

The Xerox '79 VLSI Systems Design Course

Informal Course Notes and Copies of the Instructors' View-Graphs
Days 1-3 of a one week short course

Doug Fairbairn

Dick Lyon

Members of the Research Staff, LSI Systems Area, Xerox PARC

In February, 1979 a one-week short course entitled "Introduction to VLSI Systems Design" was offered to local Xerox employees by the LSI Systems Area at the Palo Alto Research Center. This document provides viewers of video-tapes of that course with informal course notes and copies of the instructors' view-graphs for course days 1 thru 3. It can be used as a convenient place for taking notes while watching the tapes.

[Filed on: CourseNotes.title, August 14, 1979 10:41 PM]

Copyright © 1979, Fairbairn and Lyon. *All Rights Reserved.*

Purpose of Class:

To introduce attendees to the principals of designing large scale integrated systems and to provide an understanding of the issues which will determine the direction which integrated technology takes in the future. Homework will be assigned, the completion of which is important to real understanding of the material presented. After the class concludes, all attendees will be encouraged to complete an LSI design project, which will be fabricated and returned for testing.

Prerequisites:

Participants should have a basic understanding of digital system building blocks and functions and how they can be configured into systems. They should also have a basic understanding of the electrical circuit notions of voltage, resistance, and capacitance.

Course Text:

C. Mead, L. Conway, "Introduction to VLSI Systems"

[OUTLINE OF COURSE GIVEN AT PARC IN FEBRUARY, 1979]

1st Day:

Fairbairn:

Introduction

Welcome

The future for PARC in LSI design

Course overview

Goals

Strategy/ methodology

Course outline

Finite-state machine as an overall goal

Simple models for MOS transistors

Resistance/capacitance model

V/I characteristics

Enhancement and depletion mode devices

Relation between circuit, layout, and sticks representations

Nand/Nor logic as examples of above

Lyon:

Mixed notations

Registers

Shift Registers

Pass transistor steering

Logic blocks

Transistor ratios/Delays

Homework:

1. Read in M&C: Ch. 3, pp. 8-27; Ch 7, pp. 1-17; Skim Ch. 5.
2. Design priority circuit (stick representation)

2nd Day:

Basic digital systems structures:

Lyon:

Structured design methodologies (cont.)

Arrays: ROMs, PLAs, FSMs

Static/Dynamic memories

Fairbairn:

Useful building blocks:

Adders, ALUs

Shifters

Stacks (gating clocks to control function)

Smart Memories:

CAMs

Homework:

1. Read M&C: Ch. 1, pp. 19-32, all Ch. 2, Ch. 4, pp. 1-5.
2. Do stick diagram of clocked circuit of own choosing (e.g.)
 - Comparator
 - Clocked PLA
 - Counter
 - Two-dimensional stack

3rd Day:

Fairbairn:

Fabrication
Layout/Design Rules
Icarus tutorial

Lyon:

Real NMOS transistors
Timing/Delays/Fanout/Ratios/Body effect/ switched loads
Super buffers
Pad I/O

Homework:

1. Read M&C: Ch. 4, pp. 37-41; Icarus Manual
2. Do layout of function which was stick-diagrammed on 2nd day; design clock drivers for an array of this function and predict delays down diffusion or poly lines. (use Icarus or hand layout)

4th Day:

Chip Design examples:

Fairbairn:

Content addressable cache - System design considerations and the use of circuit simulation.

Lyon:

Serial Memory - System tradeoffs and power considerations, etc.

Fairbairn:

Scaling
Future technologies

Homework:

1. Read M&C: Ch. 4, pp. 31-36; Hand-outs
2. Continue with design of previous assignment; use symbols (PLA or other example)

5th Day:

Fairbairn:

Testing
Design methodologies
Design languages/tools
The automated line - ADF

Lynn Conway:

Summary of LSI activities in universities
Course summary and the MIT experience

Homework:

1. Finish Mead & Conway
2. Finish project chip by April 20

VLSI offers grand potential as well:

In next 10 years we can:

- **Scale circuit dimensions by $\sim 1/10$**
- **Therefore scale circuit areas by $\sim 1/100$**
- **Increase chip area by ~ 5 to 10**
- **Total effect:**
 - ~ 500 times as much circuitry /chip**

Problems to be confronted:

- **Complexity** - This is nothing new to the hardware or software people.
- **New hardware and software architectures**
- **Concurrency** - The ability to perform operations with a great deal of concurrency has caused a lot of people to begin looking more seriously at this problem. We need better ways of identifying and representing it.
- **Linking of computer science to physics.**
With the problems so interwoven within a single technology there may be a chance for progress in this area.

SYSTEM LEVEL DESIGN

(BLOCK DIAGRAMS)

REGISTER TRANSFER LEVEL

LOGIC DESIGN

STICK/LOGIC DESIGN

CIRCUIT DESIGN

PC LAYOUT

(LAYOUT)

LAYOUT

FAB

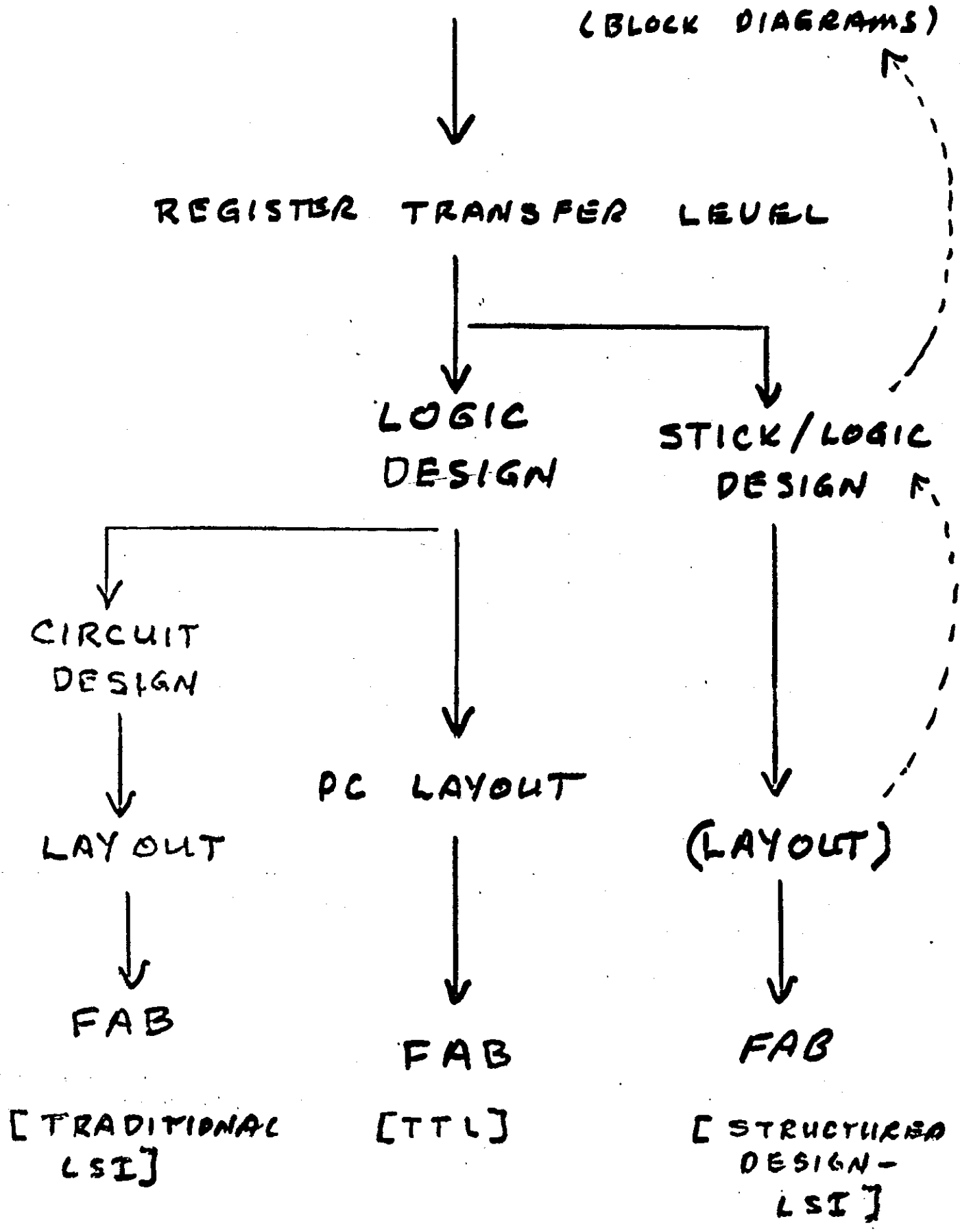
FAB

FAB

[TRADITIONAL LSI]

[TTL]

[STRUCTURED DESIGN - LSI]



Course Outline:

Pre-Course reading: Mead & Conway, Chapter 1, pp 1-18; Chapter 3, pp. 1-7.

1st Day:

Fairbairn:

Introduction

Welcome

The future for PARC in LSI design

Course overview

Goals

Strategy/ methodology

Course outline

Finite-state machine as an overall goal

Simple models for MOS transistors

Resistance/capacitance model

V/I characteristics

Enhancement and depletion mode devices

Relation between circuit, layout, and sticks representations

Nand/Nor logic as examples of above

Lyon:

Mixed notations

Registers

Shift Registers

Pass transistor steering

Logic blocks

Transistor ratios/Delays

Homework:

1. Read in M&C: Ch. 3, pp. 8-27; Ch 7, pp. 1-17; Skim Ch. 5.
2. Design priority circuit (stick representation)

2nd Day:

Basic digital systems structures:

Lyon:

Structured design methodologies (cont.)

Arrays: ROMs, PLAs, FSMs

Static/Dynamic memories

Fairbairn:

Useful building blocks:

Adders, ALUs

Shifters

Stacks (gating clocks to control function)

Smart Memories:

CAMs

Homework:

1. Read M&C: Ch. 1, pp. 19-32, all Ch. 2, Ch. 4, pp. 1-5.
2. Do stick diagram of clocked circuit of own choosing (e.g.)
 - Comparator
 - Clocked PLA
 - Counter
 - Two-dimensional stack

3rd Day:

Fairbairn:
Fabrication
Layout/Design Rules
Icarus tutorial

Lyon:
Real NMOS transistors
Timing/Delays/Fanout/Ratios/Body effect/ switched loads
Super buffers
Pad I/O

Homework:

1. Read M&C: Ch. 4, pp. 37-41; Icarus Manual
2. Do layout of function which was stick-diagrammed on 2nd day; design clock drivers for an array of this function and predict delays down diffusion or poly lines. (use Icarus or hand layout)

4th Day:

Chip Design examples:

Fairbairn: Content addressable cache - System design considerations and the use of circuit simulation.

Lyon: Serial Memory - System tradeoffs and power considerations, etc.

Fairbairn:
Scaling
Future technologies
Homework:

1. Read M&C: Ch. 4, pp. 31-36; Hand-outs
2. Continue with design of previous assignment; use symbols (PLA or other example)

5th Day:

Fairbairn:

Testing
Design methodologies
Design languages/tools

7
The automated line - ADF

Lynn Conway:

Summary of LSI activities in universities
Course summary and the MIT experience

Homework:

1. Finish Mead & Conway
2. Finish project chip by April 20

Advanced Seminar Topics: (not complete)

System Design philosophy - Carver Mead

Trade-offs in memory system design - Lynn Conway

Self-timed logic - Chuck Seitz

Towards automated layout - Dave Johannsen, Ron Ayers

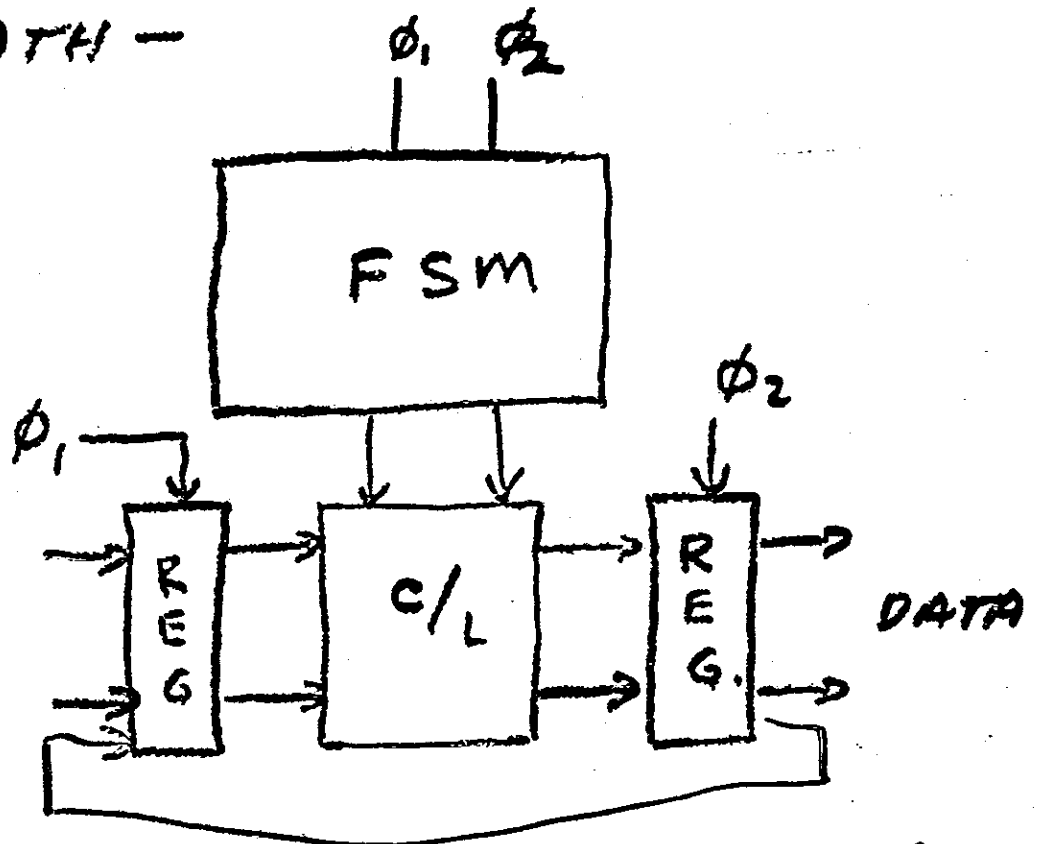
Chip design examples - by the class

WHAT TYPES OF THINGS WILL WE TALK ABOUT?

• DIGITAL SYSTEMS

⇒ SPECIFICALLY :

FINITE STATE MACHINES
CONTROLLING A DATA
PATH -



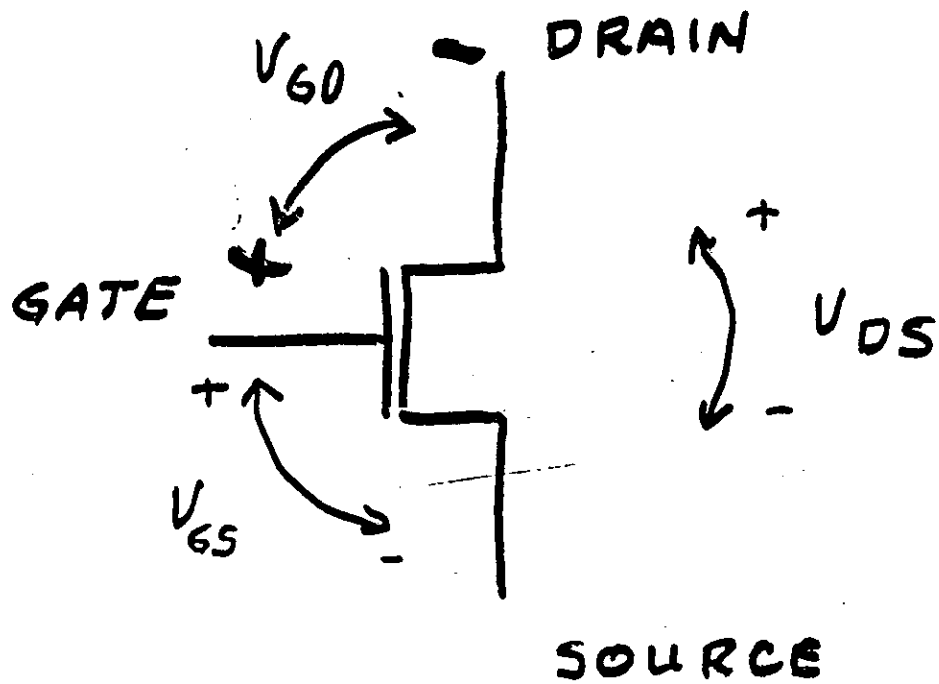
2 TYPES OF COMPONENTS
CONSISTENT SET OF CLOCK
(2 ϕ , NON-OVERLAPPING)

Slide #9

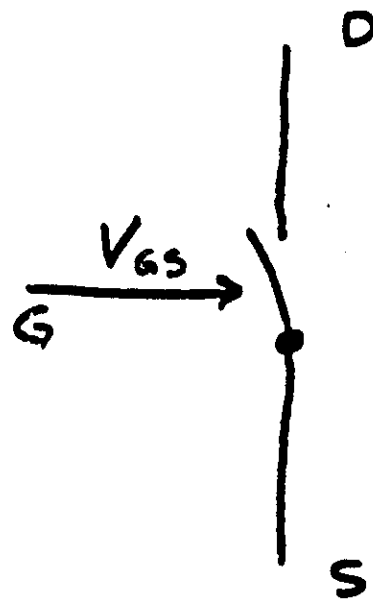
Why design in nMOS?

