A Modular Design Approach to Large-Scale-Integration of High-Speed Digital Systems

Mohamed I.Y. Elmasry

This Disseration is Submitted to the
Department of Electrical Engineering
University of Ottawa
in Partial Fulfillment of
the Requirements for the Degree of
Doctor of Philosophy

Department of Electrical Engineering
Faculty of Science and Engineering
University of Ottawa
Ottawa, Ontario
January 1974

A MODULAR DESIGN APPROACH TO LARGE-SCALE-INTEGRATION OF HIGH-SPEED DIGITAL SYSTEMS

ABSTRACT

There is an increasing demand for speed in Large-Scale-Integrated digital data processing systems to meet the requirements of real-time processors, high throughout computers and wideband digital channels. This high-speed requirement can be met by properly matching system and logic designs to the technology used for hardware realization. This thesis demonstrates that this match is possible by adopting a modular approach in circuit, logic and system design. A new circuit structure is developed based on current-mode operation of bipolar transistors. The structure realizes several levels of logic in a single circuit and special logic design techniques are given. A single structure can realize a storage element with less power dissipation and silicon area than conventional ECL circuits. Measurements indicate a significant improvement in effective power-delay products over existing logic families and 500 MHz operation of a master-slave D-type flip-flop has been obtained with a power dissipation of only 16 mW.

A MODULAR DESIGN APPROACH TO
LARGE-SCALE-INTEGRATION OF HIGH-SPEED DIGITAL SYSTEM

INDEX TERMS

High-speed Logic, Large-Scale Integration, Modular Design, Universal Logic Modules, Cellular Arrays, Parallel Organization, Pipeline Organization, Logic-in-Memory, Bipolar Technology, Emitter-Current-Logic, Emitter-Function-Logic, Multi-Emitter Two-Level Logic, Iterative Multipliers, Fast Carry Adders

TABLE OF CONTENTS

INDEX

LIST OF FIGURES

LIST OF SYMBOLS

Symbol		Page
TTL	Transistor-Transistor-Logic	46
STTL	Schottky-Transistor-Transistor-Logic	46
ECL	Emitter-Coupled-Logic	47
C	Clock input	65
Ind.	Indeterminate state	66
$\emptyset$	Don't care condition	68
MECL	Motorola ECL	91
LSTTL	Low-power Schottky TTL	91
LTTL	Low-power TTL	92
HTTL	High-threshold TTL	92
$\leqslant$	Less or equal to	103
$\neq$	Not equal to	104
Σ	The summation of	105
$\cap$	Intersection of sets	106
$\cup$	Union of sets	107
α, β	Matrices	107
γ, ζ	Matrices	108
P	Prime implicant	109
T	Maximum delay time for a carry	130
C_1, C_2	Two-phase clocks	139
Maj.	Majority of	144
x, y	Binary variables	145

LIST OF TABLES

ACKNOWLEDGMENT

The author wishes to express his thanks to his supervisors Professor P.M. Thompson and Dr. C.L. Sheng for their guidance in carrying out this research work. Special thanks to Professor Thompson for his constructive suggestions in the preparation of this thesis. He would also like to thank Mr. Z.E. Skokan and Mr. C. Shinn of Hewlett-Packard Company, Palo Alto, California for fabricating test samples, Mr. P. Okulich of Consolidated Computer Inc. for building the logic trainer, Mr. M. King and Dr. M. Caughey of Bell-Northern Research, Ottawa, Dr. S.R. Das and Dr. D.K. Banerjee of the University of Ottawa for their interest and encouragement. He wishes to thank Mr. T. Curtis of Bell-Northern Research, Ottawa for building the hybrid models, the staff of the Electronics Department at the Communication Research Center, Ottawa and the staff of the Electron Devices Laboratory of Bell-Northern Research, Ottawa for use of their facilities. He also wishes to thank his colleagues at the University of Ottawa; Mr. M. Brule, Mr. N. Agrawal, and Mr. N. Kabra for the useful discussions. The author wishes to thank his wife, Elizabeth, for her encouragement. Finally, the graduate fellowship awarded to the author by the National Research Council of Canada is gratefully acknowledged.

Dr. C.L. Sheng is now President of the National Chiao Tung University, Hsinchu, Taiwan.

Dr. S.R. Das is now with the Institute of Radio Physics and Electronics, Calcutta, India.

CHAPTER 1

INTRODUCTION

A MODULAR DESIGN APPROACH TO LARGE-SCALE-INTEGRATION OF HIGH-SPEED DIGITAL SYSTEMS

The speed of processing digital information in computer systems not only affects the performance of a system but also the cost of computation. In real time applications, where the time taken for processing a given amount of information should be minimized, high-speed processing is essential. Also, in digital communication systems where the cost per channel is important, high speed processing allows the number of channels served by a single piece of hardware to be increased and this results in lower cost. For wide band-width channels, the data rate is such that high-speed processing is necessary.

This high-speed requirement can be met using integrated technology. Recent developments in technology offer the possibility of economical realization, either in bipolar or MOS. MOS integrated technology is subdivided into single channel (P or N channel) and complementary-MOS (CMOS). MOS devices can be operated either in a static or dynamic mode and can be processed using silicon-gate, metal-gate or advanced dielectrically isolated MOS processes. Although MOS devices have the potential of operating at high-speed, their speed of operation is still technology limited. Alternatively, conventional diffused bipolar technology offers devices which can operate with subnanosecond delays. A recently developed dielectrically isolated bipolar process, Isoplanner II [1], reduces the power dissipation of the circuit elements and the silicon area taken by them.

It has been demonstrated [2, 3] that the use of Large-Scale-Integration (LSI) results in a reduction of cost and an increase in reliability over Small-Scale-Integration (SSI) systems. However, the currently available high-speed logic families (Transistor-Transistor-Logic (TTL), Schottky-Transistor-Transistor-Logic (STTL), Emitter-

Coupled-Logic (ECL)) are not suitable for LSI because of their high power dissipation and large silicon area per gate. The maximum functional complexity achieved is in the Medium-Scale-Integration (MSI) range and is usually equivalent to 100 or fewer gates per chip.

This thesis explores the possibility of the Large-Scale-Integration of high-speed digital logic systems. A criterion which is considered important for LSI realization is modularity i.e. the use of one or a few types of building blocks in a system. The modular realization of logic functions has been studied and researchers in the field have reported many interesting results with two main approaches namely universal logic and cellular logic. However, these studies were not related to the limitations and requirements of LSI technology and in some cases modularity was achieved at the expense of complex interconnection patterns or great silicon area. This would result in expensive designs which may be impractical. Alternatively, a modular approach is required which meets the limitations and the requirements of LSI technology and offers hardware realization with subnanosecond delays. Such an approach should be used with a system organization suitable for high-speed processing, e.g. parallel, pipeline, and logic-in-memory.

Fig. 1.1 shows the interface relationship between technology, circuit, logic design and computer architecture. It also shows how this research work is related to the state-of-the-art. It is possible to obtain an optimum design only if all aspects of design are studied. In the course of this research work the fabrication technology chosen was conventional diffused bipolar technology. The circuit operation of the basic structure was chosen to be current-mode operation. This resulted in a development of a Multi-Emitter Two-Level logic (METLL) structure by the author. The structure was characterized, integrated and compared with other families of logic. Then logic design techniques were developed to use the structure effectively and economically. Finally, its use in the realization

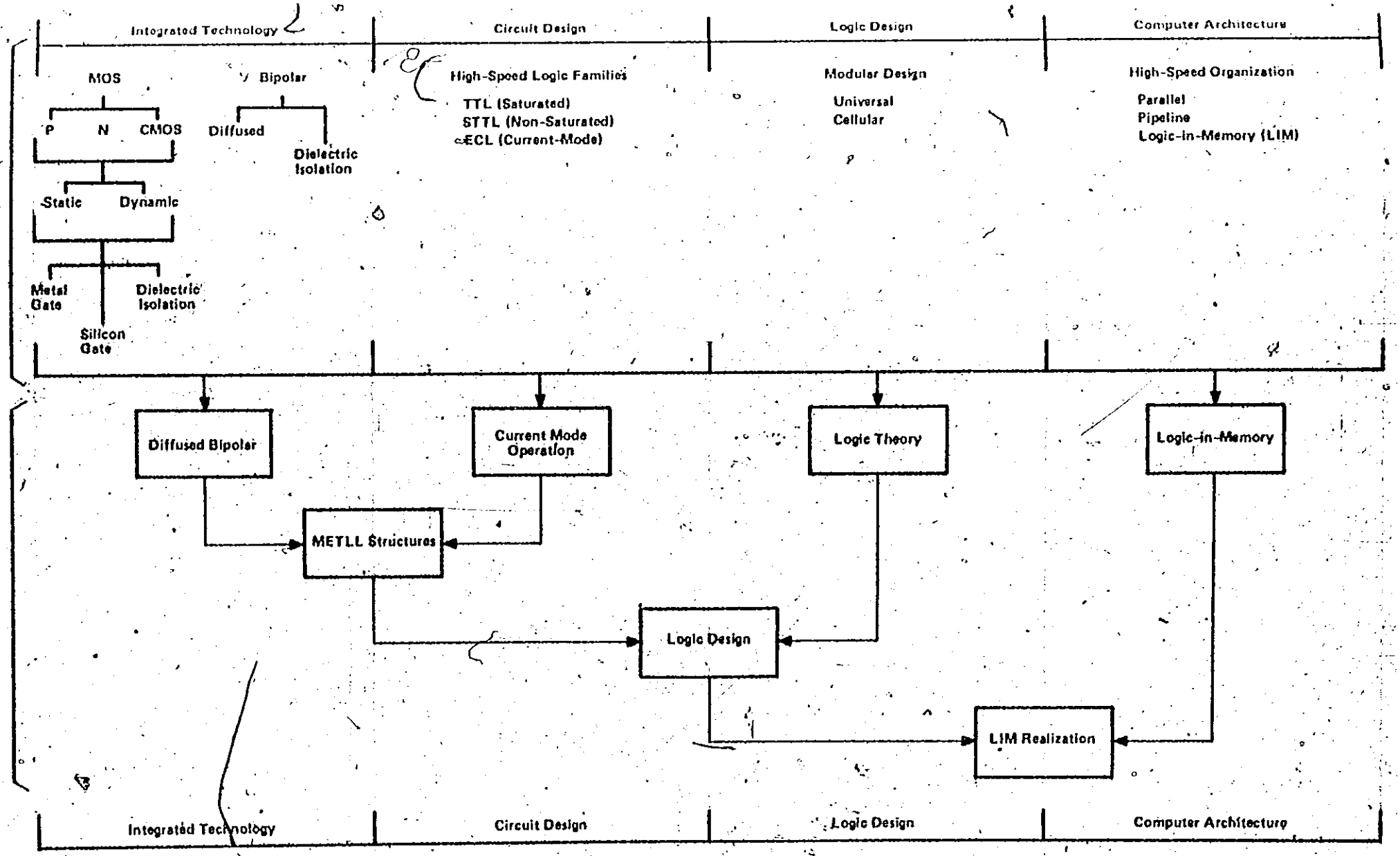


Fig. 1.1 Different Design Levels of LSI High-Speed Digital Systems

of a high-processing rate organization, namely logic-in-memory (LIM) arrays was demonstrated.

This thesis consists of 7 chapters. Chapter 2 contains a brief survey of some approaches for high-speed computer organization. These are parallel, pipeline and logic-in-memory. In this chapter, we indicate the relationship between LSI and computer system organization and conclude that hardware realization of a digital machine greatly affects its system architecture.

Chapter 3 contains a detailed review of two modular logic design approaches; universal and cellular. The modularity of the approaches is demonstrated and examples of building blocks used are given. The purpose of this chapter is to indicate the shortcomings of these approaches regarding practical LSI realization. It concludes that a more practical approach to modularity could lead to an optimum overall design. This approach is referred to as 'function realization'.

Chapter 4 contains a model for logic systems and a discussion of high-speed circuit considerations. It also contains the development of a new circuit structure, referred to a METLL. The development of the structure was originated at the University of Ottawa by Professor P. M. Thompson and the author and was first reported in October 1971 [4]. The integrated form was realized at Hewlett-Pachard Company, Palo Alto, California by Skokan and Shinn [6]. Part of the material of this chapter has been published [7-14].

Chapter 5 contains logic design techniques which are used in realizating logic functions. These techniques were developed from the general techniques of Curtis [2] by the author. Examples are given to demonstrate their usefulness. The example of the full-adder was suggested by a member of the digital circuit research group at the University of Ottawa, Mr. P. Okulich.

In chapter-6, some advantages of Logic-in-Memory (LIM) arrays are indicated. The application of METLL structures in LIM arrays is presented and it is largely the author's original work [10].

CHAPTER 2

COMPUTER ARCHITECTURE FOR HIGH-SPEED PROCESSING

COMPUTER ARCHITECTURE FOR HIGH-SPEED PROCESSING

In computers, the speed of processing information is a function of both the system organization and the hardware realization [1]. The system organization can increase the speed effectively by adopting one, or a combination of three main approaches.

(1) The first approach is to subdivide the data to be processed among many identically constructed subsystems. This is commonly referred to as parallelism [2-10].
(2) The second is to overlap the operations of the different parts of the system, and is commonly called pipelining [6-12].
(3) The third approach is to handle blocks of information in similar cells which can perform both logic and memory functions, and this is commonly called logic-in-memory [13-16].

In this chapter a brief survey of the above three approaches is presented to demonstrate the relationship between LSI and computer organizations for high-speed processing.

2.1 THE PARALLEL ORGANIZATION

Figure 2.1 shows a block diagram of first and second generation computers [1]. The memory is central and it

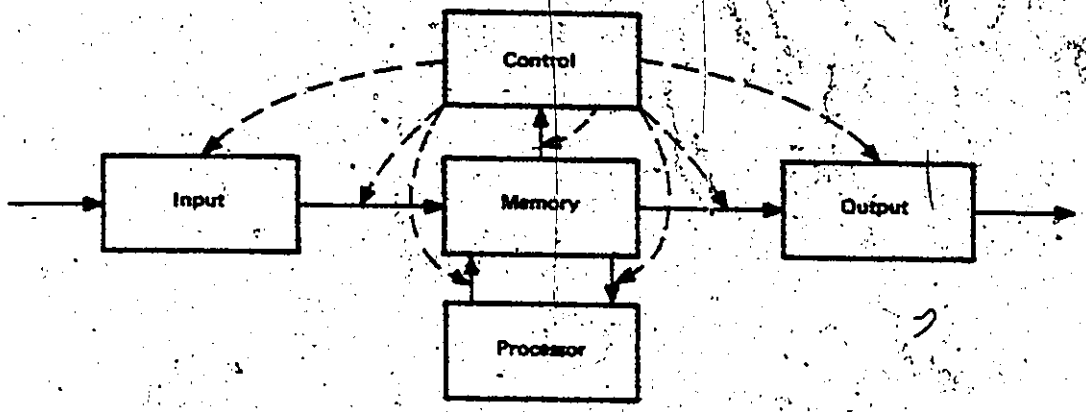


Fig. 2.1 First and Second Generation Computer System

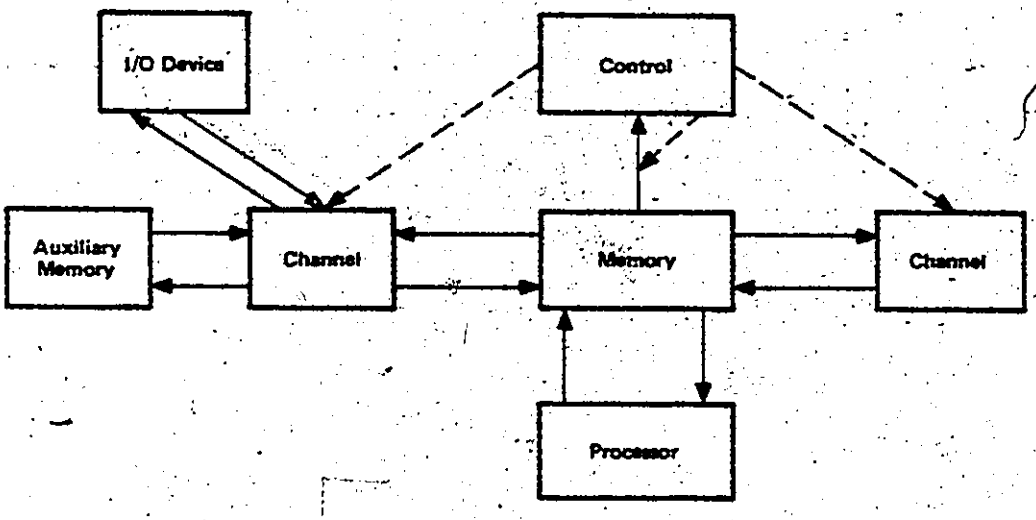


Fig. 2.2 Third Generation Computer System

receives information from input devices and furnishes information to output devices. The control subsystem obtains the program from memory and assigns tasks, one at a time, to the other subsystems. The processor receives data from the memory under the direction of the central subsystem and it returns processed data to the memory. Figure 2.2 shows a block diagram of a third generation computer. The main difference is that the devices, whether input or output, do not report directly to the memory but are reached through a channel controller.

For the execution of a particular instruction, the following sequence of operations is undertaken:

(1) The control requests and fetches from the memory the next instruction to be executed.

(2) The control decodes the instruction and either

(a) An operand is fetched from the memory, stored in a register of the processor and control is given to the latter in order to perform an operation; or

(b) An operand is stored, from a register of the processor, in the memory; or

(c) A request is made to the input to accept a word from the outside world; or

(d) A request is made to the output to transmit a word to the outside world.

Some or all of the above operations can be performed in parallel by similar units. Since the control unit may be sophisticated and expensive, a considerable amount of money is saved by having only a single control unit in charge of W identical processing units. Figure 2.3 shows the parallel processor sequence of activities in W processing elements. T_p is the time taken for a specific computation step to be

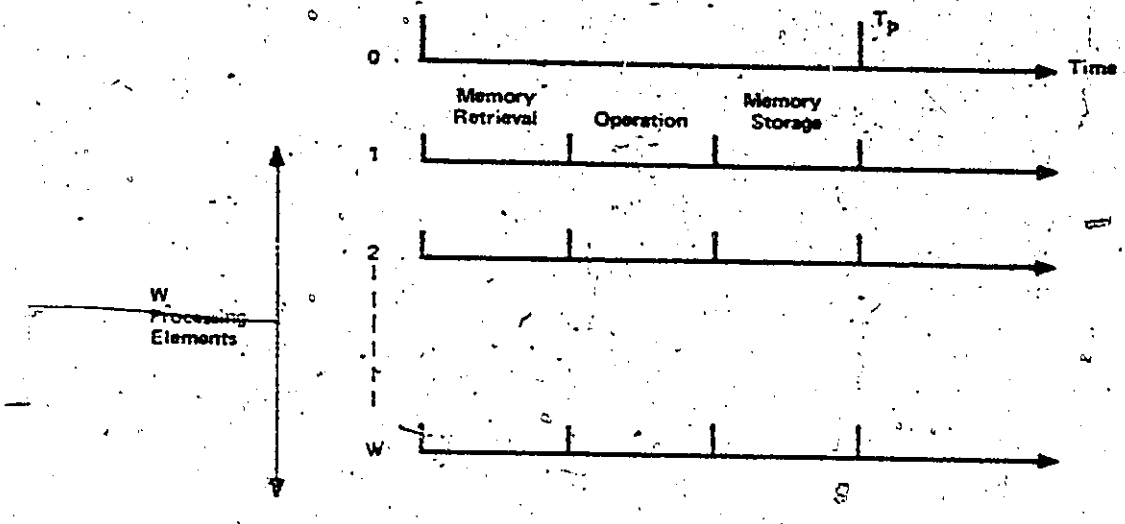


Fig. 2.3 Parallel Processor-Sequence of Activities

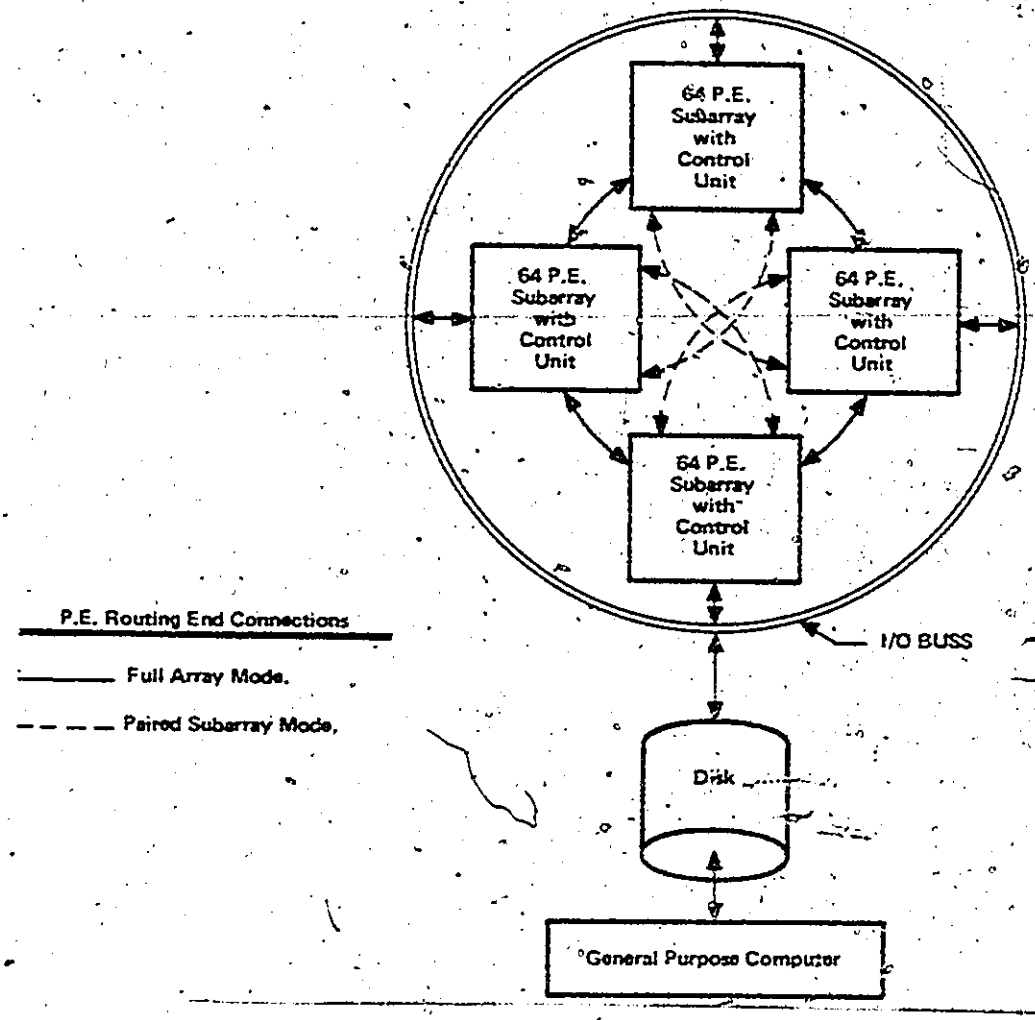


Fig. 2.4 ILLIACIV General Organiztion

performed. N computation steps are suitable for parallel computation and form a parallel stage if:

(1) No step depends upon the result of any other in the stage.
(2) All steps require the same operation to be performed.
(3) The operands are properly distributed among the arithmetic unit memories.

For a stage of N steps, the time required for execution by a parallel machine is

$$t_{NP} = T_p \left[\frac{(N-1+W)}{W} \right]$$

where $[\ ]$ indicates the integer part of.

A typical example of a parallel system is the ILLIAC IV which has been developed by the University of Illinois and Burroughs Corps. Figure 2.4 shows the general arrangement of the system which is organized in four subarrays, each containing 64 processing elements (PE) under the direct control of a subarray control unit. Data is transferred between the memory units of the processing elements of the subarrays and a large scale disk file buffer memory via a highly parallel input/output buss. Such input/output transfers are controlled by an external general purpose computer which also supervises the ILLIAC IV program runs. The general purpose computer is provided with limited set of connections to the subarray control units for this purpose. Each PE is provided with three 64-bit arithmetic registers and high speed adders for full 64-bit operation. The processor is also provided with 2K 64-bit words of memory.

The batch processing of LSI makes parallel computers economically attractive [1] ; LSI memory prices are approaching the price per bit of magnetic memories and the cost/bit increases almost linearly from a small memory to a large memory. This means that a small memory for each PE is economically feasible.

Moreover the reduced physical size and the higher packing density offered by LSI technology has eliminated many practical problems associated with the implementation of parallel machines. Higher reliability of LSI components also reflects itself on the total cost of the parallel machine [4].

2.2 THE PIPELINE ORGANIZATION

The pipeline organization, shown schematically in Fig. 2.5, gains its speed advantage by starting the retrieval of a second set of operands, each located in memory adjacent to the first, before the first result has been returned to the memory. The arithmetic unit is also built as a pipeline. It can receive and start working on a second set of operands before finishing the calculation for the first set. To complete the memory-to-memory pipeline, the arithmetic unit must deliver the results back into memory at the same rate that it receives new pairs of operands from the memory.

Figure 2.6 shows the sequence of activities for a pipeline processor where τ is the cycle time and T_{PL} is the time required for a complete operation. The time required for the pipeline computer to execute a stage of N steps is

$$t_{NPL} = T_{PL} + \tau (N-1)$$

Generally, N computing steps are suitable for pipeline computation and form a pipeline stage if

(1) No step depends upon the result of any other in the stage.
(2) All steps require the same operation to be performed.
(3) The members of each operand string are packed in successive memory locations.

A typical example of a pipeline computer is the STAR, which is under development at the Control Data Corp. The

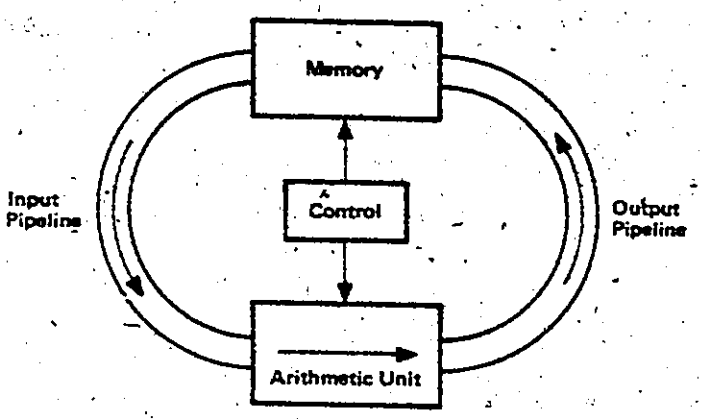


Fig. 2.5 The Pipeline Organization

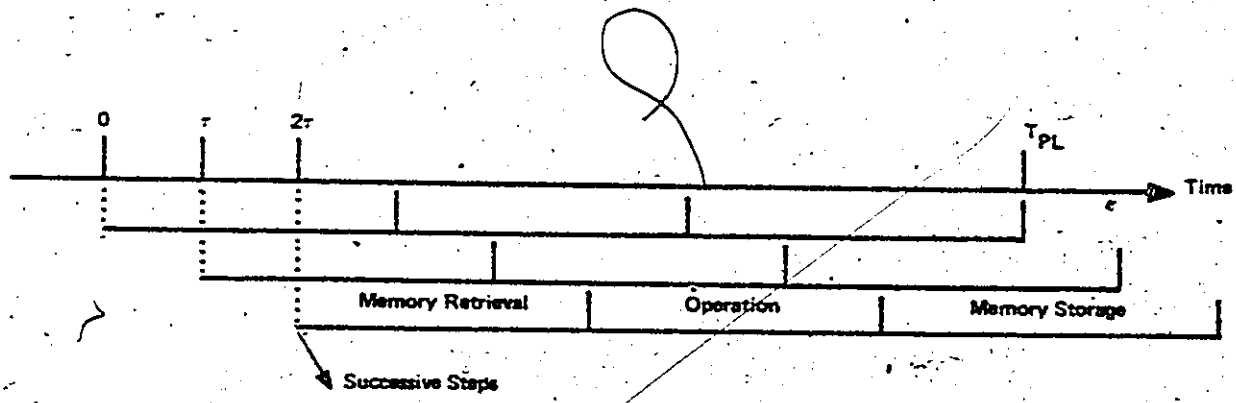


Fig. 2.6 Pipeline Processor-Sequence of Activities

hardware of this machine is realized using LSI technology. One main reason for this is the economic realization offered by LSI. Moreover, the hardware required to make the arithmetic and memory units work as a pipeline is generally expensive, so that if the pipeline processor is to be economical, the pipe must be kept full a substantial part of the time. This requires a high speed memory with high speed output rate, which is again offered by LSI.

2.3 THE LOGIC-IN-MEMORY ORGANIZATION

A logic-in-memory computer is organized around a logic-enhanced high-speed "cache" memory array. Used as a cache, a logic-in-memory array performs as a high-speed buffer between a conventional CPU and a conventional memory. The effect on the computer system of the cache and its control mechanism is to increase the speed of processing information by reducing the retrieval time.

This approach of integrating the logic and the high-speed memory function of a computer was made possible by the advances of LSI. In LSI technology, the cost of a package is only partly influenced by the number of gates or the number of devices per package. As a result a module of high complexity is desirable to be chosen as modular building block. The computer memory is inherently modular by nature and if control logic is included, the result is a logic-in-memory cell which is suitable for LSI. These cells are organized in regular arrays (LIM arrays) to perform the basic functions in a computer system e.g. transfer and sorting. In chapter 6, the organization of these arrays is explained further and their realization, using advanced high-speed LSI, is presented.

One, or a combination, of the above organizations can be implemented in a computer system to increase the speed of processing data and hence the throughput. The realization of such a system using LSI can offer a hardware operating at subnanosecond delays as will be shown in the following chapters.

CHAPTER 3

MODULAR LOGIC DESIGN

MODULAR LOGIC DESIGN

In the realization of digital systems using LSI the following phases of the design can be identified [1-4]:

1 - System Study

The system is defined and partitioned into subsystems. Timing and interface relations among parts of the system are specified.

2 - Technology Study

A Large-Scale-Integration technology is chosen for the system realization, e.g. bipolar, dynamic N-channel-MOS, Complementary-MOS or static P-channel-MOS.

3 - Subsystem Partitioning

Each subsystem is partitioned into functional blocks, where each block is realizable on a single LSI chip. Such partition is a function of design factors including size of chips, power dissipation per chip and the total number of chips.

4 - Logic Design

Each functional block (chip) is translated into logical blocks e.g. logic gates, flip-flops, counters, decoders, etc. In this phase, a logic simulator is usually used to verify the design with respect to race conditions, total delays and correct operation.

5 - Circuit Design

A circuit design for each chip is done using the basic elements for the LSI technology chosen, e.g. in bipolar LSI technology the most commonly used elements are the npn transistor and the diffused resistor. A computer aided circuit analysis program may be used to analyze the circuits. It is possible to reduce the design effort if a single circuit structure is used to realize logical blocks on all the chips.

6 - Layout and Digitization

A layout of the chip is made for each mask level and the different dimensions are fed into a computer-controlled mask cutting machine for cutting the mask. Here a great use is made of the similarity between the different parts of the chip. For example, step, repeat; flip, rotate statements are frequently used in the computer program to lay out similar sections of the chip.

7 - Photoreduction

For each mask level, a photographic plate is made to be used in the processing of a wafer which contains some number of chips. A step-and-repeat camera is used to reproduce a number of copies of the same chip on a single plate.

8 - Processing

Using the photographic plates, the wafer is processed according to the technology used.

9 - Testing and Packaging

In this final phase, testing the individual chips before and after packaging is done. Here also similarities between different chips simplifies the process of generating test sequences and detecting faulty functions.

Throughout these design phases, modularity of the design i.e. using similar modules to build different blocks, greatly reduces the total design effort and hence the total cost [4].

Because of this modularity requirement, a great deal of research has been done in modular approaches for LSI logic design. There have been two main approaches:

(1) The universal logic module approach [5, 6, 7], and

(2) The cellular network approach [8, 9, 10].

In the universal logic module approach, the problem is to find an n-variable universal module which can realize all functions of m variables ($m \leq n$). In will be shown that as n increases, the complexity of the module increases drastically and it becomes uneconomical to realize functions of M variables ($M \ll n$). For cellular networks the problem is different. The building block is usually specified and for a specific cost function the interconnection pattern is to be determined and, in most cases, it is not simple [10].

The above approaches are not based entirely on meeting the LSI technology requirements and limitations. For example, the power dissipation and the silicon area for a given logic function should be considered in designing such modules. In section 3.3 of this chapter we shall introduce such an approach where a "nearly optimum" module is designed to meet LSI high-speed system requirements.

In section 3.1 and 3.2 we review the universal logic module and cellular structure approaches indicating the feasibility of realizing each block using today's LSI technology.

3.1 UNIVERSAL LOGIC MODULES

3.1.1 Definitions

Definition 1

A logic function $U(Z_1, Z_2, \ldots Z_m)$ is universal in n variables $x_1, x_2, \ldots x_n$, if replacement of each Z_i by some member of a set S (set of source variables) produces

$$U(Z_1, Z_2, \ldots Z_m) = f(x_1, \ldots x_n),$$

for every function $f(x_1, \ldots x_n)$, $n \geq 1$.

For example, the function

$$U(Z_1, Z_2, \ldots Z_3) = Z_1 Z_2 + \overline{Z}_2 Z_3$$

is universal in 2 variables since it can realize all functions of 2 variables

Definition 2

A universal logic module (ULM) is a network which realizes a universal logic function (ULF).

For the above example, Fig. 3.1 shows an ULM realization using AND/OR gates.

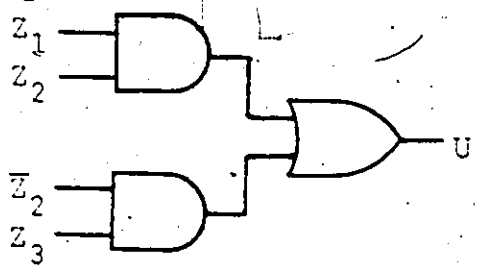


Fig. 3.1 ULM for $U = Z_1 Z_2 + \overline{Z}_2 Z_3$

Definition 3

One criterion for comparing different ULF's is the number m of its input variables. We introduce here a figure of merit (M) for a ULF which is defined as $\frac{n}{m-n}$. It indicates the extra inputs required to realize all functions of n variables.

For the above example, m = 3, n = 2, M = 2.

3.1.2 The Set "S" Of The Source Variables

Design methods for ULM's and their realization depend upon the choice of the set of the source variables S from which replacements for Z_i are obtained. Three cases are distinguished

in the literature:

Case 1

$$S = \{0, 1, x_1, x_2, \ldots x_n, \bar{x}_1, \bar{x}_2, \ldots \bar{x}_n\}$$

i.e. each variable is available in complemented and uncomplemented form and, in this case, S has 2 (1 + n) elements.

Case 2

$$S = \{0, 1, x_1^*, x_2^*, \ldots x_n^*\}$$

where x_j^* denotes either x_j or $\bar{x}_j$, i.e. each variable is available in only one form and in this case S has (n + 2) elements.

Case 3

$$S = \{0, 1, x_1^*, \ldots x_{j-1}^*, x_{j+1}^*, \ldots x_n^*, x_j, \bar{x}_j\}$$

in which only one variable is available both in the uncomplemented and the complemented form and in this case S has (n + 3) elements.

Case 1 above represents a practical situation, where logic operations are performed directly from a memory system or a chain of flip-flops, where each variable and its complement are available. As indicated later, this case has the highest figure of merit, which is obtained at the expense of making the variables and their complements available at the input of ULM's.

Case 2 represents another practical situation where logic operations are performed and designed to use a variable or its complement but not both. This simplifies the interconnection patterns, but it has the lowest M.

Case 3 represents intermediate situation between Case 1 and Case 2 and is only of theoretical interest.

3.1.3 The Universal Logic Module

There has been research in the area of universal modules recently directed towards the following problems:

(1) Given a set containing V functions, each of which is a function of not more than n variables, find a universal module with m inputs which can produce each of the V functions.

(2) Find the upper and lower bounds on the number of inputs m.

(3) Find algorithms to synthesis universal functions and optimize such algorithms.

Forsulund and Waxman [11] and Elspas et al [12] have studied ULM's but no algorithms for their synthesis are given. The results of Meo [13] and Waxman [14] contain such algorithms, but no optimization procedure is given. The work of Yan and Tang [6] includes some optimization procedure for $n \geq 6$. Theoretical work developed in this area due to Preparata and Muller [5] will be reviewed in this chapter and compared with Yan and Tangs' results.

3.1.4 The Universal Logic Functions

A logic function of n variables can be expanded canonically,

$$f(x_1, x_2, \ldots x_n) = f_o x_1 x_2 \ldots x_n + f_1 \bar{x}_1 x_2 \ldots x_n + \ldots + f_{2^n-1} \bar{x}_1 \bar{x}_2 \ldots \bar{x}_n$$

where f_i may be 0 or 1.

Such logic functions can be realized by a universal function U

$$U = Z_{n+1} Z_1 Z_2 \ldots Z_n + Z_{n+2} \bar{Z}_1 \ldots Z_n + \ldots + Z_{n+2^n} \bar{Z}_1 \bar{Z}_2 \ldots \bar{Z}_n$$

where $Z_1 = x_1$, $Z_2 = x_2$, $\ldots$ $Z_n = x_n$ and $Z_{n+1+i} = f_i$

This universal function has $m = n + 2^n$ variables and therefore

is uneconomical compared with the lower bound given by Elspas [12]. The lower bound is $\lceil 2n/\log_2(2n+2) \rceil$ because there are only $(2n+2)^m$ possible assignments of the Z_i's (case 1 above). Because each of 2^{2^n} logical functions of n variables must have a different assignment, m will be at least as large as $\lceil 2^n/\log_2(2n+2) \rceil$. Thus, the problem reduces to achieving a ULF with m as close as possible to the lower bound.

Let us partition the 2^n minterms in the canonical expansion of a general function $f(Z_1, \ldots Z_n)$ into mutually exclusive blocks $R_1, R_2, \ldots R_t$. Each R_i is a union of minterms, such that for each nonempty subset S_i of R_i there is at least one variable $x_j^* \in S$ (the set of source variables) such that x_j^* appears uncomplemented (or complemented) in only the minterms of S_i.

The ULF can be constructed as follows:

$$U = Z_{n+1} R_1 + Z_{n+2} R_2 + \ldots + Z_{n+t} R_t$$

In using this ULF to realize $f(x_1, \ldots x_n)$, the first n variables $x_1, \ldots x_n$ are assigned to $Z_1, \ldots Z_n$, i.e. $Z_j = x_j^*$ for $j = 1, 2, \ldots n$. The assignment of Z_{n+i} can follow this rule:

Let $S_i = R_i \cap f$

(1) if $S_i = \emptyset$ (empty set), then $Z_{n+i} = 0$

(2) if $S_i = R_i$, then $Z_{n+i} = 1$

(3) if S_i is a proper nonempty subset of R_i, then there is a variable $x_j^* \in S$ which appears in the same form in S and in only the minterms of S_i and we assign $Z_{n+i} = x_j$

Selecting all Z_{n+i} in this manner ($i = 1, 2, \ldots t$) realizes $f(x_1, \ldots x_n)$ where $m = n + t$.

Example

For the case where $S = (0, 1, x_1, \bar{x}_1, x_2, \bar{x}_2)$, let us consider $f(Z_1, Z_2)$ with 4 minterms: Z_1Z_2, $Z_1\bar{Z}_2$, $\bar{Z}_1Z_2$ and $\bar{Z}_1\bar{Z}_2$

[A] let the partition be

$$R_1 = (Z_1Z_2, Z_1\bar{Z}_2)$$

$$R_2 = (\bar{Z}_1Z_2, \bar{Z}_1\bar{Z}_2)$$

for R_1: It has two subsets $S_{11} = (Z_1Z_2)$ and $S_{12} = (Z_1\bar{Z}_2)$ where Z_2 appears uncomplemented only in S_{11} and complemented only in S_{12}.

For R_2: It has two subsets $S_{21} = (\bar{Z}_1Z_2)$ and $S_{22} = (\bar{Z}_1\bar{Z}_2)$ where Z_2 appears uncomplemented only in S_{21} and complemented only in S_{22}.

Therefore R_1 and R_2 satisfies the partition condition, and the ULF, U can be expressed as

$$U = Z_3 (Z_1Z_2 + \bar{Z}_1\bar{Z}_2) + Z_4 (\bar{Z}_1Z_2 + \bar{Z}_1\bar{Z}_2)$$

$$= Z_3Z_1 + Z_4\bar{Z}_1$$

i.e. 3 inputs are required.

(1) To realize the function $f_1 = x_1\bar{x}_2 + \bar{x}_1x_2$

let $Z_1 = x_1$, $Z_2 = x_2$

(a) $S_1 = R_1 \cap f_1 = x_1\bar{x}_2$, then $Z_3 = \bar{x}_2$

(b) $S_2 = R_2 \cap f_1 = \bar{x}_1x_2$, then $Z_4 = x_2$

(2) to realize $f_2 = x_1x_2$

let $Z_1 = x_1$, $Z_2 = x_2$

(a) $S_1 = R_1 \cap f_1 = x_1x_2$, then $Z_3 = x_2$

(b) $S_2 = R_2 \cap f_2 = \emptyset$, then $Z_4 = 0$

[B] Another possible partition is

$$R_1 = (Z_1Z_2, \bar{Z}_1\bar{Z}_2),\ S_{11} = Z_1Z_2,\ S_{12} = \bar{Z}_1\bar{Z}_2$$

$$R_2 = (Z_1\bar{Z}_2, \bar{Z}_1Z_2),\ S_{21} = Z_1\bar{Z}_2,\ S_{22} = \bar{Z}_1Z_2$$

In R_1: Z_1 and Z_2 appear uncomplemented only in S_{11} and appear complemented only in S_{12}.

In R_2: Z_1 appears uncomplemented only in S_{21} and complemented only in S_{22}.

Z_2 appears uncomplemented only in S_{22} and complemented only in S_{21}.

Therefore R_1 and R_2 satisfies the partition condition and

$$U = Z_3(Z_1Z_2 + \bar{Z}_1\bar{Z}_2) + Z_4(Z_1\bar{Z}_2 + \bar{Z}_1Z_2)$$

i.e. 4 inputs are required.

To realize $f_2 = x_1x_2$

(a) $S_1 = R_1 \cap f_2 \equiv x_1x_2$, then $Z_3 \equiv x_1$ or x_2

(b) $S_2 = R_2 \cap f_2 = \emptyset$, then $Z_4 = 0$

In order to minimize t and hence m, it is desirable to make the number of minterms in the blocks as large as possible and the problem reduces to forming such partitions of minterms. The algorithm given by Preparata and Muller, for

$$S = \{0, 1, x_1, \bar{x}_1, \ldots, x_n, \bar{x}_n\}$$

allows the designer to minimize m and it is shown that m can approach the lower bound as n approaches infinity. Table 3.1 compares the results of Preparata and Muller (P and M) and Yan and Tang (Y and T) to the lower bound. As shown, m increases rapidly as n increases, e.g. for a ULF of 5 variables, at least 15 inputs are needed.

