ldots, p\}$ and 0, p are constants.

Definition 3.2: Unary operation:

Let $x, a, b \in Q = \{0, 1, 2, \ldots, p\}$ and $a \leq b$

(a) LITERAL operation (LIT)

$$x(a,b) = p \quad \text{if } a \leq x \leq b$$
$$= 0 \quad \text{otherwise}$$

Fig. 3.3 Illustrates the table of LIT operation. A graphical illustration of LIT function x(2,4) where m = 6 is shown in Fig. 3.4.

x \ ab	00	01	. . .	0p;	11	12	. . .	1p;	. . .	pp
0	p	p	. . .	p;	0	0	. . .	0;	. . .	0
1	0	p	. . .	p;	p	p	. . .	p;	. . .	0
2	0	0	. . .	p;	0	p	. . .	p;	. . .	0
.	.	.	. . .	.;	.	.	. . .	.;	. . .	.
.	.	.	. . .	.;	.	.	. . .	.;	. . .	.
.	.	.	. . .	.;	.	.	. . .	.;	. . .	.
p	0	0	. . .	p;	0	0	. . .	p;	. . .	p

Fig. 3.3 Truth Table of x(a,b).

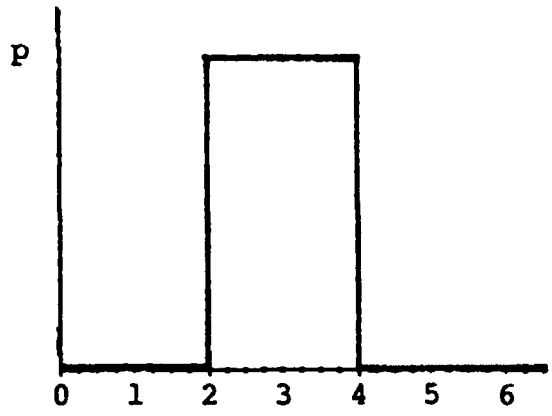


Fig. 3.4 Graphical Representation of x(2,4).

(b) COMPLEMENT LITERAL operation (CLIT)

$$\overline{x(a,b)} = p \quad \text{if } b < x < a$$

$$= 0 \quad \text{otherwise}$$

A graphical illustration of CLIT function x(2,4) where m = 6 is shown in Fig. 3.5.

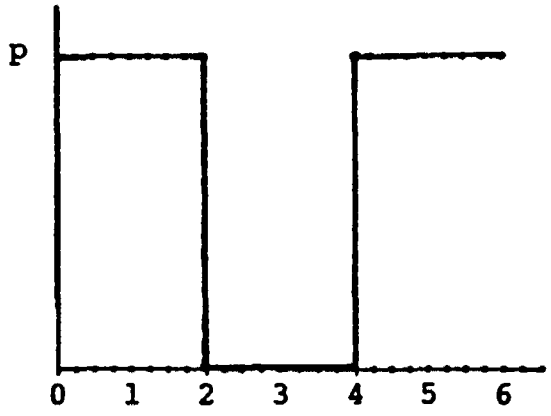


Fig. 3.5 Graphical Representation of $\overline{x(2,4)}$.

(c) CYCLE operation

$$\overrightarrow{x}^{a} = (x \text{ plus } a) \bmod m$$

Fig. 3.6 Illustrates the table of this CYCLE operation.

x \ a	0	1	2	3	4	.	.	.	.	p
0	0	1	2	3	4	.	.	.	.	p
1	1	2	3	4	5	.	.	.	.	0
2	2	3	4	5	6	.	.	.	.	1
.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.
p	p	0	1	2	3	.	.	.	.	p-1

Fig. 3.6 Truth Table of $\vec{x}^{a}$.

(d) COMPLEMENT operation (COMP)

$$\bar{x} = p - x$$

Fig. 3.7 Illustrates the table of this COMP operation.

x	$\bar{x}$
0	p
1	p-1
2	p-2
.	.
.	.
.	.
p	0

Fig. 3.7 Truth Table of COMP(x).

These operations constitute a functional complete set as shown in Table. 3.1. Fig. 3.8 illustrates their logic gate symbols.

Table 3.1 Functional Complete Set for 3-Valued Logic.

x = 0	x = 1	x = 2	Entry
0	0	0	0
0	0	1	$1 \cdot x(2,2)$
0	0	2	$x(2,2)$
0	1	0	$1 \cdot x(1,1)$
0	1	1	$1 \cdot x$
0	1	2	x
0	2	0	$x(1,1)$
0	2	1	$x(2,2)$
0	2	2	$x(0,0)$
1	0	0	$1 \cdot x(0,0)$
1	0	1	$1 \cdot x(1,1)$
1	0	2	$1 \cdot x(0,0) + x$
1	1	0	$1 \cdot x(0,1)$
1	1	1	1
1	1	2	$1 + x$
1	2	0	$\vec{x}^1$
1	2	1	$1 + x(1,1)$
1	2	2	$1 + x(1,2)$
2	0	0	$x(0,0)$
2	0	1	$\vec{x}^2$
2	0	2	$\overline{x(1,1)}$
2	1	0	$\bar{x}$
2	1	1	$1 + x(0,0)$
2	1	2	$x + \bar{x}$
2	2	0	$x(2,2)$

2	2	1	$1 + \overline{x(2,2)}$
2	2	2	2

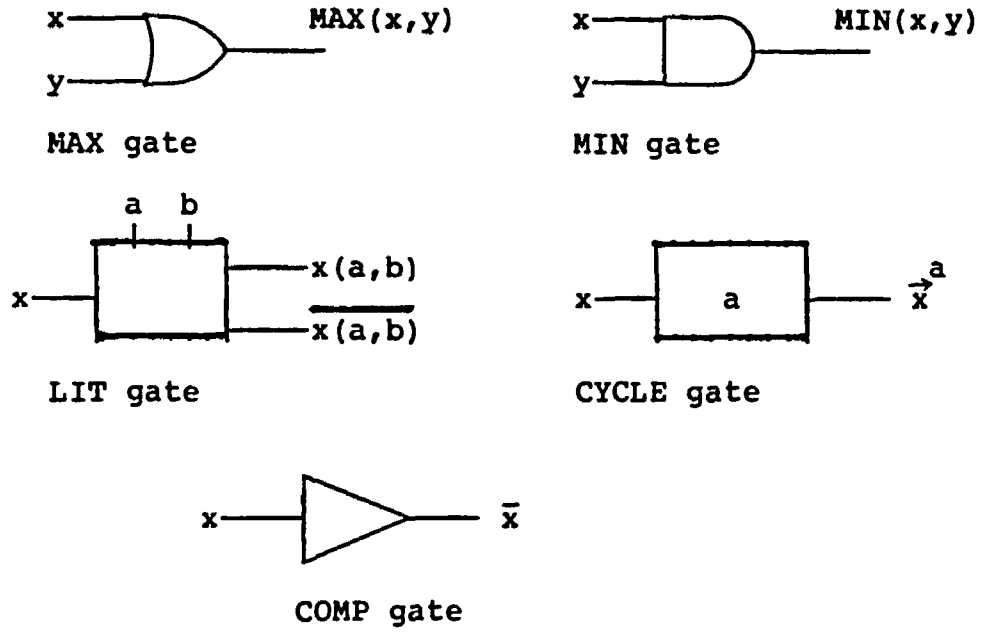


Fig. 3.8 Multi-Valued Logic Gate Symbols.

3.2 Multi-Valued Functions

In an m-valued logic system there are m^{m^n} functions for n variables. A set of operators which can describe all of these functions is called a complete set. Table 3.1 shows a complete set of 3-valued one-variable function. Any m-valued switching function can be represented by a truth table or map table as shown in Fig. 3.9.

x	f(x)
0	0
1	0
2	1
3	2
4	0
5	3
6	6
7	0

(a)

x_1 x_2	$f(x_1, x_2)$
0 0	0
0 1	1
0 2	0
1 0	0
1 1	2
1 2	0
2 0	1
2 1	2
2 2	0

(b)

x

0	1	2	3	4	5	6	7
0	0	1	2	0	3	6	0

(c)

x_2 \ x_1	0	1	2
0	0	0	1
1	1	2	2
2	0	0	0

(d)

Fig. 3.9 Multi-Valued Function Representation.

It may also be expressed by the sum-of-products in canonical form.

$$f(x_1, x_2, \ldots, x_n) = \sum_{i=0}^{n} a_i \cdot L_i \qquad \text{Eq. 3.1}$$

Where the a_i is a constant in the range (0, 1, 2,...,p) and L is the MIN of the individual LIT or COMP of variables $x_1, x_2, \ldots, x_n$. For example, the canonical form of one-variable 8-valued switching function f(x) as shown in Fig. 3.9(a) can be written as the following:

$$
\begin{aligned}
f(x) &= f(0)\cdot x(0,0) + f(1)\cdot x(1,1) + f(2)\cdot x(2,2) \\
&\quad + f(3)\cdot x(3,3) + f(4)\cdot x(4,4) + f(5)\cdot x(5,5) \\
&\quad + f(6)\cdot x(6,6) + f(7)\cdot x(7,7) \\
&= 0\cdot x(0,0) + 0\cdot x(1,1) + 1\cdot x(2,2) + 2\cdot x(3,3) \\
&\quad + 0\cdot x(4,4) + 3\cdot x(5,5) + 6\cdot x(6,6) + 0\cdot x(7,7) \\
&= 1\cdot x(2,2) + 2\cdot x(3,3) + 3\cdot x(5,5) + 6\cdot x(6,6)
\end{aligned}
$$

Eq. 3.2

The other example, the canonical form of two-variable 3-valued switching function $f(x_1,x_2)$ as shown in Fig. 3.9 (b) can be written as

$$
\begin{aligned}
f(x_1,x_2) &= 1\cdot x_1(0,0)\cdot x_2(1,1) + 2\cdot x_1(1,1)\cdot x_2(1,1) \\
&\quad + 1\cdot x_1(2,2)\cdot x_2(0,0) + 2\cdot x_1(2,2)\cdot x_2(1,1)
\end{aligned}
$$

Eq. 3.3

3.3 Multi-Valued Function Minimization

The minimization of multi-valued functions has been considered by several researchers [5,129,173]. Many different methods of MV function minimization have been programmed [147,174,181]. The map minimization method of Allen and Givone [5] is used here because it is simple and easy to use. In this method the truth table of MV function is transformed to map table representation. The following step-by-steps minimize the MV function by finding the prime implicants of an MV function:

Step 1: Set the logic value inside each cell containing

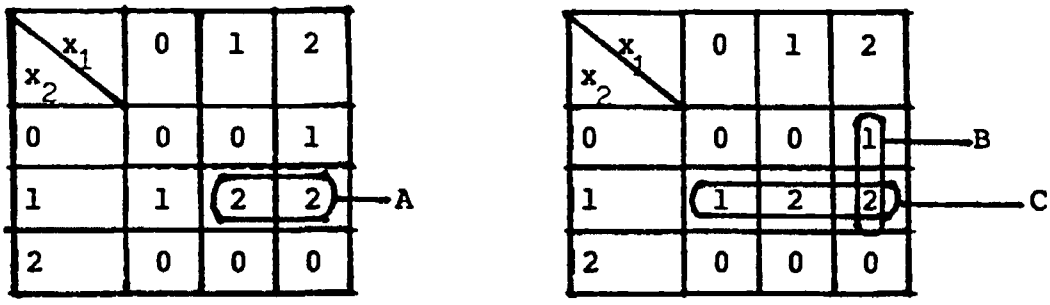
a don't care (-) to the logic value p. Let k be an index starting with the value p.

Step 2: Find all n-dimensional rectangular groupings of cells which have the logic value k, or higher, and are not totally contained in any large rectangular grouping of cells having the logic value k, or higher. The product terms which correspond to these groupings are prime implicants if and only if they subsum no term previously found to be a prime implicant.

Step 3: Set k to k-1.

Step 4: If k > 0, repeat from step 2. Otherwise, if just prime implicants have been found; terminate the process.

Example 3.1 Fig. 3.10 shows the minimization of the function of the map table in Fig. 3.9 (d). First, no don't care cells are found, only one rectangular grouping of cell value 2 is found as shown in Fig. 3.10 (a). Next all rectangular grouping containing a logic 1 or higher are shown in Fig. 3.10 (b) along with their corresponding product terms. Thus, there are 3 prime implicants of this function A, B, and C. The sum of these prime implicants is the minimization function.



(a) (b)

$$f(x_1, x_2) = A + B + C$$

Fig. 3.10 Map Minimization of Example 3.1.

3.4 Combinational Logic Design Using MVSSIs and MVMSIs

There are two basic approaches to the design of combinational MV logic: MV algebra approach and actual circuit approach. The MV algebra approach uses algebraic theory to realize the arbitrary functions by interconnecting the basic gates operator together. A drawback of this approach is that the efficiency (in terms of interconnect or packing density) is not evaluated since realizations for more complex functions, such as adders or ALUs are not considered. In fact, it is questionable whether this technique would lead to efficient designs, since the circuits realizing the basic connectives are typically rather complex [114].

The actual circuit approach was proposed by McCluskey [117]. In this approach the design of arbitrary functions is developed by studying a variety of actual circuit designs and then abstracting the connectives that produced efficient design. A measure of the success of the technique is the fact that it results in circuits that are as efficient as

those "invented" by an experienced designer.

An MV combinational logic design considered here is the design of a "BLACK BOX" to implement an MV function. This function requires an MV algebraic approach as a design tool to realize the circuit. The function is usually specified as a truth table or map table which lists the output value of a function for any combination of its input values. Fig. 3.11 shows the output $f(x_1, x_2, \ldots, x_n)$ of a black box as functions of the inputs $x_1, x_2, \ldots, x_n$.

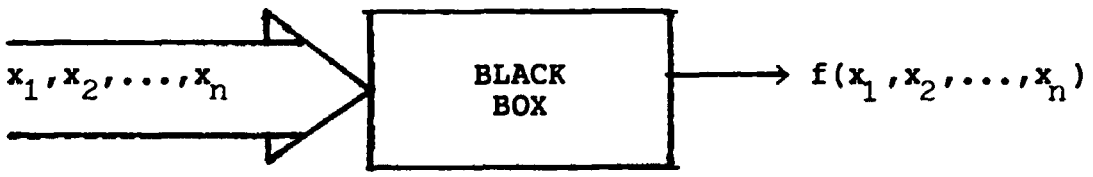


Fig. 3.11 Black Box.

The important idea required for logic designers in implementing this black box is to derive the function to get an optimal or near optimal solution of the interconnection of a set of MV gates with respect to some criterion. Criteria may be cost, speed, design flexibility, availability of complex logic functions, logic levels, noise immunity, power supply voltages, power dissipation operating temperatures, and testability of the circuit. In this section, two techniques will be considered in MV combinational logic design. One is using Multi-Valued Small Scale Integrated circuits (MVSSI), i.e. MAXs, MINs, LITs, COMPs gates, the

other is using Multi-Valued Medium Scale Integrated circuits (MVMSI) i.e. MVMUXs.

A. MV Combinational Logic Design Using MVSSI Gates

The MV algebra which was presented in the previous chapter will be used as a tool to implement MV combinational logic design using MVSSI gates. The basic procedure of MV combinational logic design presented here is based on a minimization criterion to achieve a minimized circuit realization using the primitive gates. A step-by-step procedure of designing MV combinational logic circuits is presented as follows:

1. Define the MV input and output variables of the system. Represent the various status of the input and the output by their respective variables.

2. Construct the truth table describing the system function.

3. Convert the truth table into a map table.

4. Minimize the function using the map table and the procedure given in the previous section.

5. Realize the minimized function using MVSSIs.

In order to understand the above procedure better, the following design example is given.

Example 3.2: Design a ternary full adder using MVSSIs. A full adder is a combinational circuit that forms the arithmetic sum of three input 3-valued variables. It consists of three inputs and two outputs.

Step 1: Two of the input variables, denoted by X and Y, represent the two significant 3-valued digits to be added. The third input Cin represents the carry from a previous lower significant position. Two outputs are necessary because the arithmetic sum of three ternary digits range in value from 0 to 4, and ternary 3 and 4 need two digits. The two outputs are designated by the symbols S for sum and Cout for carry. The ternary variable S gives the value of the sum. The ternary variable Cout gives the output carry.

Step 2: Construct the truth table of the ternary full adder as given in Table 3.2.

Table 3.2 Truth Table of the Ternary Full Adder.

X	Y	Cin	S	Cout
0	0	0	0	0
0	0	1	1	0
0	0	d	d	d
0	1	0	1	0
0	1	1	2	0
0	1	d	d	d
0	2	0	2	0
0	2	1	0	1
0	2	d	d	d
1	0	0	1	0
1	0	1	2	0
1	0	d	d	d
1	1	0	2	0
1	1	1	0	1
1	1	d	d	d
1	2	0	0	1
1	2	1	1	1
1	2	d	d	d
2	0	0	2	0
2	0	1	0	1
2	0	d	d	d
2	1	0	0	1
2	1	1	1	1
2	1	d	d	d

2 2 0	1 1
2 2 1	2 1
2 2 d	d d

Step 3. Convert the truth table into the map table as shown in Figs. 3.12 and 3.13.

Cin \ XY	00	01	02	10	11	12	20	21	22
0	0	1	2	1	2	0	2	0	1
1	1	2	0	2	0	1	0	1	2
2	d	d	d	d	d	d	d	d	d

Fig. 3.12 Map Table of S.

Cin \ XY	00	01	02	10	11	12	20	21	22
0	0	0	0	0	0	1	0	1	1
1	0	0	1	0	1	1	1	1	1
2	d	d	d	d	d	d	d	d	d

Fig. 3.13 Map Table of Cout.

Step 4. Minimize the function using the map table as illustrated in Figs. 3.14 and 3.15.

Cin \ XY	00	01	02	10	11	12	20	21	22
0	0	1	(2)	1	(2)	0	(2)	0	1
1	(1)	(2)	0	(2)	0	(1)	0	(1)	(2)
2	d	d	d	d	d	d	d	d	d

Fig. 3.14 Map Minimization of S.

$$
\begin{aligned}
S = {} & X(0,0)\cdot Y(2,2)\cdot Cin(0,0) + X(0,0)\cdot Y(1,1)\cdot Cin(1,1) \\
& + X(1,1)\cdot Y(0,0)\cdot Cin(1,1) + X(1,1)\cdot Y(1,1)\cdot Cin(0,0) \\
& + X(2,2)\cdot Y(0,0)\cdot Cin(0,0) + X(2,2)\cdot Y(2,2)\cdot Cin(1,1) \\
& + 1\cdot X(0,0)\cdot Y(0,0)\cdot Cin(1,1) + 1\cdot X(0,0)\cdot Y(1,1) \\
& + 1\cdot X(0,0)\cdot Y(1,1) + 1\cdot X(1,1)\cdot Y(2,2)\cdot Cin(1,1) \\
& + 1\cdot X(2,2)\cdot Y(1,1)\cdot Cin(1,1) + 1\cdot X(2,2)\cdot Y(2,2)
\end{aligned}
$$

Cin \ XY	00	01	02	10	11	12	20	21	22
0	0	0	0	0	0	1	0	1	1
1	0	0	(1)	0	(1)	1	(1)	1	1
2	d	d	d	d	d	d	d	d	d

Fig. 3.15 Map Minimization of Cout.

$$
\begin{aligned}
Cout = {} & 1\cdot X(0,0)\cdot Y(2,2)\cdot Cin(1,1) + 1\cdot X(1,1)\cdot Y(1,1)\cdot Cin(1,1) \\
& + 1\cdot X(1,1)\cdot Y(2,2) + 1\cdot X(2,2)\cdot Y(0,0)\cdot Cin(1,1) \\
& + 1\cdot X(2,2)\cdot Y(1,2)
\end{aligned}
$$

Step 5. Realize the function minimized by MVSSIs as shown in Fig. 3.16. (cost = 389)

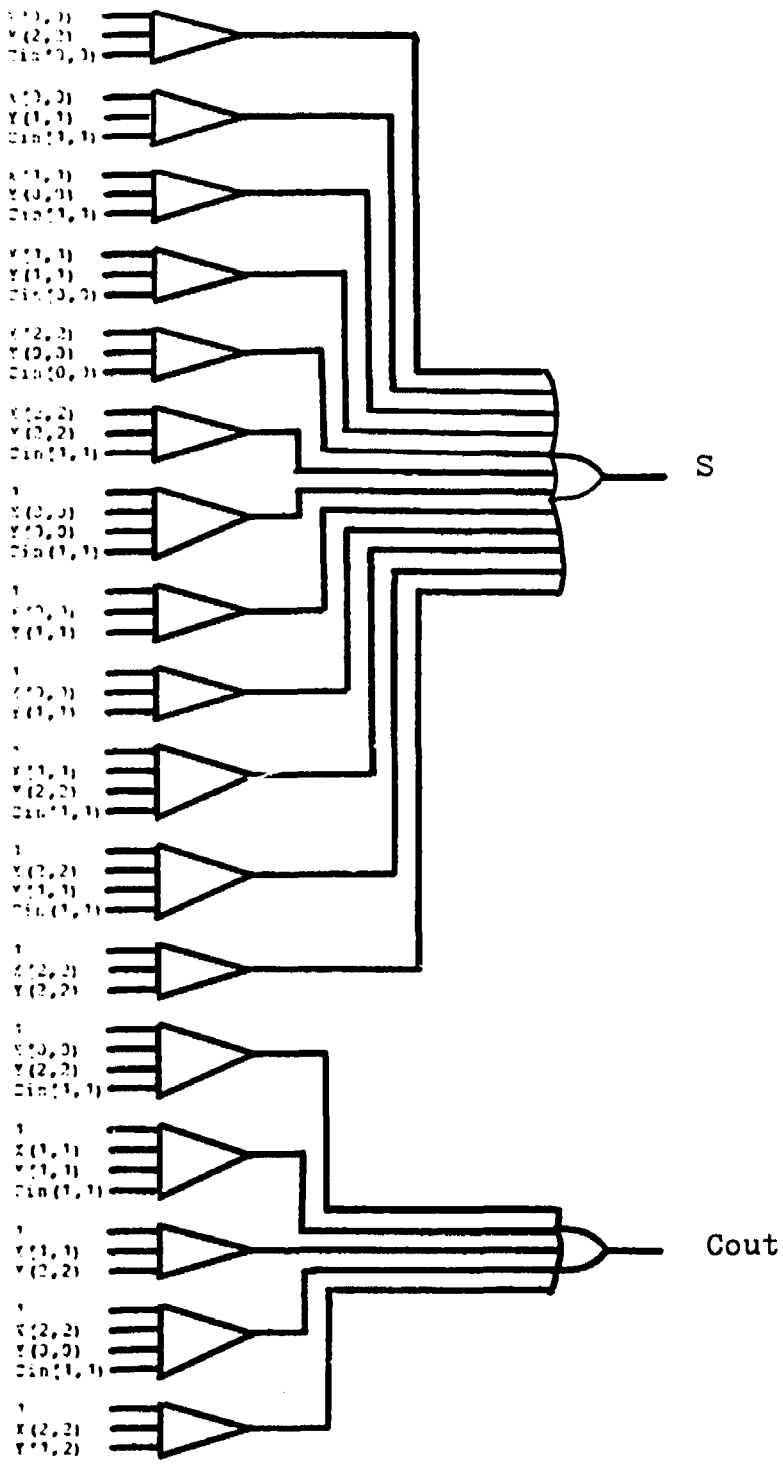


Fig. 3.16 MVSSI Circuit Realization of Ternary Full Adder.

B. MV Combinational Logic Design Using MVMSI

Based on the considerations of design flexibility, availability of complex logic functions, and cost per function criterion, MVMUX (which are MVMSI) are used in the MV combinational logic design. There are two types of methods to realize an MV function using MVMUX.

a. Constant-input Type Realization Method

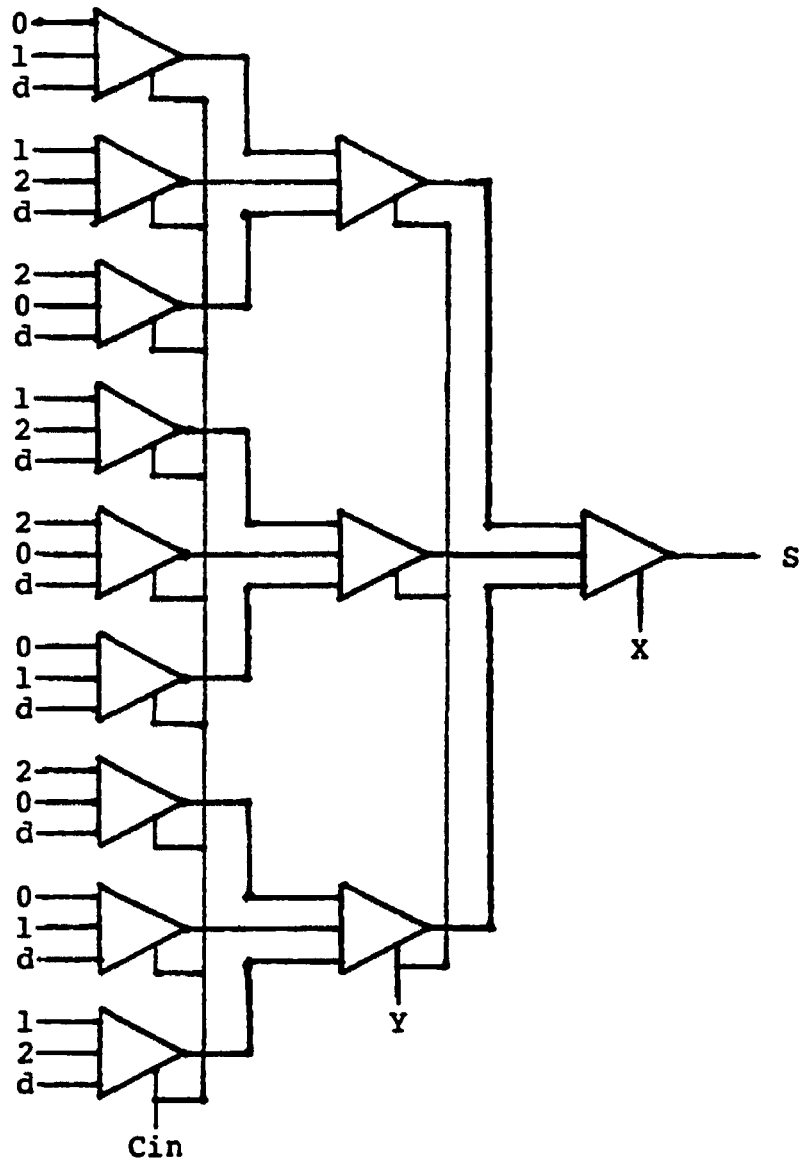
This method is very simple; it includes the following steps:

1. The same as steps 1 and 2 in subsection A. Let r be the number of input variables.

2. For each output variable, construct a r-level tree-structured MVMUX networks with m data inputs.

3. Enter the values of the output variables as the constant inputs of the MVMUX networks.

For example, to realize the truth table of the ternary full adder of Table 3.2, we need a two 3-level MVMUX networks, each has a total of $3^3 = 27$ inputs. The 27 constant inputs of these networks are the 27 rows of the output values of S and Cout. Fig 3.17 illustrates MVMUXs circuit realization of ternary full adder.


0
1
d
1
2
d
2
0
d
1
2
d
2
0
d
0
1
d
2
0
d
0
1
d
1
2
d
Cin
Y
X
S

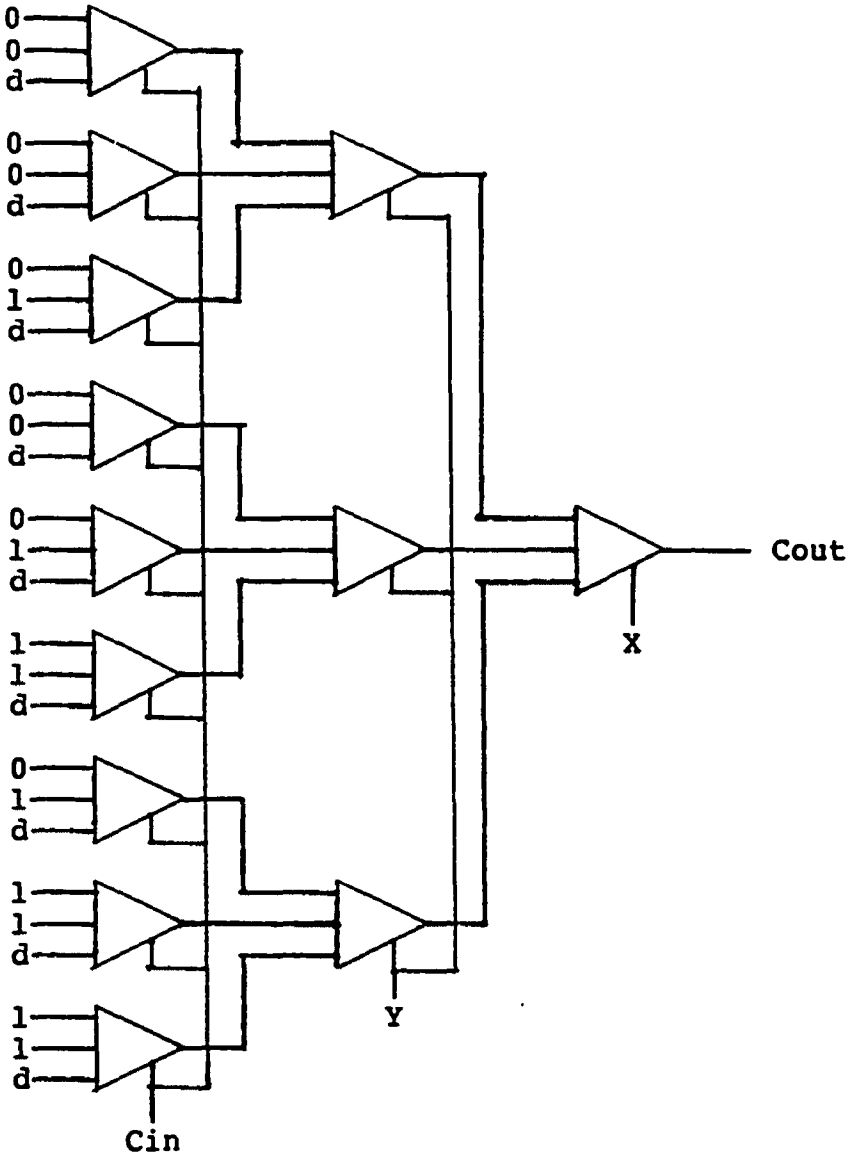


Fig. 3.17 MVMUXs Circuit Realization of Ternary Full Adder.

b. Variable-input Type Realization Method

In practice, the number of "data-select" inputs of MVMUX chip is limited. So it is often desirable to reduce the set of input variables to be connected to the "data-select" lines. A new technique using an MV "map-entered"

variables to reduce the number of "data-select" lines of MVMUX is introduced. This technique shows how the concept of map-entered variable may be used to considerably simplify the procedure for determining the data input conditions. The technique shows the reduction of the m-to-1 MVMUX and also reduces the number of levels in a tree-structured MVMUX network. The map-entered variable method has already been used to reduce the dimension map in 2-valued logic system [30]. This method is more complicated than the 2-valued case, because of the m^{m^n} . The MV map-entered variable method is used here to enter the map-entered variable as "data-input" of MVMUX so that the "data-select" lines may be reduced. In order to understand the method, a 3-valued logic example is given. Table 3.3 shows rules for eliminating a map variable of the map table for one variable, and its relative cost.

Table 3.3 Rules for Eliminating a Map Variable of the Map Table.

Value of function for map-entered variable location x = 0	x = 1	x = 2	Entry	Cost(I L)
0	0	0	0	0
0	0	1	1•x(2,2)	8
0	0	2	x(2,2)	4
0	1	0	1•x(1,1)	8
0	1	1	1•x	4
0	1	2	x	0
0	2	0	x(1,1)	4
0	2	1	x(2,2)	6
0	2	2	x(0,0)	5
1	0	0	1•x(0,0)	8
1	0	1	1•x(1,1)	9
1	0	2	1•x(0,0)+x	15
1	1	0	1•x(0,1)	8
1	1	1	1	0
1	1	2	1 + x	3
1	2	0	$\vec{x}^1$	5
1	2	1	1 + x(1,1)	7
1	2	2	1 + x(1,2)	7
2	0	0	x(0,0)	4
2	0	1	$\vec{x}^2$	5
2	0	2	$\overline{x(1,1)}$	5
2	1	0	$\bar{x}$	1
2	1	1	1 + x(0,0)	7

2	1	2	$x + \overline{x}$	4
2	2	0	x(2,2)	5
2	2	1	$1 + \overline{x(2,2)}$	8
2	2	2	2	0

The following procedure describes the synthesis of an MV function using variable-input type MVMUXs:

1. The same as steps 1, 2 and 3 in subsection A.

2. Use Table 3.3 to enter the values corresponding to the variable reduction into its cell. Repeat all possible variables and compare their cost, then select the lowest cost to implement the function.

3. Select the MVMUX to implement the function by entering the value in each cell position to the "data-input" and map variables to the "data-select" terminals.

Figures 3.18 and 3.19 show the map-entered variables of the ternary full adder. From these map-entered variables, when Cin is used as a entered variable the total cost is 99, when Y is used as a entered variable the total cost is 98. Therefore, the entered Y variable is chosen to implement this system as shown in Fig. 3.20.

XY

00	01	02	10	11	12	20	21	22
Cin	$\overrightarrow{Cin}^{1}$	Cin(0,0)	$\overrightarrow{Cin}^{1}$	Cin(0,0)	Cin	Cin(0,0)	Cin	$\overrightarrow{Cin}^{1}$

S

XY

00	01	02	10	11	12	20	21	22
0	0	Cin	0	Cin	1	Cin	1	1

Cout

Fig. 3.18 Use Cin as a Map-Entered Variable.

XCin

00	01	0d	10	11	1d	20	21	2d
Y	$\overrightarrow{Y}^{1}$	d	$\overrightarrow{Y}^{1}$	$\overrightarrow{Y}^{2}$	d	$\overrightarrow{Y}^{2}$	Y	d

S

XCin

00	01	0d	10	11	1d	20	21	2d
0	1·Y(2,2)	d	1·Y(2,2)	1 Y	d	1·Y	1	d

Cout

Fig. 3.19 Use Y as a Map-Entered Variable.

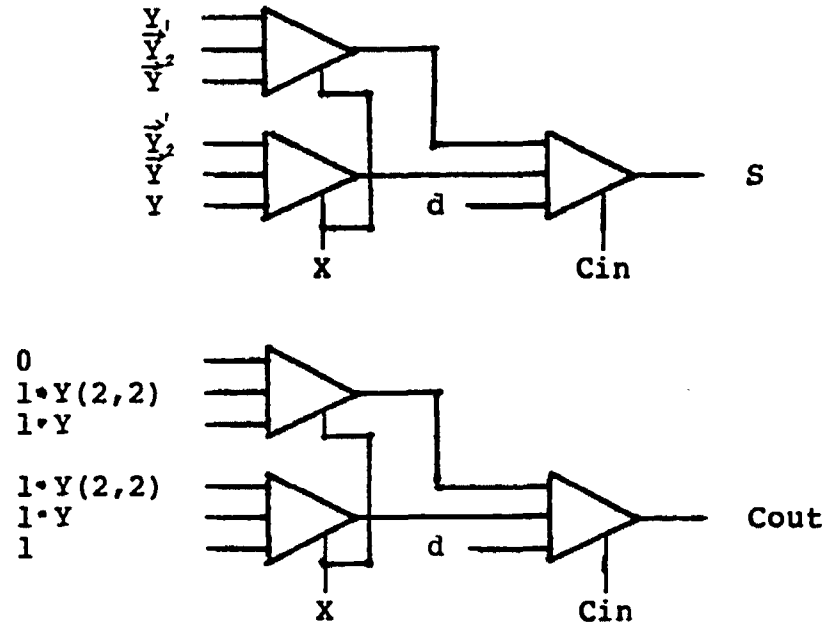


Fig. 3.20 Map-Entered Variable Circuit Realization.

In the above, two combinational logic design techniques, one using MVSSI gates and the other using MVMSI (MVMUX), are presented and their relative costs (number of transistors) (see Appendix B) are compared. It is found that for the realization of complex logic functions, the latter is not only more economical, but also more systematic and is easier to apply. Another MVMUX synthesis technique which is specially suitable for MV LSI/VLSI design will be discussed in the next chapter.

CHAPTER IV

SYNTHESIS OF MULTI-VALUED FUNCTION WITH ULMs

4.1 Introduction

Recent advances in integrated circuit technology and its potential advantage in logical design have motivated a search for synthesis techniques for logic networks using appropriate logic function packages as the modules or building blocks for two-valued [122,217,218] and for multi-valued systems [53,66,67,68,83,172]. In order to design MV LSI/VLSI circuits with near minimal components (to maximize the function per chip area), the synthesis of functions is necessary. In this thesis, the hierarchical tree-structured modular approach is chosen for the design of the MV LSI/VLSI because with this approach simple algorithms are available to realize any given logic function and are also easy to implement in an automation design. Moreover, the structure of this approach is easily partitioned into subunits, so that circuits can be rapidly and economically tested. The basic theory of the tree-structured modular approach and its synthesis methods are investigated in this chapter.

In general, the synthesis of an MV function may be

divided into two categories. One is two-level synthesis and the other is multiple-level synthesis. Two-level synthesis is often performed by minimizing the function in the map [6], cube [182], or tabular [199] representation. Two-level synthesis is only practical for very small systems involving few variables. However, a minimum for two-level minimization may not be optimum due to the fan-in limit for practical implementation [37].

Multi-level synthesis is often used to reduce the cost of the function [199] and in the synthesis of a large and complicated system. There are two possible ways to synthesize a function into a multiple-level: one is by factoring from a two-level function and the other is by using the decomposition function technique. It has been shown [24,25,127] that by extending the Ashenhurst [10] and Curtis [37] partitioned matrix technique a fanout-free network can be synthesized. A fanout-free network is a network in which the gate outputs and primary inputs connect to at most one gate input. The tree-structure of such a network implies that there is a unique path between each primary input and the network output. Thus, when testing the fanout-free network for correct operation, the fault-masking problem associated with the multiple paths does not occur. The absence of multiple paths also implies the absence of hazards. The fanout-free networks are of special interest when it is expensive to implement gates whose output drive has more than one gate input such as magnetic bubble logic and I^2L

multi-valued logic. Basically, there are two types of MV components used in multi-level circuits: primitive components and modular components. The primitive component synthesis is based on the minimization of primitive components [124,187,203,223]. This type is suitable for synthesis of relatively small circuits since the MV primitive logics are quite complex in themselves, and thus the result is extremely complex circuits for even simple MV functions of two or three variables [172].

The other type, the modular component synthesis is based on the minimization of the modular components. This type is suitable for synthesizing large and complicated circuits, mainly because the MV modular components are constructed from actual circuit approach [117], and can be easily realized the MV function in systematic process. In this thesis, our attention will be focused on the multi-level synthesis with modular component approach since our interest is mainly focused on the MV LSI/VLSI design. The multi-level synthesis with modular component approach has been studied by Kameyama and Higuchi [83], and Fang and Wojcik [53].

The method introduced by Kameyama and Higuchi [83] is an extension of the early work of binary system done by Meo, Yau, Tang, and Voith [122,198,218]. This method minimizes MVMUX by finding the compatible set of ULM implicants which contains the possible reduction of the maximum number of

MVMUXs. However, this method is quite complicated. Fang and Wojcik's method [53] searches for a minimal number of subfunctions of a given function and implements it by using multiple data selector MVMUXs. The synthesis starts from the primary input end to the single output end using certain common properties. The networks obtained by this method are non-fanout-free.

In this thesis, a new heuristic multi-level synthesis with modular component approach for synthesizing MV functions which is simple and systematic is presented. This method is different from all the previous methods in that (1) it guarantees a fanout-free tree-structured network, (2) the synthesis starts with the output end of the network, and (3) the procedure is simple and can be implemented by computer-aided design (CAD).

4.2 Theory of Tree-Structured Modular Network

The modular packages used in the tree-structured technique are of a universal logic module (ULM) type. The MVMUX which is one type of ULM will be considered here.

Definition 4.1: A Universal Logic Module (ULM) of n input variables is a logic function package that can be used to implement any logic function of up to n input variables by simply carrying its I/O terminal connections.

Definition 4.2: An MVMUX is a ULM which has one terminal data output, k terminals data selector input, and m^k

terminals data input; each data input can be selected by data selector as an output (illustrated in Fig. 4.1).