- Most common LSI technology available**
- High performance, high density, scalable**
- Easier to design than bipolar circuits**
- Techniques are transferrable to other MOS technologies**

THE BASIC MOSFET :



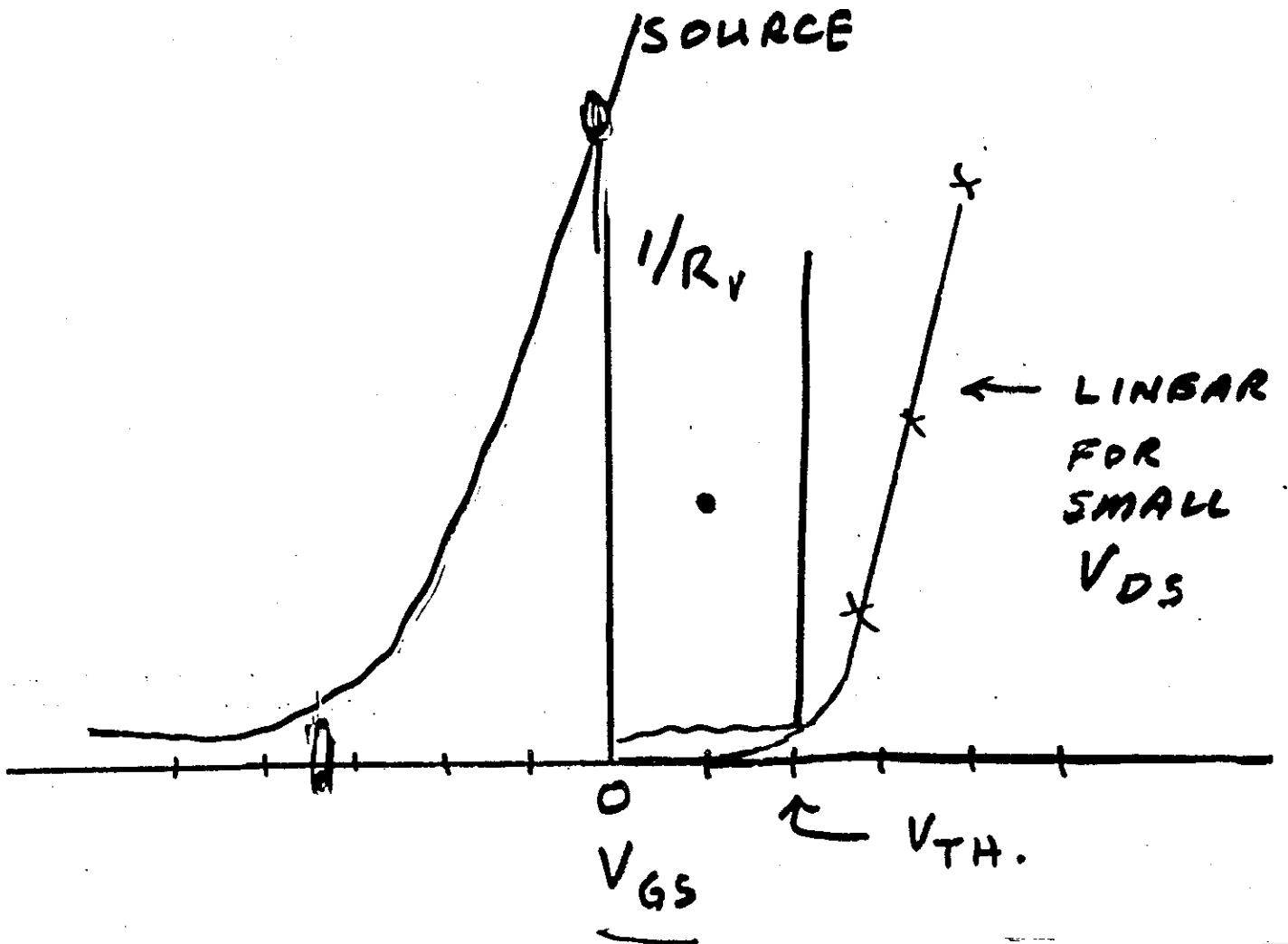
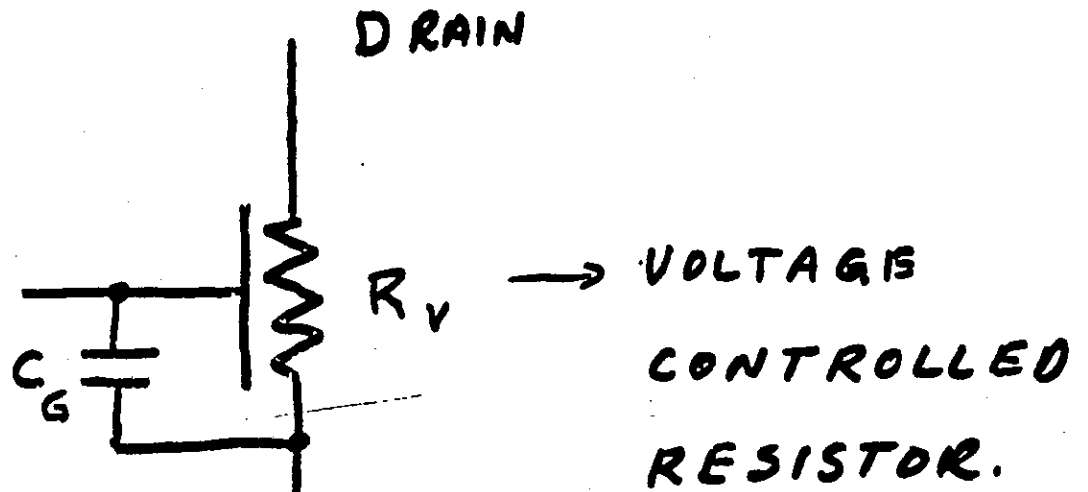
SIMPLE MODEL :



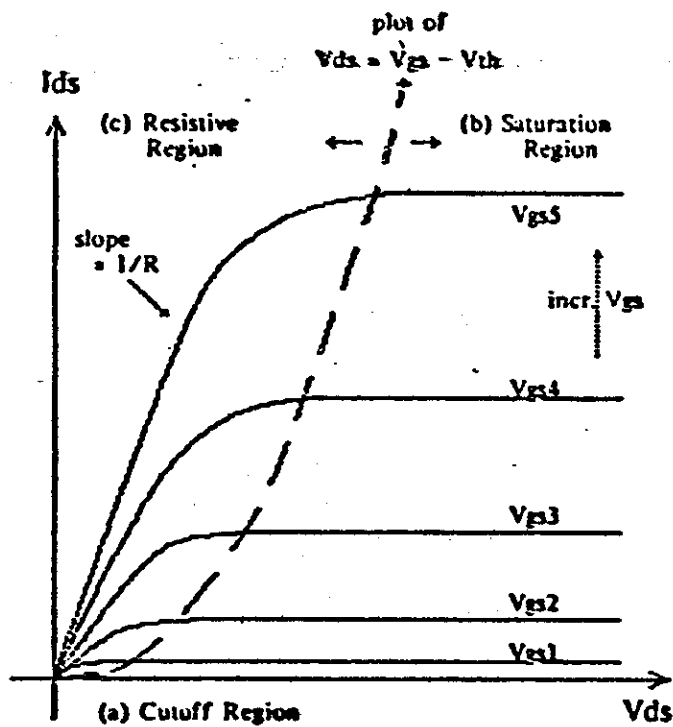
WHEN $V_{GS} > V_{TH}$
SWITCH CLOSES

MOSFET

⇒ MORE COMPLICATED MODEL



142



(a) Cutoff Region:

$V_{gs} < V_{th} . I_{ds} = 0$

(b) Saturation Region:

$V_{gs} \geq V_{th} . V_{ds}$ sufficiently high so
 $V_{gd} < V_{th} .$ (i.e.: $V_{ds} \geq V_{gs} - V_{th}$)

MOSFET acts as current source, with
 I_{ds} proportional to $(V_{gs} - V_{th})^2$

(c) Resistive Region:

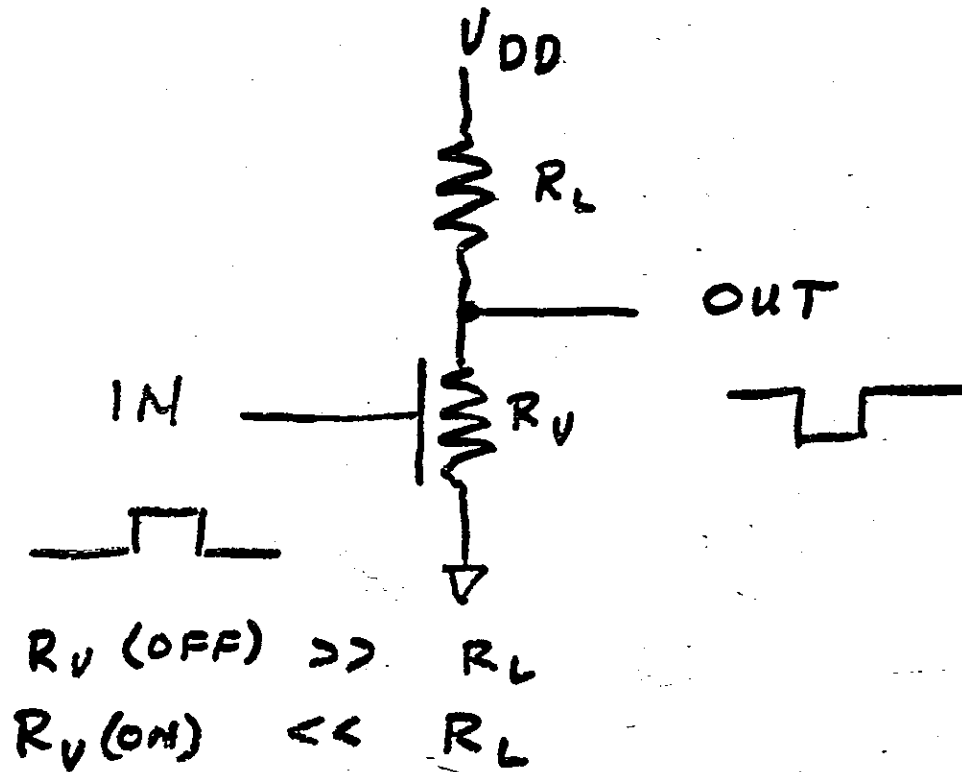
$V_{gs} \geq V_{th} . V_{ds}$ sufficiently low so
 $V_{gd} \geq V_{th} .$ (i.e.: $V_{ds} < V_{gs} - V_{th}$)

MOSFET acts as resistor, with resistance
inversely proportional to $(V_{gs} - V_{th})$

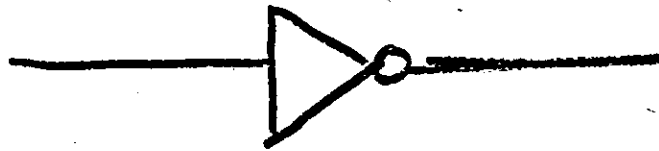
Fig. 9a. Summary of MOS Transistor Characteristics

BASIC LOGIC ELEMENT 1

⇒ INVERTER



LOGIC SYMBOL:



BUT RESISTORS ARE HARD TO MAKE IN CMOS...

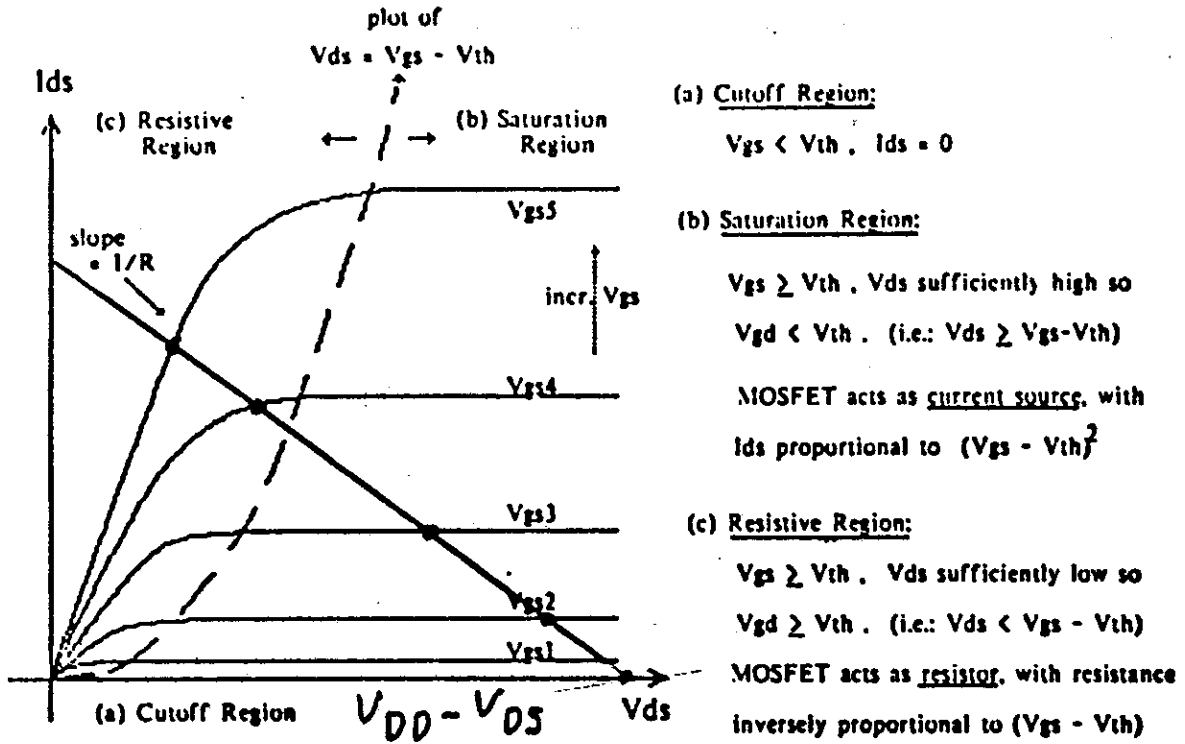


Fig. 9a. Summary of MOS Transistor Characteristics

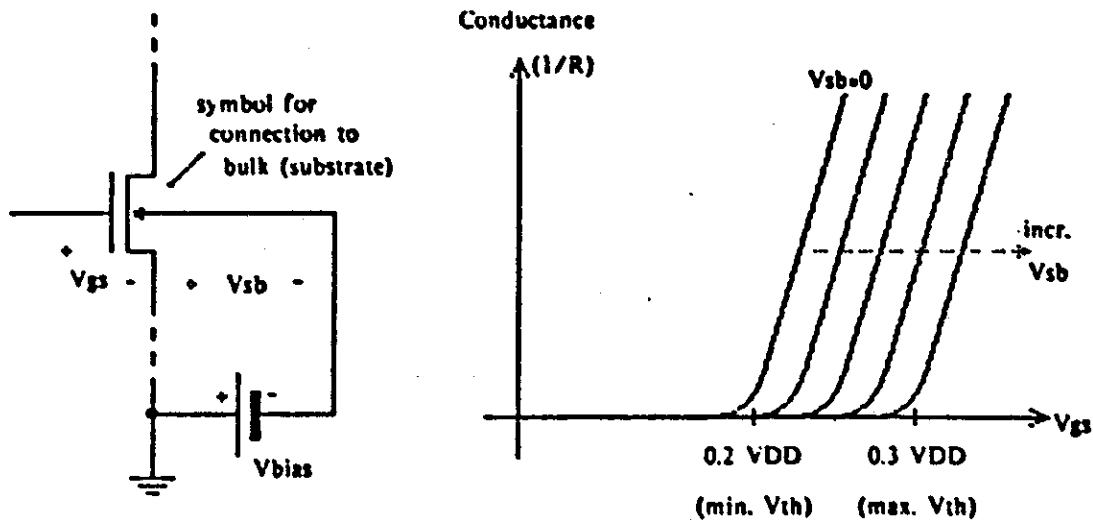


Fig. 9b. The Body Effect

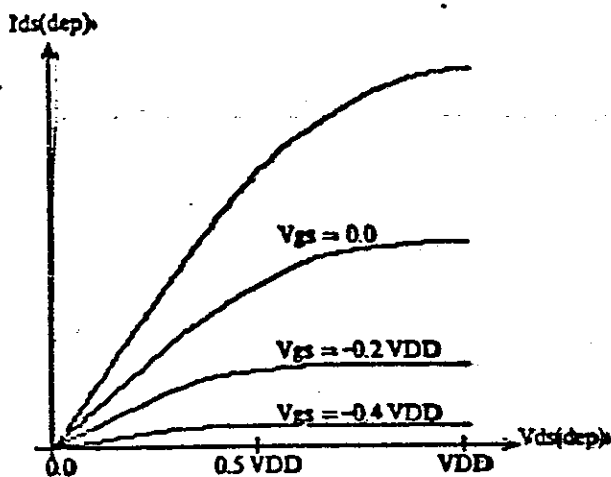


Fig. 3a. Inverter Pullup Characteristics

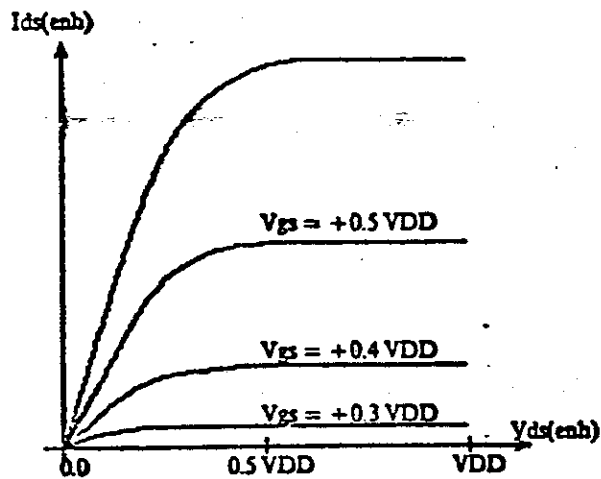


Fig. 3b. Inverter Pulldown Characteristics

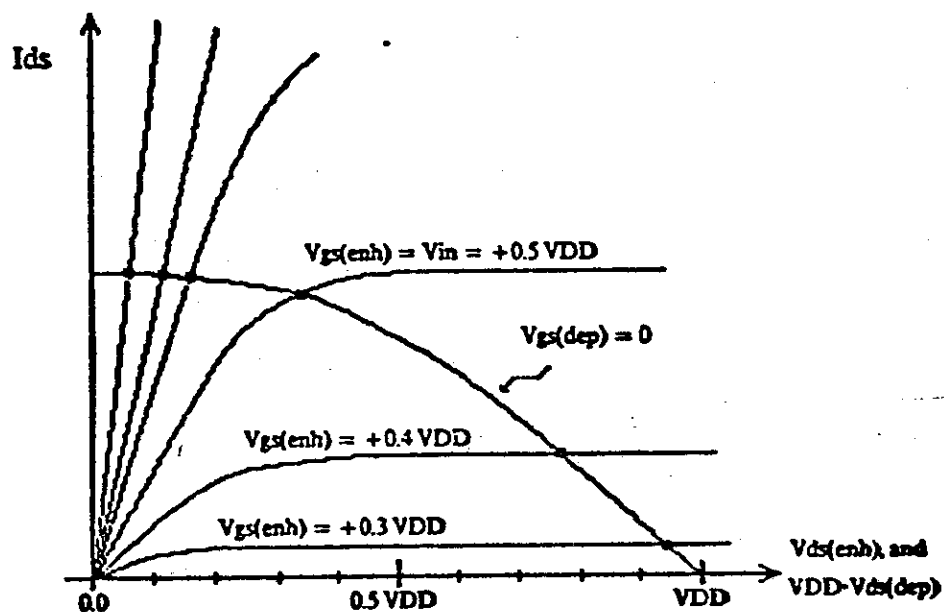


Fig. 3c. $I_{ds}(enh)$ vs $V_{ds}(enh)$, and $I_{ds}(dep)$ vs $[VDD - V_{ds}(dep)]$

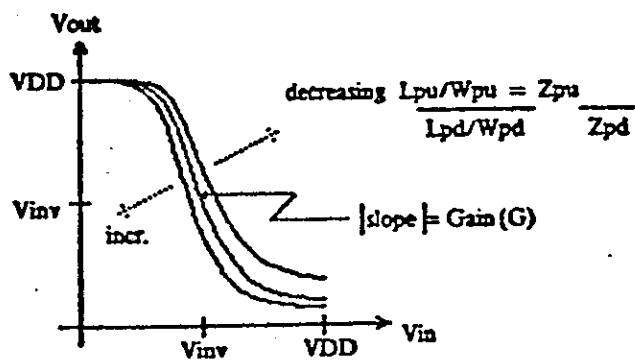
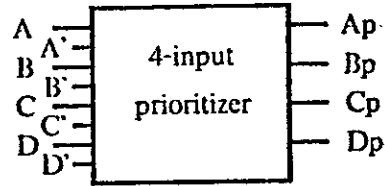


Fig. 3d. V_{out} vs V_{in} for the Basic Inverter

Homework for Monday PM -- due Tuesday AM before class.

1(a). Construct a stick diagram for an MOS structure implementing a 4-input prioritizer as described below:

A	B	C	D	Ap	Bp	Cp	Dp
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	X	0	0	1	0
0	1	X	X	0	1	0	0
1	X	X	X	1	0	0	0



X=dont'care

Hint: -you'll probably need the GND (logic 0) input, and some solutions use the VDD (logic 1) input.

1(b). Explain the principle behind your solution to 1(a), i.e., the basic idea of how your design works. Could your design be expanded in some natural way to implement prioritizers having more inputs? How?

2. Construct stick diagrams for at least 2 distinctly different ways of implementing the following 3-input combinatorial function in NMOS:

A	B	C	OUT
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Hint: some solutions make use of the fact that this is a 3-input Exclusive-OR or Parity function. There are many ways to attack the implementation of this function.

Reading Assignment: M&C Ch. 3, pp 8-27, and Ch. 7, pp 1-17; skim Ch. 5.