TABLE 3.1

RESULTS OF MULLER & PREPARATA (M & P)
AND YAN & TANG (Y & T)

n	2	3	4	5	6	7	8	9
2^n	4	8	16	32	64	128	256	512
Lower Bound of m								
Case 1: $\frac{2^n}{\ln (2n+2)}$	2	4	5	9	17	32	62	119
Case 2: $\frac{2^n}{\ln (n+2)}$	2	4	8	16	22	41	77	148
Case 3: $\frac{2^n}{\ln (n+3)}$	2	4	8	11	21	39	75	143
m of (M & P): Case 1	3	5	9	15	26	-	-	-
m of (Y & T): Case 1	-	-	-	-	-	-	-	-
m of (M & P): Case 2	4	7	12	20	36	61	107	196
m of (Y & T): Case 2	4	7	12	20	34	51	84	149
m of (M & P): Case 3	4	6	10	19	32	55	99	186
m of (Y & T): Case 3	3	5	9	19	37	70	135	264

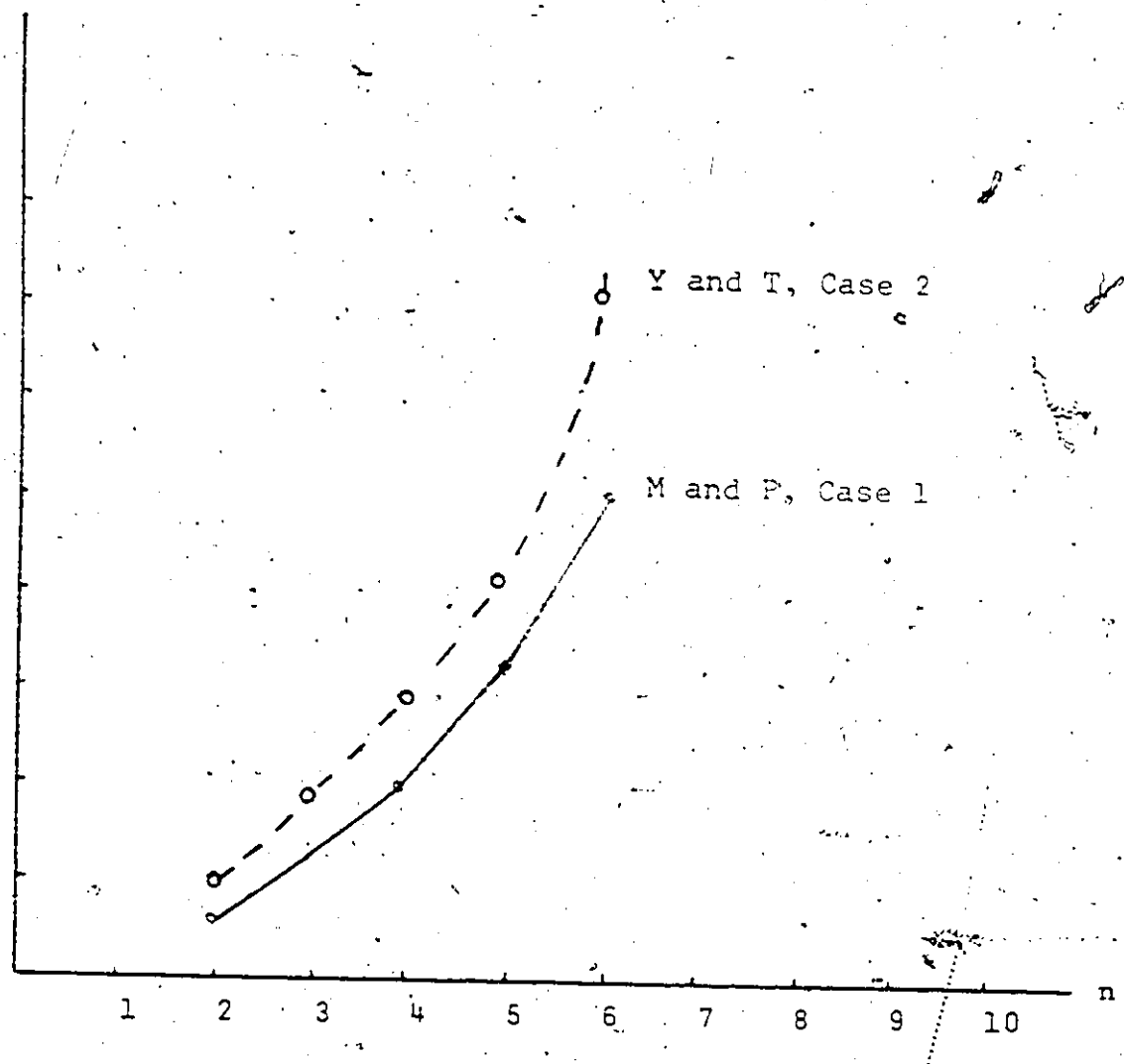


Fig. 3.2 n vs m of Table 3.1

3.1.5 Multirealization Using ULM's

The large number of inputs m of a ULM is a practical limitation. In this case, ULM's with small numbers of variables can be used as building blocks to implement any logic function in a tree structure. The method of Yan and Tang is indicated below.

Any logic function $f(x_1, \ldots x_n)$ can be expanded with respect to any (n-1) of the n variables as follows:

$$f(x_1, \ldots x_n) = \sum_{i=0}^{2^{n-1}-1} x_1^{i_1} x_2^{i_2} \ldots x_{n-1}^{i_{n-1}} f(i_1, i_2, \ldots i_{n-1}, x_n) \quad (1)$$

where the superscripts $i_1, i_2, \ldots i_{n-1}$ form the binary representation of i and

$$x_j^0 = \bar{x}_j \quad , \quad x_j^1 = x_j \quad , \quad j = 1, 2, \ldots, n-1. \quad (2)$$

The residue function $f(i_1, \ldots i_{n-1}, x_n)$ is a function of a single variable x_n and hence can assume one of the four values: x_n, $\bar{x}_n$, 0 or 1. It is seen from (1) that any logic circuit that realizes the function U_n of the variables $c_1, \ldots, c_{n-1}, a_0, a_1, \ldots a_{2^{n-1}-1}$ can be used to construct a ULM of n variables where

$$U_n = \sum_{i=0}^{2^{n-1}-1} c_1^{i_1} c_2^{i_2} \ldots c_{n-1}^{i_{n-1}} a_i \quad (3)$$

where $i_1, i_2, \ldots i_{n-1}$ form the binary representation of i and

$$c_j^0 = \bar{c}_j \quad , \quad c_j^1 = c_j, \; j = 1, 2, \ldots, n-1 \quad (4)$$

The simplest NAND realization of (3) and its symbol is shown in Fig. 3.3 where C_i and A_j correspond to c_i and a_j. To implement $f(x_1, \ldots x_n)$, C_i must be connected to x_i and A_j to the residue function $f(j_1, j_2, \ldots j_{n-1}, x_n)$.

It is seen that the module has a NAND gate with a fan-in (number of input at that gate) of 2^{n-1} e.g. for n=5, fan-in=16. The total number of inputs is $2^{n-1} + n-1$ (for n=5, the number of inputs=20).

ULM's of n variables can be used to implement fuhctions of m variables (m > n) in a tree structure. As a result, the fan-in number reduces, while the number of gates and the total delay increase. The total number of inputs remains the same. This can be illustrated by using ULM's of 3-variables to implement functions of 5-variables.

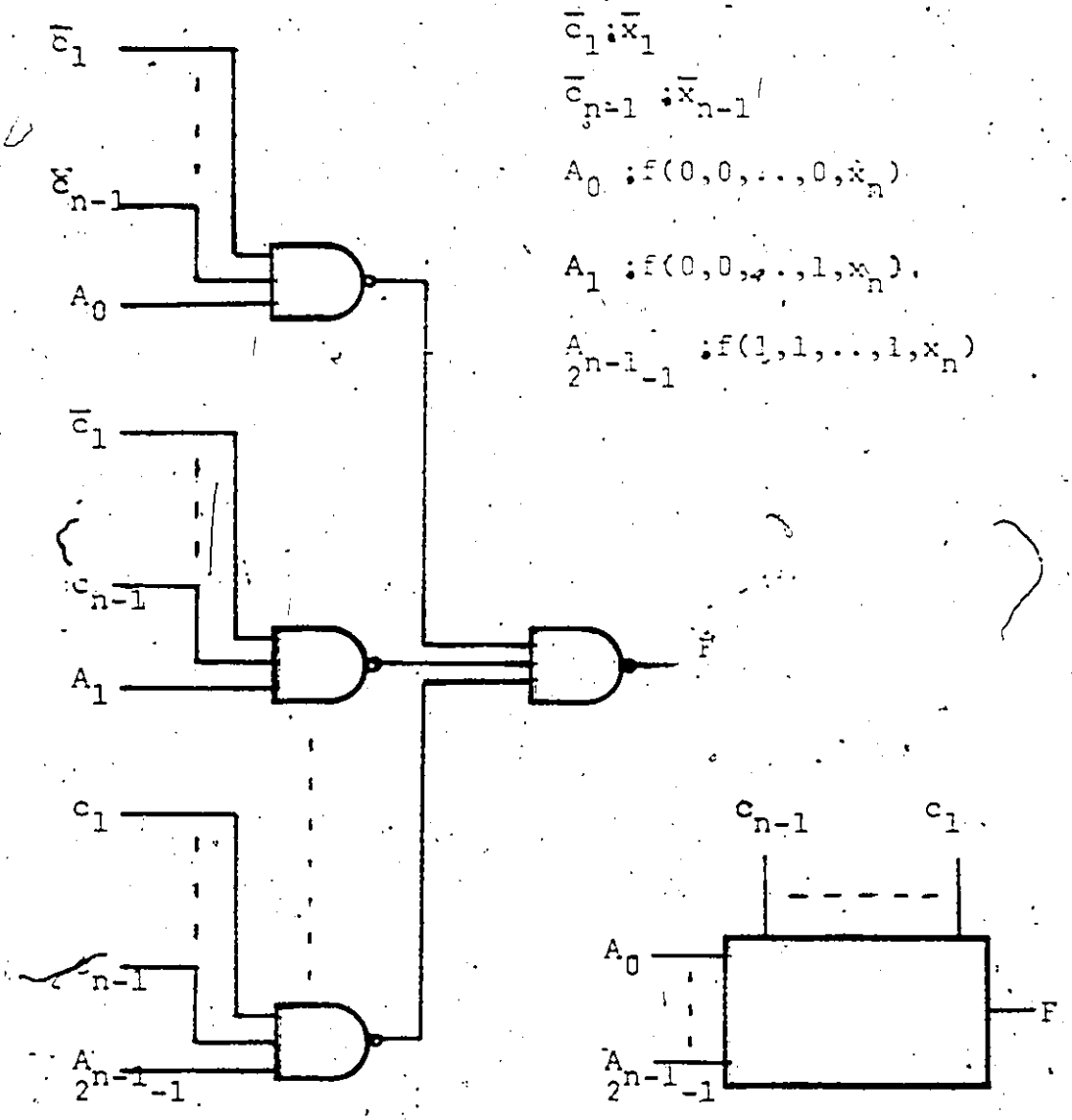


Fig. 3.3 NAND Gate Realization of General ULM

The function $f(x_1, x_2, \ldots x_5)$ is first expanded with respect to the variables x_1 and x_2 as follows

$$f_5 = \bar{x}_1\bar{x}_2\, f(0, 0, x_3, x_4, x_5) + \bar{x}_1 x_2\, f(0, 1, x_3, x_4, x_5)$$

$$+ x_1\bar{x}_2\, f(1, 0, x_3, x_4, x_5) + x_1 x_2\, f(1, 1, x_3, x_4, x_5) \qquad (5)$$

Since each residue function in (5) is a function of three variables, they can be realized using ULM's of 3-variables, as shown in Fig. 3.4. The method can be extended to implement any logic function of n variables by using ULM's of k variables ($k < n$).

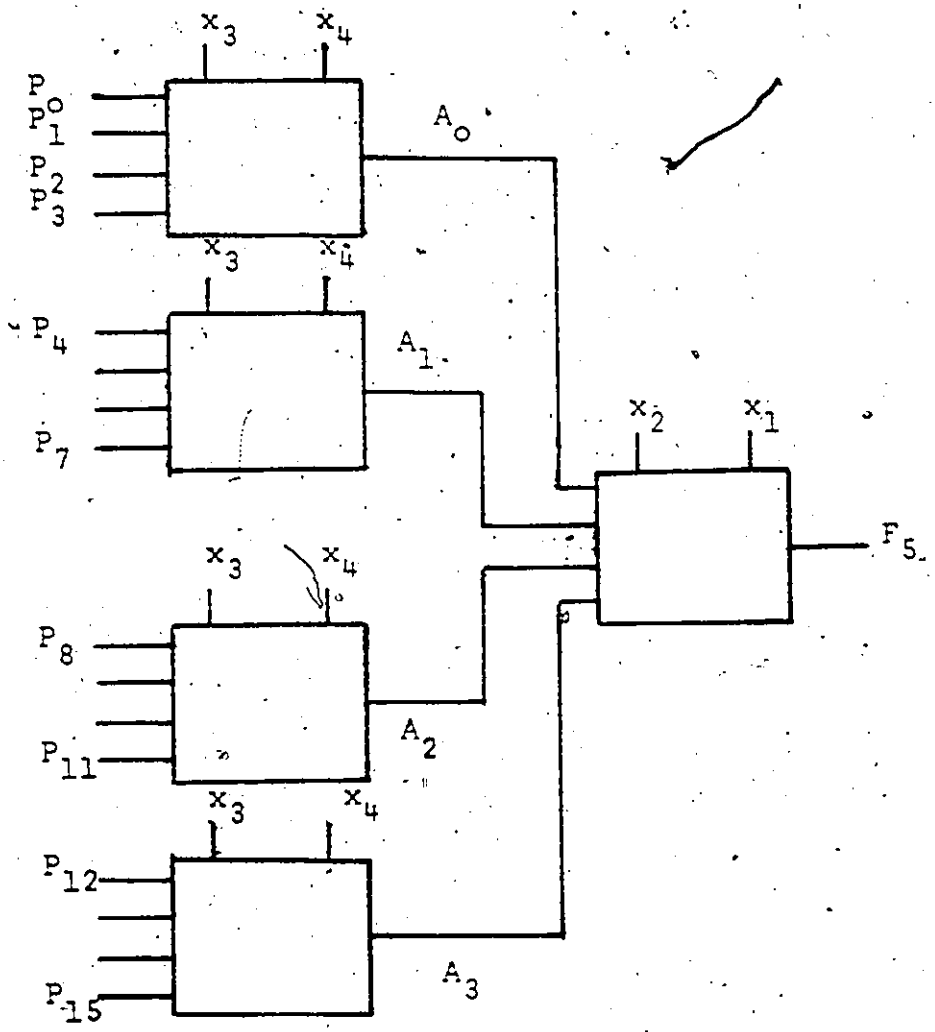


Fig. 3.4 Multirealization of 5-variable functions using ULM of 3-variable.

A_0, A_1, A_2, A_3 are the residue functions and P_0-P_{15} are x_5, $\bar{x}_5$, 0 or 1.

3.1.6 Practical Limitations Of Realizing ULM's

(1) Fan-in Number

The high fan-in number (F_i) required by ULM's, where m is its upper bound, limits the performance of the LSI realization in terms of speed, power dissipation, silicon area and particularly the complexity and reliability of the interconnection system. These limitations increase as the number m of the universal inputs increases and there is a practical fan-in number; applying current-technology it is between 5-10. i.e. only ULM's of up to 5 variables are economically realizable. To overcome such problems, the multirealiration approach can be used to reduce the fan-in number. However, this increases the overall number of interconnections and decreases the reliability. As a result, a compromise is required to select a ULM of near optimum complexity.

(2) System Cost

When designing a system using ULM's it is usually not possible to make full use of the logic complexity in all the modules. Sometimes it may be necessary to use a fairly complex module to perform a simple invertion function. The underutilization of the modules increases the system cost. Thus, the complexity of ULM chosen for any system is the result of a compromise and a function of the technology available [15].

(3) Logic Flexibility

The realization of a digital system using ULM's imposes a restriction on the logic designer; only one type of building block is available. For example, if a ULM of 3 variables is used as a building block, it will also be used to realize a simple inverter, if required, in a system.

(4) Power Dissipation

The total power dissipation on a LSI silicon chip is limited and the selection of ULM's must meet this limit. As the number of variables of a ULM increases, its complexity and hence its power dissipation increases. At the limit, special cooling systems must be used or the ULM's must be realized on several silicon chips. This will result in a lowering of production yields.

(5) Modularity of Circuit Structures

The design of a system of ULM's must be considered in relation to the realization of separate modules. If the module is realized in terms of simple gates [6, 7], there will be unused gates in the underutilized modules, and the full advantage of the LSI technology will not be gained. However, there is an alternative approach where the function chosen for the module is dependent on what may be simply performed within the contraints of the technology employed. Such an approach will be indicated in section 3.3.

3.2 CELLULAR ARRAYS

In section 3.1, it was indicated that the high complexity of the universal logic modules presents a serious limitation for their LSI realization. In this section we shall review an alternative approach towards modularity, namely, the cellular arrays. In this approach, the complexity of the building block is reduced at the expense of increasing the complexity of interconnection-patterns. The practical realization of such ararys is discussed related to current LSI technology.

3.2.1 Cellular Logic

Cellular logic is based on selecting a simple cell or module and interconnecting a number of such cells, in an array form, to realize a particular logic function. Two types of arrays can be defined [16]:

(1) Fixed Cell-Function Arrays

In these arrays, the logic function produced by each cell is fixed, and the modification of the interconnection structure of these cells results in different logic functions.

(2) Variable Cell-Function Arrays

In these arrays, the modification of the interconnection structure as well as the function produced by each cell is permitted, resulting in a greater flexibility.

(1) Fixed Cell-Function Arrays

In the study of fixed cell-function arrays the most used function has been one of the following:

(1) NOR or NAND gate of two-eight variable,
(2) Three-input majority gate, or
(3) Exclusive-OR gate with AND gate

The above cells can be used in arrays with different interconnection patterns. Some arrays are indicated below.

(A) The Eight-Neighbor Array

Each cell in this array is the 8-input NOR gate [17], while the cells are interconnected into an array. Such an array could take the form of a tetrahedral net based on the cell shown in Fig. 3.5.

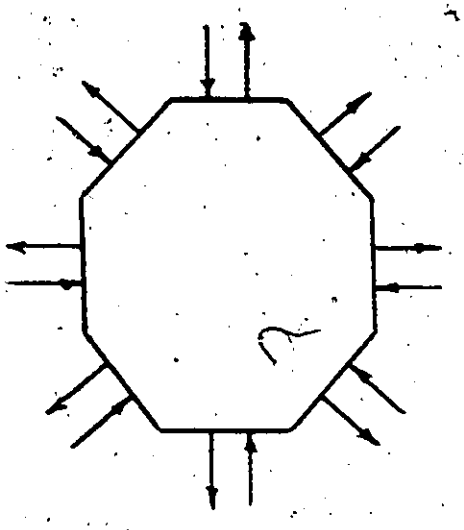


Fig. 3.5 Eight-Neighbor Array Cell

Logic design procedures to use this array are given [16, 18]. The array requires $2(n+1)(2^n+1)$ cells to realize an n-variable function. The array growth rate, which is the number of cells required as n increases, is $n\,2^{n+1}$. As a result a function of 5 variables may require 320 cells, each has a fan-in and a fan-out of 5.

(B) Majority-Gate Arrays

These arrays have been considered in the literature [19, 20, 21], where each cell is a 3-input majority gate. Such an array is due to Short [16, 21] which has a growth rate of $3n\,2^n$. A 5-variable function may require 480 cells with a fan-in of 3 and a fan-out of 2.

(C) Adder-Arrays

The simplest form of these arrays [21] is shown in Fig. 3.6 which consists of two-input, two output cells that produce the Exclusive OR horizontally and the AND vertically. Such arrays produce all the minterms P_o, P_1, ... P_{v-1} of the set of variables x_1, x_2, ... , x_n where $v = 2^n$. This is called a decoder array. By collecting a subset of these minterms in a second type of cell that produces the OR at the bottom of the decoder array, it is possible to synthesize arbitrary combinational functions. The second type of array is called a collector array.

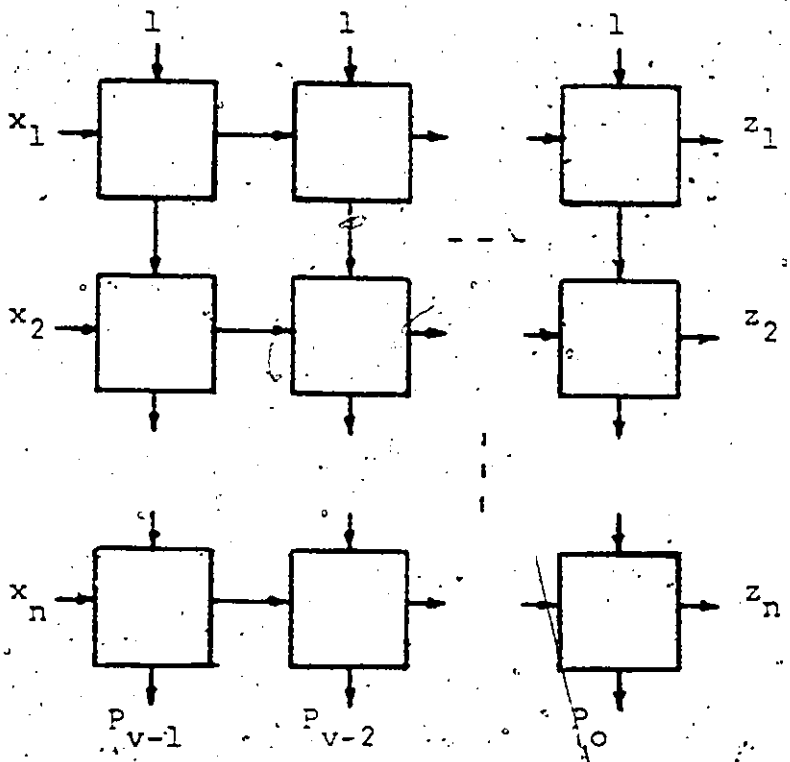


Fig. 3.6 Two-dimensional Decoder Array

Another type of adder array is the folded adder array which is formed by the combination of a decoder and a collector array of the same dimensions and has $n2^n$ cells [21] e.g. a 5-variable function may require 160 cells.

(2) Variable Cell-Function Arrays

A one-dimensional array, which is considered to be a basic variable cell-function array, was studied by Maitra [22] and is shown in Fig. 3.7 and is referred to as Maitra Cascade.

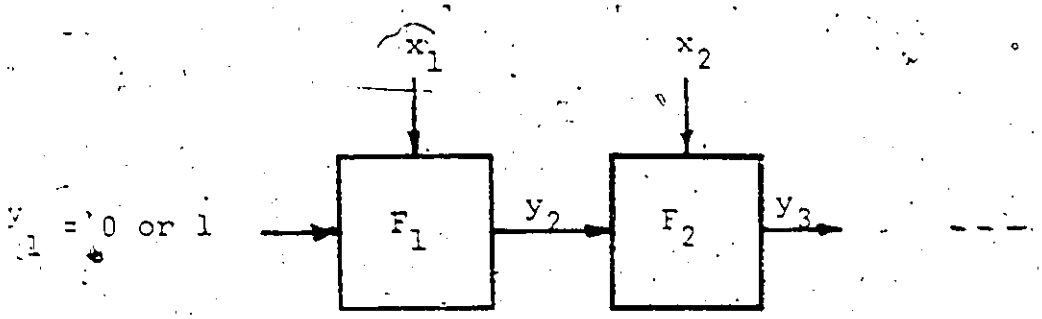


Fig. 3.7 The Maitra Cascade

Each cell can be any of the 16 functions of two or fewer variables. It was shown [23] that only 6 of the 16 functions are necessary. The cascade realizes a fraction (x) of functions of n variables. If certain variables are repeated, the fraction x increases and some design procedures have been suggested [24]. A two dimension array, which consists of arrays of such cascades feeding a single collector cascade of AND or OR cells, can be used to realize any n-variable function as shown in Fig. 3.8. An optimal design procedure has been given [24].

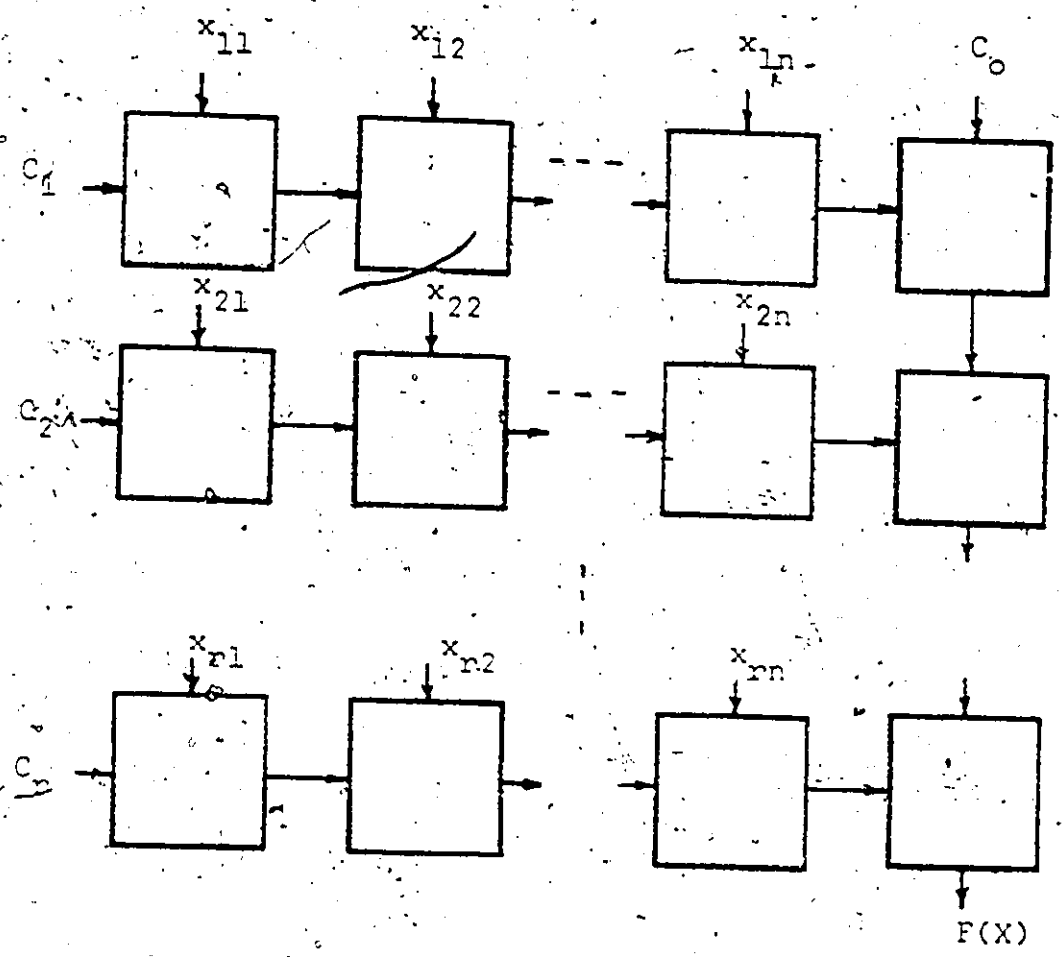
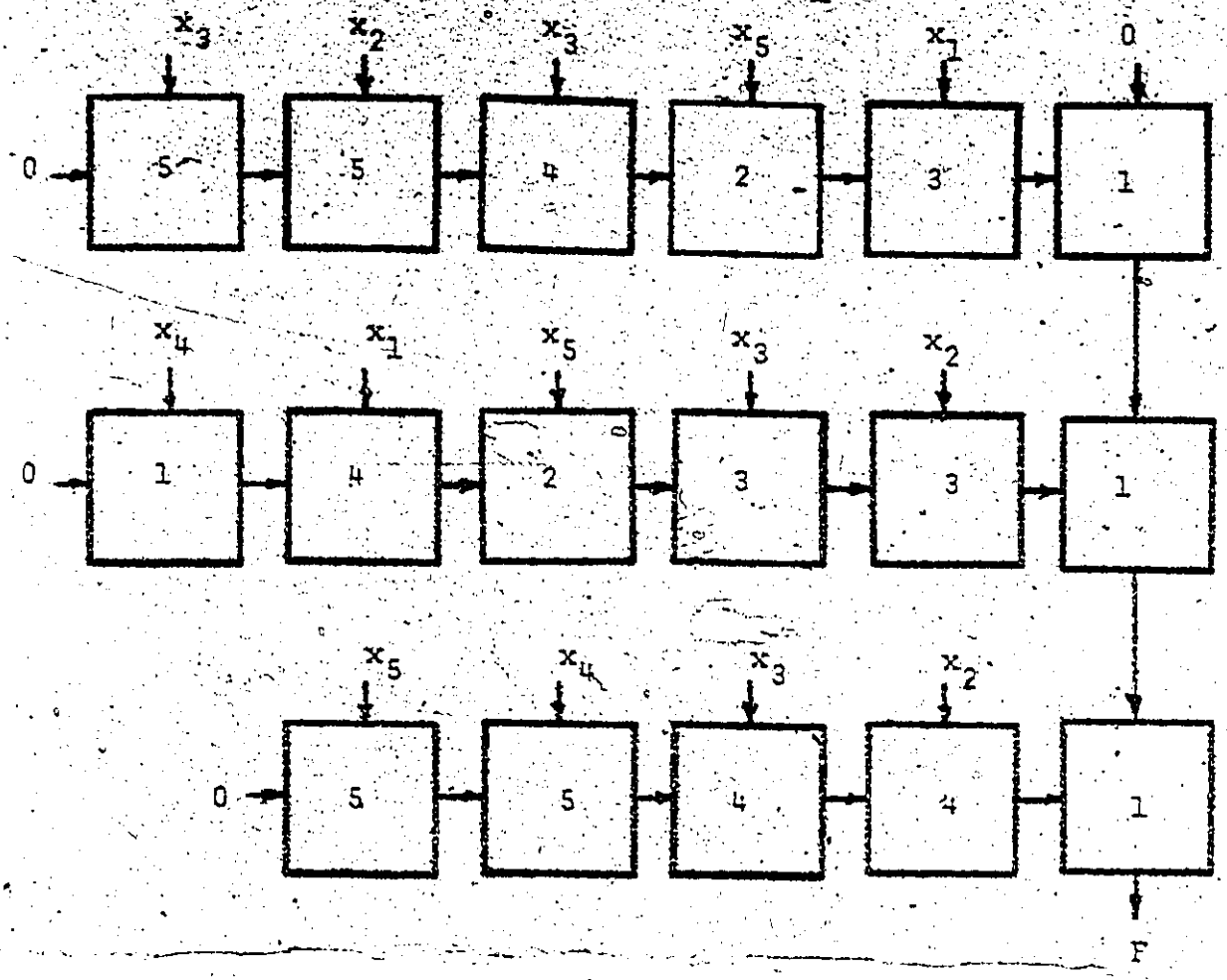
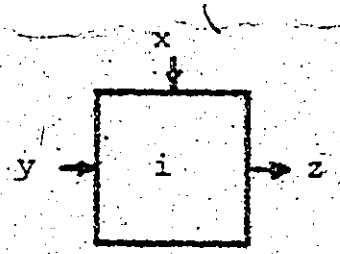


Fig. 3.8 An array of cascades feeding a common collector cascades

If for every i, $x_{ij} \neq x_{ik}$ $j \neq k$, then the array is an irredundant cascades realization of F(x). Otherwise, it is called a redundant cascades realization. Fig. 3.9 shows an optimal realization of the function [25]:

$F_5(x_1, x_2, x_3, x_4, x_5) = \Sigma m(0,2,4,6,7,8,10,11,12,13,14,16,19,29,30)$ using 17 cells.





Cell Type i	z	Cell Type i	z
1	$\overline{x+y}$	6	$\overline{x+y}$
2	$\overline{x}+y$	7	$\overline{x}+\overline{y}$
3	$\overline{x}\overline{y}$	8	$\overline{x}\overline{y}$
4	xy	9	$x\overline{y}$
5	$x\overline{y}+\overline{x}y$	10	$x\overline{y}+\overline{x}y$
	$= \overline{x} \oplus y$		$= x \oplus y$

Fig. 3.9 Realization of F_5 and a table for cell types used.

Several cellular arrays have been reported in the literature based on the Maitra Cascade [26, 27]. They have a growth rate which can be expressed as $K\, n2^n$ where K is a constant depends on the type of the array and is independant of n. For example, the cutpoint array [21] has a growth rate of $\frac{n}{4}\, 2^n$. Thus a 5-variable function may require 40 cells using the cutpoint array.

3.2.2 Practical Limitations of Realizing Cellular Arrays

It has been shown that the building cell is usually simple although the interconnection pattern is complex for today's LSI interconnection technology, and high numbers of crossovers may be required which presents a serious limitation. The regularity of the interconnection pattern does not represent a great advantage, since this is achieved at the expense of large numbers of cells with high fan-in and fan-out numbers. As a result, the cellular array may not be the optimum approach for logic design for LSI. Alternatively, if the basic building block is based on a cell designed according to LSI requirements and limitations, an optimum design may result.

3.3 FUNCTION REALIZATION

In the last two sections, we have reviewed the Universal and Cellular logic design approaches and have indicated the difficulties in their LSI realization. We have also shown that the optimal approach to make full use of LSI technology is to take a modular function approach i.e., an approach based on the following:

(a) To design a circuit structure to meet LSI requirements and limitations.

(b) To use the structure modularly in the realization of logic functions, flip-flops, memory cells, ...etc.

(c) To develop logic techniques to use the structure efficiently in logic realization.

(d) To demonstrate the flexibitity of the structure in the realization of function cells, adders, multipliers,... etc.

The above approach was taken in this research work and resulted in a circuit structure for high-speed LSI arithmetic and logic operations. The following chapter will discuss this circuit structure.

CHAPTER 4

METLL STRUCTURES-
A HIGH-SPEED LSI LOGIC SYSTEM

METLL STRUCTURES-
A HIGH-SPEED LSI LOGIC SYSTEM

In this chapter the Multi-Emitter Two-Level Logic (METLL) circuit structure is presented. Its function, its use in the realization of logic gates and flip-flops and its dc and transient characteristics along with practical results are given. A comparison is made between the METLL structures and some available high-speed logic families. The METLL structures in logic realization is sometimes referred to as EFL logic family (Emitter-Function-Logic) [1, 2, 3].

Before we present the METLL structure, design parameter specification of logic systems is introduced in section 4.1 while section 4.2 deals with high-speed circuit design considerations and the basic operation of Emitter-Coupled Logic (ECL).

4.1 SYSTEM MODEL FOR LOGIC CIRCUITS

System aspects of logic circuits have been described in the literature [4-11]. In this section, a system model is used to define system parameters.

In a logic system, one of the basic logic operations is transferring a logic signal from a logic block (logic source) to the input of another logic block (logic detector) through a transmission system (Fig. 4.1). A logic block has usually more than one input and each output is used as an input variable for a number of other logic blocks. The number of inputs is called the fan-in and the number of blocks driven by one output is called the fan-out. For simplicity only single-input-single-output logic blocks are considered.

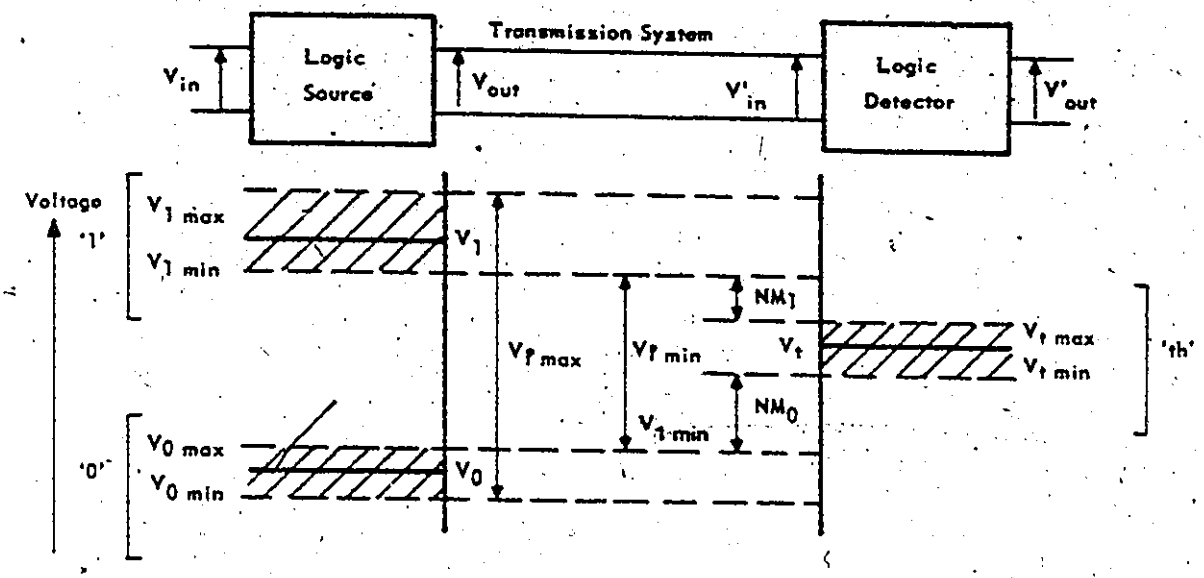


Fig. 4.1 System Model and Logic Levels

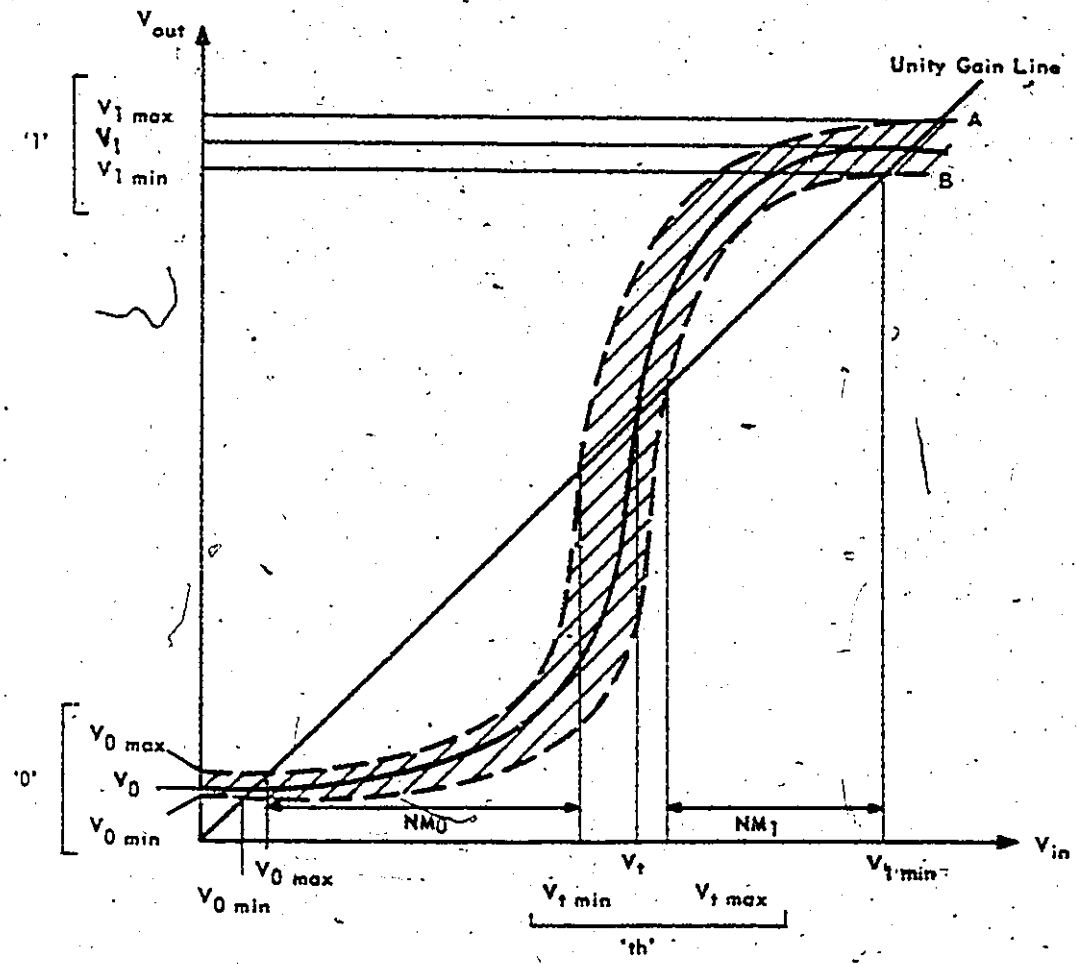


Fig. 4.2 DC Transfer Characteristic of e Non-inverting Logic Block

The output signal is at one of its binary levels, representing a bit of information, '0' or '1'. These two levels can be voltage, current, or impedence. However, in present day integrated circuits voltage levels, V_0 and V_1 are used. The logic system used here is a positive logic system, where the logical '1' level is positive with respect to the logical '0'.

At the input, the voltage signal is detected with respect to a threshold voltage V_t. In non-inverting logic blocks, if the input voltage is greater than V_t, the output voltage will be at logical '1' and if the input voltage is less than V_t the output voltage will be at logical '0'.

4.1.1 Noise Margins

Signal distortion, due to attenuation or system noise, causes a reduction of system reliability. Since distortion is primarily introduced by the noise in the system, it is conventional to refer to margins allowed for distortion of the logic levels as noise margins. Two noise margins are defined (Fig. 4.1):

$(NM)_0$ = noise margin for logical '0' = $V_{t\ min} - V_{0\ max}$

and

$(NM)_1$ = noise margin for a logical 'I' = $V_{1\ min} - V_{t\ max}$

4.1.2 DC Transfer Characteristic

Signal and noise margin definitions have been stated by considering the transmission system, the input and the output ports. Alternatively the input-output gate relationship(the DC transfer characteristic) also provides information

about the system. In Fig. 4.2, the unity gain line (with slope $\frac{dV_{out}}{dV_{in}}$ = unity) intersects the transfer characteristics at three points, the logical '0', the logical '1' and the threshold level 'th'. As the parameters of a particular design are varied over their tolerance ranges, the transfer characteristic (of a non-inverting gate) will be as shown by the dotted curves A and B. The shaded region between the limit envelopes A and B represents the region of possible transfer characteristics. Since the output does not reach its static value at the same instant as the input, a further spreading of the transfer characteristic results. Because of these static and dynamic tolerances, each level becomes a zone as shown in Fig. 4.1 and Fig. 4.2. The above definitions of noise margins are shown. It is also shown that:

(a) when the input is in the region ($V_{0\ min} \leq V_{in} \leq V_{t\ min}$)

the output is in the region ($V_{0\ min} \leq V_{out} \leq V_{t\ min}$)

(b) when the input is in the region ($V_{t\ max} \leq V_{in} \leq V_{1\ max}$)

the output is in the region ($V_{t\ max} \leq V_{out} \leq V_{1\ max}$).

i.e., the circuit will recognize and maintain two determined levels; a '1' above $V_{t\ max}$ and a '0' below $V_{t\ min}$. The region ($V_{t\ min} \leq V_{in} \leq V_{t\ max}$) represents an undetermined output and the logic signal lies in this region only during transitions from one level to the other.

4.1.3 Logic Swing And Delays

The difference between the two levels V_1 and V_0 is called the logic swing V_ℓ. The time it takes the output signal to transfer from one level to another through the undetermined

region is called the transient time (T) and the time between the input signal crosses the undetermined region and the output signal crosses the undetermined region is called the delay time (t_d). Propagation delay is sometimes used to refer to (T $\neq$ t_d).

4.2 HIGH-SPEED CIRCUIT CONSIDERATIONS

In today's LSI, there are two main technologies; bipolar and MOS. Bipolar technology is based mainly on the npn transistor and the diffused resistor, both fabricated in a single silicon crystal [12]. A Schottky-diode can be added to the technology to improve the speed of operation of some circuits e.g. in Schottky-Transistor-Transistor Logic(STTL) logic family [12]. MOS technology can be divided into

(a) single channel technology, N or P channel, where the N or P channel MOS transistor is the basic device, and

(b) Complementary-MOS (CMOS) technology, where a complementary pair (N and P channel MOS) is the basic building device.

Although MOS technology has the potential of operation at high-speed (for example by operating the devices in dynamic mode [13] and/or using silicon-on-dielectric-substrate technology, e.g., Silicon-on-Sapphire) it is still limited to hundreds of nanoseconds operation [13]. Alternatively bipolar technology offers the possibility of circuit operation with delays close to the nanosecond and subnanosecond range.

4.2.1 Saturating And Non-saturating Bipolar Logic Circuits

Bipolar circuits used to implement logic functions can be classified as saturating and non-saturating logic circuits [14].

In saturated logic circuits, e.g., TTL, the two

states of operation are in the cutoff and saturation regions. In non-saturated logic circuits, e.g. STTL, the two states are in the cutoff and the active region, as shown in Fig. 4.3.

In transistor operation, the excess stored-charge, which results when one of the states of a transistor lies in the saturation region, causes extra delay [6]. For high-speed operation several circuit arrangements, which prohibit saturation of the transistor, have been developed and one way of doing this is by controlling the collector current. This can be done by designing the circuit so that the emitter current is determined by the circuit elements and essentially independent of the base drive. Circuits which use this design philosophy are called current mode logic circuits and the emitter-coupled logic is one form of this class.