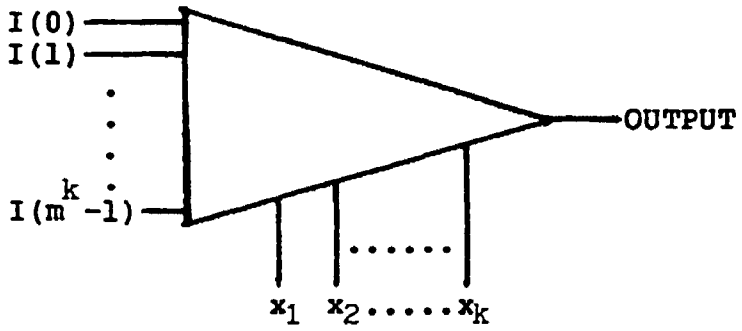


Fig. 4.1 MVMUX Diagram.

Definition 4.3: The interconnection of the MVMUXs in the following fashion is called a tree-structured MVMUX network (shown in Fig. 4.2)

1. There is only one MVMUX at the output level (1st-level)

2. Each data input and data input selector of MVMUX may be connected to the output of another MVMUX or an input variable or constant value.

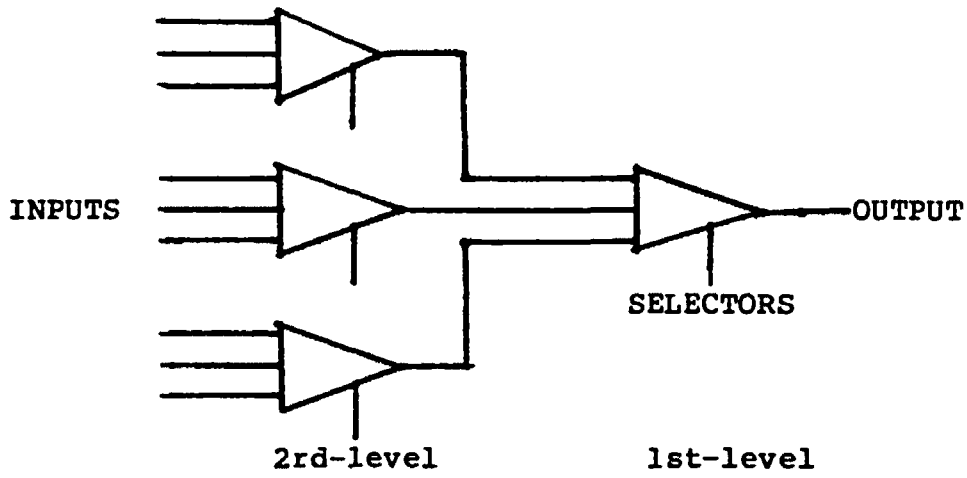


Fig. 4.2 Tree-Structured MVMUX Network Diagram.

Definition 4.4: In an m-valued logic system, let the truth value be the element of a set M = {0,1,2,...,m-1}, a $\in$ M, and x be a variable defined on M. The a and x are called multi-valued scalars. The $\underset{\sim}{a} = (a_1,a_2,\ldots a_n) = a_1a_2\ldots a_n$ and $\underset{\sim}{x} = (x_1,x_2,\ldots x_n) = x_1x_2\ldots x_n$ are called multi-valued constant and variable multi-valued vectors respectively.

Definition 4.5: The symbol $\sum(\cdot)$ denotes the maximum of the quantities $(\cdot)$.

Definition 4.6: Define

$$x_i(a_i) = \begin{cases} p & \text{for } x_i = a_i \\ 0 & \text{for } x_i \neq a_i \end{cases}$$

$\underset{\sim}{x}(a) = (x_1(a_1),x_2(a_2),\ldots x_n(a_n)) = x_1(a_1)x_2(a_2)\ldots x_n(a_n)$,

and $\text{MIN}[\underset{\sim}{x}(a)] = x_1(a_1)\ x_2(a_2)\ \ldots x_n(a_n)$,

where p = m-1.

Definition 4.7: The symbol $\underset{\sim}{\sum}(x,y,z)$ means x or y or z, let

"or" = $\bigvee$, therefore

$$\sum\!\!\!\!\bigvee(x,y,z) = x \vee y \vee z \qquad \text{Eq.(4.1)}$$

Definition 4.8: Define

$$\sum_{\underset{\sim}{x} = \underset{\sim}{0}}^{\underbrace{m-1}_{n}} (\cdot) = \sum_{x_1 x_2 \ldots x_n = 00\ldots0}^{(m-1)(m-1)\ldots(m-1)} (\cdot) \qquad \text{Eq.(4.2)}$$

and

$$\bigvee_{\underset{\sim}{x} = \underset{\sim}{0}}^{\underbrace{m-1}_{n}} f(x) = \bigvee_{x_1 x_2 \ldots x_n = 00\ldots0}^{(m-1)(m-1)\ldots(m-1)} f(x_1 x_2 \ldots x_n) \qquad \text{Eq.(4.3)}$$

Definition 4.9: Any m-valued k-selector MVMUX may be represented as

$$m\text{-MUX}[I(\underset{\sim}{x}) : \underset{\sim}{x}] = \left[\bigvee_{\underset{\sim}{x} = \underset{\sim}{0}}^{\underbrace{m-1}_{n}} I(\underset{\sim}{x})\right]_{\underset{\sim}{x}} \qquad \text{Eq.(4.4)}$$

where $\underset{\sim}{x} = x_1 x_2 \ldots x_n$, is the MVMUX selectors

$I(\underset{\sim}{x}) = I(x_1 x_2 \ldots x_n)$, MVMUX inputs

and the subscript $\underset{\sim}{x}$ of $[\ ]_{\underset{\sim}{x}}$ indicates that $\underset{\sim}{x}$ is the vector of the MVMUX.

For example, the four-valued multiplexer with one input data selector is represented by

$$4\text{-MUX}[I(\underset{\sim}{x}) : \underset{\sim}{x}] = [I(0) \vee I(1) \vee I(2) \vee I(3)]_{\underset{\sim}{x}}$$

which means that which of the four inputs selected depends

upon the value of $\underset{\sim}{x}$, i.e. the I(0), I(1), I(2), or I(3) is selected when $\underset{\sim}{x}$ = 0, 1, 2, or 3, respectively. Fig. 4.3 shows this MVMUX.

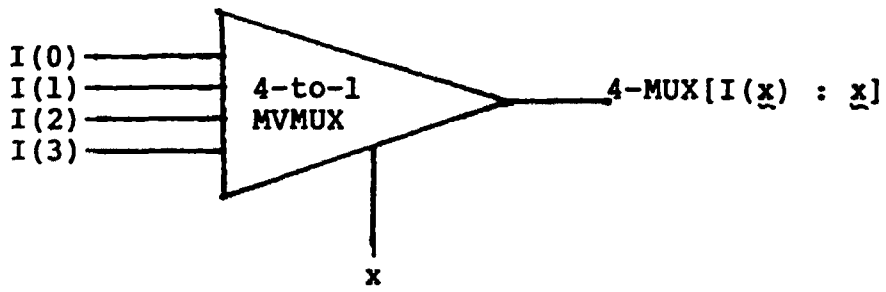


Fig. 4.3 4-Valued One-Variable (selector) MVMUX.

Definition 4.10: The tree-structured MVMUX network can be represented by

$$\text{m-MUX}[I(\underset{\sim}{x}) : \underset{\sim}{x}] = [\underbrace{\underbrace{(m-1)(m-1)\cdots(m-1)}^{k_1\ k_2\ \quad k_j}}_{\underset{\sim}{x}_1\underset{\sim}{x}_2\cdots\underset{\sim}{x}_j = \underset{\sim}{0}\underset{\sim}{0}\cdots\underset{\sim}{0}} I(\underset{\sim}{x}_1, \underset{\sim}{x}_2, \cdots \underset{\sim}{x}_j)]^{\underset{\sim}{x}_1\underset{\sim}{x}_2\cdots\underset{\sim}{x}_j} \quad \text{Eq.(4.5)}$$

The general tree-structured MVMUX network diagram is shown in Fig. 4.4.

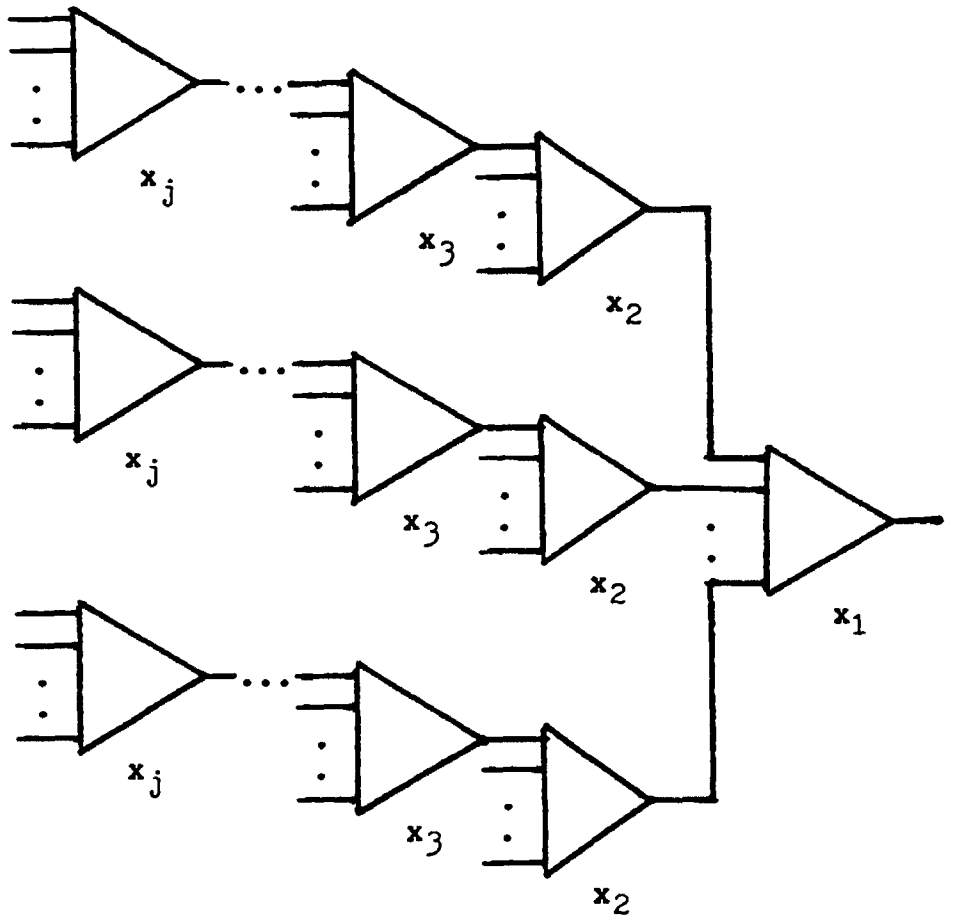


Fig. 4.4 General Tree-Structured MVMUX Network Diagram.

For example, a three-valued four-variable (x_1, x_2, x_3, x_4) tree-structured MVMUX network which has $\underset{\sim}{x}_1 = x_1$ as a selector in the 1st-level and $\underset{\sim}{x}_2 = x_2x_3x_4$ in the 2nd-level of the MVMUX tree-structured can be represented as

$$3\text{-MUX}[I(\underset{\sim}{x}) : \underset{\sim}{x}] = [\overset{(\underset{\sim}{2}),(\underset{\sim}{26})}{\underset{\underset{\sim}{x}_1,\underset{\sim}{x}_2=(\underset{\sim}{0},\underset{\sim}{0})}{\circledcirc}} I(\underset{\sim}{x}_1,\underset{\sim}{x}_2)]_{\underset{\sim}{x}_1,\underset{\sim}{x}_2}$$

$$= [I(0,0) \textcircled{v} I(0,1) \textcircled{v} I(0,2) \textcircled{v} I(1,0) \textcircled{v} \ldots I(2,26)]$$

Fig. 4.5 shows this MVMUXs representation.

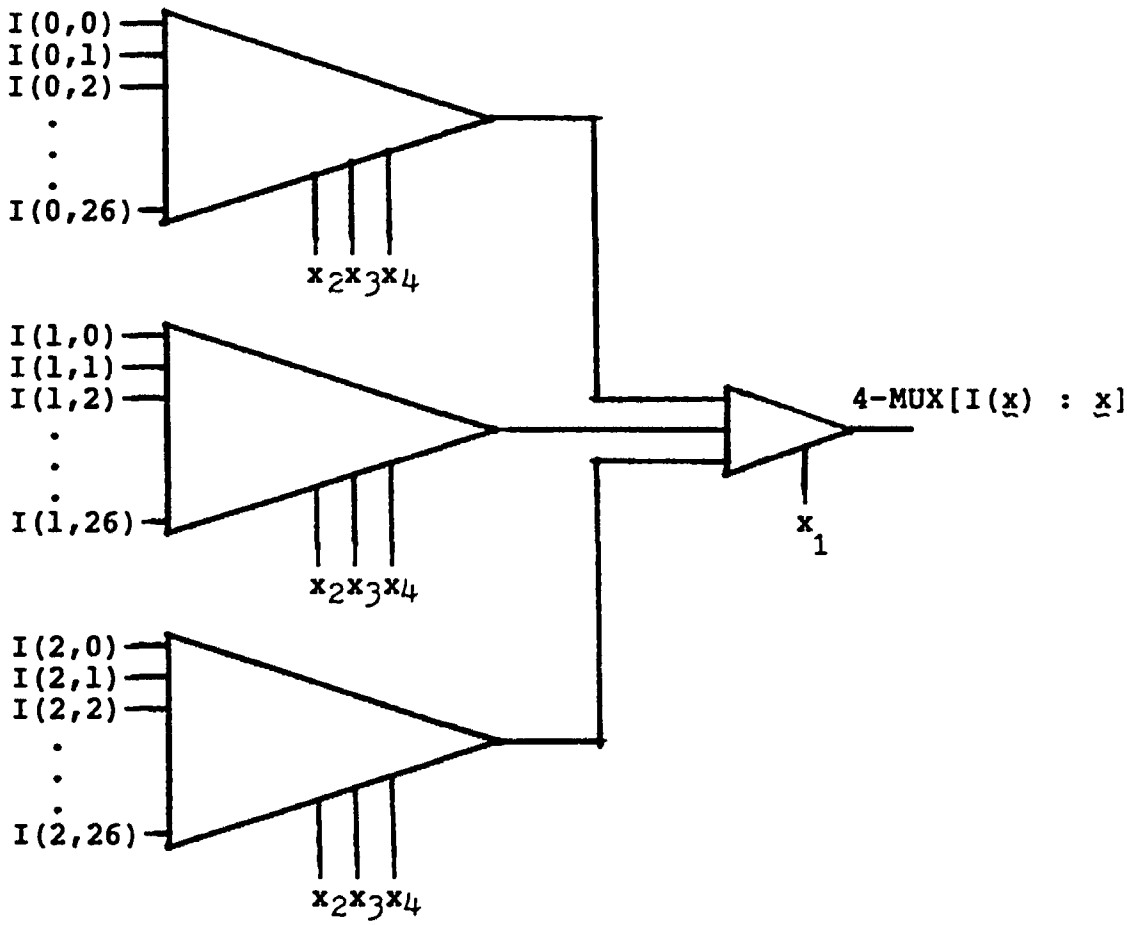


Fig. 4.5 MVMUX Tree-Structured Representation

Theorem 4.1: Any m-valued function $f(\underline{x})$ of n variables (i.e. $\underline{x} = x_1, x_2, \ldots x_i, \ldots x_n$) can be expanded with respect to any one of the n variables, x_i, as follows:

$$f(\underline{x}) = \left[\sum_{x_n(a)=0}^{m-1} [x_i(a) \cdot f(x_1, x_2, \ldots, a, \ldots, x_n)] \right] \qquad \text{Eq. (4.6)}$$

This expansion can be repeated until the function is a constant.

Table 4.1 Arbitrary Truth Table Represents f(x).

x	0	1	2	3
f(x)	2	3	2	0

Example 4.1 A 4-valued one variable function f(x) represented by Table 4.1 can be written as

$$f(x) = x(0)\cdot f(0) + x(1)\cdot f(1) + x(2)\cdot f(2) + x(3)\cdot f(3)$$

Table 4.2 Arbitrary Truth Table Represents $f(x_1, x_2, x_3)$.

	x_1		
x_2x_3	0	1	2
00	0	2	2
01	2	0	1
02	1	0	2
10	0	0	1
11	1	2	1
12	0	0	0
20	1	2	0
21	0	0	1
22	1	1	0

$f(x_1, x_2, x_3)$

Example 4.2 A 3-valued function $f(x_1, x_2, x_3)$ represented by Table 4.2 can be expanded as

(a) about x_1,

$$f(\underset{\sim}{x}) = x_1(0)\cdot f(0,x_2,x_3) + x_1(1)\cdot f(1,x_2,x_3) + x_1(2)\cdot f(2,x_2,x_3)$$

Eq. (4.7)

(b) about x_2,

$$\begin{aligned} f(\underset{\sim}{x}) = {} & x_1(0)\cdot[x_2(0)\cdot f(0,0,x_3) + x_2(1)\cdot f(0,1,x_3) + x_2(2)\cdot f(0,2,x_3)] \\ & + x_1(1)\cdot[x_2(0)\cdot f(1,0,x_3) + x_2(1)\cdot f(1,1,x_3) + x_2(2)\cdot f(1,2,x_3)] \\ & + x_1(2)\cdot[x_2(0)\cdot f(2,0,x_3) + x_2(1)\cdot f(2,1,x_3) + x_2(2)\cdot f(2,2,x_3)] \end{aligned}$$

Eq. (4.8)

(c) about x_3,

$$
\begin{aligned}
f(\underset{\sim}{x}) = {} & x_1(0)\cdot[x_2(0)\cdot\{x_3(0)\cdot f(0,0,0) + x_3(1)\cdot f(0,0,1) \\
& \qquad + x_3(2)\cdot f(0,0,2)\} \\
& + x_2(1)\cdot\{x_3(0)\cdot f(0,1,0) + x_3(1)\cdot f(0,1,1) \\
& \qquad + x_3(2)\cdot f(0,1,2)\} \\
& + x_2(2)\cdot\{x_3(0)\cdot f(0,2,0) + x_3(1)\cdot f(0,2,1) \\
& \qquad + x_3(2)\cdot f(0,2,2)\}] \\
& + x_1(1)\cdot[x_2(0)\cdot\{x_3(0)\cdot f(1,0,0) + x_3(1)\cdot f(1,0,1) \\
& \qquad + x_3(2)\cdot f(1,0,2)\} \\
& + x_2(1)\cdot\{x_3(0)\cdot f(1,1,0) + x_3(1)\cdot f(1,1,1) \\
& \qquad + x_3(2)\cdot f(1,1,2)\} \\
& + x_2(2)\cdot\{x_3(0)\cdot f(1,2,0) + x_3(1)\cdot f(1,2,1) \\
& \qquad + x_3(2)\cdot f(1,2,2)\}] \\
& + x_1(2)\cdot[x_2(2)\cdot\{x_3(0)\cdot f(2,0,0) + x_3(1)\cdot f(2,0,1) \\
& \qquad + x_3(2)\cdot f(2,0,2)\} \\
& + x_2(1)\cdot\{x_3(0)\cdot f(2,1,0) + x_3(1)\cdot f(2,1,1) \\
& \qquad + x_3(2)\cdot f(2,1,2)\} \\
& + x_2(2)\cdot\{x_3(0)\cdot f(2,2,0) + x_3(1)\cdot f(2,2,1) \\
& \qquad + x_3(2)\cdot f(2,2,2)\}]
\end{aligned}
$$

Eq.(4.9)

From the above example, it follows immediately that

Theorem 4.2: Any MV function $f(x_1,x_2,\ldots,x_n)$ can be expanded as:

(a)

$$f(\underset{\sim}{x}) = \sum_{x_1(a_1)=0}^{p} (x_1(a_1)(\sum_{x_2(a_2)=0}^{p} x_2(a_2)..(\sum_{x_n(a_n)=0}^{p} x_n(a_n) \cdot f(a_1, a_2, \ldots a_n))..)) \qquad \text{Eq.(4.10)}$$

(b) Since all $\underset{\sim}{x}$ are linearly independent of each other, (a) can also be written as:

$$f(\underset{\sim}{x}) = \sum_{x_1(a_1)=0}^{p} \sum_{x_2(a_2)=0}^{p} .. \sum_{x_n(a_n)=0}^{p} x_1(a_1)\ x_2(a_2)..x_n(a_n) \cdot f(a_1, a_2, .. a_n) \qquad \text{Eq.(4.11)}$$

Definition 4.11: The expression of Eq.(4.11) is called the canonical form of $f(\underset{\sim}{x})$

Theorem 4.3: Any multi-valued function can be written as

$$f(\underset{\sim}{x}) = \sum_{\underset{\sim}{a} = \underset{\sim}{0}}^{\underset{\sim}{m^n-1}} MIN[\underset{\sim}{x}(a)] \cdot f(\underset{\sim}{a}) \qquad \text{Eq.(4.12)}$$

This theorem can be proved by using Theorems 4.1 and 4.2.

Theorem 4.4: Any function can be realized by a tree-structured MVMUX representation in Eq.(4.5).

Proof: By applying the following transformation between the terms of the canonical form of the function of Eq.(4.11) and the modules of the tree-structured network,

Function notation		Multiplexer notation
$\sum$	$\longrightarrow$	$\sum_{\otimes}$
$\underset{\sim}{x}(a)$	$\longrightarrow$	$\underset{\sim}{x}$
MIN[x(a)]	$\longrightarrow$	[ ]
f(a)	$\longrightarrow$	$I(\overset{\underset{\sim}{x}}{\ })$

we immediately obtain Eq.(4.5).

Note that Eq.(4.11) is valid with the summations (maximum operations) over the a in any order. For example,

$$f(x) = \sum_{x_n(a_n)=0}^{p} .. \sum_{x_2(a_2)=0}^{p} \sum_{x_1(a_1)=0}^{p} x_1(a_1) \cdot x_2(a_2) .. x_n(a_n) \cdot f(a_1, a_2, \ldots a_n) \qquad \text{Eq.(4.13)}$$

Definition 4.12: The realization $f(\underset{\sim}{x})$ in the form of Fig. 4.4 is called a canonical realization of $f(\underset{\sim}{x})$.

The number of MVMUX in a canonical realization of an m-valued n-variable single-selector tree-structured network is given in the following theorem.

Theorem 4.5: An arbitrary logic function can be completely expressed with a maximum of $(m^n - 1)/(m-1)$ MVMUXs [83].

Proof: Suppose the 1st-level of tree-structured MVMUX network requires h MVMUXs, by the mathematic induction method

the 1st-level used h MVMUXs

the 2nd-level used hm^1 MVMUXs

the 3nd-level used hm^2 MVMUXs

. . .

. . .

. . .

the nth-level used $hm^{(n-1)}$ MVMUXs

therefore, the total number of MVMUXs (S) is

$$S = h + hm^1 + hm^2 + \ldots hm^{(n-1)}$$

$$S = (hm^n - h)/(m-1)$$

since the 1st-level of tree-structured MVMUX network has only one MVMUX, which implies h = 1, thus,

$$S = (m^n - 1)/(m-1)$$

Now derive a general formula for computing the number of MVMUX in a canonical realization using any multiple-selector MVMUX.

Theorem 4.6: For a given values $\underset{\sim}{x} = \underset{\sim}{a}$, the output z of the MVMUX is equal to

$$z = [\ \overset{n}{\underset{x = 0}{\overset{m-1}{\otimes}}}\ I(x)\ ]_{x} = I(a_1, a_2, \ldots a_n) \qquad \text{Eq.(4.15)}$$

Proof

By Definition 4.8

$$[\ \overset{n}{\underset{x = 0}{\overset{m-1}{\otimes}}}\ I(x)\ ]_{x} = [\ \overset{(m-1)(m-1)\ldots\ (m-1)}{\underset{x_1 x_2 \ldots x_n = 00\ldots0}{\otimes}}\ I(x_1, x_2, \ldots x_n)\ ]_{x_1 x_2 \ldots x_n} \qquad \text{Eq.(4.16)}$$

for $x = a$ and by Definition 4.7

$$= I(a_1, a_2, \ldots a_n)$$

Definition 4.13: Let $N[1; k_1, k_2, \ldots k_1]$ be a 1-level tree-structured MVMUX network. The numbers of selectors of the ith-level MVMUXs are the same, that is equal to k .

Theorem 4.7: Any k-line-selector MVMUX can be replaced by a tree-structured MVMUX network. Suppose the numbers of the selector lines of the MVMUXs of each level are the same. Let them be $k_1, k_2, \ldots$ and k_1 which are less than k and $k = k_1 + k_2 + k_3 + \ldots k_1$

Proof: Let N_1 and N_2 be the original and tree-structured MVMUX networks respectively. In N_1 ,any k-selector MVMUX can select any one of the m inputs and send it to the output of MVMUX. Let $I(x)$ be the inputs of the MVMUX. In N_2, the total number of inputs of the network is given as follows:

$$I(\underset{\sim}{x}) = \left[\overset{(\underset{\sim}{m}^{k_1}-\underset{\sim}{1}),(\underset{\sim}{m}^{k_2}-\underset{\sim}{1}),\ldots(\underset{\sim}{m}^{k_1}-\underset{\sim}{1})}{\searrow} I(\underset{\sim}{x}_1,\underset{\sim}{x}_2,\ldots\underset{\sim}{x}_1) \right]^{\underset{\sim}{k}_1,\underset{\sim}{k}_2,\ldots\underset{\sim}{k}_1}_{\underset{\sim}{x}_1,\underset{\sim}{x}_2,\ldots\underset{\sim}{x}_1 = \underset{\sim}{0},\underset{\sim}{0},\ldots,\underset{\sim}{0}} \quad \text{Eq.(4.17)}$$

the number of inputs of the 1st-level $= m^{k_1}$

the number of inputs of the 2nd-level $= m^{k_1} \cdot m^{k_2}$

the number of inputs of the 3rd-level $= m^{k_1} \cdot m^{k_2} \cdot m^{k_3}$

and the number of inputs of the last level $= m^{k_1} \cdot m^{k_2} \cdot m^{k_3} \ldots m^{k_1}$

$$= m^{(k_1+k_2+\ldots k_1)}$$

$$= m^{k}$$

Therefore, the two networks N_1 and N_2 have the same number of inputs.

Let $\underset{\sim}{x}_1 = (X_1, X_2, \ldots X_{k_1})$
$\underset{\sim}{x}_2 = (Y_1, Y_2, \ldots Y_{k_2})$
.
.
.
$\underset{\sim}{x}_1 = (Z_1, Z_2, \ldots Z_{k_1})$

be the selector lines of the MVMUXs of N_2 when $\underset{\sim}{x}_1 = \underset{\sim}{a}_1$, $\underset{\sim}{x}_2 = \underset{\sim}{a}_2, \ldots \underset{\sim}{x}_1 = \underset{\sim}{a}_1$, the output of the N_2 be $I(\underset{\sim}{a}_1, \underset{\sim}{a}_2, \ldots \underset{\sim}{a}_1)$. Let $\underset{\sim}{x} = (\underset{\sim}{x}_1, \underset{\sim}{x}_2, \ldots \underset{\sim}{x}_1)$ and $\underset{\sim}{a} = (\underset{\sim}{a}_1, \underset{\sim}{a}_2, \ldots \underset{\sim}{a}_1)$.

By Definition 4.9, k-selector MVMUX is

$$\text{m-MUX}[I(\underset{\sim}{x}) : \underset{\sim}{x}] = \Big[\bigotimes_{\underset{\sim}{x} = \underset{\sim}{0}}^{\underset{\sim}{m}^{k}-1} I(\underset{\sim}{x}) \Big]_{\underset{\sim}{x}} = N_1 \qquad \text{Eq. (4.18)}$$

for $\underset{\sim}{x} = \underset{\sim}{a}$

$$= I(\underset{\sim}{a})$$

$$= N_1$$

By Definition 4.10

$$\text{m-MUX}[I(\underset{\sim}{x}) : \underset{\sim}{x}] = \Big[\bigotimes_{\underset{\sim}{x}_1, \underset{\sim}{x}_2, \ldots \underset{\sim}{x}_1 = \underset{\sim}{0}, \underset{\sim}{0}, \ldots \underset{\sim}{0}}^{\underset{\sim}{(m}^{k_1}-1), \underset{\sim}{(m}^{k_2}-1), \ldots \underset{\sim}{(m}^{k_1}-1)} I(\underset{\sim}{x}_1, \underset{\sim}{x}_2, \ldots \underset{\sim}{x}_1) \Big] = N_2 \qquad \text{Eq. (4.19)}$$

for $\underset{\sim}{x}_1, \underset{\sim}{x}_2, \ldots \underset{\sim}{x}_1 = \underset{\sim}{a}_1, \underset{\sim}{a}_2, \ldots \underset{\sim}{a}_1$ and since $\underset{\sim}{x}_1, \underset{\sim}{x}_2, \ldots \underset{\sim}{x}_1$ are linearly independent of each other; therefore,

$$N_2 = I\Big(\Big[\bigotimes_{\underset{\sim}{x}_1 = \underset{\sim}{0}}^{\underset{\sim}{m}^{k_1}-1} x_1 \Big]_{\underset{\sim}{a}_1}, \Big[\bigotimes_{\underset{\sim}{x}_2 = \underset{\sim}{0}}^{\underset{\sim}{m}^{k_2}-1} x_2 \Big]_{\underset{\sim}{a}_2}, \ldots \Big[\bigotimes_{\underset{\sim}{x}_1 = \underset{\sim}{0}}^{\underset{\sim}{m}^{k_1}-1} x_1 \Big]_{\underset{\sim}{a}_1} \Big) \qquad \text{Eq. (4.20)}$$

By Definition 4.7

$$= I(\underset{\sim}{a}_1, \underset{\sim}{a}_2, \ldots \underset{\sim}{a}_1)$$

$$= I(\underset{\sim}{a})$$

which shows that for the same set of selector values, N_1 and N_2 will always produce the same output. Therefore, N_1 and N_2 are equivalent, i.e. $N_1 = N_2$.

Theorem 4.8: In a canonical tree-structured MVMUX network the number of MVMUXs in the ith-level is given by:

$$\#\text{MUX in the ith-level} = \begin{cases} 1 & \text{for 1st-level} \\ \prod_{i=2}^{l} m^{k(i-1)} & \text{for the ith-level (other than the first)} \end{cases}$$

Proof: By Definition 4.3, the first level has only one MVMUX. By the mathematical induction method, the number of MVMUX in the 2nd-level depends on the number of inputs (m^{k_1}) of the 1st-level; the 3rd-level, 4th-level, 5th-level, ..., to the ith-level depend on the number input

$$(m^{k_1}m^{k_2},\ m^{k_1}m^{k_2}m^{k_3},\ m^{k_1}m^{k_2}m^{k_3}m^{k_4},\ \ldots,\ m^{k_1}m^{k_2}\ldots m^{k(i-1)})$$

of the 2nd-level, 3rd-level, 4th-level, ...the (i-1)th-level respectively. Therefore, for any ith-level ($l > 1$) the maximum number of MVMUX is

$$\prod_{i=2}^{l} m^{k(i-1)}$$

Theorem 4.9: In any canonical tree-structured MVMUX network in which N represents an MV function with n variables there will be

$$1 + \sum_{i=2}^{j} \left(\prod_{i=2}^{j} m^{n(i-1)} \right)$$

MVMUX modules in N. Note that the $\sum$ in the above expression denotes the ordinary mathematical summation.

This theorem can be proved by using Theorem 4.8. This formula includes the one given by Higuchi and Kameyama [83] (see Theorem 4.5) as a special case where a single-selector is used in each level in tree-structured network.

Theorem 4.10: Let N be a tree-structured MVMUX network and DT_i be the delay time of the MVMUX of the ith-level, then the total delay time TDT(N) of the network i.e. the time measuring from the signal propagating from the input to the output of the network depends on the number of levels (l) of the network and the delay time of the MVMUX of each level,

$$TDT(N) = \sum_{i=1}^{l} DT_i \qquad \text{Eq.(4.21)}$$

4.3 The Kameyama and Higuchi's Method

This section is, in part, based on the work of Kameyama and Higuchi [83]. In this method, an optimal MVMUX (T-gate) in tree-structured network is obtained by finding the compatible set of ULM implicants containing the possible reduction of a maximum number of MVMUXs. One ULM implicant implies a possible reduction of a certain number of MVMUXs in a tree-structured network. Each ULM implicant has coefficient value according to the reduced number of MVMUXs. The compatibility of ULM implicants was also defined so that all

the residue functions corresponding to the ULM implicants in a compatible set can be trivial functions.

A ULM implicant is defined to be the product term whose residue function is equal to a constant or an input variable. If the residue function of a ULM implicant I_j of level n-p+1 is trivial, then the number a_j of MVMUXs are reduced in the tree-structured network, where a_j is given as:

$$a_j = (m^p - 1)/(m-1) \qquad \text{Eq.(4.22)}$$

where p is the number of variable in the column of the decomposition matrix.

The subset C of ULM implicants is a compatible set, if a tree structure is constructed without conflict such that all the residue functions corresponding to the ULM implicants in C can be trivial. The synthesis problem is attributed to finding the compatible set C which gives a maximum saving of MVMUXs. Let all the ULM implicants be $\{I_i\}$, (i = 1, 2,...n) and consider the 0-1 integer variable X_i such that a ULM implicant I_i is in the set C if and only if X_i = 1. The objective function W which represents the reduced number of MVMUXs is given as:

$$W = a_1X_1 + a_2X_2 + \ldots + a_nX_n \qquad \text{Eq.(4.23)}$$

where a was given in Eq.(4.22).

In Eq.(4.23), the constraints characterizing the MVMUX network must be considered. First, the set $C = \{I_i \mid X_i=1\}$ is a

compatible set. Second, the subsumption relationships must be considered between ULM implicants. If a ULM implicant I_j subsumes I (i.e. $I_j = I_k$), then X_j and X_k cannot be simultaneously 1. The set of second constraints are denoted by V. As a result, the formulation of synthesis problem is done as seen in Eqs.(4.24) and (4.25).

$$\text{MAX} \quad W = a_1X_1 + a_2X_2 + \ldots + a_nX_n \qquad \text{Eq.(4.24)}$$

subject to

$$\text{V: } X_j + X_k < 1 \quad (\text{for all } I_j = I_k)$$

$$C = \{I_i \mid X_i=1\} \text{ is a compatible set} \qquad \text{Eq.(4.25)}$$

$$X_i = 0, 1 \ (i = 1, \ldots, n)$$

Eqs.(4.24) and (4.25) can be transformed into the equivalent formulation by replacing $X_i = 1-Y_i$.

$$\text{MIN} \quad F = a_1Y_1 + a_2Y_2 + \ldots + a_nY_n \qquad \text{Eq.(4.26)}$$

subject to

$$\text{V: } Y_j + Y_k > 1 \quad (\text{for all } I_j = I_k)$$

$$C = \{I_i \mid Y_i=0\} \text{ is a compatible set} \qquad \text{Eq.(4.27)}$$

$$Y_i = 0, 1 \ (i = 1, \ldots, n)$$

The problem is then to minimize Eqs.(4.26) and (4.27).

4.4 The Fang and Wojcik's Method

This section is, in part, based on the work of Fang and Wojcik [53]. In this method, the systematic design of multi-valued functions uses a decomposition technique that uses a small number of modules (ULM) to implement the functions. This decomposition technique incorporates the concept

of systematic routing from subfunctions to a single output. The function will be rewritten such that the subfunction becomes the function values in the logic equation. The function is reexpressed recursively until it is represented entirely by the subfunctions. A two-variable ULM (MVMUX or T-gate) is assumed to be the component used to implement the function. Since the component selected is a two-variable ULM, only two variables are used to partition the function. No attempt is yet made to try three, four or more variables in an attempt to decompose the function.

Rather than using only a single basic component as a building block to implement all three-valued two-variable functions, they analyze these entire classes of functions into six classes of functions as shown in Fig. 4.6.

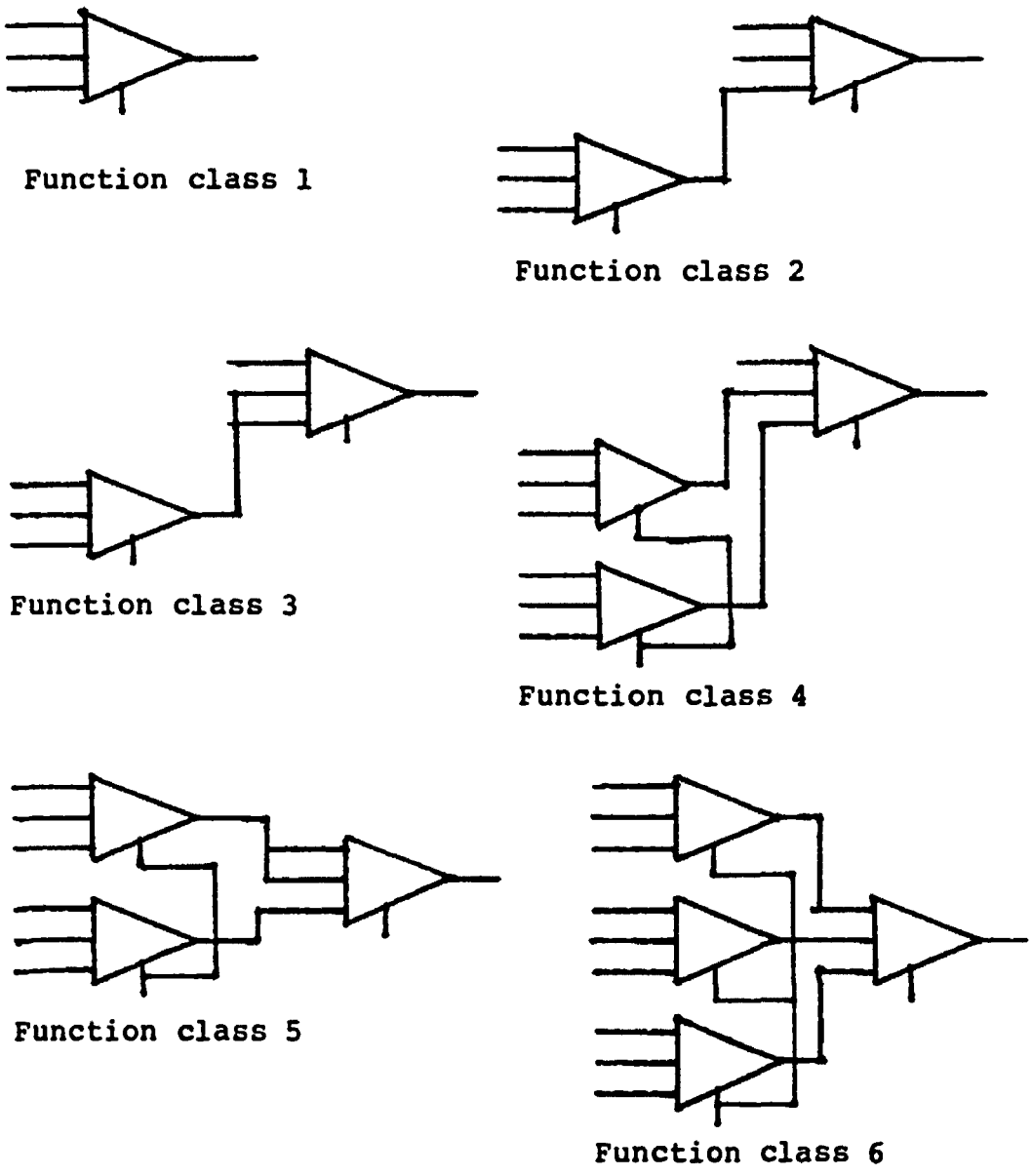


Fig. 4.6 Building Block Using MVMUX to Implement the Six Classes of Functions.

This method can be illustrated by the following example. Fig. 4.7 shows a three-valued two-variable function. If we select x_1 and x_2 as a subfunction variables, this function can be rewritten as a three-variable function $F(x_3, x_4, x_5)$,

and its function values become the subfunctions defined with respect to x_1 and x_2. There are ten partitioned matrices to be searched to identify which two of the five variables will so partition the function as to give a minimal number of subfunctions. In this example, x_1 and x_2 give five different subfunctions along with the constant 0. The new three-variable function $F(x_3,x_4,x_5)$ is shown in Fig. 4.8. The subfunctions of x_1 and x_2 are given in Fig. 4.9.

Similarly, we select x_3 and x_4 as the subfunction variables of $F(x_3,x_4,x_5)$. A one-variable function $k(x_5)$ may be written with subfunctions $g(x_3,x_4)$ and $h(x_3,x_4)$ as its function values where $g(x_3,x_4)$ corresponds to row 0 (or 2) of Fig. 4.8 and $h(x_3,x_4)$ corresponds to row 1. A search of the 3-partitioned matrices of $F(x_3,x_4,x_5)$ shows that the selecting of x_3 and x_4 yields the minimal number of subfunctions.