383
SCIENTIFIC
AMERICAN OFFPRINTS

Microelectronics and Computer Science

by Ivan E. Sutherland and Carver A. Mead

**SCIENTIFIC
AMERICAN**

SEPTEMBER 1977
VOL. 237, NO. 3 PP. 210-228



PUBLISHED BY W. H. FREEMAN AND COMPANY 660 MARKET STREET, SAN FRANCISCO, CALIFORNIA 94104

Microelectronics and Computer Science

Large-scale integration makes logic elements fast and cheap, leaving movement of data still slow and expensive. New theories and designs are required, based on parallel processing and geometric regularity

by Ivan E. Sutherland and Carver A. Mead

Computer science has grown up in an era of computer technologies in which wires were cheap and switching elements were expensive. Integrated-circuit technology reverses the cost situation, making switching elements essentially free and leaving wires as the only expensive component. In an integrated circuit the "wires," actually conducting paths, are expensive because they occupy most of the space and consume most of the time. Between integrated circuits the wires, which may be flat conducting paths on a printed circuit board, are expensive because of their size and delaying effect. Computer theory is just beginning to take the cost reversal into consideration. As a result computer design has not yet begun to take advantage of the full range of capabilities implicit in microelectronics. As we learn to understand the changed relative costs of logic and wiring and to take advantage of the possibilities inherent in large-scale integration we can expect a real revolution in computation, not only in the forms of computing machines but also in the theories on which their design and use are founded.

Why is it that computation theory needs to be revised? Suppose one sets out to develop some theories of computation, hoping to put them to work toward two ends: to establish upper bounds on what is computable and to serve as a guide to the design and use of computing machines. Such theories would presumably also advance understanding of computation processes and perhaps shed light on the nature of knowledge and thought. The theories might be based purely on mathematical reasoning or might also be based on fundamental physical principles. By mathematical reasoning alone one can prove many things about computers without resorting to physical principles. Only by attending to physical principles, however, can one make more quantitative

statements about how long a computer of given physical dimensions must take to accomplish a given process, based on the fact that information cannot move around in the computer faster than the speed of light and that it takes a certain amount of matter, energy and space to represent one bit, or binary digit, of information with a given reliability.

Computer science as it is practiced today is based almost entirely on mathematical reasoning. It is concerned with the logical operations that take place in computing devices. It touches only lightly on the necessity to distribute logic devices in space, a necessity that forces one to provide communication paths between them. Computer science as it is practiced today has little to say about how the physical limitations to such communications bound the complexity of the computing tasks a physically realizable computer can accomplish.

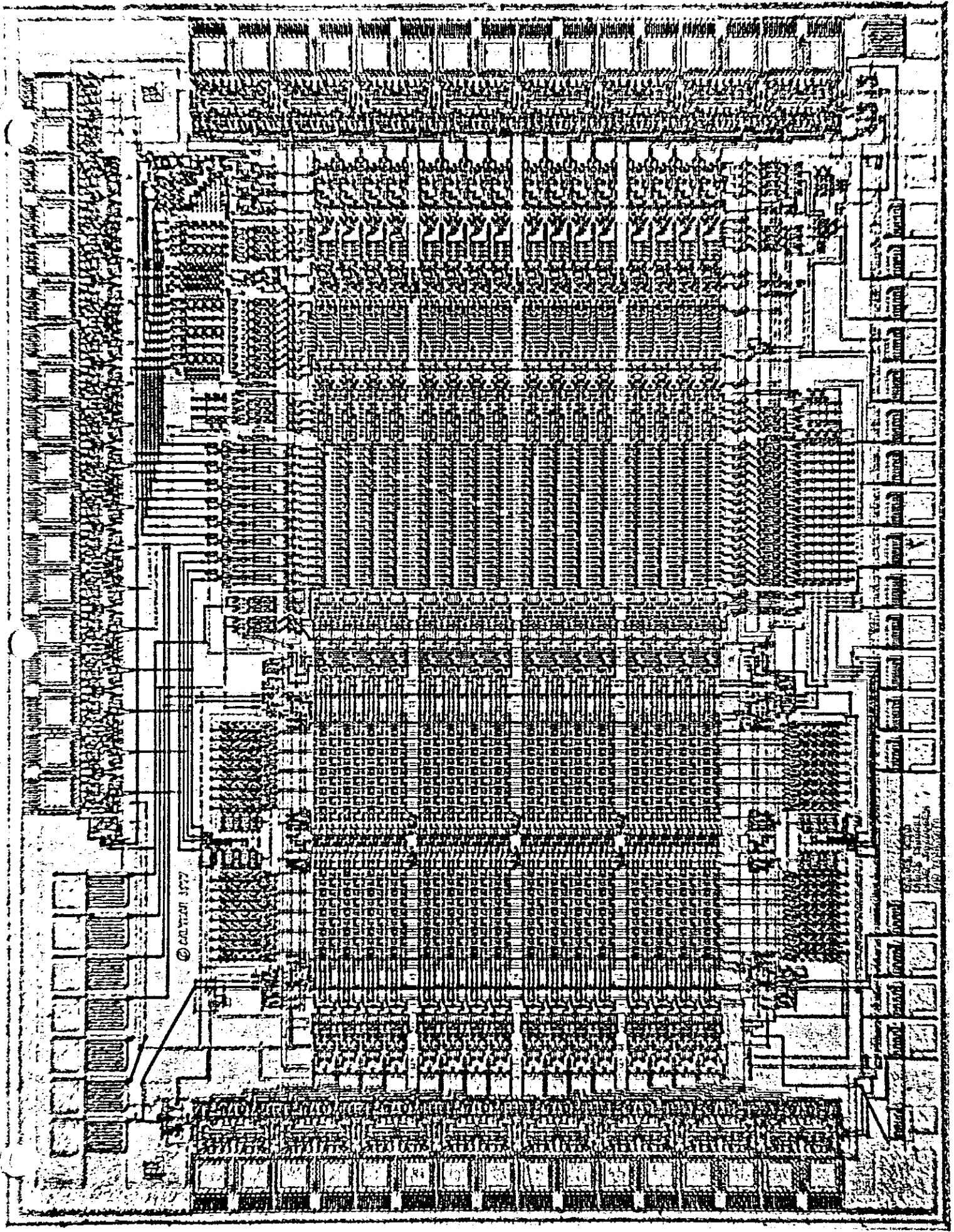
That is so in part because anyone who thinks of a computer as a logical machine that performs logical, numerical or algebraic operations on data will naturally think of the machine in terms of the mathematical notation relevant to those fields. In such notations the symbol x written in one place on the page is identical in meaning with the symbol x written in another place on the page. The idea that communication in space is required if such values are to be identical as represented in a computer storage device has no place in the notation. The notation itself focuses attention on the logical operations, reflecting the fact that human beings think most effectively about only one thing at a time. A mathematical proof is a sequence of steps we absorb over a period of time, and it is easiest to think of computing devices that also do only one thing at a time. The sequential approach to mathematics is not required inside a comput-

er, but the mathematical approach we normally take to problems does not encourage us to think of approaches other than sequential ones for the solution of problems. Nearly all computers in operation today perform individual steps on individual items of data one after another in time sequence.

It was appropriate to ignore the costs of communication when logic elements were slow and expensive and wires were relatively fast and cheap. Sequential machines are appropriate to such technologies because they can be built with a minimum number of switching elements. We have been led—by natural inclination, by our accustomed notations for mathematics and by technology—to develop a style of computing machines and a body of computing theory both of which are rendered obsolete by integrated-circuit technology. We have been able to ignore the limitations placed by physical principles on communications inside computers because those communications did not slow down our operations appreciably and were only a small part of the cost of the machines we built. By making logic elements essentially free and leaving communication cost the dominant factor, integrated-circuit technology forces us into a revolution not only in the kinds of machines we build but also in their theoretical basis.

Developing a new theoretical basis

"OM" CIRCUIT, an experimental microprocessor designed by the authors at the California Institute of Technology, is notable for its high degree of regularity, which makes it possible to pack more logic and memory functions on the chip. The main body of the chip (omitting the communication interfaces at top and bottom) is made up of 16 nearly identical columns in four groups of four; each column represents one bit of a 16-bit computer. About 40 percent of the chip (*lower portion*) is memory; middle 20 percent is the "shifter" section and top 20 percent is the arithmetic section.

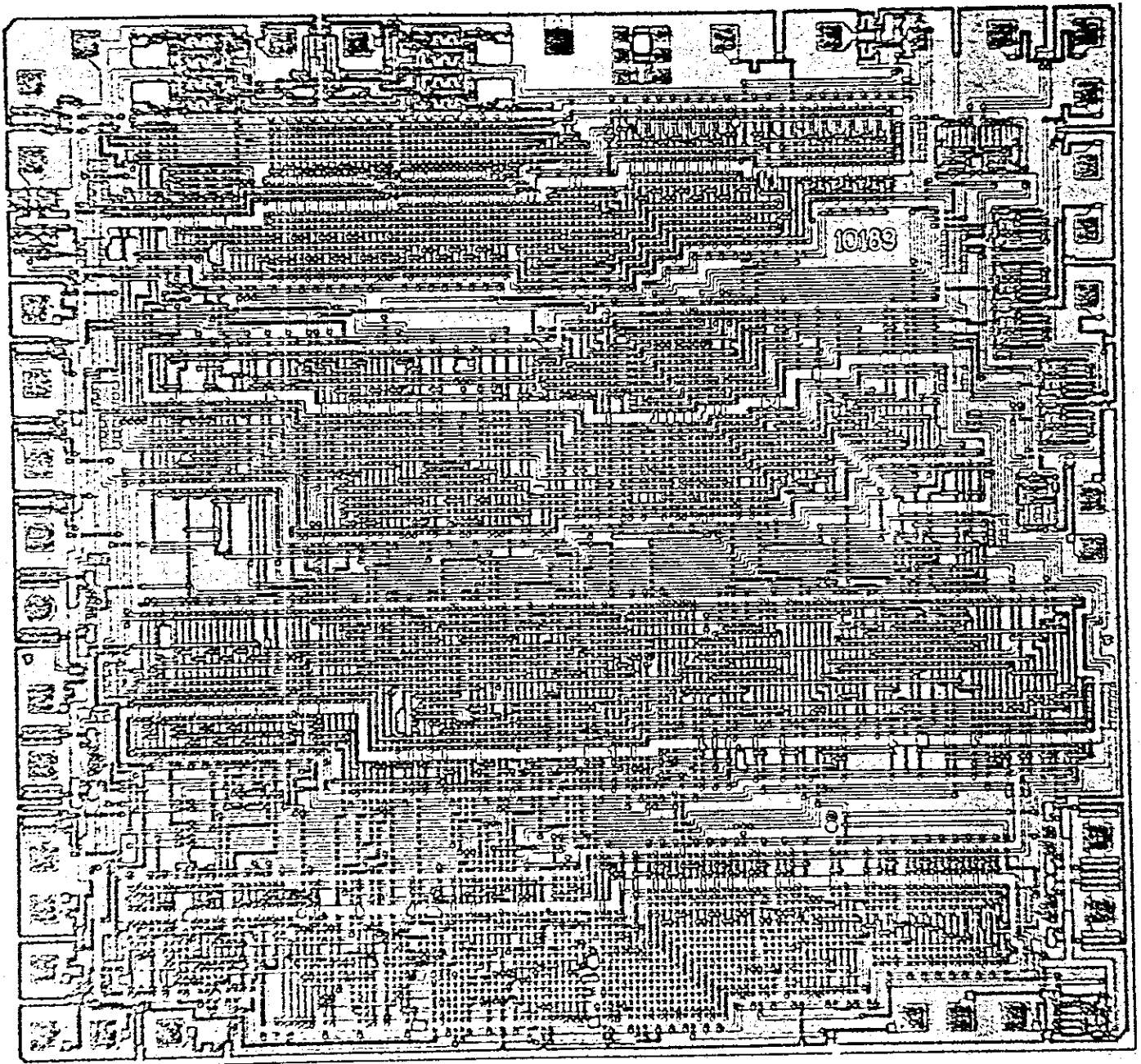


for computer science will not be easy: indeed, the task has been put off in part because it is very difficult to combine notions of logic with notions of topology, time, space and distance, as a new theory will require. In this article we shall outline some of the elements such a theory must include, first by examining the inadequacies of a simple existing theory applicable to small logic networks. Then we shall see how the changes in the relative costs of wiring and logic must change the nature of the com-

puters built in the future. Finally we hope to outline some elements we feel belong in a theory of computation appropriate to the new structures. Such a theory will be quite unlike the present basis of computer science, and so we feel justified in describing as revolutionary the effect of integrated-circuit technology both on the design of computing machines and on the intellectual framework within which such machines are exploited.

Most computer-science curriculums

include a course in switching theory, even though it is largely irrelevant to the present-day practice of computer design. Switching theory, which was developed to help design the relay-operated switching networks of automatic telephone systems, provided guides that enabled a designer to formulate a network with the minimum number of relays for accomplishing some given logical operation. It has been extended to the design of networks of newer kinds of logic elements, for example a logic network with



INTERCONNECTIONS among the logic elements of an integrated circuit have become more expensive than the elements themselves. Moreover, as the complexity of a randomly wired array of elements increases, the interconnections become longer and more numerous.

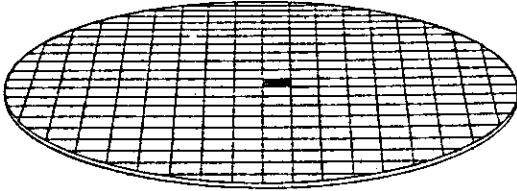
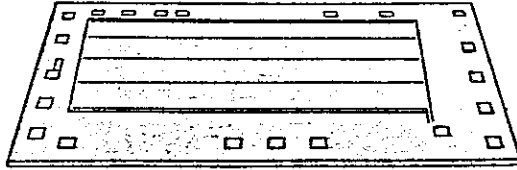
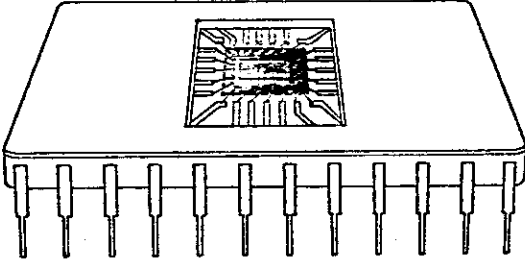
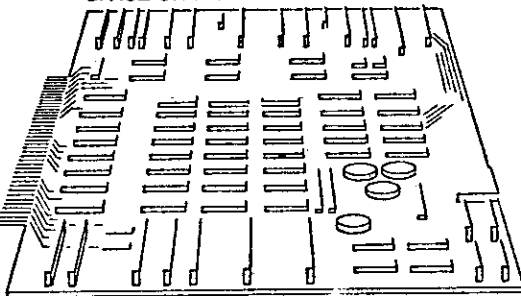
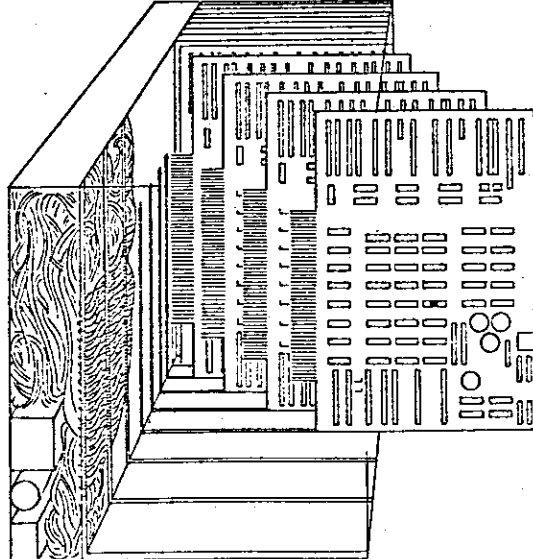
Even at modest levels of complexity "wiring" occupies most of the available space on an integrated-circuit chip. This is a comparatively simple integrated circuit dating from about 1971. Note that the linear connectors running between active elements occupy most of the space.

the minimum number of conventional logic gates.

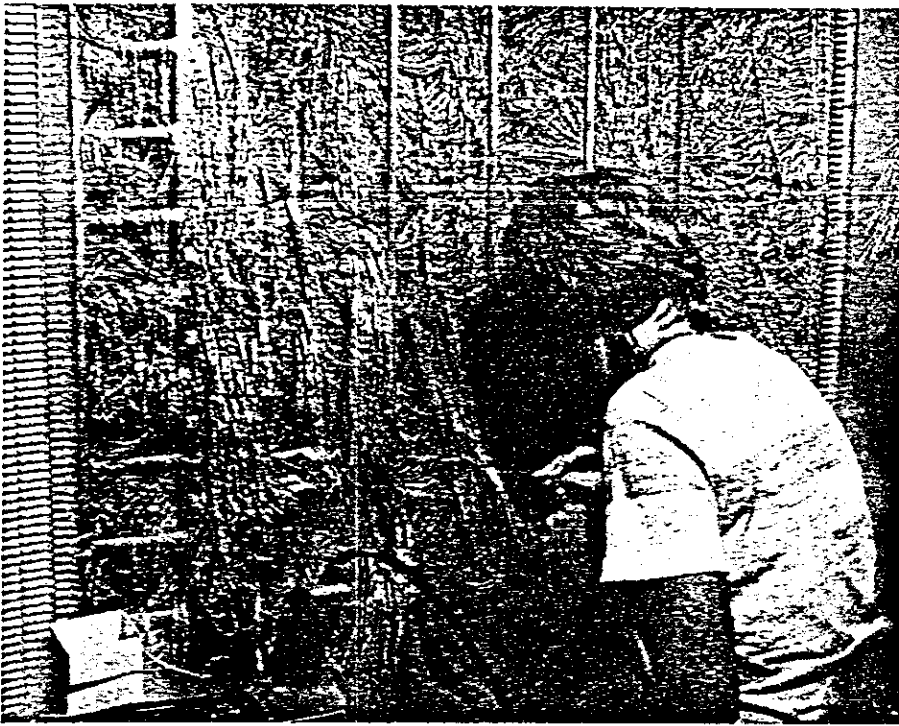
There is no guarantee, however, that such a minimum-number network will occupy the minimum space in an integrated circuit or perform its task in the minimum time. Integrated-circuit designers find they can often add transistors to a design and thereby save space or time, because adding to the minimum number may simplify the pattern of conductors in the design and may speed up its operation. Switching theory does minimize the number of switching components, but it ignores the cost and delay of the communication paths. In today's technology the area of a circuit devoted to communication between elements usually far exceeds the area devoted to switching elements, and communication delays are much longer than logic delays. What is needed, therefore, is a theory that minimizes the cost of computational tasks, considering not only the cost in area and time of the switching elements but also the much larger area and time costs of transporting data from one place to another. Because switching theory as it is known today is based on an obsolete cost function it is largely useless for the design of integrated circuits.

Switching theory is even less useful at the level of design where one is combining integrated circuits into a larger system. In most cases it costs much more to test, package and interconnect integrated circuits than to manufacture the circuits themselves. These costs are largely independent of the particular function of the circuit involved. Even if the cost of the circuits is ignored, communication from one integrated-circuit chip to another is much slower than communication on a single chip. Given a catalogue of standard circuits, there is great motivation to introduce more complex integrated circuits because fewer of them are required, so that the large cost of mounting and interconnecting them is reduced. In fact, designers often specify integrated circuits containing superfluous elements because there is no cost advantage to eliminating the unneeded switching elements. Switching theory has nothing to say about these important issues of cost and speed.