4.2.2 Emitter-Coupled Logic (ECL)

Figure 4.4 shows a simplified ECL circuit containing only those elements necessary for the explanation of its operation. The logical '1' and logical '0' voltage values are chosen to be more and less positive than the reference voltage V_r. Suppose that the input variable is '0', Q_1 will be cutoff and the current I_o will flow into Q_2 and the output at the collector of Q_2 is '0'. If the input variable change to '1', I_0 will flow into Q_1 and no current will flow in Q_2 and as a result the output voltage changes to '1'. This behavior is summarized by Fig. 4.5.

In general, the output logic signal consists of voltage levels which vary about a reference level different from the input reference level. Direct coupling, therefore, cannot be employed unless a very low logic swing is required; otherwise, coupling circuits are used to translate the output signal to the proper input level [14]. This coupling introduces

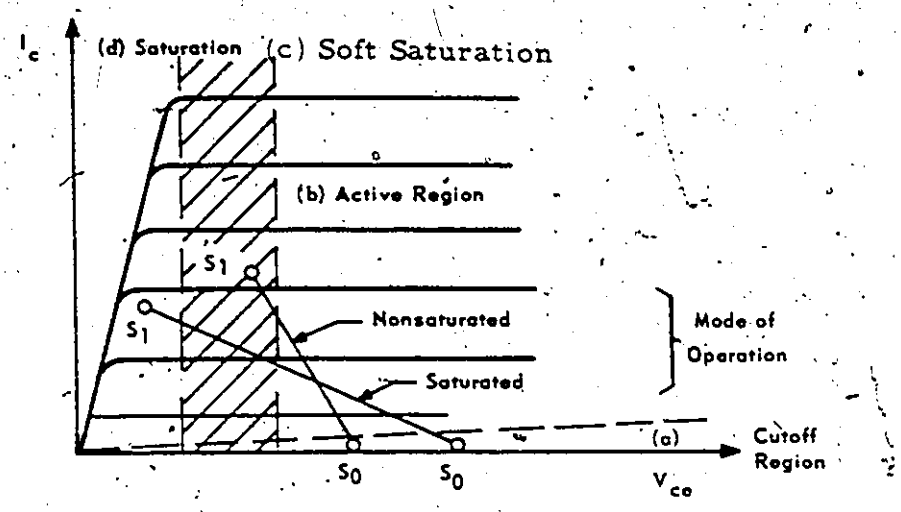


Fig. 4.3 Saturated and Nonsaturated Mode of Operation

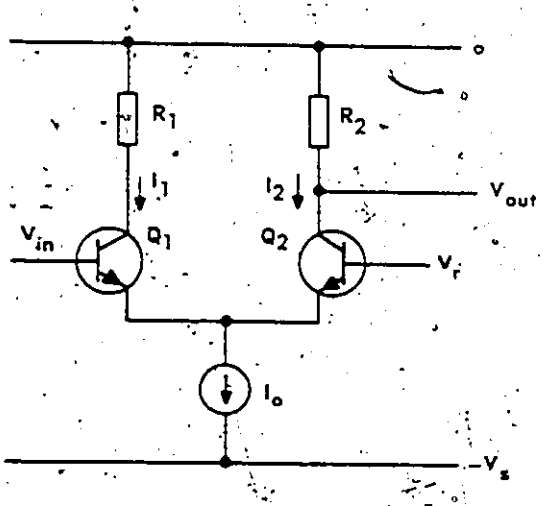


Fig. 4.4 Emitter Coupled Logic

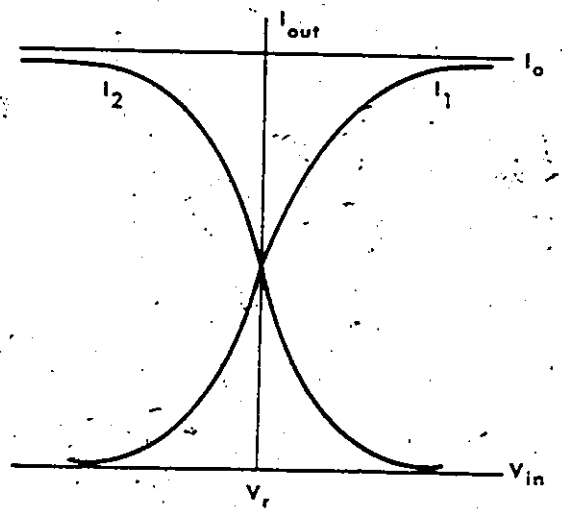


Fig. 4.5 I_{out} vs V_{in} for ECL

extra delay and power dissipation, in addition to complicating the circuit and using extra silicon area and it is advantageous to eliminate it. Another element which introduces delays in the circuit is the feedback capacitance between the collector of Q_1 and its base. As shown in the next section this delay can be eliminated by using a single load R_2 and connecting the collector of Q_1 to ground.

4.3 MULTI-EMITTER TWO-LEVEL LOGIC STRUCTURES

In the above section, we have indicated that the non-saturated Emitter-Coupled-Logic (ECL) circuit structures offer the possibility of operating with subnanosecond delays. These delays are achieved with relatively high power dissipation per logic function performed [14]. In order to adopt ECL for large-scale-integration it is necessary to reduce the total power dissipation on the chip as well as to increase the number of logic functions performed. This approach has led to the developement of the Multi-Emitter Two-Level Logic (METLL), a circuit structure based on ECL and suitable for LSI.

4.3.1 Repartitioning ECL To METLL

The following explains the development of METLL structures. Figure 4.6 shows the basic ECL gate, performing the OR and the NOR functions in positive logic. The circuit has a current source I_0 and a reference transistor Q_r with reference bias voltage V_r. The input variables are applied to the bases of Q_1 to Q_m. An emitter follower stage is usually required at the output for dc and transient coupling. Since the inverted output $\overline{F}$ is usually delayed more than the non inverted F due to base width modulation and collector feed back capacitance, only one output is used in the ECL circuit of Fig. 4.7. Another level of logic can be achieved by

replacing Q_r with multi-emitter transistor as shown in Fig. 4.8. The function realized in this case is the product of sums. By adding a second level of current steering devices, the inversion function can be realized along with three levels of logic, as shown in Fig. 4.9. The structure of Fig. 4.9 has two fixed biased transistors Q_{r1} and Q_{r2} with reference voltage V_{r1} and V_{r2} respectively. Q_{r1} is a multi-emitter transistor (only two emitters are shown in Fig. 4.9) where each emitter is coupled to the emitters of parallel input transistors (Q_{z1} to $Q_{z\ell}$ and Q_{y1} to Q_{ym} in Fig 4.9) and also to the collector of the reference transistor Q_{r2} or the collectors of the lower level transistors (Q_{x1} to Q_{xn}). The function performed in Fig. 4.9 is

$$F = XY + \bar{X}Z$$

where $X = x_1 + x_2 + \ldots + x_n$

$Y = y_1 + y_2 + \ldots + y_m$

$Z = z_1 + z_2 + \ldots + z_\ell$

Since the upper level input transistors (Q_{y1} to Q_{ym} and Q_{z1} to $Q_{z\ell}$) have a common collector, they can be associated with the output in a multi-emitter transistor form. In this case the structure reduces to the partitioned form of Fig. 4.10. This is the simplest form of the METLL structures with two input emitters and one base input at the lower level. The result of this partition is a saving of silicon area in LSI realization.

Moreover, if V_{r2} is a junction voltage (V_j) negative of V_{r1}, the logic levels of the output are compatible with the thresholds of both upper and lower level inputs. This is demonstrated in Fig. 4.11 and 4.12, where a logic realization is shown for functions F_3 and F_4 both using Two-Level Multi-Emitter ECL and METLL. It is seen that a saving

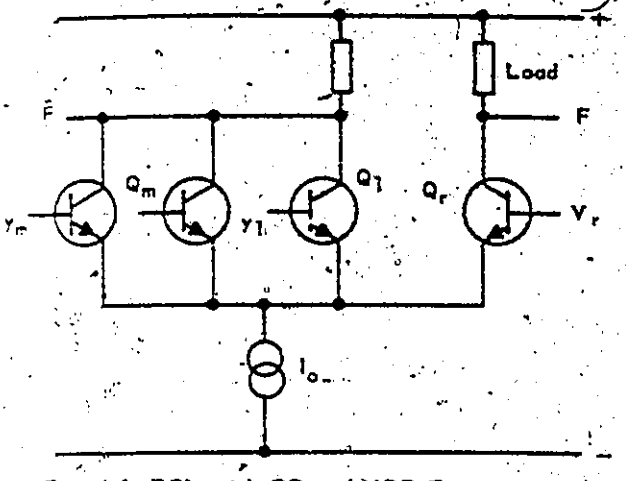


Fig. 4.6 ECL with OR and NOR Functions

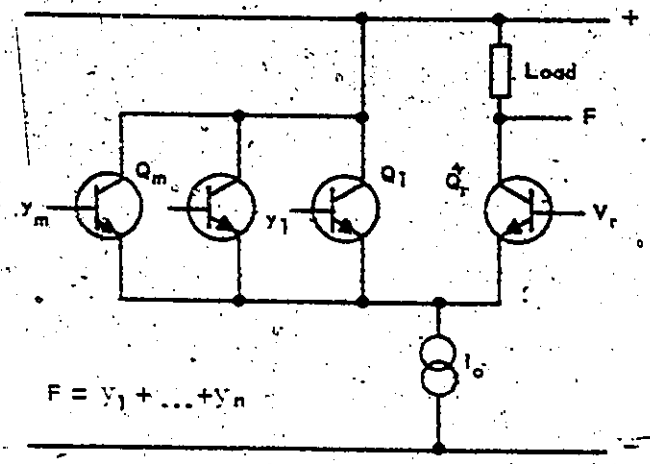


Fig. 4.7 ECL with Single Load-OR Function

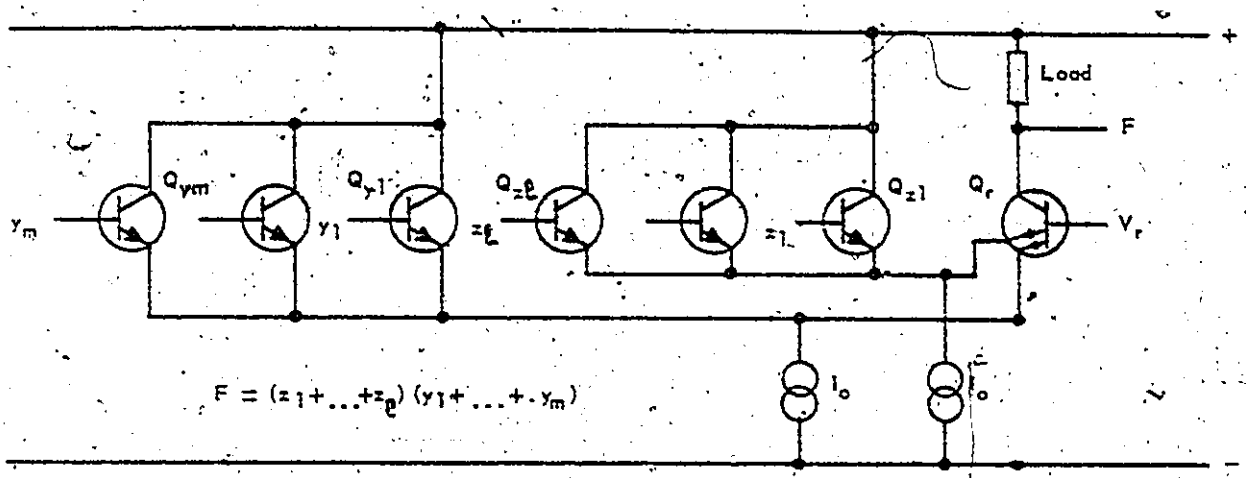


Fig. 4.8 Multi-Emitter ECL-OR/AND Function

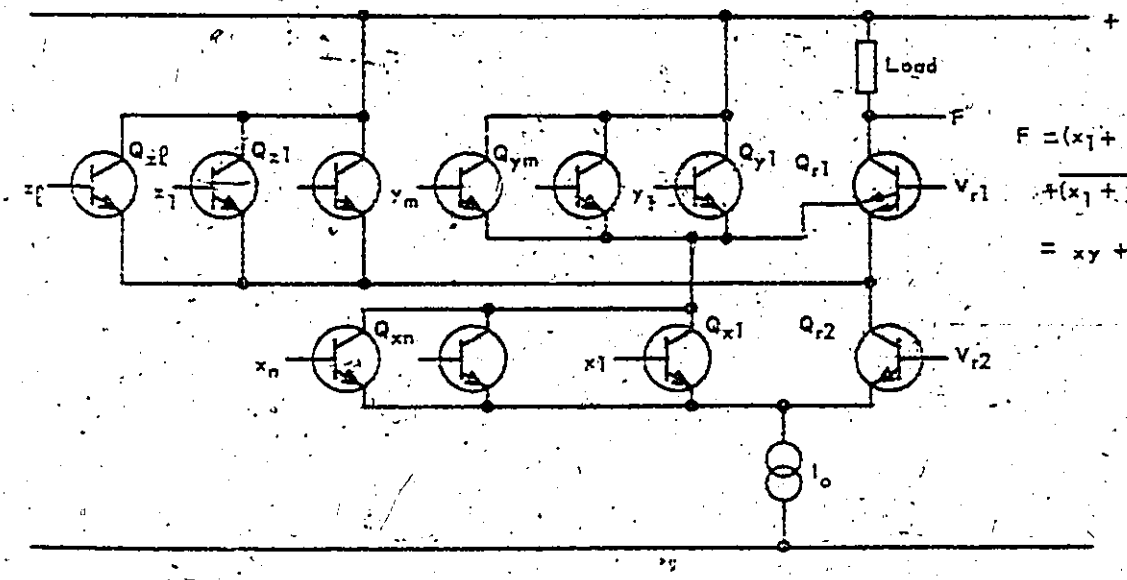


Fig. 4.9 Two-Level Multi-Emitter ECL-OR/AND/OR Function

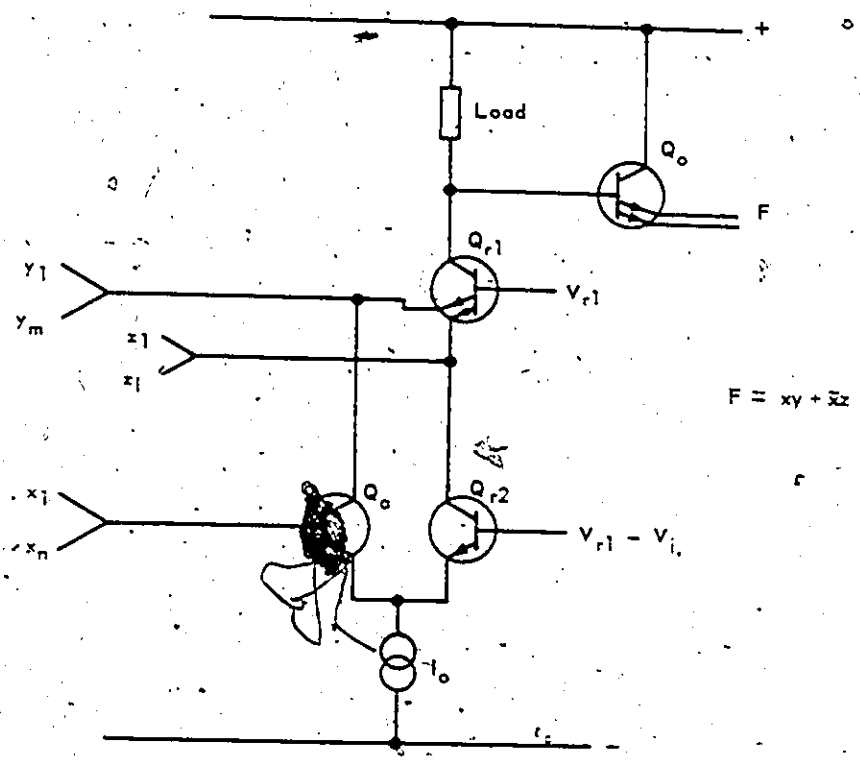


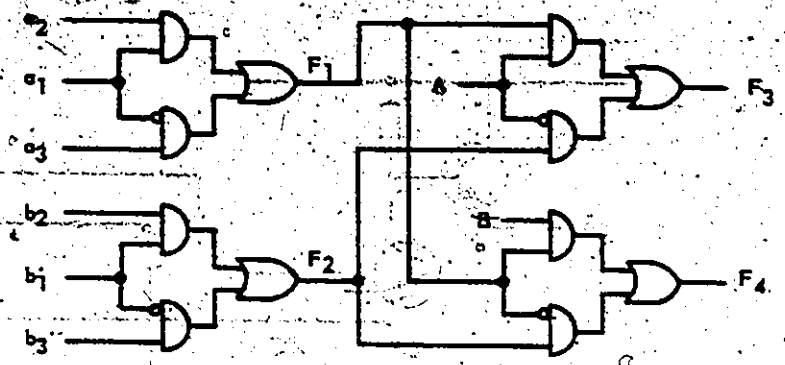
Fig. 4.10 Repartitioned Two-Level Multi-Emitter ECL
(Multi-Emitter Two-Level Logic = METLL)

$F_1 = a_1 a_2 + \bar{a}_1 a_3$

$F_2 = b_1 b_2 + \bar{b}_1 b_3$

$F_3 = A F_1 + \bar{A} F_2$

$F_4 = \bar{F}_1 B + \bar{F}_1 F_2$



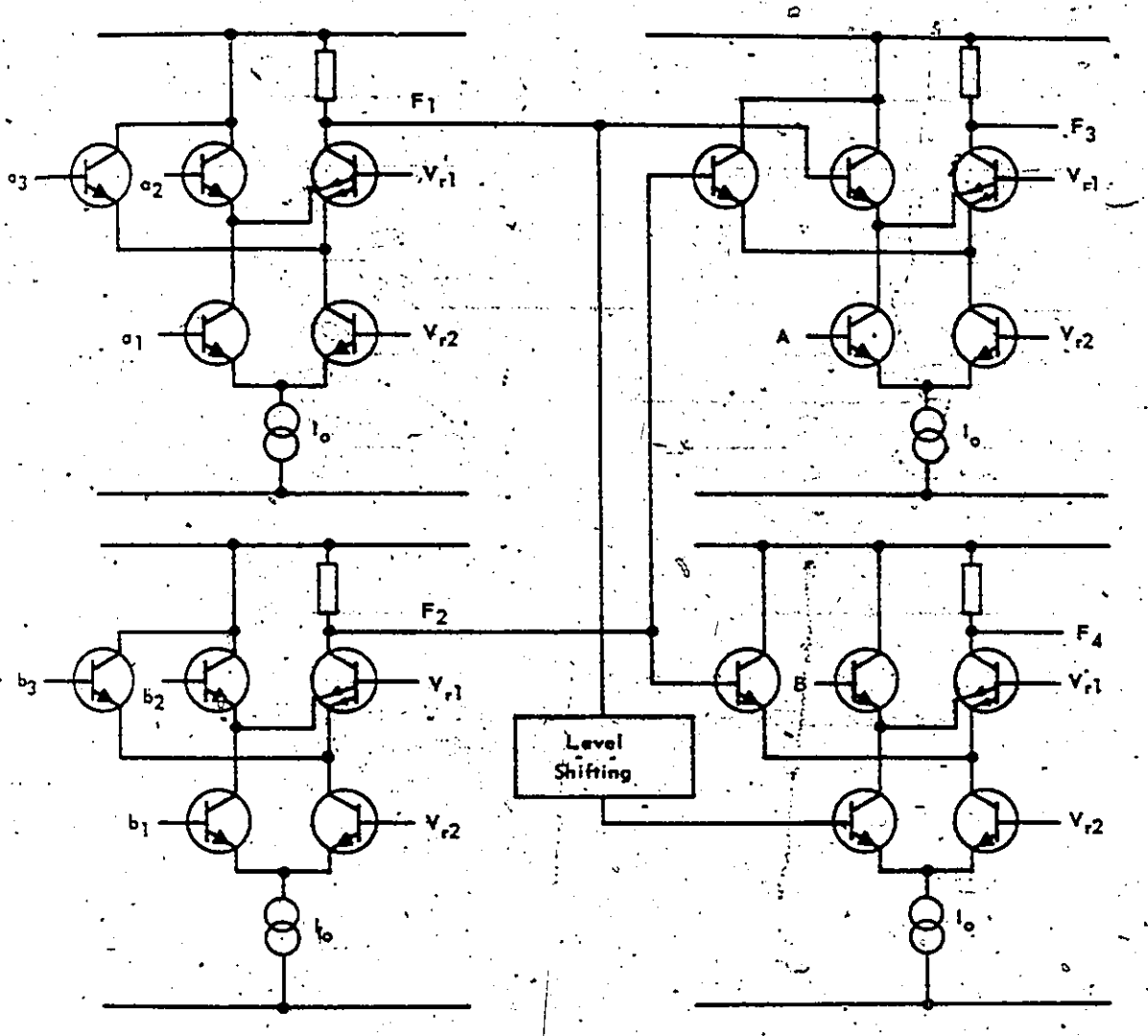


Fig. 4.11 Example 1, Circuit Realization Using Two-Level Multi-Emitter ECL, Number of Switching Transistors 20

of silicon area is possible and a level shifting stage is not required.

4.3.2 Logic Performed By METLL Structures

With reference to Fig. 4.10, the logic performed is $F = XY + \bar{X}Z$ where X, Y and Z are wired-OR functions as shown in Fig. 4.13. X will be referred as the control input and Y and Z as the direct inputs. The logic symbol of Fig. 4.14 is chosen for convenience where each output emitter is represented by a separate arrow ">", indicating that the fan-out number is equal to the number of output emitters.

Although the structure can be used to realize a NOR, OR or AND gate, as shown in Fig. 4.15, and conventional logic design techniques can be adopted, this is the least economical way to use the structure. Alternatively, modular logic design and logic partition techniques have been developed and will be discussed in Chapter 5.

Similar silicon area and power dissipation can be utilized to realize more complex logic functions, by adding emitters to the input transistor Q_{r1} and extra control inputs to the lower level. Figure 4.16 shows an (2n + 1) input METLL structure, where A_1 , ... A_n are control inputs and a_1 , ... a_n, a_{n+1} are direct inputs. The output function F is a logical '1', if both A_i and a_i are logical '1', i=1, ..., n or if all A_i are '0' and a_{n+1} is '1'. Normally only one control input (A_i) is allowed to be '1' at a given time and when more than one A_i is '1', the output must be a "don't-care" case. If F = '0' for the don't-care cases, F can be expressed as:

$$F = A_1a_1 + A_2a_2 + \ldots + A_na_n + \bar{A}_1\bar{A}_2 \ldots \bar{A}_na_{n+1}$$

where A_i and a_j, i = 1, .. n, j = 1, .. , n+1 are first-order wired-OR functions.

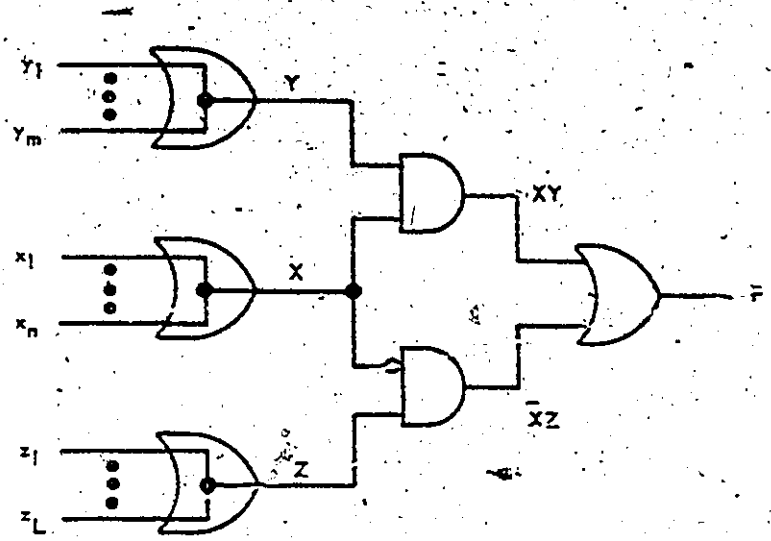


Fig. 4.13 Logic Performed By the Simplest Form of METLL

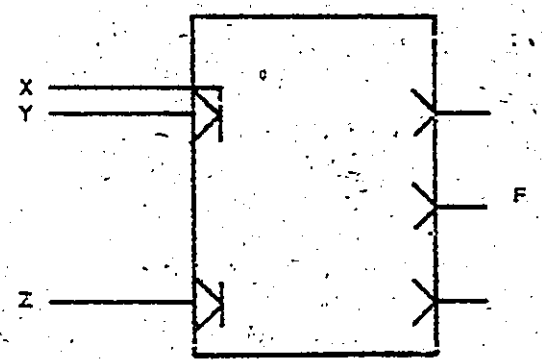
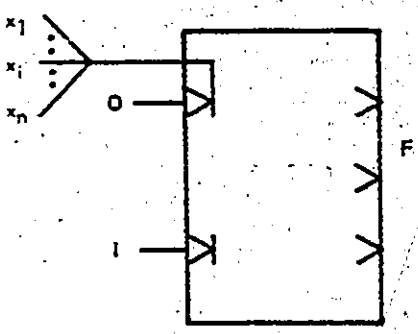
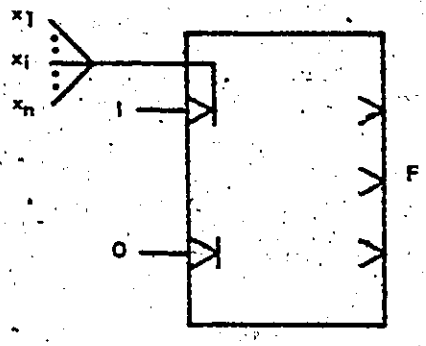


Fig. 4.14 Logic Symbol of the Structure of Fig. 4.13



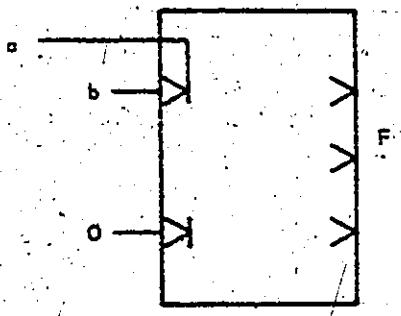
$F = \overline{x_1 + .. + x_n}$

(a) NOR Gate



$F = x_1 + x_2 + .. + x_n$

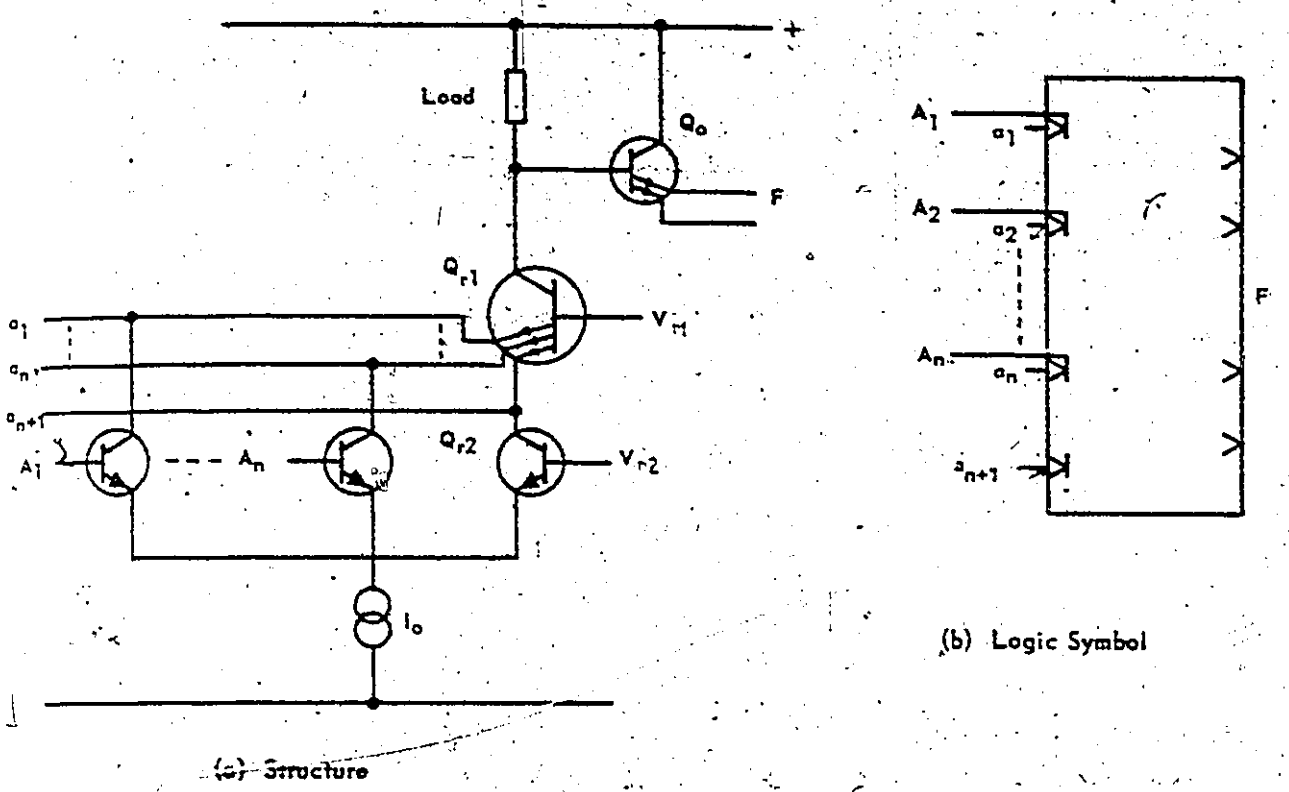
(b) OR Gate



$F = ab$

(c) AND Gate

Fig. 4.15 METLL Realization of Conventional Logic Gates



(a) Structure

(b) Logic Symbol

Fig. 4.16. Multi-Input METLL Structure

Inputs $A_1, \ldots, A_n$ are called 'control inputs' because they can be regarded as controlling the logic performed by the structure, while inputs $a_1, \ldots, a_{n+1}$ are called 'direct inputs'. At any time, the current through the load is either zero (F = '1') or the full current I_0 (F = '0'). If the load current is not zero or I_0, the output is undetermined and no input combination should allow this state to occur.

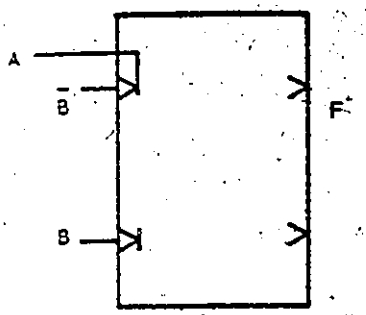
Figure 4.17 shows three examples to demonstrate the logic capability of METLL structures; Exclusive-OR, Equality and Full-Adder functions. A single structure can realize the Exclusive-OR or the Equality functions in a single level of logic. Conventional realization uses 3 gates (e.g. NAND gates) in two levels of logic. The example of the Full-Adder demonstrates this advantage further, where four METLL structures are used, compared with the conventional realization using a total of 9 gates for the sum and the carry.

4.3.3 Flip-Flop Realization

One of the useful properties of the METLL structure is its ability to perform flip-flop operations with minimum interconnections. This can be achieved in two basic configurations:

1) Configuration A, as shown in Fig. 4.18, where one of the output emitters, representing the output Q, is connected to the control input X. The logic symbol in Fig. 4.18(b) is used for convenience. The structure has similar characteristic to the RS flip-flop, as can be seen from the state table and state diagram shown in Fig. 4.18 (c). This realization compares favourably with the conventional cross-coupled 2 gate realization [5].
2) Configuration B, as shown in Fig. 4.19 , where one of the output emitters, representing the output Q, is connected to the direct input Z (or Y). As shown from Fig. 4.19(c), the structure has the characteristic of a data latch or D-type flip-flop, where the input X is the clock input. This METLL realization consumes less power and less silicon area than the conventional realization of 4 gates [5].

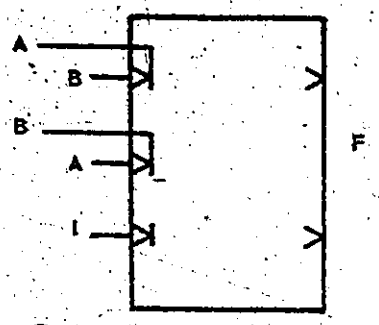
The above two configurations can be used to design more complex flip-flops. Two METLL structures are used to



$F = A \oplus B$

$= A\bar{B} + \bar{A}B$

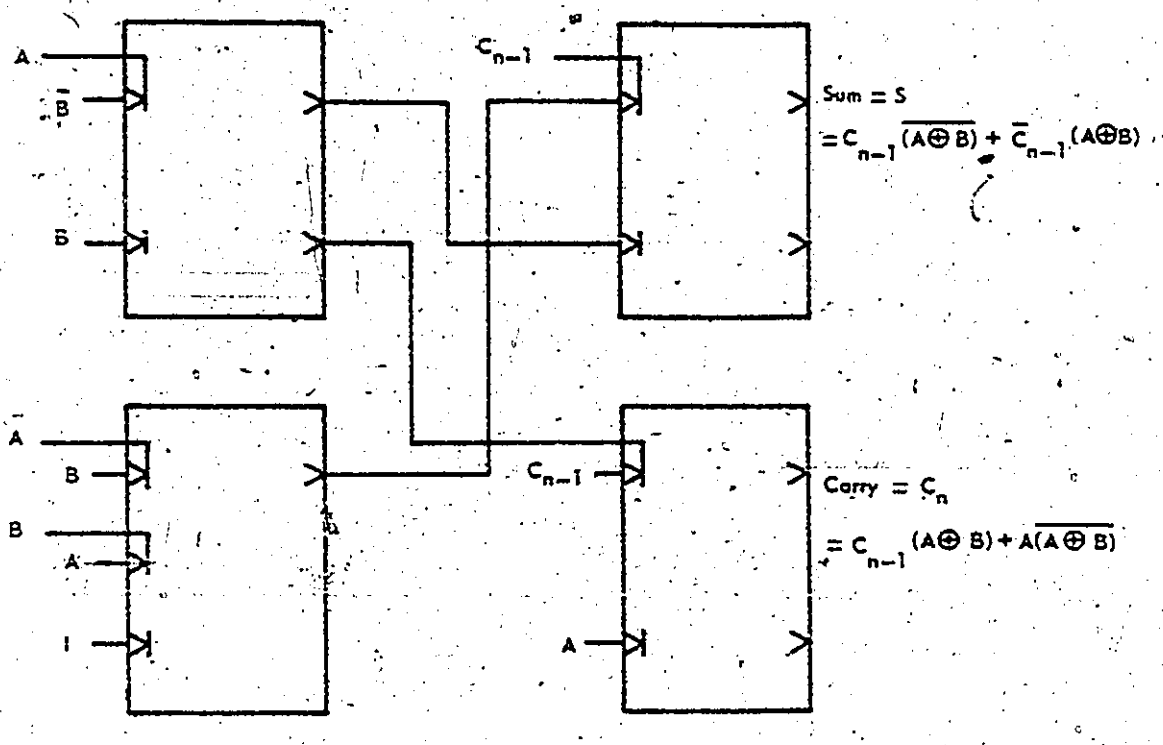
(a) Exclusive – OR Function



$F = \overline{A \oplus B}$

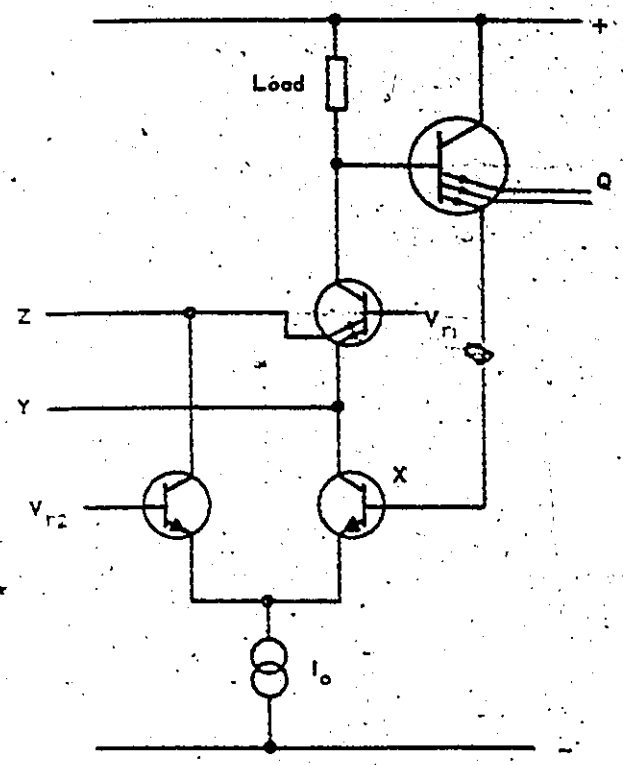
$= AB + \bar{A}\bar{B}$

(b) Equality Function

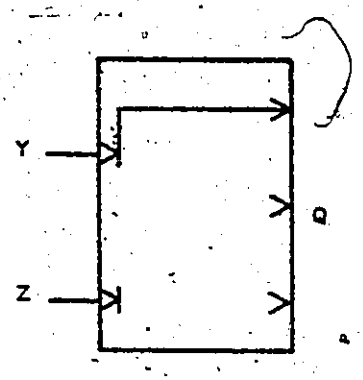


(c) Full Adder

Fig. 4.17 METLL Realization of Logic Functions



(a) Structure



(b) Logic Symbol

Y	Z	Q_t
0	0	0
1	1	1
1	0	Q_{t-1}
0	1	Indeterminate

(c) State Table and State Diagram

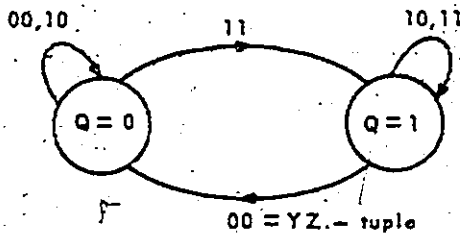
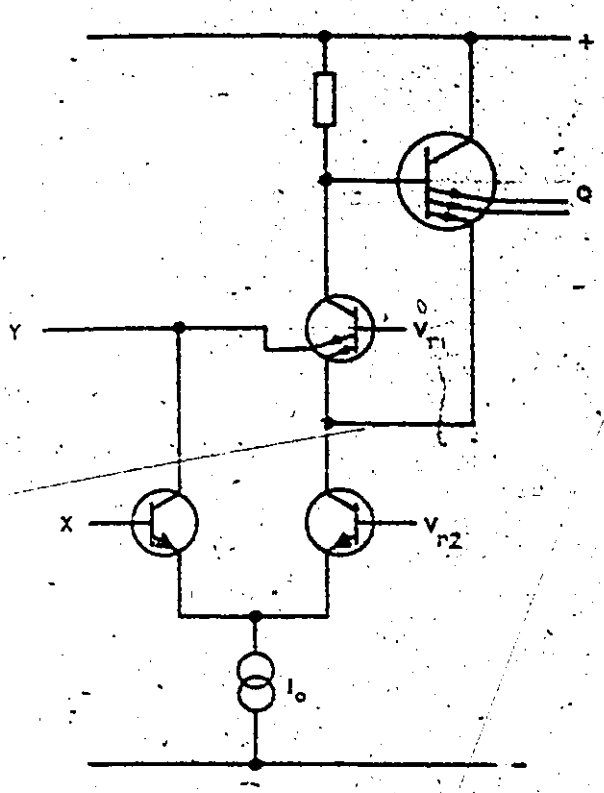
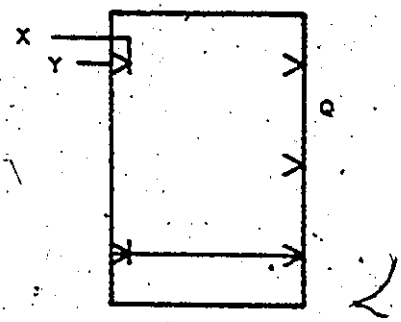


Fig. 4.18 METLL Flip-Flop, Configuration A (Analogous to RS Flip-Flop)



(a) Structure



(b) Logic Symbol

(c) State Table and State Diagram

X	Y	Q_t
0	0	Q_{t-1}
0	1	Q_{t-1}
1	0	0
1	1	1

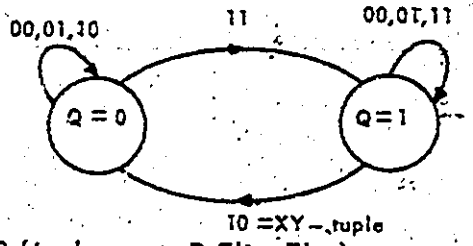


Fig. 4.19 METLL Flip–Flop, Configuration B (Analogous to D Flip–Flop)

realize a gated RS flip-flop, as shown in Fig. 4.20, as compared with 4 gates used in conventional realizations [5]. Figure 4.21 shows a master-slave realization of an RS flip-flop. Two METLL structures are used to realize a master slave D flip-flop, as shown in Fig. 4.22, as compared with 10 gates used in conventional realizations [5]. Three structures are used to realize a master slave JK flip-flop, as shown in Fig. 4.23, as compared with 9 gates used in conventional realizations. In realizing the T flip-flop of Fig. 4.24, three structures are used.

Although saving of silicon area and power consumption is possible for all commonly used flip-flop configurations, the most significant saving is for the master-slave-D-type flip-flop. For this reason it is used in preference to the others for the realization of functional blocks, as shown in Chapter 6. Table 4.1 gives the number of METLL and NAND gates for logic gates and flip-flops.

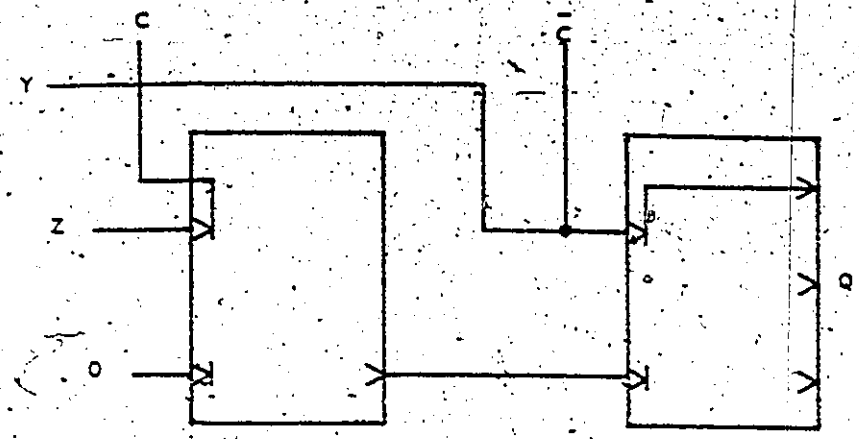
4.3.4 Circuit Characteristics

If a positive power supply system is assumed, Fig. 4.25 shows the voltage levels and the resulting noise margins. A 0.7 v is assumed across the current source where the current source is a transistor or a diffused resistor. The reference voltage V_{r2} at the lower level transistor Q_{r2} is 1.4 v and $V_{r1} = V_{r2} + 0.7 = 2.1$ v. If the base-collector junction of Q_{r1} is allowed to be forward biased by a maximum of $V_j/2$ and if the logic swing at the load is assumed to be V_j then the minimum power supply is $(V_{r1} + V_j/2)$ i.e. 2.45 v. Fig. 4.25 (b) shows the logic levels and the noise margins of such a circuit. A logic swing of V_j is obtained and the noise margins are equal and is $V_j/2$.