Fig. 4.10 gives the subfunctions of x_3 and x_4. Fig. 4.11 is the final representation of function $f(x_1,x_2,x_3,x_4,x_5)$ which is the original function composed of two subfunctions g and h. Fig. 4.12 illustrates the design of the function in Fig. 4.7 and uses the building blocks in Fig. 4.6 to realize the function.

	x_5, x_4, x_3																										
	0	1	2	0	1	2	0	1	2	0	1	2	0	1	2	0	1	2	0	1	2	0	1	2	0	1	2
	0	0	0	1	1	1	2	2	2	0	0	0	1	1	1	2	2	2	0	0	0	1	1	1	2	2	2
x_2x_1	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2	2	2
0 0	1	0	0	0	0	1	1	1	0	0	1	1	1	1	0	2	0	1	1	0	0	0	0	1	1	1	0
0 1	1	2	2	2	2	1	0	1	2	0	0	0	0	0	0	2	0	0	1	2	2	2	2	1	0	1	2
0 2	2	0	0	0	0	2	1	2	0	0	2	2	2	2	0	0	0	2	2	0	0	0	0	2	1	2	0
1 0	2	0	0	0	0	2	1	2	0	0	2	2	2	2	0	0	0	2	2	0	0	0	0	2	1	2	0
1 1	1	0	0	0	0	1	1	1	0	0	1	1	1	1	0	2	0	1	1	0	0	0	0	1	1	1	0
1 2	1	2	2	2	2	1	0	1	2	0	0	0	0	0	0	2	0	0	1	2	2	2	2	1	0	1	2
2 0	1	2	2	2	2	1	0	1	2	0	0	0	0	0	0	2	0	0	1	2	2	2	2	1	0	1	2
2 1	1	0	0	0	0	1	1	1	0	0	1	1	1	1	0	2	0	1	1	0	0	0	0	1	1	1	0
2 2	2	0	0	0	0	2	1	2	0	0	2	2	2	2	0	0	0	2	2	0	0	0	0	2	1	2	0

Fig. 4.7 $f(x_1, x_2, x_3, x_4, x_5)$.

	x_4, x_3								
x_5	00	01	02	10	11	12	20	21	22
0	a	b	b	b	b	a	c	a	b
1	0	d	d	d	d	0	e	0	d
2	a	b	b	b	b	a	c	a	b

Fig. 4.8 $F(x_3, x_4, x_5)$.

	x_2														
x_1	0	1	2	0	1	2	0	1	2	0	1	2	0	1	2
0	1	2	1	0	0	2	1	1	0	1	2	0	2	0	2
1	1	1	1	2	0	0	0	1	1	0	1	1	2	2	2
2	2	1	2	0	2	0	1	0	1	2	0	2	0	2	0
	$a(x_1, x_2)$			$b(x_1, x_2)$			$c(x_1, x_2)$			$d(x_1, x_2)$			$e(x_1, x_2)$		

Fig. 4.9 Subfunction of f.

	x_4					
x_3	0	1	2	0	1	2
0	a	b	b	0	d	d
1	b	b	a	d	d	0
2	c	a	b	e	0	d
	$g(x_3,x_4)$			$h(x_3,x_4)$		

Fig. 4.10 Subfunction of F.

x_5		
0	1	2
g	h	g
$k(x_5)$		

Fig. 4.11 Final Representation.

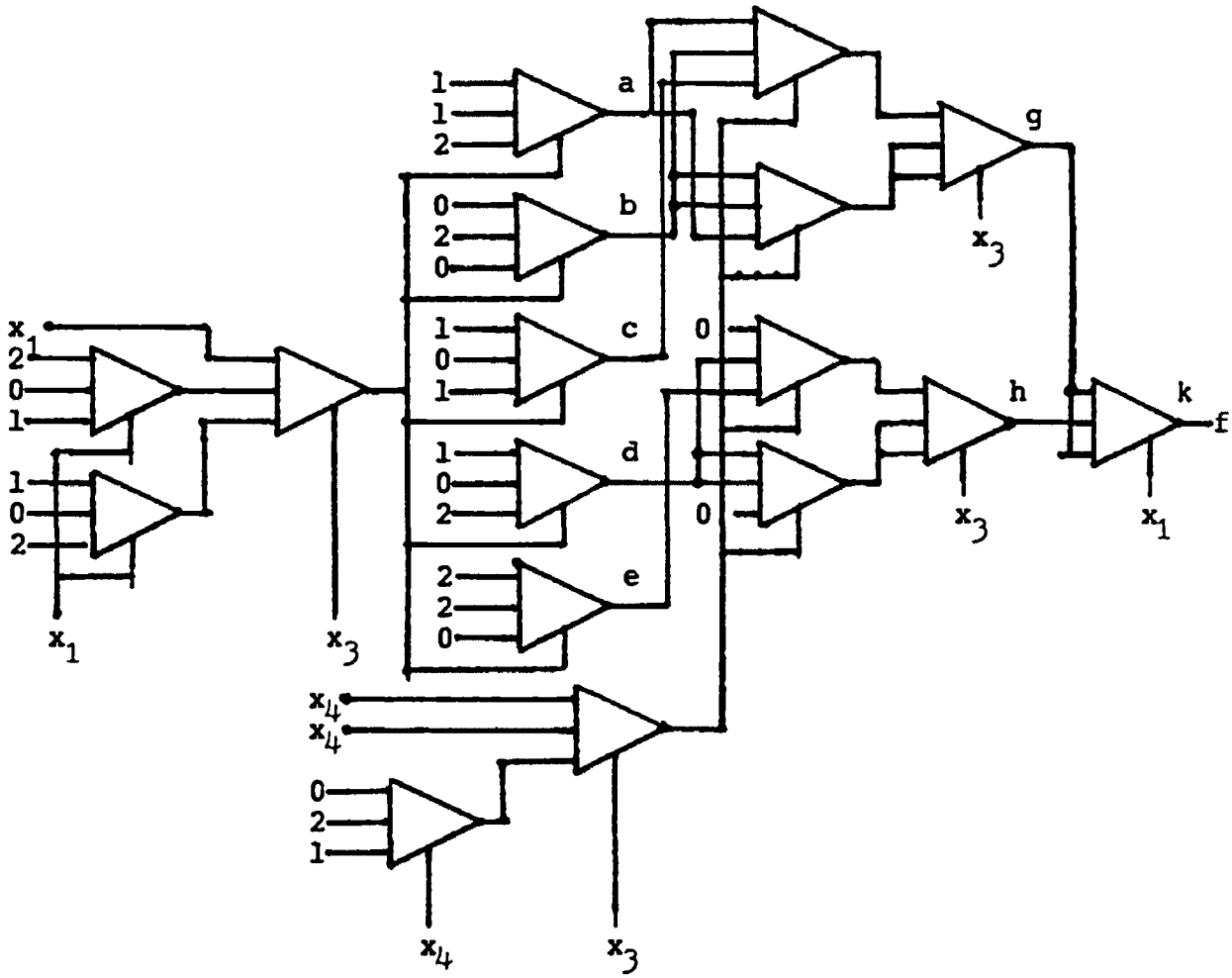


Fig. 4.12 MVMUX Structure Solution.

Even though the networks obtained by this method may require much fewer MVMUX in certain cases, this method has little or no interest to us because it will always produce non-fanout-free networks.

4.5 An Optimal Heuristic MVMUX Modular Synthesis Method

Before presenting the synthesis method of the MV function, some definitions and illustrations are introduced in their respective order.

Definition 4.14: An MV function is said to be trivial if it is a constant function or a projection function, i.e. it depends only on one of its variables or its complement.

Definition 4.15: An MV decomposition function is a re-expression of a function into a set of simpler subfunctions.

Definition 4.16: If $f(x_1,x_2,\ldots,x_n)$ can be decomposed in the form of $F(\underset{\sim}{\phi}(y_1,y_2,\ldots,y_s),z_1,z_2,\ldots,z_{(n-s)})$, this expression is called a simple disjunctive decomposition of $f(x_1,x_2,\ldots,x_n)$.

The reason this decomposition is called simple is because the composite function has but one subfunction O aside from the given variables; it is called disjunctive because the variable $\underset{\sim}{y} = y_1,y_2,\ldots,y_s$, and $\underset{\sim}{z} = z_1,z_2,\ldots,z_{(n-s)}$ are the disjoint subsets of the variables $\underset{\sim}{x} = x_1,x_2,\ldots,x_n$. Therefore,

$$\underset{\sim}{y} \cap \underset{\sim}{z} = \underset{\sim}{\phi}, \qquad \underset{\sim}{y} \cup \underset{\sim}{z} = \underset{\sim}{x}$$

Definition 4.17: The decomposition matrix, a simple disjunctive decomposition of function $f(\underset{\sim}{x})$, is a rectangular array of the m^n functional values f_i arrange in $m^{(n-s)}$ rows and m^s columns. The rows correspond to the $m^{(n-s)}$ configurations of $z_1,z_2,\ldots,z_{(n-s)}$ in order, and the columns correspond to the m^s configurations of $y_1,y_2,\ldots,y_s$ in order. Each position in the matrix is occupied by a functional value corresponding to a unique configuration of $y_1,y_2,\ldots,y_s$, and $z_1,z_2,\ldots,z_{(n-s)}$, and hence to a unique configuration of x_1,x_2

$,\ldots,x_n$. Fig. 4.13 shows a decomposition matrix of z|y

z \ y	0	1	.....	m^s-1
0	f_0	f_1	.....	$f_{(m^s-1)}$
1	f_{m^s}	$f_{(m^s+1)}$	.....	$f_{(m^{s+1}-1)}$
	.	.	.....	
	.	.	.....	
$m^{n-s}-1$	$f_{m^{n-s}}$	$f_{(m^{n-s}+1)}$	.....	$f_{(m^n-1)}$

Fig. 4.13 z|y Decomposition Matrix of f(x).

The subscripts identifying the fi have been arbitrarily taken for the case in which $z_1 = x_1, z_2 = x_2, \ldots, z_{(n-s)} = x_{(n-s)}$, $x_{(n-s+1)} = y_1$, The functional values need not occur in this order and usually will not.

For the sake of simplicity and clarity, yet without loss of generality, 3-valued functions are used to present and illustrate the following definitions. For instance, in a 3-valued system the $x_1 x_2 | x_3$ decomposition matrix for any function f(x) takes the form shown in Fig. 4.14.

$x_1 x_2$ \ x_3	0	1	2
00	f_1	f_9	f_{18}
01	f_2	f_{10}	f_{19}
02	f_3	f_{11}	f_{20}
10	f_4	f_{12}	f_{21}
11	f_5	f_{13}	f_{22}
12	f_6	f_{14}	f_{23}
20	f_7	f_{15}	f_{24}
21	f_8	f_{16}	f_{25}
22	f_9	f_{17}	f_{26}

Fig. 4.14 3-Valued $x_1 x_2 | x_3$ Decomposition Matrix.

Assignments of values 0, 1, and 2 to f_i of the decomposition matrix of Fig. 4.14 yield

x_2x_3 \ x_1	0	1	2
00	0	1	0
01	0	0	0
02	0	1	2
10	2	1	0
11	1	1	1
12	2	0	0
20	2	1	0
21	0	1	0
22	2	2	2

Fig. 4.15 An Arbitrary 3-Valued $x_2x_3|x_1$ Decomposition Matrix.

Definition 4.18: An arbitrary logic function of n variables can be realized by a full-blown tree-structured MVMUX network with a maximum of $1 + \sum_{j=1}^{n-1}(\prod_{i=2}^{j} m^{n_{i-1}})$ MVMUXs. For example, the decomposition matrix in Fig. 4.15 can be implemented by a maximum of 13 MVMUX as shown in Fig. 4.16.

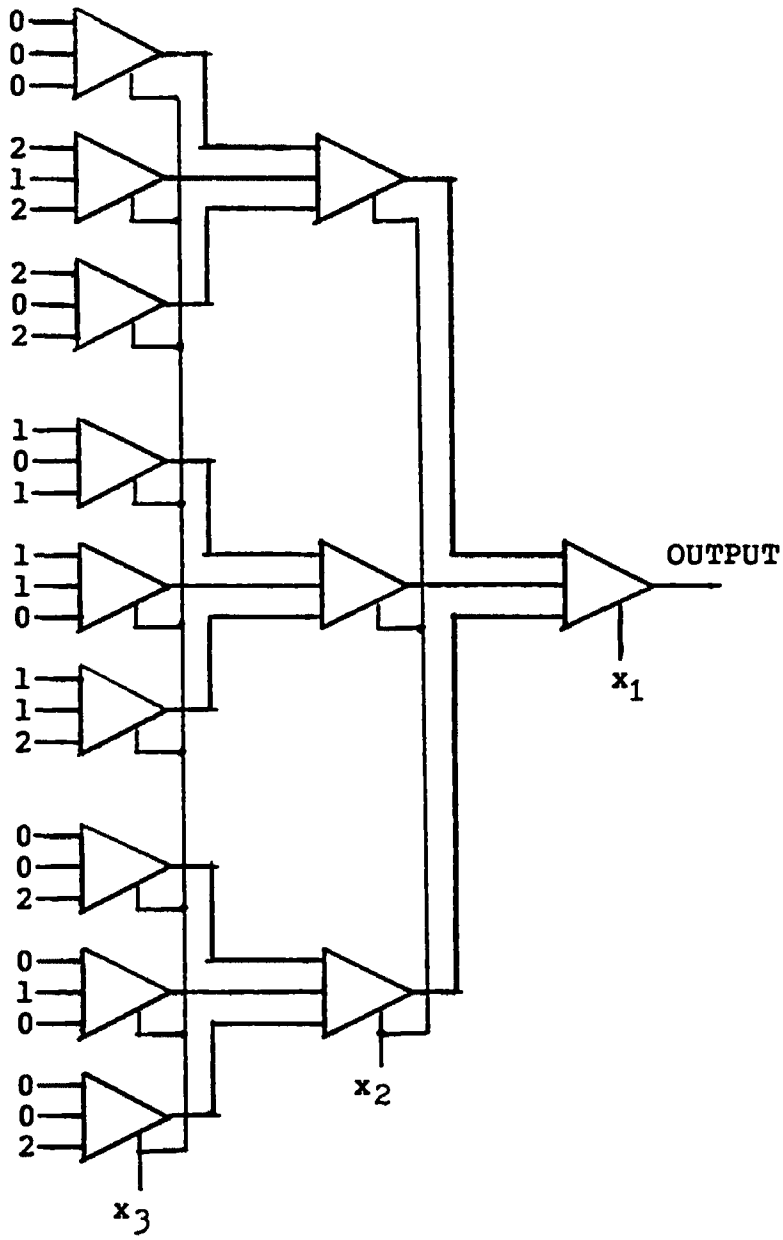


Fig. 4.16 Maximum MVMUX Implementation from Decomposition Matrix in Fig. 4.15.

Definition 4.19: A decomposition matrix is called trivial if it represents a trivial function.

Definition 4.20: A row vector, shown under the row variables in a decomposition matrix is called a vertex, e.g. in the

decomposition matrix of Fig. 4.15, 00, 01,...,and 22 are vertexes under the row variable x_2 and x_3.

Definition 4.21: The function defined by the row on the right hand side of a vertex is called the residue function of the vertex, e.g. in the decomposition matrix of Fig. 4.15, 010 is the residue function of the vertex 00.

Definition 4.22: A residue function which is a constant (000, 111, 222), or an input variable (012), or a complement of input variable (210) is called a trivial residue function of that vertex vector. For example, in Fig. 4.15 there are six trivial residue functions: 000, 012, 210, 111, 210, and 222; these trivial residue functions correspond to vertex vectors 01, 02, 10, 11, 20, and 22 respectively.

Definition 4.23: Any n-variable function can be decomposed into n possible decomposition matrices. For example, (x_1,x_2,x_3) can have $x_2x_3|x_1$, $x_1x_3|x_2$, and $x_1x_2|x_3$ decomposition matrices.

Definition 4.24: In a decomposition matrix, identical non-trivial residue functions of different vertex vectors are called common residue functions. For example, in Fig. 4.7 the residue functions corresponding to x_2x_3= 00 and 21 are 010 which is a common residue function of the decomposition matrix.

Definition 4.25: A function is called reducible if there is at least one trivial residue function in any of the possible

decomposition matrices of the function. Otherwise, it is irreducible.

Note that any irreducible decomposition function requires $1 + \sum_{i=2}^{j} (\prod_{i=2}^{j} m^{n_{i-1}})$ MVMUXs to realize it.

The following is a heuristic synthesis procedure for realizing any MV n-variable function into a minimal tree-structured single-selector MVMUX network. This approach is different from all the previous methods in that it starts from the output of the network and proceeds step-by-step towards its input.

Step 1: Partition the n-variable function into n possible decomposition matrices.

Step 2:

Case 2.1: If any one of the n decomposition matrix is trivial, one has a trivial solution, i.e. a constant, variable or its complement.

Case 2.2: If all of the n decomposition matrices do not have any trivial residue function, i.e. the function is irreducible, then select any one of them. Construct a tree-structured MVMUX network directly from the decomposition matrix.

Case 2.3: If the function is reducible, that is there is at least one decomposition matrix which has at least one trivial residue function, select the decomposition matrix with the least number of trivial residue functions and form

m branches of decomposition matrices from this node. Each branch has n-1 decomposition matrices constructed from the n-1 variables.

Step 3: For the decomposition matrices of each of the m branches,

Case 3.1: If any one of the n-1 decomposition matrix is trivial, one has a trivial solution for that branch, i.e. to put this variable or its complement at the input of that branch.

Case 3.2: If all of the n-1 decomposition matrices do not have any trivial residue function, then select any one of them. Construct a tree-structured MVMUX subnetwork directly from the decomposition matrix.

Case 3.3: Same as Case 2.3.

Step 4: Repeat Step 3 as many times as needed until there are no variables in any of the branch functions.

This procedure is best illustrated by means of an example.

Example 4.3 To realize the nontrivial function $f(\underset{\sim}{x})$ which is given in the example of Higuchi and Kameyama [83] by the heuristic method:

(1) Partition the 3-variable into 3 decomposition matrices $x_1 x_2 | x_3$, $x_1 x_3 | x_2$, and $x_2 x_3 | x_1$ as shown in Fig. 4.17.

$x_1 x_2$ \ x_3	0	1	2
0 0	0	0	1
0 1	1	1	2
0 2	2	2	0
1 0	0	2	1
1 1	1	2	2
1 2	0	0	0
2 0	1	1	1
2 1	2	1	0
2 2	0	1	2

$x_1 x_3$ \ x_2	0	1	2
0 0	0	1	2
0 1	0	1	2
0 2	1	2	0
1 0	0	1	0
1 1	2	2	0
1 2	1	2	0
2 0	1	2	0
2 1	1	1	1
2 2	1	0	2

$x_2 x_3$ \ x_1	0	1	2
0 0	0	0	1
0 1	0	2	1
0 2	1	1	1
1 0	1	1	2
1 1	1	2	1
1 2	2	2	0
2 0	2	0	0
2 1	2	0	1
2 2	0	0	2

Fig. 4.17 Possible Decomposition Matrices.

(2) The numbers of trivial residue functions of the three $x_1 x_2 | x_3$, $x_1 x_3 | x_2$, and $x_2 x_3 | x_1$ decomposition matrices are 4, 3 and 1 respectively.

(3) Choose the decomposition matrix $x_2 x_3 | x_1$ since it has the least number of trivial residue functions. Therefore, x_1 is used as the data-selector MVMUX at the output level (1st-level).

(4) Eliminate x_1 from the first-generation decomposition matrix $x_2 x_3 | x_1$ by constructing 3 branches of decomposition matrices, each of which contains the variables x_2 and x_3. In the branch $x_1 = 0$, partition the matrix into two decomposition matrices $x_2 | x_3$ and $x_3 | x_2$, do similarly branches $x_1 = 1$ and 2. They are shown in Figs.

4.18 (a), (b), and (c). Find trivial residue functions in each of these matrices which are tabulated in Table 4.3.

x_2 \ x_3	0	1	2
0	0	0	1
1	1	1	2
2	2	2	0

x_3 \ x_2	0	1	2
0	0	1	2
1	0	1	2
2	1	2	0

(a) branch $x_1 = 0$

x_2 \ x_3	0	1	2
0	0	2	1
1	1	2	2
2	0	0	0

x_3 \ x_2	0	1	2
0	0	1	0
1	2	2	0
2	1	2	0

(b) branch $x_1 = 1$

x_2 \ x_3	0	1	2
0	1	1	1
1	2	1	0
2	0	1	2

x_3 \ x_2	0	1	2
0	1	2	0
1	1	1	1
2	1	0	2

(c) branch $x_1 = 2$

Fig. 4.18 Possible Decomposition Matrices.

Table 4.3 Number of Trivial Residue Functions in Fig. 4.18.

Branch x_1	Number of trivial residue functions $x_2 \mid x_3$	$x_3 \mid x_2$
$x_1 = 0$	0	2
$x_1 = 1$	1	0
$x_1 = 2$	3	1

(5) Choose the second-generation decomposition matrices $x_2 \mid x_3$, $x_3 \mid x_2$ and $x_3 \mid x_2$ for branches $x_1 = 0$, 1 and 2 respectively. Since they have the least number of trivial residue functions in the decomposition matrices of their respective branch.

(6) Eliminate the variables x_3, x_2, and x_2 from the above three chosen decomposition matrices respectively. Now these decomposition matrices have only one variable left as seen in Figures 4.19 (a), (b), and (c).

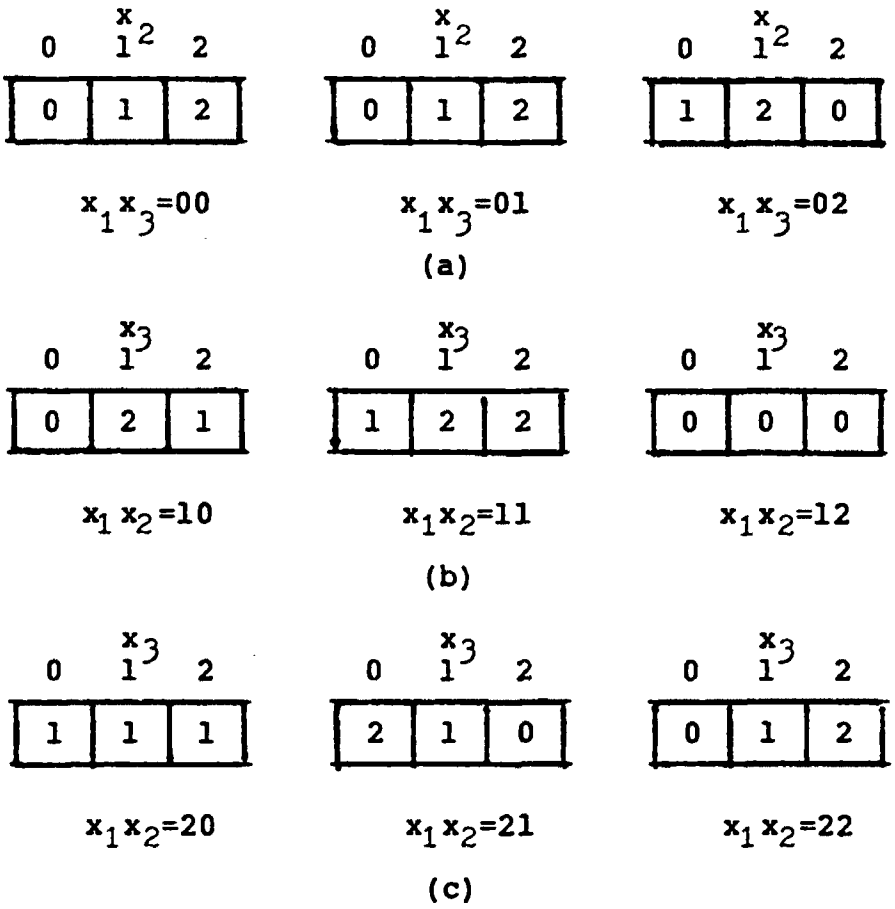


Fig. 4.19 Decomposition Matrices with
(a) $x_1x_3=00$, $x_1x_3=01$, $x_1x_3=02$
(b) $x_1x_2=10$, $x_1x_2=11$, $x_1x_2=12$
(c) $x_1x_2=20$, $x_1x_2=21$, $x_1x_2=22$.

(7) Since the third-generation decomposition matrices x_1x_3 =00 and 01, x_1x_2=12, and x_1x_2=20,21, and 22 are trivial (Case 3.1), the branches containing them are terminated. Put these variables or their complements at the inputs of those branches.

(8) The decomposition matrices left are those of x_1x_3=02, x_1x_2=10, and x_1x_2=11 which are irreducible. Hence these branches are assigned in the following values: 1, 2, and 0 to branches $x_1x_3x_2$=020, 021, and 022; 0, 2, and 1

to branches $x_1x_2x_3$=100, 101, and 102; 1, 2, and 2 to branches $x_1x_2x_3$=110, 111, and 112 respectively.

(9) Since there are no variables left in any of the branch functions, the synthesis is thus completed.

The tree-structured MVMUX network realization obtained by the heuristic method is shown in Fig. 4.20. This network is virtually the same as the one given in Higuchi and Kameyama [83] except that $\bar{x}_3$ of the third MVMUX in the second level has replaced the MVMUX which complements x_3.

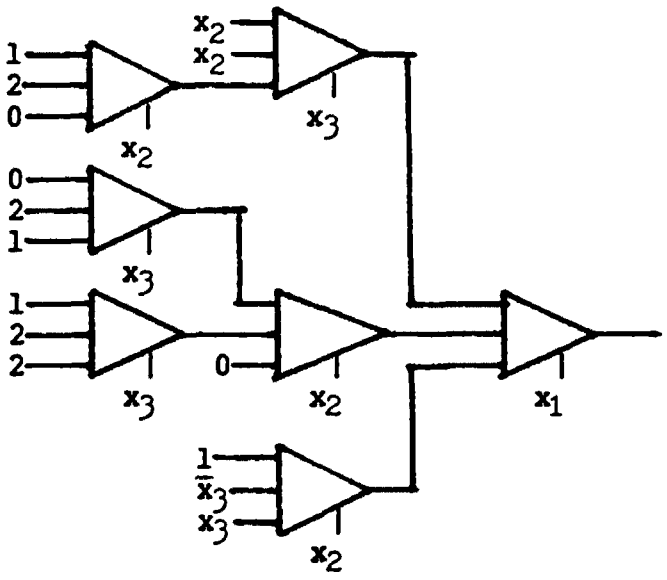


Fig. 4.20 Tree-Structured MVMUX Network Realization.

Example 4.4 Realize the decomposition matrix in Fig. 4.15 with the heuristic method procedure:

(1) Partition the 3-variable into 3 decomposition matrices, $x_2x_3|x_1$, $x_1x_3|x_2$, and $x_1x_2|x_3$ as shown in Fig. 4.21.

x_2x_3 \ x_1	0	1	2
0 0	0	1	0
0 1	0	0	0
0 2	0	1	2
1 0	2	1	0
1 1	1	1	1
1 2	2	0	0
2 0	2	1	0
2 1	0	1	0
2 2	2	2	2

x_1x_3 \ x_2	0	1	2
0 0	0	2	2
0 1	0	1	0
0 2	0	2	2
1 0	1	1	1
1 1	0	1	1
1 2	1	0	2
2 0	0	0	0
2 1	0	1	0
2 2	2	0	2

x_1x_2 \ x_3	0	1	2
0 0	0	0	0
0 1	2	1	2
0 2	2	0	2
1 0	1	0	1
1 1	1	1	0
1 2	1	1	2
2 0	0	0	2
2 1	0	1	0
2 2	0	0	2

Fig. 4.21 Possible Decomposition Matrices.

(2) The numbers of trivial residue functions of the three $x_2x_3|x_1$, $x_1x_3|x_2$, and $x_1x_2|x_3$ decomposition matrices are 6 , 2, and 1 respectively.

(3) Choose the decomposition matrix $x_1x_2|x_3$ since it has the least number of trivial residue functions. Therefore, x_3 is used as the data-selector MVMUX at the output level.

(4) Eliminate x_3 from the decomposition matrix $x_1x_2|x_3$. Now n-1 = 2. In the branch x_3 = 0, partition the matrix into decomposition matrices $x_1|x_2$ and $x_2|x_1$, do similarly to branch x_3 = 1 and 2 as shown in Figures 4.22 (a), (b), and (c) respectively. The trivial residue functions of each matrix are tabulated in Table 4.4.

x_1 \ x_2	0	1	2
0	0	2	2
1	1	1	1
2	0	0	0

x_2 \ x_1	0	1	2
0	0	1	0
1	2	1	0
2	2	1	0

(a) branch $x_3 = 0$

x_1 \ x_2	0	1	2
0	0	1	0
1	0	1	1
2	0	1	0

x_2 \ x_1	0	1	2
0	0	0	0
1	1	1	1
2	0	1	0

(b) branch $x_3 = 1$

x_1 \ x_2	0	1	2
0	0	2	2
1	1	0	2
2	2	0	2

x_2 \ x_1	0	1	2
0	0	1	2
1	2	0	0
2	2	2	2

(c) branch $x_3 = 2$

Fig. 4.22 Possible Decomposition Matrices.

Table 4.4 Number of Trivial Residue Functions in Fig. 4.22.

Branch x_3	Number of trivial residue functions $x_1 \mid x_2$	$x_2 \mid x_3$
$x_3 = 0$	2	2
$x_3 = 1$	0	2
$x_3 = 2$	0	2

(5) From Table 4.4, when x_3 is equal to 0, the possible decomposition matrices have the same trivial residue functions. In this case, we should process both variables and choose the one that has the least number of MVMUXs. If they use the same number of MVMUXs, then choose the one that has fewer total numbers of variable data inputs to the MVMUXs. The final results are shown in Figs. 4.23 and 4.24. However, Fig. 4.23 is chosen because it has fewer variable data inputs than Fig. 4.24. Even the number of MVMUXs of the two networks are the same.

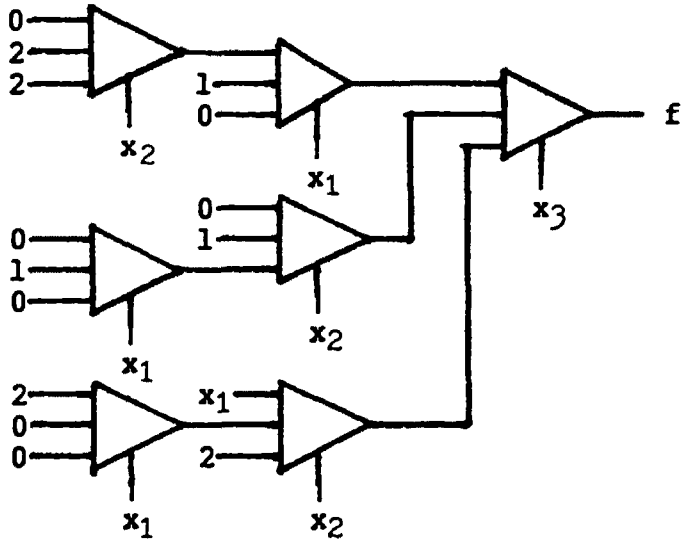


Fig. 4.23 Tree-Structured MVMUX Network Realization.

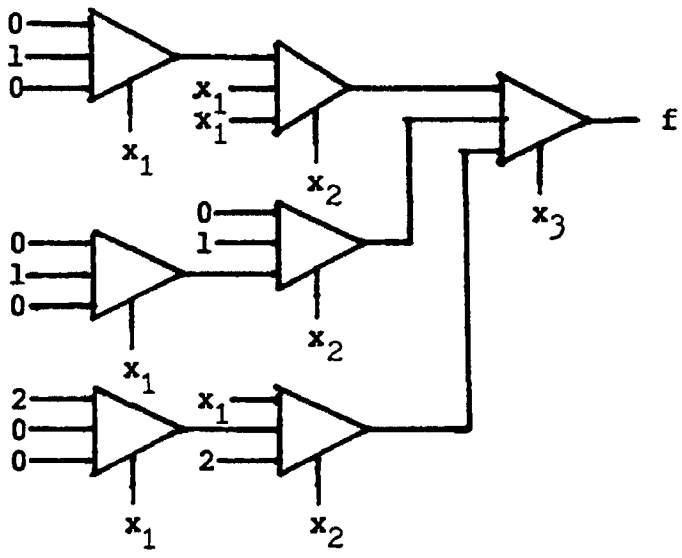


Fig. 4.24 Tree-Structured MVMUX Network Realization.

In conclusion, the heuristic modular synthesis method presented in this section guarantees a fanout-free network with a possible minimal number of MVMUXs. This approach is much simpler when compared with any of the existing methods and is, therefore, much more easily included in an automated design process in making the MV LSI/VLSI.

So far, however, we have only discussed the MVMUX synthesis methods for realizing the MV combinational logic. The synthesis of sequential logic using MVMUX will be presented in the next chapter.

CHAPTER V

MVMUX SEQUENTIAL LOGIC DESIGN USING AN MVASM CHART

The MV sequential circuit is composed of combinational circuits and memory circuits. The output signal of the sequential circuit depends on the present inputs and past history of inputs. The basic design of MV memory circuits required in a sequential circuit and an efficient and systematic design of an MV sequential logic using an algorithmic state machine (ASM) chart will be presented in this chapter.

5.1 Multi-Valued Memory Circuits Design

The MV memory circuits considered in this thesis are similar to the flip-flops in a binary system. A name "m-flop" will be called for multi-valued flip-flop. The m-flops were first introduced by Irving [74]. These m-flops used MAX, MIN, CYCLE, and COMP gates in a cross-couple fashion. There are also other components which can be used to design m-flops such as T-gate (multiplexer) and m-valued m-threshold devices which can be found in Wills' thesis [206].

In order to design the m-flops, the following properties [74] must exist:

1. The device must be defined for any m (m is a finite integer ≥ 2).

2. It must have m stable states.

3. It must have at least one output which presents a different logic value for each of the m stable states.

4. It must remain in each stable state indefinitely in the absence of external excitation.

5. It must be able to obtain any stable state A from any other stable state B in a single transition with proper excitation.

Having the same property as binary flip-flops, any type of flip-flop (JK flip-flop, D flip-flop and T flip-flop) can be realized from SR flip-flop by adding some gates. Hence, the SR m-flop will be designed first. As an extension of binary SR flip-flop, an SR m-flop requires:

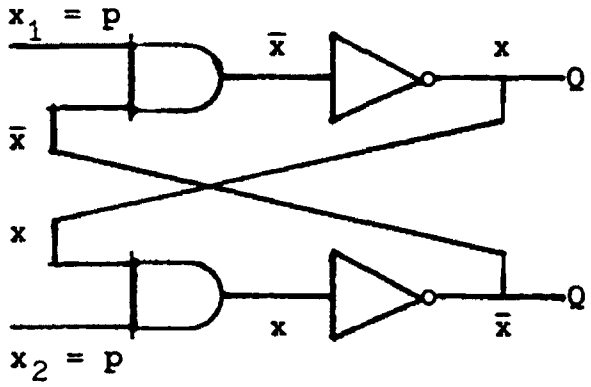
(a) If the AND operator is used, the non-deterministic input conditions occur when $x_1 + x_2 < p$.

(b) If the OR operator is used, the non-deterministic input conditions occur when $x_1 + x_2 > p$.

For example, using AND and COMP operators, the next-state equation for an SR 3-flop is

$$Q(t+1) = \bar{x}_1 + x_2 \cdot Q(t). \qquad (5.1)$$

The logic diagram, stable states, and state table of an SR 3-flop are shown in Fig. 5.1.



(a) Logic Diagram

Q (= x)	0	1	2
$\bar{Q}$ (= $\bar{x}$)	2	1	0

(b) Stable States

$x_1(t)$		0	0	0	1	1	1	2	2	2	
$x_2(t)$		0	1	2	0	1	2	0	1	2	
Q(t)	0	x	x	2	x	1	1	0	0	0	Q(t+1)
	1	x	x	2	x	1	1	0	1	1	
	2	x	x	2	x	1	2	0	1	2	

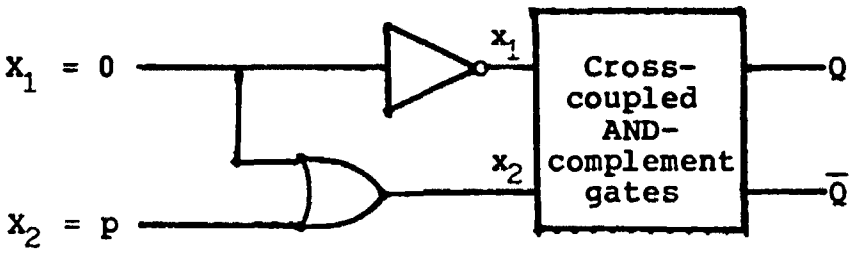
The non-deterministic input conditions occur when $x_1 + x_2 < p$

(c) State Table

Fig. 5.1 SR 3-Flop.

The next states for values of x_1 and x_2 which satisfy $x_1 + x_2 < p$ are not defined. This eliminates $p(p^2 - 1)/2$ combinations of values of x_1 and x_2 to be applied as inputs. To

avoid this, a simple gating arrangement which gives an improved SR 3-flop shown in Fig. 5.2 can be used.



(a) Logic Diagram

	X_1	0	0	0	1	1	1	2	2	2	
	X_2	0	1	2	0	1	2	0	1	2	
	x_1	2	2	2	1	1	1	0	0	0	
	x_2	0	1	2	1	1	2	2	2	2	
Q(t)	0	0	0	0	1	1	1	2	2	2	Q(t+1)
	1	0	1	1	1	1	1	2	2	2	
	2	0	1	2	1	1	2	2	2	2	

(b) state Table

Q(t)	0	0	0	1	1	1	2	2	2
Q(t+1)	0	1	2	0	1	2	0	1	2
X_1	0	1	2	0	001	2	0	011	2
X_2	d	d	d	0	12d	d	0	101	d

d = don't care

(c) SR 3-flop Input Table

Fig. 5.2 Improved SR 3-Flop.

From this improved SR 3-flop, the following D 3-flop can be constructed and is given in Fig. 5.3. The state table, excitation table, excitation map, and logic diagram of D 3-flop are given in Figs. 5.3 (a), (b), (c), and (d) respectively. It should be mentioned that I^2L SR m-flop, D

m-flop, master-slave SR m-flop, SR m-valued latch, and etc. have been designed by Dao, McCluskey and Russell [41], and Pugsley and Silio [151].

D / Q(t)	0	1	2
0	0	1	2
1	0	1	2
2	0	1	2

(a) State table

D / Q(t)	0	1	2
0	0d	1d	2d
1	00	01 02 1d	2d
2	00	01 11 01	02 12 2d

$X_1 X_2$

(b) Excitation table

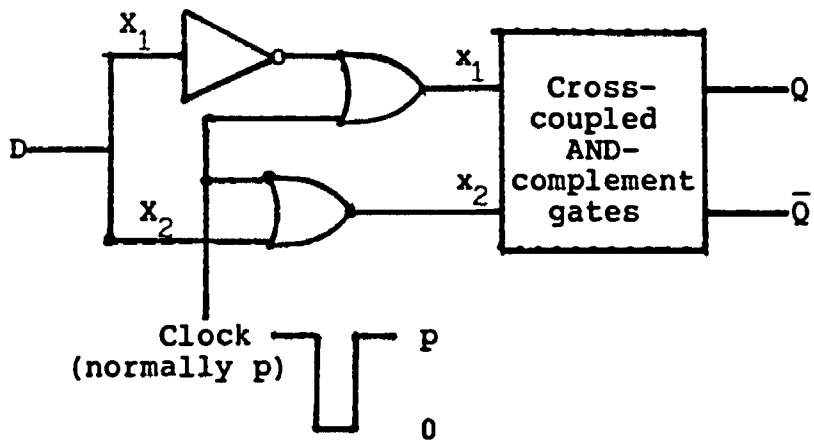
D / Q(t)	0	1	2
0	0	1	2
1	0	1	2
2	0	1	2

$X_1 = D$

D / Q(t)	0	1	2
0	d	d	d
1	0	d	d
2	0	1	d

$X_2 = D$

(c) Excitation maps



(d) Logic diagram of D 3-flop

Fig. 5.3 D 3-Flop.

The other D 3-flop and master-slave D 3-flop designed by using multiplexer or T-gate can be found in Higuchi and Kameyama's paper [66].