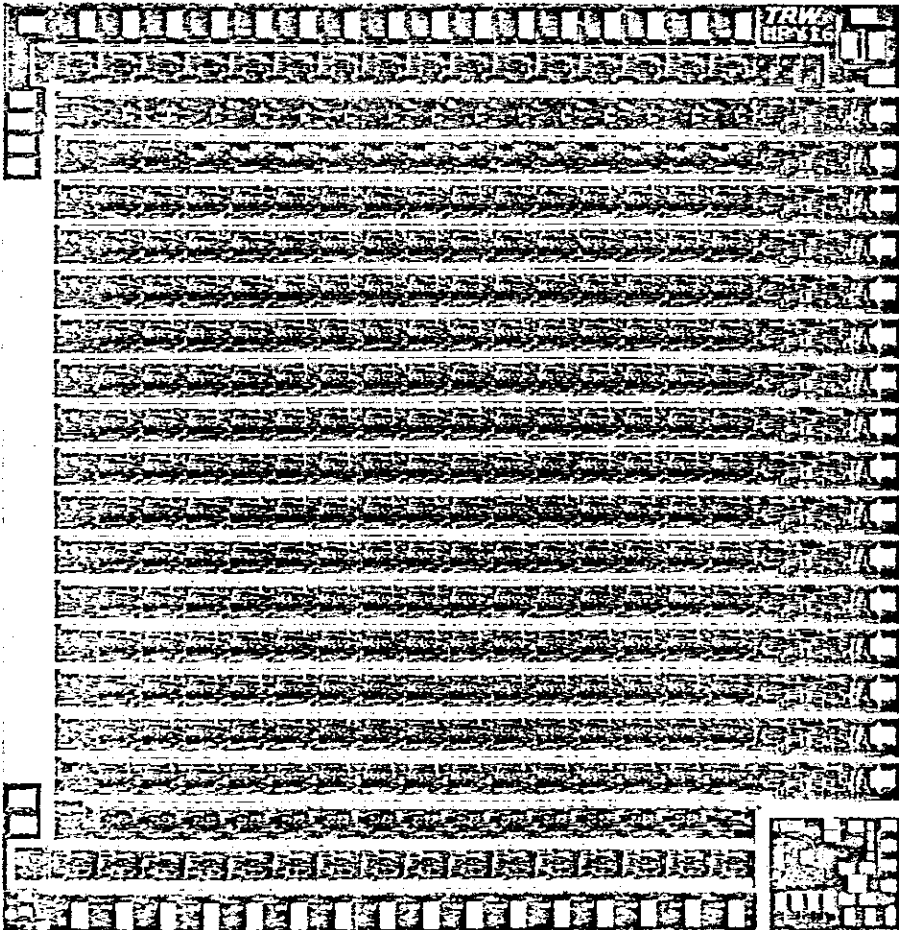
Although the cost of communication has so far found no real place in the theoretical results of computer science, it does play a role in the thinking of practical designers. Seymour Cray, the designer of many of the most powerful computers, cites the "thickness of the mat" and "getting rid of the heat" as the two major problems of machine design. It is obvious that controlling the geometry of the interconnections is essential. If connections can be made to follow regular patterns, they can be produced by less expensive methods and can also be

	AVERAGE COST	CUMULATIVE COST
 <p>CHIP IN WAFER, UNTESTED</p>	\$.10	\$.10
 <p>TESTING AND YIELD PER GOOD CHIP</p>	1.00	1.10
 <p>PACKAGE, PACKAGING AND TESTING</p>	.50	1.60
 <p>SPACE ON PRINTED-CIRCUIT BOARD</p>	1.00	2.60
 <p>SHARE OF BACK PANEL AND WIRING SHARE OF CABINET AND POWER SUPPLY</p>	.15 .20	2.75 2.95

COST OF AN INTEGRATED CIRCUIT is a small part of the cost of a complete system. As is shown here, the cost of a single typical integrated-circuit die in a wafer is only 10 cents. Given about a 20 percent yield of good chips, after packaging and testing each good chip costs \$1.60. Assuming that 100 chips are assembled on each of 20 printed-circuit boards, the cost per chip is almost doubled by each chip's share of board, back panel, cabinet and power supply.



THICK MAT OF WIRES covers the back panel of a large general-purpose computer, in this case the Cray Research, Inc., CRAY-1. Moving data over the wires takes time and costs money, and the thickness of the mat makes repairs difficult. Arranging elements of a computer so that wires are all parallel would greatly reduce the complexity and thickness of the mat.



REGULARITY is a characteristic of memory circuits and of certain arithmetic circuits, such as this 16-bit multiplier array made by TRW Inc. The chip, about .28 inch square, contains more than 18,000 transistors and resistors. Regularity makes for high logic-element density.

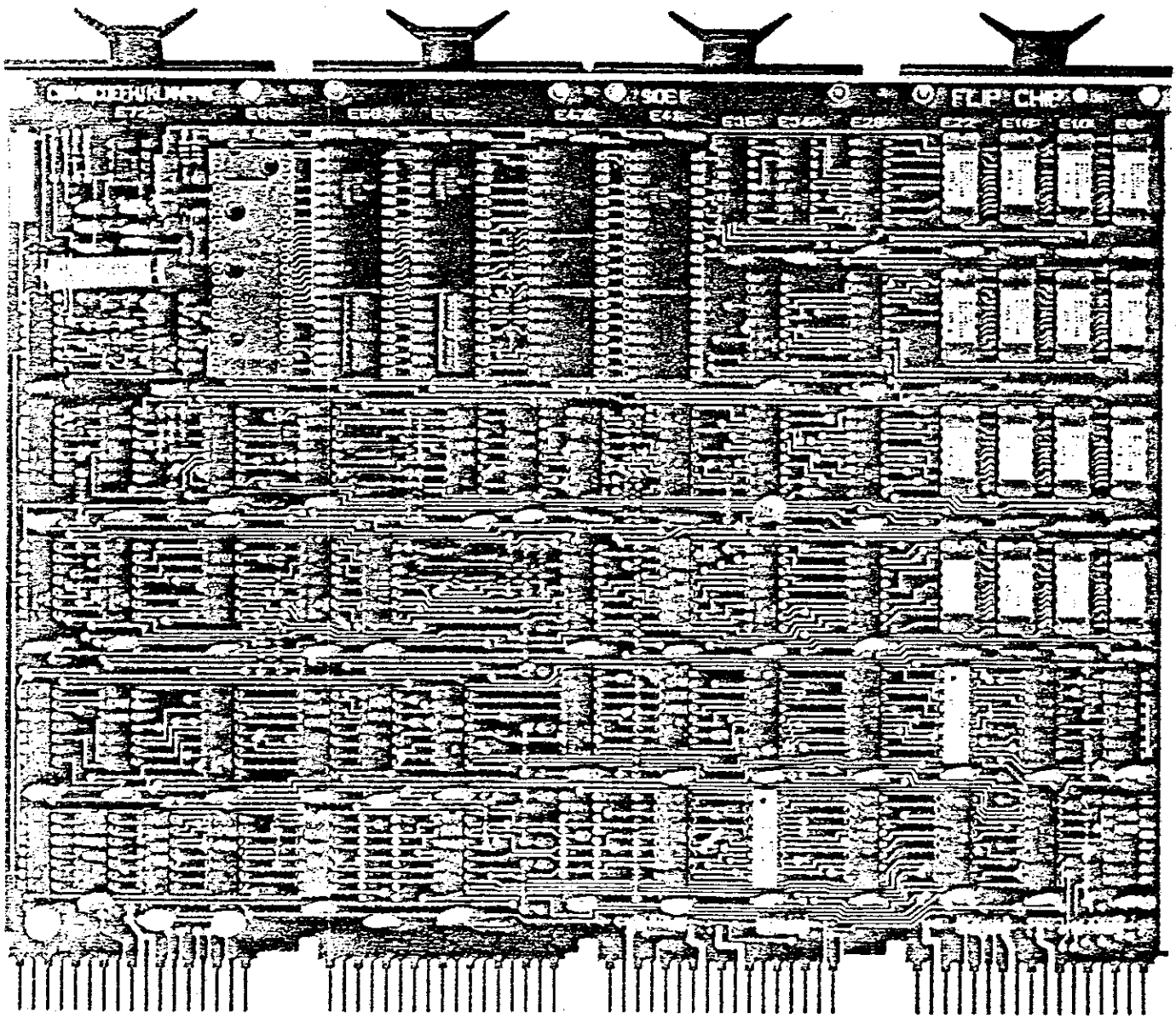
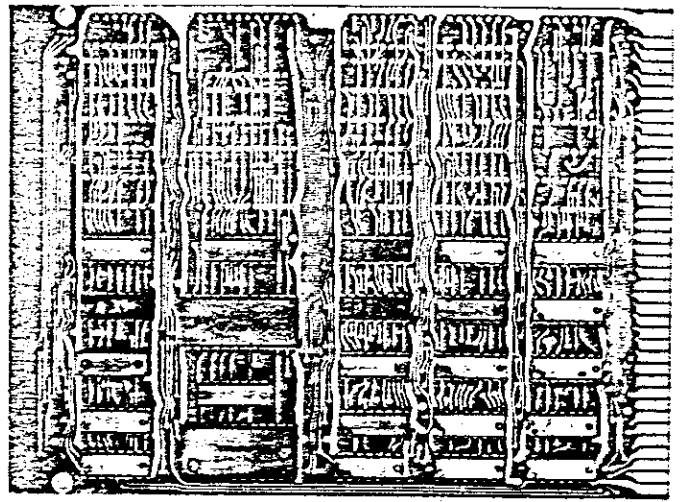
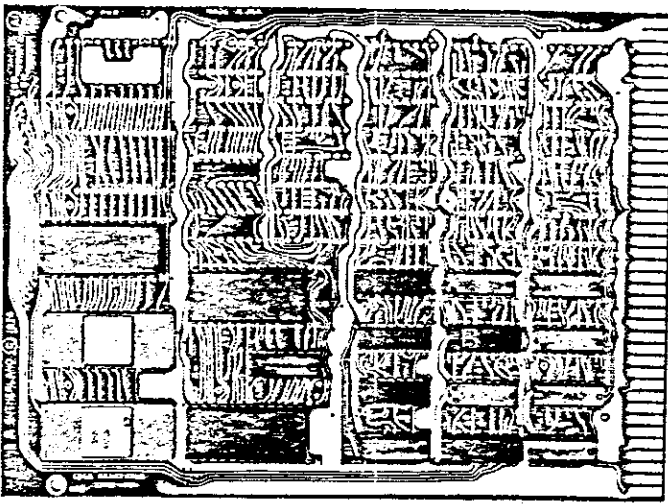
made to occupy less space and so be faster.

If the geometry of interconnection paths is not carefully controlled, the space required for them grows more than linearly as the number of logic elements to be connected is increased. This nonlinear growth comes about because bigger systems require more wires, which are on the average also longer. Because the interconnection paths grow both in number and in length the total area or volume devoted to communication becomes disproportionately larger: to interconnect twice as many randomly placed devices requires four times as much communication space. To accommodate greater wiring space larger printed circuit boards must have wider spacing between components than small boards have; Los Angeles suffers more from freeway congestion than Plains, Ga., does.

Not only do longer communication paths occupy a disproportionate amount of space but also they function more slowly than short ones. That is because signals traveling even at the speed of light take some time to travel down a path and also because longer paths store more energy. (Inside integrated circuits the speed limit set by the speed of light is not yet an important issue because the distances are short compared with the switching times of the logic elements; the energy-storage delays, however, are important.) Before a signal path can be switched from one electrical state to another, the energy stored in the path must be removed and converted into heat. One must either design a larger driving circuit to provide for the larger power required to switch long wires quickly or suffer the delays of passing the larger amounts of energy through a less powerful driver. More powerful drivers must themselves be driven, and they are therefore not only larger in area but also inherently slower than small drivers.

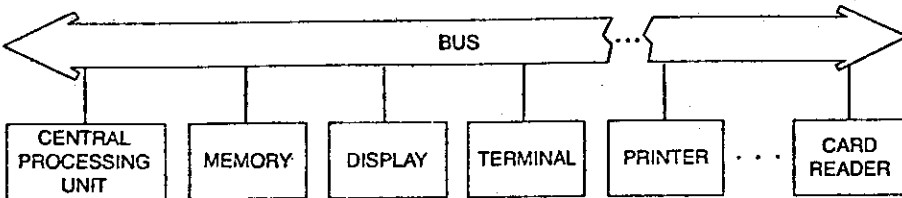
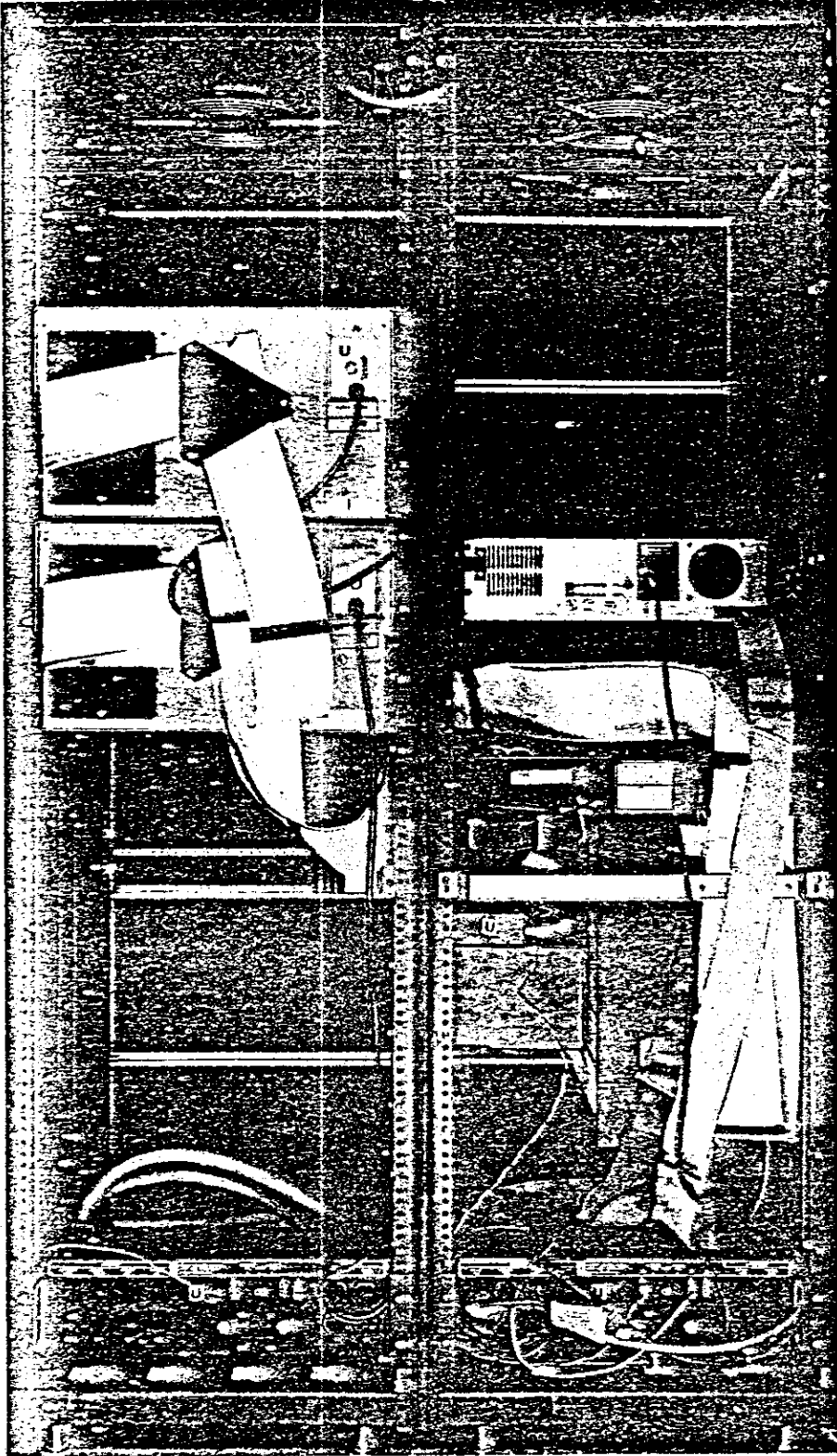
Moreover, the heat generated by the more powerful drivers must be dissipated in some structure, which itself occupies space. It is quite possible that the signaling energies required in a given technology and the size of the structures provided to dissipate heat may set an upper limit to the complexity of the systems that can be built in that technology. Above such a limit the increase in wire length required to provide the space to house what is required to drive longer wires may exceed the original increase in length of wires that was made possible by the larger drivers! There is so far no theory addressing the limits to speed and complexity that may be imposed by this possibility.

The disproportionate growth of interconnections can be avoided by building



PRINTED-CIRCUIT BOARDS can also be designed to minimize the preponderance of communication paths. Regularity decreases the amount of wiring (*top*). A five-by-seven-inch board of irregular logic made by the Evans & Sutherland Computer Corporation (*top left*) is compared with a more regular memory board of the same size (*top right*). The larger the board, the greater the preponderance of wiring.

An 8 1/2-by-10-inch board, the Digital Equipment Corporation's LSI-11 microcomputer, shows how much area is occupied on a conventional large board by communication paths (*bottom*). If the close packing characteristic of the regularly wired memory circuits that can be seen in the top right portion of board could have been attained throughout the board, the board would have been only half as big.

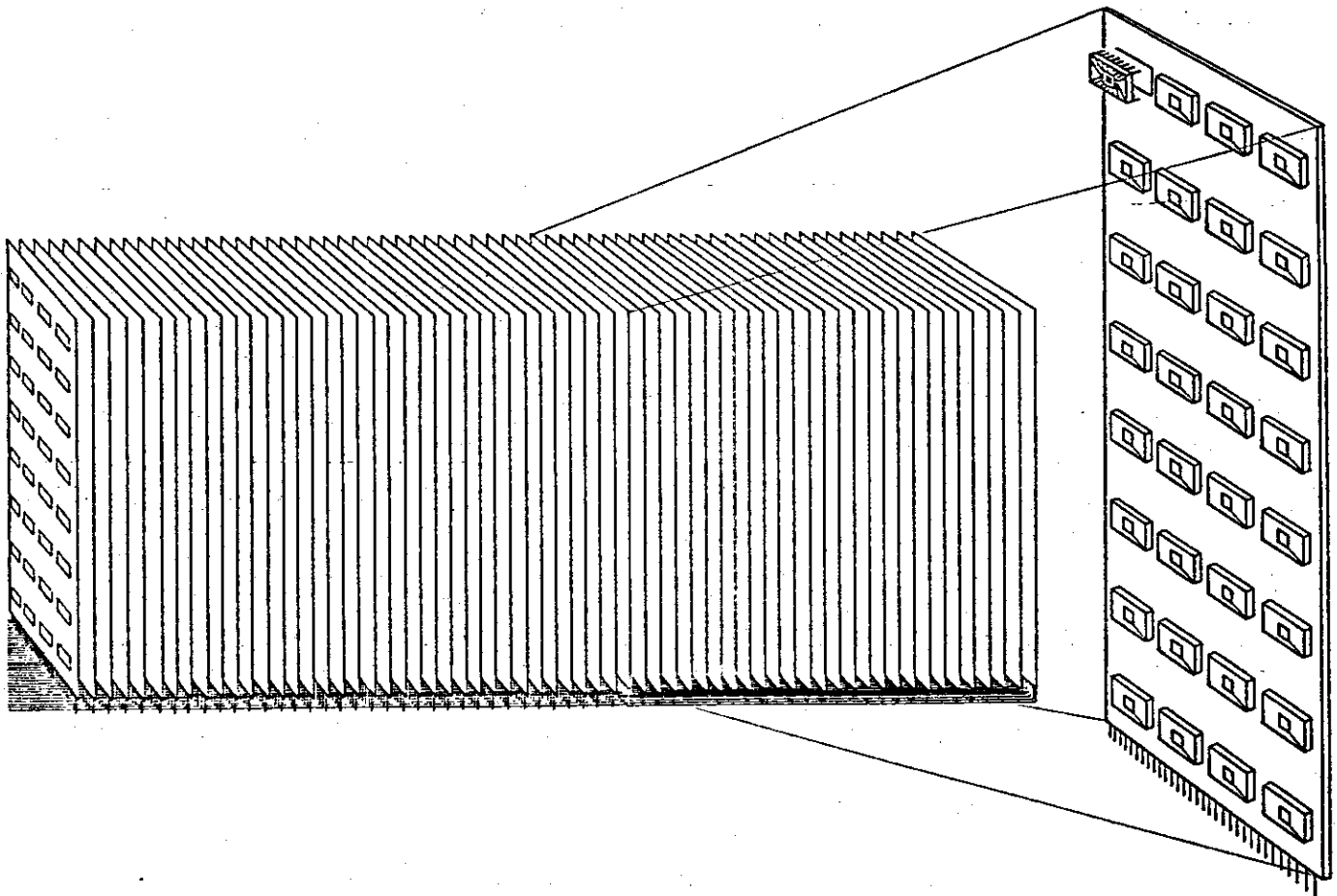
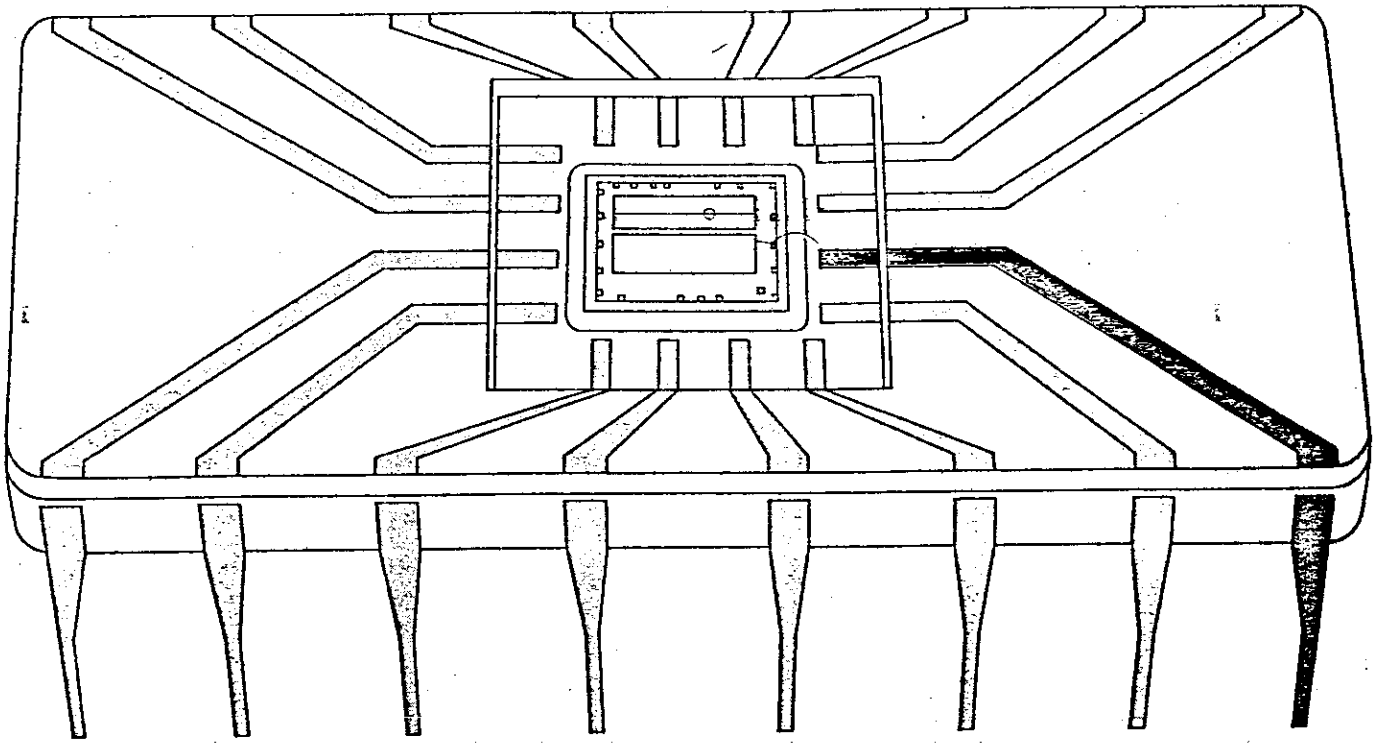