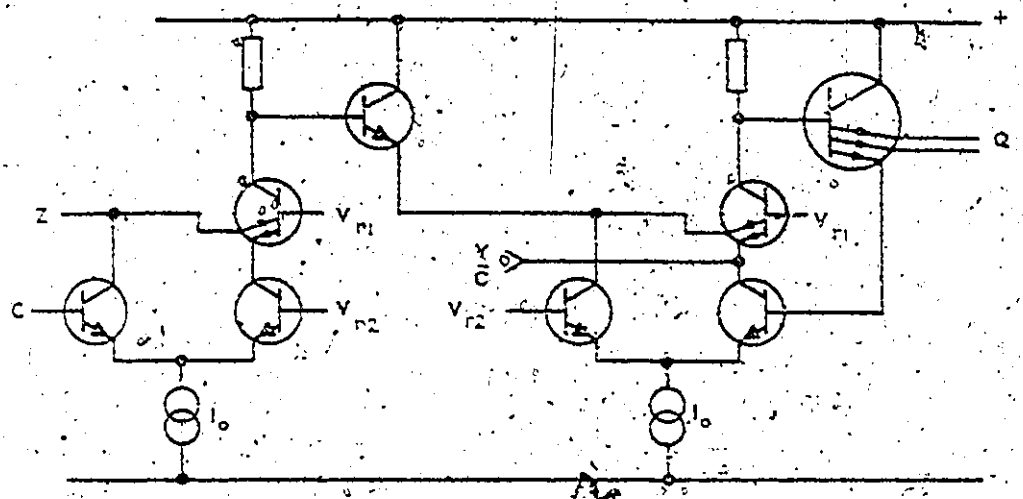
Table 4.1

METLL-NAND Realization of Logic Gates and Flip-Flops

Function	Number of METLL Structures	Number of NAND Gates
Exclusive-OR	1	3
Equality	1	3
Full-Adder	4	9
RS Flip-Flop	1	2
D-Type Latch	1	4
Gated RS Flip-Flop	2	4
Master-Slave D-type	2	10
Master-Slave JK	3	9



(a) Logic Symbol



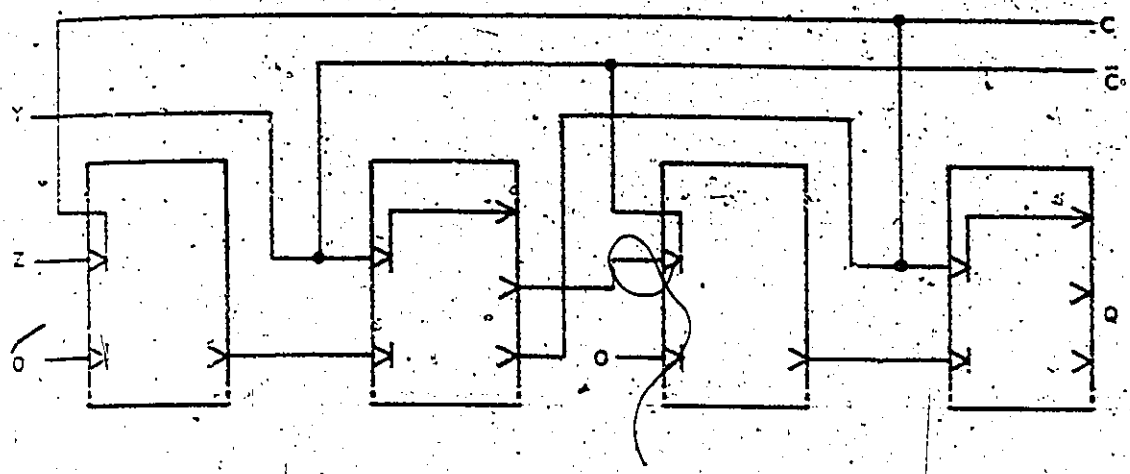
(b) Structure

(c) State Table

C	Y	Z	Q_t
H	0	0	0
	1	1	1
L	1	0	Q_{t-1}
	0	1	Ind.
H	ϕ	ϕ	Q_{t-1}
L			

ϕ = Don't Care

Fig. 4.20 METLL Gated R/S Flip-Flop

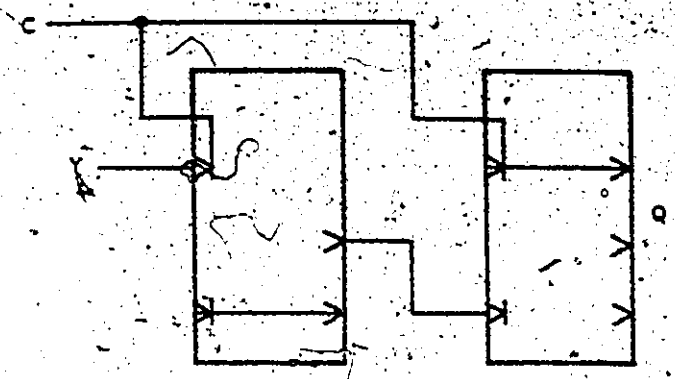


(a) Logic Symbol

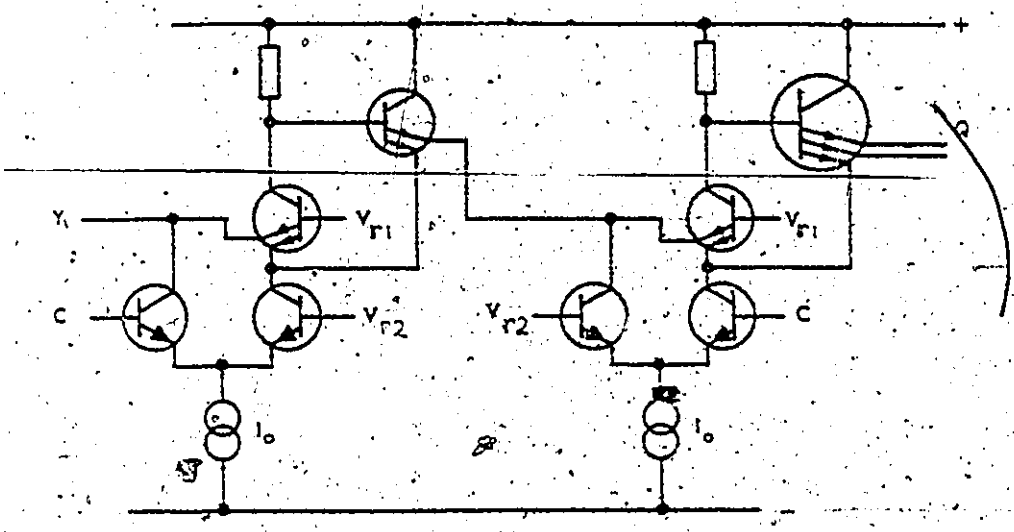
(b) State Table

C	Y	Z	Q_t
H	ϕ	ϕ	Q_{t-1}
L			
H	0	0	0
	1	1	1
	1	0	Q_{t-1}
L	0	1	Ind.

Fig. 4.21 METLL Master–Slave RS Flip–Flop



(a) Logic Symbol

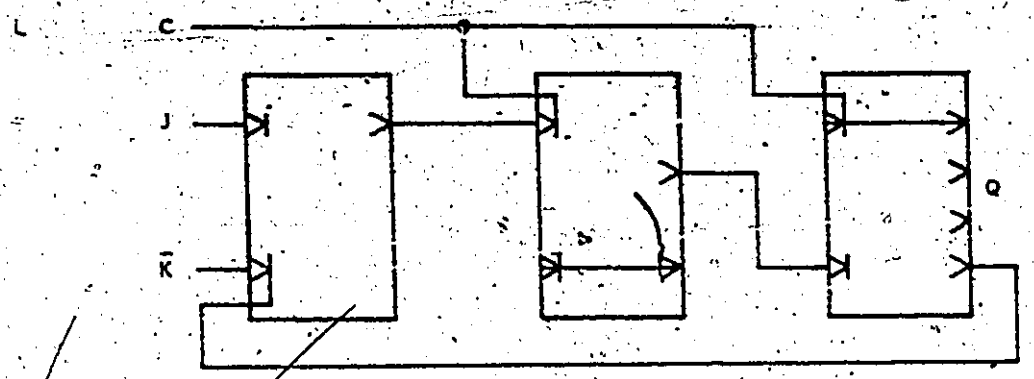


(b) Structure

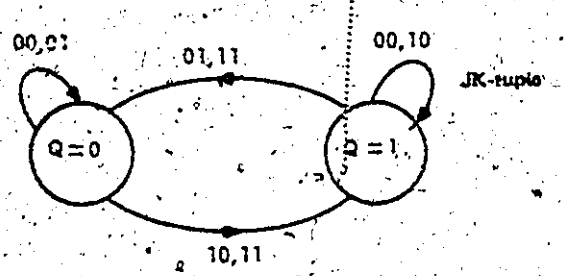
C	Y	Q_t
H → L	Φ	Q_{t-1}
	+	
≠H → L	0	0
	1	1

(c) State Table

Fig. 4.22 METLL Master–Slave D Flip–Flop



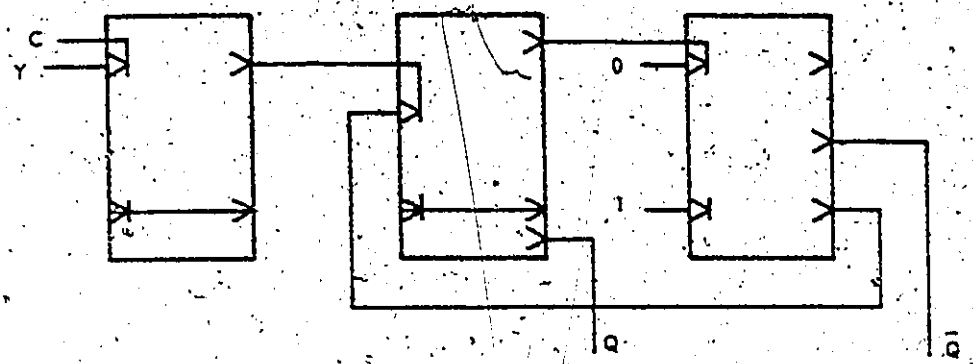
(a) Logic



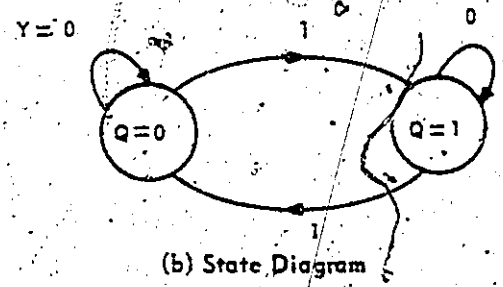
C	J	K	Q_t
H	ϕ	ϕ	Q_{t-1}
L			
H	0	0	Q_{t-1}
	0	1	0
	1	0	1
L	1	1	$\bar{Q}_{t-1}$

(b) State Diagram and State Table

Fig. 4.23 METLL Master–Slave JK Flip–Flop

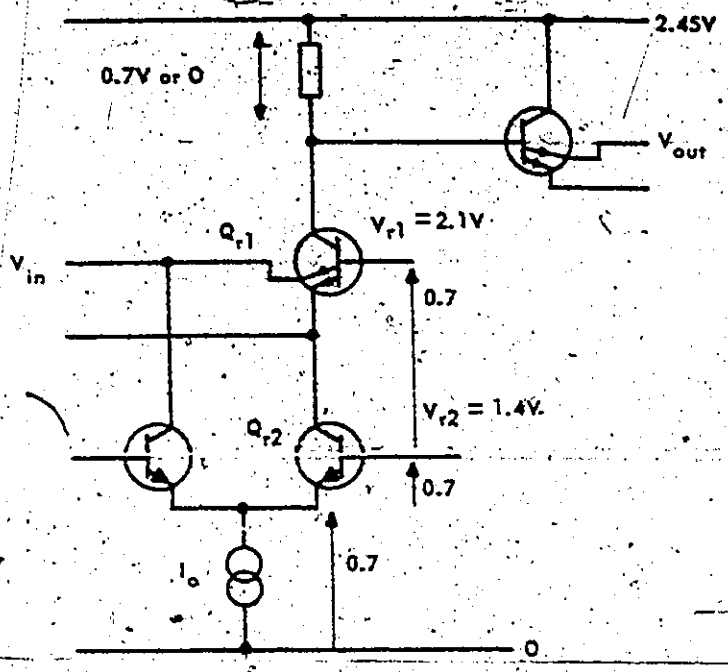


(a) Logic

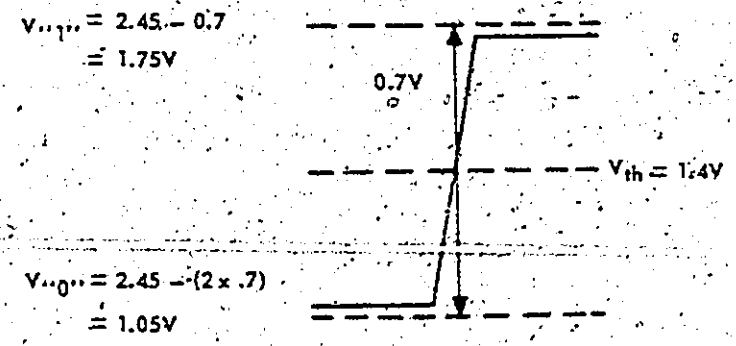


(b) State Diagram

Fig. 4.24 METLL T Flip-Flop



(a)



(b)

Fig. 4.25 Logic Levels and Noise Margins of METLL With Positive Power Supply

Although the positive power supply system can always be used in an overall system with METLL as its building block, the system does not provide the flexibility of interfacing with other ECL high-speed logic families e.g. MECL 10,000 [15]. If such interfacing is required, a negative power supply system can be used with compatible logic levels as shown in Fig. 4.26.

It was indicated above that a logic swing of V_j is assumed across the load. Many load structures can provide such a swing e.g. a resistor with a clamping diode. The problem of choosing the load structure is considered in greater detail in Appendix 1 and in [16] where we conclude that a clamped-type load structure gives the optimum dc and transient behavior. The temperature compensation of logic levels of such loads is treated in [23] where it is shown that it is simpler to compensate for logic level variations as compared with the case of conventional diffused resistor loads.

The basic METLL structure can be designed over a wide range of speed and power. The main power determining factors are the supply voltage and the current used, which is determined by the biasing circuit. On the lower power side the feasibility limit is the high value of resistors required in the biasing circuit and in the load structure. On the high power side, devices with high f_τ must be used and the number of structures on a chip will be limited by the total power dissipation on the chip. A range of circuits from (15 nsec delay, 100 μW dissipation and 1.5 pJ power-delay product) to (1 nsec delay, 5 mW dissipation, 5 pJ power-delay product) is considered practical.

For function verification of the circuit configurations shown in sections 4.3.2, 4.3.3 and 4.3.4 a logic module

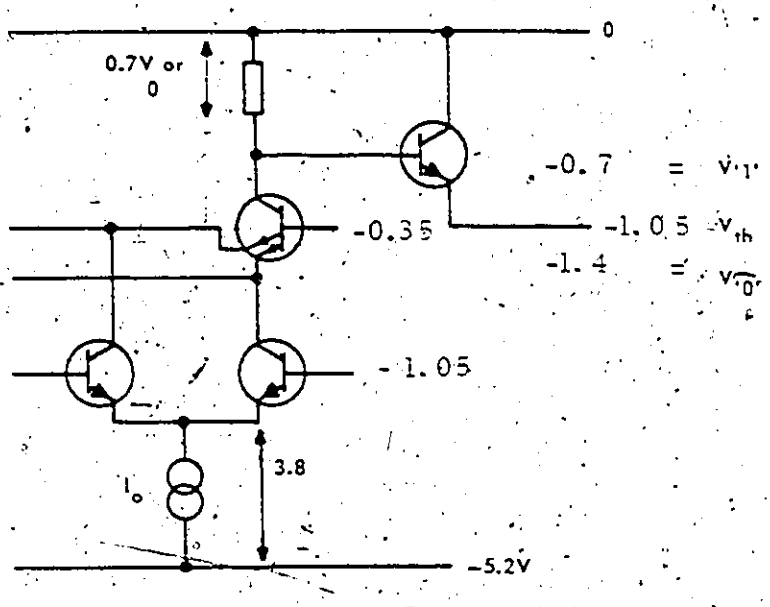


Fig. 4.26 METLL with Negative Power Supply System

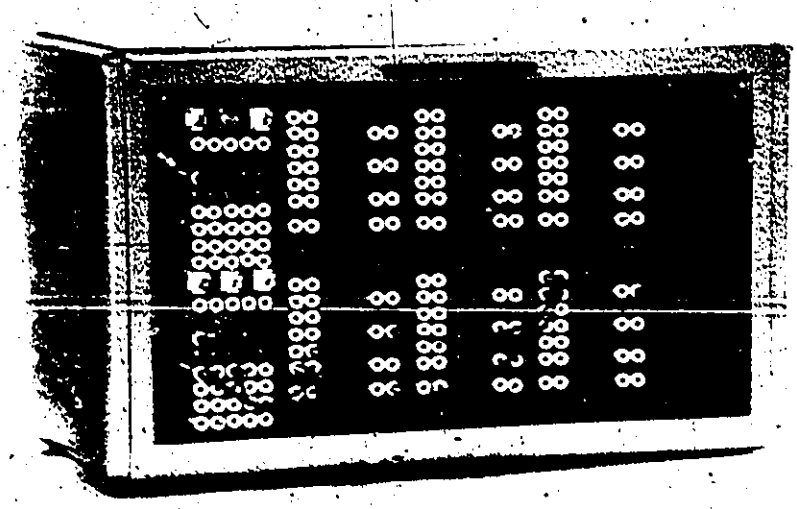
was built from discrete components (Fig. 4.27.a).

Figure 4.27.b shows the METLL test circuit with 4-input emitters and 4-output emitters at Q_{r1} and Q_o which are realized from discrete transistors with f_τ of 300 MHz. An input interface circuit is also shown to interface with 50 Ω output impedence pulse generators. An output interface circuit is used to interface with the measuring equipment e.g. scope. Figure 4.27.c shows the biasing circuit used to provide the two reference voltages V_{r1} and V_{r2}. The resistance divider R_2, R_3 provides V_{r2} and the diode D gives $V_{r1} = V_{r2} + V_j$. The transistors pairs Q_1 and Q_2 along with R_4, R_5, R_1 and the feed-back transistor Q stabilize any changes in V_{r1} and V_{r2}. Figure 4.27.d shows the transfer characteristic (f_1 vs Y_1) measured at room temperature, of the input at Y_1 (Y_2, Y_3, X_2, X_3; Z are at '0' and X_1 at '1') and the output at f_1, the transfer characteristic (f_1 vs X_1) of the input at X_1 (Y_1 at '1' and X_2, X_3, Y_2, Y_3, Z are at '0') and the output at f_1. In both cases adequate noise margins of at least 300 mV are demonstrated.

Hybrid models of a single structure were built from discrete transistors of f_τ of 5 MHz and from integrated transistors of f_τ of 300 MHz (Fig. 4.28) in order to verify logic levels, power supply requirements and power dissipation of the structures shown in Fig. 4.25 and Fig. 4.26.

Figure 4.29 shows the realization of divide-by-two stage using hybrid models of METLL structures. A logic swing of 750 mV is available at the output and correct master-slave operation is achieved.

Finally a master-slave D-type flip-flop was integrated [1] using diffused bipolar technology with transistor f_τ of



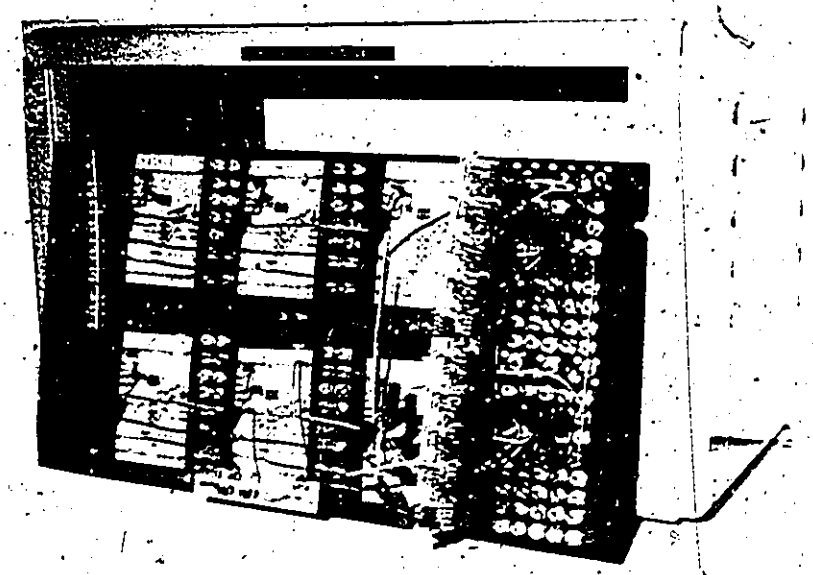
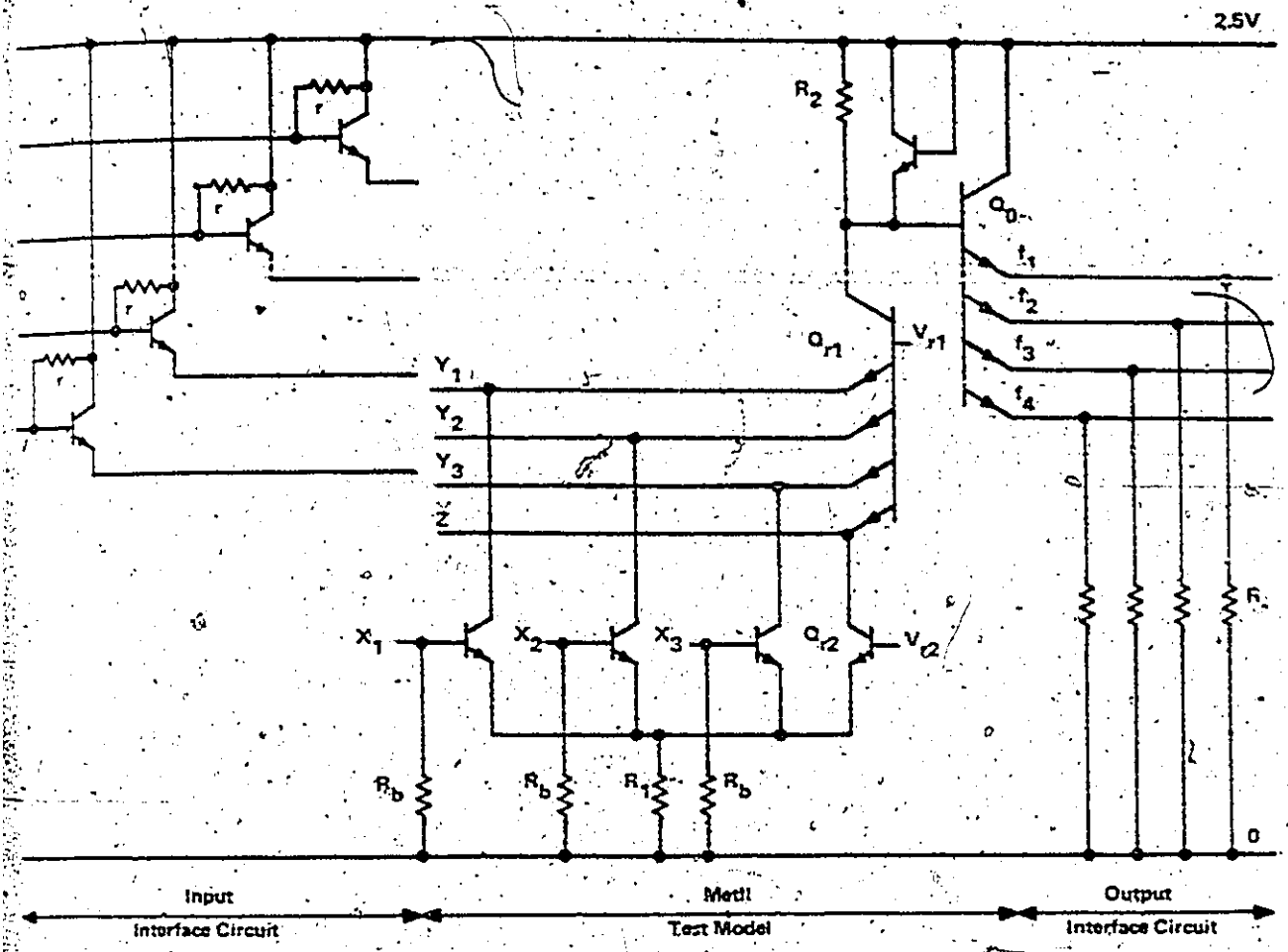
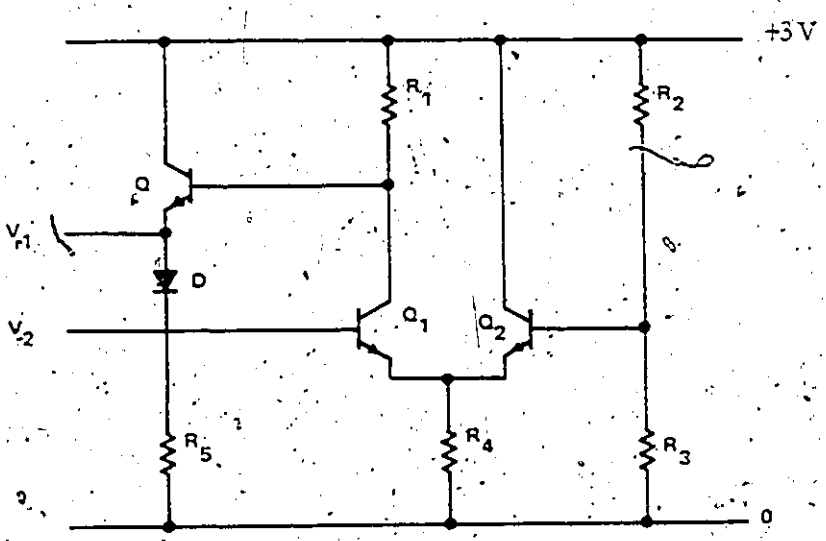


Fig. 4.27.a METLL Logic Modules. It consist of 6 Structures (each with 4-Input Emitters, 3-Input Bases and 4-Output Emitters) and 10 Input-Interface-Circuits and 6 Output-Interface-Circuits.



R_1 = 0.35K
R_2 = 0.17K
R = 50Ω
R_b = 1M
r = 50Ω
V_{r1} = 2.1V
V_{r2} = 1.4V

Fig. 4.27.b Test Circuit and Interface Circuits



R_1 = 0.2K
R_2 = 0.7K
R_3 = 0.5 K
R_4 = 0.35K
R_5 = 3K

Fig. 4.27.c Biasing Circuit Used

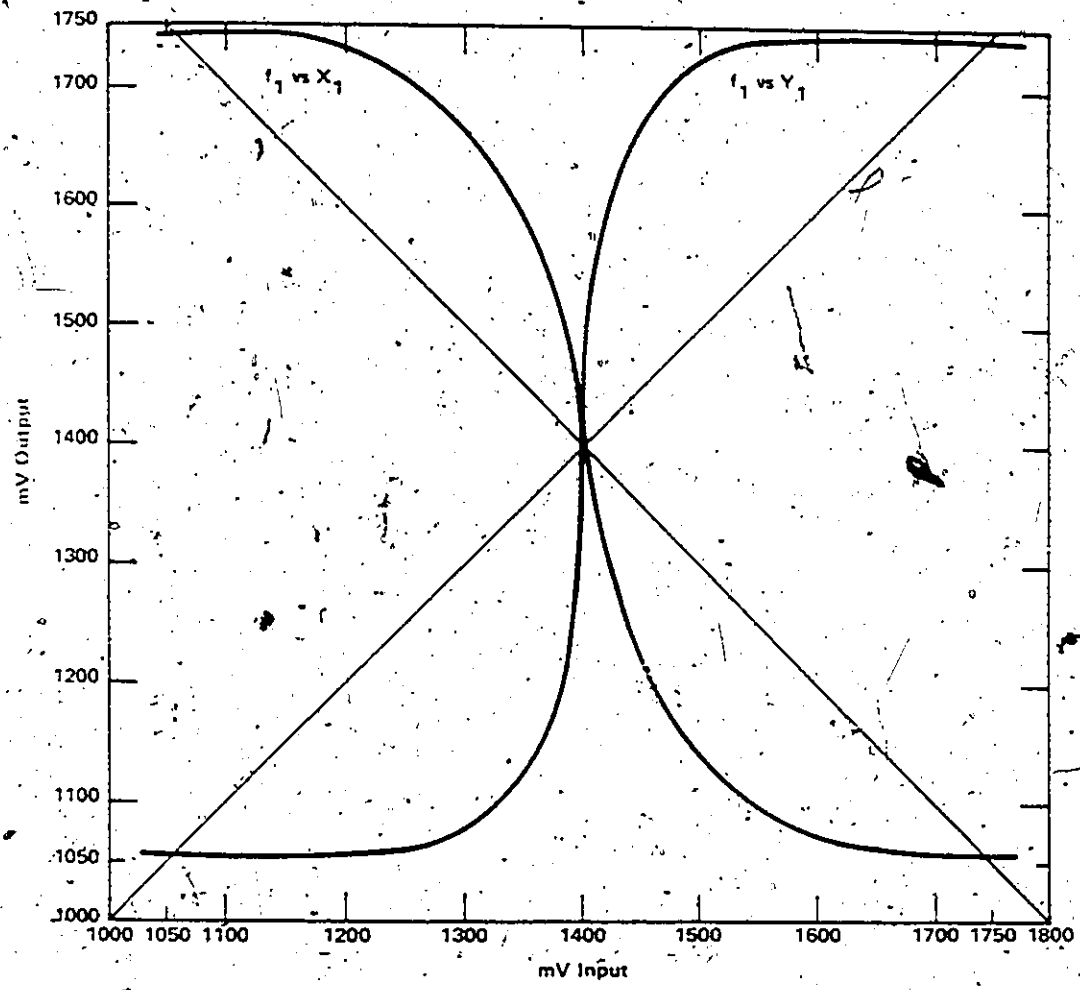


Fig. 4.27d Transfer Characteristics of METLL Test Circuit

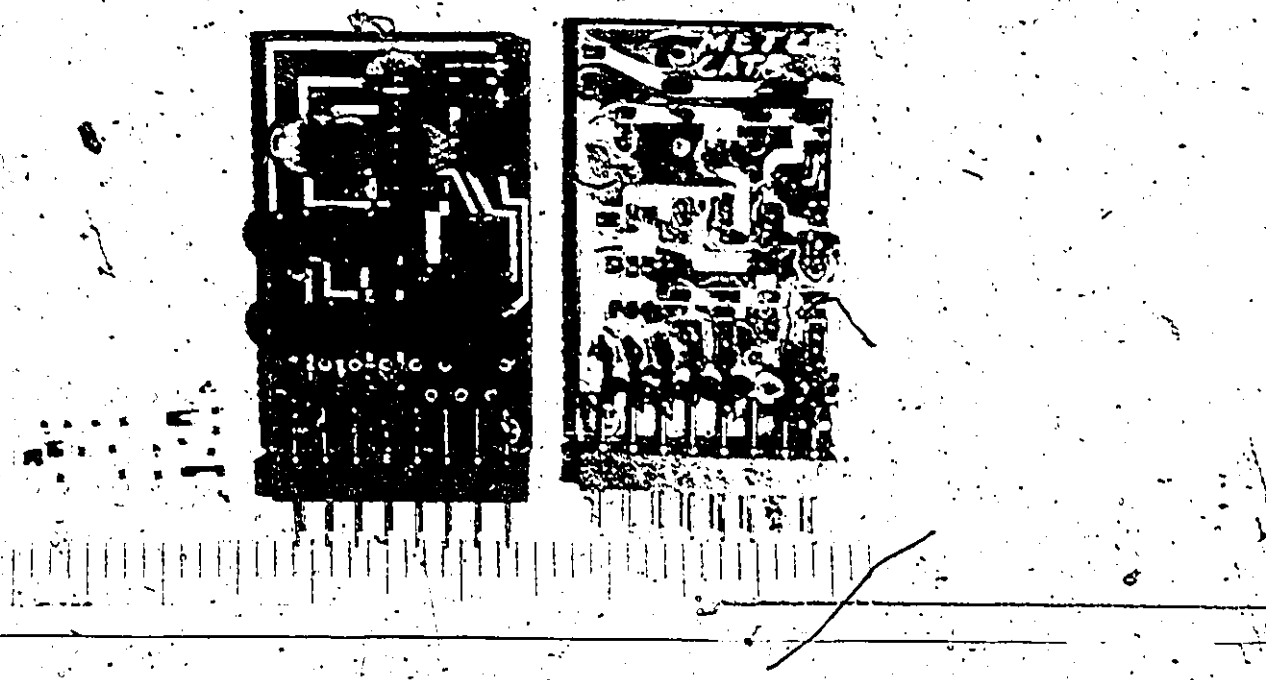
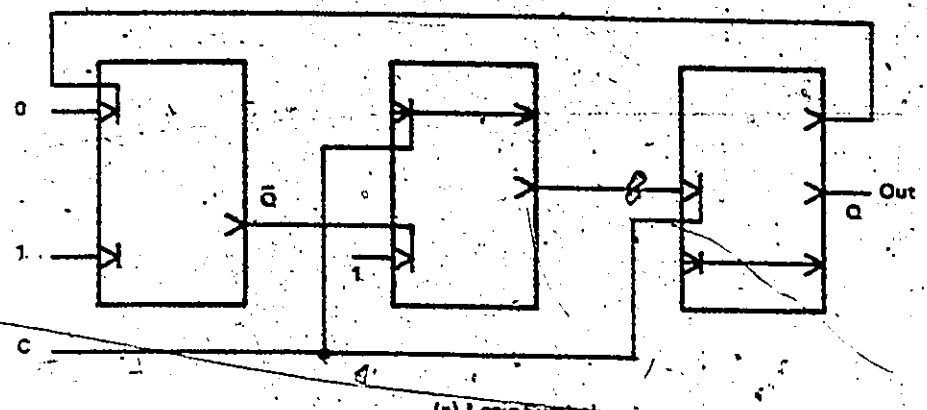
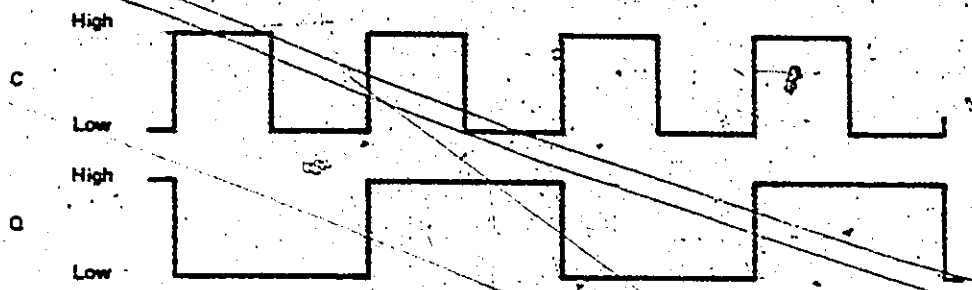


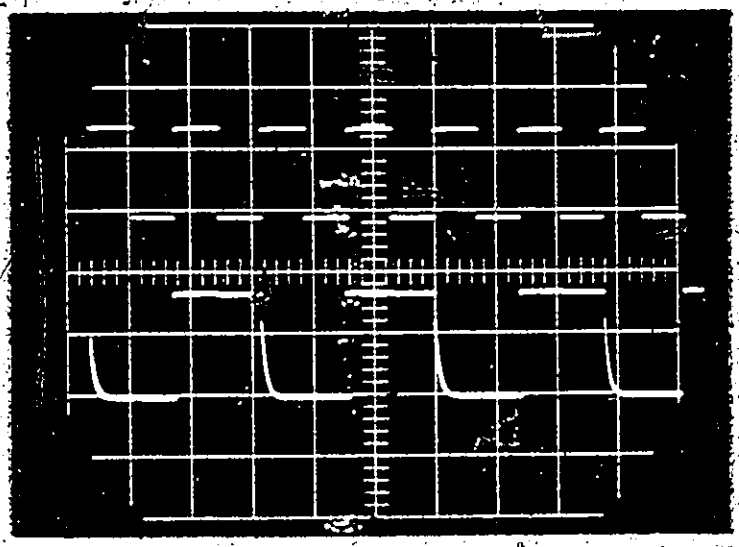
Fig. 4.28 Hybrid-MTTLL Models, Built from Descrete Devices (The Two to the Right) and from Integrated and Thick Film Devices (to the Left).



(a) Logic Symbol



(b) Timing Diagram



(c) Waveforms

Fig. 4.29. Divide-by-two METLL Using Integrated-Hybrid Realization

1.8 GHz. With a power dissipation of 16 mW it toggles at 500 MHz while the area occupied by the circuit is 400 X 400 μ. Figures 4.30-4.32 show the circuit, its photomicrograph and the clock/output waveforms. The layout was designed according to the standard bipolar design rules of Santa Clara Division of Hewlett Packard Company, California. In the layout of the clamped-diode-load-structure in association with the input and output multi-emitter transistors, only the input transistor Q_{n1} requires a separate isolation, the diode and the output transistor Q share an isolation common for the whole chip (Fig. 4.31.b and Fig. 4.31.c). The resistors are formed by a base diffusion.

4.3.5 Comparison Between METLL and High-Speed Logic Families

As indicated in sections 4.3.2 and 4.3.3 a single METLL structure can perform the equivalent of several basic Boolean gates. Figure 4.33 shows the power-delay product per basic Boolean gate against the number of Boolean functions realized for a structure with a 5pJ power-delay product.

In order to evaluate the potential of METLL, a comparison is made with other high speed logic families:

(1) Table 4.2 compares the delay and the power dissipation of a METLL structure packaged separately with separately packaged logic gates of other families.

(2) Tables 4.3 - 4.6 compares MSI implemented logic of some other families with the performance of MSI implemented METLL and, where not available, with multipackage METLL. The circuits include Exclusive-OR, Full-Adders and JK Flip-Flop.

(3) Finally, table 4.7 compares the performance of METLL LSI realization with existing LSI technologies, which include p-MOS, n-MOS, Dynamic n-MOS.