5.2 Multi-Valued Algorithmic State Machine

Due to the increasing complexity of the integrated circuit, the classical design technique is almost impossible. Thus, an automated design technique is necessary in the LSI/VLSI circuit. Simulation is an automated design technique widely used in a variety of engineering disciplines. When it is too difficult to verify the correctness of the design by inspection, by proof, or by test, simulation may help. Simulation allows the designers to test a design before building it by modeling in detail the components from which the design is built and by computing their interactions under various conditions. Simulation has been used in different levels of LSI/VLSI circuit design; behavioral,

register-transfer, functional, gate, circuit, etc. Unfortunately, these simulators do not really work for the whole design process because of the difficulty in preparing the data inputs in a form suitable for the simulator to simulate. For example, to write a program for a register-transfer simulator is very simple but to write the design language as the input of simulator is much more difficult to implement. The lack of commonality of design methods in the digital system design field have resulted in a delay in the availability of such programs.

The objective of this chapter is to develop a design aid which is called the MV Algorithmic State Machine (ASM) chart that may solve the problem of translating the data part of a design from a functional (behavioral) level to a structural and logical level through the specification of information.

The general design of the MV digital circuit represented in Chapter 3 was strictly in MV combinational circuits. In this chapter, an MVASM chart technique is introduced to be used as a design aid for the MV sequential logic circuit. This technique is an extension from the binary ASM chart which was introduced by Dines Biorner in 1970 [19], C.R. Clare in 1973 [30], and S.C. Lee in 1976 [96]. This new technique of designing MV sequential logic circuits using the MVASM chart allows the designers to realize a tree-structured MVMUX directly from the MVASM chart.

The steps of designing MV logic circuits using the MVASM chart are given in Fig. 5.4.

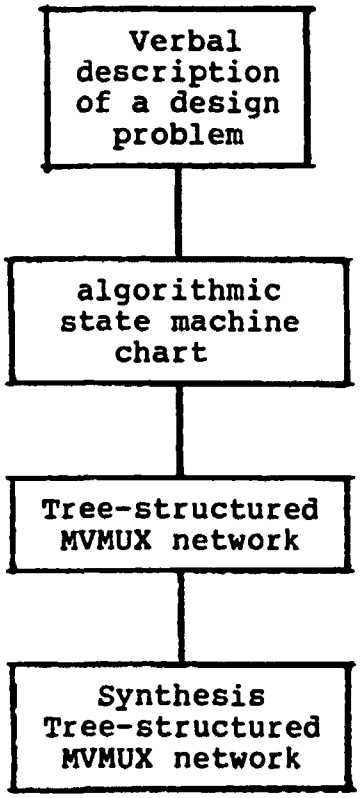
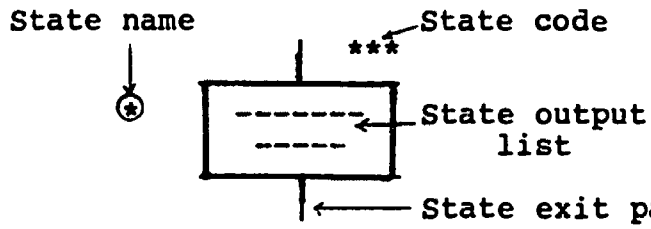
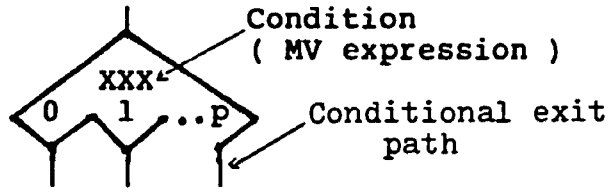
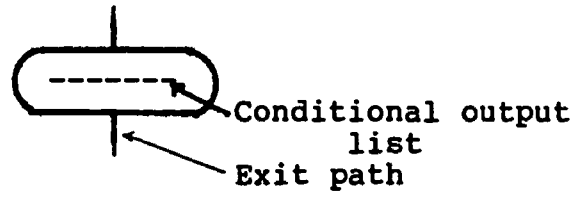


Fig. 5.4 Flow Diagram for Sequential Circuit Design.

A. MVASM Chart Description of Logic

The MVASM chart is a state diagram that describes the overall behavior of the MV sequential machine. It can be described in terms of the three basic building blocks of Table 5.1.

Table 5.1 Basic Building Blocks of an MVASM Chart.

Name	Symbol
State box	
Decision box	
Conditional output box	

Multi-Valued State Box

The MV state box is used to represent a single state of the MV sequential machine. The letter or number or name of the state is encircled on the left or right of the MV state box; the m-ary code for the state is written along the upper edge of the box. If there are state outputs of the state, they are listed in MV state box. If the output is to become active immediately, an I precedes the output name. Delayed outputs have no prefix. The MV state box has a single exit path which may lead to multi-valued decision boxes, multi-valued conditional output boxes, or other MV state boxes.

Multi-Valued Decision Box

The MV decision box describes the inputs or qualifiers to the MV sequential machine. Each MV decision box has m exit paths (m is a number of the levels of the logic), one of these is taken when its condition is active. Arrows are used to indicate a conditional exit path. It should be noted that the exit paths do not indicate any time dependence. Only the MV state box represents functions of time.

Multi-Valued Conditional Output Box

The MV conditional output box is used to describe outputs that are dependent on one or more inputs as well as the state of the machine. The conditional outputs are listed in the box with immediate and delay operations permitted. The input to the box must be a conditional exit path and there must be only a single exit path.

MVASM Block

An MVASM block is a structure consisting of an MV state box and a network of MV decision boxes and MV conditional output boxes. An MVASM block is denoted by a dashed line as illustrated in Fig. 5.5. There is only one entrance to a block and any number of exit paths depends on the network structure of the MV decision boxes within the block. Each MVASM block exit path must connect to a state, and each possible path from one state to the next is called a LINK PATH. Therefore, each exit path is a link path. Note that in

practice the dashed line is often omitted as a shortcut, but the block structures are still evident since they consist of a network of the MV conditional output boxes and the MV decision boxes between the MV state box and the next.

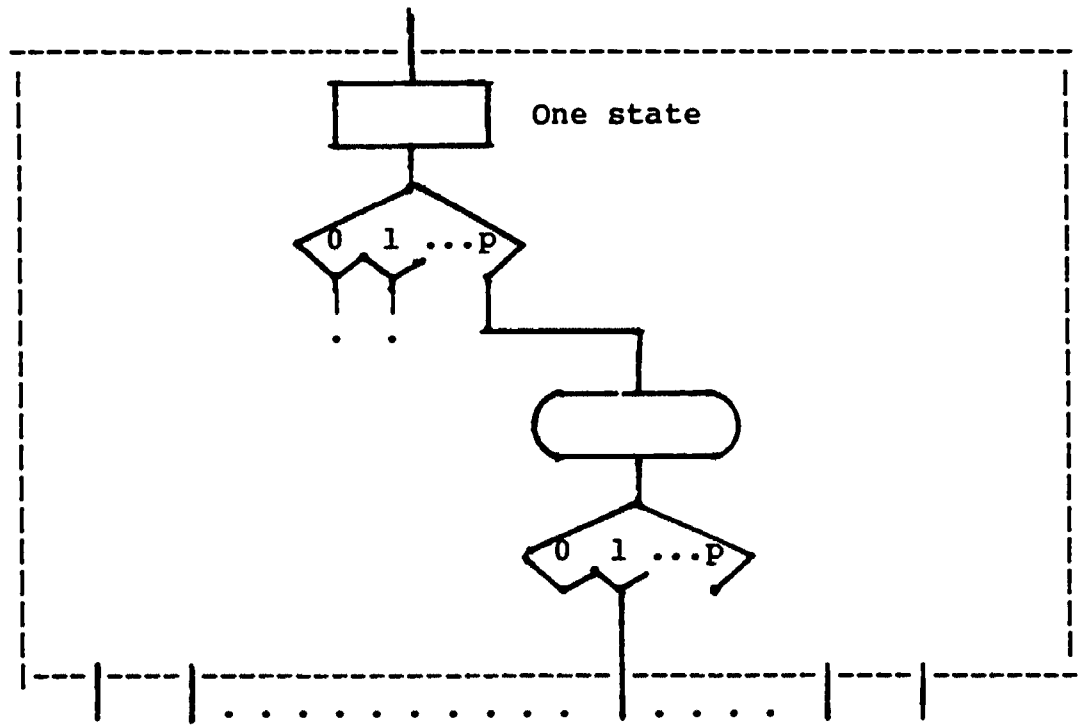


Fig. 5.5 An MVASM Block.

One MVASM block describes the state machine operation during one state at a time. Each MVASM block represents the present state, Q[t]; the state outputs, f[Q(t)]; the conditional outputs, f[Q(t),I], and the next state, Q(t+1)[Q(t),I], for a set of inputs, I, of the general state machine.

5.3 MVMUX Sequential Logic Design Using MVASM Chart

An MVMUX sequential logic design using the MVASM chart starts with describing the design using an MVASM chart of Fig. 5.6 which is to be realized by tree-structured MVMUX/D m-flop networks of Fig. 5.7. From these two diagrams we see that the input values of the MVMUXs of the network in Fig. 5.7 can be found directly from the MVASM chart of Fig. 5.6 and thus this design process can be automated by a computer. The procedure of the MV sequential machine design using the MVASM chart may be outlined as follows:

1. Define the inputs, outputs, and states of the MV logic circuit to be designed.

2. Draw state boxes in the first row of the diagram in Fig. 5.6 to represent the states of the machine. Each state is then described by a set of values of the state variables $q_1 q_2 \ldots q_n$, where $n = \lceil \log_m s \rceil$ and s represents the number of states. In other words, each state is coded by an n-tuple of m-valued numbers.

3. Indicate the state output of each state in its state box.

4. For each state box, connect a series of decision boxes to realize the verbal statements described by the problem. Output condition boxes are inserted in the paths whenever needed. An MVASM block is formed.

5. From each exit path of an MVASM block, draw the transition line to its next state. Now the MVASM chart to describe the system is completed.

6. Each m-valued state variable q_i of the MVASM chart is realized by a tree-structured MVMUX network in cascade with a D m-flop as shown in Fig. 5.7. The data-select lines of MVMUX network are the state variables and input variables.

7. The data inputs of the MVMUX network are the next state code values.

This design procedure is best illustrated by an example. Without loss of generality, consider the MVASM chart of nine states shown in Fig. 5.8 where the 3-valued logic is used. The tree-structured MVMUX/D 3-flop network realization of this chart is shown in Fig. 5.9. The input values to the MVMUXs of Fig. 5.9 are obtained from the state variables and the transition paths from one state to another indicated on the MVASM chart. For example, the values of 1 and 2 at the inputs of the first MVMUXs of the two blocks are obtained from the state variable values $q_1 = 1$ and $q_2 = 2$ of the final state 6 of the transition path from state 1 to state 6 as indicated in dark line. The rest of input values to the MVMUXs are found in a similar manner.

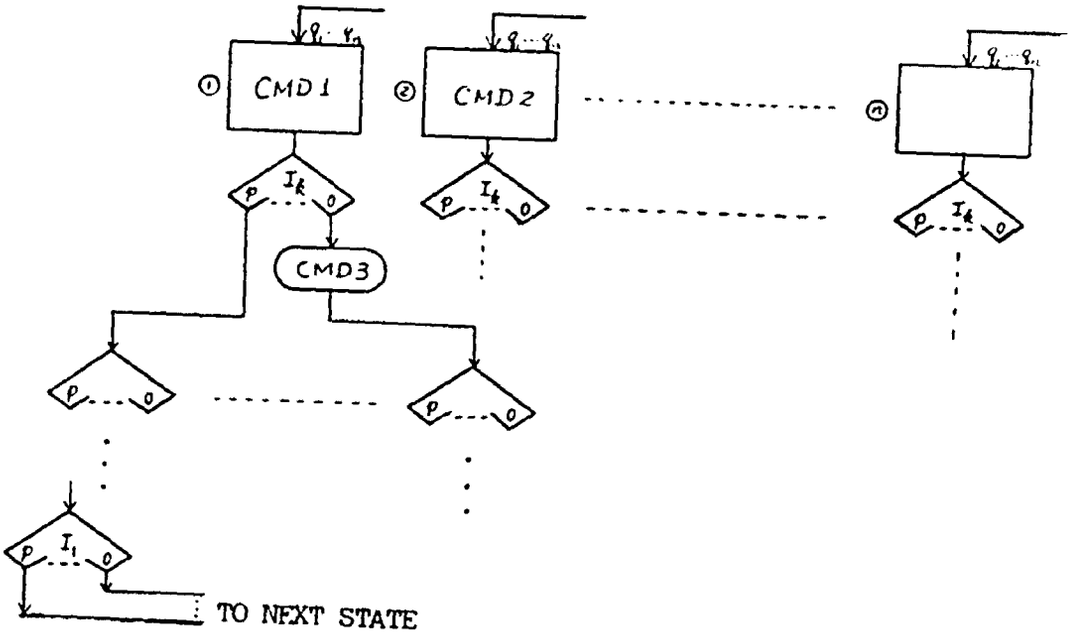


Fig. 5.6 An MVASM Chart to Describe a System.

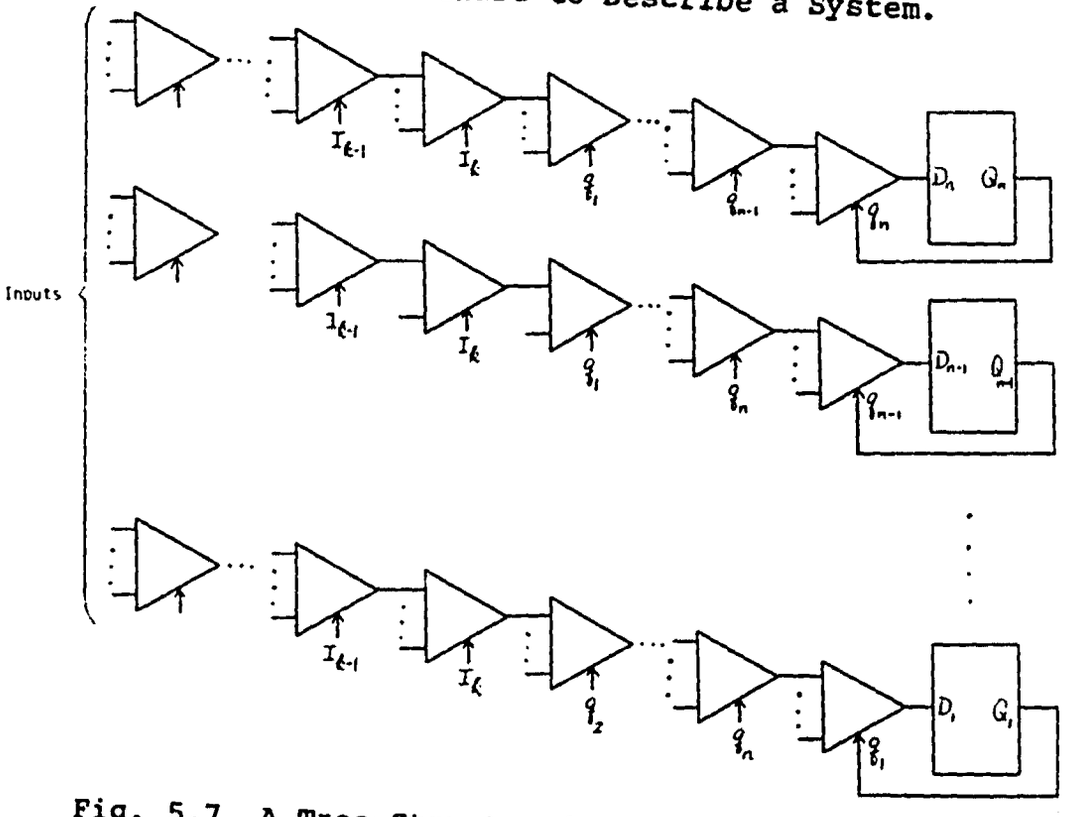


Fig. 5.7 A Tree-Structured MVMUX/D m-Flop.

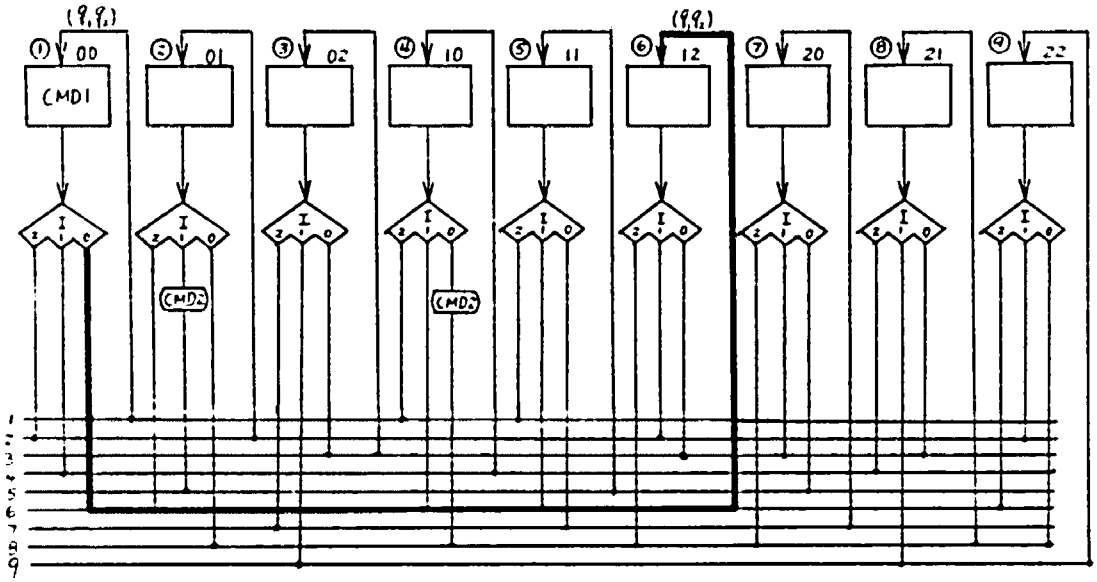


Fig. 5.8 An Example of 3-Valued ASM Chart.

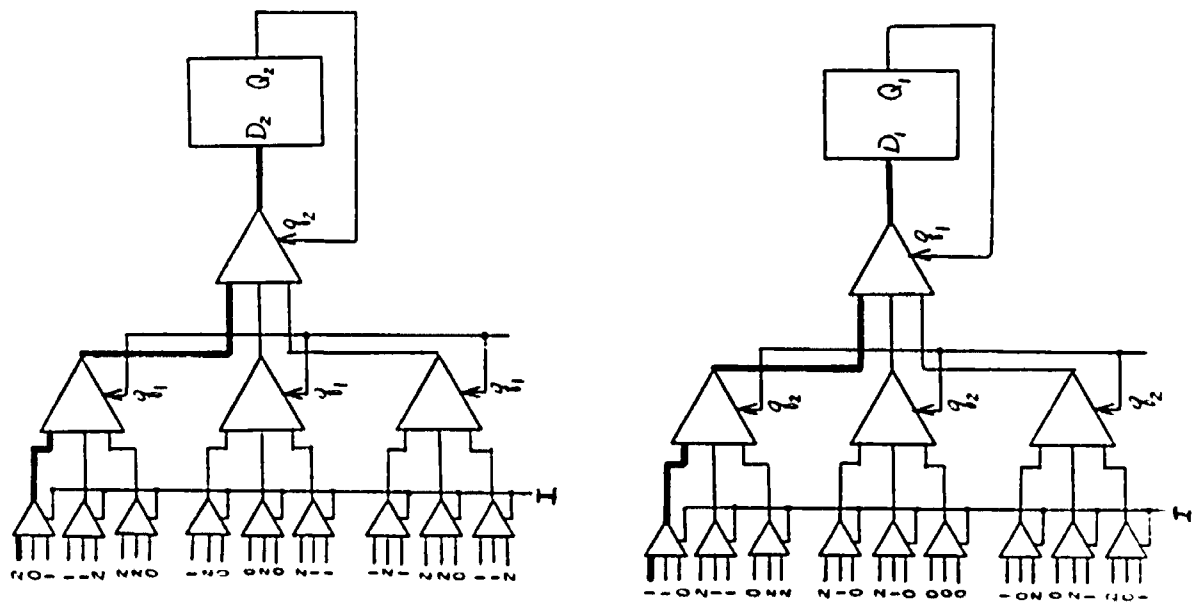


Fig. 5.9 The Tree-Structured MVMUX/D 3-flops Circuit Realization of the 3-Valued ASM Chart of Fig. 5.8.

To this point, we have presented several general and practical procedures for designing both combinational and sequential LSI/VLSI. Because of their high complexity and to ensure these circuits are fault-free after fabrication, any external test technique used will be extremely costly and almost impossible. The design of MV LSI/VLSI with built-in test circuits, hence, becomes necessary. The design of MV LSI/VLSI with built-in test circuits hence becomes necessary. Several state-of-the-art techniques for binary LSI/VLSI design are presented in the next chapter.

CHAPTER VI

STRUCTURED DESIGN FOR TESTABILITY OF BINARY LSI/VLSI

The rapid evolution of the semiconductor technology towards an ability to put hundreds of logic gates on a single chip of silicon offers a great potential for reducing power, increasing speed, and reducing cost. Unfortunately, several problems must first be solved in order to fully exploit these advantages of the LSI/VLSI. Testing of the system is one of those problems that needs to be solved before realizing this system.

Testing must be done throughout the life of a system [22] since faults may occur or be introduced into a circuit during manufacturing, assembly, storage and service. During each of these periods, the nature of the faults introduced, and consequently the type of testing which must be performed, is different. During manufacturing, typical faults which may exist are open bonds, open interconnections, bulk shorts, shorts due to scratches, shorts through dielectric, pin shorts, cracks, etc. Hence, a newly manufactured circuit may contain multiple faults, some permanent and some intermittent faults, such as the shorting of two leads due to mechanical or voltage stressing. Some of these faults can be

modeled as logical faults while others cannot. Faults may also be introduced during assembly and testing. In addition, faulty elements may not be discovered until after assembly. During storage new faults may occur in a circuit due to factors such as temperature, humidity, leakage of sealed element, and aging. These factors usually cause parametric rather than logical faults in a circuit. Finally, in service, these same factors occur as well as others caused by heat dissipation, vibration, and voltage and current stress.

The purpose of this chapter is to present the basic concepts in the testing of two-valued LSI/VLSI system which will be developed in multi-valued logic in the next chapter. This chapter begins with the logical fault models, followed by the traditional function and logical test techniques, design techniques for testability, and structured design for testability.

6.1 Logical Fault Models

In this section, the definitions of tests for logical faults and developed logical fault models for the most common faults which occur in the current technologies are presented.

Definition 6.1: A single application of values to the input terminals is called a fault-detection test.

Definition 6.2: A set of tests, which leads to a definite conclusion as to whether or not the circuit operates

correctly for all input combinations is called a fault-detection experiment.

Definition 6.3: A set of inputs which detects all possible (detectable) faults is called a complete detection test set.

Definition 6.4: A set of tests which distinguishes all pairs of possible faults is called a complete location test set.

Definition 6.5: A minimal complete test set of a circuit is a complete test set that contains a minimum number of tests.

Definition 6.6: An intermittent fault is, or appears to be present at some times but not at others.

Definition 6.7: A permanent fault is a fault always being present.

Definition 6.8: Exhaustive testing is a testing technique that uses all possible input combinations to generate test patterns (no fault model is used).

Definition 6.9: A random test or probabilistic test is a testing technique that uses some input combinations to generate test patterns (the fault model may be used).

Traditionally, faults have been modeled as a single permanent stuck-at fault model [22]. This model assumes that an input or output of a circuit element is fixed to either a logic 0 (stuck-at-0) or a logic 1 (stuck-at-1). However, such an assumption does not generally cover the

bridging faults [121] that may occur. Usually, the bridging faults have been detected by having a high level that is in the high 90% of the single stuck-at fault coverage where the single stuck-at fault coverage is defined to be the number of faults that are tested and divided by the number of faults that are assumed [205].

The other important fault models are multiple stuck-at faults and bridging faults. The multiple stuck-at faults is a fault which is composed of several single stuck-at faults that occur simultaneously.

A bridging or short circuit fault [22,121,213] is a fault which causes two or more lines in a network to connect together. In general, the stuck-at faults can be considered as a special case of the bridging faults, but some kinds of the bridging faults (for instance, a feedback bridging fault which occurs between two lines and forms a loop) cannot be modeled as a stuck-at fault. Since increasing numbers of components are being fabricated into an IC chip, the possibility of bridging faults increases.

It is known [205] that there are two major facets of the functional testing problem: test generation and test verification. Test generation is the process of enumerating stimuli for a circuit which will demonstrate its correct operation. Test verification is the process of proving that a set of tests are effective toward the end.

6.2 Types of Tests Used in Testing ICs

In general, there are three types of test which are usually performed on the digital integrated circuits: DC parametric, AC parametric, and functional tests [62].

a. The DC parametric tests

The electrical DC parametric tests verify specific parameters specified in terms of voltage or current. A DC test is performed by forcing a current and measuring a resultant voltage or by forcing a voltage and measuring the resultant current. A pure voltage measurement would assume a forced current of zero. A differential voltage measurement measures the voltage difference between two floating points. The most common DC parameters measured are continuity, leakage, power consumption, voltage high/low levels, drive capability, and noise. The important characteristics to be considered when performing a DC parametric test are the accuracy and test time per parameter per device pin.

b. The AC parametric tests

The AC parametric tests verify time-related parameters specified in terms of seconds. The basic characteristic of the AC parametric tests is the measurement of the time relationships at which a device operates, for example, the time it takes the output of a device to switch from 10% of its output level to 90%. The AC tests also measure the delay until the device output is produced after an input is

applied. Varying input timing relationships for an acceptable output is also an AC test. The most common AC parameters measured are rise and fall time, propagation delay, set-up and release times, and access time.

c. The functional or logical tests

The functional or clock rate tests are the tests required to verify that the device performs its operations or its function as the design intended. Logical zeros and ones are propagated through the device in such a manner that each device internal node is verified to operate properly. Functional testing is sometimes referred to as a clock rate, node or truth table testing. The basic characteristics of a functional test are the application of parallel and random data and the comparison of the device output to a predicted data pattern. The data is applied at rates specified for the device. The most significant testing considerations include the efficiency of pattern generation, edge-to-edge timing control, and input/output and mask switching.

The functional test may be divided into two classes: traditional test techniques and design for testability techniques.

6.3 The traditional functional and logical test techniques

The traditional test techniques are usually suitable for small scale integrated circuits (SSI) and medium scale integrated circuits (MSI). This class of tests [22] is the

tests that applied to the circuit (unit) under the test (UUT) using automatic test equipment (ATE). In one class of such systems, binary patterns are applied to the UUT and also to a reference unit realizing the same function as the UUT, and the outputs are compared. In more sophisticated computer controlled testing systems, test programs are automatically translated to the appropriate input stimuli, and the output signals are automatically interpreted and processed by the computer. A typical configuration for such a test system is shown in Fig. 6.1. Here X is the applied test stimuli, and Z' is the observed response from the UUT. The processor compares Z' with Z which is the known good response from the UUT. Based upon this information the ATE can determine whether or not the UUT is faulty (fault detection), and if so, where is the site of the fault (fault location). Often, more accurate fault location information can only be obtained by probing signals internal to the UUT. A sophisticated ATE would instruct the test operator, via the display, as to exactly which signals should be probed. The central problem in test generation is creating the input X and computing the normal response Z and the responses due to each fault of interest such that fault detection and location can be efficiently carried out.

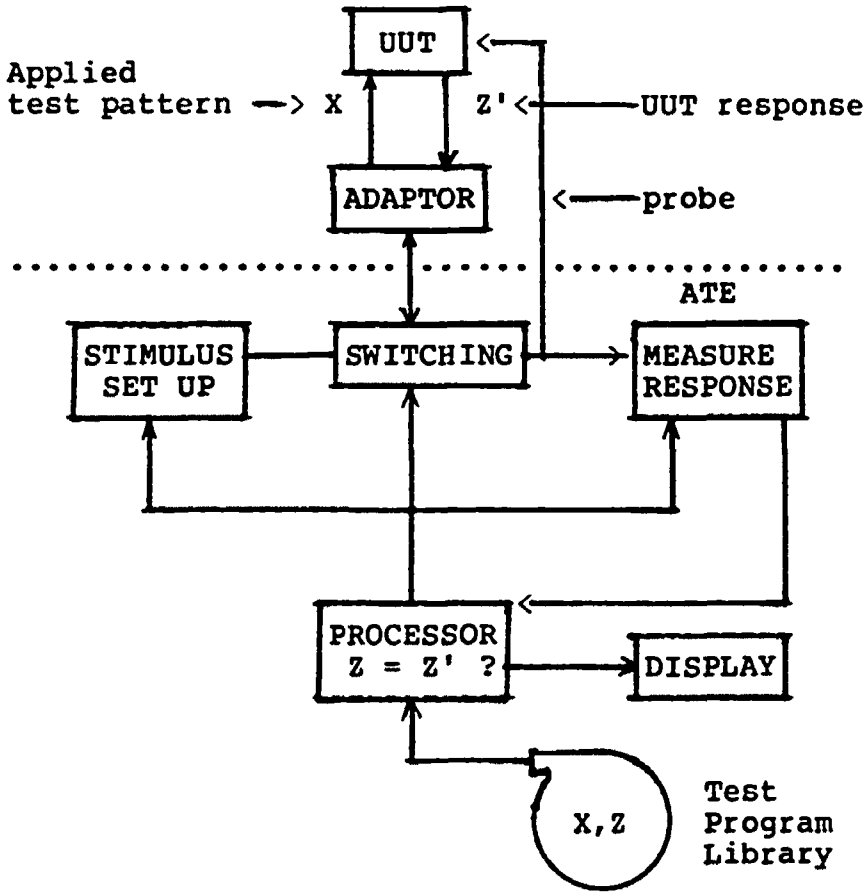


Fig. 6.1 Store Program Automatic Test Equipment System.

The generation of test patterns to be applied by the digital tester to the UUT is an important and difficult problem. Such test patterns are sometimes computer generated. Fig. 6.2 shows a block diagram for the typical automatic test pattern generation system. The inputs to the system are a description of the circuit for which test patterns are to be generated including the faults to be tested and the initial state information. Test patterns are then generated, simulated, and the circuit response is analyzed to produce dictionaries which specify circuit response to

tests under various fault conditions in a format which is easily utilized for repair. This is repeated for each test pattern until the pattern generation is concluded.

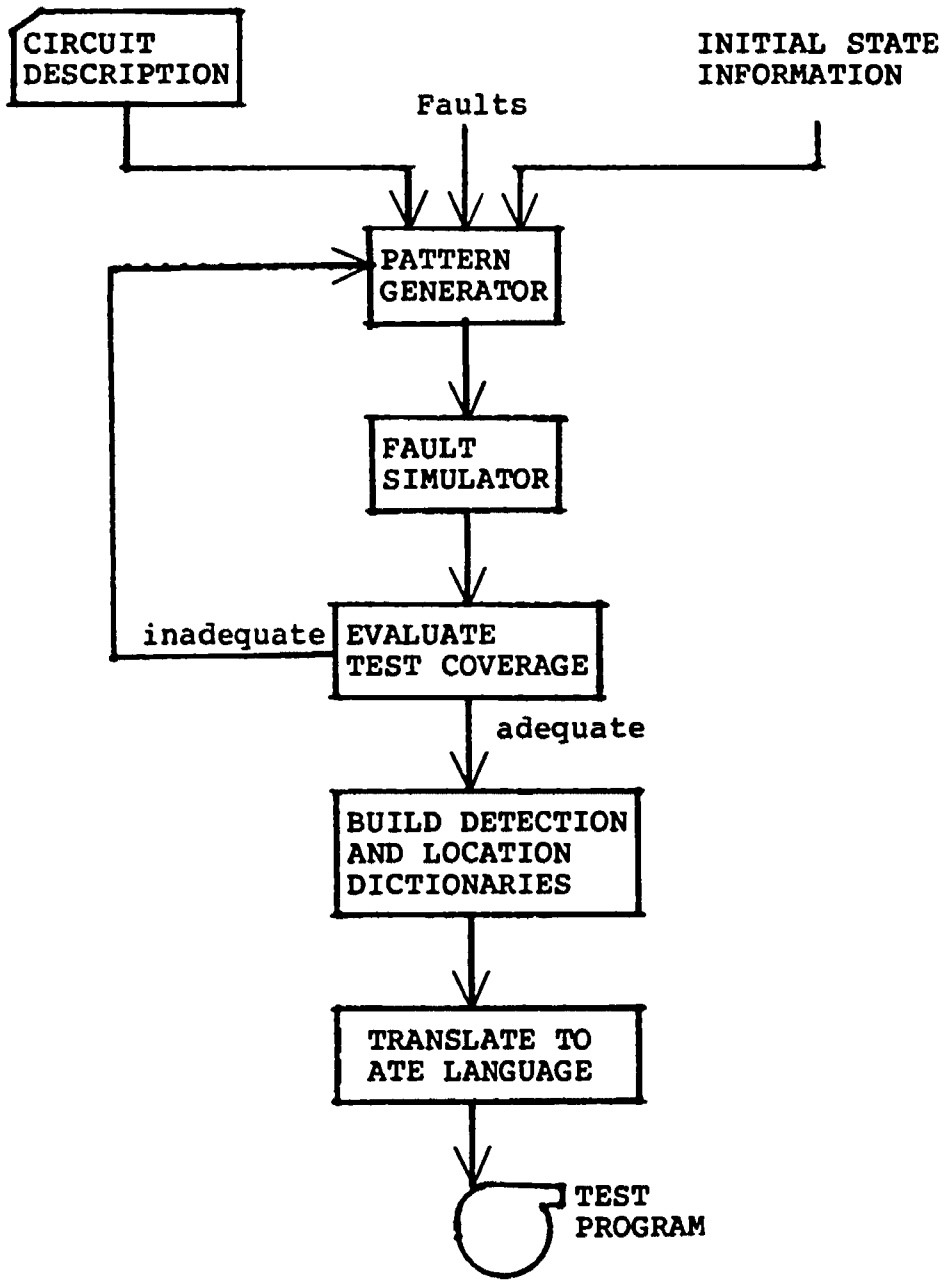


Fig. 6.2 Automatic Test Pattern Generator System.

Therefore, it is evident that in order to test the LSI/VLSI systems, this class encounters many difficult problems as follows [118,119,]:

(1) A fault model is required. In the LSI/VLSI circuits, the classical assumption that only single stuck-at-faults need to be modeled may no longer be valid. More complex models are possible, but they substantially increase the difficulty of test pattern generation.

(2) Test pattern generation is required. Automatic test pattern generation is very costly and typically does not provide sufficiently high fault coverage. Manual test pattern generation has the added disadvantage of a long delay inserting into the production cycle. For sequential circuits, automatic generation may be broken down completely, and manual generation can be very lengthy and produce poor results.

(3) An expensive tester is required. When test generation produces many patterns, the tester is tied up for a long time so that many testers may be needed.

(4) Fault coverage is too low. Because of the expense of running test generation programs, it is necessary to stop before tests for all of the stuck faults have been determined. In fact, it may not be practical to obtain tests which detect more than 80% of the single stuck faults. Low fault coverage does not really eliminate

costs, but shifts the cost from test generation to repair of systems with defective parts.

6.4 Design Techniques for Testability

The design techniques for testability are techniques that allow the IC tester to be controllable and observable so that the IC chip can be tested economically and within a reasonable time. This class of test is usually applicable for the LSI/VLSI systems. In order to enhance the testability of the IC, the following may be required.

1. Partitioning of the circuit into manageable subunits.

2. Improving the controllability of the circuit

3. Improving the observability of the circuit

The latter is the most effective means as it is evident from the fact that most test procedures are either exponential-time algorithms or at least proportional in time to the number of gates to the power of n with $n > 2$.

Almost any implementation of the above enhancements will be in conflict with the LSI/VLSI design objectives; improving the observability and controllability will require additional pins and silicon area, and the partitioning for testability may not be in accordance with normal functional partitioning. Thus, there will always be a trade-off between the testability and overhead.

The design for testability techniques to alleviate testing problems are divided into two categories [205,119]. The first category is that of the ad hoc technique for solving the testing problem. These techniques solve a problem only for a given design and are not generally applicable to all designs. This is contrasted with the second category of structured approaches. These techniques are generally applicable and usually involve a set of design rules by which designs are implemented. The objective of a structured approach is to reduce the sequential complexity of a network to aid test generation and test verification.

6.5 Structured Design for Testability

A considerable number of papers on the structured design for testability of LSI/VLSI have been published in the literature [8,46,55,89,119,165,178,184]. Among these publications the Level Sensitive Scan Design (LSSD) [46], Scan path [55], Built-In Logic Block Observation (BILBO) [89], Syndrome Testing [165], and Autonomous Testing [119] have received the most attention.

A. Level Sensitive Scan Design (LSSD)

The basic idea of the LSSD approach (as shown in Fig. 6.3) is to transform the difficult task of testing a sequential circuit into a simple task of testing a combinational circuit. This approach uses the concept that the memory elements in the IC can be threaded together into a shift regis-

ter; the memory elements values can be both controlled and observed. This technique enhances both controllability and observability, allowing us to augment testing by controlling inputs and internal states and to easily examine internal state behavior. However, there are some disadvantages in this approach:

1. The LSSD is a passive test aid in the sense that external devices are required for generating test patterns and evaluating test answers. This is not a drawback as long as the system is large enough to justify an external test or maintenance processor. But, in smaller and distributed systems, the LSSD is less useful.

2. The LSSD process is the serialization of the test; it potentially costing more time for actually running a test. It follows the idea that the increasing chip complexity will lead to longer scan paths and test times and thus cancel a certain degree gained by transforming the test problem into a combinational one.

3. In considering the cost performance impacts, there are a number of negative impacts associated with the LSSD philosophy. First of all, the shift register latches in the shift register are, logically, two or three times as complex as simple latches. Up to four additional primary inputs/outputs are required at each package level for control of the shift registers. External asynchronous input signals must not change more than

once every clock cycle. Finally, all timing within the subsystem is controlled by externally generated clock signals.

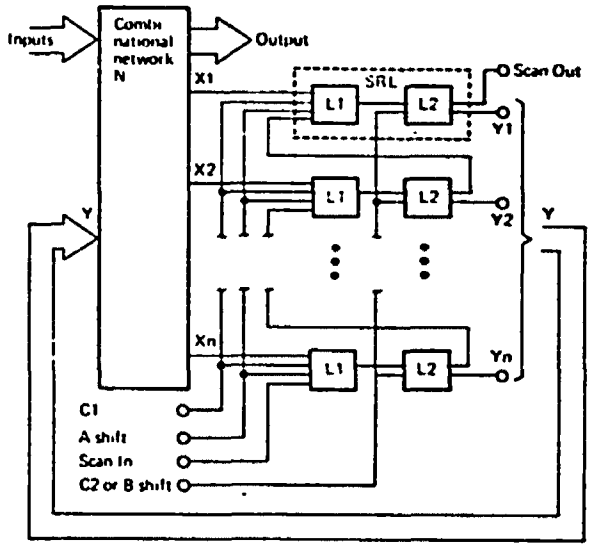


Fig. 6.3 General Structure of an LSSD Subsystem with Two System Clocks.

B. Scan Path

The basic idea of the scan path approach [55] is the same as that of LSSD approach. This approach used raceless D-flip flop as a memory element in the scan path circuit as shown in Fig. 6.4. The difference between the scan path and the LSSD approach is that the LSSD is a level-sensitive operation- the ability to operate the clocks in such a fashion that no races will exist. The LSSD used a separate clock to operate latch 1 and latch 2, while the scan path used both clocks to operate latch 1 and latch 2.

In the system operation (normal mode operation), only clock 1 is used to operate the D-flip flop by keeping clock 2 at logic 1 for the entire operation. When clock 1 is 0, the data can be loaded into latch 1. As long as clock 1 is 0 for sufficient time to latch up the data, it can then turn off. As it turns off, the result data in latch 1 will be loaded in latch 2. This assumes that as long as the output of latch 2 does not come around and feed the system data input to latch 1, latch 2 is active. The period of time that occurs is related to the delay of the inverter block for clock 1. This race condition is exposed to the use of only one system clock.

In the scanning operation (testing mode operation), the D-flip flop with scan path has its own scan input called test input. It operates by clock 2; for example, when clock 2 is 0, the data is loaded into latch 1, and the result in latch 1 is loaded into latch 2 when clock 2 is 1. The delays to avoid the race problem are the same as in the above discussion.

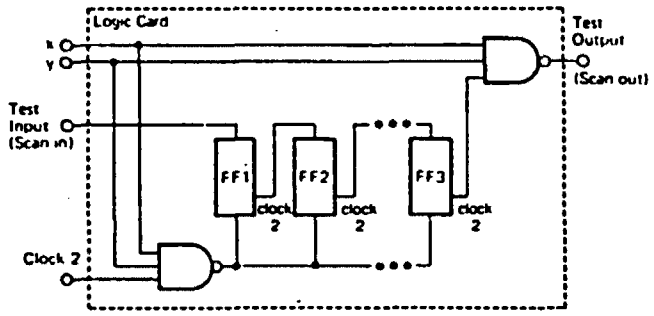


Fig. 6.4 Configuration of Scan Path on Card.