COMMUNICATION "BUS" connects a computer's central processing unit to memory modules and other peripheral units (top). It is typically a flat cable of between 20 and 100 long wires that are tapped as they pass through each of the connected units. Photograph shows Digital Equipment Corporation's UNIBUS (wide, light-colored flat cable) connecting two disk memory units (upper left) with central processing unit (lower right) of DEC's PDP-11/40 computer.

very regular patterns of interconnection. There is already a trend toward very regular wiring patterns for integrated circuits and the interconnections among circuits. Read-only memories, for example, implement complex and irregular logic functions with a simple and very regular integrated circuit pattern. This regularity is desirable not only because it makes the specification of such functions simple but also because it may be the most efficient layout from an interconnection point of view. We believe regular patterns of wiring will play an increasing role in future designs. In part, computer science will become the study of the regularity of these structures.

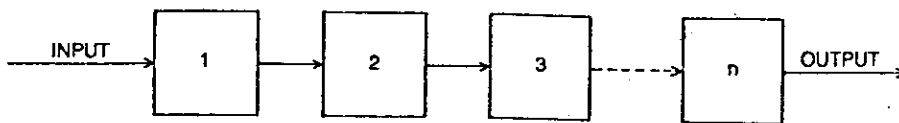
The architecture of a typical computer includes a single logical processing element that communicates with a random-access memory through 20 to 100 long wires combined into a "bus," which, like its namesake, provides public transportation for data but is actually more like a telephone party line. The communication bus is often a flexible cable 50 to 100 feet long. A signaling protocol is specified for the bus so that all the units to which it is connected communicate in a common way and avoid interfering with one another. The great advantage of a bus structure in a computer is that any unit connected to the bus can communicate directly with all other units. Moreover, the protocol and the bus structure may survive several generations of hardware development, so that a line of computing equipment can adopt new storage devices, new input-output units and even new processing elements. In addition, the number of switching elements devoted to communication in each unit on the bus is minimized because each unit needs to communicate with only the one bus to send messages anywhere.

The drawback of the bus structure is that it provides a communication bottleneck. Consider a typical computer with, say, one million words (32 million bits) of integrated-circuit storage built out of 2,048 circuits that store some 16,000 bits each. Any one reference to memory can potentially sense the values of 128 bits on each of the 2,048 integrated circuits constituting the memory. Of these quarter-million bits to which access is obtained on the integrated circuits only 2,048 (one from each integrated circuit) are delivered outside the integrated-circuit package, and of these 2,048 only 32 are delivered over the communication bus to the logical processing unit of the computer. It is assumed that the communication bus connects the memory and the logical processing unit; we assert that in fact it separates them. Each memory access in a large computer wastes access to many thousands of bits

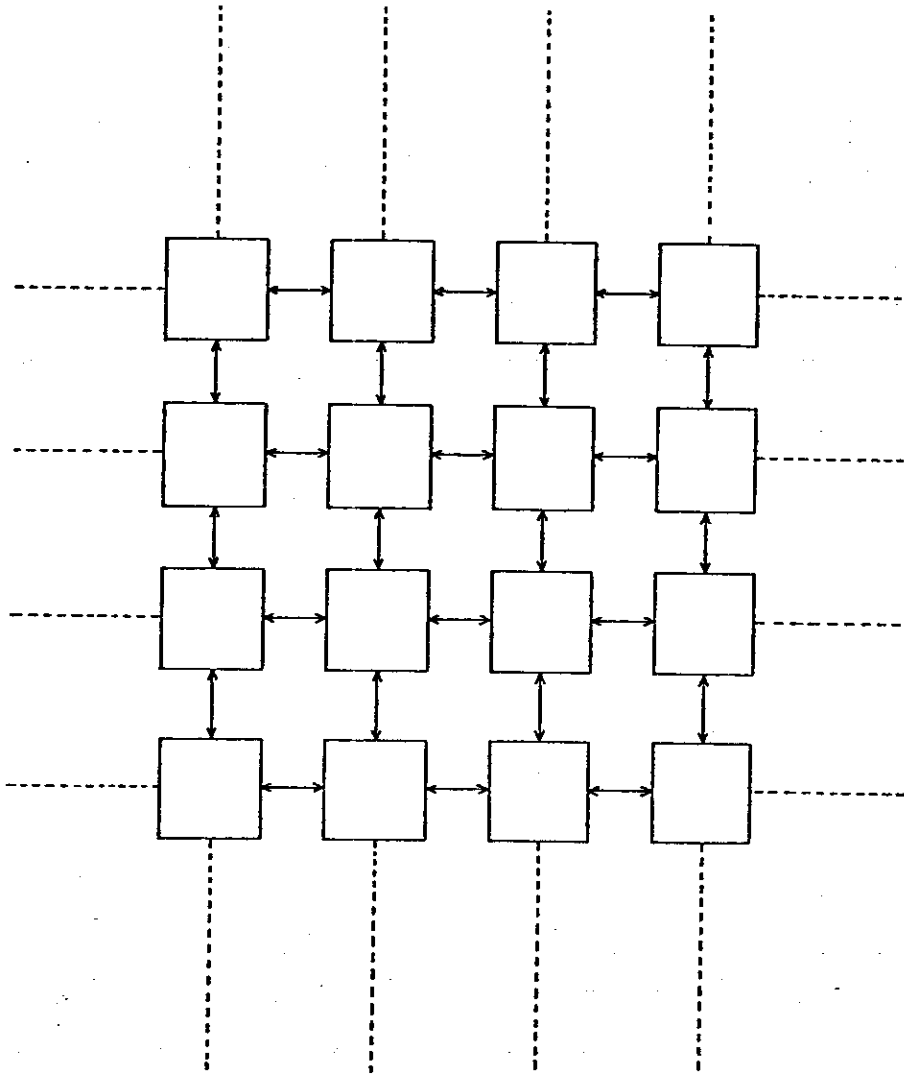


TYPICAL MEMORY CHIP has 16,384 bits arranged in a 128×128 array (top). An entire row of 128 bits can be accessed at one time, but a selector enables only a single bit to pass to an output pin (dark color). A typical memory system is made up of 2,048 such chips, say 64 groups of 32 (bottom). Only 32 chips can place their outputs on

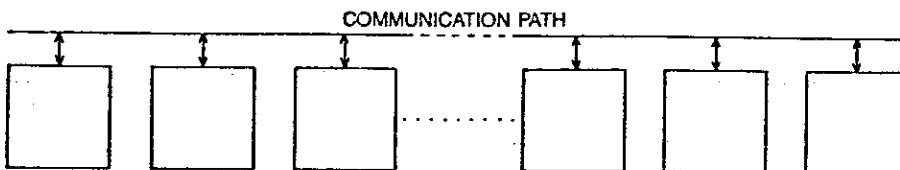
the 32 wires that join the bus to the central processor. Of the 262,144 ($128 \times 2,048$) bits that moved less than a millimeter on each chip, only 2,048 moved three millimeters to get off their chip and only 32 moved a meter to the processor. In other words, the bus utilizes only about an eight-thousandth of the memory chips' available "bandwidth."



"PIPELINE" PROCESSOR is one of three kinds of parallel processor, illustrated on this page, that have been effective. In a pipeline processor data are passed along from one specialized processing element to the next, with each element performing a successive operation on the data. The pipeline is analogous to an assembly line: all operations are conducted simultaneously but not on the same material. The pipeline configuration is optimum as long as the same basic type of operation is to be performed; it is less effective when the operations are variable.



ARRAY PROCESSOR is effective when much identical processing is to be done on many items of data. All the processors receive the same instructions, like a company of soldiers drilling "by the numbers." The limitation here is that individual computations must depend only on the data in a particular element and its immediate neighbors. This can be effective, however, in operations such as weather simulation, where local atmospheric interactions are significant.



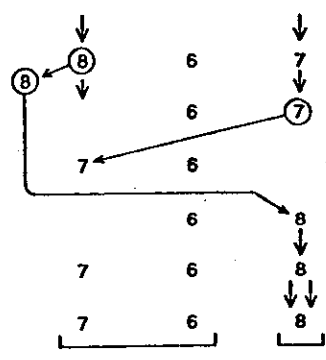
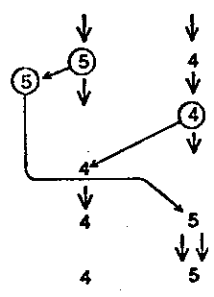
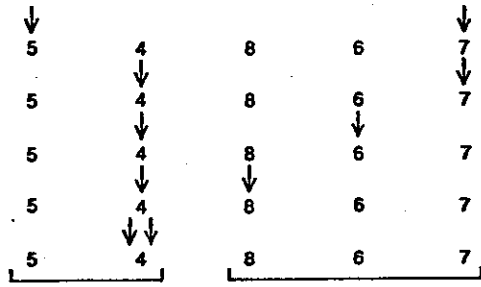
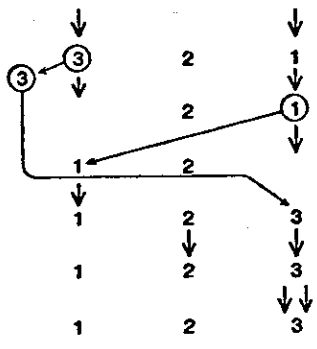
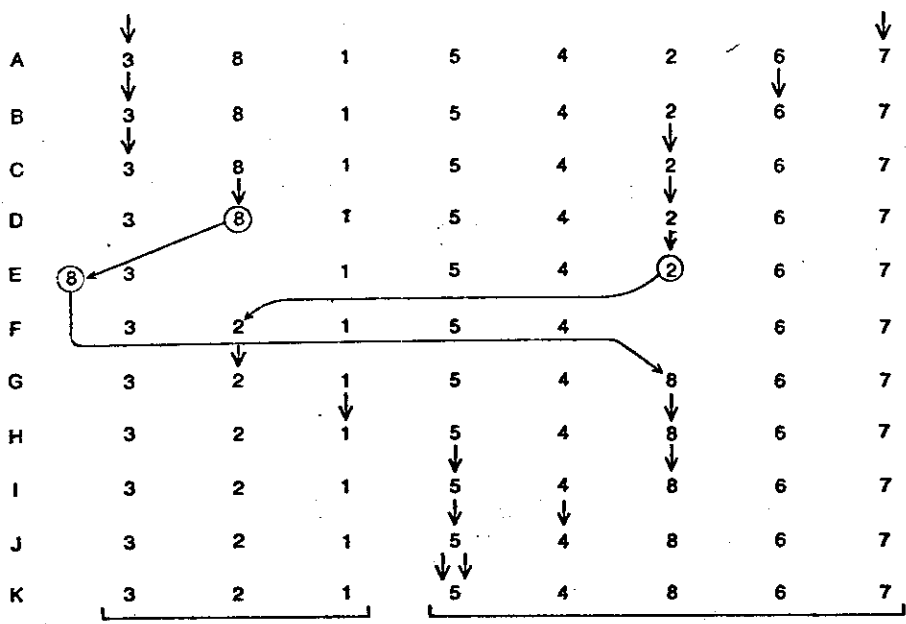
INDEPENDENT PROCESSORS connected by a communication path constitute the most flexible arrangement for parallel execution of different operations. Tasks are given to each processor as is required, as they are to the individual workers in a cottage industry. The system works best when each element can do much processing and need not communicate much with other elements; bottlenecks develop when tasks require elements to wait for the party line.

by selecting only a few bits to send over the memory bus to the central processing unit. This waste is tolerated for two reasons. First, it simplifies our conception of the machine and matches it to our natural inclination to do one thing at a time. Second, it provides a single, simple interface between various parts of the machine.

We pay a high price for this convenience. In an age when memories and logical processing elements were made by different technologies, we had little choice. Now, however, with the silicon integrated circuit dominating both the memory and the logical processing tasks in computers, there is little justification for continuing to accept such waste. Now it is possible to distribute the memory bus over many thousands of integrated circuits, in effect giving each logic element the memory it needs by moving information less than a millimeter from memory to processing facilities located together in the same integrated circuits on each of many thousands of chips. We are just beginning to explore systems with this unconventional architecture. To employ them effectively we must learn how to match the complexities of given problems to the simple fixed patterns of communication provided in the systems we can build.

Machines in which large numbers of logic elements operate simultaneously are called parallel processors. (To some extent, to be sure, every computing machine is a parallel processor. The separate bits that together represent a number are moved simultaneously on parallel communication paths; binary addition is performed by an adder circuit that operates on all the bits of the number at once; multiplication is performed either by sequential addition or, in faster models, by including a number of separate adder circuits and operating them in parallel. Levels of parallelism above basic arithmetic, however, are rare in today's computers.) The simplest form of real parallel processing now available has a few independent processors operating on a common memory; a typical large computing system has from two to a few dozen processors at work. Typically, however, these processors serve quite independent functions and their very existence may be hidden from the user. For example, a separate processor may be involved in communication with the user's keyboard, in operating magnetic-tape or magnetic-disk input-output units or in scheduling the resources of the central processor. Such "multiprocessing" systems have little impact on the user's algorithms.

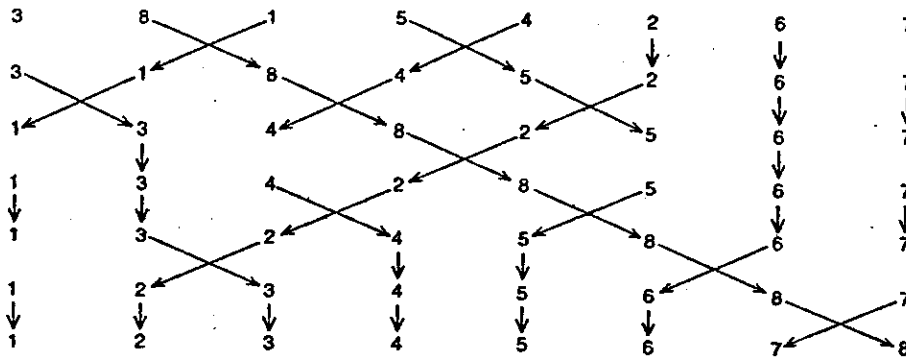
Three kinds of systems that can truly be classed as parallel processors have been built. In one of them, the "pipeline" processor, several processing elements,



each of which is specialized for some particular task, are connected in sequence. The work to be processed flows through these processors much as workpieces move along an assembly line. Communication is simple because information flows along a fixed pathway and has only a short distance to move between processing stages. A pipeline processor gains efficiency for the same reasons an assembly line does: functions are specialized and communications are minimized. Pipelining enables the arithmetic sections of very fast computers to process sequences of numbers with a high overall speed. Pipeline processors are less effective where the tasks to be performed are highly variable.

In a second form of parallel processor many identical processing elements are brought to bear on separate parts of a problem under the control of a single instruction sequence. Several such machines have been built, of which the largest and best known is ILLIAC IV. A modern parallel processor of this type was proposed at a recent Rand Corporation workshop on hydrodynamic simulations. In this hypothetical machine there would be 10,000 processors, each with arithmetic capability and memory, each built on a single integrated circuit and all under the command of a common instruction device. All the processors would execute commands in rigid lock-step. The processors would be arranged in a square array, 100 x 100, and each would communicate data only with adjacent processors to its north, south, east and west in the array, with relatively slow bit-serial communication on a single wire in each direction. We estimate that such a machine would take about five microseconds to communicate a single 64-bit number from one processor to its neighbor, which is very slow by today's standards. Of course, it could communicate 10,000

"QUICKSORT" is a typical sequential algorithm for arranging numbers in ascending order. Numbers pointed to by arrows are compared with the number farthest to the left. If the pointer farthest to the right indicates a number greater than the reference value (row A), it is advanced to the left until it rests on a number less than the reference value (C). Then the left pointer is advanced to the right until it rests on a number greater than the reference value (D). At that stage the numbers pointed to are interchanged (E-G). The process is continued until the pointers rest on the same number (K); at that stage all numbers to right of pointers are greater than the reference value, and all numbers to left are less than or equal to it. The same algorithm is then applied to each subset; to complete the sorting illustrated here requires five more steps than are shown. To sort n numbers requires $n(\log_2 n)$ comparisons of numbers that may be stored in distant locations. The communication cost is the dominant cost of executing the algorithm.