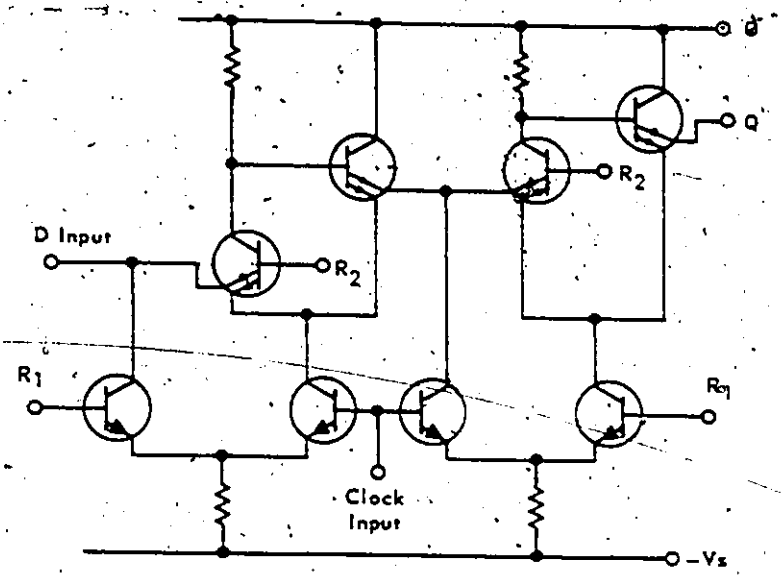


Fig. 4.30 Simplified METLL circuit for a master–slave D flip–flop

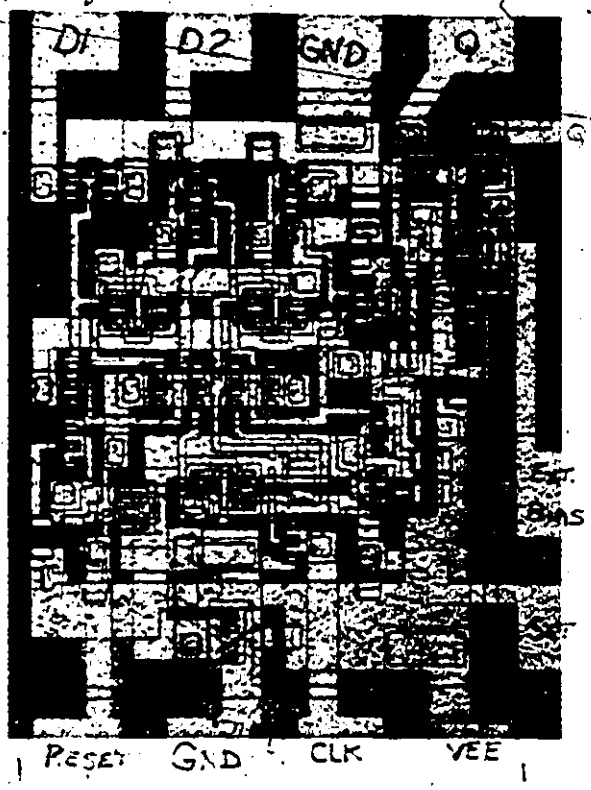
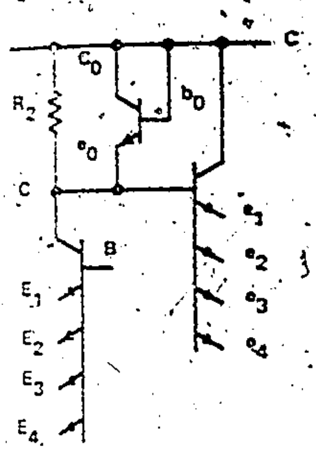


Fig. 4.31.a Photomicrograph of 500 MHz Master-Slave METLL D Flip-Flop



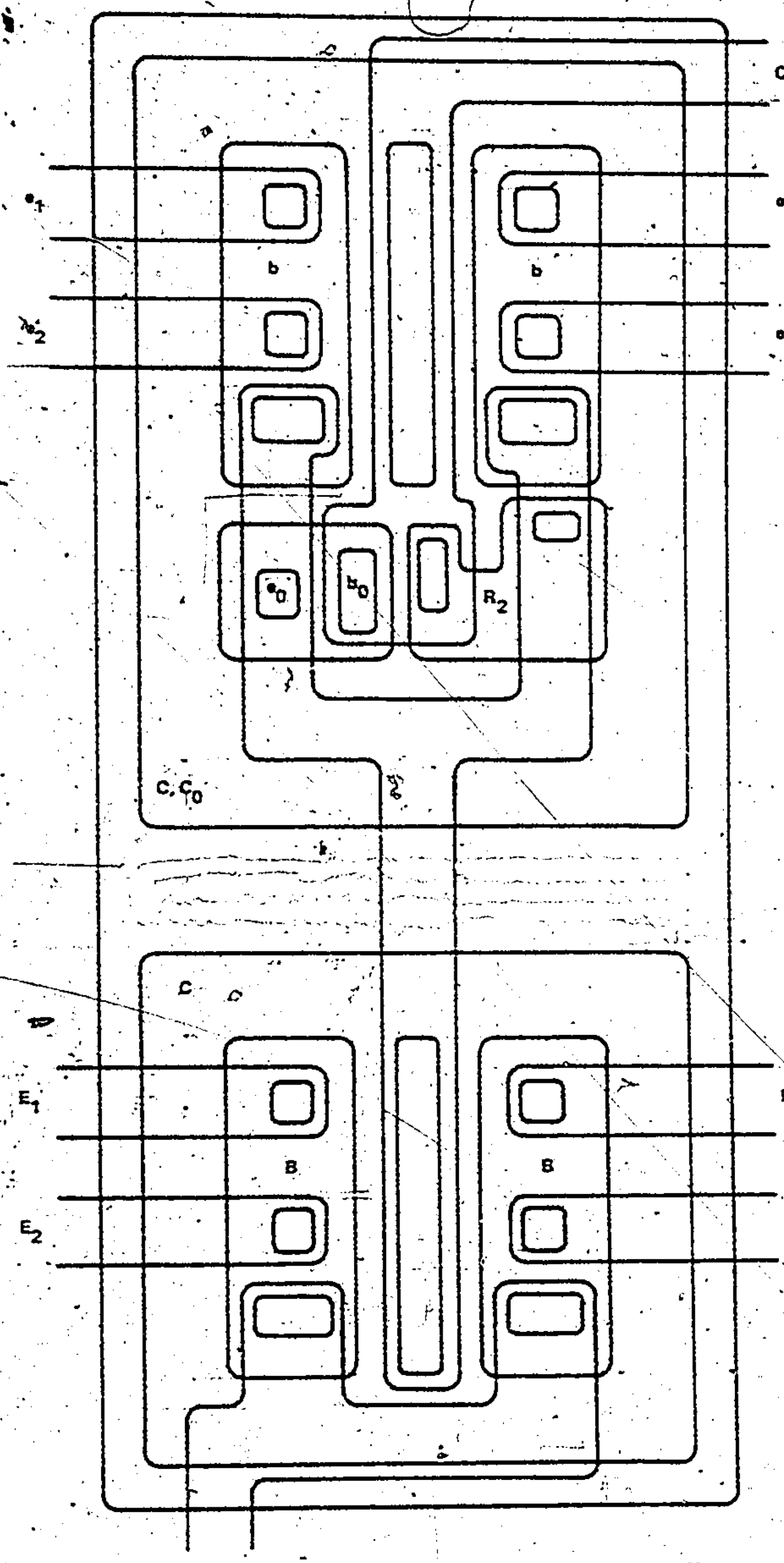


Fig. 4.31.b Layout of Input–Output–Transistors and Load Structure

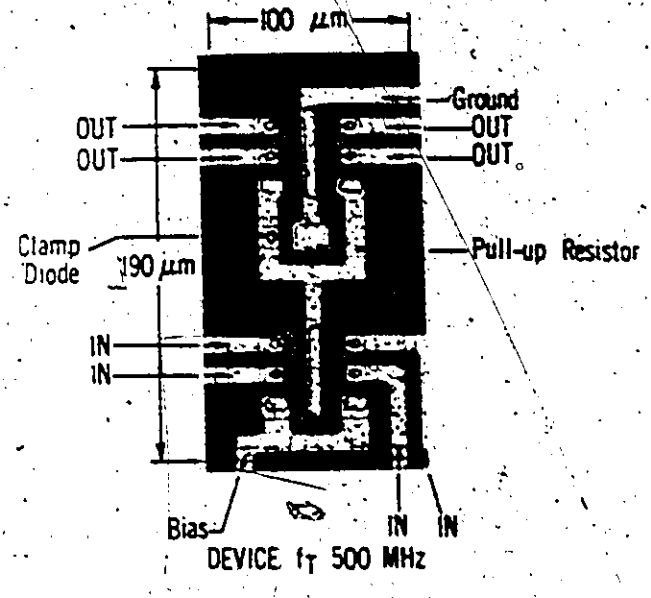


Fig. 4.31.c Photomicrograph of Fig. 4.31.b

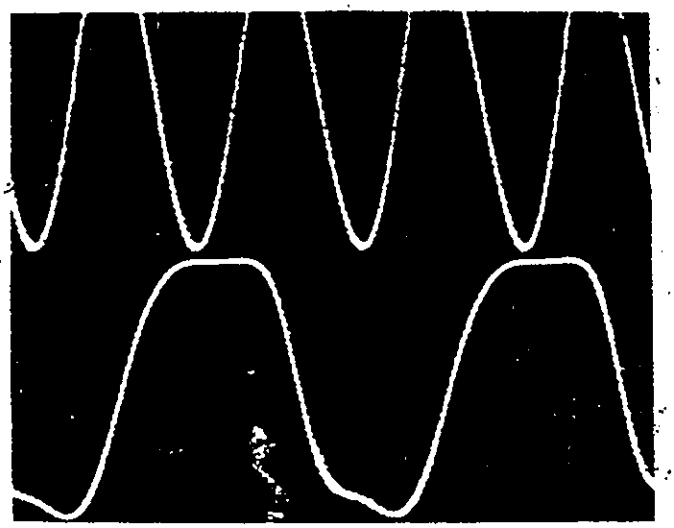


Fig. 4.32 Clock and Output Waveforms for 500 MHz D Flip-Flop

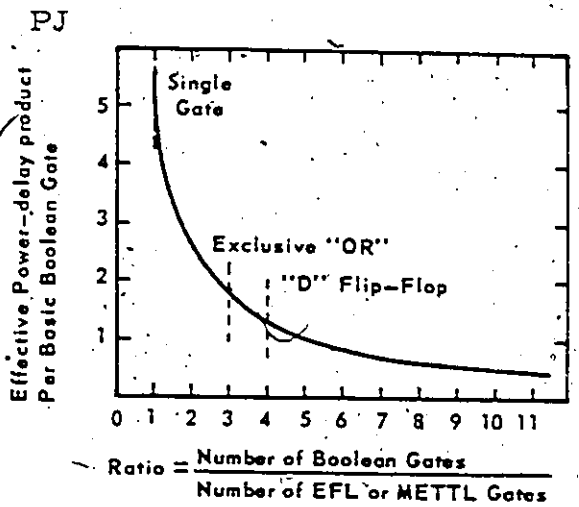


Fig. 4.33 The improvement of effective power-delay product with logic complexity for a 5pJ METLL gate

Table 4.2

Power-Speed Performance Comparison of Separately Packaged Gates

Logic Family	Source	Delay ns	Power mW	Power x Delay pJ	Comments
TTL 54L/74L	17	33	1	33	
TTL 54/74	17	10	10	100	
TTL 54 H/74 H	17	6	23	138	
TTL 54 S/74 S	17, 22	3	19	57	
ECL MECL II	18	4	40	160	
ECL MECL II 1/2	18	2	60	120	
ECL MECL III	18	1	60	60	
ECL MECL IV	18	1	15	15	
	18	0.5	50	25	
METLL	Measured	10	0.5	5	$R_1=R_2=$ 3.5K
Load:	Fig. 4.25	4	1.4	5.6	$R_1=R_2=$ 1K
a resistor (R_2)	with				
Current Source:	descrite				
a resistor (R_1)	transistors				
Power Supply:	(Philips				
2.5V	3F2N)				
Fan-out = 2	f_T= 300MHz				

Table 4.3
Power-Speed Performance Comparison of
MSI Logic (Product of Two Sums)

Logic Family	Source	Delay ns	Power mW	Power x Delay pJ	Comments
TTL Satur. MSI	21	6	6	36	
TTL Schottky MSI	22	2.25	13	29.2	
	22	2.2	9.4	20.7	
METLL Load: a resistor (R_2) Current Source: a resistor (R_1) Power Supply: 2.5V Fan-out = 2	Measured Fig. 4.25 with descrite transistors (NE 1352) f_T= 300 MHz	4.3	0.5	2.15	$R_1=R_2$= 3.5K

Table 4.4
Power-Speed Performance Comparison of
Exclusive OR Gates

Logic Family	Source	Delay ns	Power mW	Power x Delay pJ	Comments
TTL 54/74	17	12	37.5	445	
ECL MECL II	18	5	60	300	
ECL MECL III	18	1.5	90	135	
METLL	Measured Fig. 4.25 Integrated f_T = 1.8 GHz	5	1.4	7	$R_1=R_2$= 1K Sample supplied by Hewlett Packard

Table 4.5

Power-Speed Performance of Full Adders

Logic Family	Source	Delay of Sum ns	Delay of Carry ns	Power mW	Power x Delay pJ	Comments
TTL SN74LS	17	17.5	12.5	18.7	285	
SN54/74	17	8	6-8	75	600	
SN74H	17	5.5	5.5	110	605	
ECL MECL II	18	5	5	100	500	
CMOS RCA 10V	20	133	64	25(8MHz)	2450	
5V	20	255	80	2.5(4MHz)	380	
5V	20	225	80	0.75(1MHz)	114	
METLL	Measured Fig. 4.25 & Fig. 4.17 Integrated f_T=1.8GHz	7	5	6	72	$R_1=R_2=1K$ Samples Supplied by Hewlett Packard

Table 4.6

Power-Speed Performance Comparison of J-K Flip-Flops

Logic Family	Source	f_{max} MHz	Delay Clock-out ns	Power mW	Speed-Power pJ	Comments
TTL	[17]	10-20	25-40	50	1250-2000	
ECL MECL II	[18]		6	125	750	
			4	250	1000	
MECL III	[18]	350	1.75-2	190	333-380	
METLL	Calculated Fig.4.25 & Fig.4.23	10-200	10-20	4.5	45-70	$R_1=R_2=1K$

Table 4.7
LSI Families Comparison

Technology	Range of max. logic freq. MHz	Min. Risetime ns	Size		No. of Gates Per Chip	Power-Speed perform pJ
			Chip mil	Gate mil^2		
PMOS TI Intel HP-35	< 2 0.75 0.2	180	200x200 150x150 170x170		500 750 520+ 1K ROM	70-90
NMOS 5V Dynamic NMOS	< 4	35 6	120x120	8.2	1000	50 6
METLL	1-200	1-2	100x100	16	500	1-2

The delay-power comparisons are summarized in Fig. 4.34 - 4.37. The propagation delay is plotted versus power dissipation where the diagonal lines show constant power delay product. Figure 4.34 shows a 20 pJ for METLL versus 200-400 for other families, which Fig. 4.35 shows a 2 pJ for METLL versus 20 pJ for others. The last comparison shown in Fig. 4.37 is for full adders. The full adders speed usually determines the performance of computing systems. The double points indicate the delays for carry and sum. Not only bipolar families are shown but also CMOS to demonstrate that a METLL adder operating at 20 MHz shows the same delay-power product as the 20 times slower CMOS at 1 MHz.

Thus, we conclude that the METLL structures have the following features:

1 - A low delay-power product in comparison with available logic families, at least a factor of 2 improvement is achieved.
2 - A high density layout in both logic and flip-flop realization.
3 - A modular design approach can be easily implemented because only one basic structure can be used to realize any logic, flip-flop or memory function.
4 - Compatibility with ECL logic families can be easily achieved.
5 - Simplicity of layout and interconnection results in a higher yield and lower cost.
6 - The structure can be adopted directly for more advanced bipolar technology e.g. Isoplanner II [24] and this results in further reduction of power dissipation, silicon area and delays.

Because of these features several computer and semiconductor industries have shown interest in using the METLL structures including Burroughs Corp., Hewlett-Packard Company and Fairchild Semiconductors.

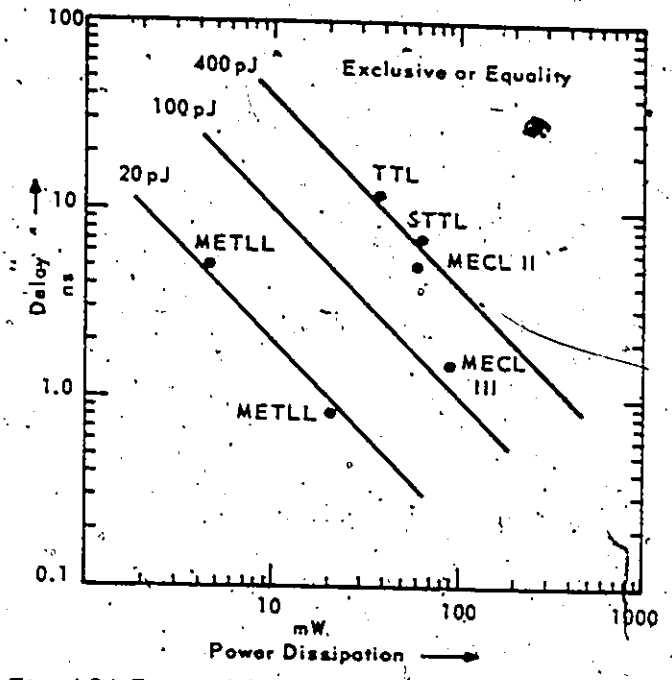


Fig. 4.34 Power–delay product comparison between METLL and circuits currently on the market for a single gate

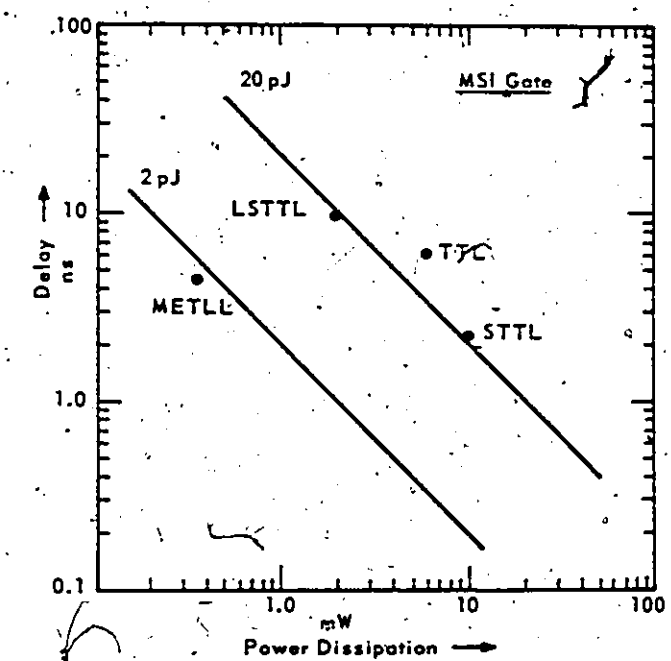


Fig. 4.35 Power–delay product comparison for an MSI gate.

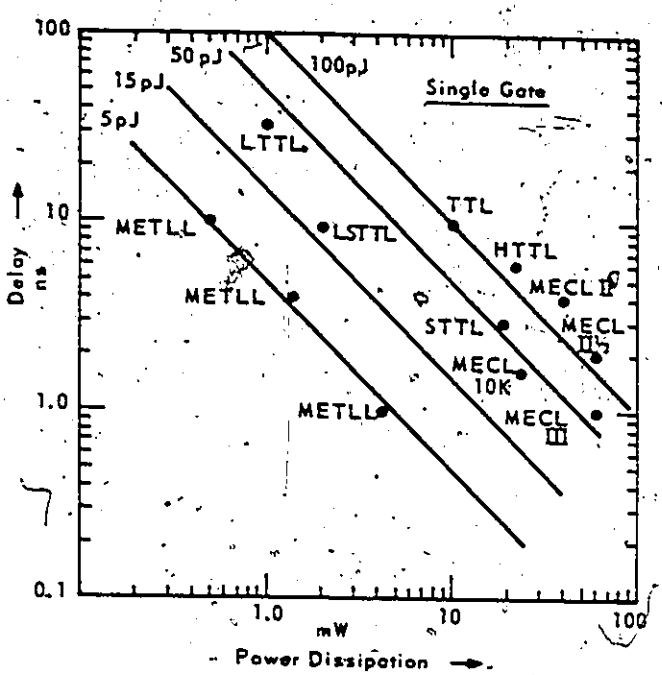


Fig. 4.36 Power–delay product comparison for Exclusive OR's

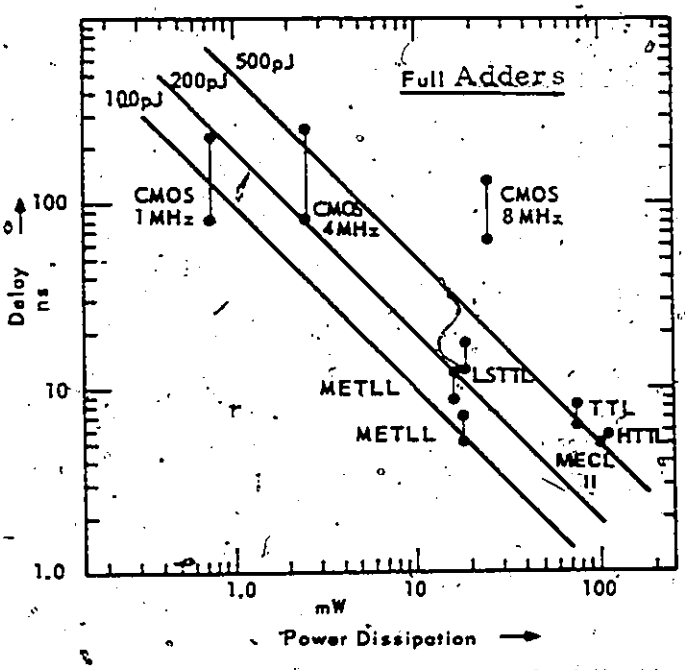


Fig. 4.37 Power–delay product comparison for full adders

CHAPTER 5

LOGIC DESIGN USING METLL STRUCTURES

LOGIC DESIGN USING METLL STRUCTURES

The METLL structure in its simplest form with one base input X and two input emitters Y and Z performs more logic than a simple gate. The input X is called control input because it can be regarded as controlling the logic performed by the structure; if X = '1', the output F = Y and if X = '0', the output F = Z. The two inputs X and Y are called direct inputs. This property of the structure, where one input controls the logic performed by other inputs, is useful in logic design. For example, if the variable X is available before Y and Z, a minimum delay results at the output because X will control the state of the lower level transistors (Q_{r2} or Q_a is conducting in Fig. 4.10) before Y and Z define the state of the upper level transistor (Q_{r1} conducting or not conducting in Fig. 4.10). This property is applicable to more complex METLL structures; with more than one control input and more than two direct inputs, and is used in developing the logic partition technique given in selection 5.4.

Because the simple METLL structure can realize NOR, OR or AND gates conventional logic design techniques can be adopted. However, this is the least economical way to use the logic capability of the structure. Alternatively, logic design techniques can be developed to use the structures, in their simplest form or in a more complex form, effectively and economically. Accordingly, the following problems can be identified:

(1) If the simplest structure is assumed to be the building block of a logic system;
   (a) what are the properties of one-realizable functions i.e. the functions which can be realized using one structure?
   (b) Based on such properties, it is possible to develop a logic design technique to realize an arbitrary

logic function using one structure.

(c) If the function is not one-realizable, what are the minimum number of structures which can be used to realize that function?

(2) If the simplest structure and any more complex ones are assumed to be the building blocks of a logic system;

(a) If a function is not one-realizable using the simplest structure, what is the next more complex structure which can realize it?

(b) How is the realization (2a) compared with (1c)?

(3) In the realization of logic functions, how can the property of control inputs controlling the logic performed by direct inputs, be used in logic design?

Problems (1a), (1b) and (3) are treated in this chapter. Although problems (1c) and (2) are not formally treated, practical examples are given to show the simplicity of the design using complex structures and multiple of simple structure, as shown in Fig. 5.12 and Fig. 5.14.

In section 5.1, some useful logic properties of the stuucture are presented as results with proofs. 'A structure' always means a METLL structure with one control input X and two direct inputs Y and Z where X, Y and Z are first-order OR functions i.e.

$$X = x_1 + x_2 + \ldots + x_n$$

$$Y = y_1 + y_2 + \ldots + y_m$$

$$Z = z_1 + z_2 + \ldots + z_\ell$$

and the output function $F = XY + \bar{X}Z$. Based on these properties a logic design technique is developed in section 5.2 to be used in realizing a logic function. The application of the technique to some examples are given in section 5.3. Finally, a

logic partition technique, which allows a function to be partitioned into logic blocks, such that a block controls the logic performed by another, is presented in section 5.4. This partition can be based on any design criteria. This criteria is chosen to be the time of arrival of the input variables and it is demonstrated in section 5.5 that this results in a design with minimum delays.

5.1 LOGIC PROPERTIES

Definition 1

We define the sets S_x, S_y, S_z as "the set of literals" of the input X, Y and Z of a structure.

Example: $X = x_1 + \bar{x}_2$

$Y = x_3$

$Z = \bar{x}_3$

then

$$S_x = (x_1, \bar{x}_2) \quad , \quad S_y = (x_3) \quad , \quad S_z = (\bar{x}_3)$$

Definition 2

A METLL structure is called 'a structure with biased input(s)' if one (or more) of the inputs is at fixed logical '0' or logical '1'. Accordingly, the following cases result :

Number	X	Y	Z	$\bar{F} = XY + \bar{X}Z$
1	0	Y	Z	Z
2	1	Y	Z	Y
3	X	0	Z	$\bar{X}Z$
4	X	1	Z	X + Z
5	X	Y	0	XY
6	X	Y	1	$\bar{X}$ + Y
7	0	0	Z	Z
8	0	1	Z	Z
9	1	0	Z	0
10	1	1	Z	1
11	X	0	0	0
12	X	0	1	$\bar{X}$
13	X	1	0	X
14	X	1	1	1
15	0	Y	0	0
16	0	Y	1	1
17	1	Y	0	Y
18	1	Y	1	Y

Table 5.1 Structures with biased inputs

Result 1

A structure with biased input(s) can realize

(a) a conventional OR gate (case 1, 2, 4, 7, 8, 13, 17 and 18 in the above table)

(b) a conventional NOR gate (case 12 in the above table)

(c) a two-level gating logic; OR-NOR/AND (case 3), OR-OR/AND (case 5), NOR-OR/OR (case 6).

Result 2

A structure is a complete logic element and can realize any arbitrary logic function.

Proof

Since the structure can realize a NOR gate, which is a complete logic element, then the structure is a complete logic element.

Definition 3

A logical function is called one-realizable (1-realizable) if it can be realized using a single structure.

Result 3

All logical functions of 1 and 2 variables are 1-realizable.

Proof

An assignment table is constructed to prove this result (Table 5.2).

Result 4

If $\overline{f_n(a_1, \ldots a_n)}$ is 1-realizable, then $F = b + f_n$ is also 1-realizable, where b is a first-order OR function.

Proof

Since f_n is 1-realizable, it can be expressed as $f_n = XY + \overline{X}Z$, where X, Y, and Z are first-order OR functions then

$F = b + XY + \bar{X}Z$

$= X(Y + b) + \bar{X}(Z + b)$

$= XY_1 + \bar{X}Z_1$

where X, Y_1 and Z_1 are first-order OR functions. Then F is 1-realizable.

f	An Assignment			$F = XY + \bar{X}Z$
	X	Y	Z	
0	0	Ø	0	
1	1	1	Ø	
a	1	a	Ø	
$\bar{a}$	a	0	1	
0	0	Ø	0	
1	1	1	Ø	
a	1	a	Ø	
$\bar{a}$	a	0	1	
b	1	b	Ø	
$\bar{b}$	b	0	1	
a+b	a+b	1	0	
a+$\bar{b}$	a+$\bar{b}$	1	0	
$\bar{a}$+b	$\bar{a}$+b	1	0	
$\bar{a}$+$\bar{b}$	$\bar{a}$+$\bar{b}$	1	0	
ab	$\bar{a}$+$\bar{b}$	0	1	
a$\bar{b}$	$\bar{a}$+b	0	1	
$\bar{a}$b	a+$\bar{b}$	0	1	
$\bar{a}\bar{b}$	a+b	0	1	
ab+$\bar{a}\bar{b}$	a	b	$\bar{b}$	
$\bar{a}$b+a$\bar{b}$	a	$\bar{b}$	b	

Table 5.2 1 and 2 variables 1-realizable functions
(Ø = don't care condition)

Result 5

If F is 1-realizable, then F^D (the dual function of F) is also 1-realizable.

Proof

Since F is 1-realizable, it can be expressed as

$F = XY + \overline{X}Z$ X, Y and Z are first-order OR functions.

$F^D = (X + Y)(\overline{X} + Z)$

$= XZ + \overline{X}Y + YZ$

$= XZ + \overline{X}Y$

Where X, Y and Z are first-order OR functions.

Then F^D is 1-realizable.

Result 6

If F is 1-realizable, then $\overline{F}$ is at most 2-realizable.

Proof

Since F is 1-realizable, $\overline{F}$ can be realized using a second structure, by applying F as an X input, Y and Z are '0' and '1' respectively.

Result 7

If F is 1-realizable, then $\overline{F^D} = \overline{F}^D$ is at most 2-realizable.

Proof

This results directly from Result 5 and 6 above.

Result 8

A function expressed as $F = F_1 + F_2 + .. + F_M$ where $F_1, ...F_M$ are 1-realizable, can be realized using:

(a) at most M structures in two logic levels, if at most (M - 1) of the M functions F_i, i = 1, ..., M are 1-realizable with Z or Y = 1

(b) at most (M + 1) structures in two logic levels, if all the M functions F_i, i = 1, ..., M are 1-realizable with Z or Y = 1

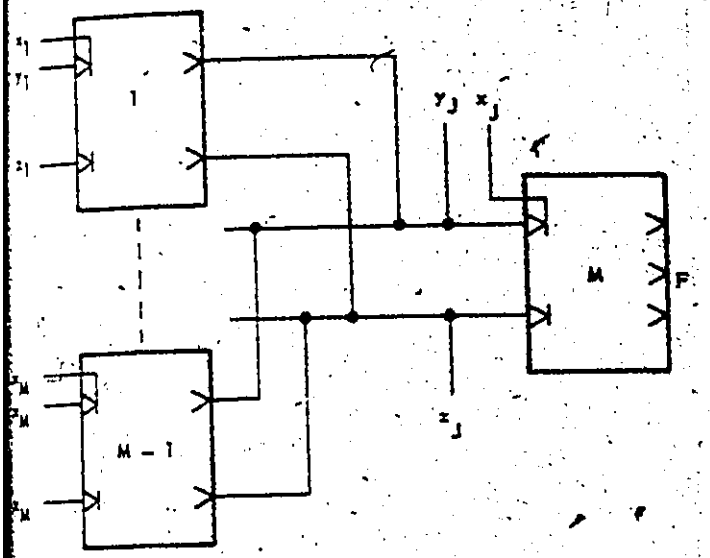


Fig. 5.1(a) Case (a) of Result 8

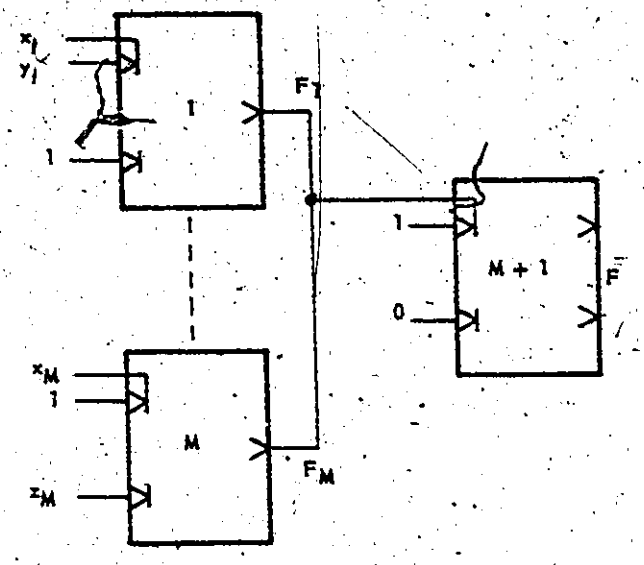


Fig. 5.1(b) Case (b) of Result 8

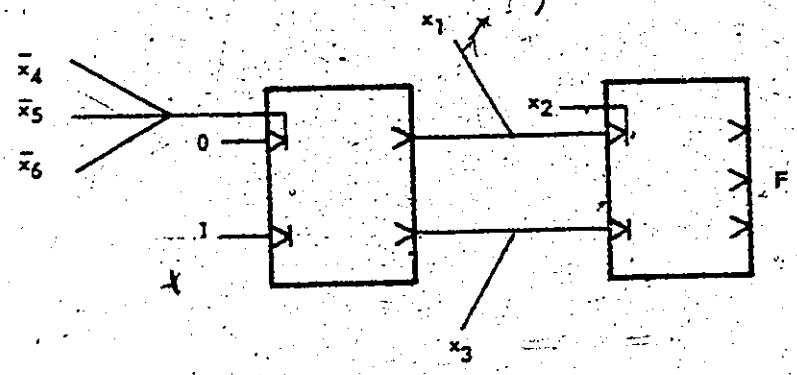


Fig. 5.2 Example $F = x_1x_2 + x_3\bar{x}_2 + x_4x_5x_6$

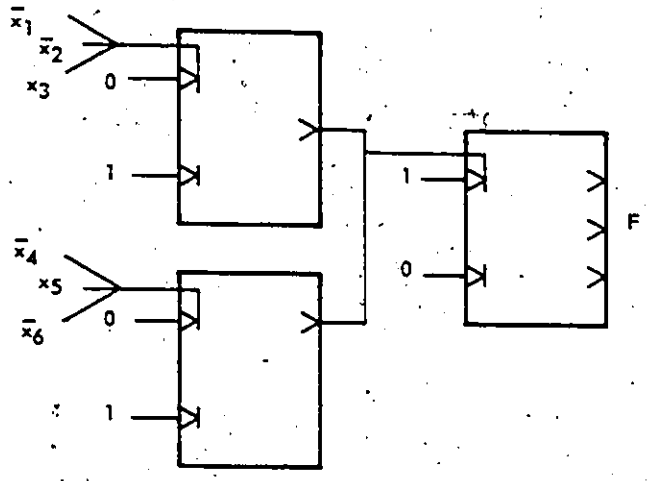


Fig. 5.3 Example $F = x_1 x_2 \bar{x}_3 + x_4 \bar{x}_5 x_6$

Proof

Since F_i, i = 1, ..., M is 1-realizable then

$F_i = X_i Y_i + \bar{X}_i Z_i$

(a) if at most (M - 1) of the M functions are 1-realizable with Z or Y = 1, then there exists F_j where

$F_j = X_j Y_j + \bar{X}_j Z_j$

Y_j and $Z_j \neq$ '1' then F can be expressed as

$$F = F_1 + F_2 + \ldots + F_{j-1} + F_{j+1} + \ldots F_M + X_j Y_j + \bar{X}_j Z_j$$

$$= X_j (Y_j + F_1 + F_2 + \ldots + F_{j-1} + F_{j+1} + \ldots + F_M) +$$

$$\bar{X}_j (Z_j + F_1 + F_2 + \ldots + F_{j-1} + F_{j+1} + \ldots + F_M)$$

Then at most M structures are needed as shown in Fig. 5.1 (a).

(b) if all the M functions F_i, i = 1, ..., M are 1-realizable with Z or Y = 1, then F_i is in the form $F_i = \bar{X}_i + Y_i$ or $X_i + Z_i$ then F can not be expressed as in (a) and in this case an extra structure is required to act as an OR gate as shown in Fig. 5.1 (b).

Example (a)

$$F = (X_1 X_2 + X_3 \bar{X}_2) + X_4 X_5 X_6 = F_1 + F_2$$

Can be realized using 2 structures as shown in Fig. 5.2.

Example (b)

$$F = X_1 X_2 \bar{X}_3 + X_4 \bar{X}_5 X_6$$

Can be realized using 3 structures as shown in Fig. 5.3.

Result 9

A function expressed in the minimal sum-of-products form can be realized using M structures where $M \leq N+1$, where N is the number of products.

Proof

This results directly from Result 8 above.

Result 10

A function expressed as $F = XF_1 + \bar{X}F_2$ where F_1 and F_2 are 1-realizable, X is a first-order OR function can be realized using at most 3 structures.

Proof

This results directly from the definition of 1-realizable functions.

Definition 4

Given a logical function F expressed as sum of prime implicants, a "row coincidence table" of this function is a table which contains the literals of the function and the prime implicants (PI) where they appear.

If F is 1-realizable function, then F can be expressed as:

$$F = XY + \bar{X}Z$$

$$= (X_1 + X_2 + .. X_n)(Y_1 + Y_2 + .. Y_m)$$

$$+ (\bar{X}_1\bar{X}_2 .. \bar{X}_n)(Z_1 + Z_2 + .. + Z_\ell)$$

$$= X_1Y_1 + X_1Y_2 + ... + X_1Y_m$$

$$+ X_2Y_1 + X_2Y_2 + ... + X_2Y_m$$

$$+ .....$$

$$+ X_nY_1 + X_nY_2 + ... + X_nY_m$$

$$+ (\bar{X}_1 .. \bar{X}_n)Z_1 + (\bar{X}_1 .. \bar{X}_n)Z_2 + .. + (\bar{X}_1 .. \bar{X}_n)Z_\ell$$

where $X_i \neq Y_j$, $i = 1, ..., n$, $j = 1, ..., m$

$X_i \neq Z_K$, $K = 1, ..., \ell$

$Y_j \neq Z_K$

i.e. there is no common literal between S_x, S_y and S_z. Then each product is a prime implicant i.e.

$$F = P_{11} + P_{12} + \cdots + P_{1m}$$
$$+ P_{21} + P_{22} + \cdots + P_{2m}$$
$$+ \cdots\cdots$$
$$+ P_{n1} + P_{n2} + \cdots + P_{nm}$$
$$+ P_1 + P_2 + \cdots + P_\ell$$
$$= \sum_{i=1}^{n} \sum_{j=1}^{m} P_{ij} + \sum_{K=1}^{\ell} P_k$$

and its row-coincidence table is written as shown:

Row Name	Literal	PIs
α_1	X_1	$P_{11}, P_{12}, \cdots, P_{1m}$
α_2	X_2	$P_{21}, P_{22}, \cdots, P_{2m}$
α_3	X_3	$P_{31}, P_{32}, \cdots, P_{3m}$
.	.	.
α_n	X_n	$P_{n1}, P_{n2}, \cdots, P_{nm}$
α_{n+1}	Y_1	$P_{11}, P_{21}, \cdots, P_{n1}$
α_{n+2}	Y_2	$P_{12}, P_{22}, \cdots, P_{n2}$
.	.	.
α_{n+m}	Y_m	$P_{1m}, P_{2m}, \cdots, P_{nm}$
α_{n+m+1}	Z_1	P_1
α_{n+m+2}	Z_2	P_2
.	.	.
$\alpha_{n+m+\ell}$	Z_ℓ	P_ℓ
$\alpha_{n+m+\ell+1}$	$\bar{X}_1$	$P_1, P_2, \cdots, P_\ell$
$\alpha_{n+m+\ell+2}$	$\bar{X}_2$	$P_1, P_2, \cdots, P_\ell$
.	.	.
$\alpha_{n+m+2\ell}$	$\bar{X}_n$	$P_1, P_2, \cdots, P_\ell$

Definition 5

We define the order of a row in the row-coincidence table as the number of prime implicants in that row.

The ordering sets among the rows, for a 1-realizable function are:

S_1, A set of rows whose elements of order 1

$= \{\alpha_{n+m+1}, \alpha_{n+m+2}, \cdots, \alpha_{n+m+\ell}\}$

S_m, A set of rows whose elements of order m

$= \{\alpha_1, \alpha_2, \cdots, \alpha_n\}$

S_n, A set of rows whose elements of order n

$= \{\alpha_{n+1}, \alpha_{n+2}, \cdots, \alpha_{n+m}\}$

S_ℓ, A set of rows whose elements of order ℓ

$= \{\alpha_{n+m+\ell+1}, \cdots\cdots, \alpha_{n+m+2\ell}\}$

and

U, The unity set which contains all the rows of the coincidence table

$= \{\alpha_1, \alpha_2, \cdots, \alpha_n, \alpha_{n+1}, \alpha_{n+2}, \cdots \alpha_{n+m}, \alpha_{n+m+1},$

$\alpha_{n+m+2}, \cdots, \alpha_{n+m+\ell}, \alpha_{n+m+\ell+1}, \alpha_{n+m+\ell+2}, \alpha_{n+m+2\ell}\}$

Definition 6

We define a set S_{α_j} as the set with principle element α_j. S_{α_j} is a subset of U which contains

(1) all elements α_i, $i = 1, \ldots, M$, where M is the total number of elements in U such that $\alpha_i \cap \alpha_j \neq \emptyset$ where α_i and α_j are two sets of PIs corresponding to the rows α_i and α_j.

and

(2) all elements α_s , s = 1 , ... , M ; such that $\alpha_s \cap \alpha_i \neq \emptyset$

For 1-realizable function, in order to obtain S_{α_1} we start with the principle element α_1 which contains P_{11}, P_{12}, ... , P_{1m} and then find all the rows which contain P_{11} , in this case it is α_{n+1} and then all the rows which contain P_{12} (it is α_{n+2}) and then all the rows which contain P_{1m} (it is α_{n+m}). Then starting with $\alpha_{n+1} = \{P_{11} , P_{21} , \ldots , P_{n1}\}$, find all the rows which contain P_{11} , P_{21} , .. , P_{n1} and in this case they are α_1 , α_2 , .. , α_n , i.e.

$$S_{\alpha_1} = \{\alpha_1 , \alpha_{n+1} , \alpha_{n+2} , \alpha_{n+3} , \ldots , \alpha_{n+m} , \alpha_2 , \alpha_3 , .. , \alpha_n\}$$

$$= S_a$$

Similarly

$$S_{\alpha_{(n+m+\ell+1)}} = \{\alpha_{(n+m+\ell+1)} , \alpha_{(n+m+\ell+2)} , \alpha_{(n+m+2\ell)} \alpha_{(n+m+1)} , \alpha_{(n+m+2)} , \ldots\ldots , \alpha_{(n+m+\ell)}\}$$

$$= S_b$$

i.e. U can be partitioned into two sets with two principle elements and

$$S_a \cap S_b = \emptyset \quad \text{and} \quad S_a \cup S_b = U$$

Definition 7

If each row of the row-coincidence table is considered to be a row in a matrix, then a matrix can be written corresponding to each subset of U.

For 1-realizable function S_a can be partitioned to two sets S_α and S_β ($S_\alpha \cup S_\beta = S_a$) , such that $\beta = \alpha^T$, where α , β are two matrices corresponding to S_α and S_β respectively

and T represents the transpose of a matrix, i.e.

$$\alpha = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \cdot \\ \cdot \\ \cdot \\ \alpha_n \end{bmatrix} = \begin{bmatrix} P_{11} & P_{12} & \cdots & P_{1m} \\ P_{21} & P_{22} & \cdots & P_{2m} \\ \cdot & & & \\ \cdot & & & \\ \cdot & & & \\ P_{n1} & P_{n2} & \cdots & P_{nm} \end{bmatrix}_{n \times m}$$

$$\beta = \begin{bmatrix} \alpha_{n+1} \\ \alpha_{n+2} \\ \cdot \\ \cdot \\ \cdot \\ \alpha_{n+m} \end{bmatrix} = \begin{bmatrix} P_{11} & P_{21} & \cdots & P_{n1} \\ P_{12} & P_{22} & \cdots & P_{n2} \\ \cdot & & & \\ \cdot & & & \\ \cdot & & & \\ P_{1m} & P_{2m} & \cdots & P_{nm} \end{bmatrix}_{m \times n} = \alpha^T$$

where the set of literals corresponding to α and β have no common literals. Similarly S_b can be partitioned to two sets S_γ and S_ζ ($S_\gamma \cup S_\zeta = S_b$), such that $\gamma = [\zeta^T]$ where γ, ζ are two matrices corresponding to the sets S_γ and S_ζ respectively, ζ has n identical rows and $[\,.\,]$ represents the first column-of, i.e.

$$\zeta = \begin{bmatrix} \alpha_{n+m+\ell+1} \\ \alpha_{n+m+\ell+2} \\ \cdot \\ \cdot \\ \cdot \\ \alpha_{n+m+\ell+\ell} \end{bmatrix} = \begin{bmatrix} P_1 & P_2 & \cdots & P_\ell \\ P_1 & P_2 & \cdots & P_\ell \\ \cdot & & & \\ \cdot & & & \\ \cdot & & & \\ P_1 & P_2 & \cdots & P_\ell \end{bmatrix}_{n \times \ell}$$

where the set of literals corresponding to ζ ; $\bar{x}_1$, $\bar{x}_2$, .. $\bar{x}_n$ contains the complements of the set of literals corresponding to α ; x_1 , x_2 , .. x_n and

$$Y = \begin{bmatrix} \alpha_{n+m+1} \\ \alpha_{n+m+2} \\ \vdots \\ \alpha_{n+m+\ell} \end{bmatrix} = \begin{bmatrix} P_1 \\ P_2 \\ \vdots \\ P_\ell \end{bmatrix}_{\ell X 1} = \left[\mathcal{J}^T \right]$$

5.2 LOGIC DESIGN TECHNIQUE

Given a function f_n as sum of M prime implicants (PI), the following presents a design procedure for its realization, based on definitions 4 - 7 and result 4. The design procedure is summarized and followed at each step by an example.

Step 1

Test to find if f_n can be expressed as $f_n = F + b$, where b is a first-order OR function.

Example

$$f = X_1 + X_2X_3 + X_4\overline{X}_3 = b + F$$

where

$$b = X_1$$

$$F = X_2X_3 + X_4\overline{X}_3 = P_1 + P_2$$

Step 2

Write the "row-coincidence table" for F where each row α_i represents a literal and the prime implicants where it appears.

Example

Row Name	Literal	PIs
α_1	X_2	P_1
α_2	X_3	P_1
α_3	$\overline{X}_3$	P_2
α_4	X_4	P_2

Step 3

Write the ordering sets among the table rows, where the order of a row is the number of PIs in that row.

Example

$S_1 = (\alpha_1 , \alpha_2 , \alpha_3 , \alpha_4)$
has elements of the same order I.

Step 4

Starting with any α_j of the largest set, form the set S_{α_j} with a principle element α_j.

Example

$$S_{\alpha_1} = S_a = \{\alpha_1 , \alpha_2\} = \{(P_1) , (P_1)\}$$

$$S_b = \{\alpha_3 , \alpha_4\} = \{(P_2) , (P_2)\}$$

where $S_a \cap S_b = \emptyset$ and $S_a \cup S_b = U$

where $\emptyset$ and U are the empty set and the union set respectively

Step 5

If $S_{\alpha_j} = S_a$, $S_a \cap S_b = \emptyset$ and $S_a \cup S_b = U$
check if both conditions (a) and (b) below are satisfied.
(a) $\beta = \alpha^T$

where α and β are two matrices corresponding to a single

partition of S_a (or S_b) and $S_\alpha \cup S_\beta = S_{\alpha_j}$, T represents a transpose of a matrix, and the set of literals corresponding α and β have no common element.

(b) $\gamma = [\beta^T]$

where γ and β are two matrices corresponding to a single partition of S_b (or S_a) and $S_\gamma \cup S_\beta = S_b$, β has n identical rows, $[\ ]$ represents first column of a matrix, the set of literals corresponding to β contains the complements of the same set corresponding to α.

Example

(a) $\alpha = [P_1]$, $\beta = [P_1]$

$\beta = \alpha^T$ is satisfied

The set of literals corresponding to α and β have no common element.

(b) $\gamma = [P_2]$, $\beta = [P_2]$

$\gamma = [\beta^T]$ is satisfied; β has identical rows.

The set of literals corresponding to β contains the complements of the same set corresponding to α.

Step 6

If S_a is the set U of all PIs, check if only one of the above conditions (a) or (b) is satisfied.

Example

Not applicable.

Step 7

If (5) and (6) are not satisfied, F is not 1-realizable.

Example

Fig. 5.4 shows 1-realizable realization of f.

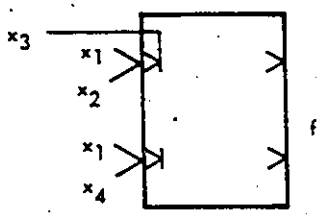


Fig. 5.4 Example $f = x_1 + x_2x_3 + x_4\bar{x}_3$

5.3 GENERAL EXAMPLES

The above logic design technique can be adapted to the realization of an arbitrary function and the usefulness of the technique can be determined from its application to general examples.

Example 1

$$F_1 = \bar{X}_1\bar{X}_2X_4 + X_1X_3 + X_2X_3$$

$$= P_1 + P_2 + P_3$$

(1) Not applicable.

(2)

Row Name	Literal	PIs
α_1	X_1	P_2
α_2	X_2	P_3
α_3	X_3	P_2 , P_3
α_4	X_4	P_1
α_5	$\bar{X}_1$	P_1
α_6	$\bar{X}_2$	P_1

(3) $S_1 = \{\alpha_1, \alpha_2, \alpha_4, \alpha_5, \alpha_6\}$

$S_2 = \{\alpha_3\}$

(4) $S_{\alpha_3} = S_a = \{\alpha_3, \alpha_1, \alpha_2\} = \{(P_2, P_3), (P_2), (P_3)\}$

$S_b = \{\alpha_4, \alpha_5, \alpha_6\} = \{(P_1), (P_1), (P_1)\}$

where $S_a \cap S_b = \emptyset$ and $S_a \cup S_b = U$

(5) (a) S_a can be partitioned to S_α and S_β ($S_\alpha \cup S_\beta = S_a$), where $S_\alpha = \{\alpha_3\}$ and $S_\beta = \{\alpha_1, \alpha_2\}$

and

$$\alpha = [\alpha_3] = [P_2 \; P_3]$$

$$\beta = \begin{bmatrix}\alpha_1 \\ \alpha_2\end{bmatrix} = \begin{bmatrix}P_2 \\ P_3\end{bmatrix} = \alpha^T$$

Then condition (a) is satisfied.