C. Built-In Logic Block Observation (BILBO)

The BILBO technique [89,139] combines the LSSD, scan path, and signature analysis [1,143] techniques together. The basic concepts of BILBO are based on the following principles:

1. All the test patterns are generated in the IC chip.

2. The test results for UUT are evaluated on the IC chip.

3. The overhead (additional pins and silicon area) is kept minimal.

4. The external testing equipment is reduced to a minimum; the only action from outside the IC chip is initializing the test and reading the go/no go information from the IC chip.

In general, the BILBO technique operates almost the

same as the LSSD and the scan path techniques, except that the BILBO technique has the circuit or memory storage to generate the test patterns within itself. Usually, the test patterns can be generated by the following methods:

1. Derive a minimum complete test patterns set and store it in the memory storage of that chip. This method uses greater amount silicon area when the number of test patterns increases.

2. Design a combinational circuit or linear feedback shift register which can generate a minimum complete test patterns set. This method is often difficult to design to match all those sets, and the circuit may be very complicated.

3. Design a counter or linear feedback shift register to generate all possible test patterns (2^n), which is called an exhaustive test. This method is only good for the small numbers of input, and no fault model is needed.

There are four basic operation modes in the BILBO technique [89]: basic system operation, linear shift register or LSSD operation, signature analysis operation, and reset mode operation.

1. Basic system operation mode (C_1C_2) = 11. When C_1C_2= 11, the BILBO of Fig. 6.5(a) will be reduced to the circuit shown in Fig. 6.5(b). Under this operation, the Z_i

values are loaded into D_i , and the outputs are available on Q_i for system operation. This would be a normal register function.

2. LSSD operation mode (C_1C_2= 00). When C_1C_2= 00, the BILBO register takes on the form of a linear shift register as shown in Fig. 6.5(c). Data scan-in is input to the left through some NOT gates, and it basically lines up the registers into a single scan path until the data scan-out is reached.

3. Signature analysis operation mode (C_1C_2= 10). When C_1C_2 = 10 in this mode as shown in Fig. 6.5(d), the BILBO register takes on the attributes of a linear feedback shift register of a maximum length with multiple linear inputs. If there are inputs to the BILBO registers, Z_i can be controlled with fixed values in this mode operation; the BILBO will output a sequence of patterns which are very close to random patterns.

4. Reset mode (C_1C_2= 01). When C_1C_2= 01 in this mode, it would force a reset on the register.

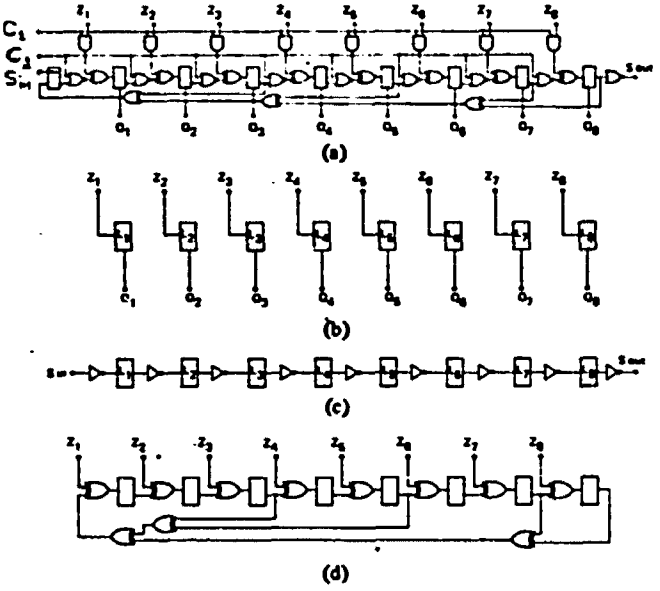


Fig. 6.5 BILBO and Its Different Modes.
(a) General Form of BILBO Register
(b) C_1C_2 = 11 System Orientation Mode
(c) C_1C_2 = 00 Linear Shift Register Mode
(d) C_1C_2 = 10 Signature Analysis Register with m Multiple Inputs (Z_1, Z_2,...,Z_8).

D. Syndrome Testing

The syndrome testing technique [165,166] is used to test the permanent stuck-at faults in the combinational system. This technique is based on the number of minterms realized by switching function and requires all the test patterns (2^n) be applied to the input of the circuit (each input combination is applied exactly once) and count the number of ones appearing at its output. Therefore, the only difference between the syndrome and the ones-count is the location of the binary point. Thus, there is no essential difference between the syndrome and the ones-count, and a binary counter can serve the purpose of measuring the

syndrome. If the syndrome stored in the counter by the time the test has been completed is equal to the fault-free syndrome, the circuit is declared fault-free; otherwise the circuit is faulty. It should be noted that in order to make the syndrome-test procedure of acceptable length, large circuits with many inputs must be partitioned to subcircuits so that each subcircuit will have no more than 20-25 inputs. Each subcircuit then is designed to be syndrome-testable.

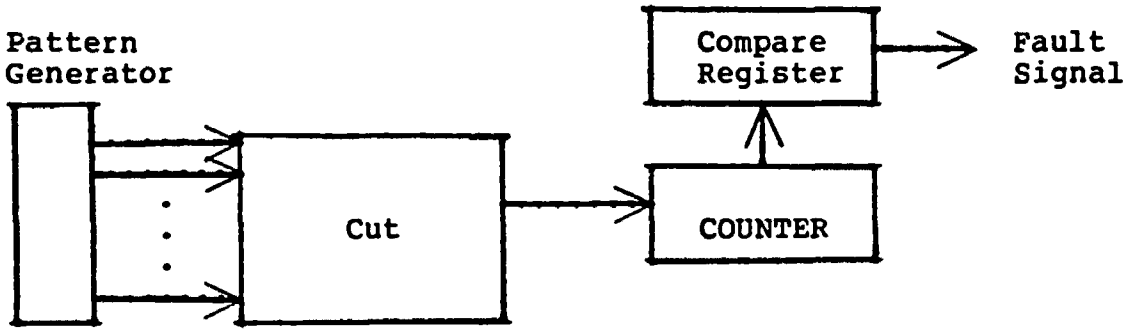


Fig. 6.6 Syndrome Test Structure.

In order to design testable combinational circuits, some extra I/O pins have to be inserted so that the final circuit will be syndrome-testable. The advantage of the syndrome testing technique is a very low storage requirement for implementation; therefore, the expensive stage of test generation can be avoided. Fig. 6.6 shows the syndrome testing system structure procedure where the syndrome-testable circuit does not include the pattern generator, counter, or compare register.

E. Autonomous Testing

The autonomous testing technique [119] is similar to the syndrome testing technique in that they both require the all possible patterns be applied to the system inputs. However, with the autonomous testing, the outputs of the system must be checked for each pattern against the value for the good machine. Consequently, irrespective of the fault model autonomous testing will detect the faults by assuming that the faulty machine does not turn into the sequential machine from the combinational machine. In order to help the system apply its own patterns and accumulate the results of the tests rather than observing every pattern for 2^n input patterns, a structure similar to the BILBO register is used. This register has some unique attributes and is shown in Figs. 6.7-6.10. If a combinational system has 100 inputs, the system must be modified so that the subsystem can be verified and, thus, the whole system will be tested.

In order to exhaustively test each subsystem, all the subsystem inputs must be controllable at the input of the system, and all subsystem outputs must be observable at the system outputs. This can be achieved in two ways: hardware partitioning and sensitized partitioning. The hardware partitioning technique is performed by inserting multiplexers and connecting the embedded inputs and outputs of each subsystem to those primary inputs and outputs that are not used by the subsystem under the test mode operation. The sensi-

tized partitioning technique can be done by applying the appropriate input pattern to some of the input lines. The effect achieved is similar to that of the hardware partitioning technique. The paths from the primary inputs to the subsystem inputs and the paths from the subsystem output to the primary output can be sensitized. Using these paths, each subsystem can be tested exhaustively.

The autonomous testing technique has improved the testability of the system as follows:

1. It is not required to have the fault model for testing since all possible test patterns are used.

2. Memory storage for the test patterns is not required since all the possible test patterns can be generated by counter circuit or feedback shift register circuit.

3. It is not required to have an expensive external testing equipment.

4. There is no serial scan in and scan out of the test pattern to the system; therefore, the testing time is reduced.

However, there are some disadvantages to the autonomous testing technique in which partitioning a system into subsystems is needed to add some components in order to be controllable for the testing and system operation mode. This adding component also reduces the speed operation of the

system.

M =

N	S	Mode
1	X	Normal Operation
0	0	Input Generator
0	1	Signature Analyzer

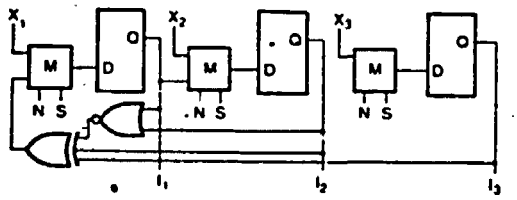
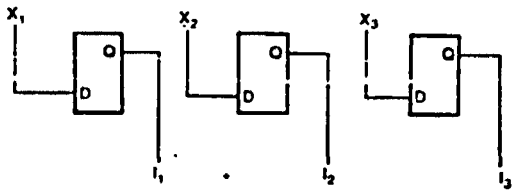
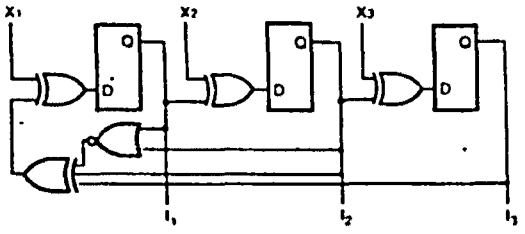


Fig. 6.7 Reconfiguration of 3-Bit LFSR Module.



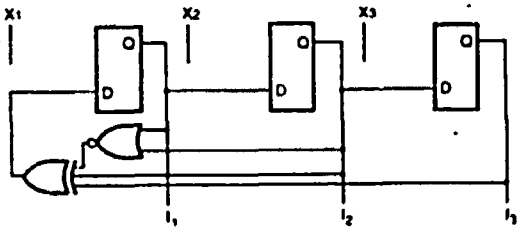
N = 1: Normal operation

Fig. 6.8 Reconfiguration of 3-Bit LFSR Module.



N = 0, S = 1: Signature Analyzer

Fig. 6.9 Reconfiguration of 3-Bit LFSR Module.



N = 0, S = 0: Input Generator

Fig. 6.10 Reconfiguration of 3-Bit LFSR Module.

In conclusion, several design-for-testability techniques for the two-valued LSI/VLSI and their advantages and disadvantages have been presented. Some of these techniques may be implemented in a new design for testability with the built-in testing capability of the MV LSI/VLSI tree-structured MVMUX network. This new design technique is intended to avoid the use of any expensive external testing devices and yet is still able to thoroughly test the chip within a reasonable test time. Moreover, such a test may be

conducted at any time during the lifetime of the chip. A detailed discussion of this new design technique is presented in the next chapter.

CHAPTER VII

HIERARCHICAL DESIGN OF MV LSI/VLSI WITH BUILT-IN PARALLEL TESTING CAPABILITY

In this chapter, the concept of a hierarchical design process which handles the complexity problem of the MV LSI/VLSI will be introduced. The tree-structured modular MVMUX network discussed in Chapter 5 will be used here to design the testability of MV LSI/VLSI systems with parallel built-in test capability. With the help of the MVASM chart introduced in Chapter 4 as a design tool, it will be shown that the design process is very efficient and easy to implement.

7.1 The Need for MV LSI/VLSI Design for Testability

It is known that MV LSI/VLSI ICs offer even more advantages over the discrete component circuits than the two-valued ICs which were discussed in chapter 2. However, the trade-off is that in the design of MV LSI/VLSI ICs, the problem of testing becomes an even more complicated one. This is because the chip density due to the use of MV logic is increased, and the components in MV LSI/VLSI chip are also increased, and hence the complexity of the chip. As

mentioned in Chapters 2 and 6, testing is one of the most important problems in two-valued LSI/VLSI circuits and also is in MV LSI/VLSI. The problem of testing in LSI/VLSI circuits is a problem of complexity with the well-known tendency towards exponentially increasing expenditure in both test time and test equipment (conventional test). In MV LSI/VLSI, the test time increases exponentially (m^n) with m-valued logic and n variables. Therefore, conventional testing for MV LSI/VLSI circuits is almost impossible. One way to solve this MV LSI/VLSI chip test problem is to integrate the testability of the circuit as a part of the design. As discussed in chapter 6, this approach allows us to test the MV LSI/VLSI chip, not only to eliminate expensive external test equipment, but also to shorten the test time.

The design for testability in MV LSI/VLSI uses the same fundamental approach as in the two-valued LSI/VLSI which is controllable and observable. In order to avoid expensive external testing devices and be able to test the chip at any time, the built-in test capability technique could be used. Because of the complexity of MV LSI/VLSI increasing exponentially with respect to m valued logic and n variables, the hierarchical approach could be used to reduce the complexity problem of the system. Because the networks are tree-structured and have identical subunits at each level, one can reduce the complexity of the network by applying the parallel testing technique which will be presented in this

chapter. Instead of using fault models which are very complicated, especially for MV LSI/VLSI circuits, exhaustive tests will be applied; this ensures that the network is fault-free once it passes the tests. It is worth noting that because of the absolute reliability feature of this approach, the designer should consider the trade-off between the amount and cost of the built-in hardware and the testing time savings.

As discussed in Chapter 1, a hierarchical design process of MV LSI/VLSI consists of three levels (see Fig. 1.4).

(1) Behavioral level

(2) Structural and logical level

(3) Physical level

Here only the structural and logical level design will be discussed. In this method, this level consists of five steps as shown in Fig. 7.1.

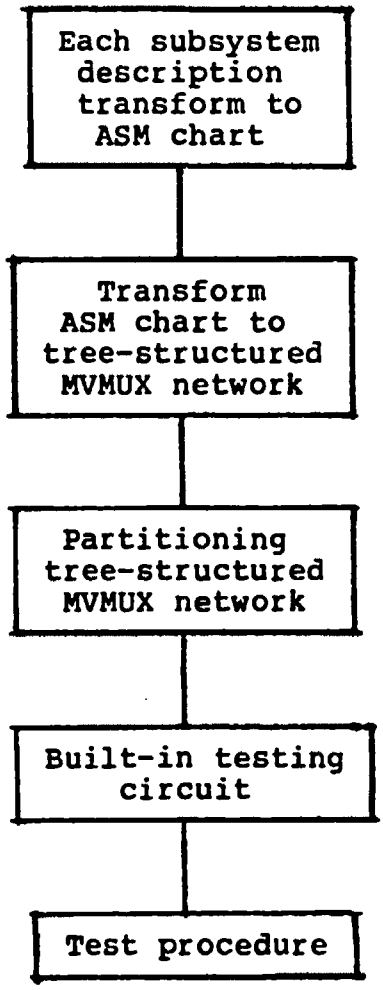


Fig. 7.1 Structural and Logical Level of an MV LSI/VLSI Design Process.

The first two steps have been discussed in Chapter 4. The last three steps will be discussed in the next three sections.

7.2 Partitioning for Testability Circuit

It is evident that the larger and more the complicated the circuit is, the longer it will take to test it. It is desirable to partition a large circuit into smaller subcircuits and to test them simultaneously. In doing so, not

only may the subcircuit testing process in general be greatly simplified, but also the total testing time may be shortened. However, the size of the subcircuits of the partition decided upon should be based on the trade-off between the desirable or required testing time for the chip and the total amount of built-in test circuit hardware needed for conducting the parallel test.

In Section 4.3, a general design procedure presented can transform any sequential logic into a tree-structured MVMUX network composed of nearly identical sections, such as the one shown in Fig. 7.2 (see the network of Fig. 4.9). Such a section of the network will be referred to as a Basic Modular Network (BMN). Since all the BMN of a tree-structured MVMUX network are of the same structure, one can apply the built-in parallel testing technique to test this type of network. Note that in applying such a test, further partitioning of the BMNs may be needed as shown in the example of Fig. 7.2. It should also be noted that with only slight modification, the test procedure described below may be applied to any BMNs with missing tree branches. Such BMNs may result from a minimization process during the synthesis of the network. The two BMNs of the network of Figs. 5.20 and 5.23 are examples of such cases.

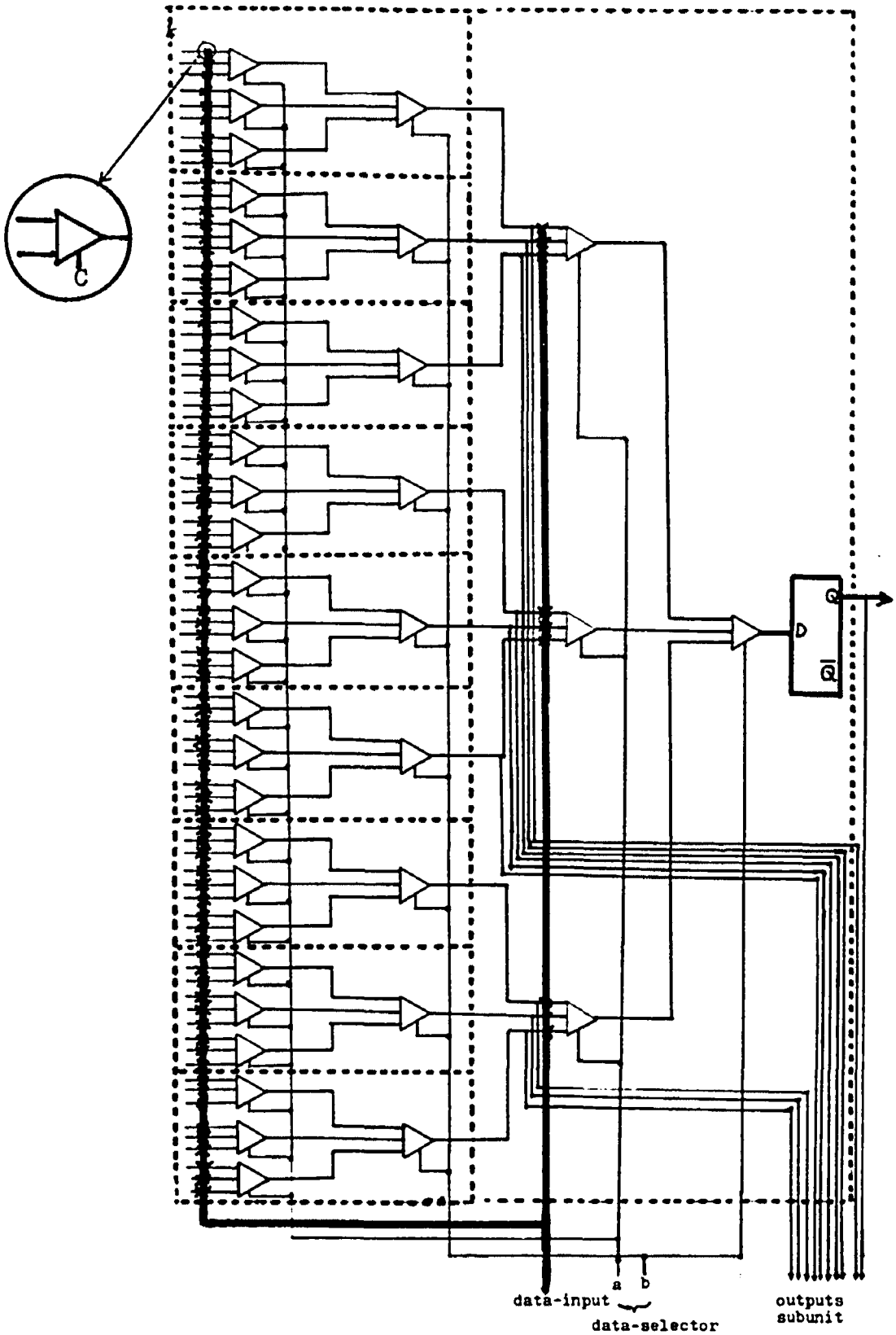


Fig. 7.2 Tree-Structured MVMUX Network with D 3-Flop.

7.3 Built-in Test Circuits for BMN

There are three functional circuits that will be built inside an MV LSI/VLSI chip. One is the test data input generator which will generate all the possible signal levels (i.e. m = 3, the signal levels are 0,1, and 2) to the inputs. The second is the data-selector generator which will generate all possible m^n patterns (i.e. for m = 3 and n = 4, a total of 3^4 = 81 patterns will be generated). The last one is the test verification circuit which will identify a fault in each partition subunit. In designing these circuits, the following requirements must be considered.

1. The built-in circuit must be much simpler than the main circuit itself.

2. In order to ensure that the built-in test circuit is initially fault-free, one should be able to check it by using an external probe before packaging. After packaging, the circuit should have a high probability of fault-free maintainance throughout the lifetime of the chip.

3. The number of IC pins used for the built-in tests should be less than 10% of the total IC pins.

4. The circuit should be able to test the system at any time.

After a tree-structured MVMUX network is partitioned,

the designer will insert the three types of circuits in it for the purpose of parallel testing. They are:

(1) Test data-input generator: It is an m-valued generator which furnishes input data to the input leads of each partition block. This unit is controlled by a synchronous clock and by control line C. The unit is inactive when C = 0 (see Fig. 7.3). The symbol —[$\vec{y}^1$]— in Fig. 7.3 denotes the cycling gate whose output is determined by (y + 1) mod 3.

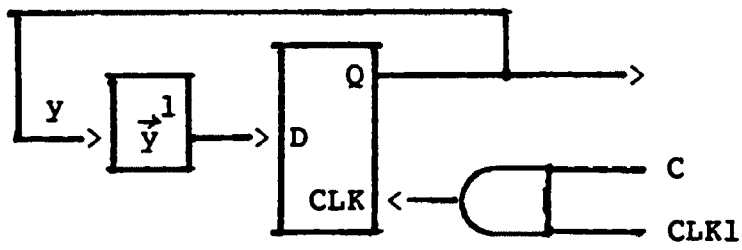


Fig. 7.3 Test Data-Input Generator Circuit.

(2) Test data-selector generator: It is a modular m^h counter when h is the number of selectors of the partition block. It is also controlled by the synchronous clock and by control line C. It becomes inactive when C = 0. It is used to select one of the m^h inputs of the partition block. (see Fig. 7.4)

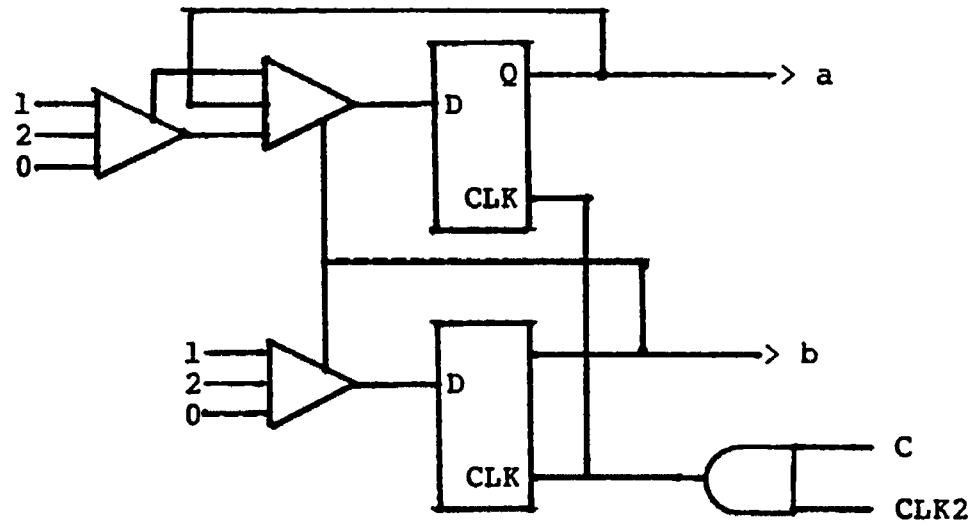


Fig. 7.4 Test Data-Selector Generator Circuit.

(3) Test verification circuit: It is a multi-bus comparator circuit. (see Fig. 7.5)

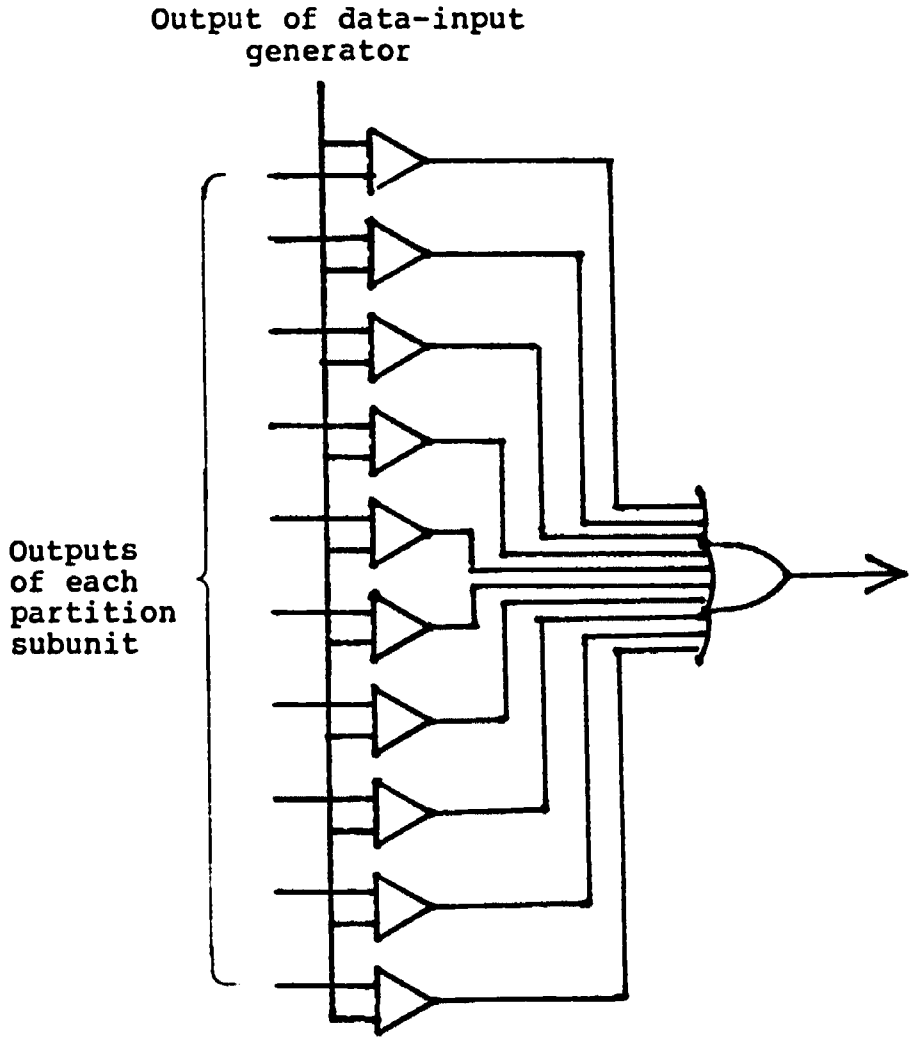


Fig. 7.5 Test Verification Circuit.

It should be noted that during the normal operation, these testing circuits can be completely isolated from the main circuit by the switches , x, with the control line, C = 0, as shown in Fig. 7.2.

7.4 Test Procedure

The built-in test capability for the MV LSI/VLSI circuit consists of two basic operation modes, normal mode and

test mode. These two modes are controlled by the signal that apply to a control IC pin which is called the test control terminal.

1. Normal mode: In this mode, the built-in test units are isolated from the main circuit.

2. Test mode: In this mode, the MV LSI/VLSI circuit is connected to the built-in test units. The values of data inputs and data selectors are supplied by the data-input generator and data-selector generator respectively. Each BMN is tested exhaustively by a parallel process. The output response of each subcircuit is verified by the verification unit.

After the test circuits are in place as shown in Fig. 7.6, the following test procedure will be in order.

Step 1: Set C = 1 to isolate the inputs and outputs of all the partitioned blocks and also to reset all the test circuits.

Step 2: Test data-input generator which generates a zero and sends it to all partitioned block inputs. The test data-selector will generate 0 and send it to the data-selector lines of all the partitioned blocks. The outputs of the partitioned blocks are compared with the inputs at the verification circuit. If any of the two sets of data are different, a fault is detected; this process will be repeated for test data-selector values 1, 2 ...m.

Step 3: Repeat Step 2 for data-input values 1, 2, ..., p. The process, however, will be terminated whenever a fault is detected by the verification circuit.

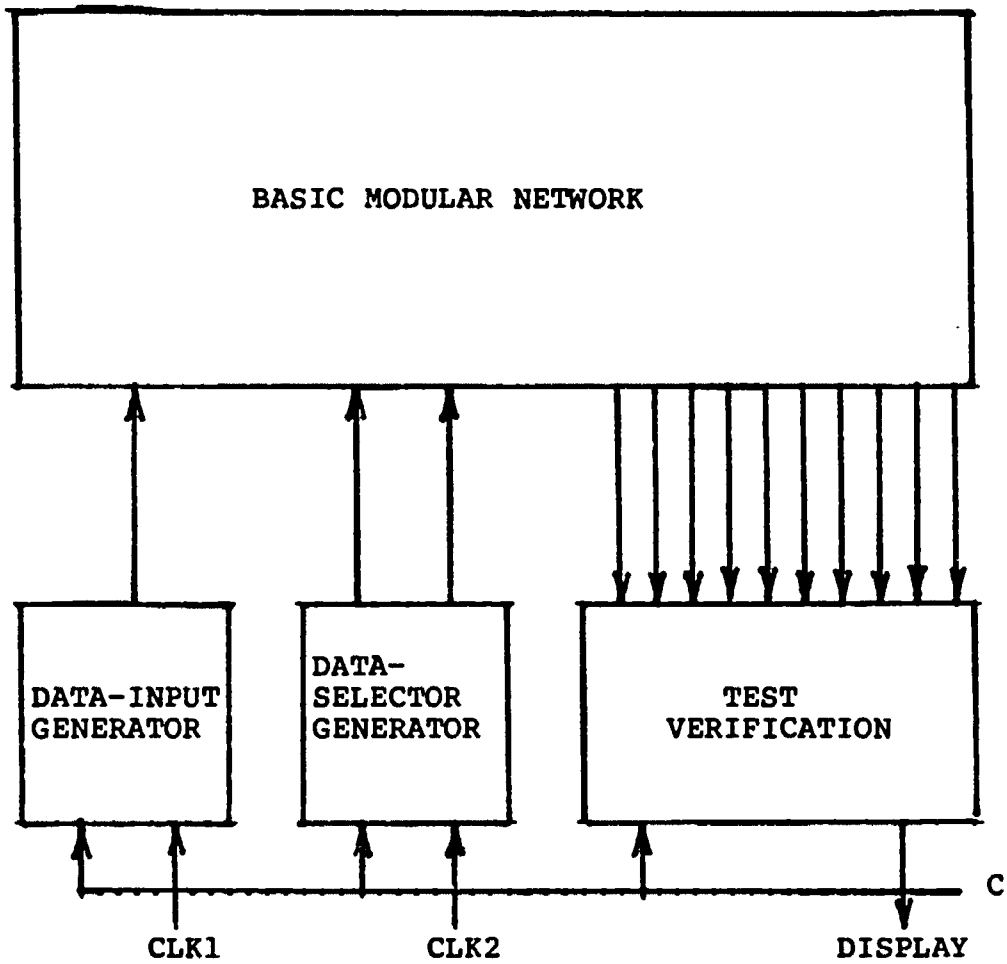


Fig. 7.6 BMN with Testing Circuit.

In order to ensure that all operations of the testing are synchronized, two clock generators are employed: a test data-selector generator clock and a test data-input generator clock. Their timing diagrams and the outputs of their respective units are shown in Fig. 7.7.

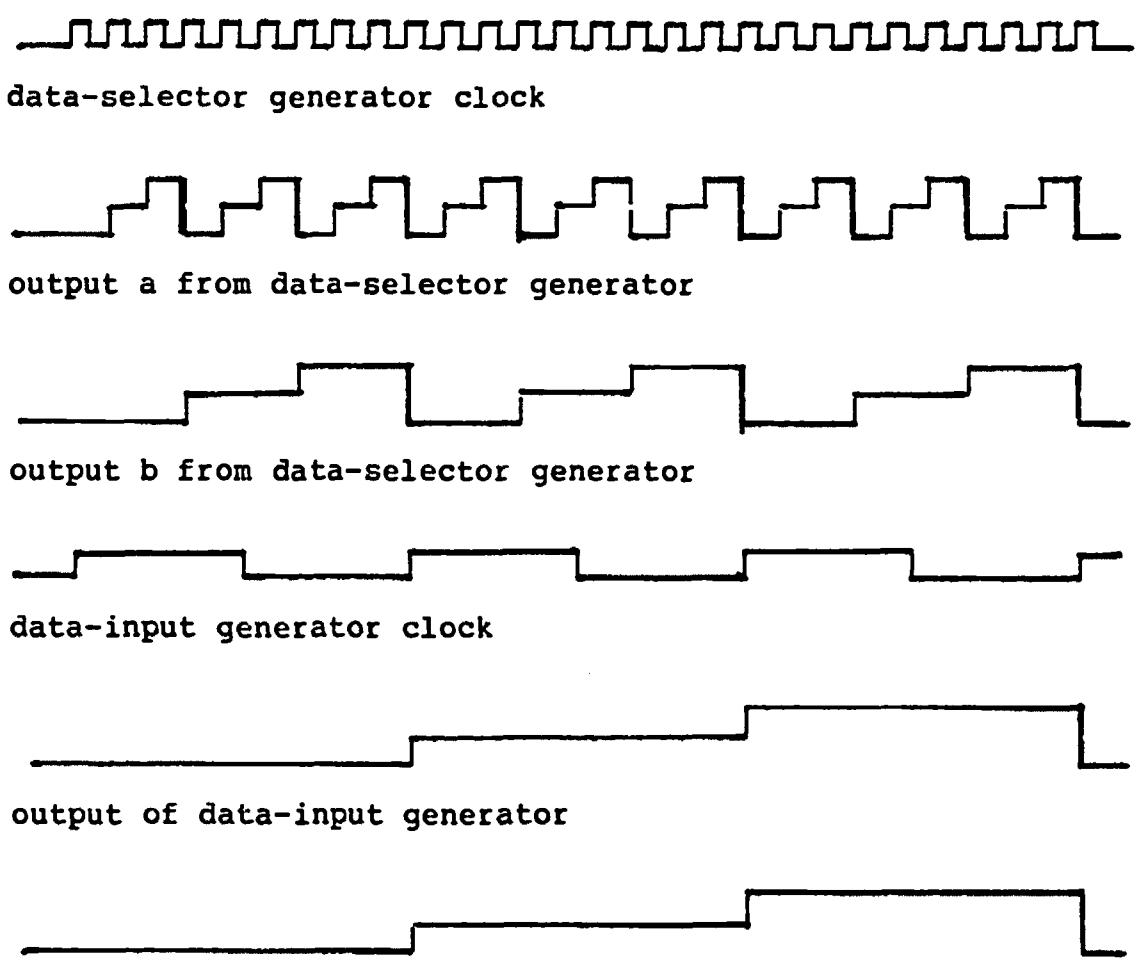


Fig. 7.7 Timing Diagrams of Testing Circuit.

As a final note, the definition of BMN was rather loose; it could be defined, for example, by a network with twice (or larger) the size of the network in Fig. 7.2. However, this will not effect the basic test procedure outlined above.

In summary, an easy-to-apply hierarchical MV LSI/VLSI design technique with built-in test circuits to provide

highly efficient parallel testing capability has been presented. The synthesis method included in this technique is completely general and is applicable to any sequential logic design; including the binary as a special case. Moreover, unlike most of the known techniques for the binary LSI/VLSI circuits, this technique requires only two extra pins (the control pin and the verification pin), very simple built-in test circuits, and virtually no external test equipment; yet it provides a thorough and exhaustive test for the entire circuit-both its combinational and D m-flop parts. The test may be conducted at any time during the lifetime of the chip. If a chip passes this test, the chip is guaranteed to be fault-free.

CHAPTER VIII

CONCLUSION

Four major considerations in the design of binary LSI/VLSI are functional complexity, chip density, pin limitation, and chip testability. It has been shown that MV logic offers many advantages over binary logic especially in the areas of reducing functional variables and therefore pins; this increases the information per unit area and thereby reducing the hardware components. Consequently, a MV logic system provides a solution which not only reduces the complexity of the system but also increases the chip density.

Due to the complexity and size of LSI/VLSI, any attempt to test these circuits using external testing means is almost impossible, especially when an exhaustive test is desired. It is therefore suggested that a built-in circuit test technique be used. A set of requirements for the built-in test circuits has been given. A new hierarchical design procedure using the ASM chart as a design tool has been presented. This design procedure can be used for synthesizing any combinational and sequential logic with a tree-structured MVMUX network plus D m-flops. It has been

shown that the use of the tree-structured MVMUX network in the design of LSI/VLSI offers the following advantages:

1. It makes the synthesis procedure simple and systematic, especially for designing complex function circuits.

2. One can insert simple built-in test circuits in the network for conducting exhaustive tests at any time.

3. The network can be partitioned into nearly identical subnetworks, each of which is connected to a built-in test circuit and can be tested simultaneously.

Finally, it should be noted that the technique for designing MV LSI/VLSI presented in this dissertation includes the technique for designing binary LSI/VLSI as a special case. It should also be noted that the process of designing MV LSI/VLSI can be adapted to automation.

BIBLIOGRAPHY

1. "A Design's Guide to Signature Analysis," Hewlett-Packard Application Note 222, Hewlett Packard, 5301 Stevens Creek Blvd., Santa Clara, CA 95050.

2. Again, J., "Setun, Nauchno-Tekh.," Obshchestva SSSR, 2(3), 25, March, 1960.

3. Akins, D., "A Suggested Approach to Computer Arithmetic for Designers of Multi-valued Logic Processors," Proc. Eight Int., Symp. on Multiple-valued Logic, Rosemont, ILL, 1978, pp. 33-46.

4. Allen, C.M. and Givone, D.D. "A Minimization Technique for Multiple-valued Logic Systems," IEEE Trans. on Computers, Vol. C-17, February, 1968, pp. 182-184.

5. Allen, C.M. and Givone, D.D.,"The Allen-Givone Implementation Oriented Algebra," Computer Science and Multiple-valued Logic, Chapter 9, Rine, D.C. (Editor), North Holland, 1977.

6. Allen, C.M., "A Switching Algebra and Associated Minimization Techniques Applicable to the Design of Multiple-Valued Logic Systems," Ph.D. Thesis, University of New York at Buffalo, June, 1968.

7. Anderson, D.J., and Dietmayer, D.L., "A Magnetic Ternary Device," IEEE Trans. on Computers, December, 1963, pp. 911-914.

8. Ando, H., "Testing VLSI with Random Access Scan," in Dig. Papers COMPCON 80, IEEE Pub. 80CH1491-0C, Feb. 1980, pp. 50-52.

9. Armstrong, J.R., Singh, A.D., and Gray, F.G., "Combinational and Sequential Multivalued Logic Design Using Iterative Tree Structures," Proc. the ninth Int. Symp. on Multiple-Valued Logic, Bath, England, 1979, pp. 182-189.

10. Ashenhurst, R.L., "The Decomposition of Switching Functions," Proc. of an Int. Symp. on Theory and Switching, April 2-5, 1957, Ann. Computation Lab., Harvard University, Vol. 29, 1959, pp.74-116.

11. Ballew, W.D., "B(2):P(m) Dual Radix Systems- Theory, Design, and I^2L Implementation," Ph.D. Thesis, University of Oklahoma, 1981.

12. Barbe, D.F., Very Large Scale Integration (VLSI) Fundamentals and Applications, Springer-Verlag Berlin Heidelberg New York, 1981.

13. Baskin, H.B., "Design of N-valued Logic Network," AIEE Conf. 1962, Paper CP-62-500.

14. Baskin, H.B., "N-valued Logic Circuit," IBM Technical Disclosure Bull., 3, 81, March 1961.

15. Beach, A., and Armstrong, J.R., "Results of a Chip Layout Study for Multivalued I^2L," Proc. Twelfth Int. Symp. on Multiple-Valued Logic, Paris, France, 1982, pp. 55-68.

16. Berlin, R.D., Synthesis of N-valued Switching Circuits," IRE Trans. on Electronic Computers, 7, 1958, pp. 52-56.

17. Bernstien, B.A., "Representation of Three-element Algebras," Amer. Jour. Math., 46, 1924, pp. 110.

18. Birk, J., and Farmer, D., "An Algebraic Method for Designing Multi-Valued Logic Circuits using Principally Binary Components," IEEE Trans. on Computers, Vol. c-24, November 1975, pp. 1101-1104.