PARALLEL EXECUTION speeds the sorting task. In this algorithm adjacent members of number pairs are compared and are interchanged if the left-hand member is larger than the right-hand one. (In the first row a "pair" is defined as two numbers the left one of which is in an even column; in the second row the left member of each pair is in an odd column, and so on alternately.) Sorting the entire set requires $n^2/2$ comparisons, always of nearby numbers. Interchange sorting has been considered slow, but if comparison and interchange elements are attached to each memory element in integrated circuits, comparisons required for one sweep can be accomplished in one memory cycle and the sorting can be completed in n cycles at most.

such numbers in any one five-microsecond period. It would also take about five microseconds to perform a multiplication, but again it could perform 10,000 multiplications in that time, for an average rate of two billion multiplications per second. Machines such as this one are called array processors or single-instruction-stream, multiple-data-stream machines. They are most suitable for highly regular tasks such as hydrodynamic computations, numerical simulation of the weather and the inversion of large matrixes.

A third type of parallel processor is one where separate, independent processors under separate, self-contained control structures perform independent parts of the task, communicating data and instructions as is required. The advent of the microprocessor has, of course, suggested to many people the possibility of making systems consisting of thousands of separate microprocessors and having them work in concert on large tasks. Few such multiple-instruction-stream, multiple-data-stream machines have been built, and their properties are poorly understood.

The challenge in designing or using a parallel processor of any of these three types lies in discovering ways in which simple patterns of communication within the processor can be made to match the communication tasks inherent in the problem being solved. As integrated-circuit technology progresses there will be individual circuits of increasing speed and complexity. No relief is in sight, however, for the costs and delays inherent in communicating information from one circuit to another. To provide better communication will require more connections to the integrated

circuit, a bigger housing for it, more or larger communication-driving circuits and consequently more heat dissipation. To obtain maximum performance from large computing systems programmers will have to face up to the limitations on communication that are imposed by physical reality. High-performance communication cannot be provided from every element to every other element; the programmer will have to match his formulation of the problem to the available communication paths. Although this is a difficult task, success in accomplishing it will provide unprecedented processing power.

We believe that just as an important part of today's computer science concerns itself with sequences of instructions distributed in time, so an important aspect of computer science in the future will be the study of sets of communications distributed in space. If processors can communicate only with their nearest neighbors, what kinds of arrangements are possible? Obviously one can wire processors in a linear string. Such processors can operate in the pipeline fashion described above, with each one passing data along to the next, or by performing common operations under command from a central instruction device. Such near-neighbor connections are highly effective for tasks such as sorting, in which the local communication of data suffices. Alternatively, one can connect processors in an array, with each processor having more than two neighbors. Such arrays have a basic structure much like the structure of a crystal, and various forms of local communication are possible. In our laboratory at the California Institute of Technology we are considering the properties of the different communi-

cation paths that might be included in such structures.

Some years ago R. S. Gaines and C. Y. Lee, who were then working at Bell Laboratories, described three types of interconnection path. One kind of interconnection has connections that are common to all processors; it is effective for sending commands to the processors and for "broadcasting" to all processors certain values that may be important in the course of a computation. A second kind of communication path enables each processor to "talk" simultaneously to its neighbors. Such a path can handle the communications required in pipelining or, if each processor has its own storage function, open up a space anywhere in the store by moving information simultaneously away from the location of the desired gap. A third kind of communication path enables the processing elements to say something collectively about their results. Such a path can indicate whether no processor, one processor or more than one contains a given condition, which of the processors contains the smallest value or which are between the beginning and end of a particular string.

In our laboratory we have taken on the task of building and using some simple parallel processors involving one-, two- and three-dimensional interconnection patterns. We hope to learn more about the relation of communication paths to the performance of such processors. We have become convinced that the performance of parallel processors can depend critically on the design of communication paths that enable processing elements to make collective statements about their actions. Without such paths how does one find the smallest value stored in the array? How can one identify the set of processors that lie between two processors with designated properties? How does one obtain answers from a number of processors in sequence when more than one of them has something to report? Such paths are electrically complex. Either they involve each local processor as the driving element in a "global" communication task or they require intermediate circuitry specialized for collecting such information, and such intermediate circuitry inevitably introduces time delays and cost. The best structures for this kind of communication appear to be similar to those in the carry circuits of fast parallel adders, but the communication costs of such circuits have not yet been adequately analyzed.

A cornerstone of computer science today is the theoretical analysis of sequential algorithms. There is a large and growing body of theory for selecting efficient algorithms for sequential machines. This body of theory, as one

might expect, focuses on algorithms that minimize the number of logical operations required to accomplish some task. For example, it has been shown that for putting numbers into sequence, "quicksort" algorithms are the best to use because they require only $n(\log_2 n)$ comparisons. This body of theory assumes that all data elements in storage are equally accessible and that the movement of data is free.

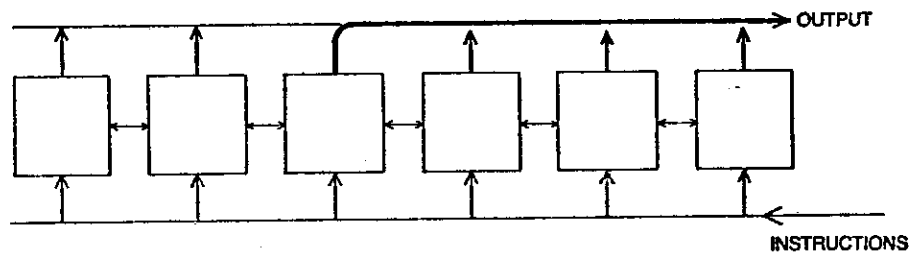
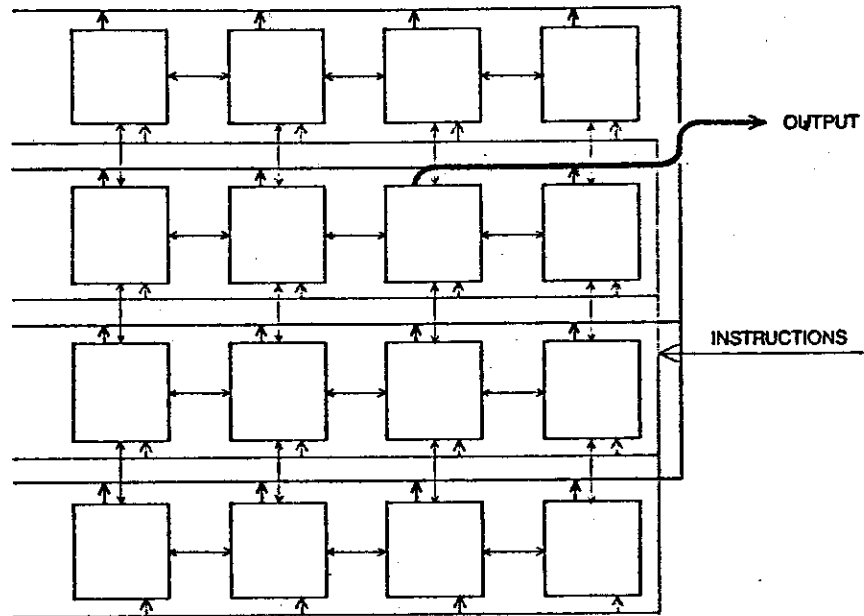
Data elements in storage are never really equally accessible, although they can arbitrarily be made equally inaccessible in a random-access device by making the access time for all elements as slow as the time required for the most inaccessible element. Because information in a random-access storage device must be moved over long distances the data rate in a random-access storage device of a given technology is inevitably lower than what can be achieved with a more orderly sequential-access mechanism. Moreover, transporting data from a memory cell to a comparison circuit is never really free; in most machines the transportation time far exceeds the comparison time. And so, for example, an analysis of algorithms seeking to show that a particular sorting algorithm is best is based on giving what may be a less than optimum machine the task of performing that algorithm; given a different structure, sorting might be done much faster.

Work on the theory of algorithms has not yet focused on the true relation of computing costs to communication costs. Given that one starts with a blank piece of silicon and is free to place wires, logic gates and so on anywhere one chooses, what choices should one make to accomplish a given computation task in the least time or on the smallest possible area of silicon? We have no basis at all for making sensible choices as to the computing structures we should build. We do have a large body of experience with a particular structure: sequential machines with random-access storage. It may be that such machines are effective because overall they best match a wide variety of computing tasks. There is mounting evidence, however, that a parallel structure can outperform the standard computer by many orders of magnitude on tasks that are suitable for parallel execution. As such unconventional structures have appeared, a wider range of tasks is being discovered for which they are suitable. Certainly one would expect to obtain better performance by paying attention to the real costs of systems, which is to say to communication, than by simply considering the cost—now an almost vanishing cost—of logical processing.

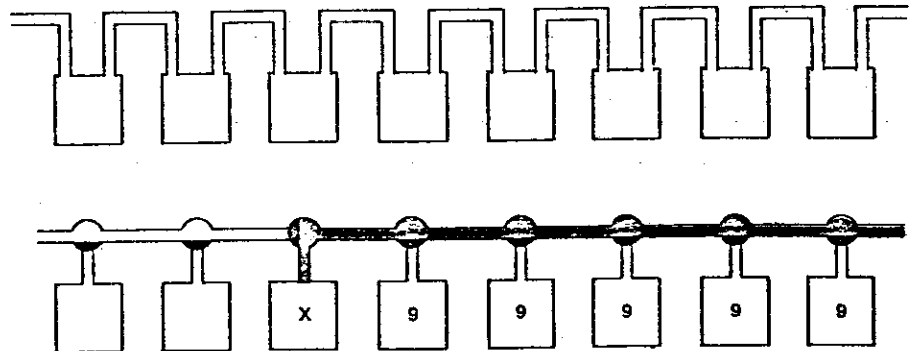
We believe adequate theories that account properly for the costs of communication will be an important guide for

designing the machines that have been made possible by the integrated-circuit revolution. We believe such theories will have their basis in the study of regularity, so that computer science will come to include a body of theory akin to that of topology or crystallography. Al-

though such a development is revolutionary in some ways, it is essentially a continuation of the search for regularity in all programming tasks. Computer scientists will simply add geometric regularity to the logical regularity they have already come to know and value.



WIRES THAT INTERCONNECT MODULES of an array processor can be exploited in three distinct ways. One type of connection "broadcasts" information from a control center to all modules (gray). A second type (black) moves information from a selected module to a control destination, one module at a time (heavy black line). A third type passes data from each module to its nearest neighbor (color); in this case all the modules can "talk" at the same time.



TWO SUBTYPES of the third type of wiring are in common use. In one subtype (top) information passes into a module during one step, is processed and then passed on to the next module during the next step. The other subtype (bottom) is designed merely to "discover" something about a module (or a number of modules collectively) and to do so quickly; information is moved past a module unchanged during a single processing step, provided that the module is in a specified state. Such wiring might discover, for example, where the 9's are in a parallel adder.

The Authors

IVAN E. SUTHERLAND and CARVER A. MEAD are respectively director of the computer-science department and professor of computer science at the California Institute of Technology. Sutherland was educated at the Carnegie Institute of Technology, Cal Tech and M.I.T., receiving his Ph.D. in electrical engineering from M.I.T. in 1963. He then entered the Army Signal Corps and was stationed at the National Security Agency, where he designed a new line of computer-display equipment. In 1964 Sutherland was made director for information-processing techniques at the Advanced Research Projects Agency (ARPA) of the Department of Defense, staying in that position after his discharge from the Army. He left ARPA in 1966 to become associate professor of electrical engineering at Harvard University. Two years later he moved to Salt Lake City to become president of the Evans & Sutherland Computer Corporation, of which he was cofounder. At the same time he continued his research at the University of Utah, where he helped to design a computer-graphics system that could create color images with lifelike shading and perspective. He joined the Cal Tech faculty in 1976, but he still is a consultant for Evans & Sutherland and serves on the Defense Science Board. Mead was educated at Cal Tech, where he received his Ph.D. in electrical engineering in

1959. His research began in the area of solid-state-device physics, with a brief foray into the biophysics of nerve membranes. It was during his calculation of the physical limitations on how small a transistor could be and still operate that he became absorbed in the problems of managing the complexity of large-scale integrated systems. Mead escapes occasionally from the world of high technology to raise hazelnuts on his 70-acre ranch in Oregon.

Bibliography

THE SHORTEST PATH THROUGH A MAZE.

Edward F. Moore in *The Annals of the Computation Laboratory of Harvard University: Vol. XXX, Proceedings of an International Symposium on the Theory of Switching, Part II*. Harvard University Press, 1959.

FUNDAMENTAL LIMITATIONS IN MICRO-ELECTRONICS, I: MOS TECHNOLOGY.

B. Hoeneisen and Carver A. Mead in *Solid-State Electronics*, Vol. 15, No. 7, pages 819-829; July, 1972.

LIMITATIONS IN MICROELECTRONICS, II: BIPOLAR TECHNOLOGY.

B. Hoeneisen and Carver A. Mead in *Solid-State Electronics*, Vol. 15, No. 8, pages 891-897; August, 1972.

HOW BIG SHOULD A PRINTED CIRCUIT BOARD BE?

Ivan E. Sutherland and Donald Oestreicher in *IEEE Transactions on Computers*, Vol. C-22, No. 5, pages 537-542; May, 1973.

SCALE IN λ :
 1 2 3 4 5 6

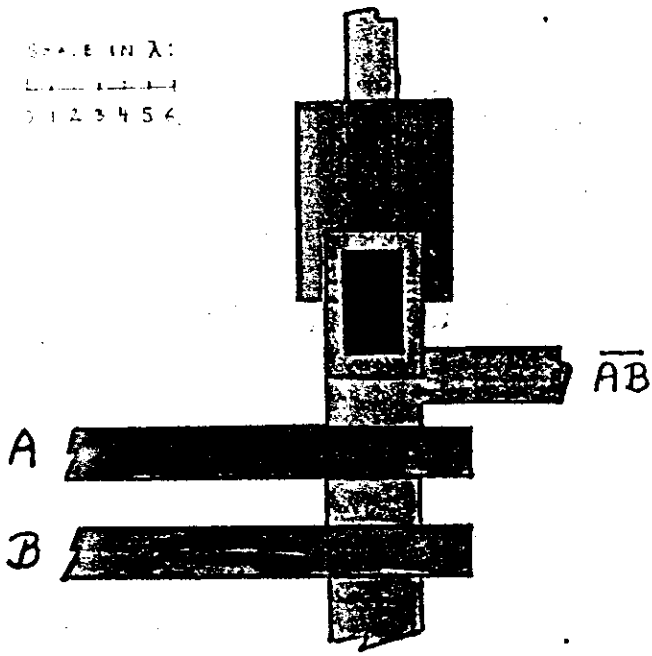


Fig.2a. NAND Gate: Layout Geometry

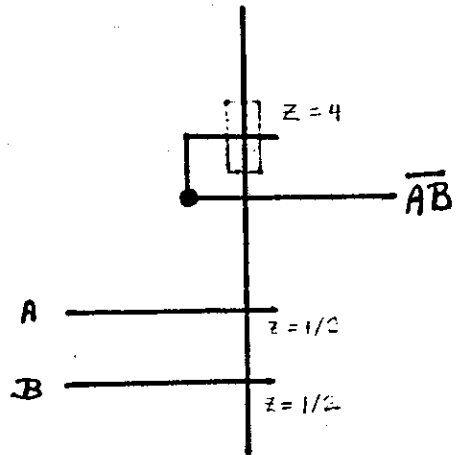


Fig.2b. NAND Gate: Topology (Stick Diagram)

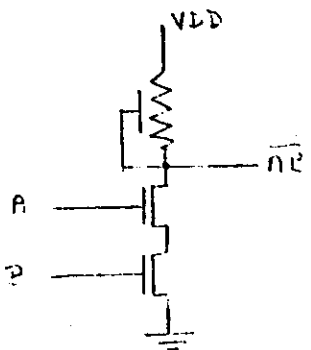


Fig.2c. NAND Gate: Circuit Diagram



Fig.2d. NAND Gate: Logic Symbol

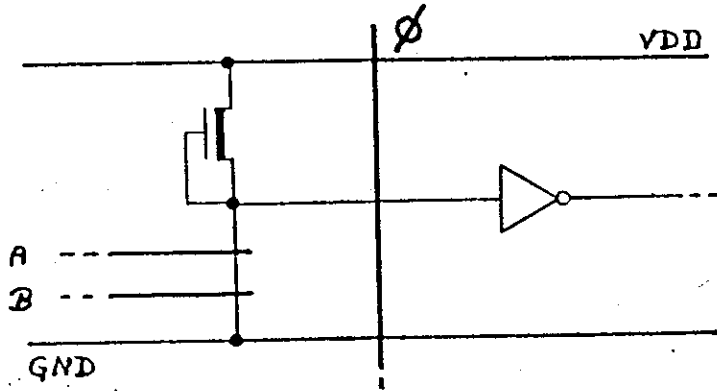
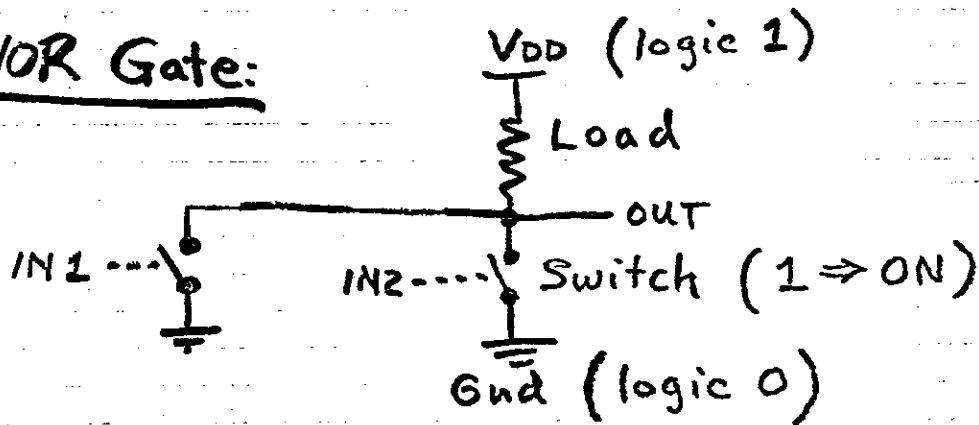


Fig. 2e. Example of Mixed Notation

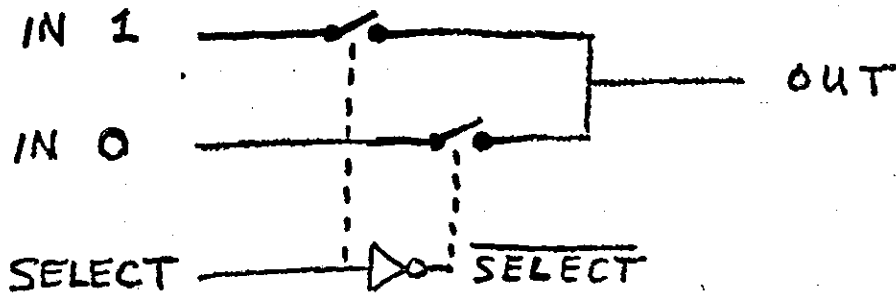
(Technology Independent)
Simplified Transistor Models
for the Digital Illusion:
Switches and Loads

Rule: Every Transistor is
either a switch or a load.