(b) S_b can be partitioned to S_γ and S_β ($S_\gamma \cup S_\beta = S_b$) where $S_\beta = \{\alpha_4, \alpha_5\}$ and $S_\gamma = \{\alpha_6\}$

and

$$\beta = \begin{bmatrix} \alpha_4 \\ \alpha_5 \end{bmatrix} = \begin{bmatrix} P_1 \\ \overline{P}_1 \end{bmatrix}$$

$$\gamma = [\alpha_6] = [P_1] = [\beta^T]$$

Then condition (b) is satisfied.

(6) Not applicable.

(7) From (5), F_1 is 1-realizable in the form

$$F_1 = (X_1 + X_2)X_3 + (\overline{X_1 + X_2})X_4$$

i.e.

$X = X_1 + X_2$

$Y = X_3$

$Z = X_4$

Example 2

$$F_2 = \overline{X}_1\overline{X}_2X_3 + X_1X_2\overline{X}_4 + \overline{X}_3\overline{X}_4$$
$$= P_1 + P_2 + P_3$$

(1) Not applicable.

(2)

Row Name	Literal	PIs
α_1	X_1	P_2
α_2	X_2	P_2
α_3	X_3	P_1
α_4	$\overline{X}_4$	P_2, P_3
α_5	$\overline{X}_1$	P_1
α_6	$\overline{X}_2$	P_1
α_7	$\overline{X}_3$	P_3

(3) $S_1 = \{\alpha_1, \alpha_2, \alpha_3, \alpha_5, \alpha_6, \alpha_7\}$

$S_2 = \{\alpha_4\}$

(4) $S_{\alpha_4} = S_a = \{\alpha_4, \alpha_1, \alpha_2, \alpha\}$

$= \{(P_2, P_3), (P_2), (P_2), (P_3)\}$

$S_b = \{\alpha_3, \alpha_5, \alpha_6\}$

$= \{(P_1), (P_1), (P_1)\}$

where $S_a \cap S_b = \emptyset$

and $S_a \cup S_b = U$

(5) (a) No single partition of S_a satisfies condition (a).
No single partition of S_a satisfies condition (b).
(b) No single partition of S_b satisfies condition (b).

(6) Not applicable.

(7) From (5), F_2 is not 1-realizable.

Example 3

$$F_3 = \overline{X}_1\overline{X}_2\overline{X}_3X_4 + X_2$$

(1) $F_3 = X_2 + \overline{X}_1\overline{X}_2\overline{X}_3X_4 = b + F$

$F = \overline{X}_1\overline{X}_2\overline{X}_3X_4 = P$, $b = X_2$

(2)

Row Name	Literal	PIs
α_1	$\overline{X}_1$	P
α_2	$\overline{X}_2$	P
α_3	$\overline{X}_3$	P
α_4	X_4	P

(3) $S_1 = \{\alpha_1, \alpha_2, \alpha_3, \alpha_4\} = U$

(4) $S_a = \{\alpha_1, \alpha_2, \alpha_3, \alpha_4\} = U$

(5) Not applicable.

(6) S_a can be partitioned to S_β and S_γ ($S_\beta \cup S_\gamma = S_a$) where $S_\beta = \{\alpha_1, \alpha_2, \alpha_3\}$ and $S_\gamma = \{\alpha_4\}$ and

$$\beta = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix} = \begin{bmatrix} P \\ P \\ P \end{bmatrix}$$

$$= \begin{bmatrix} \alpha_4 \end{bmatrix} = \begin{bmatrix} P \end{bmatrix} = \begin{bmatrix} \beta^T \end{bmatrix}$$

Then condition (b) is satisfied.

(7) From (6), F and F_3 is 1-realizable in the form

$$F = (X_1 + X_2 + X_3)0 + (\overline{X_1 + X_2 + X_3})X_4$$

i.e.

$$X = X_1 + X_2 + X_3$$

$$Y = 0$$

$$Z = X_4$$

$$F_3 = (X_1 + X_2 + X_3)(0 + X_2) + (\overline{X_1 + X_2 + X_3})(X_4 + X_2)$$

i.e.

$$X = X_1 + X_2 + X_3$$

$$Y = X_2$$

$$Z = X_4 + X_2$$

5.4 LOGIC PARTITION TECHNIQUE

In the previous section we have indicated that not all logical functions are 1-realizable. If a function is not 1-realizable or it is 1-realizable with a large number of fan-in, it is possible to partition the input variables such that

$$f_n(x_1, \ldots, x_n) = f(y_1, \ldots y_m, x_{j+1}, \ldots x_n)$$

where $y_i = f_i(x_1, \ldots x_j)$, $i = 1, \ldots m$

A logic partition technique is developed in this section, using the Karnaugh map and is based on the simple disjoint decomposition techniques given by Ashenhurst [1, 2] and Curtis [3].

Let us consider a single output combinational circuit with n inputs $(x_1, \ldots x_n)$. The inputs can be partitioned into N groups or sub-input sets $(I_1, I_2, \ldots I_N)$ where the sub-input I_i contains n_i inputs as shown in Fig. 5.5. The partitioning can be done on the fan-in capability of the METLL structure. Alternatively, if the delay per logic operation and the signal path can be estimated from the overall system design, the partitioning can be on the basis of the time of arrival of the inputs and a minimum delay system can be designed.

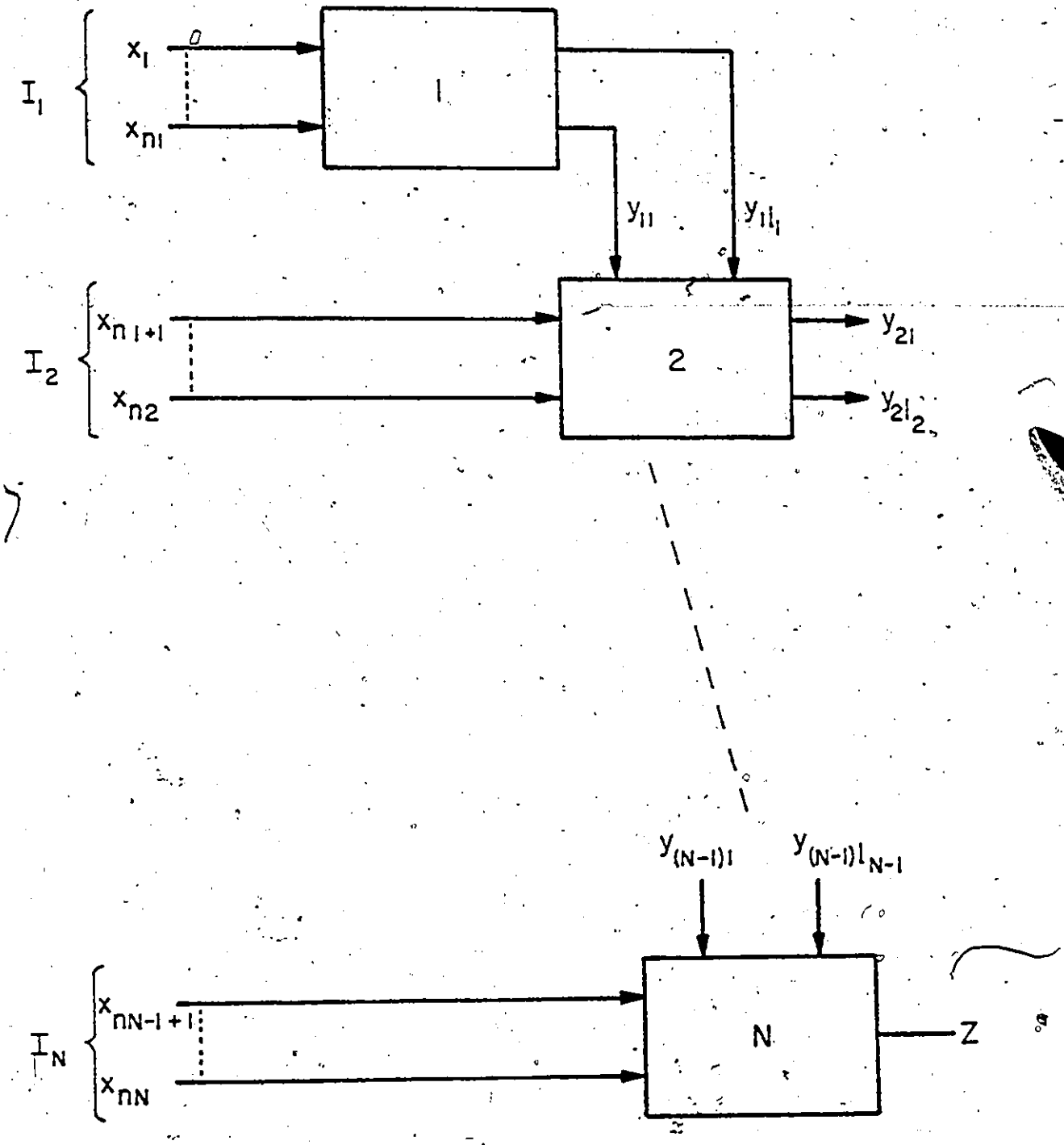


Fig. 5.5 Partition of Single Output Logic Function

Although, in this case, the time sequence at the input is assumed to be I_1 , I_2 , ... I_N the operation of the circuit is combinational and the final value of output is not affected by the sequence.

The outputs of a block are used as input control functions for the next block and are labelled y_{i_1} , ... $y_{i\ell_i}$ for the ith block where ℓ_i is the number of the control functions. The design procedure is now summarized and followed at each step by a general example.

Step 1

Starting with the truth table of the logic function, the Karnaugh-map (K-map) is used to plot the function as a partition with two blocks according to the time of arrival of the inputs. The sub-input sets I_1 and I_2 can be assumed to be $(x_1 , x_2 , \ldots x_p)$ and $(x_{p+1} , x_{p+2} , \ldots x_n)$ respectively and are indicated on the side and across the top of the K-map. The partition of the input variables is indicated by I_1/I_2.

Example

Figure 5.6 shows the truth table of a 4-variable function:

$$Z = x_1\bar{x}_2x_3 + x_1x_3\bar{x}_4 + \bar{x}_2x_3\bar{x}_4 + \bar{x}_1x_2\bar{x}_3\bar{x}_4 + \bar{x}_1\bar{x}_2\bar{x}_3x_4 + x_1x_2\bar{x}_3x_4$$

$$+ \bar{x}_1x_2x_3x_4$$

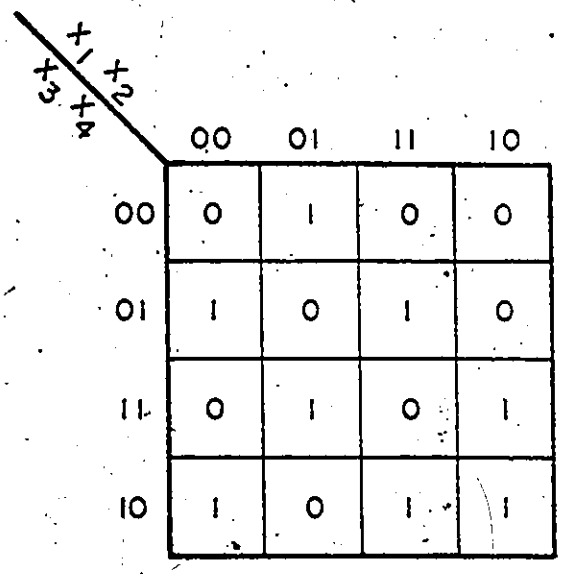
Two different arbitrary partitions are chosen for illustration and are shown in Fig. 5.7:

partition (a): $I_1/I_2 = \left[(x_1,x_2) \;/\; (x_3,x_4)\right]$

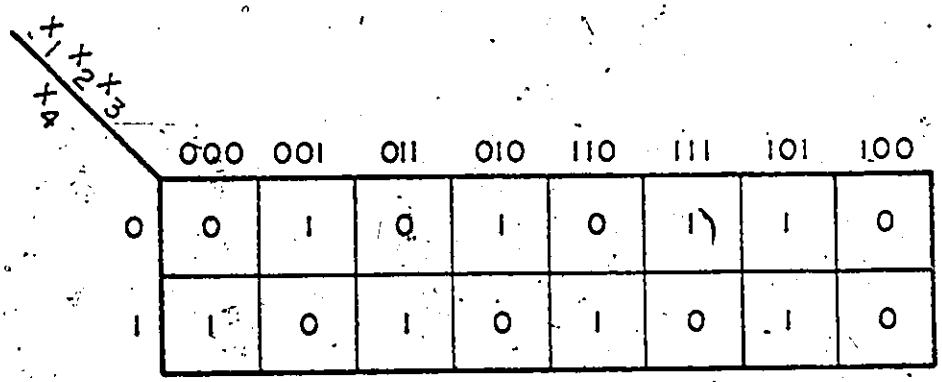
partition (b): $I_1/I_2 = \left[(x_1,x_2,x_3) \;/\; (x_4)\right]$

X_1	X_2	X_3	X_4	Z
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

Fig. 5.6 Truth Table of a 4–Variable Function



(a) $\left[(x_1, x_2)/(x_3, x_4)\right]$ K-map



(b) $\left[(x_1, x_2, x_3)/(x_4)\right]$ K-map

Fig. 5.7 K-Maps for Partition (a) and (b)

Step 2

A control function map (C-map) is generated from a K-map by combining similar columns. Each column in the K-map represents a cube with 2^{n-p} vertices and the values of the function for each column can be expressed as an (n-p)-tuple. Similar columns are defined as these with an identical (n-p)-tuple.

To construct the C-map, assign the left hand column of the K-map a state S_1 and assign S_1 to all similar columns. Then assign state S_2 to the next unassigned column and repeat until all columns have assigned states. Each column in the C-map represents a state and each state is coded in an assignment chosen to simplify the control variables y_i , .. $y_{i\ell_i}$ as a function of the sub-input I_1 [4].

Example

Figure 5.8 shows the C-maps for the above example.

Step 3

The control function (y) can be expressed, according to the assignment given, as function of $(x_1, x_2, \ldots x_p)$. For each state the output Z is expressed as function of $(x_{p+1}, \ldots x_n)$.

Example

(a) $y_{11} = x_1\bar{x}_2$

$y_{12} = \bar{x}_1 x_2$

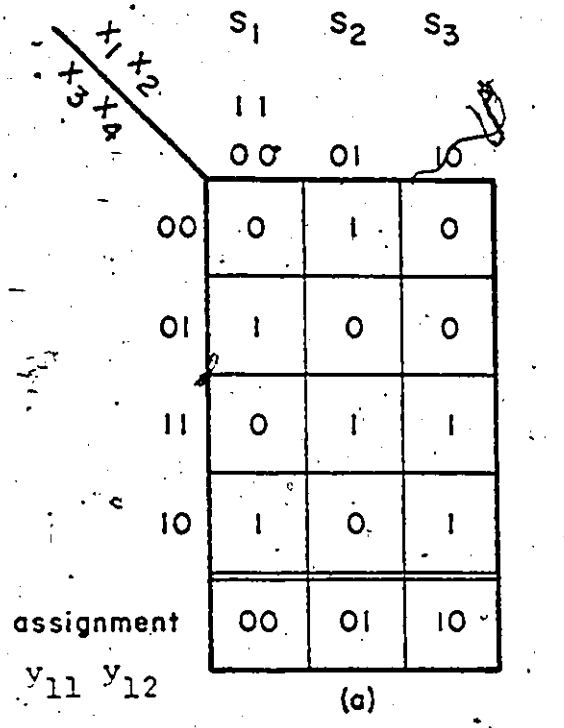
In state S_1 : $Z = x_3\bar{x}_4 + \bar{x}_3 x_4$

In state S_2 : $Z = x_3 x_4 + \bar{x}_3\bar{x}_4$

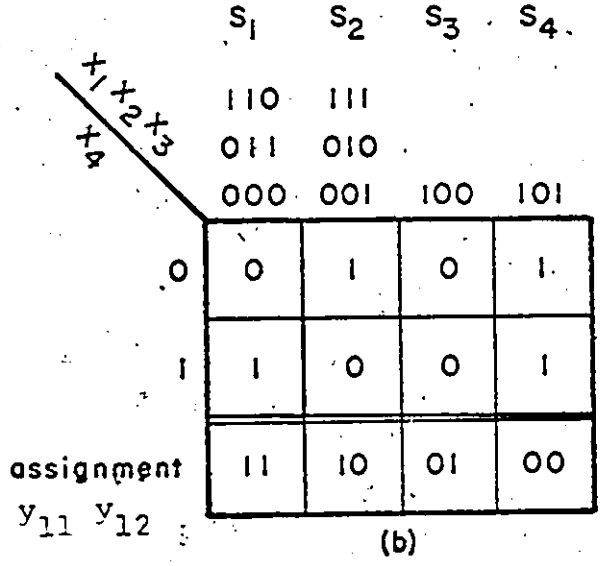
In state S_3 : $Z = x_3$

(b) $y_{11} = \bar{x}_1 + x_2$

$y_{12} = \bar{x}_2\bar{x}_3 + x_1\bar{x}_3 + \bar{x}_1 x_2 x_3$



(a)



(b)

Fig. 5.8 C-Maps for Partition (a) and (b)

In state S_1 : $Z = x_4$, in state S_2 : $Z = \bar{x}_4$, in state S_3 : $Z = 0$, in state S_4 : $Z = 1$.

Figure 5.9 shows the realization of the above example with two blocks, where the output is realized as function of the direct and the control functions:

(a) $Z = \bar{y}_{11}\bar{y}_{12}(x_3\bar{x}_4 + \bar{x}_3x_4) + \bar{y}_{11}y_{12}(x_3x_4 + \bar{x}_3\bar{x}_4) + y_{11}\bar{y}_{12}x_3$

(b) $Z = y_{11}y_{12}x_4 + y_{11}\bar{y}_{12}\bar{x}_4 + \bar{y}_{11}\bar{y}_{12}$

Step 4

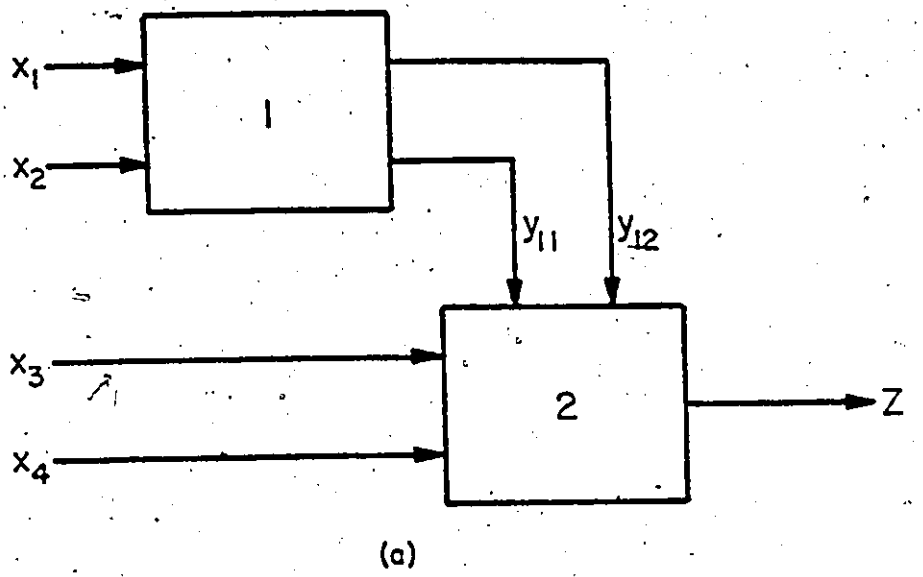
The above procedure yields a partition with two blocks. Partition of higher order can be obtained if the procedure is repeated on the first of the two blocks, considering one control function at a time.

Example

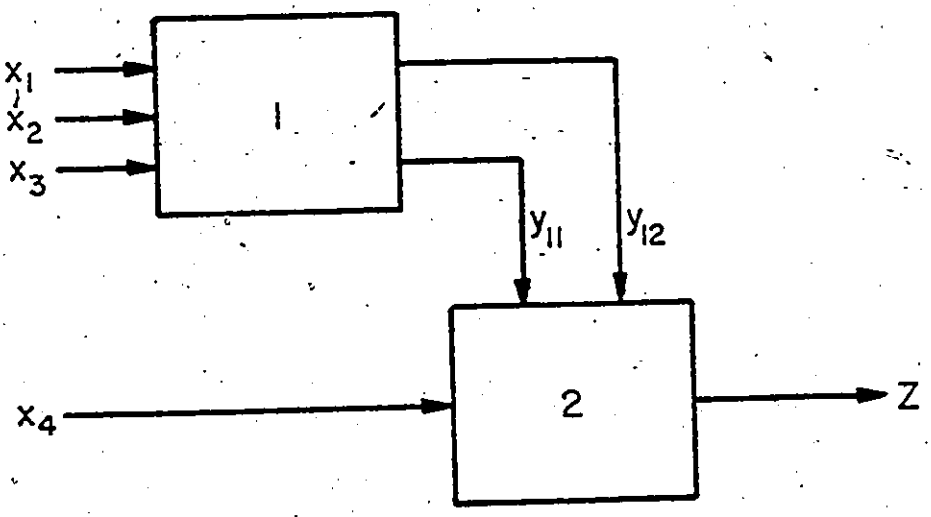
Only case (b) is considered (Fig. 5.10) since the first block in case (a) has two inputs and its partition will be trivial. Figure 5.11 shows a 3 block realization of the above example, where the function is expressed in the form:

$$Z = \bar{y}_{21}(y_{22}x_3 + \bar{y}_{22}\bar{x}_3)x_4 + \bar{y}_{21}\bar{x}_4(\bar{y}_{22} + \bar{x}_3)(x_{22} + x_3)$$
$$+ y_{21}(\bar{y}_{22} + \bar{x}_3)(y_{22} + x_3)$$

Although the 2-block realization of Fig. 5.9 (a) is merely part of a general example used as an illustration of the partition technique, for the sake of completeness a METLL realization of the two blocks is given in Fig. 5.12.



(a)



(b)

Fig. 5.9 Two-Block Realization of Partition (a) and (b)

$[(x_1, x_2) / (x_3)]$ K-map for y_{11}

x_1x_2 \ x_3	00	01	11	10
0	1	1	1	0
1	1	1	1	0

C-map

	S_1	S_2
	11	
	01	
x_1x_2 \ x_3	00	10
0	1	0
1	1	0
assignment y_{21}	0	1

$y_{21} = x_1 \bar{x}_2$

In $S_1 : y_{11} = 1$

In $S_2 : y_{11} = 0$

$y_{11} = \bar{y}_{21}$

$[(x_1, x_2) / (x_3)]$ K-map for y_{12}

x_1x_2 \ x_3	00	01	11	10
0	1	0	1	1
1	0	1	0	0

C-map

	S_1	S_2
	10	
	11	
x_1x_2 \ x_3	01	00
0	0	1
1	1	0
assignment y_{22}	1	0

$y_{22} = \bar{x}_1 x_2$

In $S_1 : y_{12} = x_3$

In $S_2 : y_{12} = \bar{x}_3$

$y_{12} = y_{22} x_3 + \bar{y}_{22} \bar{x}_3$

Fig. 5.10 K-Map to C-Map Transformation

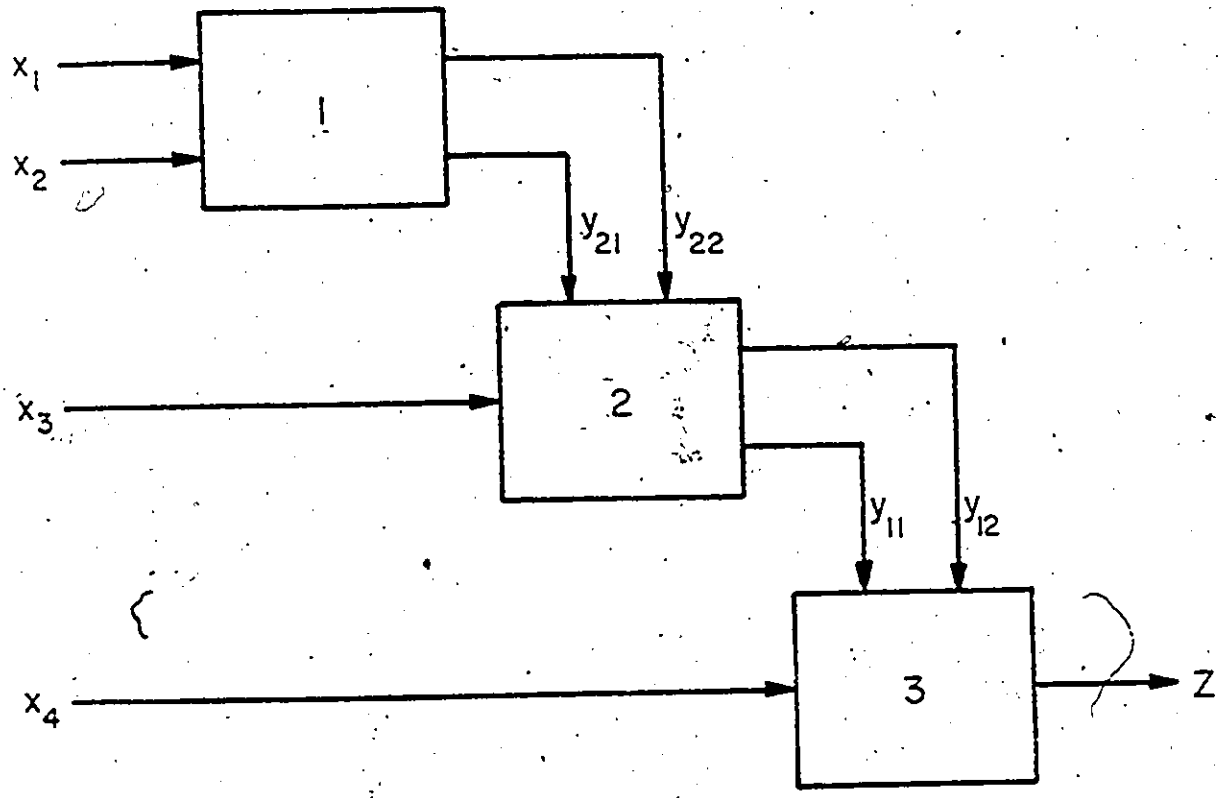


Fig. 5.11 Three–Block Realization

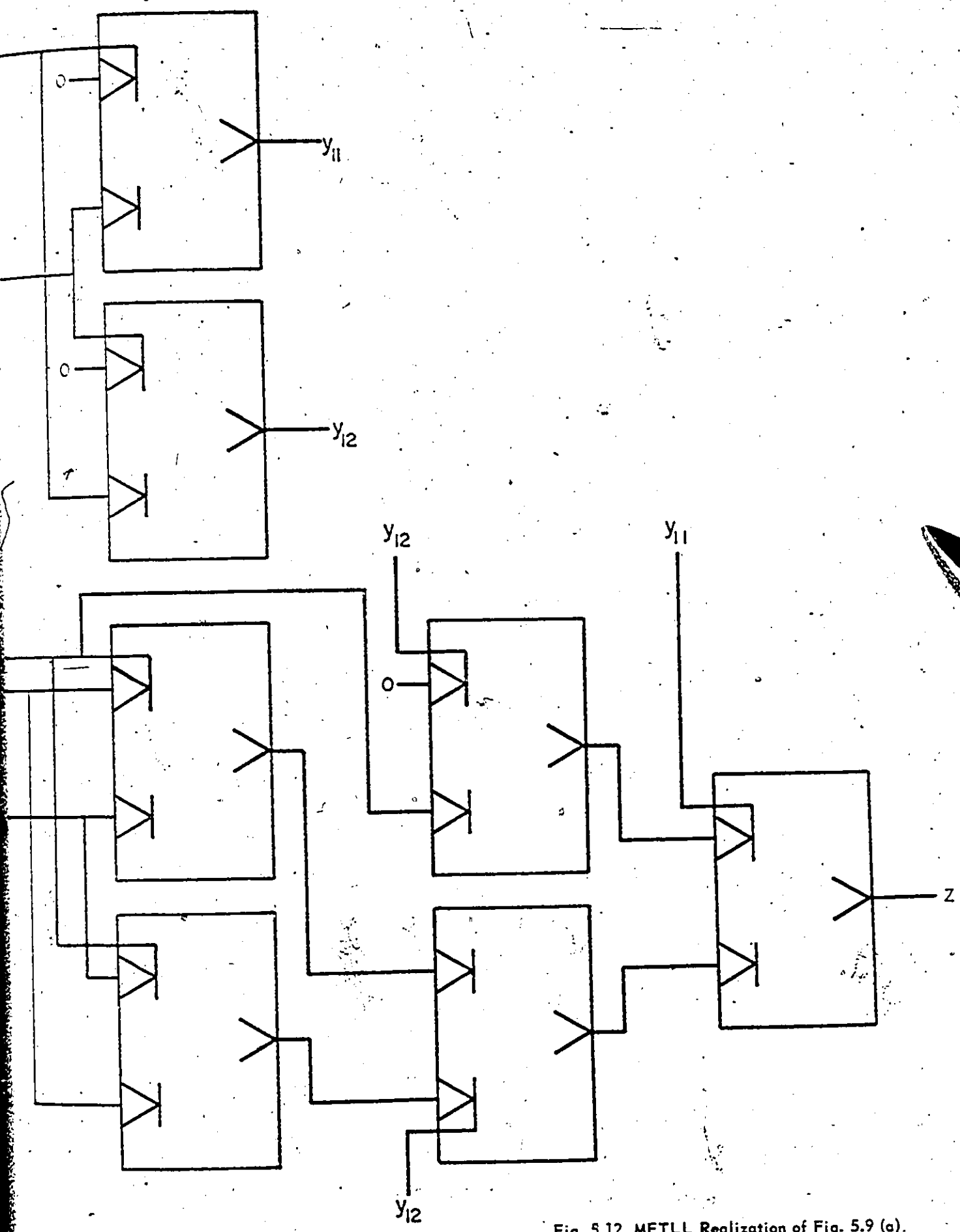


Fig. 5.12 METLL Realization of Fig. 5.9 (a).

5.5 DESIGN EXAMPLES

The partition procedure given above can be extended to functions of any number of variables using any number of partitioning steps. However, it becomes harder to perform the partition for a large number of variables, without a computer program. However the usefullness of the procedure can be determined from its application to some well-known examples, such as the design of carry stages for high-speed ripple-through adders. The problem of finding the time sequence of the input variables, in order to perform the appropriate partition, is discussed using an iterative multiplier module as an example.

5.5.1 High-Speed Full Adders

The time taken by an arithmetic operation depends not only on the speed of the basic logic circuit but also on the logic organization to suit that circuit e.g. there are several ways of reducing the time required for addition of two n-digit numbers.

1) The carry logic can be separated from the sum logic in a ripple-through system and the number of delays per stage required for the propagated carry reduced to a minimum [5].
2) Anticipated or "look-ahead" carry techniques may be applied [6]. However these become complicated and require large fan-out numbers if the numbers of 'look-ahead' stages is large.
3) There are compromise "look-ahead" techniques in which the carry is bridged over small number of stages and large fan-out numbers are avoided with significant reductions in adding time.

The two level METLL structure provides a convenient method of combining techniques (1) and (3) above. A typical carry logic for bridging two stages is shown in Fig. 5.13. As shown, the system is in the general form of Fig. 5.5 and can be realized efficiently with METLL circuits. Figure 5.14 gives the truth table and a METLL realization for the carry logic of each stage.

5.5.2 An Iterative Multiplier Module

In the above example, the sequence of arrival of the input variables is determined by inspection. A more complicated system is chosen here to illustrate a method of estimating the total signal delay at each point. The example given is an iterative multiplier, composed of similar modules [7]. Fig. 5.15 shows such multiplier for multipling 3-digit numbers; $h_o h_1 h_2$ and $k_o k_1 k_2$. It is composed of similar modules and in order to determine the sequence of inputs at each module, the overall system must be studied. In the multiplier, carries are propogated both horizontally and vertically. It can be shown that the maximum delay time for the carry P_5 to reach the output is given by:

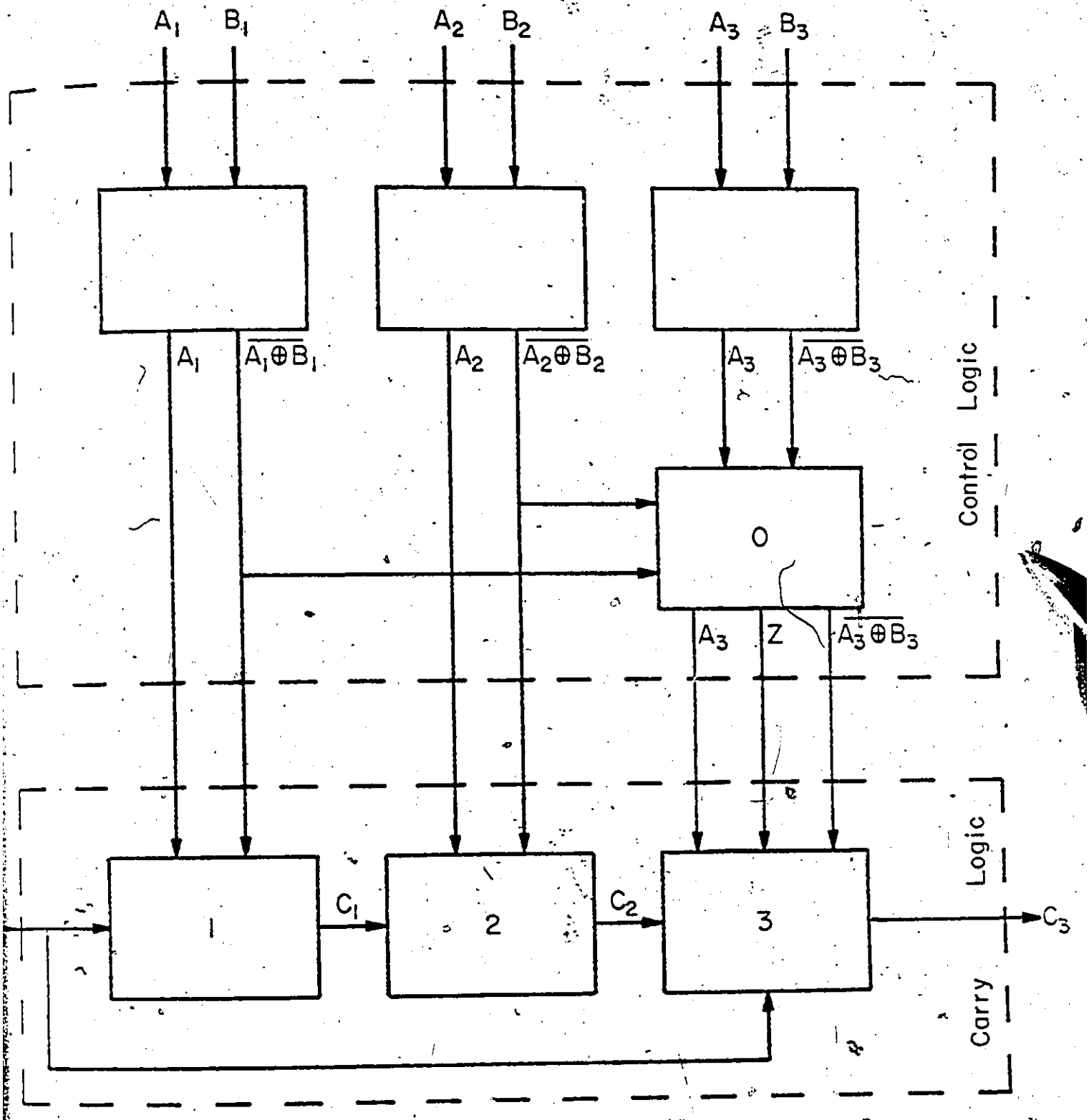
$$T = (2N-1)h + (N-1)v$$

where N is the number of digits in each number (3 in this example),

h is the horizontal delay for each module,

v is the vertical delay for each module.

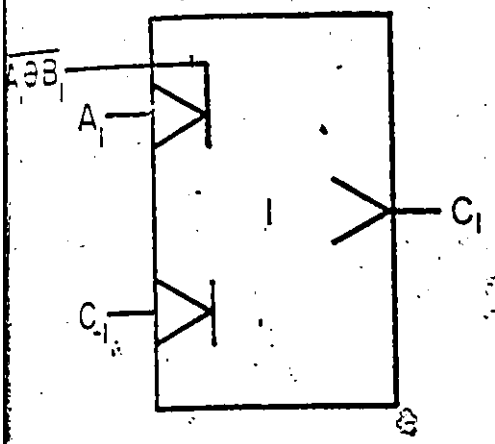
Thus the horizontal delay is more critical than the vertical one and the full adder of each module should be designed for fast horizontal carry z_1. The module now becomes an AND gate with full adder. The full adder has two control inputs (x_1 and x_2) which are available before the direct input x_3 (Fig. 5.15).



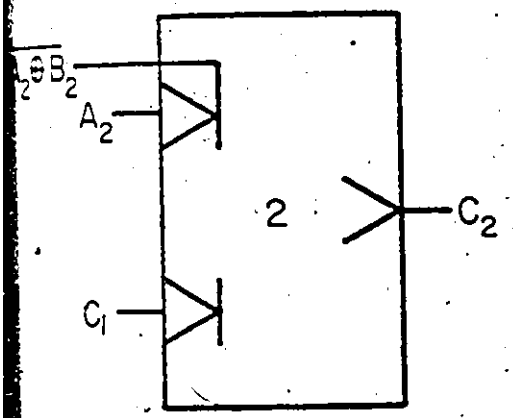
$$Z = (A_3 \oplus B_3)\, R$$

$$R = (A_2 \oplus B_2)\,(A_1 \oplus B_1)$$

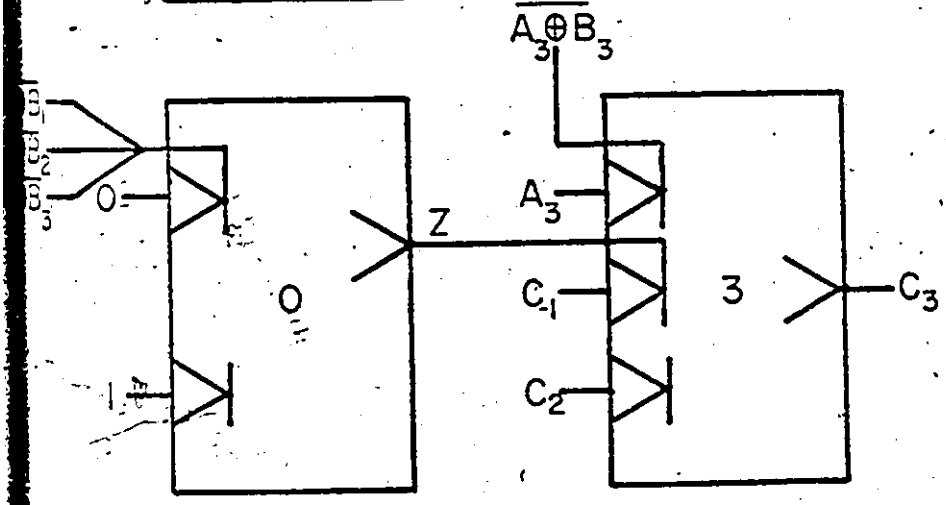
Fig. 5.13 Two-Stage Bridging of a Full-Adder



$A_1 \oplus B_1$	A_1	C_1
0	0	A_1
0	1	A_1
1	0	C_{-1}
1	1	C_{-1}



$A_2 \oplus B_2$	A_2	C_2
0	0	A_2
0	1	A_2
1	0	C_1
1	1	C_1

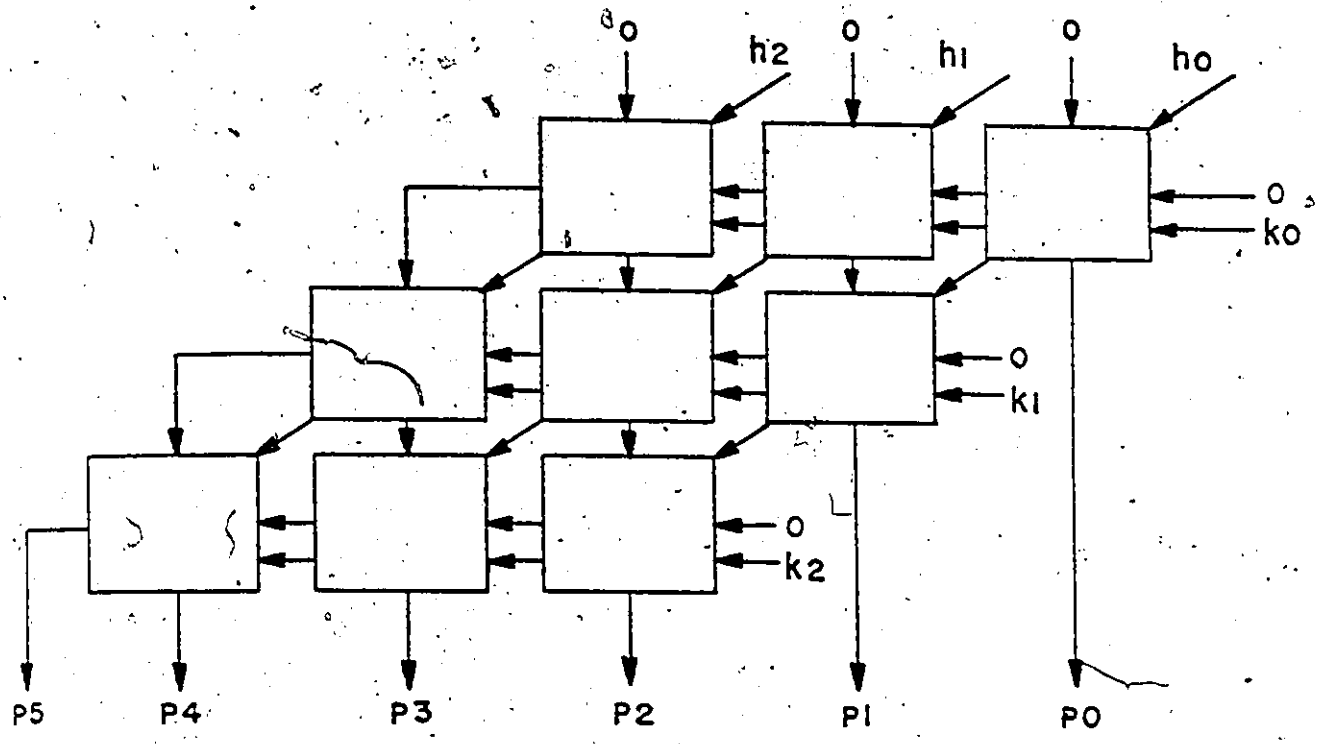


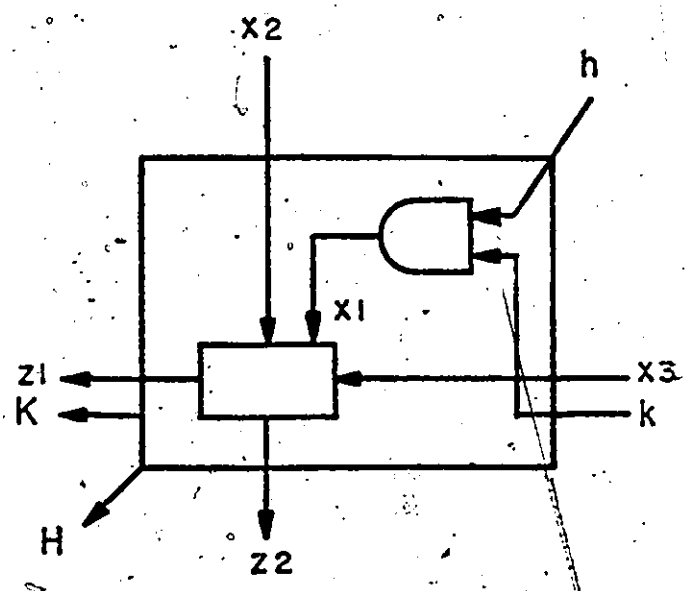
$A_3 \oplus B_3$	R	Z	C_3
0	0	0	A_3
0	1	0	A_3
1	0	0	C_2
1	1	1	C_{-1}

$$Z = (\overline{A_3 \oplus B_3})\, R$$

$$R = (\overline{A_2 \oplus B_2})\,(\overline{A_1 \oplus B_1})$$

Fig. 5.14 METLL Realization of the Carry of Fig. 5.13





FOR MULTIPLIER CELLS

X1 = hk
X2 = PREVIOUS SUM
X3 = PREVIOUS CARRY
H = h
K = k
Z2 = (X1 ⊕ X2 ⊕ X3)
Z1 = CARRY OF (X1 ⊕ X2 ⊕ X3)

Fig. 5.15 Modular Multiplier

5.6 CONCLUSIONS

In this chapter, we have established the basis for logic theory for METLL by developing logic properties, logic design and logic partition techniques. Future research work could include the following:

(1) The logic properties given are for the simplest structure. Similar properties could be established for multi-input structures and a logic design technique could be developed for multi-input realization.