19. Bjorner, D., "Flowchart Machines," IBM Research Laboratory, San Jose, Calif., RJ-685 (No. 13346), April 7, 1970.

20. Booth, A.D., and Ringrose, J., "A Three-state Flip-flop," Electronic Engng., 23, 1951, pp. 133.

21. Braddock, R., Epstein, G., and Yamanaka, H., "Multiple-Valued Logic Design and Applications in Binary Computers," Conf. Rec. of the 1971 Symp. on the Theory and Applications of Multiple-valued Logic Design, New York, 1971, pp. 13-25.

22. Breuer, M.A., and Friedman, M.D., Diagnosis and reliable Design of Digital Systems. Woodland Hills, CA: Computer Science Press, 1976.

23. Brusentzov, N.P., "Development of Ternary Computer," Vestnik Mosk. Univ. Math. Mech., 2, 1965, pp. 39-48.

24. Butler, J.T., "Enumeration of Functions Realized by Fanout-Free Networks of General Multi-Valued Gates," Proc. the Ninth Int. Symp. on Multiple-Valued Logic, Bath, England, 1979, pp. 94-103.

25. Butler, J.T., "Fanout-Free Networks of Multivalued Gates," Proc. the seventh Int. Symp. on Multiple-Valued Logic, North Carolina, 1977, pp. 39-46.

26. Capece, R.P., "Tackling the Very Large-Scale Problems of VLSI: a Special Report," Electronics, November 23, 1978, PP. 111-125.

27. Carnes, J.E., "Fundamental Packing Density Limitations of CCD and RAM Memories," Electro 1977, Sec. 19, April 1977, pp. 19.4-1/9.

28. Cheung, P.T., and Purvis, D.M., "A Computer-Oriented Heuristic Minimization Algorithm for Multiple-output Multi-valued Switching Functions," Proc. 1975 Int. Symp. on Multiple-valued Logic, Indiana, 1975, pp. 112-120.

29. Chiang, K.W., and Vranesic, Z.G., "Fault Detection in Ternary NMOS and CMOS Circuits," Proc. twelfth on Multiple-Valued Logic, Paris, France, 1982, pp. 129-138.

30. Clare, C.R., Designing Logic Systems Using State Machines, McGraw-Hill, New York, 1973.

31. Coy, W., "A Common Approach to the Description Implementation and Test Generation of Multi-Valued Functions," Proc. Eleventh Int. Symp. on Multiple-valued Logic, Oklahoma, 1981, pp. 90-94.

32. Current, K.W., "Quaternary-to-Analog Converters," Proc. twelfth Int. Symp. on Multiple-valued Logic, France, 1982, pp. 4-7.

33. Current, K.W., Wheaton, L.B., Luich, T.M., and Mow, D.A., "Characteristics of Integrated Quaternary Threshold Logic Full Adders," Proc. Tenth Int. Symp. on Multiple-Valued Logic, Evanston, ILL, 1980, pp. 24-30.

34. Current, K.W., and Mow, D.A., "Four-Valued Threshold Logic Full Adder Circuit Implementation," Proc. Eight Int. Symp. on Multiple-Valued Logic, Rosemont, ILL, 1978, pp. 95-100.

35. Current, K.W., and Mow, D.A., "Implementing Parallel Counters with Four-Valued Threshold Logic," IEEE Trans. on Computers, Vol. c-28, No. 3, March 1979, pp. 200-204.

36. Current, K.W., and Mow, D.A., "Proposed Digital Correlator Design with Four-Valued Threshold Logic," IEEE J. on Electronic Circuits and Systems, Vol. 2, No. 4, July 1978, pp. 115-120.

37. Curtis, H.L., A New Approach to the Design of Switching Circuits, D.Van Nostrand Company, Inc., Princeton, New Jersey, 1962.

38. Dahl, V., "A Three-valued Logic for Natural Language Computer Applications," Proc. tenth Int. Symp. on Multiple-valued Logic, ILL, 1980, pp.102-107.

39. Daniel, M.E., and Gwyn, C.W., "CAD Systems for IC Design," IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, Vol. CAD-1, No. 1, January 1982, pp. 2-12.

40. Dao, T.T., "Recent Multi-Valued Circuits," Dig. of Papers VLSI, COMPCON81, February 1981, pp. 194-203.

41. Dao, T.T., McCluskey, E.J., and Russell, L.K., "Multivalued Integrated Injection Logic," IEEE Trans. on Computers, Vol. c-26, No. 12, Dec 1977, pp. 1233-1241.

42. Deschamps, J.P., and Thayse, A., "The Module Structure of Discrete Functions," Proc. Seventh Int. Symp. on Multiple-valued Logic, North Carolina, 1977, pp. 14-19.

43. Druzeta, A., Vranesic, Z.G., and Sedra, A.S., "Application of Multi-Threshold Elements in the Realization of Many-Valued Logic Networks," IEEE Trans. on Computers, Vol. c-23, 1974, pp. 1194-1198.

44. Dussault, J., Metze, G., and Krieger, M., "A Multivalued Switching Algebra with Boolean Properties," Proc. Sixth Int. Symp. on Multiple-valued Logic, Utah, 1976, pp. 68-74.

45. Economou, N.P., "Developing a Technology Base for Advanced Devices and Circuits," Proceedings of the IEEE, Vol. 71, NO. 5, May 1983, PP. 601-611.

46. Eichelberger, E.B., and Williams, T.W., "A Logic Design Structure for LSI Testability," 14th Design Automation Conf., 1977, pp. 462-468.

47. Epstein, G., "The Lattice Theory of Post Algebras," Trans. Amer. Math. Soc., 95, 1960, pp. 300-317.

48. Etiemble, D., "TTL Circuits for Four Valued Bus," Proc. Eighth Int. Symp. on Multiple-valued Logic, ILL., 1978, pp. 7-13.

49. Etiemble, D., and Israel, M., "A New Concept for Ternary Logic Elements," in Proc. 1974 Int. Symp. Multiple-Valued Logic, May 1974, pp.437-455.

50. Etiemble, D., and Israel, M., "Implementation of Ternary Circuits with Binary Integrated Circuits," IEEE Trans. on Computers, Vol. c-26, No. 12, December 1977, pp. 1222-1233.

51. Etiemble, D., and Israel, M., "Some New Results for Ternary Circuits," Proc. Ninth Int. Symp. on Multiple-Valued Logic, Bath, England, 1979, pp. 167-169.

52. Etiemble, D., and Israel, M., "T.S.C. Multivalued TTL Circuits," Proc. Tenth Int. Symp. on Multiple-Valued Logic, Evanston, ILL, 1980, pp. 31-35.

53. Fang, K.Y., and Wojcik, A.S., "An Approach to the Modular Design of Multiple-Valued logic Functions," Proc. the Twelfth Int. Symp. on Multiple-Valued Logic, Paris, France, 1982, pp. 260-266.

54. Frieder, G., Fong, A., and Chao, C.Y., "A Balanced Ternary Computer," Conf. Rec. of the 1973 Int. Symp. on Multiple-valued Logic, Canada, 1973, pp. 68-88.

55. Funatsu, S., Wakatsuki, N., and Arima, T., "Test Generation Systems in Japan," in Proc. 12th Design Automation Symp., June 1975, pp. 114-122.

56. Greniewski, M., "The Use of Trivalent Logics in the Theory of Automatic Mechnisms: Realization of the Fundamental Functions by Circuits," Com. Acad. R.P. Romine, 6, 1956, pp. 225-229.

57. Grosh, H.R.J., "Signed Ternary Arithmetic," Memorandum M-1496, Digital Computer Lab., MIT Cambridge, 1952.

58. Hallworth, R., and Heath, G., "Semiconductor Circuits for Ternary Logic," Proc. Inst. Elec. Engrs., November, 1961.

59. Hamacher, V.C., and Vranesic, Z, "Multivalued Versus Binary High Speed Multipliers," Conf. Rec. of the 1971 Symp. on the Theory and Applications of Multiple-valued Logic Design, New York, 1971, pp. 42-53.

60. Hanson, W.H., "Ternary Threshold Logic," IRE. Trans. on Electronic Computers, Vol. EC-12, June, 1963, pp. 191-197.

61. Hatzakis, M., "Resists for Fine-Line Lithography," Proceedings of the IEEE, Vol. 71, No.5, May 1983, PP. 570-574.

62. Healy, J.T., "Automatic Testing and Evaluation of Digital Integrated Circuits, Reston Publishing Company, Inc, Reston, Virginia, 1981.

63. Henle, R.A., "Multistable Transistor Circuit," Trans. A.I.E.E., November, 1955, pp. 568-570.

64. Herriott, D.R., "The Development of Device Lithography," Proceedings of the IEEE, Vol. 71, No. 5, May 1983, PP. 566-570.

65. Higuchi, T., and Hoshi, H., "Special Propose Ternary Computer for Digital Filtering," Proc. Eighth Int. Symp. on Multiple-valued Logic, ILL., 1978, pp. 47-54.

66. Higuchi, T., and Kameyama, M., "Static-Hazard-Free T-Gate for Ternary Memory Element and its Application to Ternary Counters," Proc. of the sixth Int., Symp. on Multiple-Valued Logic, Utah 1976, pp. 127-134.

67. Higuchi, T., and Kameyama, M., "Synthesis of Multiple-Valued Logic Networks Based on Tree-Type Universal Logic Modules," Proc. of the 1975 Int. Symp. on Multiple-Valued Logic, Bloomington, 1975, pp. 121-130.

68. Higuchi, T., and Kameyama, M., "Ternary Logic System Based on T-Gate," Proc. of the 1975 Int. Symp. on Multiple-Valued Logic, Bloomington, 1975, pp. 290-304.

69. Hirata, K., Ozaki, Y., Oda, M. and Kimizuka, M., "Dry Etching Technology for 1-um VLSI Fabrication," IEEE Trans. on Electron Devices, Vol. ED-28, No. 11, November 1981, PP 1323-1331.

70. Hobson, G.S., Charge -Transfer Devices, John Wiley & Sons, New York, 1978.

71. Hu, M., and Smith, K.C., "A New Type of Self-Checking Synchronous Sequential Machine Based on 2-of-3-valued Logic Circuits," Proc. Twelfth Int. Symp. on Multiple-valued Logic, France, pp. 139-145.

72. Huertas, J.L., and Carmona, J.M. "Low-Power Ternary C-MOS Circuits," Proc. Ninth Int. Symp. on Multiple-Valued Logic, Bath, England, 1979, pp. 170-174.

73. Hurst, S.L., "The Harr Transform in Digital Network Synthesis," Proc. Eleventh Int. Symp. on Multiple-valued Logic, Oklahoma, 1981, pp. 10-18.

74. Irving, T.A., Jr., "Memory Elements for Multiple Valued Sequential Circuits," Ph.D. Dissertation, Auburn University, Alabama, August 24, 1973.

75. Janczewski, L.J., "Geometrical Approach to Multi-valued Logic Function Synthesis," Conf. Rec. of the 1973 Int. Symp. on Multiple-valued Logic, Canada, 1973, pp. 106-118.

76. Johnson, K.C., "A three-state Flip-flop," Electronic Engng., 23, 1951.

77. Kabat, C.W., and Wojcik, A.S., "On the Design of 4-valued Digital Systems," Proc. the tenth Int. Symp. on Multiple-Valued Logic, Evanston, Ill, 1980, pp. 153-170.

78. Kallmann, H.E., "Resistors Having Stair-shaped Characteristics," Trans. on Computers, July, 1965, p. 763.

79. Kallmann, H.E., "Zero Loop Resistance Circuitry Including Multistate Memory Circuits," Trans. on Computers, July, 1963, p. 762.

80. Kameyama, M., and Higuchi, T., "A New Digital Image Processor Using Multiple-Valued Logic," Proc. Twelfth Int. Symp. on Multiple-valued Logic, France, 1982, pp. 8-16.

81. Kameyama, M., and Higuchi, T., "Design of Radix 4 Signed-Digit Arithmetic Circuits for Digital Filter," Proc. Tenth Int. Symp. on Multiple-valued Logic, ILL, 1980, pp. 272-277.

82. Kameyama, M., and Higuchi, T., "Practical State Assignment for Multiple-Valued Synchronous Sequential Circuits," Proc. Seventh Int. Symp. on Multiple-valued Logic, North Carolina, 1977, pp. 70-76.

83. Kameyama, M., and Higuchi, T., "Synthesis of Optimal T-Gate Networks in Multiple-Valued Logic," Proc. the Ninth Int. Symp. on Multiple-Valued Logic, Bath, England, 1979, pp. 190-195.

84. Karp, J.M., "Functional Decomposition and Switching Circuit Design," J. Soc. Indust. Appl. Math II (2), June, 1963.

85. Karpovsky, M., "Spectral Methods for Decomposition, Design anf Testing of Multiple-Valued Logical Networks," Proc. Eleventh Int. Symp. on Multiple-valued Logic, Oklahoma, 1981, pp. 1-9.

86. Karpovski, M., "Testing for Multiple-Valued Computations," Proc. Twelfth Int. Symp. on Multiple-valued Logic, France, 1982, pp. 77-80.

87. Kerkhoff, H.G., and Tervoert, M.L., "The Implementation of Multiple-Valued Functions Using Charge-Coupled Devices," Proc. Tenth Int. Symp. on Multiple-Valued Logic, Evanston, ILL, 1980, pp. 6-15.

88. Knudsen, M.S., "A Nine-Valued Logic Simulator for Digital NMOS Circuits," Proc. Twelfth Int. Symp. on Multiple-valued Logic, France, 1982, pp. 293-297.

89. Koenemann, B., Mucha, J., and Zwiehoff, G., "Built-In Logic Block Observation Techniques," in Dig. Papers, 1979 Test Conf., IEEE Pub. 79CH1509-9C, Oct. 1979, pp. 37-41.

90. Kohout, L., "Application of Multi-valued Logics to the Study of Human Movement Control and of Movement Disorders," Proc. Sixth Int. Symp. on Multiple-valued Logic, Utah, 1976, pp. 224-232.

91. Kool, C.F., and Weaver, J.L., "Nonlinear Hall Effect Ternary Logic Element," Solid-state Electronics, Pergamon Process, Vol. 7, 1964, pp. 311-321.

92. Lam, C.L., and Vranesic, Z.G., "A Hashing Method Using Multi-Valued Feedback Shift Registers," Proc. Seventh Int. Symp. on Multiple-valued Logic, North Carolina, pp. 116-119.

93. Ledley, R.S., and Huang, H.K., "Multivalued Logic Design and Postian Matrices," Proc. 1975 Int. Symp. on Multiple-valued Logic, Indiana, 1975, pp. 67-75.

94. Lee, C.Y., and Chen, W.H., "Several-valued Combinatorial Switching Circuits," Trans. Amer. Inst. Electr. Eng. 75 (AI) (25), July, 1956, pp. 278-283.

95. Lee, S.C., "Vector Boolean Algebra and Calculus,"IEEE Trans. on Computers, Vol. C-25, Sep., 1976, pp. 865-874.

96. Lee, S.C., Digital Circuits and Logic Design. Englewood Cliffs, N.J.: Prentice-Hall, 1976.

97. Lee, S.C., and Tull, M.P.,"A New Method For Realizing Parallel Processing Machines Using Multiple-valued Logic," Proc. Tenth Int. Symp. on Multiple-valued Logic, ILL., 1980, pp. 36-44.

98. Lee, S.C., and Ajabnoor, Y.M., "Digital Calculus," Proc. Eighth Int. Symp. on Multiple-valued Logic, ILL., 1978, pp.128-139.

99. Lee, S.C., and Ajabnoor, Y.M., "Static Hazard Detection and Elimination in Multi-Level Combinational Circuits," Proc. Tenth Int. Symp. on Multiple-valued Logic, ILL, 1980, pp. 187-194.

100. Lee, S.C., and Ballew, W.D., "P(m) Dual Radix Logic and Its I^{2}L Implementation," Proc. Eleventh Int. Symp. on Multiple-valued Logic, Oklahoma, 1981, pp. 290-298.

101. Lee, S.C., and Keren-Zvi, Y., "A Generalized Boolean Algebra and Its Application to Logic Design," Proc. 1975 Int. Symp. on Multiple-valued Logic, Indiana, 1975, pp. 88-98.

102. Lee, S.C., and Lee, E.T., "On Multivalued Symmetric Functions," IEEE Trans. on Computers, March, 1972.

103. Lee, S.C., and Lo, Hao-Yung, "A Map-partition Method for the Fault Detection on Multi-level Combinational Logic Circuits," Proc. Eleventh Int. Symp. on Multiple-valued Logic, Oklahoma, 1981, pp. 283-289.

104. Lepselter, M.P., Alles, D.S., Levinstein, H.J., Smith, G.E., and Watson, H.A., "A System Approach to 1-μm NMOS," Proceedings of the IEEE, Vol. 71, No. 5, May 1983, PP. 640-656.

105. Liebler, M., and Roesser, R., "Multiple-Real-Valued Walsh Functions," Conf. Rec. of the 1971 Symp. on the Theory and Applications of Multiple-valued Logic Design, New York, 1971, pp. 84-102.

106. Linear and Digital Semi-Custom IC Design Programs, EXAR Integrated Systems, Inc., Sunnyvale, CA., 1980.

107. Lo, H.Y., "Generalized Fault Detection for Multivalued Logic Systems," Proc. Tenth Int. Symp. on Multiple-valued Logic, ILL, 1980, pp. 178-186.

108. Lukasiewicz, J., and Tarski, A, "Untersuchungen uber den Aussagenkalkul," C.R. Soc. Sci. Lett. Varsovic, Classe III, vol. 23, 1930, pp. 1-21.

109. Lukasiewicz, J. "O Logice Trojwartosciocvej (On 3-valued logic)," Ruch Filosoficzny 5, 1920, pp. 169-171.

110. MCS-85 User's Manual, Intel Corporation, 1977.

111. Mackay, R.S., and McIntyre, R., "Ternary Counters," Trans. IRE, Vol. EC-4, December, 1955, pp. 144-149.

112. Maitra, "Cascaded Switching Networks of Two-input Flexible Cells," IRE Trans. on Electronic Computers, Vol. EC-11 42, April, 1962, pp. 136-143.

113. Mattrey. R.F., Givone, D.D., and Allen, C.M., "Applying Multiple-valued algebra Concepts to Neural Modeling," Conf. Rec. of the 1973 Int. Symp. on Multiple-valued Logic, Canada, 1973, pp. 127-136.

114. McCluskey, E.J., "A Discussion of Multiple-Valued Logic Circuits," Proc. Twelfth Int. Symp. on Multiple-Valued Logic, Paris, France, 1982, pp. 200-205.

115. McCluskey, E.J., "Logic Design of MOS Ternary Logic," Proc. Tenth Int. Symp. on Multiple-Valued Logic, Evanston, ILL, 1980, pp. 1-5.

116. McCluskey, E.J., "Logic Design of Multi-Valued I^2L Logic Circuits," Proc. the Eight Int. Symp. on Multiple-Valued Logic, Rosemont, Ill, 1978, pp. 14-22.

117. McCluskey, E.J., "Logic Design of Multivalued I^2L Logic Circuits," IEEE Trans. on Computers, C-28, No.8, Aug. 1979, pp. 546-559.

118. McCluskey, E.J., "Verification Testing," ACM IEEE Nineteenth Design Automation Conf. Proceedings, Las Vegas, Nevada, June 14-16, 1982, pp. 495-500.

119. McCluskey, E.J., and Bonzorgui-Nesbat, S., "Design for Autonomous Test," IEEE Trans. on Computers, Vol. C-30, No. 11, November 1981, pp. 866-875.

120. McDonald, J.F., "Extensions of the Weiner-Smith Algorithm for m-ary Synchronous Sequential Circuit Design," Proc. 1974 Int. Symp. on Multiple-valued Logic, Virginia, 1974, pp. 135-154.

121. Mei, K.C.Y., "Bridging and stuck-at-Faults," IEEE Trans. on Computers, Vol. C-23, No. 7, July 1974, pp. 720-727.

122. Meo, A.R., "Modular Tree Structures," IEEE Trans on Computers, Vol. c-17, May 1968, pp. 432-442.

123. Merrill, R.D., "A Tabular Munimising Procedure for Ternary Switching Functions," IEEE Trans. on Computers, Vol. EC-15, August, 1966, pp. 578-585.

124. Miller, D.M., "Decomposition in Many-Valued Design," Ph.D. Thesis, University of Minitoba, March, 1976.

125. Miller, D.M., "Spectral Symmetry Tests," Proc. Eleventh Int. Symp. on Multiple-valued Logic, Oklahoma, 1981, pp. 130-134.

126. Miller, D.M., "Spectral Signature Testing for Multiple-Valued Combinational Networks," Proc. Twelfth Int. Symp. on Multiple-valued Logic, France, 1982, pp. 152-158.

127. Miller, D.M., "The Fanout-Free Realization of Multiple-Valued Functions," Proc. The Eleventh Int. Symp. on Multiple-Valued Logic, University of Oklahoma, 1981, pp. 246-255.

128. Miller, D.M., and Muzio, J.C., "A Ternary Cellular Array," Proc. 1974 Int. Symp. on Multiple-valued Logic, Virginia, 1974, pp. 469-482.

129. Miller, D.M., and Muzio, J.C., "On the Minimization of Many-valued Functions," Proc. Ninth Int. Symp. on Multiple-valued Logic, England, 1979, pp. 294-299.

130. Moisil, G.C., "The Use of Trivalent Logics in the Theory of Automatic Mechanisms. II. Characteristic Equation of a Polarized Relay," Com. Acad. R.P. Romine 6, 1956, pp. 231-234.

131. Moisil, G.C., "The Use of Trivalent Logics in the Theory of Automatic Mechanisms. III. Circuits with Real Contacts," Com. Acad. R.P. Romine 6, 1956, pp. 385-386.

132. Moisil, G.C., "The Use of Trivalent Logics in the Theory of Automatic Mechanisms. IV. Realization of Work Functions During Real Operation," Com. Acad. R.P. Romine 6, 1956, pp. 971-973.

133. Moore, G.E., "Are We Really Ready for VLSI," Caltech Conf. on VLSI, 1979, pp. 3-14.

134. Moraga, C., and Bittner, A., "Ternary Positional Automatic Control System," Cof. Rec. of the 1973 Multiple-valued Logic, Canada, 1973, pp. 137-155.

135. Moraga, C., "Ternary Spectral Logic," Proc. Seventh Int. Symp. on Multiple-valued Logic, North Carolina, 1977, pp. 7-12.

136. Morris, D.J., and Alexander, W., "An Introduction to the Ternary Code Number System," Electronic Engineering (32), 1960, p. 554.

137. Mouftah, H.T., and Jordan, I.B., "Integrated Circuits for Ternary Logic," Proc. 1974 Int. Symp. on Multiple-Valued Logic, West Virginia, May 1974, pp. 285-302.

138. Mouftah, H.T., and Smith, K.C., "Ternary Logic in a Positional Control System," Proc. Sixth Int. Symp. on Multiple-valued Logic, Utah, 1976, pp. 135-141.

139. Mucha, J., "Hardware Techniques for Testing VLSI Circuits Based on Built-In Test," in Dig. of Papers VLSI, IEEE Pub. 81-CH1626-1, Feb. 1981, pp. 366-369.

140. Muehldorf, E.I., "Circuits for Ternary Switching Variables," AEU 12, 1958, pp. 176-182.

141. Mukhopadyay, A., "Symmetric Ternary Switching Functions," IEEE Trans. on Computers, Vol. EC-15 (5), October, 1966, pp. 731-739.

142. Muzio, J.Z., and Bate, J.A., "Three Cell Structures for Ternary Cellular Arraya," Proc. Sixth Int. Symp. on Multiple-valued Logic, Utah, 1976, pp. 55-60.

143. Nadig, H.J., "Signature Analysis-Concepts, Examples, and Guide-Lines," Hewlett-Packard J., May 1977, pp. 15-21.

144. Niessen, C., "Hierarchical Design Methodologies and Tools for VLSI Chips," Proceeding of the IEEE, Vol. 71, No.1, January 1983, pp. 66-75.

145. Pao, Y.H., and Altman, J., "Use of Associative Memory Techniques in Implementation of Multi-valued Logic System," Proc. Sixth Int. Symp. on Multiple-valued Logic, Utah, 1976, pp. 93-96.

146. Papachristou, C.A., "Content-Addressable Memory Requirements for Multivalued Logic," Proc. Eleventh Int. Symp. on Multiple-valued Logic, Oklahoma, 1981, pp. 62-72.

147. Pomper, G., and Armstrong, J.R., "An Efficient Multivalued Minimization," Proc. Ninth Int. Symp. on Multiple-valued Logic, England, 1979, pp. 300-304.

148. Porat, D.I., "Three Valued Digital Systems," Proc. IEE. Vol. 116, No. 6, June, 1969, pp. 947-954.

149. Post, E.L., "Introduction to a General Theory of Elementary Propositions," American J. of Math., Vol. 43, 1921, pp. 163-185.

150. Pradhan, D.K., "A Multivalued Switching Algebra Based on Finite Fields," Proc. 1974 Int. Symp. on Multiple-value Logic, Virginia, 1974, pp. 95-112.

151. Pugsley, J.H., and Silio, C.B., Jr., "Some I^2L Circuits for Multiple-Valued Logic," Proc. Eight Int. Symp. on Multiple-Valued Logic, Rosemont, ILL, 1978, pp. 23-31.

152. Raymond, T.C., "LSI/VLSI Design Automation," IEEE Computer Tutorial Series 10, July 1981, pp. 89-101.

153. Rideout, V.L., " Limits to Improvement of Silicon Integrated Circuits," Dig. of Papers VLSI: New Architectural Horizons, COMPCON 80, IEEE Pub., Catalog No.

80CH1491-OC, Feb. 1980, pp. 2-6.

154. Right, R., "The Switching Algebra of Electrical Circuits with Multi-position Switch Elements," Ingegneria Ferroviara, 10, 1958, pp. 881-898; 11, 1958, pp. 1019-1030.

155. Rine, D.C., "Picture Processing Using Multiple-Valued Logic," Proc. Eleventh Int. Symp. on Multiple-valued Logic, Oklahoma, 1981, pp. 73-78.

156. Romeo, D.E., and Schuegraf, K.K., "I^2L/LSI for Complex Logic Arrays," Wescon 77, Section 11.

157. Rosenbloom, P.C., "Past Algebras. I. Postulates and General Theory," Am. J. Math. 64, 1942, pp. 167-188.

158. Russell, L.K., "Multi-Level NMOS Circuits," Proc. COMP. Con. 1981.

159. Salter, F., "A Ternary Memory Element Using a Tunnel Diode," IEEE Trans. on Electronic Computers, April, 1964.

160. Santos, J., Arango, H., and Lorenso, F., "Threshold Synthesis of Ternary Digital Systems," IEEE Trans. on Electronic Computers, Vol, EC-15, February, 1966, pp. 105-107.

161. Santos, J., Arango, H., and Pascual, M., "A Ternary Storage Element Using a Conventional Ferrite Tore," Proc. IEEE , Vol. EC-13, October, 1964, pp. 608-609.

162. Santos, J., and Arango, H., "A Fast Carry Propagation Circuit for Base 3 Signed Non-redundant Arithmetic," IEEE Trans. on Electronic Computers, April, 1966, p. 254.

163. Santos, J., and Arango, H., "A Graphic Method for the Synthesis of Threshold Ternary Functions," IEEE Trans. on Computers, Oct., 1970, pp. 975-976.

164. Santos, J., and Arango, H., "On the Analysis and Synthesis of 3-valued Digital Systems," AFIPS Conf. Proc. Vol. 25, 1964, pp. 463-475.

165. Savir, J., "Syndrome- Testable Design of Combinational Circuits," IEEE Trans. on Computers, Vol. C-29, June 1980, pp. 442-451, (corrections: Nov. 1980).

166. Savir, J., "Syndrome-Testing of "Syndrome-Untestable" Combinational Circuits," IEEE Trans. on Computers, Vol. C-30, No. 8, August 1981, pp. 606-608.

167. Sequin, C.H., and Tompsett, M.F., Charge Transfer Devices, Academic Press, Inc. New York, 1975.

168. Shannon, C.E., "A Symbolic Analysis of Relay and Switching Circuits," Trans. AIEE, 57, 1938, pp. 713-723.

169. Sheppard, D.A., "An Application of Many-valued Logic to Fault Detection," Conf. Rec. of the 1973 Int. Symp. on Multiple-valued Logic, Canada, 1973, pp. 192-204.

170. Siegel, B.M., and Hanson, G.R., "Ion Sources and Lithography," Proceedings of the IEEE, Vol. 71, No. 5, May 1983, PP. 591-592.

171. Silio, C.B., Pugsley, J.H., and Jeng, B.A., "Control Memory Reduction Using Multivalued ROM's," Proc. Ninth Int. Symp. on Multiple-valued Logic, England, 1979, pp. 19-26.

172. Singh, A.D., Armstrong, J.R., and Gray, F.G., "Combinational and Sequential Multivalued Logic Design Using Universal Iterative Tree Structures," Proc. the Ninth Int. Symp. on Multiple-Valued Logic, Bath, England, 1979, pp. 182-189.

173. Smith, W.R. III., "Minimization of Multivalued Functions," Computer Science and Multiple-valued Logic, Chapter 8, Rine, D.C. (Editor), North-Holland, 1977.

174. Smith, W.R. III., "Programs KNPI, PITBL," University Bridgeport, 1974.

175. Smith, W.R. III., "Some Algebraic Methods and Minimization Techniques for Multi-valued Lattice Logics," Proc. 1972 Symp. on Theory and Applications of Multiple-valued Logic Design, New York, 1972, pp. 163-174.

176. Spillman, R., "Single Stuck-type Fault Detection in Multi-valued Combinational Circuits," Proc. Sixth Int. Symp. on Multiple-valued Logic, Utah, 1976, pp. 97-101.

177. Stark, M., "Two Bits Per Cell ROM," Dig. of Papers VLSI, COMPCON81, Feb. 1981, pp. 209-212.

178. Stewart, J.H., "Future testing of Large LSI Circuit Cards," in Dig. Papers 1977 Semiconductor Test Symp., IEEE Pub. 77CH1261-7C, Oct. 1977, pp. 6-17.

179. Strasilla, U.J., "A Multiple-valued Memory System Using Capacitor Storage," Proc. 1974 Int. Symp. on Multiple-valued Logic, Virginia, 1974, pp. 457-468.

180. Su, S.Y.H., and Cheung, P.T., "Computer-oriented Algorithm for Minimizing Multi-Valued Switching Functions," Conf. Rec. of the 1971 Symp. on the Theory and Applications of Multiple-valued Logic Design, New York, 1971, pp. 140-152.

181. Su, S.Y.H., and Cheung, P.T., "Computer Minimization of Multi-valued Switching Function," IEEE Trans. on Computers, 21(1972) pp. 995-1003.

182. Su, S.Y.H., and Cheung, P.T., "Computer Simplification of Multi-Valued Switching Functions," D.C. Rine, Editor Computer Science and Multiple-Valued Logic, Chapter 7, 1977.

183. Su, S.Y.H., and Sarris, A.A., "The Relationship Between Multivalued Switching Algebra and Boolean Algebra Under Different Definitions of Complement," IEEE Trans. on Computers, Vol. C-21, May, 1972, pp. 479-485.

184. Susskind, A.K., "Testing by Verifying Walsh Coefficients," in Proc. 11th Ann Symp. on Fault-Tolerant Computing (Portland, MA), June 1981, pp. 206-208.

185. Thelliez, S., "Application of Ternary Logic to Incremental Computation," IFAC Pulse Symp., Budapest, April, 1968.

186. Thelliez, S., "On the Arborescent Complex Disjunctive Decomposition of a Ternary Function with a View to its Application to the Synthesis of Ternary Combinatorial Structures," C.R. Acad. Sci. Paris 264, 27th, February, 1967, pp. 419-421.

187. Thelliez, S., Introduction to the Study of Ternary Switching Structures, Gordon and Breach Science Publishers, 1973.

188. Thelliez, S.,"Note on the Synthesis of Ternary Combinatorial Networks Using T-gate Operators," Electronic Letters, May, 1967.

189. Trampel, K.M., "Two-transistor Ternary Inverter," IBM Tech. Disclosure Bull., 4, July, 1961, pp. 48-49.

190. Tokmen, V.H., and Hurst, S.L., "A Consideration of Universal Logic Modules for Ternary Synthesis Based Upon Reed-Muller," Proc. Ninth Int. Symp. on Multiple-valued Logic, England, 1979, pp. 248-256.

191. Tomabechi, N., Kameyama, M., and Higuchi, T., "Efficient Residue Arithmetic Circuit Using Multiple-Valued Ring Counters and Its Application to Digital Signal Processing," Proc. Twelfth Int, Symp. on Multiple-

valued Logic, France, 1982, pp. 107-112.

192. Tront, J.G., and Givone, D., "An Implementation of Multiple-Valued Logic Gates Using MESFET's" Proc. Ninth Int. Symp. on Multiple-Valued Logic, Bath, England, 1979, pp. 175-181.

193. Tull, P., "A Quaternary Logic Simulator," Proc. Tenth Int. Symp. on Multiple-valued Logic, ILL, 1980, pp. 171-177.

194. Tyal, R.V., and Liechti, C., "Gallium Arsennide Spawns Speed," IEEE Spectrum, March 1977, pp. 41-47.

195. Tyal, R.V., and Liechti, C., "High-Speed Integrated Logic with GaAs MESFET's," IEEE J. of Solid-State Circuits, Vol. SC-9, No. 5, Oct. 1974, pp. 269-276.

196. Vacca, R., "A Three-valued System of Logic and Its Applications to Base Three Digital Circuits," Proc. Int. Conf. on Information Processing, Paris, June 15-20, 1959. Paris, UNESCO, 1960, pp. 407-414.

197. Varnell, G.L., Shah, P.L., and Havemann, R.H., "MOS and Bipolar VLSI Technologies Using Electron-Beam Lithography," Proceedings of the IEEE, Vol. 71, No. 5, May 1983, PP. 612-639.

198. Voith, R.P., "ULM Implicants for Minimization of Universal Logic Module Circuits," IEEE Trans. on Computers, Vol. c-26, May 1977, pp. 417-424.

199. Vranesic, Z.G., "A Multi-Valued Switching Theory," Ph.D. Thesis, University of Toronto, April 1968.

200. Vranesic, Z.G., "Applications and Scope of Multiple-Valued LSI Technology," Dig. of Papers VLSI, COMPCON81, February 1981, pp. 213-216.

201. Vranesic, Z.G., Lee, E.S., and Smith, K.C., "A Many-valued Algebra for Switching Systems," IEEE Trans. on Computers, Vol. C-19, October, 1970, pp. 964-971.

202. Vranesic, Z.G., Smith, K.C., and Druzeta, A., "Electronic Implementation of Multi-valued Logic Networks," Proc. 1974 Int. Symp. on Multiple-valued Logic, Virginia, 1974, pp. 59-78.

203. Waliuzzaman, K.M., and Vranesic, Z.G., "On Decomposition of Multi-Valued Switching Functions," Comp. J., Vol. 13, November, 1970, pp. 359-362.

204. Webb, D.L., "Generation of Any n-valued logic by One Binary Operation," Proc. Nat. Acad. Sci. (U.S.A.) 21, May, 1935, pp. 252-254.

205. Williams, T.W., and Parker, K.P., "Design for Testability- A Survey," Proceeding of the IEEE, Vol. 71, No. 1, January 1983, pp. 98-112.

206. Wills, M.S., "A Study of Multi-Valued Logic R-Flops," Master Thesis, Illinois Institute of Technology, Chicago, Illinois, May 1977.

207. Windecker, R.C., "Stochastic Multiple-Valued Combinational Networks," Proc. Ninth Int. Symp. on Multiple-valued Logic, England, 1979, pp. 222-231.

208. Wojciechowski, W., "Multiple-Valued Combinational Logic Design Using Theorem Proving," Ph.D. Thesis, Computer Science Dept. ITT, May 1980.

209. Wojciechowski, W., "Structured Digital Design in Multiple-Valued Logic," Proc. Twelfth Int. Symp. on Multiple-valued Logic, France, 1982, pp. 206-222.

210. Wojcik, A.S., and Wojciechowski, W., "Multiple-Valued Logic Design by Theorem Proving," Proc. Ninth Int. Symp. on Multiple-valued Logic, England, 1979, pp. 196-199.

211. Wojcik, A.S., "Multi-valued Asynchronous Circuits," Conf. Rec. of the 1973 Int. Symp. on Multiple-valued Logic, Canada, 1973, pp. 217-227.

212. Wolf, E.D., Adesida, I., and Chinn, J.D., "Dry Etching for Submicron Structures," Proceedings of the IEEE, Vol. 71, No. 5, May 1983.

213. Xu, S., and Su, S.Y.H., "Testing Feedback Bridging Faults Among Internal Input and Output Lines by Two Patterns," ICCC 82 Proceedings, New York, Sep. 28-Oct. 1, 1982, pp. 214-217.

214. Yamada, M., "A New Multilevel Storage Structure for High Density CCD Memory," IEEE J. of Solid State Circuits, SC-13, Oct. 1978, pp. 688-692.

215. Yang, C.T., and Wojcik, A.S., "A Logic Simulator for 3-Valued Digital Systems," Proc. Seventh Int. Symp. on Multiple-valued Logic, North Carolina, 1977, pp. 47-50.

216. Yang, C.T., and Wojcik, A.S., "Parallel and Serial Decompositions of Multi-Valued Sequential Machines," Proc. Eighth Int. Symp. on Multiple-valued Logic, ILL, 1978, pp. 179-186.

217. Yau, S.S., and Tang, C.K., "Universal Logic Circuits and Their Modular Realizations," 1968 Spring Joint Computer Conference, AFIPS Proc., Vol. 32, Washington, D.C.: Thompson, 1968, pp. 297-305.

218. Yau, S.S., and Tang, C.K., "Universal Logic Modules and Their Applications," IEEE Trans. on Computers, Vol. c-19, February 1970, pp. 141-149.

219. Yoeli, M., and Halpern, I., "Ternary Arithmetic Unit," Proc. IEE, Vol. 115, No. 10, October, 1968, pp. 1385-1388.

220. Yoeli, M., and Rosenfeld, G., "Logic Design of Ternary Switching Circuits," IEEE Trans. on Electronic Computers, Vol. EC-14, February, 1965, pp. 19-29.

221. Yoeli, M., and Shlomo, R., "Application of Ternary Algebra to the Study of Static Hazards," J.A.C.M. 11 (1), January, 1964, pp. 84-97.

222. Zavisca, E.G., and Allen, C., "An Approach to Multiple-Valued Sequential Circuit Synthesis," Conf. Rec. of the 1971 Symp. on the Theory and Applications of Multiple-valued Logic Design, New York, pp. 206-218.

223. Zavisca, E.G., "Synthesis Techniques in Multiple-Valued Logic Systems," Ph.D. Thesis, University of New York at Buffalo, 1971.

224. vanCleemput, W.M., "Hierarchical Design for VLSI: Problems and Advantages," Caltech Conference on VLSI, January 1979, pp. 259-274.

APPENDIX A

RELEVANT PAST WORK

J.Lukasiewicz [109] (1920) was the first pioneer to publish on three-valued symbolic logic. The next, year E.L. Post [149] introduced a general theory of elementary propositions which was one of the first publications on multi-valued logics. He gave a definition of m-valued logic that was a generalization of the usual two-valued calculus. He defined the most important operations and discussed some of their properties by means of the tables of values.

B.A. Bernstein [17] (1924) presented three-element algebras. His publication gave a representation of a complete or incomplete truth table by a polynomial defined from the sum and product of modulo 3 operations.

J. Lukasiewicz and Tarski [108] (1930) introduced m-valued logics which were included in the Post calculi, but not conversely. In fact, the Post logics are symbolically complete, i.e. all operations and relations in these systems definable by tables of values are definable in terms of the primitive ideas of these logics; however, the Lukasiewicz-Tarski system do not share this property.

D.L. Webb [204] (1935-1937) reduced the required number of undefined ideas to one and proved most of the important propositions of m-valued logic by numerical interpretation in term of the congruity. This consists, essentially, in using tables of values in a general way that eliminates trial and error.

P.C. Rosenbloom [157] (1941) introduced postulate-set for Post algebras and proved the fundamental theorems from the assumptions. The Post algebras are generalized in analogy to the extension from 2-element Boolean algebras to m-element Boolean algebras. He proved that two m-valued Post algebras with the same finite number of elements were simply isomorphic, and deduced, as a corollary his postulate-set was completed when a postulate to the number of elements was added.

A.D. Booth and J. Ringrose [20] (1951) were the first to publish on the realization of tristable memory using triode values, then K.C. Johnson [76] noted supplements of the analysis of tristable memory.