NOR Gate:



Switch Logic Multiplexor:

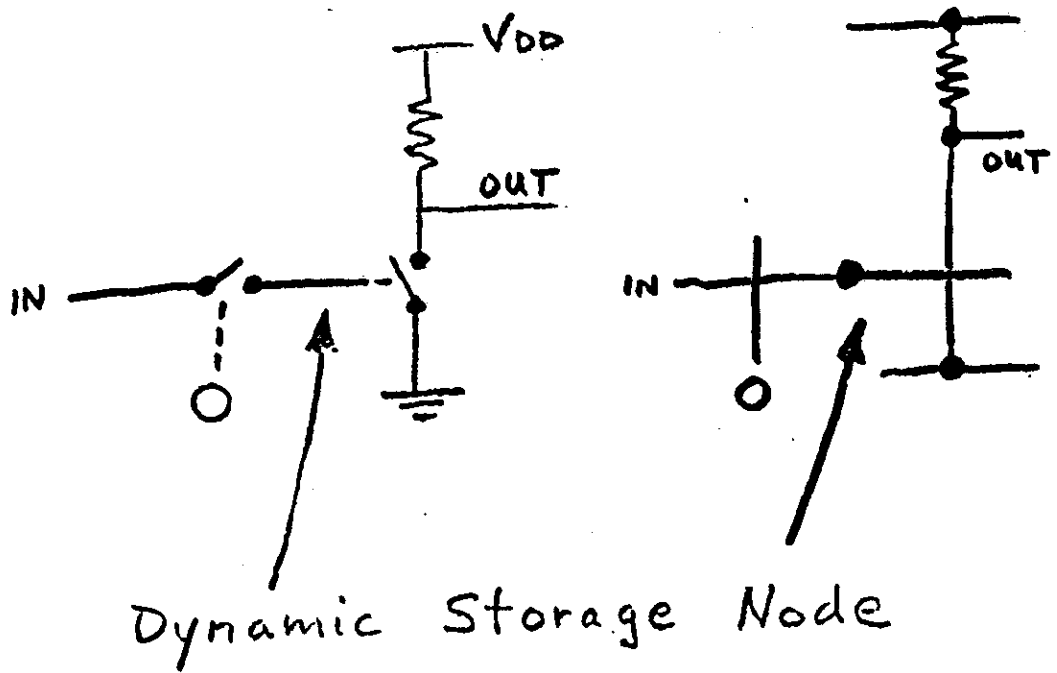


These are the two basic types
of combinatorial logic.

(FET Technologies)

Dynamic storage:

If the control line of a switch is connected to no load and no "ON" switch terminal, then it remembers its state — the controlled switch remains ON or OFF.



OUT = complement of stored value.

Two Things to Design:

Registers:

Gates with storage
nodes at inputs.

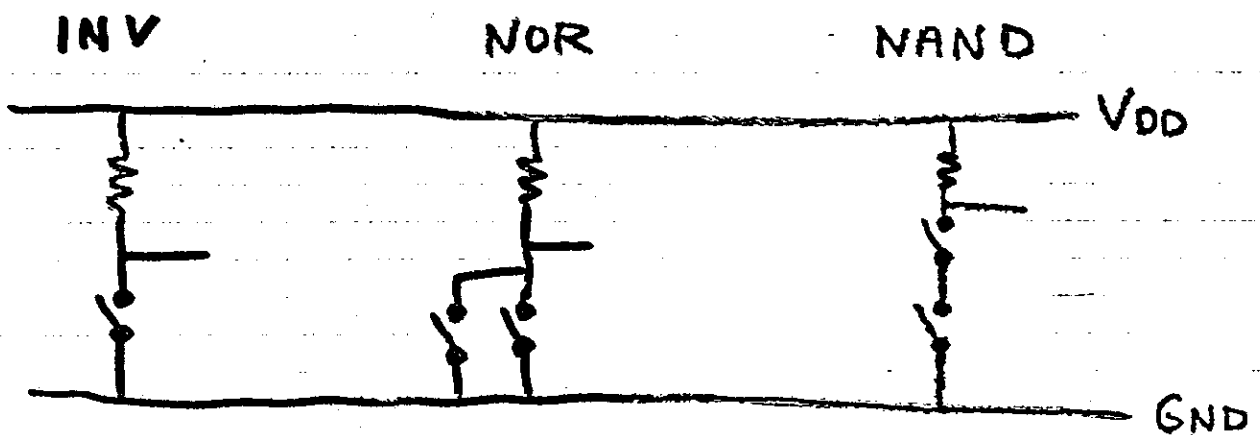
Combinatorial Logic:

1. Switch logic
(pass transistor logic)
2. Gates
(level-restoring logic)

Large memories & analog circuits
are another subject.

Gates

Def: A Gate is a load plus a switch network connected to Ground.



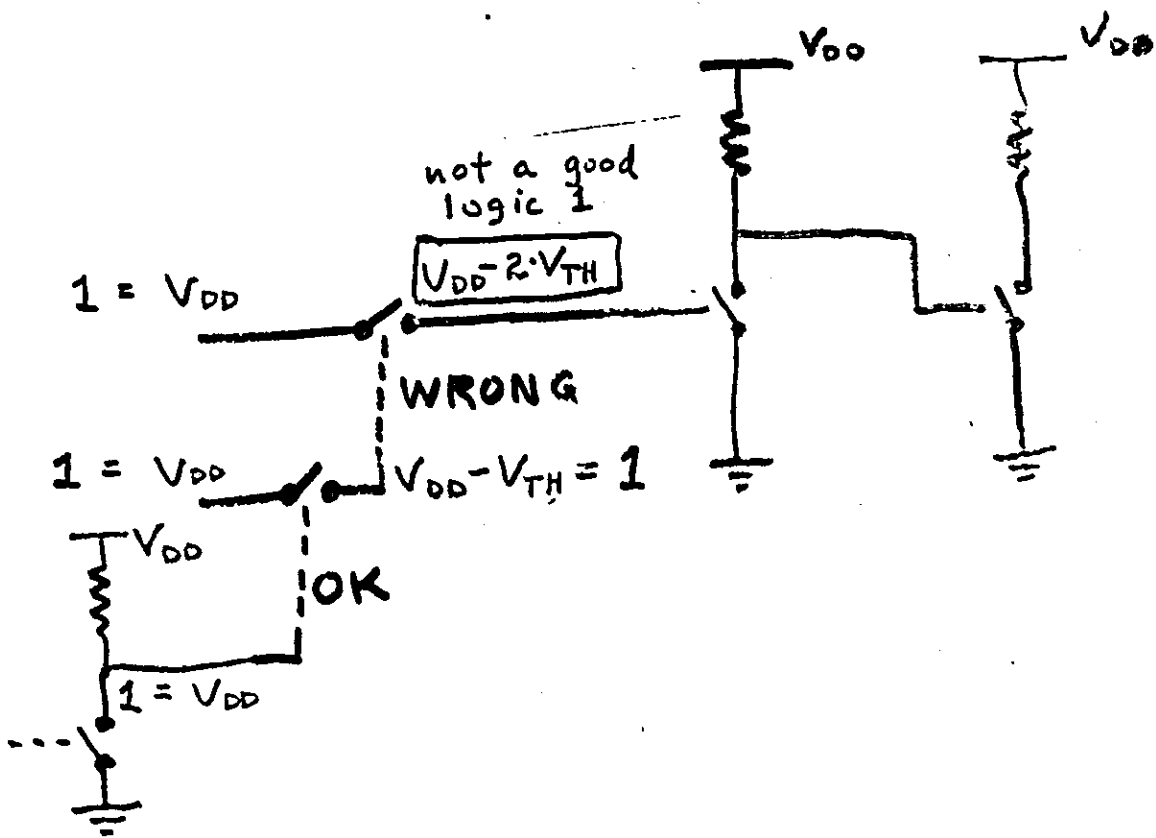
Notice: Entire systems can be built with only Inverters, if pass-transistor logic is used.

Trading Gates vs. Switch logic is a large part of the game.

Use switch model: $1 \Rightarrow \text{ON}$.

Exception: (in NMOS)

Switch logic output should not control more switch logic.



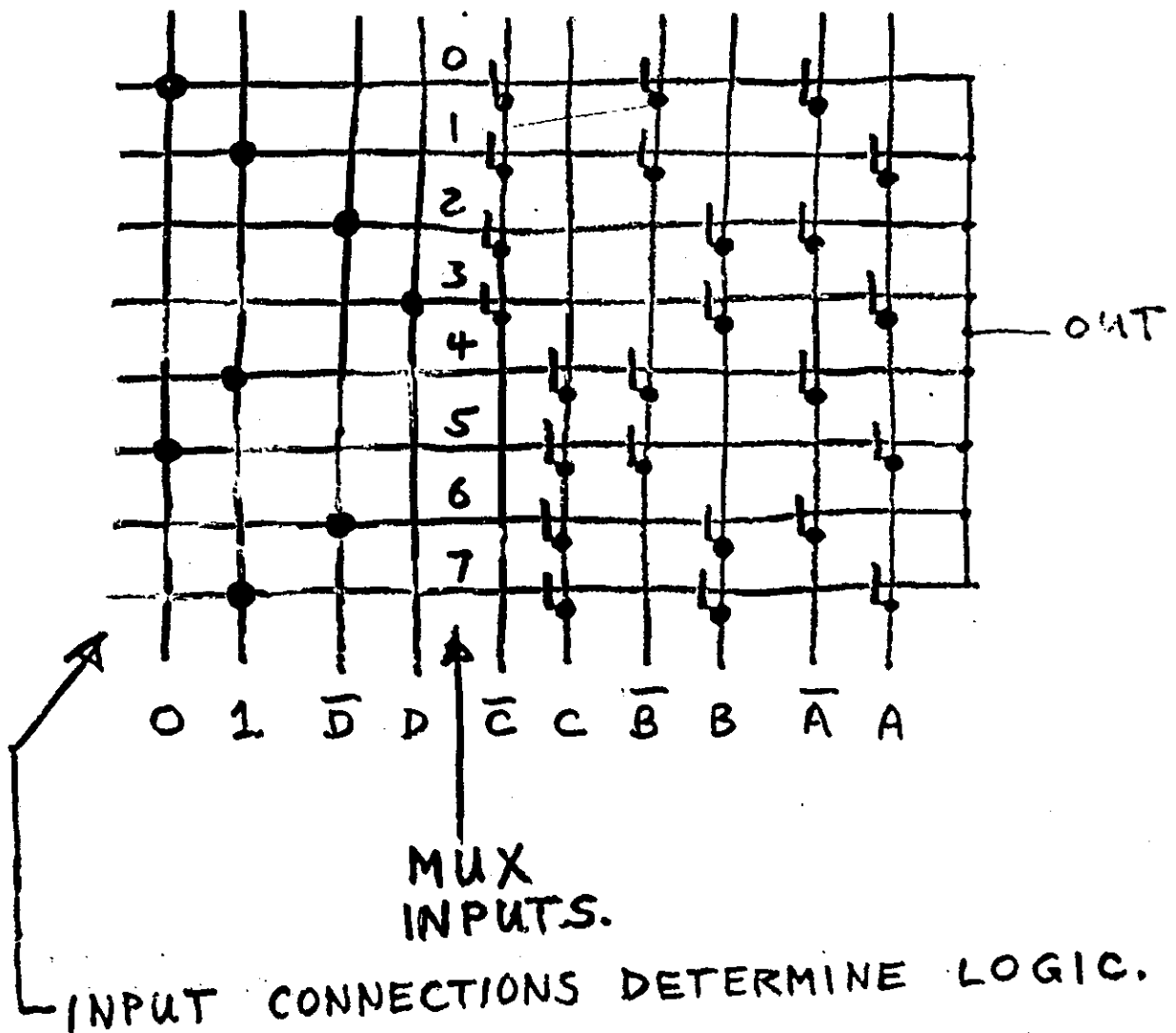
Why? Because switch is "ON" only if control gate is HI w.r.t. one of the terminals ("source"); hence level-restoring is needed.

— understand this —

Multiplexor Logic

Example:

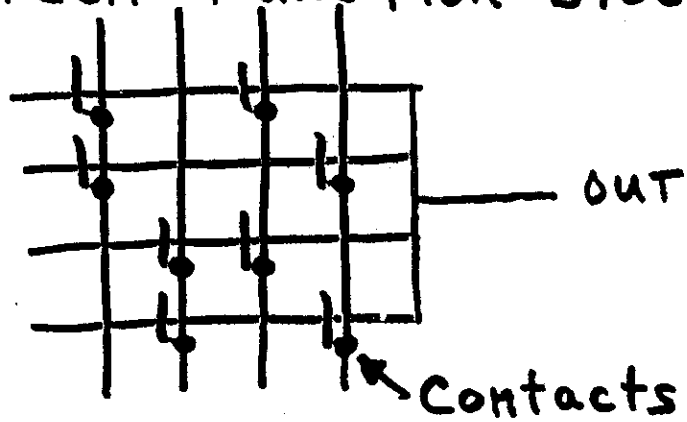
Arbitrary 4-input logic
using an 8-to-1 Mux:



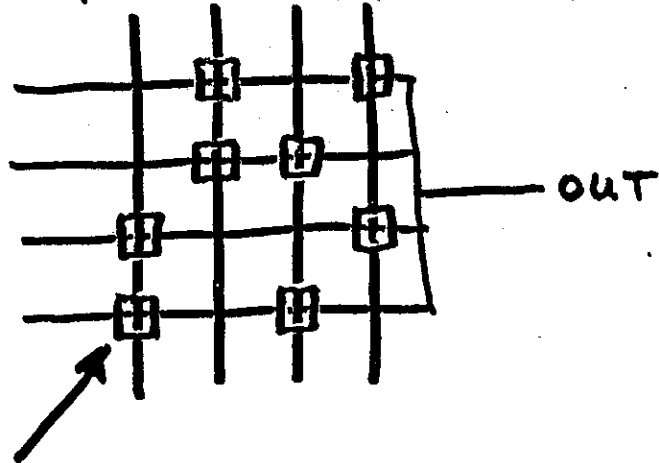
INPUT CONNECTIONS DETERMINE LOGIC.

Note: Inverse of Input
is usually useful.

Blue-Green Function Block:



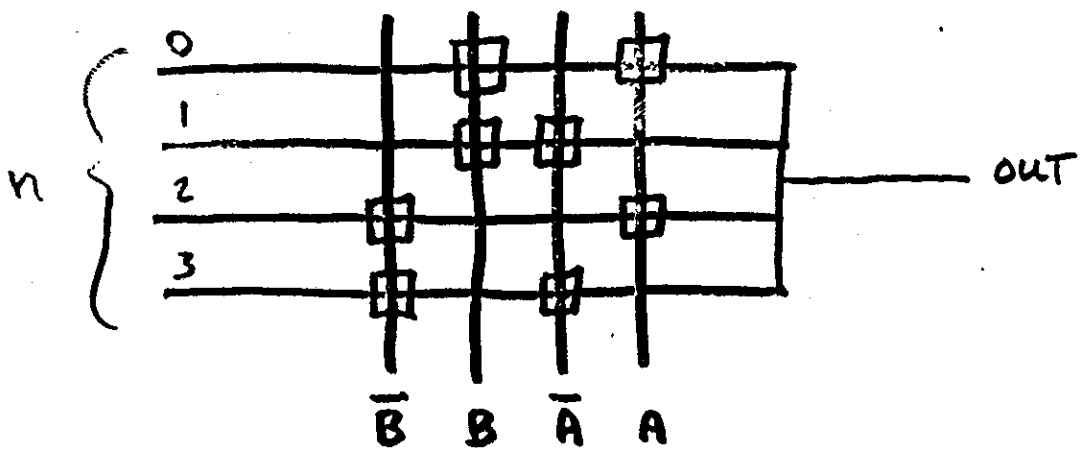
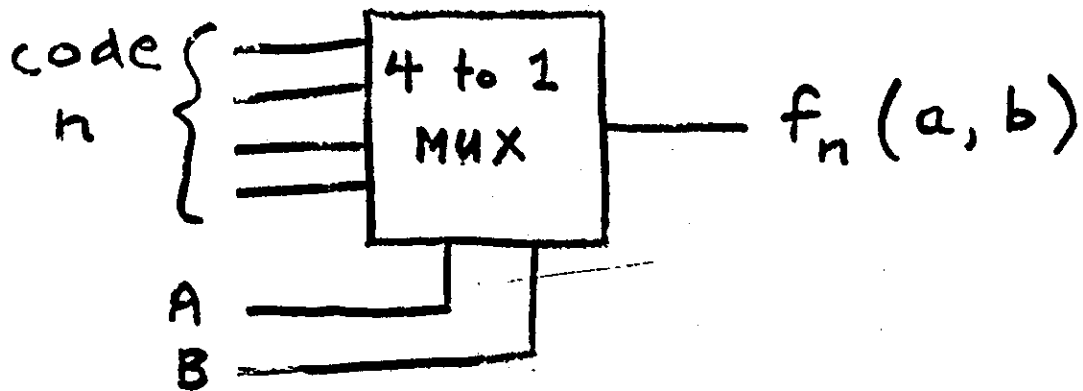
Red-Green Function Block:



"Yellow" Switch -
always ON;

a trick with advantages
and disadvantages.

Multiplexor Logic:
"Function Block" generates
all 16 functions of 2 inputs:



See OM2 H.L.U - Chapter 5.