(2) The logic design technique is developed to be used for functions expressed as the sum of products. Similar techniques could be developed if the function is expressed as the product of sums. A computer program could be written based on the technique given and an optimization procedure could be included.

(3) Functions of n-variable could be studied for 1-realizability and the ratio of 1-realizable functions to the total functions will be established. This 1-realizability might be for the simplest structure or for a more complex one.

(4) The criteria for doing the partition given in Section 5.4 could be changed to adopt any design requirement. For example, the partition could be done to result in 1-realizable blocks. A computer program could also be written to perform such partitioning with the minimum number of blocks.

CHAPTER 6

METAL STRUCTURES IN LOGIC-IN-MEMORY ARRAYS

METLL STRUCTURES IN LOGIC-IN-MEMORY ARRAYS

As indicated in chapter 2, one approach to obtain high-speed processing is to combine logic and memory functions in basic building blocks. These blocks are arranged in regular arrays, commonly called Logic-in-Memory (LIM) arrays [1-7]. These arrays may be regarded either as a logically enhanced memory array or as a logic array whose cells can be programmed to realize a desired logical function. The main advantage of these arrays is their suitability for LSI, because of the highly modular structure of the arrays and because digital machines can be built using few types of these arrays [8]. Moreover, these arrays offer the following features:

(1) Function Flexibility

Figure 6.1 shows a general form of a basic LIM array, which can be used in information transfer and information processing operations. It consist of N columns and M rows and each cell is labeled, referring to the row and the column number respectively. The cell has two sets of inputs, control inputs and information inputs. The control inputs determine the mode of operation of each cell and the information inputs carry the data to be proceesed. The control input set is partitioned into two sets, named horizontal and vertical. Similarly, for the input and output information. Two inter-connections are assumed for each of the surrounding cells. The control inputs of the boundary cells are connected to an instruction matrix and the information inputs to a boundary matrix.

The function of each cell, and hence the function of the array, can be changed by changing the contents of the

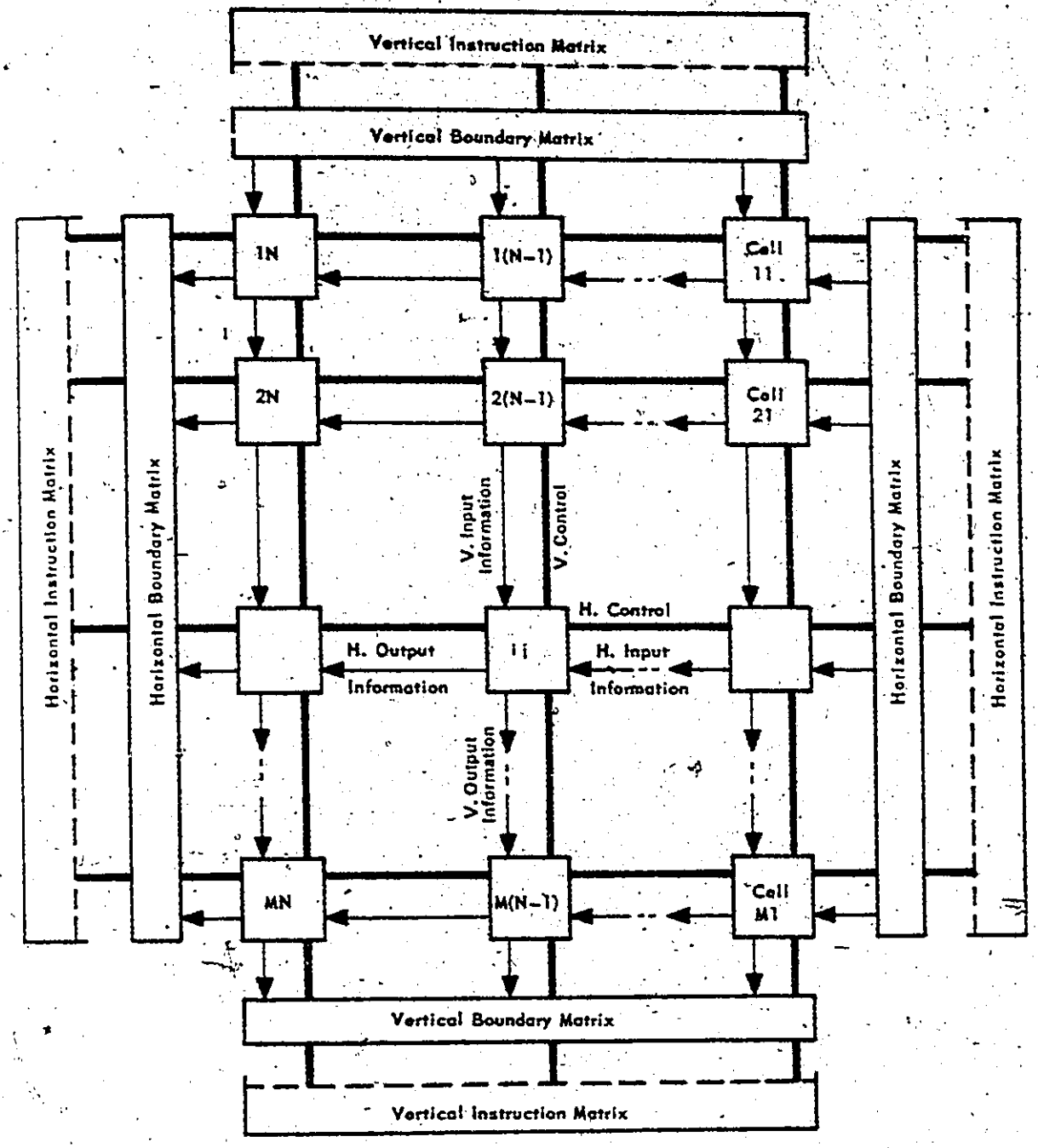


Fig. 6.1 General Form of LIM Array Using METLL Structures

instruction matrix. Thus, a cell that behaves as a full adder in one mode, may act in other modes as a counter stage or a shift register stage.

(2) Testability

Because of their regular structure, LIM arrays should be substantially easier to test for faults and to diagnose than constructured networks having the same number of elements [10, 11].

(3) Fault Accommodation

Because of the flexibility explained in (1), isolated faulty cells in an array may often be bypassed by programming the adjacent cells to ayoid active connection to the faulty cell [9, 13].

(4) Subarray Interconnectability

A cellular array may be realized as a macrocellular interconnection of cellular subarrays, each realized on an LSI chip. Since each cell has only a small number of connections to adjacent cells, the interconnection pattern on the chip is simple. Moreover, the number of external inputs-outputs that are connected to the edges of an array or subarray is relatively small.

(5) Low Power and High-Speed

Because of the low fan-in and fan-out requirements, the cells can operate at low power levels without sacrificing the high speed operation offered by the technology used for cell realization.

(6) Insensitivity to Improvement in Technology

The future availibility of more advanced technology allowing higher circuit density per chip e.g. Isoplanner II, does not imply a complete redesign of an array-organized digital system but requires only that a smaller number of subarrays (chips) be used in building up full arrays.

(7) Simplified Design Effort

Because each subarray consists of an iteration of a single, relatively simple cell, all design effort may be focused on the optimization of the circuitry of this cell with respect to speed, silicon area, tolerance, etc.

The capability of METLL structures to realize logic and memory functions economically results in its being suitable for LSI LIM arrays. In this chapter, two specific examples are given; the METLL realization of an information transfer array and a sorting array [14].

6.1 METLL REALIZATION OF TRANSFER ARRAYS

An information transfer array can accept parallel or serial information and convert it from one form to another. Figure 6.2 shows a part of this array and Fig. 6.3 shows METLL realization of the basic cell. The cell has two data inputs; a serial input data (SID) and a parallel input data (PID) and two data outputs; a serial output data (SOD) and a parallel output data (POD). All PIDs and PODs of a column are wired-OR to minimize the total number of input-outputs. Each cell has three control inputs; X, Y, and Z clocked with two phase-clocks; C_1 and C_2 to allow a master-slave operation. The METLL realization allows two logic operations to be performed at the master and one logic operation at the slave without increasing the total power dissipation and the silicon area.

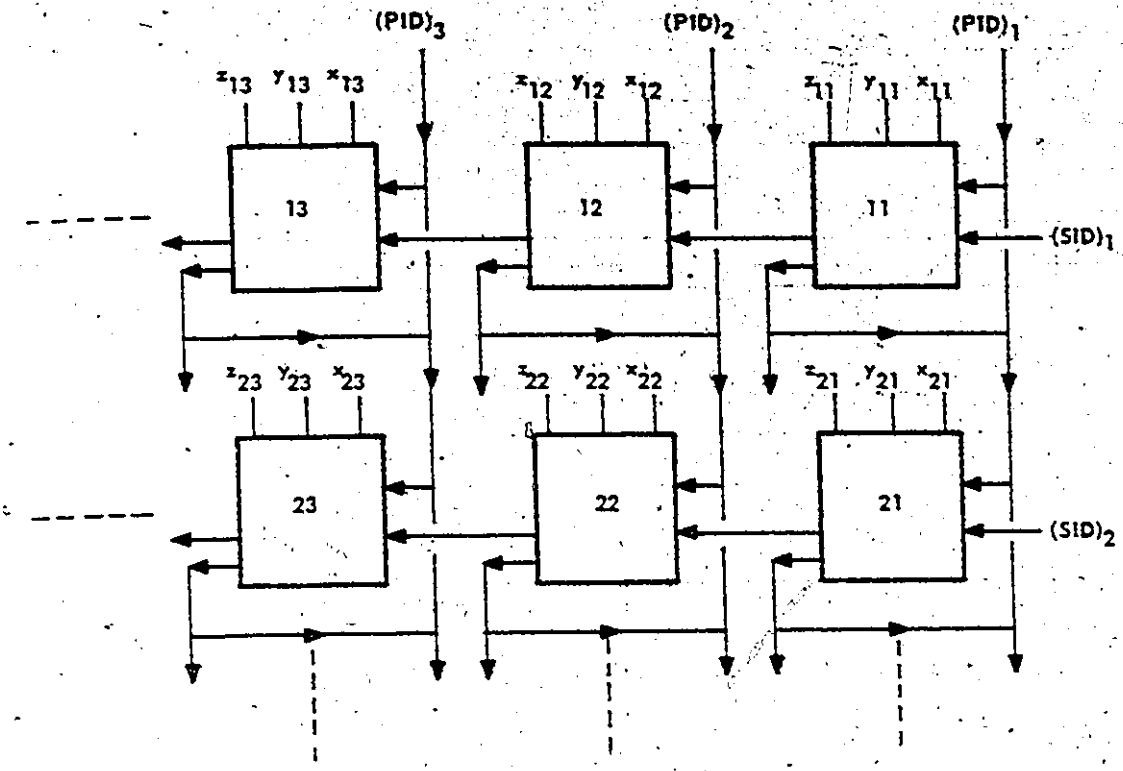
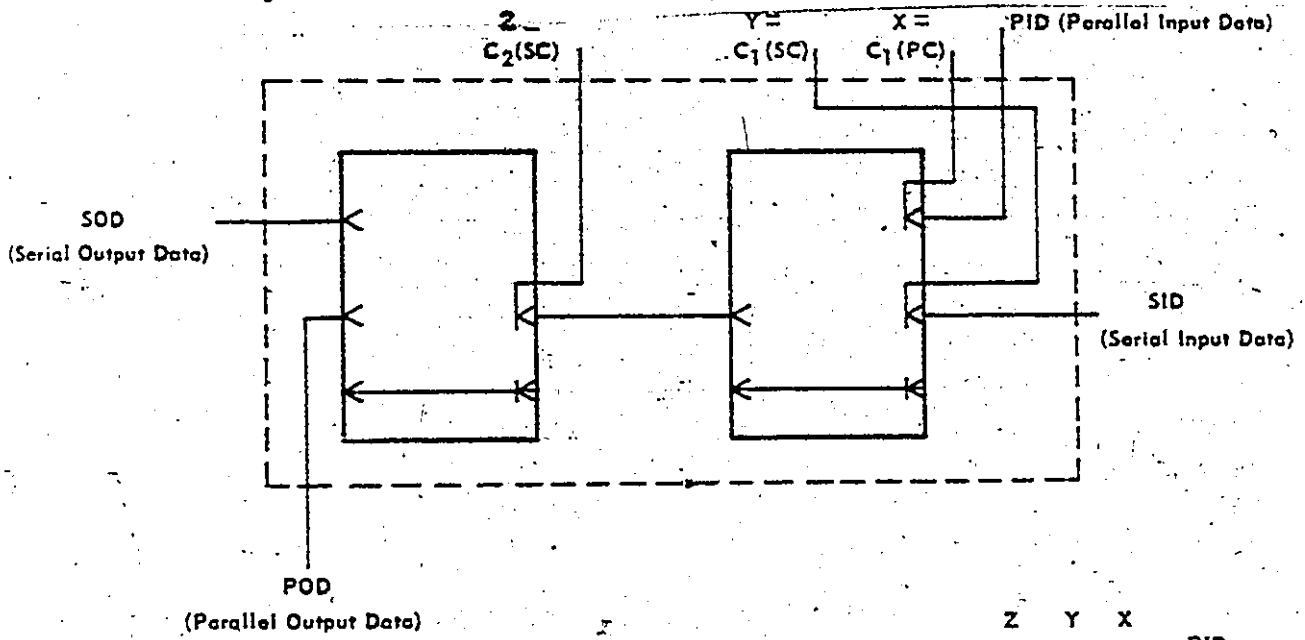


Fig. 6.2 METLL Information Transfer System



$X = C_1$. (Parallel Control) = C_1(PC)

$Y = C_1$. (Serial Control) = C_1(SC)

$Z = C_2$. (Serial Control) = C_2(SC)

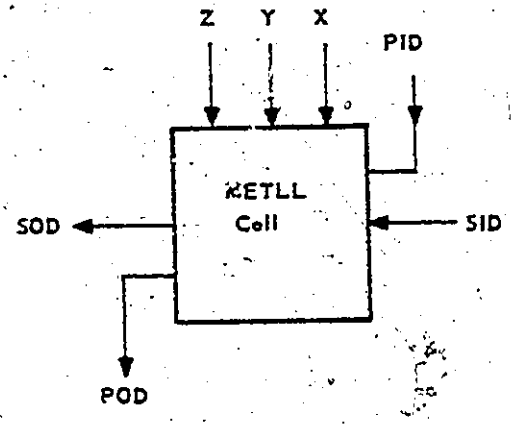


Fig. 6.3 METLL Cell for Information Transfer Array

6.2 METLL REALIZATION OF SORTING ARRAYS

A sorting array is a multiple-word memory that keeps in sorted order all data words that are fed into it. Words that are read out are obtained in order of size, with the largest word first (or alternatively, with smallest first, as desired). The words are assumed to be of maximum length n. This array can be used as a functional unit within the central processor of a general-purpose computer, or to a special purpose computing system for which sorting capability is needed.

The sorting array described here [14] is a two-dimensional iterative configuration of identical cells, each of which is a simple sequential logic circuit. Each cell is connected only to its immediate neighbors and the cells around the edges of the array are connected to fixed signals or to registers. Figure 6.4 shows the array configuration along with an input-output register X and a word selected register W. All cell input terminals on the right-hand side of the array are connected to a logical signal Z_o, (normally fixed at 1) and the outputs labeled $\hat{Z}$ from the left edge of the array serve as inputs to the stages of the W-register. Each cell contains one flip-flop, whose content is designated Y, so that the set of n flip-flops in any one of the N rows of the array may be employed to store one n-digit word. This set of words is assumed to be encoded. A binary-coded decimal number system can be used with the most significant digits at the left ends of the words. One's or two's complement representation can be used to encode negative numbers, with the minus sign encoded as a 0 at the left end of the word.

An entire word is handled as a unit during the input, output and sorting operations. One cycle of operation of the array consists of two steps:

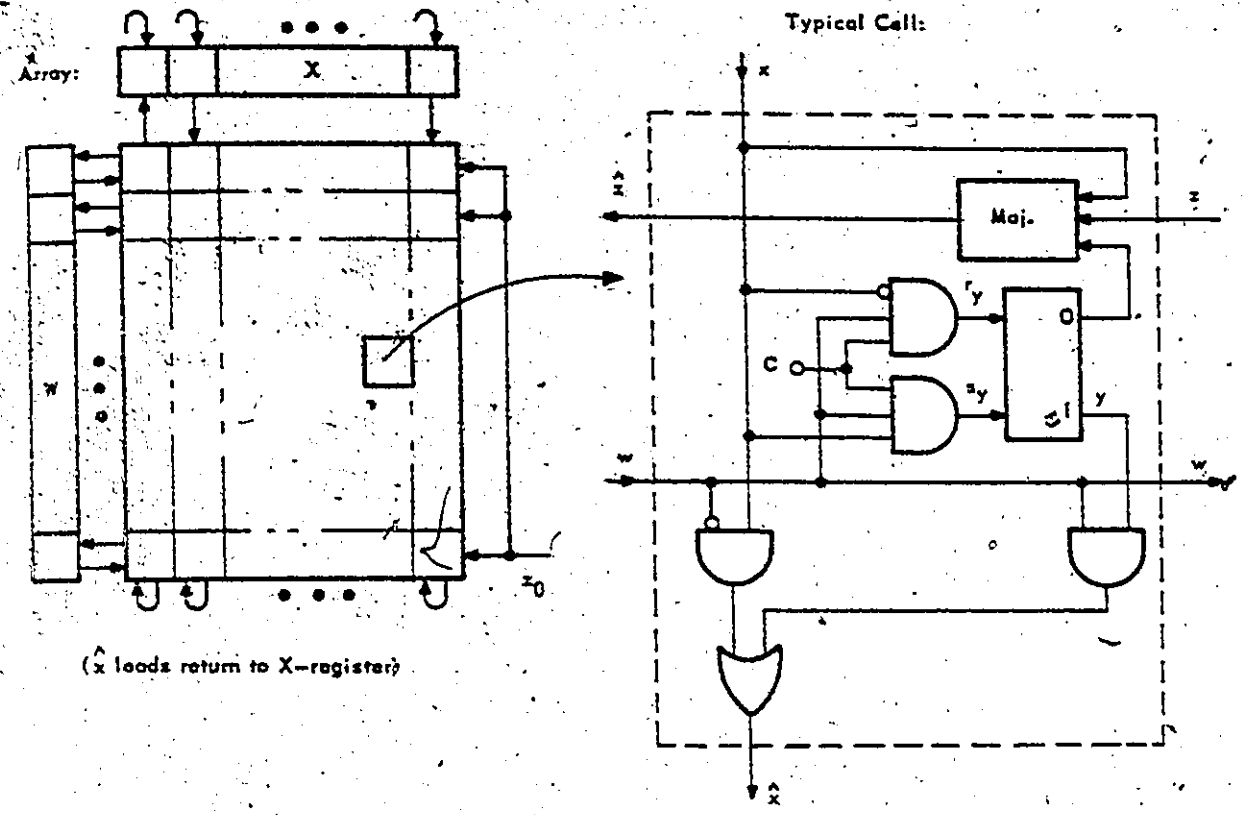


Fig. 6.4 Cellular Sorting Array

(1) A comparison step, in which the word X in the X-register is simultaneously compared with N words stored in rows of the array; a "1" is injected into the W register in those rows whose words (including blank words) are smaller than or equal to the word X, and a "0" is injected into the W-register in those rows whose words are larger than the word X.

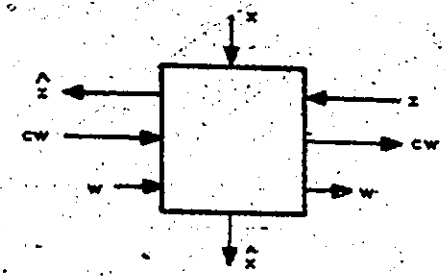
(2) An execution step, in which

(a) The set of all words that are stored in the subset or rows having a "1" in the W-register are moved downward one row within the set, while the word in the X-register is copied into the uppermost such row; and

(b) The lowermost such word is copied into the X-register. Words having a "0" in the W-register are not moved.

Sorting with this array is accomplished by maintaining a sorted file of previously entered words, with the largest at the top of the array and the smallest and any blank rows at the bottom. Each new word that is to be sorted is inserted into this file in a single operational cycle. To read out in order the words in the file, largest to smallest, place a "1" in the uppermost stage of the W-register. Now carry out step (2) of the cycle repeatedly, shifting the "1" downward in the W-register, one row with each step. If the words are desired in the opposite order, a "1" may be started at the bottom of the W-register and shifted upward.

Figure 6.5 shows the array cell and its METLL realization, four structures are used compared with 20 conventional gates. The majority gate at each cell forms part of a chain of n such gates. The inputs x and $\bar{y}$ of this gate allow it to act as a size comparator, so that the leftmost Z-output in each row takes on the value "1" when and only when the number repre-



$\hat{x} = wy + \bar{w}x$

$\hat{z} = x\bar{y} + xz + z\bar{y}$

$= \text{Maj. of } (x, \bar{y}, z)$

$y = (wcx)_{t-1}$

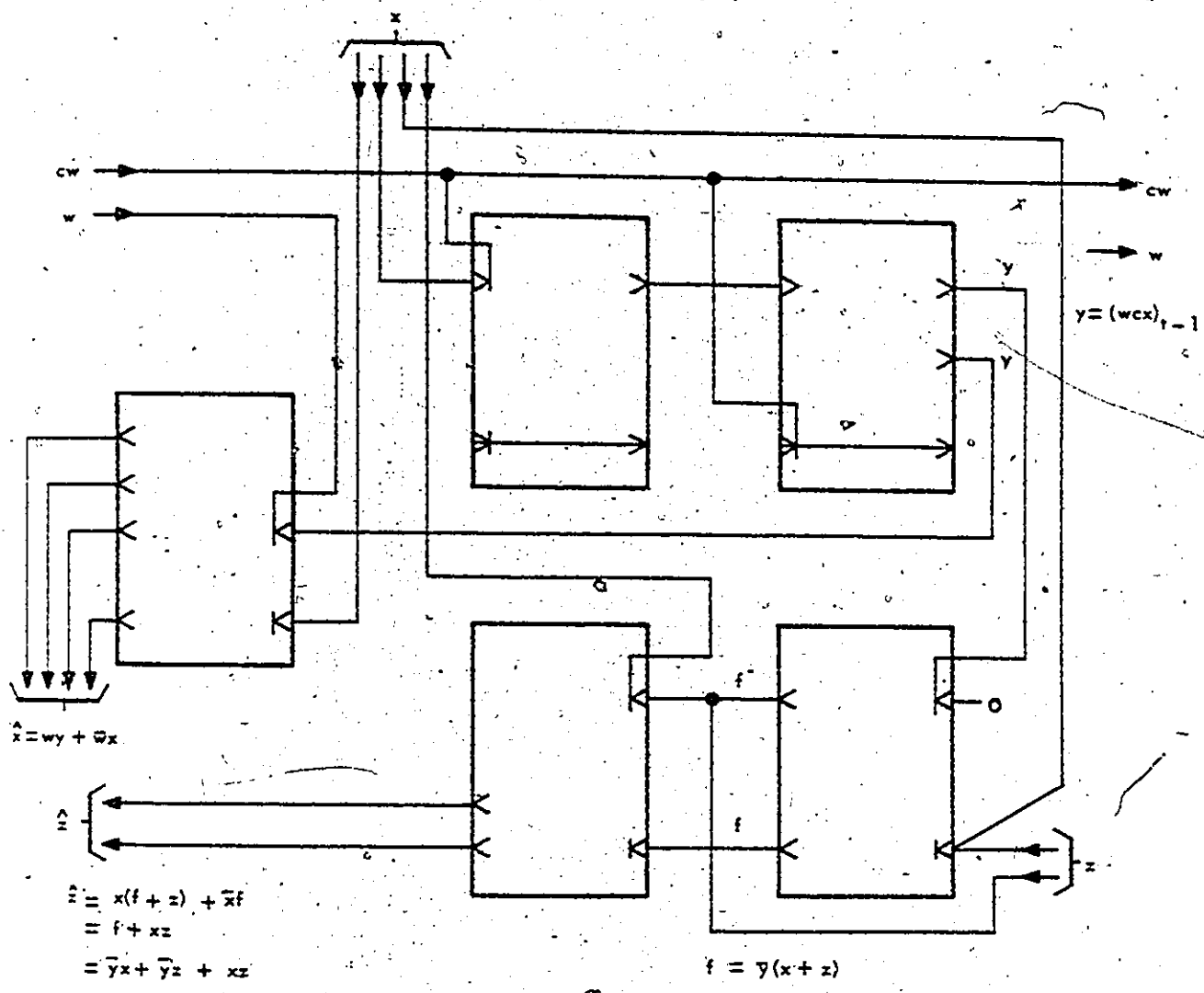


Fig. 6.5 Sorting Array Cell and its METLL Realization

sented on the set of x-lines entering this row is smaller than or equal to the number represented in the cascade of y-flip-flops in the same row. Normally, step (1) is carried out with the W-register initially empty, so that the W busses in all rows of the arrays carry the value "0". In this case the $\hat{x}$-output in each cell carries the same value as the x-input; $\hat{x} = x$, i.e. the contents of the X-register is bussed downward to all rows of the array. As a result, the comparisons in (1) are made between the word X and every word Y stored in the array.

During step (2), a row for which the W-register contains a "0", we have w = "0", so that each cell in this row behaves according to

$$\hat{x} = x \quad , \quad y = y_{t-1}$$

i.e., the row is static and behaves as if it were not even present. A row for which the W-register contains a "1", we have w = "1", so each cell behaves according to

$$\hat{x} = y \quad , \quad y = (cx)_{t-1}$$

Thus the word stored in the flip-flops in this row is transferred onto the set of x-lines that pass downward from this row. For the array as a whole, all words in the subset of rows consisting of the X-register and those rows marked with a "1" in the W-register shift down cyclically one row position within the subset. The contents of the X-register fills the top position and the contents of the lowest marked row passes back into the X-register.

The sorting array may also be used for the realization of an arbitrary logical function, as a pushdown memory (last-in, first-out) or a buffer memory (first-in, first-out) or as a content-addressed (associative) memory [15-18]. In these configurations, similiar METLL realization could also prove economical.

CHAPTER 7

CONCLUSIONS

CONCLUSIONS

This thesis has covered many aspects of digital design, from computer architecture to the circuit design of high-speed logic. Although each aspect has been reviewed in some detail, the thesis is concerned more with the relation between the different aspects than any specific one. Indeed, it is impossible to make a design change to any aspect without affecting all the others. The thesis contributed to high-speed LSI logic by the development of the METLL family. It was shown that a decision to adopt METLL required new design techniques to be developed for its most efficient use in a logic system. This was related to ways in which modularity could be exploited at the subsystem level and its impact on system architecture. The concept of modularity is important at all levels and makes the system particularly suitable for LSI realization. On the silicon chip, it permits a reduction of design time and great saving in layout cost. At the subsystem level it simplifies interconnection patterns and reduces the number of input-output connections. At the overall system level, it achieves design economy, high performance and high reliability.

Although the thesis has been specifically concerned with the need for an LSI high-speed system using conventional bipolar technology, the modular approach taken is generally applicable and it is possible to adapt the approach to other technology, e.g. I^2L (Integrated-Injection-Logic) and MOS.

It has been indicated throughout this thesis that further research work is required. The main areas for future research are summarized here:

(1) The logic design techniques given lead to the economical realization of MSI functions (full adder, multipliers). In order to adapt the METLL structures in fully LSI systems with a high degree of functional complexity, more research work must be done. Alternative minimization techniques, based on a cost function could be developed.

(2) It is possible to explore further the concept of pipeline organization as related to LIM arrays and METLL. At the time of writing, progress in that direction has been reported by the digital circuit research group at the University of Ottawa, under the direction of Professor P.M. Thompson. Similar research work has been also reported [1, 2].

(3) The logic properties of the structures as presented in chapter 5 could be proved more formally and some of the logic problems indicated could be solved.

Recent advances in technology will generate new families of logic circuits and eventually affect the over-all computer architecture of large systems. However, without knowledge of how an advance at one level affects the design decisions at another, it is possible that these advances in technology will not be properly exploited e.g. computer architecture may be studied on the basis of an obsolete technology and logic circuits may be designed to meet the requirements of obsolete system architecture. Thus the proper exploitation of a new advance at any level of design, be it an advance in logic theory, solid state technology or the design of large systems requires knowledge of inter-relationship of the various levels of digital system design.

APPENDIX 1

THE DESIGN OF A LOAD STRUCTURE FOR CURRENT-MODE SUBNANOSECOND LOGIC

For several years, current-mode logic (CML) circuit configurations have been applied where the highest operating speeds have been required. In their integrated form, CML circuits employ diffused resistors as collector loads and they are fabricated on the same Silicon chip as the other elements [1, 2]. In discrete circuits, advantages have been claimed for negative resistance load structures, e.g. tunnel diodes and tunnel diode-resistor pairs [3, 4], but they have not been integrated onto a silicon chip. It has not been proved that either the tunnel diode or the resistor provide the optimum load characteristics. When integrated circuits are used, it should be possible to synthesize an optimum load structure for any CML circuit.

Here we describe the design of an optimum static characteristic for a load structure for CML. A computer aided analysis is used to compare the transient response of CML circuits which use three specific types of load structure. It is shown that the load with the optimum static characteristics also appears to offer an optimum transient response. Practical integrated circuit realizations of the load structure are discussed and their feasibility demonstrated. The load is discussed specifically in terms of Multi-Emitter Two-Level Logic structure [5, 6, 7] because METLL has been demonstrated to give the lowest time-delay power products (2pJ for conventional and diffused n-p-n integrated circuits).

I EFFECT OF LOAD STRUCTURE ON THE STATIC CHARACTERISTICS OF CML CIRCUITS

The simple OR gate shown in Fig. 1 is chosen for the study of the effect of the load structure on CML circuits. The circuit performs the function, $Z = A + B$. The operation of the circuit is as follows: Input transistors (Q_1, Q_2) and the fixed bias transistor (Q_0) control the flow of the current I_0 through the load. If the base of either Q_1 or Q_2 is positive of the reference voltage V_R, no current will flow in the collector of Q_0 and the output voltage will be zero, i.e. a logical "1" in positive logic system. Only when both inputs A and B are negative of V_R and I_0 flows through Q_0, the output voltage will be negative, i.e. a logical "0". The output node is connected to inputs of other stages which operate in the same way.

The transfer characteristic ($V_{in} - V_{out}$) of the above logic circuit is determined by the following:

(a) The relationship between the input voltage (V_{in}) and the load current (I_L).

(b) The relationship between the load current (I_L) and the output voltage (V_{out}).

The $V_{in} - I_L$ relationship is function of the current source I_0 and the characteristics of the input transistors pair Q_1, Q_2 and the fixed bias transistor Q_0. Assuming identical transistors and a constant current source I_0 [8] the load current can be expressed as

$$I_L = \alpha_0 I_0 / \left[1 + m_d \exp (V_{in} - V_R)/V_T\right]$$

where

α_0 = common base short circuit current gain

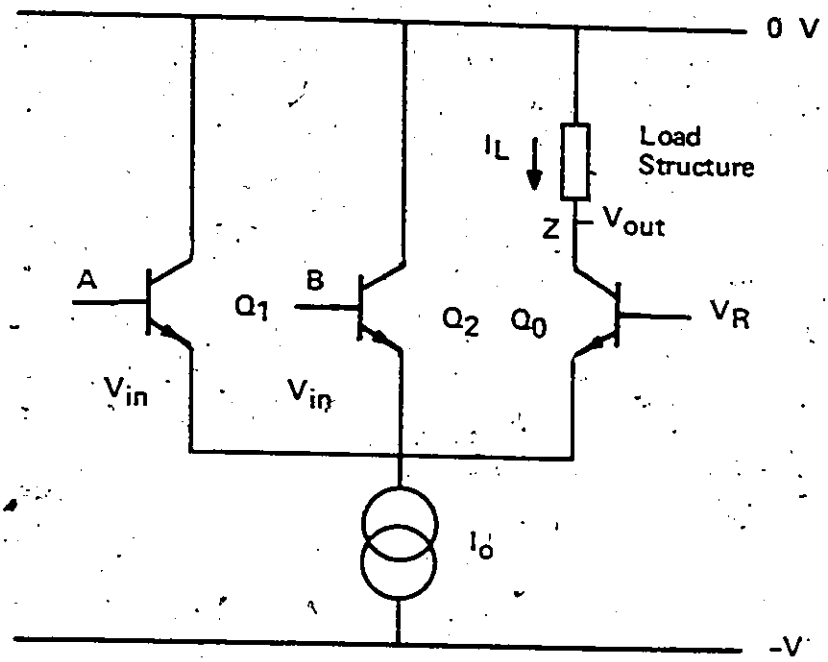


FIG. 1 CURRENT MODE OR LOGIC

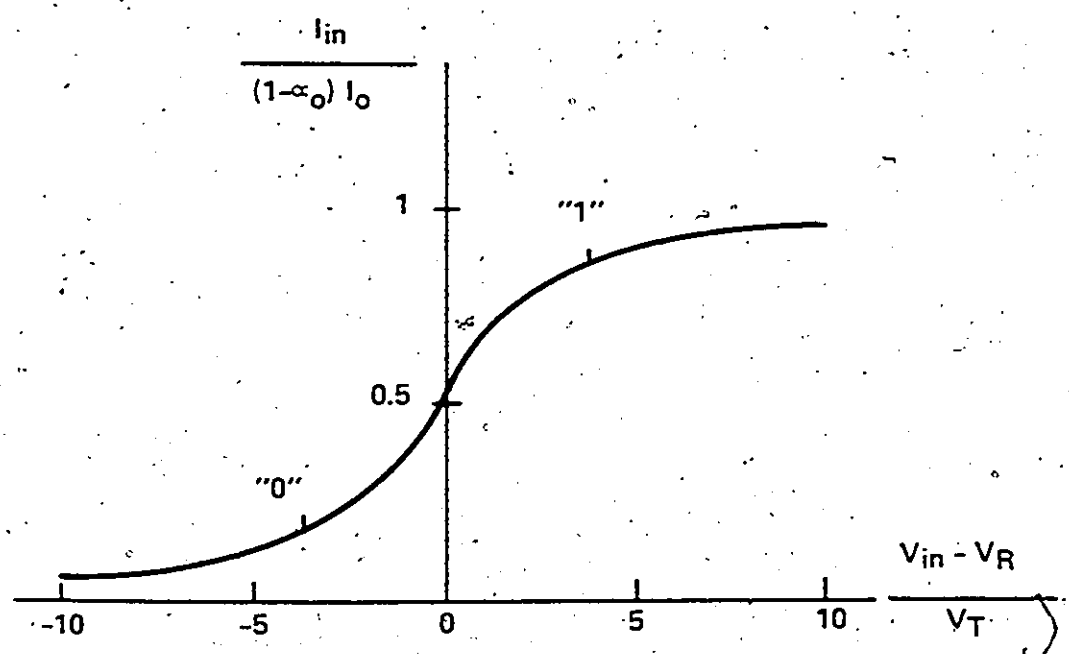


FIG. 2 CML INPUT CHARACTERISTIC

m_d = number of driven transistors
V_T = thermal voltage $= \frac{KT}{q}$ $\sim$ 25 mV at room temperature

The $I_L - V_{out}$ relationship is function of the characteristic of the load structure and the input characteristic of the next stage. Figure 2 shows the input current input voltage characteristics, where

$$I_{in} = (1 - \alpha_0)\, I_0 / \left[1 + \exp(V_R - V_{in})/V_T\right]$$

If the combined impedance of the load structure and the input of the next stage is high, small changes in current will result in significant changes in the output voltage and ill-defined logic levels. Since the input of CML circuits presents a high impedance at logical "1" and logical "0" levels, the load structure should present a low impedance at these two levels. Moreover, since the input characteristic has a minimum impedance when $V_{in} = V_R$, the load structure should have a high impedance or a negative impedance to minimize the current through the load and allow the driving transistor Q_0 to supply the maximum input current to the driven transistors of the following stages. As a result, the load structure must satisfy conflicting requirements, namely a low impedance at the logic levels "0" and "1" and high impedance or negative impedance when $V_{in} = V_R$. If a linear resistance load structure is used, its value must be a compromise between the two requirements. Alternatively, the conflicting impedance requirements can be met by non-linear structures.

Figure 3 shows a stylized collector characteristic of transistor Q_0 with three load lines superimposed:
(a) A linear diffused resistor.

(b) A non-linear negative resistance with low positive resistance limits at the logic levels "0" and "1".
(c) A non-linear infinite resistance with low positive resistance limits at the logic levels "0" and "1".

The transfer characteristics of a CML circuit with these load lines is shown in Fig. 4 and the resultant logic and threshold level system [9, 10] are shown in Fig. 5. For case (a), the linear load line, the overall transfer characteristic is that of the transistor pair (Q_1, Q_0). This results in a wide threshold and low noise margins. In case (b), the noise margins are increased because the collector current of Q_0 changes through almost its complete range, before the output voltage changes and the sum of the two noise margins is greater than the logic swing (see dotted line (b) in Fig. 4). This introduces a bistable characteristic into the logic circuit which can no longer be regarded as entirely combinational. In case (c), the high impedance section of the load line introduces a steep slope into the transfer characteristic and results into a narrow threshold region. Both noise margins are increased without the introduction of a bistable characteristic.

The output characteristics (I_{out} - V_{out}) for the three load structures are compared in Fig. 6. Two sets of curves are shown. The thin lines represent the case where I_c of Q_0 = I_0 and the correct output is logical "0". The thick lines represent the case where I_c of Q_0 = 0 and the correct output is logical "1".

The output current is defined as positive when it flows into the output node. The output voltage remains within its correct range of values for larger output currents for the nonlinear load lines cases (b) and (c). However, in case (b) for the negative resistance load line, a small output current

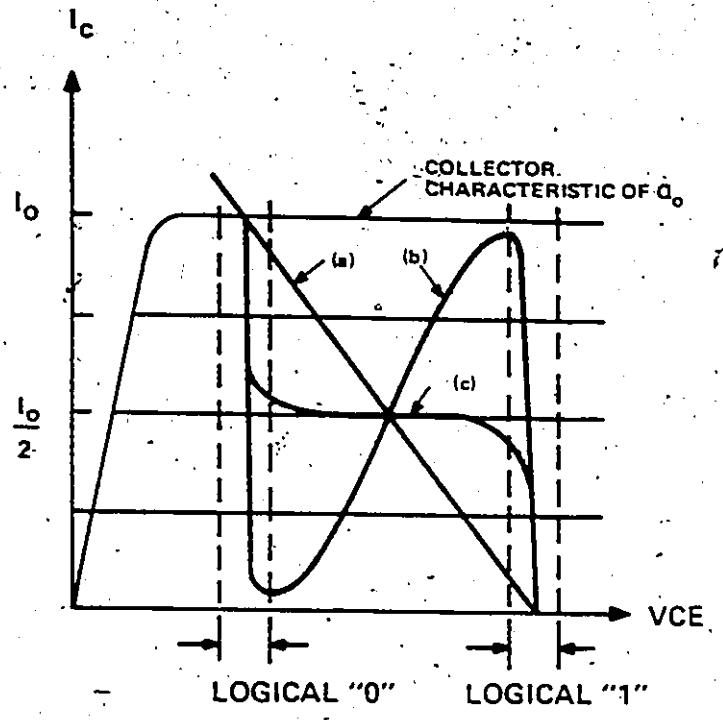


FIG. 3 LOAD STRUCTURES (a), (b), (c)

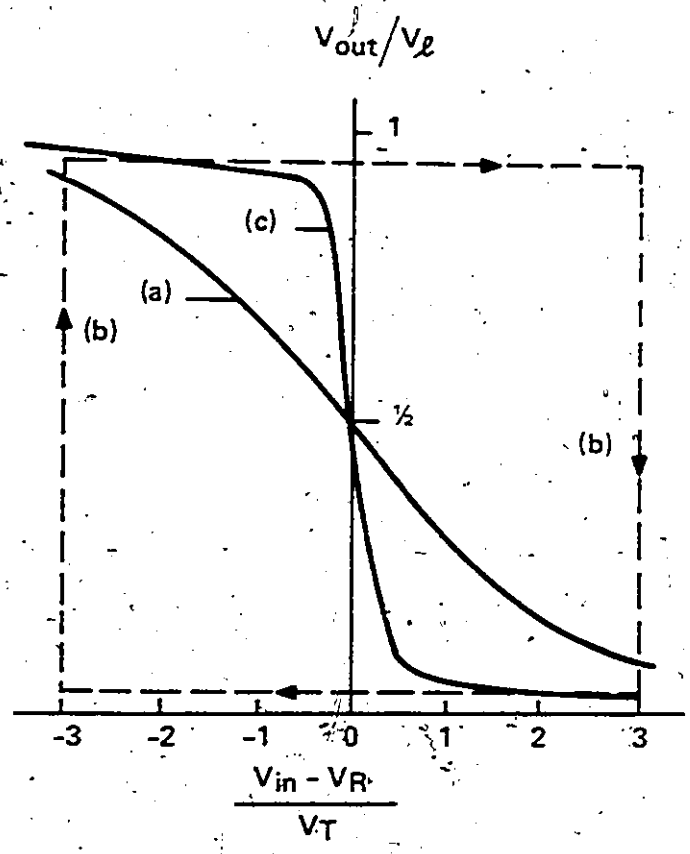


FIG. 4 CML TRANSFER CHARACTERISTICS
V_ℓ = logic swing = V"1" - V"0"

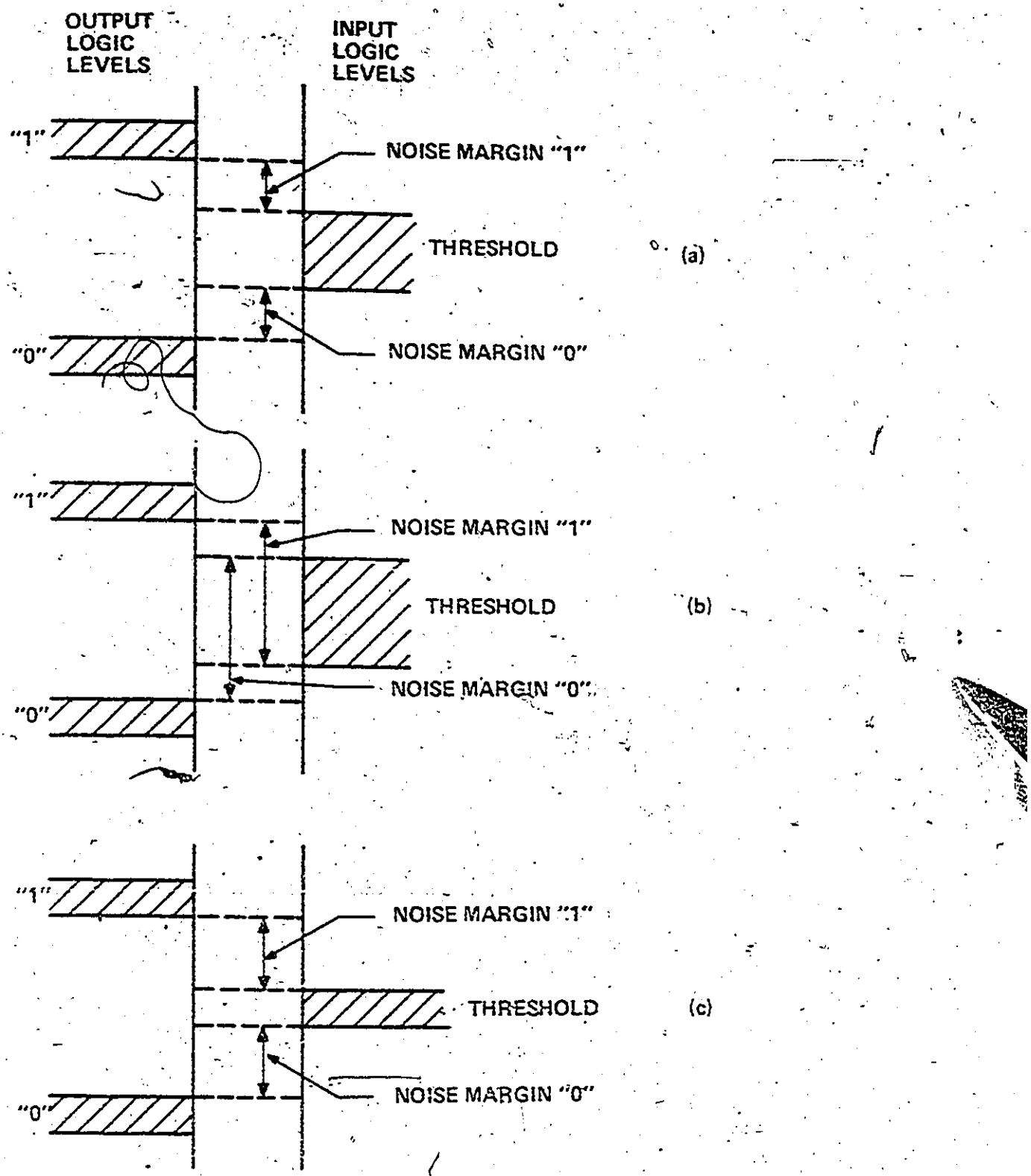


FIG. 5 COMPARISON OF LOGIC LEVELS AND NOISE MARGINS FOR LOAD STRUCTURES (a), (b), (c)

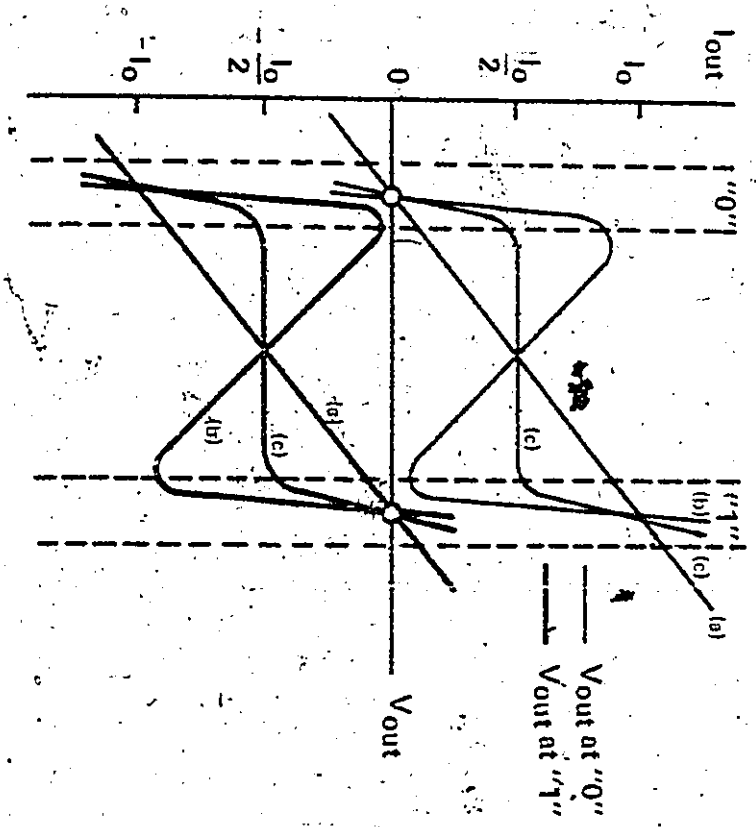


FIG. 6 CML OUTPUT CHARACTERISTICS

can prevent the output voltage from changing between logic "1" and "0" when I_c changes. Thus the preferred load line for static performance is case (c).