H.R.J. Grosh [57] (1952) studied the properties of the algebraic ternary representation of numbers applied to the arithmetic operations.

R.A. Henle [63] (1955) applied the principle of realization of multi-stable memory using closed loop systems containing nonlinearities to some transistorized circuits. This

was an interesting introduction to the design of different types of tristable trigger circuits. R.S. Mackay and R. McIntyre [111] described the design, using triode values, of a ternary pulse counter based on the use of the nonlinearities inherent in grid current. (A study of the problem of functional stability and of coupling between the stages has been completed.)

M. Greniewski [56] (1956) represented the states 0,1,2 by current -i,0,+i and by using three position relays; he also described the realization of various ternary functions of one or more variables. C.Y. Lee and W.H. Chen [94] studied the synthesis of combinational structures using operators with an application to the series of algebraic ternary adder. G.C. Moisil [130,131,132] applied 3-valued logics to the design of circuits using contacts and relays.

In 1958, the first full scale 3-valued computer was completed at Moscow state University in Soviet Union. R.D. Berlin [16] defined the concept of a functional set, and examined various functional sets and the corresponding trivial decompositions with a view of synthesis of m-ary combinatorial structures. After reviewing m-valued logics and ternary logics, E.I. Muehldorf [140] described two methods of synthesis of ternary combinatorial structures based on the existence of trivial disjunctive decompositions using the operators $J_a(x)$, $a \in \{0,1,2\}$, and maximum and minimum functions. He used the semiconductors for the reali-

zation of ternary logic circuits and applied to the ternary adder. R. Right [154] applied m-valued logic to circuits containing m-position switches.

After defining an example of complete functional combinatorial operators which comprised the maximum, minimum, $f_{210}(x)$, $f_{100}(x)$, and $f_{200}(x)$ functions and which could easily be realized by means of vacuum tubes, R. Vacca [196] (1959) examined the problem of realization of a ternary adder consisting of two half-adders. However, the problem of the representation of the numbers in ternary was not dealt with in his article; the adder structure described was not very economical.

J. Again [2] (1960) described the SETUN computer in which ternary logic was used. G. Epstein [47] used a number of operators larger than the number of operators chosen by the Rosenbloom to enable the set of axioms of the Post algebra to be simplified and also to allow this algebra to be studied with the aid of Boolean algebra and lattices. After reviewing a representation of numbers by means of a natural ternary code or an algebraic ternary code and remarking on algorithms for addition, subtraction and multiplication, D.J. Morris and W. Alexander [136] proposed the realization of ternary switching elements by means of magnetic tores.

H.B. Baskin [14] (1961) described a circuit made up with four tunnel diodes which might enable the realization of basic structure in a logic of m values. After reviewing

the advantages obtained by the use of m-ary bases in the digital systems, R. Hallworth and G. Heath [58] described ternary switching circuits realized by means of semiconductors. Designed on the basis of the elementary functions properties of the components, these circuits allow the realization of many functions. K.M. Trampel [189] described a logic circuit comprised of two transistors which realized the combinatorial function $f_{210}(x)$.

H.B. Baskin [13] (1962) studied the problem of the synthesis of the m-ary combinatorial structure by means of the maximum, minimum and J_k (x) operators. After remarking on the realization of the operators of the catalogue, two methods of synthesis were given. The first was based on the trivial decomposition of the given function and the second was the result of an examination of the truth table which represented the function.

D.J. Anderson and D.L. Dietmayer [7] (1963) used a ferrite core with two apertures with suitable winding and 3-phase excitation to obtain a large number of ternary combinatorial elements in a 0,1,2 = -v(t), 0, v(t) representation, where the quantities v(t) are current or voltage pulse. However, the introduction of combinatorial structures poses problems of insulation and excitation which did not seem to had been solved. W.H. Hanson [60] was the first to publish on the synthesis of ternary combinatorial structures using ternary threshold operators, but the methods of syn-

thesis were not very convenient. H.E. Kallmann [79] discussed the possibility of realizing memories with m stable states by means of nonlinear and negative resistances. J.M. Karp [84] studied the problem of the disjunctive and non-disjunctive decomposition of m-ary functions, and applied the results obtained by taking into account the properties of the given function to the synthesis of binary structures.

C.F. Kool and J.L. Weaver [91] (1964) studied the nonlinear Hall effect of the ternary logic element using the 0,1,2: -i, 0, +i representation, where i is the current. The nonlinear magnetoconductive properties of the semimetal of high mobility (bismuth) and the majority and dual $f_{210}(x)$ function ternary operators were realized economically. The dynamic characteristics of these elements were studied theoretically and experimentally. F. Salter [159] described the design of tristable trigger circuit by means of a tunnel diode. J. Santos, H, Arango and M. Pascual [161] described the design of a ternary memory by means of ferrite tore. J. Santos and H. Arango [164] gave an application of ternary logic to an adder, and studied a method of the synthesis of 2-layer structures. They also described a binary-coded ternary memory and the use of a tore memory in the ternary logic. M. Yoeli and R. Shlomo [221] described an application of the ternary logic to the detection of static switching hazards in the binary combinatorial structures.

N.P. Brusentzov [23] (1965) used fast magnetic amplif-

iers as physical elements and as an algebraic ternary representation of the numbers, and he also described the design of an adder in ternary logic. H.E. Kallmann [78] described the realization of nonlinear resistances having a staircase characteristic using diodes connected in opposition and in parallel. M. Yoeli and G. Rosefeld [220] developed a method for synthesizing the ternary combinatorial structures using a catalogue of operators made up from the maximum and minimum functions and the functions J (x). The structures obtained are of the two-layer type.

R.D. Merrill [123] (1966) gave a practical method of two-layer synthesis of a ternary function defined by truth table, and then described the realization by means of ternary threshold logic operators of the corresponding combinatorial structure. The method described is a generalization to the ternary case of the simplifying methods suggested by McClusky. A. Mukhopadyay [141] extended Shannon's work on the symmetric binary functions. After some definitions concerning the symmetric ternary functions were given, a number of theorems were stated and demonstrated. These theorem make it possible to work out methods of identification and synthesis of the ternary switching structures. J. Santos, H. Arango and F. Lorenso [160,162] developed the synthesis of the ternary combinatorial structure by means of threshold operators of the bivalent type. This method of the synthesis was an extension of Quine's method and enables a truth table in a two-layer type of structure to be constructed; the

advanced-carry circuit, which makes it possible to use an algebraic ternary adder, was also discussed.

S. Thelliez [186,188] (1967) described the operation of possible realization by means of transistors and diodes of Lee and Chen's T operator, and constituting with the constant functions ‖0‖, ‖1‖, ‖2‖ as a functional set. An algorithm for the synthesis of ternary combinatorial structure based on the functional decomposition was then given.

C.M. Allen and D.D. Givone [4] (1968) offered an algorithm for the synthesis of 2-layer m-ary combinatorial structure. S. Thelliez [185] introduced the design and the construction with semiconductors of reliable ternary logical operators. He also introduced the incremental computation elements with a view to obtaining specialized calculating structures. M. Yoeli and I. Halpern [219] proposed a ternary arithmetic unit which was based on the ternary symmetric number representation using digit +1, 0, -1. The advantages of this number representation were given in detail, and full adder was developed.

D.I. Porat [148] (1969) presented the developments in algebras and techniques for realization of 3-valued switching functions. Digital arithmetic, ternary codes, composition algebras, minimization, circuit and sequential circuit design were discussed. He then demonstrated the feasibility of 3-valued digital systems.

Z.G. Vranesic, E.S. Lee and K.C. Smith [201] (1970) described a many-valued switching function based on a basic set which was potentially implementable in the economic sense. They developed an algorithmic simplification technique to facilitate synthesis of nontrivial many-valued switching functions. J. Santos and H. Arango [163] described a graphic method using a modified Veitch diagram for the calculation of the weighting vector. The method was based on an algebraic procedure that allows the evaluation of each coordinate solely in terms of the value of the function and independently of the other coordinates.

R.C. Braddock, G. Epstein and H. Yamanaka [21] (1971) presented multi-valued logic design in binary computers which emphasized three transistor, tri-stable circuits. V.C. Hamacher and Z.G. Vranesic [59] compared the cost and speed of 15 digit ternary parallel multiplier and 24 bit binary parallel multiplier. S.C. Lee and E.T. Lee [102] described an algorithm for identifying multi-valued symmetric switching functions using parallel processing and investigated some general properties of these functions. They also defined the mixed multi-valued symmetric and algorithm for identifying. M.E. Liebler and R.P. Roesser [105] introduced a set of functions that took on a prime number of real values which was generalized from conventional Walsh functions. S.Y.H. Su and P.T. Cheung [108] presented a cubical representation for multi-valued switching functions which was very convenient for digital computer processing. E.G.

Zavisca and C.M. Allen [222] presented an approach to multi-valued sequential circuit synthesis.

S.Y.H. Su and A.A. Sarris [183] (1972) presented the relationship between multi-valued switching algebra and Boolean algebra by introducing different definitions for the complements of multi-valued variables. S.Y.H. Su and P.T. Cheung [181] presented a cubical representation for multi-valued switching functions; they also introduced "compound literals" which yielded a realization with less hardware than the existing methoods. W.R. Smith III [175] described some algebraic properties and minimization techniques for multi-valued lattice logics.

G. Frieder, A. Fong and C.Y. Chao [54] (1973) completed the first emulation of a full scale ternary computer. T.A. Irving [74] presented a new family of multi-valued memory devices based on a unique set of multi-valued logic operators. The devices were compared to binary memory elements (flip-flop). They also demonstrated the synthesis of the memory devices from the set of operators. L.J. Janczewski [75] introduced a new method of multi-valued logic function synthesis based on implementating a simple and practical realization operator "two Side Upper Limiter". The method was based on an assumption that every m-valued, n-variable function forms a discrete n-dimension hyper space. R.F. Mattrey, D.D.Givone and C.M. Allen [113] applied multi-valued algebra concepts to neural modeling. C. Moraga and A.

Bittner [134] showed that ternary driving of a stepping motor increased the resolution better than the binary driving with the same motor. A comparative study of parallel and serial error detectors was represented including comments on the tolerances allowed for threshold gates. D.A. Sheppard applied [169] multi-valued logic to fault detection. A.S. Wojcik [211] applied multi-valued logic to asynchronous circuits.

Z.G. Vranesic, K.C. Smith and A.Druzeta [202] (1974) designed R-stable circuits which are the kernel of R-valued storage elements. D.K. Pradhan [150] showed that for N, a power of prime number, we could have a N valued algebra where the defined operations were that of a finite field. The modular algebra which is complete for N, a prime number, was included. Furthermore, it was established that N-valued switching functions for N-power of a prime number could be expressed in a form similar to Reed-Muller expansion for binary functions. J.F. McDonald [120] extended the Weiner-Smith Algorithm for m-ary synchronous sequential circuit design. H.T. Mouftah and I.B. Jordan [137] designed the basic circuits for ternary operators (Inverters, NAND and NOR) with the COS/MOS integrated circuits. U.J. Strasilla [179] discussed a multi-valued memory system using capacitors as storage elements. D.M. Miller and J.C. Muzio [128] described a cellular array that could realize any combinational ternary switching function.

R.S. Ledley and H.K. Huang [93] (1975) extended the principles of Boolean matrix methods to Postian matrix methods and investigated its application in the multi-valued logic design. S.C. Lee and Y. Keren-Zvi [101] showed that any multi-valued logic truth table could be represented by a single (vector) function. They also found a canonical sum-of-products and product-of-sums form of this function. P.T. Cheung and D.M. Purvis [28] described a computer-oriented heuristic algorithm for the minimization of multiple-output multi-valued switching functions. The positional cubical representation for multi-valued switching functions was extended to represent multiple-output functions.

J. Dussault and G. Metze [44] (1976) introduced an m-valued (m = 2) generalized Boolean algebra obtained by extending the set of operators of m-valued Boolean algebras. The additional operators needed for functional completeness were unary operators and were selected such that the basic structure and the simplicity of Boolean algebras were retained. Y.H. Pao and J. Altman [145] introduced the use of associative memory techniques in the implementation of multi-valued systems. R. Spillman [176] examined the problem of detection of single stuck-type faults in multi-valued combinational circuits. H.T. Mouftah, K.C. Smith and Z.G. Vranesic [138] presented the application of COS/MOS integrated circuits in the construction of a three-valued positional control system. D.M. Miller and J.C. Muzio [124] considered the two-place decomposition in multi-valued logic

system. L. Kohout [90] applied multi-valued logic to brain modelling of neuromuscular and to modelling of movement control.

T.T. Dao, E.J. McCluskey, and L.K. Russell [41] (1977) introduced multi-valued integrated injection logic. C. Moraga [135] introduced the spectrum domain to represent the ternary functions which allowed special transformation properties of function to be easily detected. J.P. Deschamps and A. Thayse [42] used a unified theory that yields particular cases in canonical expansions: Newton expansions, the Nyquist expansions, the Kodandapani-Setlur expansions, and the Taylor expansions. J.T. Butler [25] presented the synthesis technique for multi-valued fanout-free networks by extending the partition matrix technique of Ashenhurst and Curtis. T.C. Yang and A.S. Wojcik [215] presented the results of simulating the ternary asynchronous logic networks at the gate level by using a FORTRAN simulator. M. Kameyama and T. Higuchi [82] described the practical state assignment for the multi-valued synchronous sequential circuits. C.L. Lam and Z.G. Vranesic [92] described a method for hashing keys in file addressing applications using the multi-valued nonlinear feedback shift registers which allowed the implementation of the hardware in a very simple way.

D. Etiemble [48] (1978) presented two different versions of voltage mode multi-valued circuits to define a 4-

valued bus: TTL circuits for a 4-valued open collector bus, and TTL circuits for a 4-valued + high impedance bus. E.J. McCluskey [116] introduced logic design of multi-valued with I^2L logic circuits. D. Akins [3] suggested an approach to computer arithmetic for designers of multi-valued logic processor. T. Higuchi and H. Hoshi [65] offered the design features of a special-propose microprogram-controlled ternary computer suited for realizing the real-time digital filters by programming. S.C. Lee and Y.M. Ajabnoor [98] extended the concept of Boolean difference (Boolean derivative) to multi-valued switching algebra, and showed that every multi-valued switching function of these algebras has a Maclaurin series expansion, and the derivation of tests for fault detection of stuck-type faults of multi-valued combinational circuits was presented. T.C. Yang and A.S. Wojcik [216] described the parallel and serial decompositions of multi-valued sequential machines. K.W. Current and D.A. Mow [34] applied the multi-valued threshold logic in large scale integrated circuits for digital signal processing circuits.

C.B. Silio, Jr., J.H. Pugsley and B.A. Jeng [171] (1979) used multi-valued read only memory to reduce the control memory. E.J. McCluskey [117] presented an algebraic method for designing multi-input, multi-valued I^2L circuits. J.L. Huertas and J.M. Carmona [72] presented a new family of ternary C-MOS circuits whose principal advantage lies in the avoidance of resistors for generating the logic levels. J.G.

Tront and D.D. Givone [192] introduced GaAs MESFETs to use as a basic circuit elements in multi-valued circuits. A.D. Singh, J.R. Armstrong and F.G. Gray [172] used universal iterative tree structures to design multi-valued combinational and sequential circuits. M. Kameyama and T. Higuchi [83] described a synthesis of multi-valued logic networks with the minimum number of T-gates by considering the universal logic module implicant and equal residue function in a tree-structured network. W. Wojciechowski and A.S. Wojcik [210] presented an approach to the design of multi-valued combinational logic using an automatic theorem prover as a design tool. R.C. Windecker [207] extended the theory of stochastic combinational networks of two-valued to multi-valued logic. V.H. Tokmen and S.L. Hurst [190] described the ternary switching functions which may be realized by the use of universal logic modules; the specification and use of such modules are based upon the canonic Reed-Muller ternary expansion.

H.G. Kerkhoff and M.L. Tervoert [87] (1980) introduced the charge-coupled device to implement multi-valued logic in large scale integrated circuits. S.C. Lee and M.P. Tull [97] introduced a method for realizing parallel processing machines using multi-valued logic. V. Dahl [38] introduced a three-valued logic for natural language computer application. S.L. Hurst suggested that optical signals may well be eminently suitable for conveying ternary and higher-valued data due to the absence of cross-talk between optical

frequencies and other factors. M.P. Tull [193] presented four-valued gate-level logic simulator which was programmed in FORTRAN and operated in a timesharing environment. H.Y. Lo [107] introduced two methods of deriving fault detection tests in algebraic and map forms of multi-valued combinational logic. S.C. Lee and Y.M. Ajabnoor [99] presented static hazard detection and static hazard elimination after they are detected in multi-level multi-valued combinational circuits. M. Kameyama and T. Higuchi [81] studied radix 4 signed-digit arithmetic circuits for high-speed digital filtering.

M. Karpovsky [85] (1981) surveyed some new theoretical results on spectral methods for functional decomposition, synthesis and testing of multi-valued logical networks with many inputs and many outputs. S.L. Hurst [73] used Harr transformation to synthesize the multi-valued networks. A. Papachristou [146] investigated the implementation of multi-valued logic by content-addressable memory processing. D.C. Rine [155] introduced picture processing using multi-valued logic. W. Coy [31] investigated the description of multi-valued functions by the use of decision diagrams which allowed a straightforward implementation of functions by either MAX- and MIN-gates (and some unary operators) or by a single gate type, the T-gate (or multiplexer). Functional tests for these implementations are easily derived from the diagrams. D.M. Miller [125] examined the spectral methods for the detection of a broad class of function symmetries.

S.C. Lee and Hao-Yung Lo [103] introduced a map partition method for fault detection in an multi-valued logic system which was simpler than the conventional map method. S.C. Lee and W.D. Ballew [100] examined the upward compatibility of binary Boolean algebras with Post algebras.

K.W. Current [32] (1982) presented the active and passive circuits for conversion of quaternary logic signals to analog waveforms by using the standard bipolar integrated circuit technology. M. Kameyama and T. Higuchi [80] proposed a digital processor called multiple-valued array processor. A. Beach and J.R. Armstrong [15] studied a chip layout for multi-valued I^2L. M. Karpovsky [86] considered methods for testing (error detection, correction, and location) in multi-valued computations which are based on systems of linear equality and inequality checks and on analysis of the corresponding syndromes. N. Tomabechi, M. Kameyama and T. Higuchi [191] proposed the residue arithmetic circuit using multi-valued ring counters which provided the capability of counting the multi-valued simultaneously. M. Hu and K.C. Smith [71] proposed a self-checking synchronous sequential machine based on a 2-of-3 valued logic circuit. D.M. Miller [126] presented the application of spectral techniques to fault detection which allowed the use of a weaker condition in the selection of constrained syndrome tests than is possible in the conventional function domain. W.S. Wojciechowski [209] presented a methodology for structured design of digital systems in multi-valued logic. K.Y. Fang

and A.S. Wojcik [53] presented an approach to systematic design of multi-valued logic functions using a decomposition technique that used a small number of modules to implement the functions. M.S. Knudsen [88] presented a nine-valued logic simulation algorithm especially suitable for a fast and accurate analysis of digital NMOS-circuit.

APPENDIX B

MULTI-VALUED CIRCUIT TECHNOLOGY

Recently, the MV circuits may be divided into three classes according to the semiconductor technology [40]: bipolar, MOS/MESFET and CCD devices. The bipolar is the current mode operation device and is not limited to any logic value. The MOS/MESFET is the voltage mode operation device, where active devices are used as voltage switches and signals are voltage level; its practical purposes is confined to the ternary logic only. The CCD is the charge operation mode device and can be used with any logic value.

B.1 MV Bipolar Device Technology

Recently, the bipolar device technology TTL, ECL and I^2L are investigated by several researchers. The I^2L is received the most attention in the MV circuits because its circuit structure is highly suitable for the implementation of "threshold" type of functions [200] which are particularly useful in the MV arithmetic circuits. It also can be fabricated in the higher MV LSI/VLSI circuits package density due to the fact that I^2L contains only active transistor components (no resistors). On the other hand, the pack-

age density of bipolar LSI/VLSI logic such as TTL and ECL are limited primarily by the area occupied by resistors and wiring. Due to practical photoetching limitations, resistor area will not easily be reduced by the same factor as I^2L transistor areas [156]. The other advantages of I^2L over the TTL and ECL are as follows [156]:

(a) Low power consumption; therefore, I^2L gate power dissipation is much lower than the equivalent TTL function (a factor of 10 to 100) and it also has higher IC package density.

(b) I^2L wafer processing is much simpler than TTL and ECL (4-5 masks as compared to 7-9).

However, to this date I^2L is not likely to directly replace high speed logic, such as ECL because the increased processing difficulty necessary for I^2L to achieve ECL performance would largely nullify the yielded advantage. But I^2L has a significant advantage over the MOS technologies in that I^2L can easily be combined with TTL and ECL on the same LSI/VLSI chip to achieve additional speed.

B.2 I^2L MV Circuits

I^2L devices functioning as current amplifiers are suitable for multi-threshold logic circuit implementation and multi-valued logic primitive gates realization. The MV I^2L devices design rules are given by Dao [40] and have been fabricated according to these rules and have proven to be

reliable and reproducible.

There are three fundamental circuit operations in current mode MV I^2L which has been introduced by Dao [41]: Input replication, weighted sums, and threshold detection.

a. Input replication

This operation replicates the input signal by using current-mirror imaging or the folded-collector method. The method used to control the saturation of the n-p-n transistors is based on adding an extra "dummy collector" to the multicollector n-p-n transistor and folding this back to the base. Fig B.1 shows the basic current-mirror circuit operation, it is assumed that the dummy collector area (DA) is smaller than the other collector areas (CA) by factor of x, and assuming the same current density in each collector,

$$i_c = \frac{x\beta}{\beta + x} \cdot i \qquad \text{Eq. B.1}$$

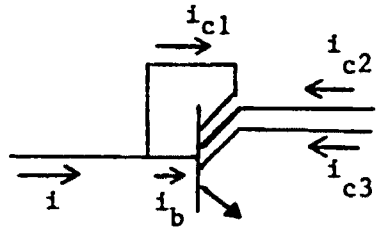


Fig. B.1 I^2L Current-Mirror Circuit.

b. Weighted sums or linear summation

This operation forms the arithmetic sum of several

weighted replicas by varying the size of the output collectors. Collectors of different sizes and from different current mirrors are connected together to form a weighted sum. For example, the weight sum in Fig. B.2 is equal to 3x + y.

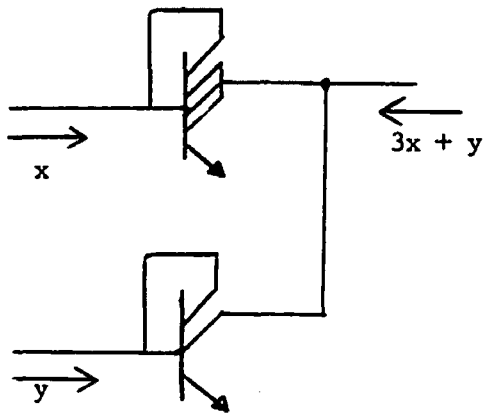


Fig. B.2 I^2L Weight Sum Circuit.

c. Threshold detection

This operation determines if the sum exceeds a determined threshold value by setting thresholds. The attractiveness of I^2L lies in the ease with which constant current sources can be integrated into the gate structure. Fig. B.3 shows this circuit configuration; for input $x < T$ the output y will be 0, for $x > T$ the output will be p.

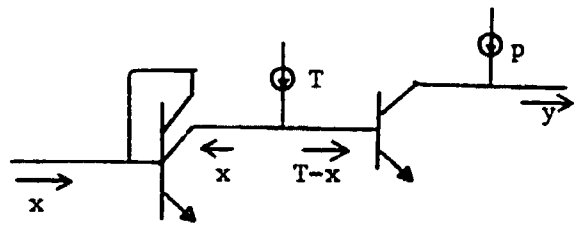


Fig. B.3 I^2L Threshold Detection Circuit.

With these three basic operations of the I^2L circuit, it is possible to generate any MV elements. The following five MV element [41,151] circuit and their relative cost are given. The relative cost is considered from the number of transistors in the circuit. These MV I^2L elements can be implemented in any m-valued.

MAX Gate Element

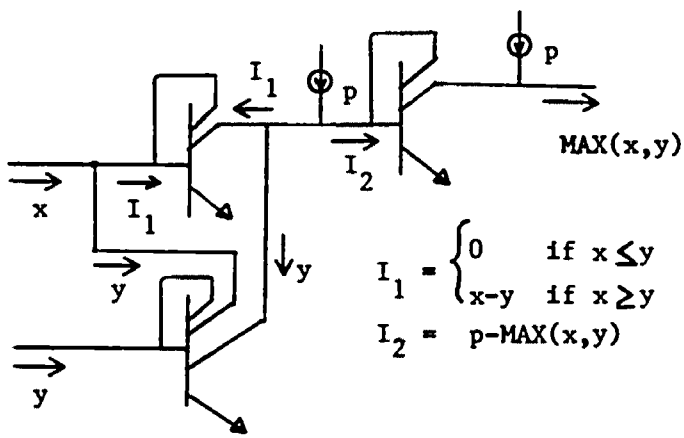


Fig. B.4 I^2L MAX Gate Circuit.

Fig. B.4 shows the circuit of this gate; x and y are m-valued input variables, two constant current sources are (p, p, where p = m-1), the output MAX(x,y) is equal to x if

x > y and is equal to y if y > x. The relative cost (number of transistors) of this gate is 3.

MIN Gate Element

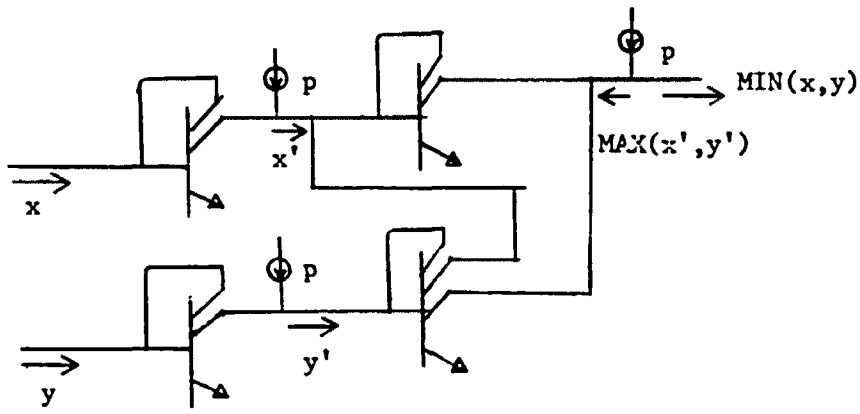


Fig. B.5 I^2L MIN Gate Circuit.

Fig. B.5 shows the circuit of this gate; x and y are m-valued input variables, three constant current sources are (p, p, p), the output MIN(x,y) is equal to x if x < y and is equal to y if y < x. The relative cost of this gate is 4.

COMPLEMENT Gate Element

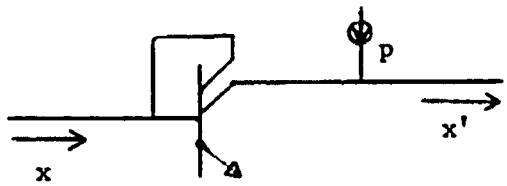


Fig. B.6 I^2L COMPLEMENT Gate Circuit.

Fig. B.6 shows the circuit of this gate; x is a m-valued input variable, one constant current source is (p),

the output $\bar{x} = p - x$. The relative cost of this gate is 1.

LITERAL Gate Element

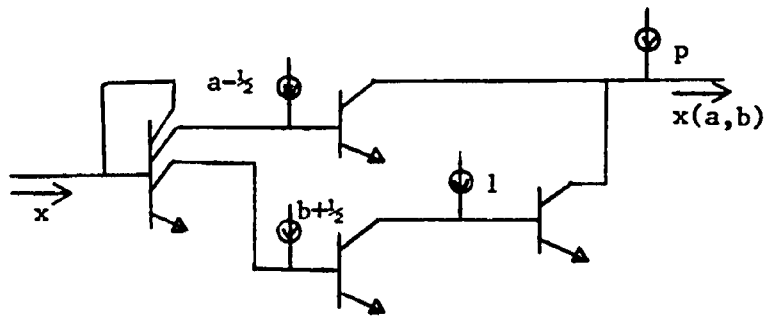


Fig. B.7 I^2L LITERAL Gate Circuit.

Fig. B.7 shows the circuit of this gate; x is an m-valued input variable, the four constant current sources are (a-1/2, b+1/2, 1, p), the output x(a,b) is the values of x between a and b when a < b. The relative cost of this gate is 4.

CYCLE Gate Element

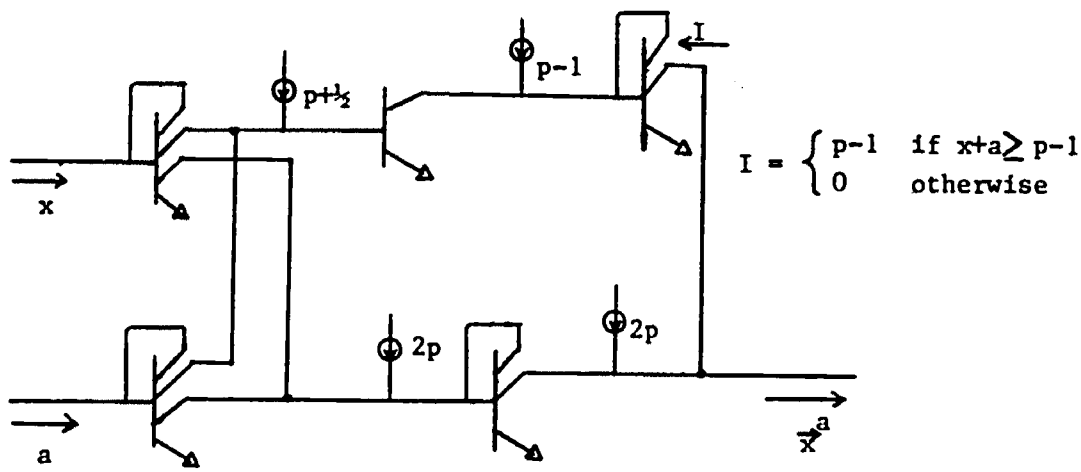


Fig. B.8 I^2L CYCLE Gate Circuit.

Fig. B.8 shows the circuit of this gate; x is an m-valued input variable, a is a m-valued constant, four constant current sources are (p+1/2, 2p, p-1, 2p), the output x = (x plus a) mod m. The relative cost of this gate is 5.

Several I^2L MV circuits have been designed with these operators, such as multiplexer, universal quad logic gate, quaternary adder, quaternary full product, quaternary ROM, quaternary quantizer, quaternary D latch, Quaternary D flip-flop, master-slave RS m-flop, quaternary-coded decimal counter, ternary to binary decoder, etc. The layout of some of these circuits has been developed by Beach and Armstrong [15]. Table B.1 illustrates the truth table for the quaternary full adder and Fig. B.9 shows the quaternary full adder (QFA) circuit implemented by the I^2L technology.

Table B.1 Quaternary Full Adder Truth Table.

	SUM				CARRY				
y \ x	0	1	2	3	0	1	2	3	
0	0	1	2	3	0	0	0	0	
1	1	2	3	0	0	0	0	1	
2	2	3	0	1	0	0	1	1	C = 0
3	3	0	1	2	0	1	1	1	
0	1	2	3	0	0	0	0	1	
1	2	3	0	1	0	0	1	1	
2	3	0	1	2	0	1	1	1	C = 1
3	0	1	2	3	1	1	1	1	

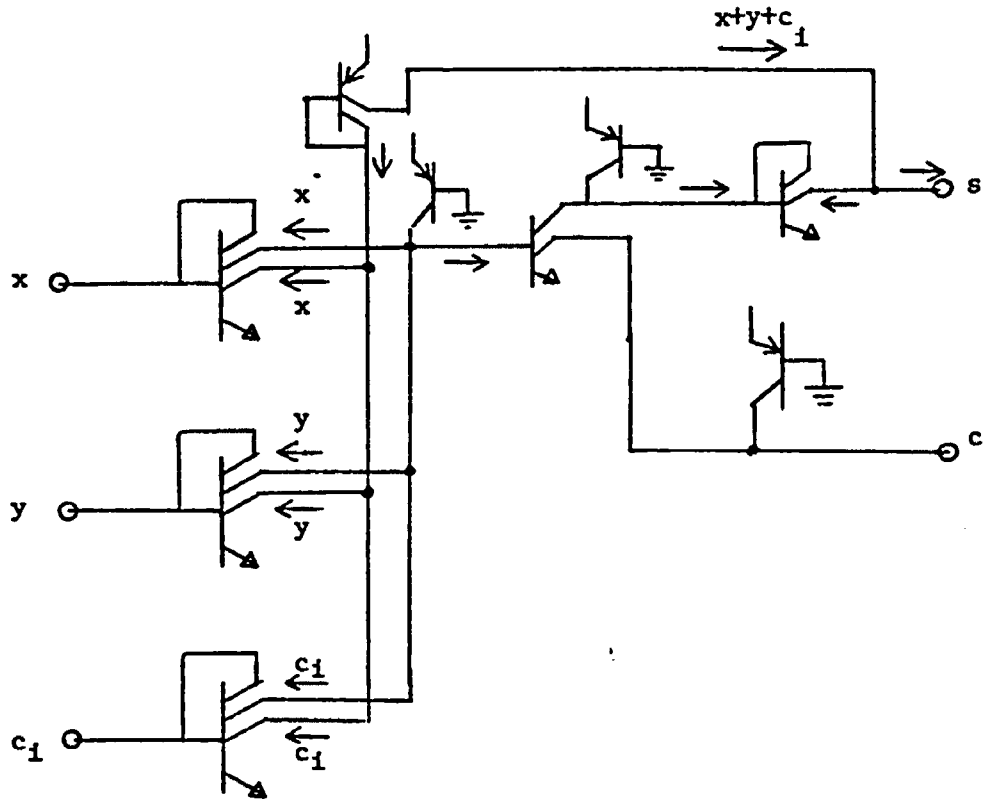
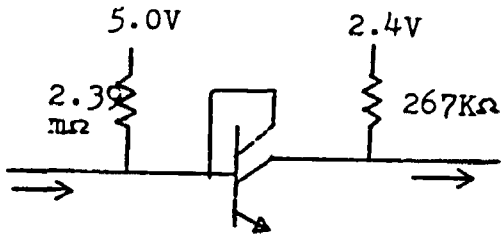
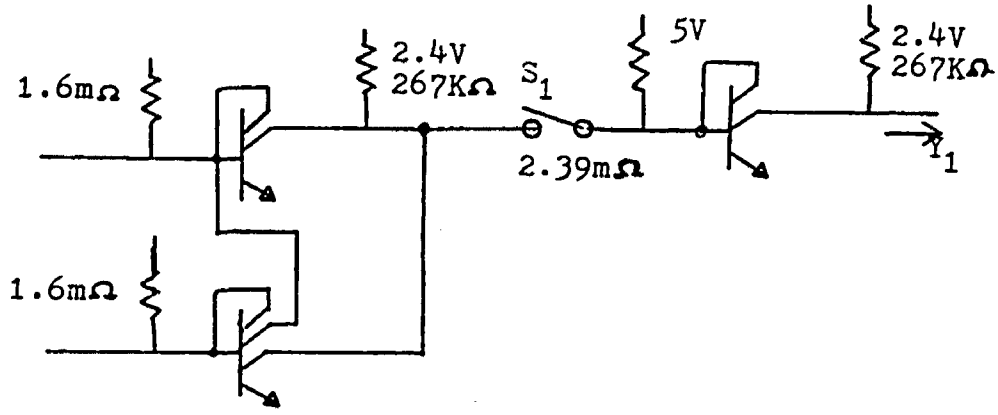


Fig. B.9 I^2L Circuit Implemented in QFA.

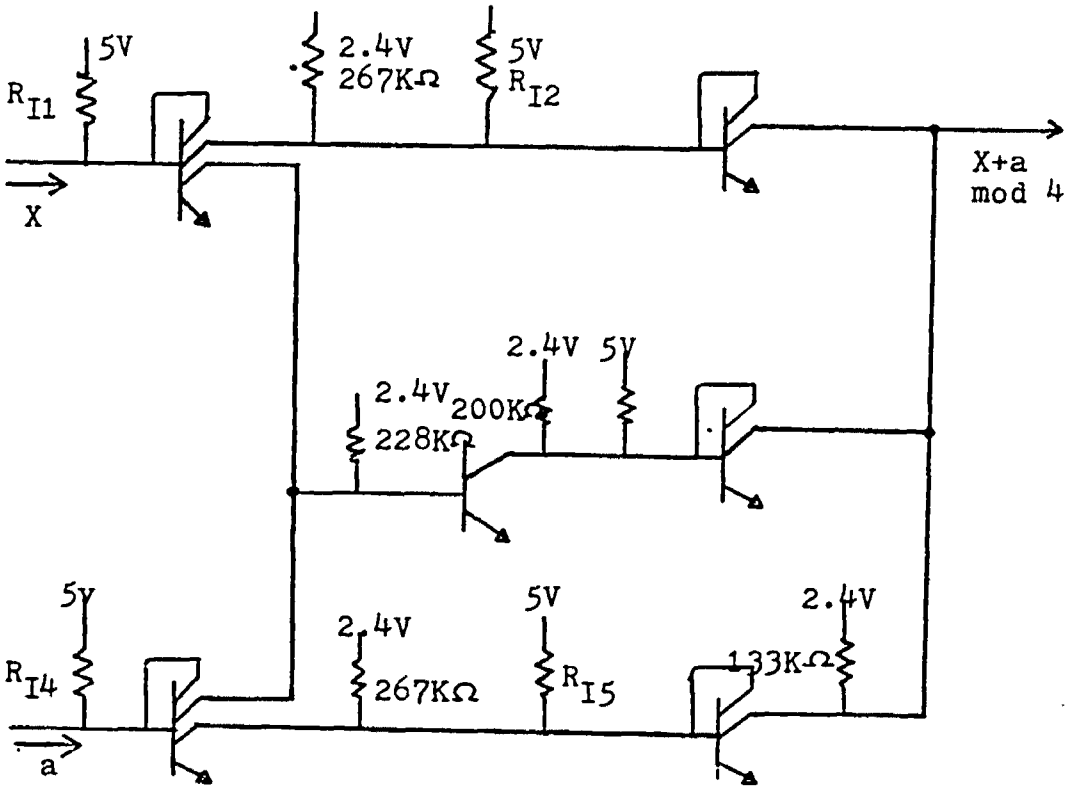
The actual construction of each of the operator circuits designed above has been done by Ballew [11]. This actual construction has been done with integrated circuits offered by EXAR integrated systems in their I^2L custom IC design kit [106]. Figures B.10 (a), (b), and (c) illustrate COMP gate, MAX gate, and CYCLE gate respectively.



(a) COMP gate construction



(b) MAX gate construction



(c) CYCLE gate construction

Fig. B.10 The Actual I^2L Circuit Gate Construction.

B.3 TTL MV Circuits

The TTL MV circuits have been studied by Etiemble and Israel [48,49,50,51,52]. Their method is based on Birk and Farmer's method [18] which implemented ternary circuits with the current technologies of the TTL binary integrated circuits. A general scheme to realize the function f(x,y) is given in Fig. B.11. The ternary to binary conversion is realized with threshold detectors (decoder circuit) and the binary to ternary conversion is realized by encoder the circuit.

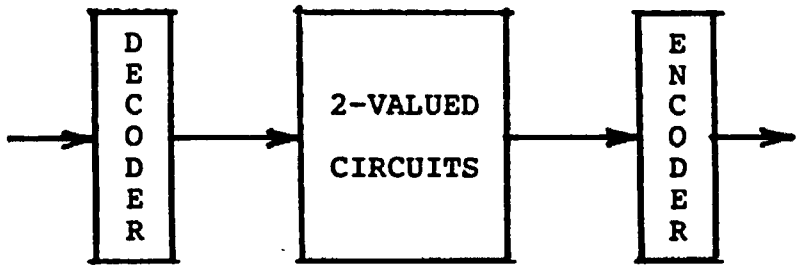


Fig. B.11 Implementation of Function f(x,y).

B.4 ECL MV Circuits

The ECL MV circuits are used to implement a multi-threshold element which was originally proposed by Druzeta [43]. The quaternary threshold logic full adder (QFA) [33,34], four-valued threshold logic digital correlator [36] and four-valued threshold logic counter [35] have been reported by Current and Mow. The ECL is the fastest standard logic family due to its nonsaturating operation [34]. The ECL threshold is not purely a current mode technique; input signals are voltage levels which are converted into current levels inside the circuits by differential pairs, currents are then linearly combined and results are converted back into voltage levels before any threshold detections which are performed by differential voltage comparators. Conventionally, fixed voltage references are created for the thresholds which raises the problem of sensitivity of the detectors with bias voltage and fabrication [40].