II EFFECT OF LOAD STRUCTURE ON THE TRANSIENT RESPONSE

Computer similation has been used to study the relationship between the transient response of CML and the static load line. Figure 7 shows simplified equivalent circuits for the three types of load structure evaluated. The diffused resistor of Fig. 7a is assumed linear and the tunnel diode characteristic of Fig. 7b is assumed to consist of three linear sections. The junction diodes of Fig. 7c are assumed to have exponential I/V characteristics and a 500 mV threshold. For reasons of convenience, the logic swing is arbitrarily taken as 1 V and the logic levels as +3.5V and +2.5V. It is assumed that the CML circuit is required to charge the bases of the next stages and various stray capacitances. Also, the total output capacitance does not differ between the types of loads and has a typical value of 0.5 pf. The collector current I_c is of the form of a linear ramp [11] and the difference between it and the load current is available to charge and discharge the output capacitance.

The results, for transient response, of a non-linear analysis program [12] are shown in Fig. 8. For all three types of load, the initial delay is directly related to the noise margin derived in Section I. This delay is shortest for the positive linear diffused resistor and longest for the negative resistance. In the case of the linear resistance load, the output voltage has a rapid initial rise, but is exponential and approaches its final value slowly. However, the rise time is dependent on the definition of the logic levels and can be shortened by increasing the margins of the logic levels and

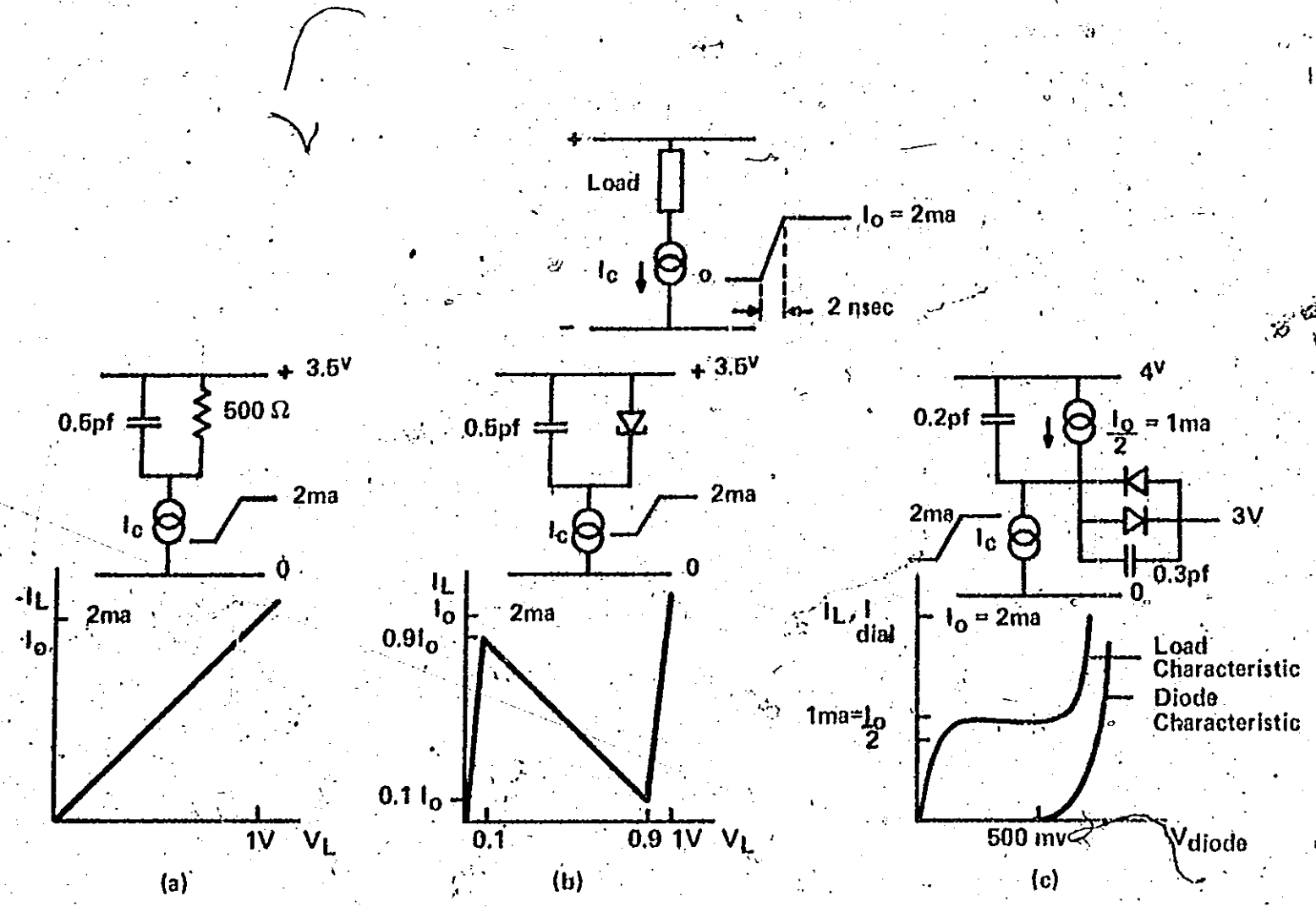


FIG. 7 SIMPLIFIED EQUIVALENT CIRCUITS

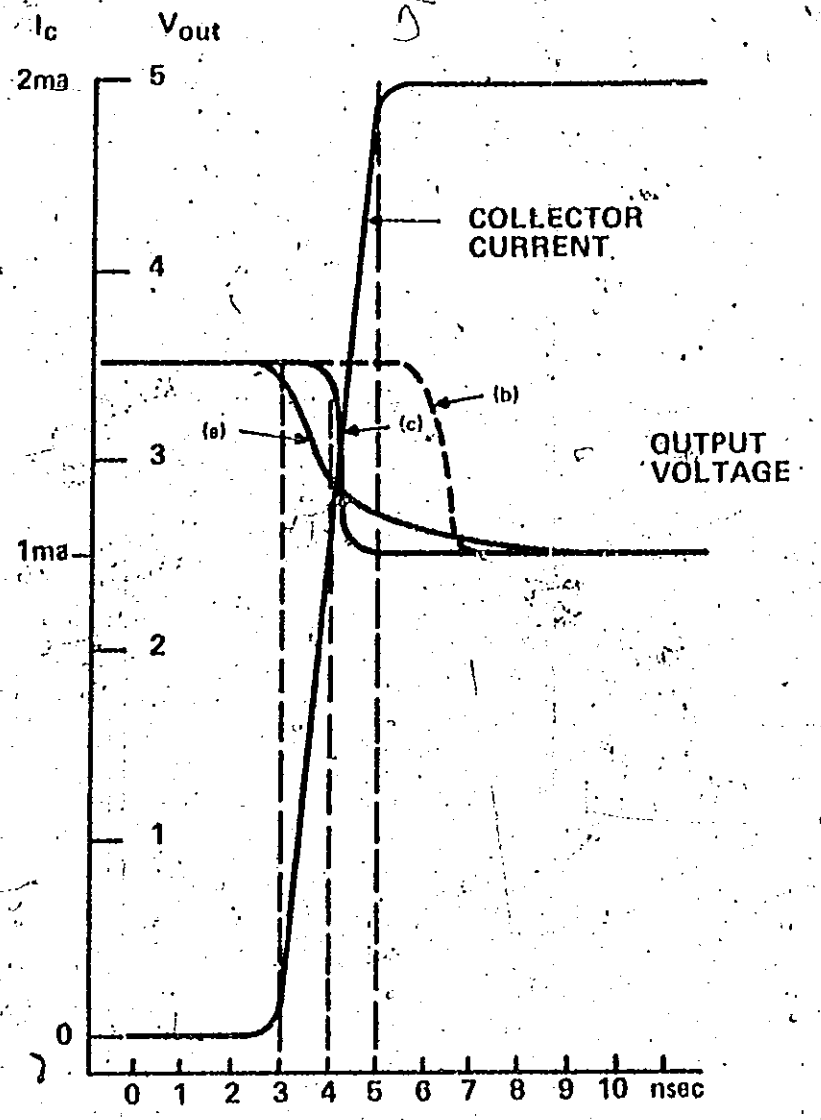


FIG. 8 CML TRANSIENT RESPONSE

reducing the noise margins. For the negative resistance load structure, the output voltage has a slow initial rise before it is accelerated by the negative resistance characteristic.

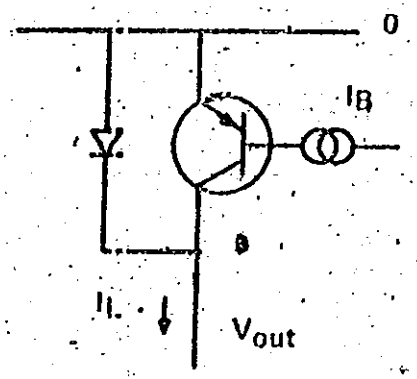
The load structure with the infinite resistance section, provides the fastest overall transient response. The initial delay is not very much longer than that of the diffused resistor and the switching transient is essentially linear. Thus this structure, which has the preferred static characteristics, also offers the best overall transient characteristics.

III INTEGRATED REALIZATION OF THE INFINITE RESISTANCE LOAD STRUCTURE

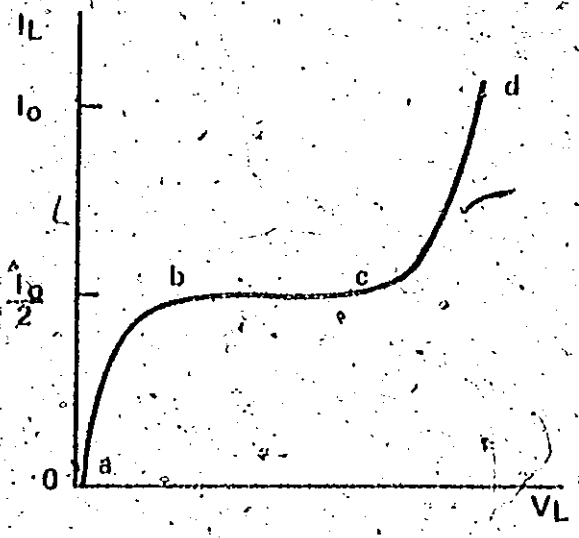
The infinite resistance load structure can be realized in a number of ways and some of these are discussed in (a), (b) and (c) below:

(a) Lateral p-n-p Transistor - Diode Circuit

If conventional diffused n-p-n technology is assumed for the basic CML circuit, the load characteristic shown in Fig. 3 (c) can be realized by a parallel connection of a lateral p-n-p transistor [12, 13] and a junction or a Schottky diode as shown in Fig. 9 (a),(b). The transistor is biased so that its collector current in the active region, $I_c \doteq I_0/2$. Then the section abc of the characteristic of Fig. 9 (b) is determined by the transistor and cd by the diode. If a junction diode is used the difference between the logic levels defined by this circuit is 650 mV, and if a Schottky diode is used, the difference is 350 mV. It is normal to integrate the diode as part of the basic CML structure and to use a single multiple-collector p-n-p transistor for general load structures. Such



(a) Circuit



(b) DC Characteristic

FIG. 9 LATERAL p-n-p - DIODE STRUCTURE

a transistor and its bias circuit is shown in Fig. 10 (a), (b). Q_3 defines a current I_0 as in the basic CML circuits while the emitter coupled pair Q_1, Q_2 amplifies any difference between I_0 and the current supplied by two collectors of the p-n-p transistor Q, to provide the base drive at which each collector current will be $I_0/2$.

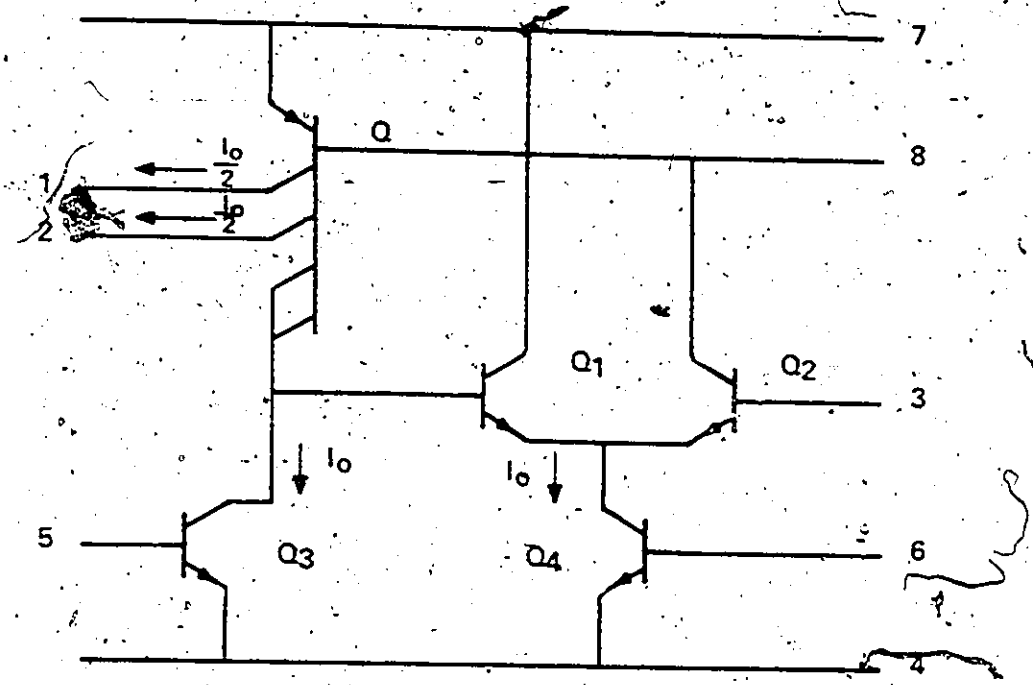
Although this circuit provides the correct static load-line, it is not suitable for sub-nanosecond logic circuits. When the transistor is in the region ab of Fig. 9 (b), minority carriers are stored in the base region at the boundary of the collectors. When I_c of Q_0 in the CML circuit switches from I_0 to 0, the output voltage can not change from logic "0" to logic "1" until these excess carriers have been collected.

(b) Current Source with Two Diodes

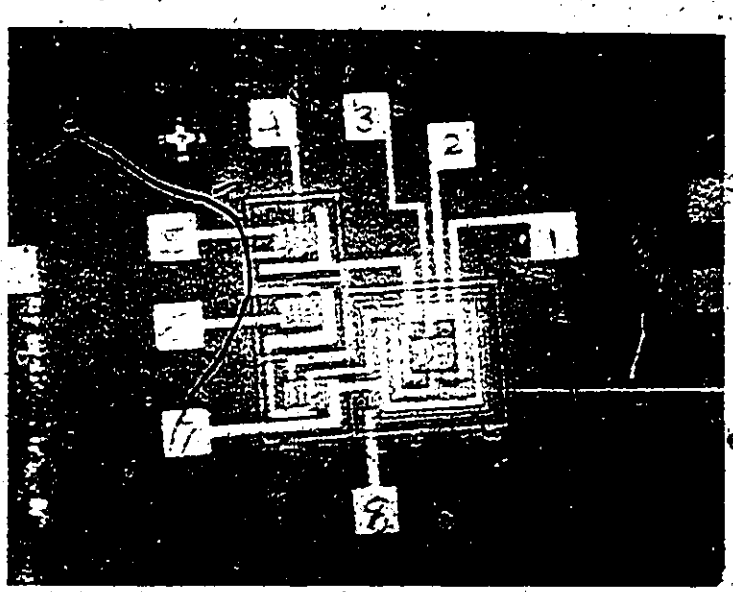
The minority carrier storage problem above can be overcome if the pnp transistor is not allowed to enter its saturation region and this can be achieved by using low capacitance Schottky diodes to define both region ab and cd. See Fig. 11 (a), (b). Here, the lateral transistor defined section bc only and functions as a constant current source of $I_0/2$ at all times. In some applications a diffused resistor can be used instead of the lateral p-n-p transistor circuit with some degradation of the static characteristic. This two-diode circuit proves to be the most practical of those discussed.

(c) MOS Circuits

With the development of compatible MOS-bipolar technology [14] it will become possible to realize load structures for bipolar logic circuits with MOS elements. A simple arrangement would be to use a MOS current source with Schottky diodes. An alternative circuit is shown in Fig. 12 (a), (b). Both depletion and enhancement elements are used. The depletion transistor Q_1 defines section abc, while the enhancement transistor defines section cd.

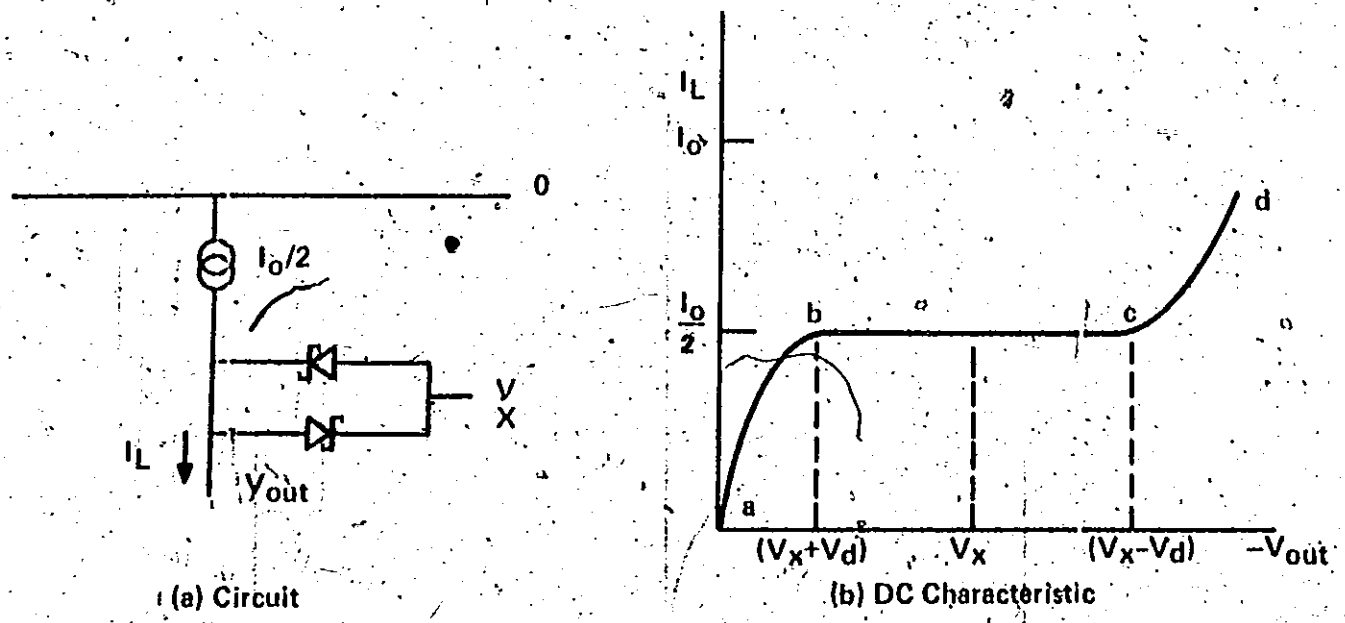


(a) Circuit



(b) Photomicrograph

FIG. 10 INTEGRATED REALIZATION OF CURRENT BIASED MULTIPLE–COLLECTOR LATERAL p–n–p STRUCTURE



(a) Circuit (b) DC Characteristic

FIG. 11 CURRENT SOURCE - TWO SCHOTTCKY DIODES

IV THE LOAD IN METLL STRUCTURES

In METLL structures, load (c) can be realized using a current-source and two-Schottky-diodes. However, on an LSI chip where external noise coupling is minimum, the circuit noise margins could be sacrificed for the ease of integration. In this case the load structure can be realized as diffused resistor acting as a current source and a junction diode resulting in a logic swing of 700 mV and noise margins of 300 mV.

V CONCLUSIONS

The analysis indicates that the high resistance diode limited non-linear load structure discussed above has advantages over linear resistive and non-linear negative resistive load structures. In addition to optimizing the coupling between circuits in terms of dc and transient characteristics, it has been found to facilitate the temperature compensation of logic and threshold levels of the METLL circuit.

REFERENCES

REFERENCES

References For Chapter 1

[1] V.A. Dhaka et al, "Subnanosecond Emitter-Coupled Logic Gate Circuit Using Isoplanner II", IEEE Journal of SSC, vol. SC-8, October 1973, pp. 368-372.

[2] S.Y. Levy et al, "System Utilization of Large-Scale-Integration", IEEE Trans. on Electronic Computers, Vol. EC-16, No. 5, October 1967, pp. 561-566.

[3] W.B. Sander, "Interconnection/Economic Trade-offs with LSI", Parallel Processor System Technologies and Applications, Chapter 9, Editors: L.M. Hobbset et al, MacMillan, London, England, 1970.

[4] P.M. Thompson and M.I. Elmasry, "A Multi-Emitter Current-Mode Structure for High-Speed LSI Arithmetic Logic Units", International E.E. Conference, Toronto, Ontario, October 1971, pp. 148-149.

[5] M.I. Elmasry and P.M. Thompson, "Non-Linear Load Structures for Current Mode Integrated Circuits", 16th Midwest Symposium on Circuit Theory, University of Waterloo, April 1973, Session VIII pp. 7.1-7.8.

[6] P.M. Thompson, M.I. Elmasry, C. Shinn and Z.E. Skokan, "Low-Power Subnanosecond Logic Family for LSI", International E.E. Conference, Toronto, Ontario, October 1973, pp. 168-169.

[7] P.M. Thompson and M.I. Elmasry, "Logic Partition for Multi-Emitter Two-Level Structures", 1973 IEEE International Symposium on Circuit Theory, Toronto, April 1973, pp. 213-216.

[8] M.I. Elmasry and P.M. Thompson, "Realization of Arbitrary Logic Functions with Multi-Emitter Current-Mode Structures", International E.E. Conference, Toronto, Ontario, Octoner 1973, pp. 170-171.

[9] M.I. Elmasry and P.M. Thompson, "Logic Partition for METLL Structures", accepted for publication as full paper in IEEE Trans. on Circuit Theory, to be published July 1974.

[10] M.I. Elmasry and P.M. Thompson, "Multi-Emitter Two-Level Structures for Logic-in-Memory Computers", 1973 National Computer Conference & Exposition, June 4-8, 1973, New York, also accepted for publication in IEEE Trans on Computers.

[11] P.M. Thompson and M.I. Elmasry, "Logic Design using the Predetermination Concept", Technical Report 72-8, Electrical Engineering Department, University of Ottawa, Ottawa, Ontario, Canada.

[12] M.I. Elmasry, P.M. Thompson and C.L. Sheng, "Modular Design of Digital Systems, Part I : Universal Logic Modules", Technical Report 72-14, Electrical Engineering Department, University of Ottawa, Ottawa, Ontario, Canada.

[13] M.I. Elmasry, P.M. Thompson and C.L. Sheng, "Modular Design of Digital Systems, Part II: Cellular Arrays", Technical Report 73-8, Electrical Engineering Department, University of Ottawa, Ottawa, Ontario, Canada.

[14] M.I. Elmasry, P.M Thompson, and C.L. Sheng, "Modular Design of Digital Systems, Part III: Function Realization", Technical Report 73-9, Electrical Engineering Department, University of Ottawa, Ottawa, Ontario, Canada.

References for Chapter 2

[1] H. Hellerman, "Digital Computer System Principles", McGraw-Hill, New York, 1967.

[2] D.L. Slotnick, "Unconventional Systems", AFIPS Conference Proceedings, SJCC 1967, pp. 477-481.

[3] D.L. Slotnick et al, "The Solomon Computer", AFIPS Conference Proceedings, FJCC 1962, pp. 97-107.

[4] L.C. Hobbs and D.J. Theis, "Survey of Parallel Processor Approaches and Techniques", Chapter 1, Parallel Processor Systems Technologies and Applications, Editors: L.H. Hobbs et al, McMillan, London, England, 1970.

[5] D.L. Slotnick, "The Fastest Computer", Scientific American, February 1971, pp. 76-87.

[6] S.Y. Levy et al, "System Utilization of Large-Scale-Integration", IEEE Trans. on Electronic Computers, Vol. EC-16, No. 5, October 1967, pp. 562-566.

[7] R. Rice, "Impact of Arrays on Digital Systems", IEEE Journal of Solid-State-Circuits, Vol. SC-2, No. 4, December 1967, pp. 148-155.

[8] R. Rice, "Interaction Between LSI and Computer System Architecture", Chapter 8, Parallel Processor Systems Technologies and Applications, Editors: L.H. Hobbs et al, McMillan, London, England, 1970.

[9] W.R. Graham, "The Parallel and the Pipeline Computers", Datamation, April 1970, pp. 68-71.

[10] T.C. Chen, "Parallelism, Pipelining and Computer Efficiency", Computer Design, January 1971, pp. 69-74.

[11] L.W. Cotten, "Circuit Implementation of High-Speed Pipeline Systems", Proceeding of the Fall Joint Computer Conference, 1965, pp. 489-504.

[12] L.W. Cotten, "Maximum-rate Pipeline Systems", Spring Joint Computer Conference, 1969, pp. 581-586.

[13] B.A. Crane and J.A. Githens, "Bulk Processing in Distributed Logic Memory", IEEE Trans. Electron Computers, Vol. EC-14, April 1965, pp. 186-196.

[14] J.S. Liptay, "Structural Aspects of the System/360 Model 85; Part II - The Cache", IBM Sys. J., Vol. 7, 1968, pp. 15-21.

[15] W.H. Kautz, "Cellular Logic-in-Memory Arrays", IEEE Trans. on Computers, Vol. L-18, August 1969, pp. 719-727.

[16] H.S. Stone, "A Logic-in-Memory Computer", IEEE Trans. on Computers, January 1970, pp. 73-78.

[17] W.B. Sander, "Interconnection/Economic Trade offs with LSI", Parallel Processor Systems Technologies and Applications, Chapter 9, Editors: L.H. Hobbset et al, MacMillan, London, England, 1970.

[18] E.C. Josheph, "Trends Toward Tomorrow's Computers", Automation, June 1969, pp. 70-76.

[19] L.D. Amdahl, "Architectural Questions of the Seventies", Datamation, January 1970, pp 66-68.

References For Chapter 3

[1] A.W. Lo, "A Comprehensive View of Digital Integrated Electronic Circuits", IEEE Proc. December 1964, pp. 1546-1564.

[2] J.J. Narud and C.S. Meyer, "Characterization of Integrated Logic Circuits", IEEE Proc., December 1964, pp. 1551-1561.

[3] D.K. Lynn et al, "Analysis and Design of Integrated Circuits", McGraw-Hill, 1967.

[4] The Staff of Science and Technology Aerospace Division-Westinghouse Electric Corporation, "Integrated Electronic Systems", Prentice-Hall, 1970.

[5] F.P. Preparata, "On the Design of Universal Boolean Functions", IEEE Trans. Computers, Vol. C-20, No. 4, April 1971, pp. 418-423.

[6] S.S. Yau and C.K. Tang, "Universal Logic Modules and Their Applications", IEEE Trans. Computers, Vol. C-19, No. 2, February 1970, pp. 141-149.

[7] Y.N. Patt, "A Complex Logic Module for the Synthesis of Combinational Switching Circuits", 1967 Proc. Spring Joint Computer Conf. AFIPS Conf. Proc., pp. 699-705.

[8] R.C. Minnick, "A Survey of Microcellular Research", J. Ass. Comput. Mach., Vol. 14, April 1967, pp. 203-241.

[9] K.K. Maitra, "Cascaded Switching Networks of Two-Input Flexible Cells", IRE Trans. Electronic Comput., April 1962, pp. 136-143.

[10] W.H. Kautz et al, "Cellular Interconnection Arrays", IEEE Trans. Comp., Vol. C-17, May 1968, pp. 443-451.

[11] D.C. Forslund and R. Waxman, "The Universal Logic Block and Its Application to Logic Design", 1966 Seventh Annual Symp. Switching and Automata Theory, Conf. Rec., pp. 236-250.

[12] B. Elspas et al, "Properties of Cellular Arrays for Logic and Storage", Stanford Research Institute, Scientific Rept. 3, June 1967, pp. 59-83.

[13] A.R. Meo, "Modular Tree Structures", IEEE Trans. Comp., Vol. C-17, May 1968, pp. 432-442.

[14] R. Waxman et al, "Automated Logic Design Techniques Applicable to IC Technology", 1966 Fall Joint Computer Conf., AFIPS Proc., Vol. 29, Washington, 1966, pp. 247-265.

[15] M.E. Conway and L.M. Spandorfev, "A Computer System Designer's View of LSI", Fall Joint Computer Conference, 1968, pp. 835-845.

[16] R.C. Minnick, "A Survey of Microcellular Research", J. Ass. Comput. Mach., Vol. 14, April 1967, pp. 203-241.

[17] W.F. King and A. Griusti, "Can Logic Arrays be Kept Flexible", Electronic Design, May 24, 1965, pp. 57-61.

[18] L.M. Spandorfer and A.B. Tnik, "Planer Interconnect Logic", Proc. of a Symp. on Microelectronics and Large Systems, Spartan Books, Washington, D.C. , 1965.

[19] S. Amarel et al, "Majority Gate Networks", IEEE Trans. EC-13, Feb. 1964, pp. 4-13.

[20] R.H. Canaday, "Two-Dimensional Iterative Logic", Proc. AFIPS 1965 Fall Joint Computer Conf., Vol. 27, Pt - 1, pp. 343-353.

[21] R.C. Minnick and R.A. Short, "Cutpoint Cellular Logic", IEEE Trans. EC-13, Dec. 1964, pp. 685-698.

[22] K.K. Maitra, "Cascaded Switching Networks of Two-Input Flexible Cells", IRE Trans. on Electronic Computers, April 1962, pp. 136-143.

[23] C.D. Weiss, "The Characterization and Properties of Cascade Realizable Switching Functions", IEEE Trans. Computers, Vol. C-18, No. 7, July 1969, pp. 624-632.

[24] C.D. Weiss, "Optimal Sythesis of Arbitrary Switching Functions with Regular Arrays of 2-Input 1-Output Switching Elements", IEEE Trans. on Computers, Vol. C-18, No. 9, September 1969, pp. 839-856.

[25] M.S. Stone and A.J. Korenjak, "Canonical Form and Synthesis of Cellular Cascades", IEEE Trans. on Electronic Computers, Vol. EC-14, No. 6, December 1965, pp. 852-862.

[26] V. Dvorak, "A Two-Rail Cascade Synthesis of Boolean Functions", IEEE Trans. Computers, Vol. C-17, No. 6, June 1968, pp. 592-596.

[27] G. Fantauzzi, "NORNAND Maitra Cascades", IEEE Trans. Computers, Vol. C-17, No. 11, November 1968, pp. 1074-1080.

References For Chapter 4

[1] Z.E. Skokan, "EFL Logic Family for LSI", ISSCC, Philadelphia, February 1973, pp. 162-163.

[2] P.M. Thompson, M.I. Elmasry, C. Shinn, S.E. Skokan, "Low-Power Sub-Nanosecond Logic Family for LSI", International Electronical Electronics Conference, Toronto, October 1973, pp. 168-169.

[3] Z.E. Skokan, "Emitter-Function Logic-Logic Family for LSI", IEEE Journal of SSC, October 1973, pp. 356-361.

[4] G. Luecke, "Noise Margins in Digital Integrated Circuits", IEEE Proc., December 1964, pp. 1565-1571.

[5] R.K. Richards, "Electronic Digital Systems", Chapter 2, John Willey and Sons, 1966.

[6] G.J. Hershowitz, "Computer-Aided Integrated Circuit Design", Chapter 3, McGraw-Hill, 1968.

[7] A.W. Lo, "A Comprehensive View of Digital Integrated Electronic Circuits", IEEE Proc., December 1964, pp. 1546-1564.

[8] L.J. Giucoletto, "On Measures of Logic Performance", IBM Journal, January 1966, pp. 51-64.

[9] J.J. Suran, "Circuit Consideration to Microelectronics", IRE Proc., February 1961, pp. 420-426.

[10] J.A. Narud and C.S. Meyer, "Characterization of Integrated Logic Circuits", IEEE Proc., December 1964, pp. 1551-1561.

[11] D.K. Lynn, C.S. Meyer and D.J. Hamilton, "Analysis and Design of Integrated Circuits", Chapter 6, McGraw-Hill 1967.

[12] The Staff of Science and Technology Aerospace Division Westinghouse Electric Corporation, "Integrated Electronic System", Chapter 4-13, Prentice-Hall, 1970.

[13] W.M. Penney and L. Lau, "MOS Integrated Circuits", Chapters 1, 4, 5 and 6, Van Nostrand Reinhold Company, 1972.

[14] D.K. Lynn, C.S. Meyer and D.J. Hamilton, "Analysis and Design of Integrated Circuits", Chapter 7 and 8, McGraw-Hill, 1967.

[15] W.R. Blood et al, "MECL System Design Handbook", Second Edition, Motorola Semiconductor Products, December 1972.

[16] M.I. Elmasry and P.M. Thompson, "Non-Linear Load Structures for Current Mode Integrated Circuits", 16th Midwest Symp. on Circuit Theory, University of Waterloo, April 1973.

[17] The Integrated Circuit Catalog 1972, Texas Instruments.

[18] Motorola MECL, April 1970.

[19] Digital Integrated Circuit Data Book, Spring 1972.

[20] RCA Solid State Division, CD 4000A Series Digital Integrated Circuits File No. 479.

[21] MSI-TTL, Texas Instruments, Bulletin CB-125.

[22] Schottky-clamped TTL Texas Instruments, Bulletin CB-147.

[23] P. Okulich and P.M. Thompson, "Temperature Compensation of Multi-Emitter Coupled Logic Circuits", International Electrical Electronics Conference, Toronto, October 1973, pp. 172-173.

[24] V.A. Dhaka et al, "Subnanosecond Emitter-Coupled Logic Gate Circuit Using Isoplaner II", IEEE Journal of SSC, October 1973, Vol. SC-8, pp. 368-372.

References For Chapter 5

[1] R.L. Ashenhurst, "The Decomposition of Switching Functions", Bell Laboratory Rept. BL-1 (II), 1952.

[2] R.L. Ashenhurst, "The Decomposition of Switching Functions", Bell Laboratory Rept. 16, III 1-72, 1956.

[3] H.A. Curtis, "A New Approach to the Design of Switching Circuits", Van Nostrand, 1962.

[4] T.L Booth, "Sequential Machines and Automata Theory", Chapter 3, John Wiley and Sons, 1967.

[5] C.W. Weller, "A High-Speed Carry Circuit for Binary Adders", IEEE Trans. Computers, Vol. C-18, No. 8, August 1969.

[6] G.A. Maley, "Manual of Logic Circuits", CH. 5, Prentice Hall, 1970.

[7] J.C. Hoffman and P. Csillag, "Multiplieur Parallele a Circuits Logiques Iteratifs". Electronics Letters, 1968, 4, pp. 178-180.

References For Chapter 6

[1] R.C. Minnick and R.A. Short, "Cellular Linear-Input Logic", Stanford Research Institute, SRI Project 4122, February 1964.

[2] R.C. Minnick et al, "Cellular Arrays for Logic and Storage", Stanford Research Institute, SRI Project 5087, April 1966.

[3] R.C. Minnick, "Cutpoint Cellular Logic", IEEE Trans. EC, Vol. EC-13, December 1964, pp. 685-698.

[4] ----, "Survey of Microcellular Research", J. ACM, Vol. 14, No. 2, April 1967, pp. 203-241.

[5] W.H. Kautz et al, "Cellular Interconnection Arrays", IEEE Trans. Computers, Vol. C-17, May 1968, pp. 443-445.

[6] A. Waksman, "A Permutation Network", J. ACM, Vol. 15, January 1968, pp. 159-163.

[7] W.H. Kautz, "A Cellular Threshold Array", IEEE Trans. EC, Vol. EC-16, October 1967, pp. 680-682.

[8] H.S. Stone, "A Logic-in-Memory Computer", IEEE Trans. on Computers, January 1970, pp. 73-78.

[9] S.E. Wahlstrom, "Programmable Arrays and Networks", Electronics, December 11, 1967, pp. 91-95.

[10] W.H. Kautz, "Fault Testing and Diagnosis in Combinational Digital Circuits", Proc. 1st Annual IEEE Computer Conf., Chicago, Ill., September 1967.

[11] ------, "Testing for Faults in Combinational Cellular Logic Arrays", Proc. 8th Annual Symp. on Switching and Automata Theory, Austin, Texas, October 1967, pp. 161-174.

[12] J. Goldberg et al, "Technique for the Realization of Ultra-Reliable Space-borne Computers", Iterim Scientific Report III, SRI Project 5580, June 1968.

[13] Ref. 2 , p. 82.

[14] W.H. Kautz, "Cellular Logic-in-Memory Arrays", IEEE Trans. on Computers, Vol. C-18, No. 3, August 1969, pp. 719-727.

[15] R.R. Seeber, "Associative Self-Sorting Memory", Proc. 1960 Eastern Joint Computer Conf., Vol. 18, pp. 179-188.

[16] B. Elspas and R.A. Short, "A Bound on the Run Measure of Switching Functions", IEEE Trans. EC, Vol. EC-13, February 1964, pp. 1-4.

[17] D.L. Shell, "A High-Speed Sorting Procedure", Commun. ACM, Vol. 2, No. 7, 1959, pp. 30-32.

[18] R.C. Bose and R.J. Nelson, "A Sorting Problem", J. ACM, Vol. 9, No. 2, 1962, pp. 282-296.

References for Appendix

[1] D.H. Chung and J.A. Palmieri, "Design of ACP Resistor-Coupled Switching Circuits", IBM Journal, July 1968, pp. 190-198.

[2] A.A. Vacca, "The Case for ECL", Electronics, April 26, 1971, pp. 48-52.

[3] D.W. Murphy and J.R. Turnbull, "Design of ACP Tunnel-Diode-Coupled Circuits", IBM Journal, November 1964, pp. 506-514.

[4] J.F. Kruy and F.T. Duran, "Integrated Conditioned OR and Inhibited OR Logic Circuits", IEEE Journal of SSC, Vol. SC-1, No. 2, December 1966, pp. 81-85.

[5] D.E. Skokan, "EFL - Logic Family for LSI", ISSCC, Feb. 1973, pp. 162-163.

[6] P.M. Thompson and M.I. Elmasry, "A MOSFET Structure for High-Speed Arithmetic Logic Units", International EECE, Toronto, October 1971.

[7] P.M. Thompson and M.I. Elmasry, "Logic Partition for Multi-Emitter Two-Level Structures", 1973 IEEE International Symposium on Circuit Theory, Toronto, April 1973.

[8] D.K. Lynn et al, "Analysis and Design of Integrated Circuits", McGraw-Hill Book Co., 1967.

[9] L.J. Giacoletto, "On Measures of Logic Performance", IBM Journal, January 1966, pp. 81-84.

[10] G.J. Hershowitz, "Computer-Aided Integrated Circuit Design", Chapter 3, McGraw-Hill, 1968.

[11] A. Barna, "Rise Time Optimization of High Speed Digital Fan-Out Circuit", IEEE Journal of SSC, June 1969, pp. 159-161.

[12] D.M. Caughey et al, "Circuit Simulation by Computer", Telesis, Vol. 2, No. 6, Fall 1972, pp. 17-24.

[13] H.C. Lin and T.B. Tan, "Lateral Complementary Transistor Structure", Proc. IEEE, Dec. 1964, pp. 1491-1495.

[14] J. Lindmayer and W. Schneider, "Theory of Lateral Transistors", Solid-State Electronics, Vol. 10, 1967, pp. 225-234.

[15] M. Polinsky and S. Graf, "MOS-Bipolar Monolithic Integrated Circuit Technology", IEEE Trans. on Electron Devices, ED-20, No. 3, March 1973, pp. 239-244.

Vita

Name: Mohamed I.Y. Elmasry

Born: Cairo, Egypt December 24, 1943

Education:

Primary: Shoubra High School
Cairo, Egypt

Secondary: Tawifikia High School
Cairo, Egypt

University: Cairo University
Cairo, Egypt
B.Sc. E.E. (honors) (1965)

University of Ottawa
Ottawa, Ontario, Canada
M.A.Sc. (1970)

Experience:

Teaching: Instructor - Cairo University (1965-1968)
Demonstrator - Ottawa University (1968-1970)

Industrial: Bell-Northern Research
Ottawa
Summers of 1970, 1971, 1972
Member of Scientific Staff since November 1972