Several ECL MV circuits have been simulated and/or breadboarded with discrete devices. One of these is the

quaternary threshold logic full adder circuit [33,34]. The QFA is implemented by either using the current steering properties of ECL based, current-mode threshold logic circuits or the current steering capabilities of I^2L. Logical variables are developed as integer multiples of an easily duplicated reference amount of current. These currents are then summed and differentiated to produce the desired logical results. The QFA accepts two four-valued input currents A and B and a binary carry input current C, and produces a two-quaternary-digit, four-valued output word CS that represents the sum of the inputs, where C is the most significant digit. The transfer characteristics for this function are shown in Fig. B.12. The circuit realization to be described makes use of a set of switching thresholds illustrated in Fig. B.12(a). Figures B.12(b) and (c) show the SUM and CARRY outputs. Fig. B.13 shows ECL-based QFA [34].

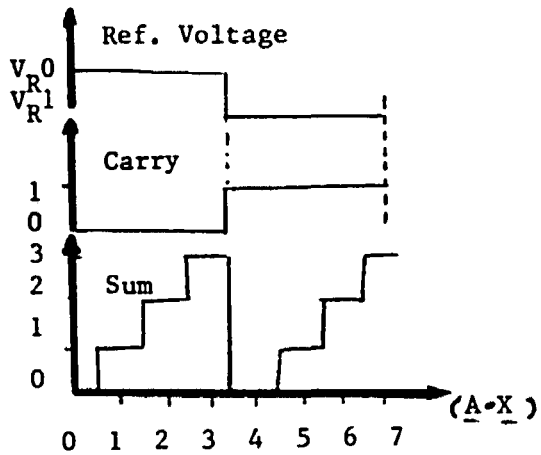


Fig. B.12 QFA Transfer Characteristic.
(a) Switching Thresholds
(b) CARRY Output
(c) SUM Output.

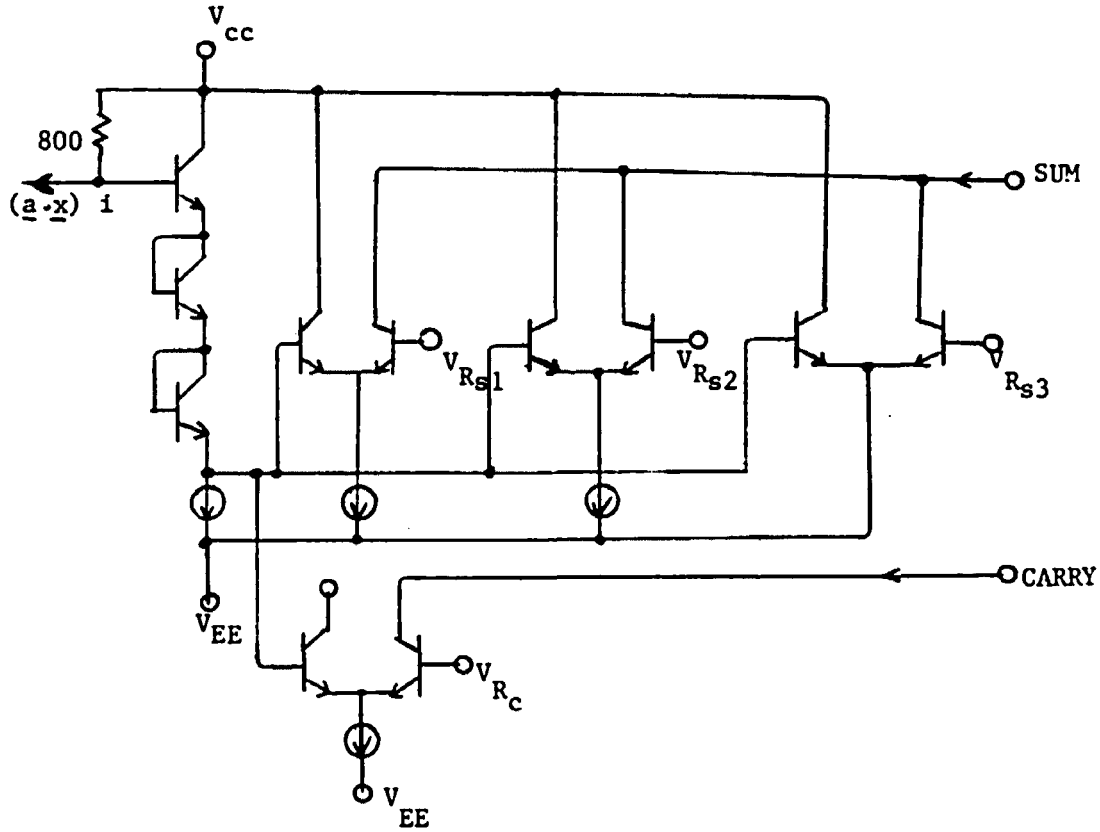


Fig. B.13 Quaternary Threshold Logic Full Adder Circuit.

B.5 MOS/MESFET MV Devices Technology

The MOS/MESFET devices technologies received particular interest in the MV LSI/VLSI due to the considerations of functional density, power consumption, high speed (MESFET only) and interfacing with the existing binary circuits. These devices are operated with voltage mode; therefore the circuit use voltage levels to represent the MV logic values.

a. MOS MV Circuits

Recently, two types of MOS circuit has been considered for use in the MV LSI/VLSI: CMOS (COSMOS) and NMOS. The CMOS MV circuits were first proposed by Mouftah and Jordan [137] who designed the basic ternary operators with the Complementary Symmetry Metal Oxide Semiconductors (COSMOS). There are two basic types of ternary operators that they considered: unary operators and binary operators. The unary operators consist of simple ternary inverter (STI), positive ternary inverter (PTI), negative ternary inverter (NTI), forward diode (FD) and reverse diode (RD). Their truth table and circuits are illustrated by Table B.2 and Fig. B.14 respectively.

Table B.2 Truth Table of Unary Basic Ternary Operators.

x	STI(x)	PTI(x)	NTI(x)	FD(x)	RD(x)
+	-	-	-	+	0
0	0	+	-	0	0
-	+	+	+	0	-

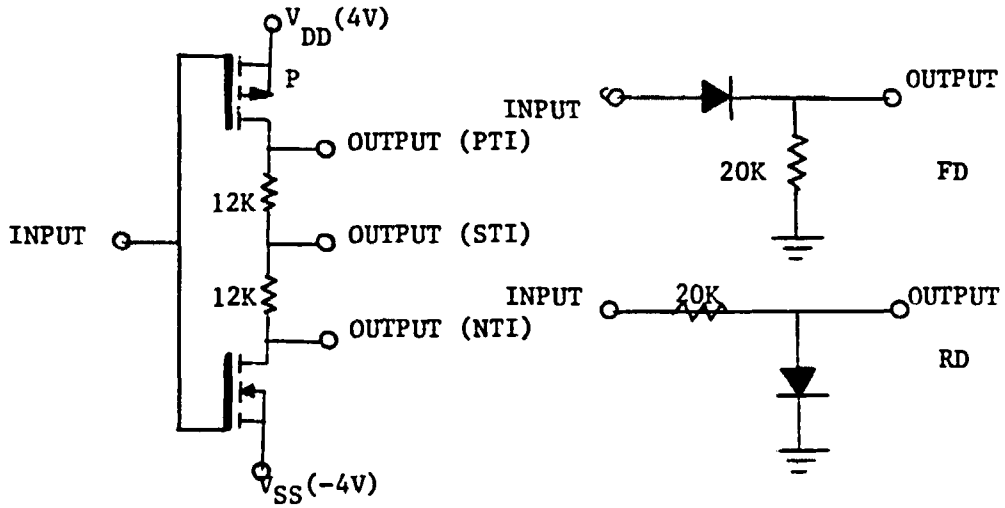


Fig. B.14 Unary Basic Ternary Operator COSMOS Circuits.

The binary operators consist of ternary NOR and ternary NAND and their truth table and circuits are illustrated by Table B.3 and Fig. B.15 respectively.

Table B.3 Truth Table of Binary Basic Ternary Operators.

x	y	TOR	TAND	TNOR	TNAND
+	+	+	+	-	-
+	0	+	0	-	0
+	-	+	-	-	+
0	+	+	0	-	0
0	0	0	0	0	0
0	-	0	-	0	+
-	+	+	-	-	+
-	0	0	-	0	+
-	-	-	-	+	+

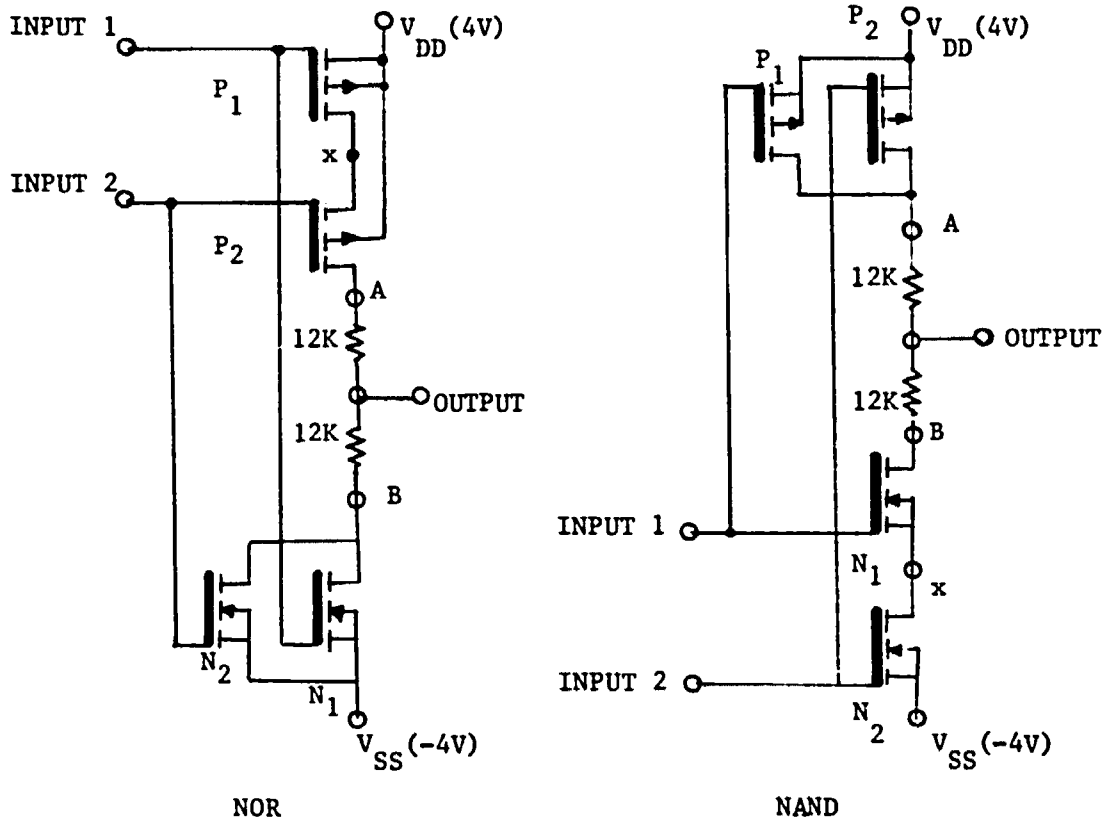


Fig. B.15 Binary Basic Ternary Operator COSMOS Circuits.

With these basic COSMOS ternary operators, several fundamental ternary circuits, such as CYCLE gate, $J_k(x)$ arithmetic circuits , T-gate or multiplexer and ternary flip-flop or memory (D-flip flop, T-flip flop) have been constructed. However, the Mouftah and Jordan design approaches [137] were mainly restricted to ternary logic exclusively. Huertas and Carmona [72] presented the C-MOS which may be extended to the four-valued logic system.

Recently, the development of CMOS and NMOS ternary logic circuits are based on the design technique depicted in Fig. B.16 [29], where a circuit is composed of three stages:

(1) The first stage consists of input decoders which convert input ternary signals into binary signals representing literal two-valued functions of single ternary variables.

(2) The second stage consists of literal logic which combines literals by means of binary logic circuits.

(3) The third stage consists of an output encoder which converts binary signals from the second stage into the correct ternary output.

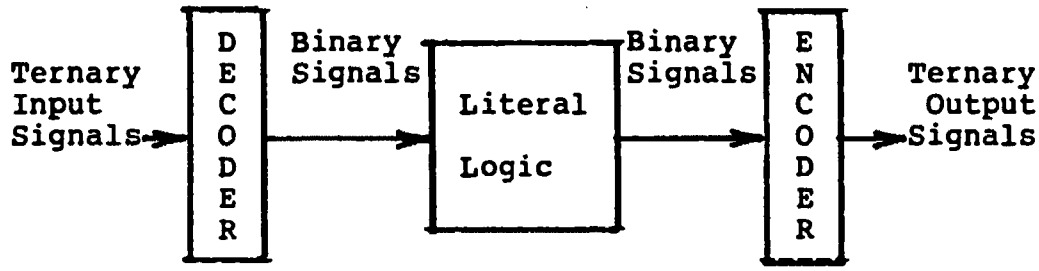


Fig. B.16 A Block Diagram for a Ternary Circuit Based on Binary Logic.

The NMOS ternary was introduced by Russell [158]. Suppose a standard NMOS depletion load technology with a single supply voltage V is considered. The built-in threshold V_T of an enhancement switch device provides an obvious way to use the standard inverter as a voltage detector of level V_T as shown in Fig. B.17(a). In order to implement the ternary logic, a second level detector with threshold set at midway between V_T and V_{CC} has to be devised. Russell proposed a totem pole structure of the switch device of the

inverter, whereby the source of normal switch is connected to another enhancement device with a shorter gate to source (V_{GS}=0) and the threshold voltage V_t. Fig. B.17(b) shows the total circuit which operates as a source follower as soon as the input level exceeds the compound threshold ($V_T + V_t$). The result of this is that V_T = 1/4 V_{CC} and $V_T + V_t$ = 3/4 V_{CC} .

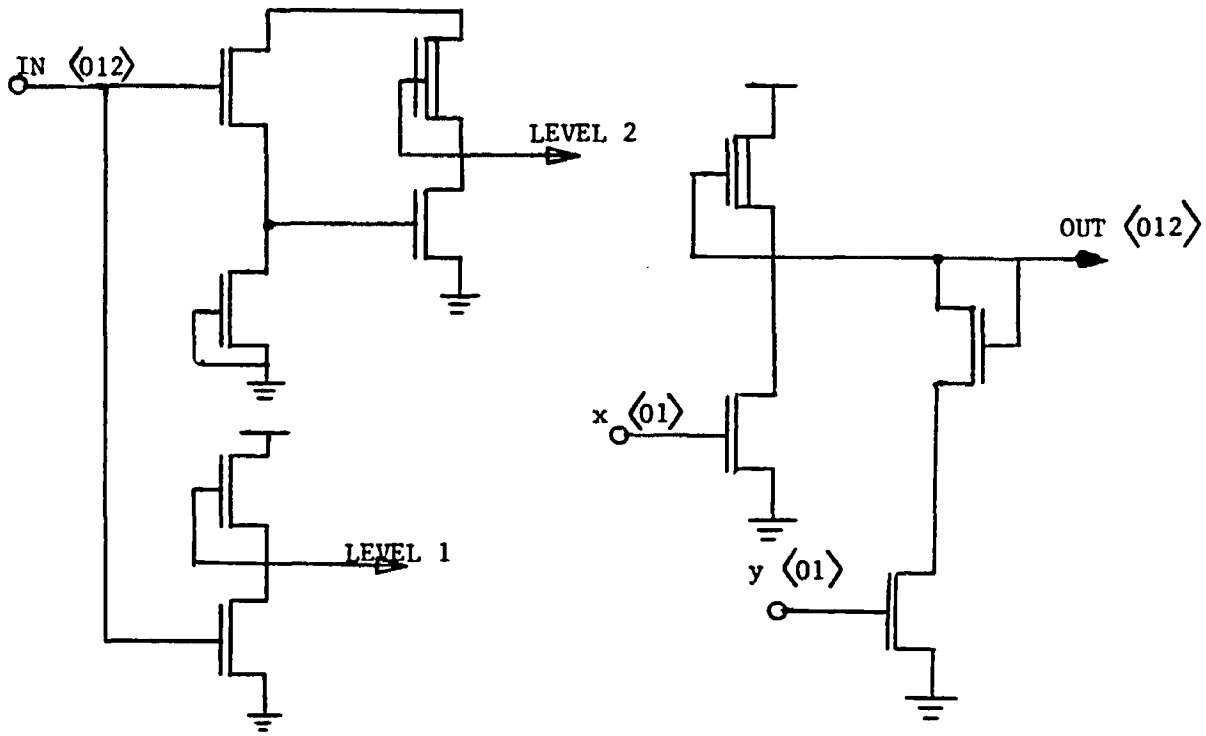


Fig. B.17 (a) Two Level Detector
(b) Output circuit.

McCluskey [115] presented a method for designing ternary NMOS logic circuits which is based on a circuit family invented by Russell [158]. The circuits are fabricated by using standard silicon gate enhancement/depletion technology which is suitable for MV LSI/VLSI implementation. The fabri-

cated circuits operate with three signal voltages: <0, 2.5, 5> volts equivalent to the logic symbol <0, 1, 2> respectively. Table B.4 shows the maps for a modulo 3 full adder and the NMOS circuit implementation of this table is shown in Fig. B.18.

Table B.4 A Modulo 3 Full Adder Truth Table.

SUM

CIN = 0

y \ x	0	1	2
0	0	1	2
1	1	2	0
2	2	0	1

CIN = 1

y \ x	0	1	2
0	1	2	0
1	2	0	1
2	0	1	2

y \ x	0	1	2
0	1	0	0
1	0	0	1
2	0	1	0

y \ x	0	1	2
0	0	0	1
1	0	1	0
2	1	0	0

$A = \overline{CIN} \cdot P + CIN \cdot Q$

y \ x	0	1	2
0	d	1	0
1	1	0	d
2	0	d	1

y \ x	0	1	2
0	1	0	d
1	0	d	1
2	d	1	0

$B = \overline{CIN} \cdot \overline{Q} + CIN \cdot P$

y \ x	0	1	2
0	d	0	d
1	0	d	d
2	d	d	0

C = 0

y \ x	0	1	2
0	0	d	d
1	d	d	0
2	d	0	d

C = 0

$$P = x_0^0 \cdot y_2^0 + x_1^2 \cdot y_1^1 + x_2^1 \cdot y_0^2$$
$$Q = x^0 \cdot y^1 + x^1 \cdot y^2 + x^2 \cdot y^0$$

CARRY

CIN = 0

y \ x	0	1	2
0	0	0	0
1	0	0	1
2	0	1	1

CIN = 1

y \ x	0	1	2
0	0	0	1
1	0	1	1
2	1	1	1

y \ x	0	1	2
0	1	1	1
1	1	1	0
2	1	0	0

y \ x	0	1	2
0	1	1	0
1	1	0	0
2	0	0	0

$A = \overline{CIN} \cdot T$ + $CIN \cdot T \cdot \overline{Q}$

y \ x	0	1	2
0	d	d	d
1	d	d	1
2	d	1	1

B = 1

y \ x	0	1	2
0	d	d	1
1	d	1	1
2	1	1	1

B =1

y \ x	0	1	2
0	d	d	d
1	d	d	0
2	d	0	0

C = 0

y \ x	0	1	2
0	d	d	0
1	d	0	0
2	0	0	0

C = 0

$$T = x^0 + y^0 + x^1 \cdot y^1$$

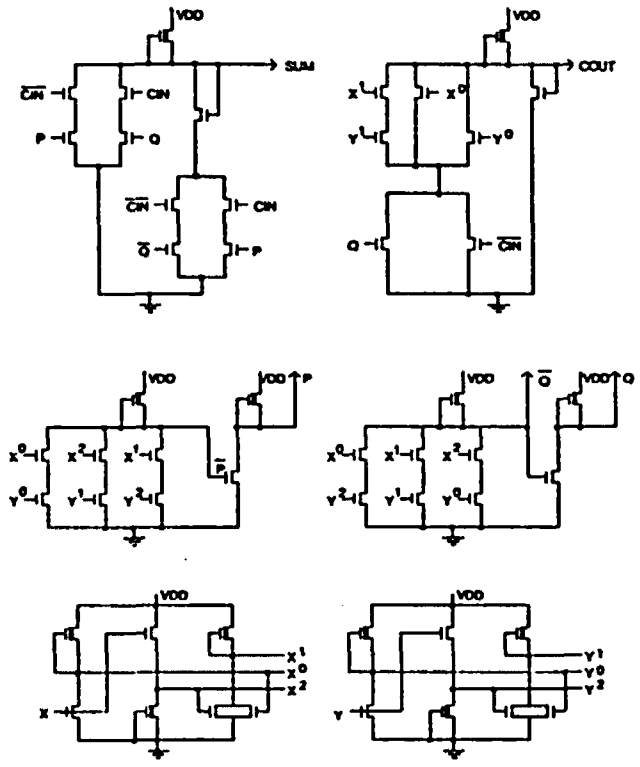


Fig. B.18 NMOS Circuits for a Modulo 3 Full Adder.

b. MESFET MV Circuits

MV GaAs MESFET device was introduced by Tront [192] in 1979. The GaAs appears superior to silicon for high frequency and high-speed devices because of its higher electron mobility and energy-band gap. The FETs are used to take advantage of the higher electron mobility. The GaAs MESFETs are majority carrier devices and are basically junction field-effect transistors, and they exhibit no charge-storage effects. They have no insulating metal oxide layer, and

their relatively simple geometry may permit easier fabrication of narrow channel lengths. The GaAs MESFET may be used when speed is the primary requirement, and silicon can still be used for low-speed less costly circuits.

Fig. B.19 shows the basic structure of a GaAs MESFET device; the device has three terminals, source, gate, and drain. In order for the GaAs MESFET device to be useful in the MV logic, the pinch off voltage Vp of the transistor must be variable. The pinch off voltage is given as

$$Vp = \frac{-qNd}{2\epsilon} \cdot a^2$$

where Nd is the channel doping concentration, a is the channel thickness, q is the electron charge, and ϵ is the semiconductor permittivity. The detail of physical operation of the MV GaAs MESFET can be found in Tyal and Liechti's work [194,195]. The MV GaAs MESFET 5-valued $\overline{MAX}$, $\overline{MIN}$, COMP, LIT and their relative cost (number of MESFET in the circuit) are shown in Figs. B.20, B.21, B.22, B.23 respectively.

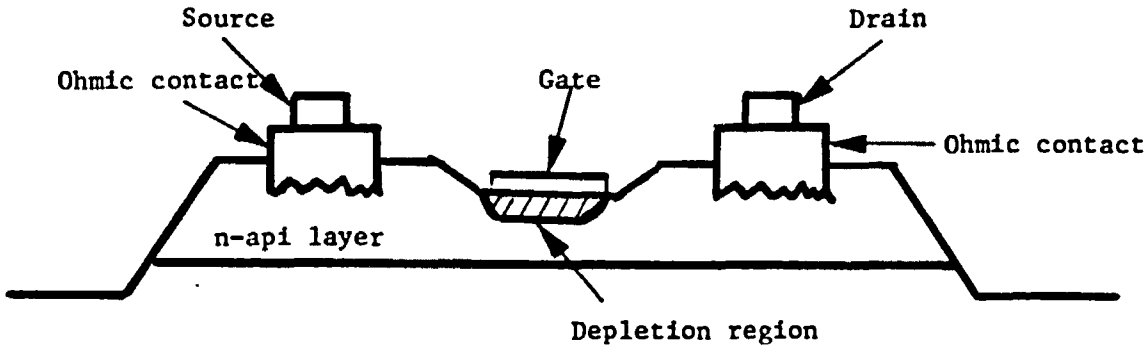


Fig. B.19 A Basic Structure of a GaAS MESFET.

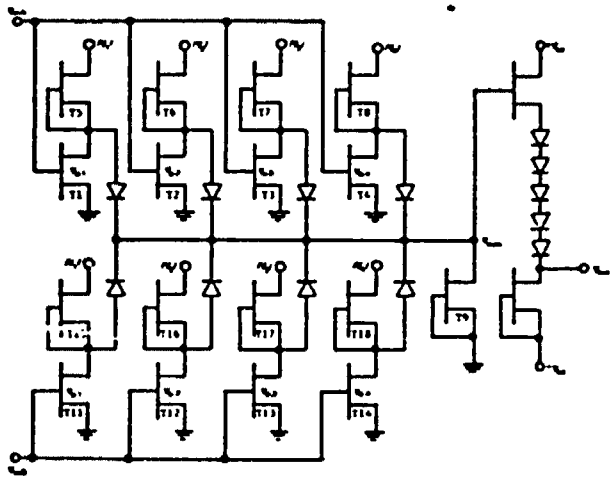


Fig. B.20 A GaAs MESFET 5-Valued $\overline{\text{MAX}}$ Gate.

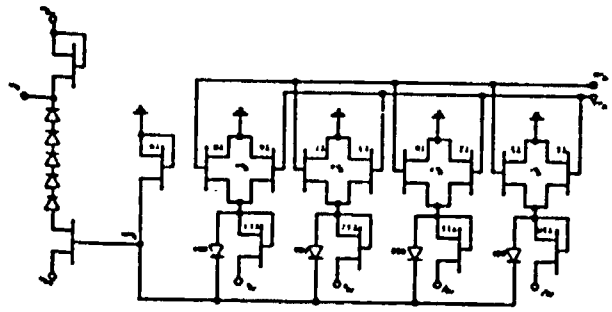


Fig. B.21 A GaAs MESFET 5-Valued $\overline{\text{MIN}}$ Gate.

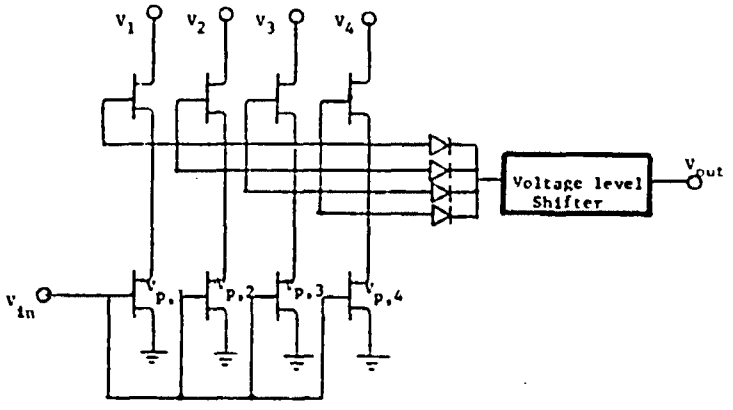


Fig. B.22 A GaAs MESFET 5-Valued COMP Gate.

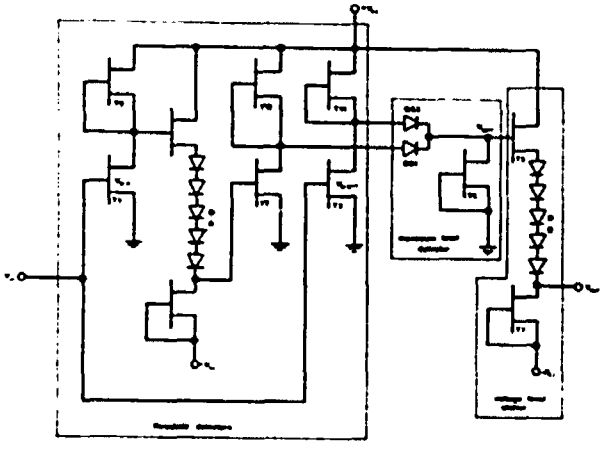


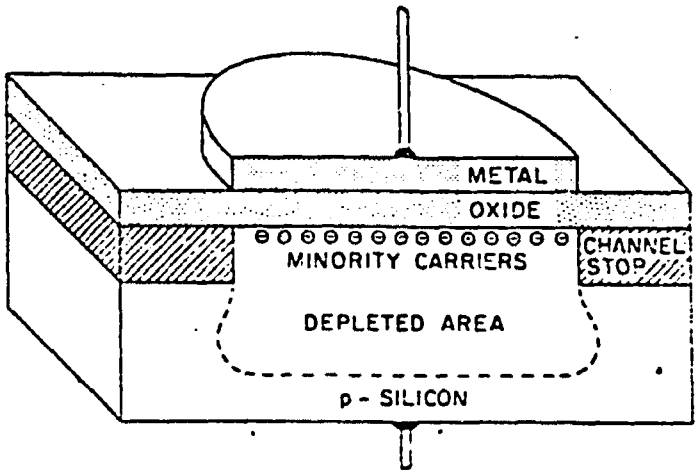
Fig. B.23 A GaAS MESFET 5-Valued LIT Gate.

C. Charge Couple Device MV Technology

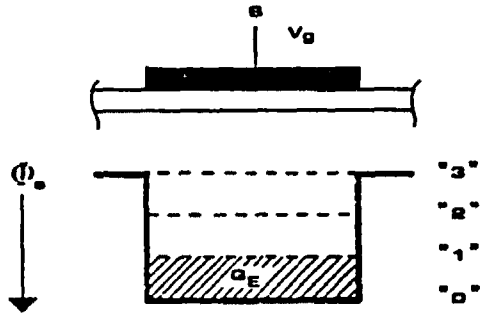
CCD was introduced for MV logic by Kerkhoff and Dijkstra [87]. This device is very well suited for the MV LSI/VLSI logic design because of its high packing density, low power consumption , and ease of fabrication. It has been predicted [27] that a four-valued CCD memory will have a

higher information density; and therefore, a lower cost per bit when compared with a binary RAM having the same minimum geometry. This fundamental idea has been put into practice by workers of Mitsubishi, who have doubled the storage capability of a binary 65 k bit CCD memory by packing two bits of memory into each storage site [214].

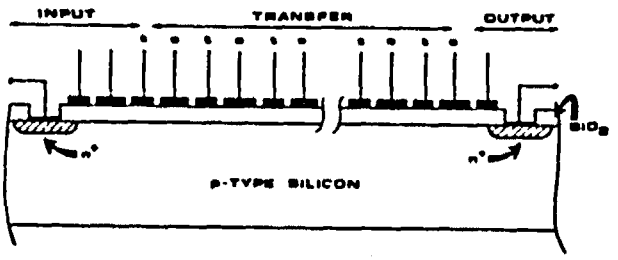
The basis of the CCD is the Metal-Oxide-Silicon (MOS) capacitor. Fig. B.24(a) [167] shows an isolated MOS capacitor formed by a metal electrode deposited on a thermally oxidized p-type silicon substrate . Fig. B.24(b) shows a 4-valued potential well (bucket) [87] which can store a certain amount of electron charge, where s is storage gate, Vg is a voltage apply to the storage gate, Φ_S is the interface potential, and Q_E is the unity charge packet. Fig. B.24(c) shows the basic structure of a charge couple device which consists of an input section, a transfer section and an output section, where t is the transfer gates. The detail of the physical operation of the CCD is given by Sequin and Hobson [167,70]. Fig. B.25 [87] shows the symbols using in the CCD MV.



(a) An Isolated MOS capacitor



(b) A Four-valued potential well



(c) A CCD structure

Fig. B.24 The Basic Structure of CCD.

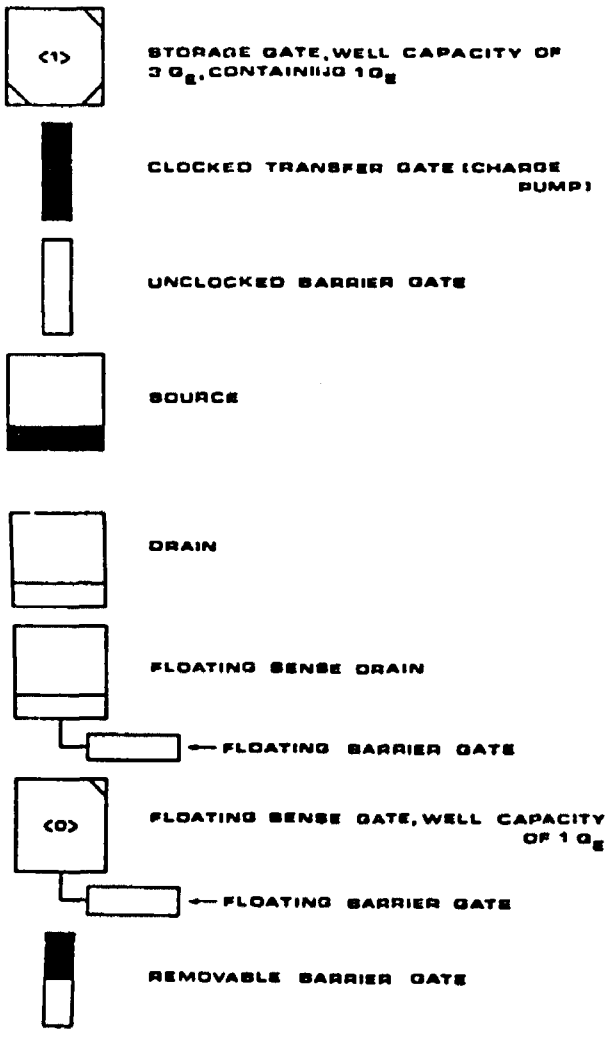


Fig. B.25 The CCD Symbolics.

There are three basic operations to design the CCD MV circuits which will be discussed as follows:

a. Charge addition

This operation can add the charge packets in different storage wells and transfer them into a common storage well. The example is illustrated in Fig. B.26.

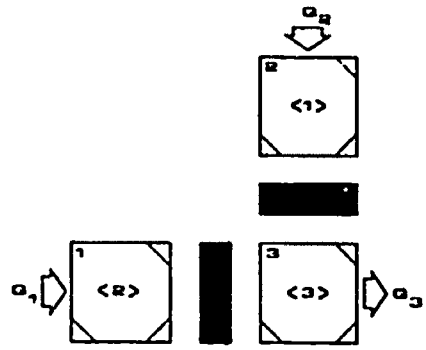


Fig. B.26 The CCD Charge Addition.

b. Charge subtraction

This operation uses the charge overflow principle. The maximum charge handling capacity of a well is proportional to its gate area and the voltage differential between the gate and its adjacent one. If the source provides a charge in excess of the capacity of the sink, then the excess charge is transferred to the subsequent sink given a proper bias of the sinks and the barrier gates. This operation is shown in Fig. B.27.

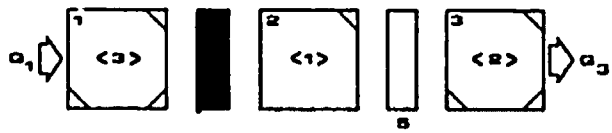


Fig. B.27 The CCD Charge Overflow Principle

B.6 Charge follower/complement

A very important structure in the CCD MV logic is the floating gate shown in Fig. B.28. It is able to detect a charge pocket non-destructively under the floating sense gate 1, and control the transfer of another charge pocket by means of the floating barrier gate. An amount of charge under the floating sense gate in this structure is converted into a certain voltage on the barrier gate. This barrier gate control propriety has been judiciously put into use by Kerkhoff et al. [87] to perform charge threshold detection, charge complementation, charge regeneration, and charge redistribution (or well extension).

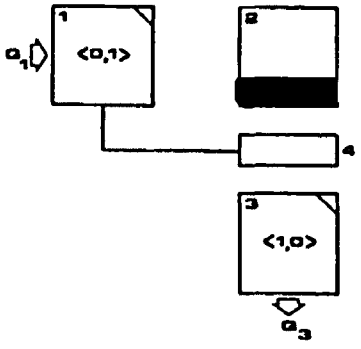


Fig. B.28 CCD Charge Control with Floating Gates.

Several CCD MV circuits have been designed [87], such as literal gate, four-valued full adder, quaternary-to-binary-converter, binary-to-quaternary-converter, successor, complement, etc. An example of the four-valued full adder is shown in Fig. B.29. The four-valued full adder performs

the addition of two four-valued inputs X and Y and a two-valued carry input Cin, resulting in four-valued sum output S and two-valued carry output Cout. In the behavior of the full adder circuit, two situations can be distinguished:

1. The total input charge (Cin + X + Y) does not exceed three unity charge packets. In this case all the input charges are transferred to the sum output.

2. The total input charge exceeds three unity charge packets. The above mentioned transfer path is now blocked. The sum and the required carry output signal are now obtained by the charge transfer via a parallel path.

The CCD four-valued full adder static operation has been simulated by Kerkhoff and is shown in Fig. B.30. All the data for calculations were derived from their design lay-out.

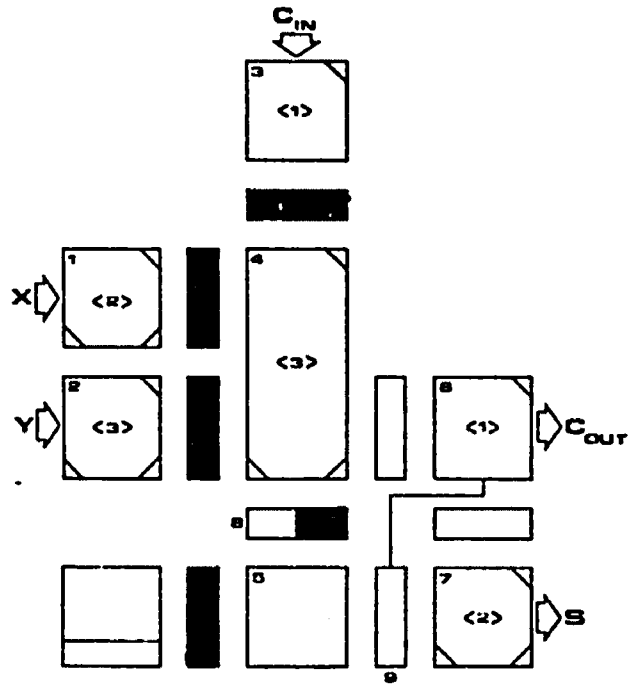


Fig. B.29 The Schematic Diagram of Four-Valued Full Adder.

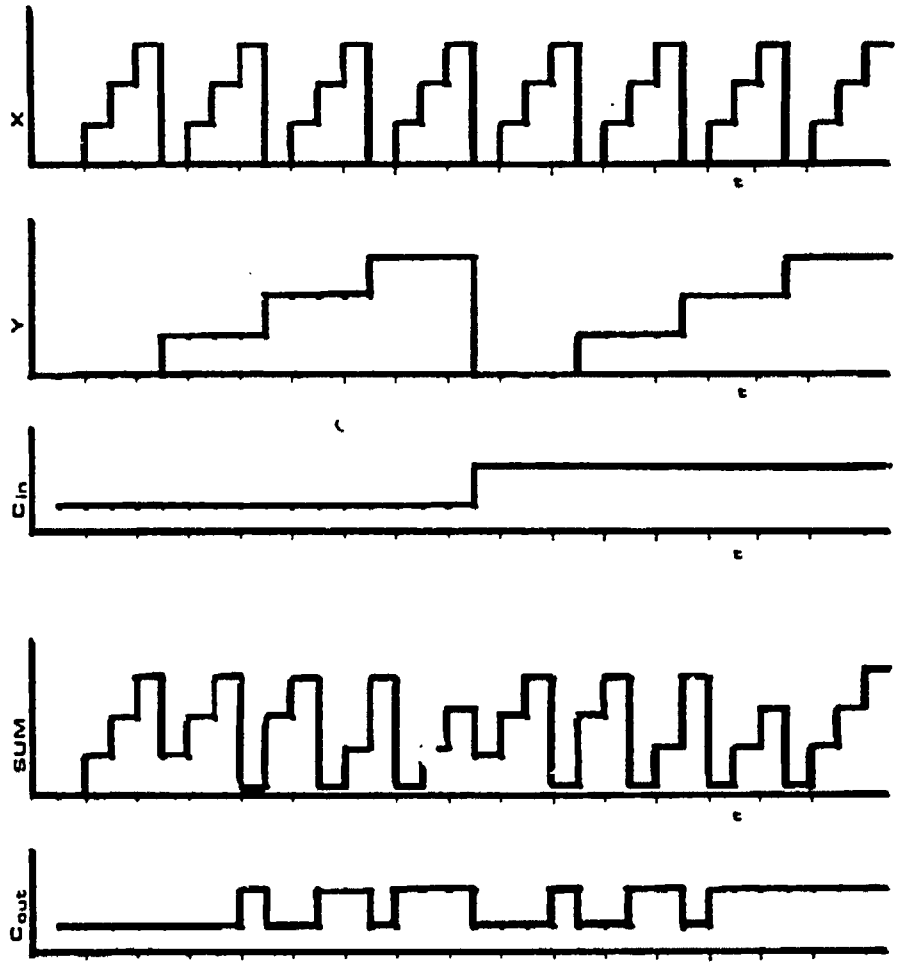


Fig. B.30 The Static Simulation of the Operation of the Four-Valued Full Adder.