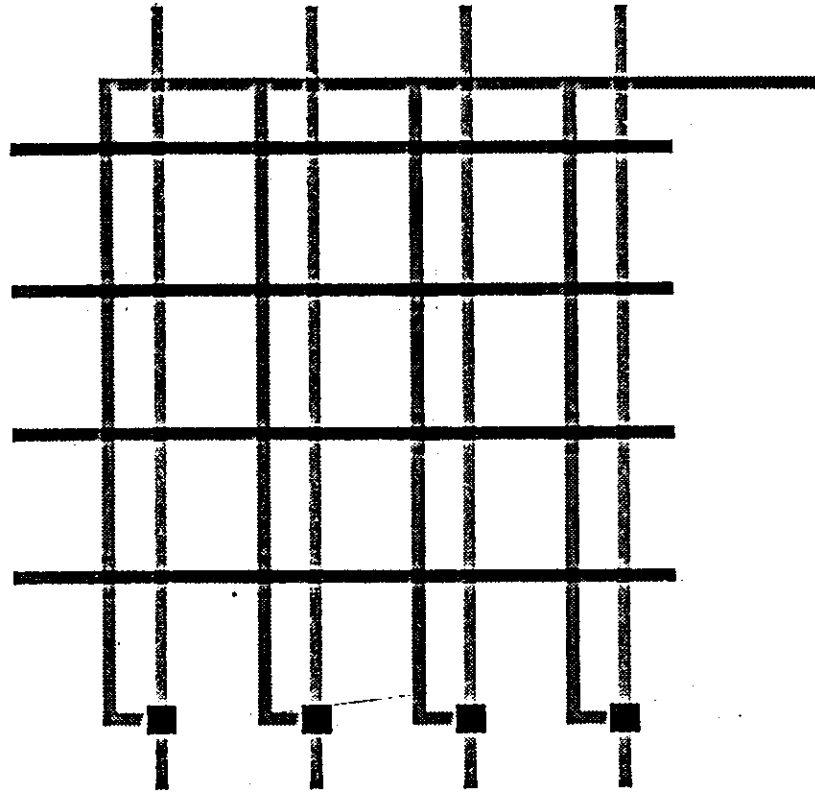


Fig 4a. Stick Diagram of the Function Block

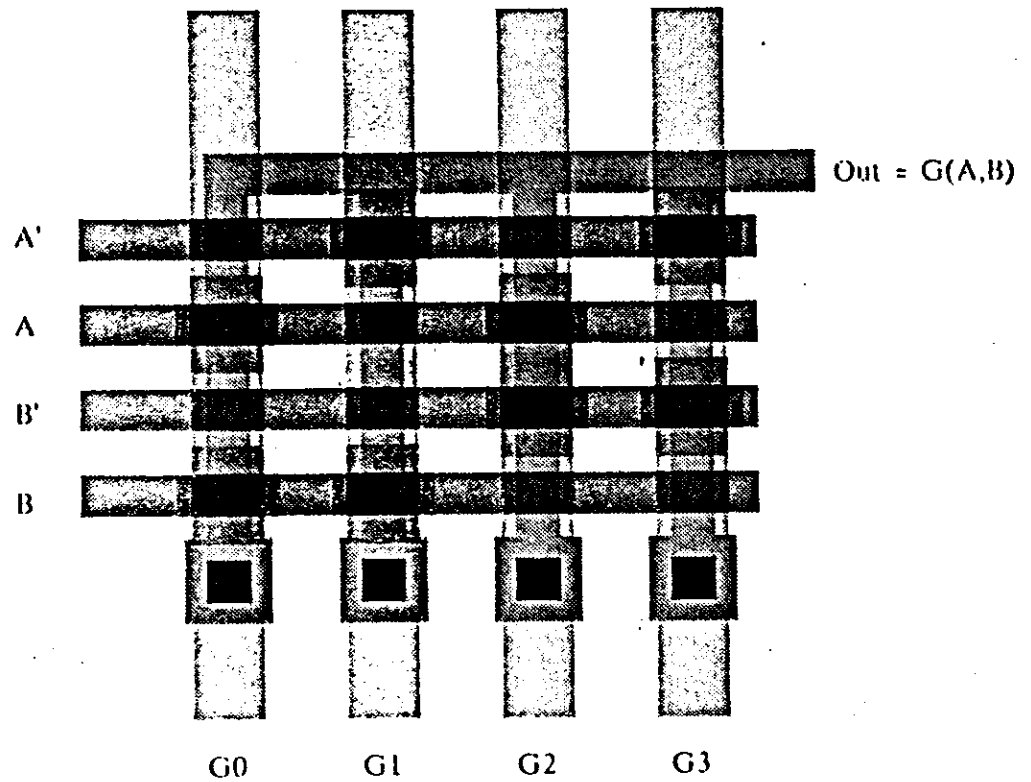


Fig 4b. Actual Layout of the Function Block

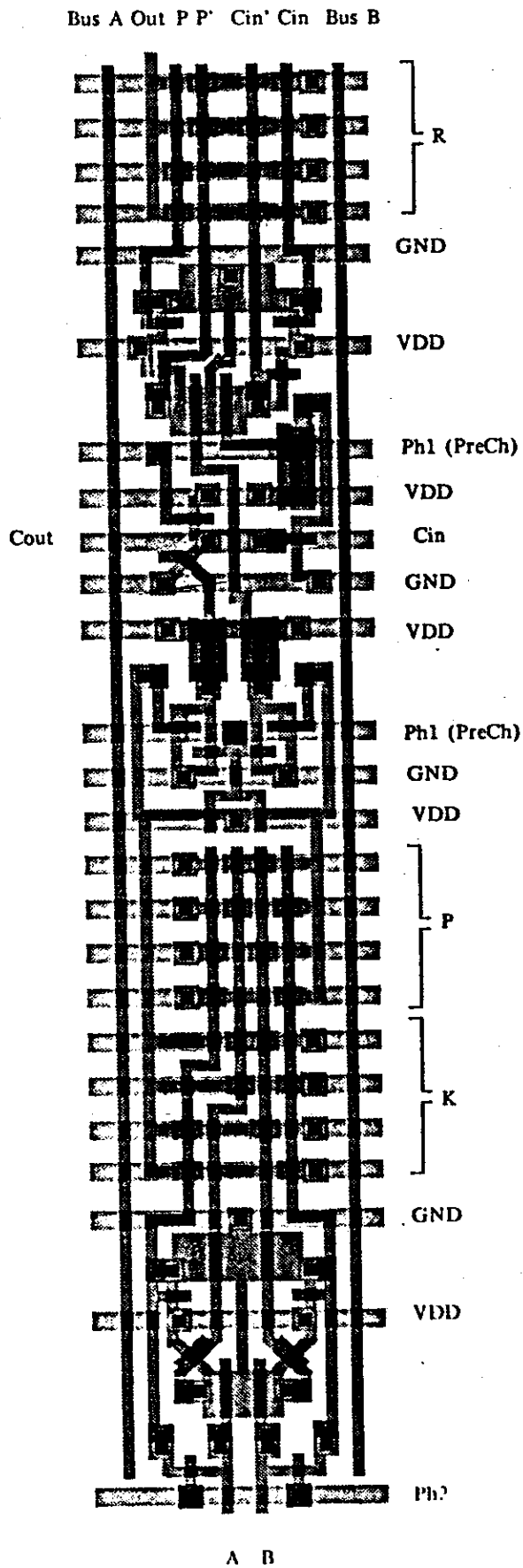


Figure 6a. Layout of ALU Bit Slice and Input Registers

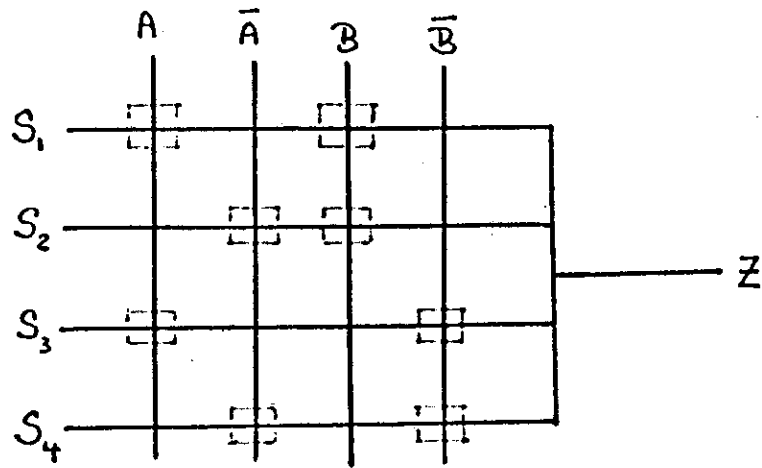


Fig.12a. Selector Logic Circuit

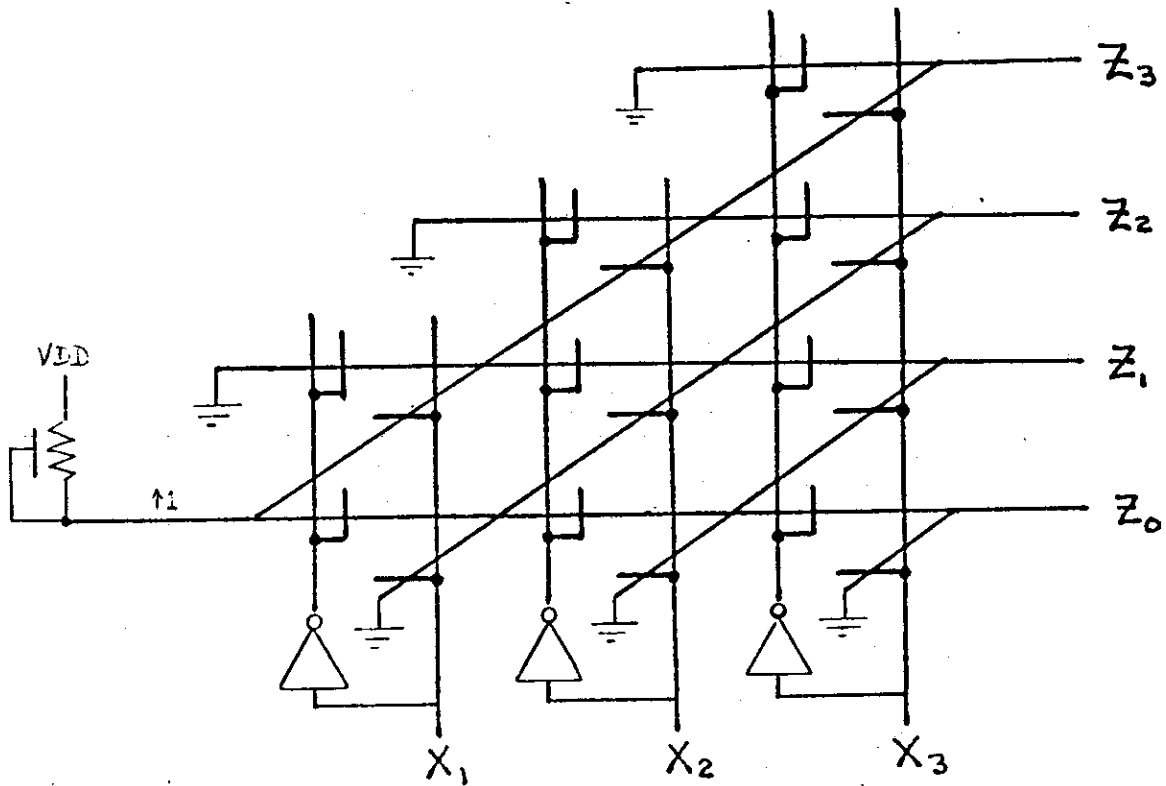
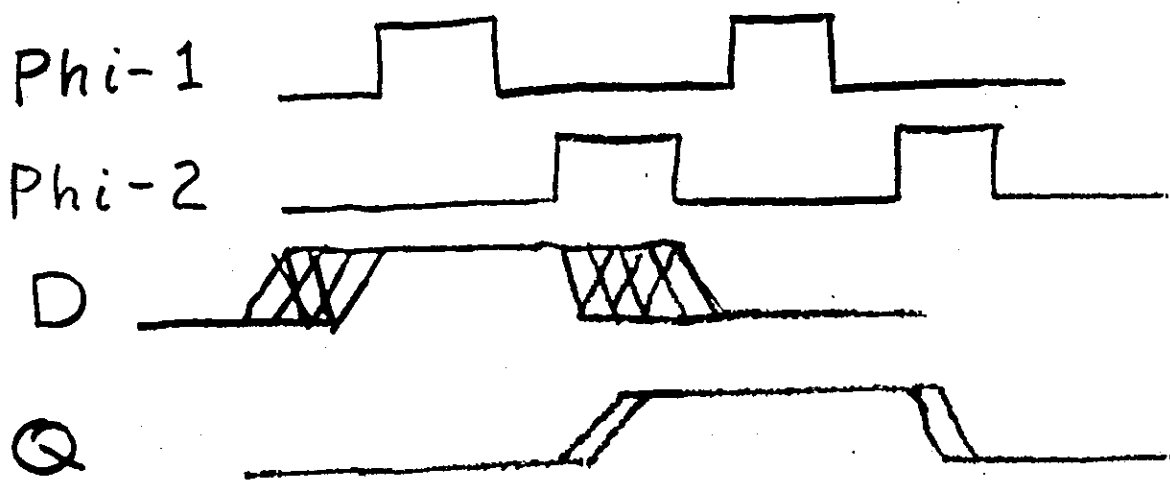
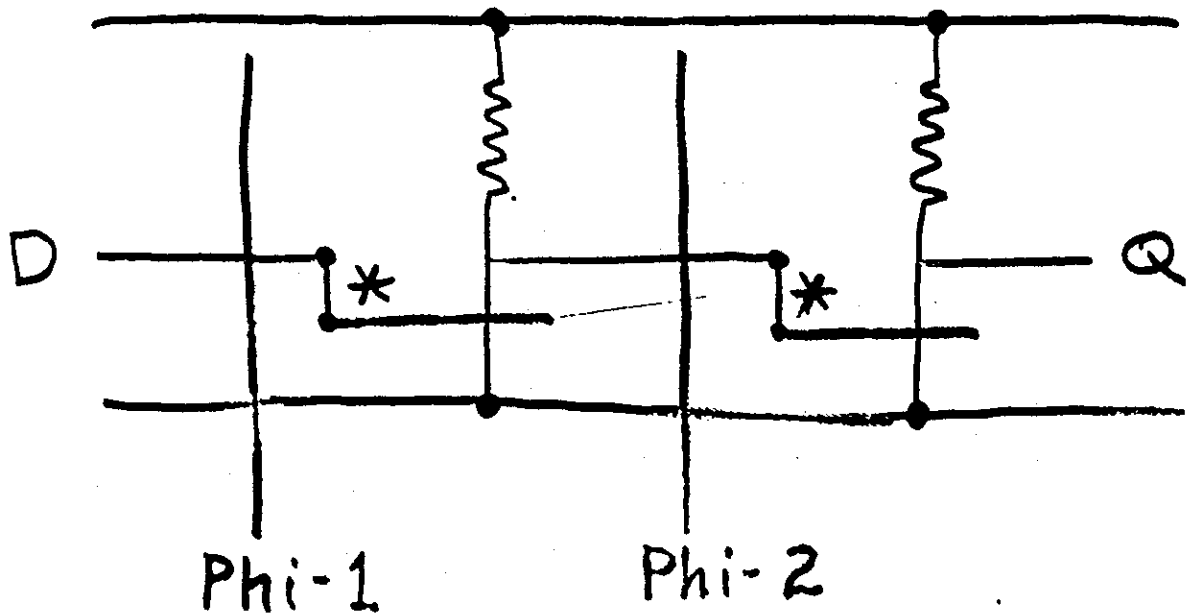


Fig.12b. A Tally Circuit

Registers

2-Phase Timing Scheme



D to Q is 1 cycle delay.

We also use Half-Registers.

2-Phase Methodology

All state is stored dynamically.

No flip-flops!

No loops in combinatorial logic.

Exceptions are seldom needed!

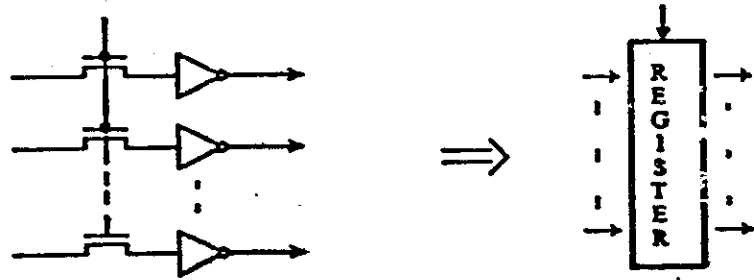


Fig. 7. A Dynamic Register

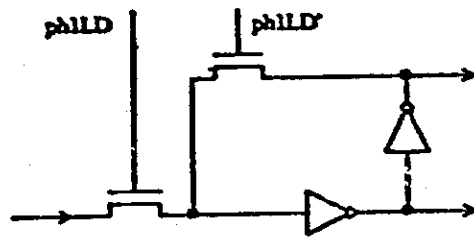


Fig. 8. A Selectively Loadable Dynamic Register Cell

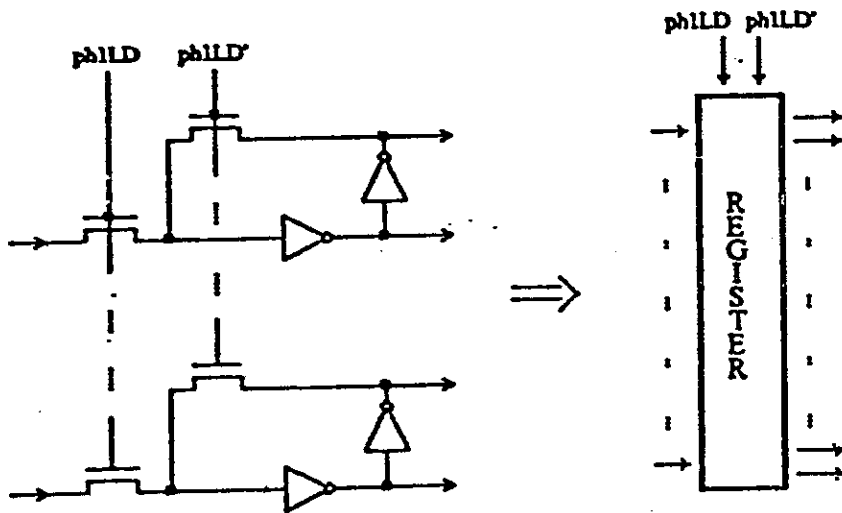


Fig. 9. A Selectively Loadable Dynamic Register

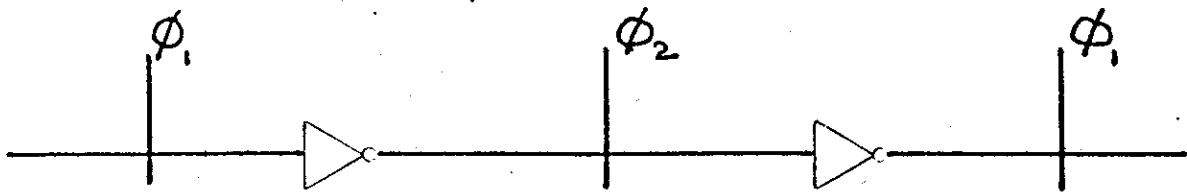


Fig.4c. Shift Register: More Mixed Notation

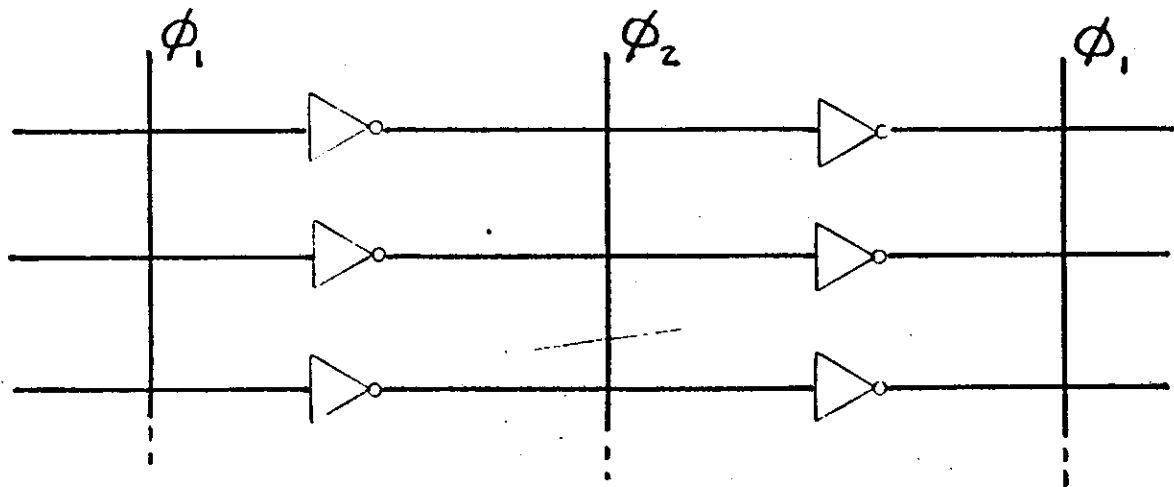


Fig.5a. Array of Shift Registers

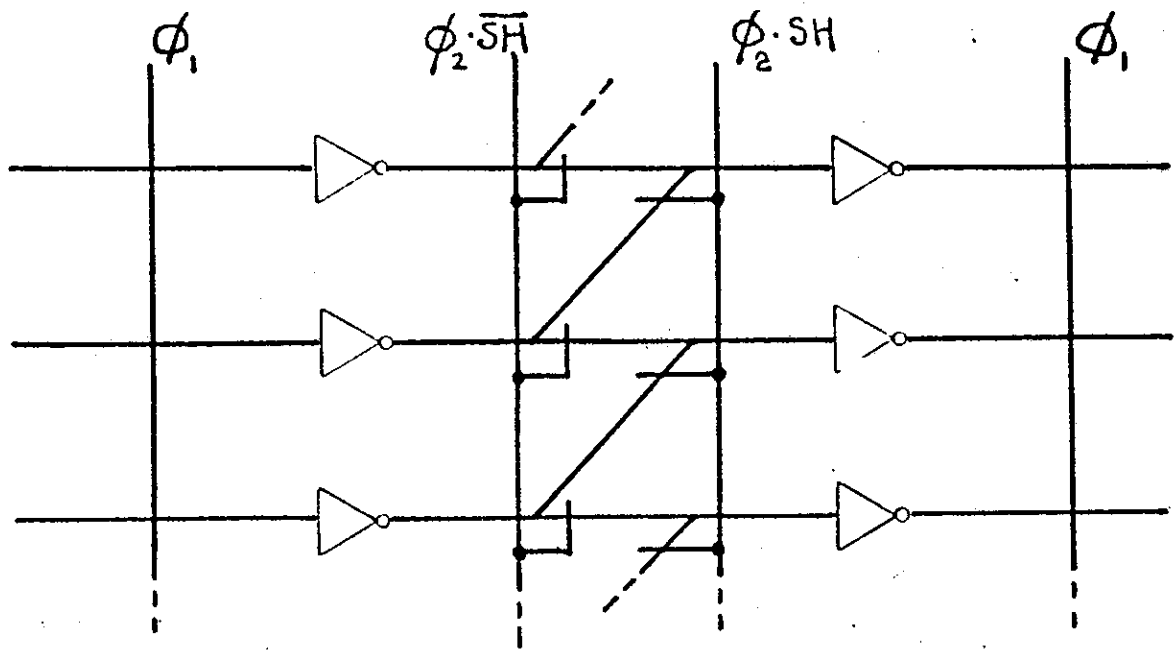
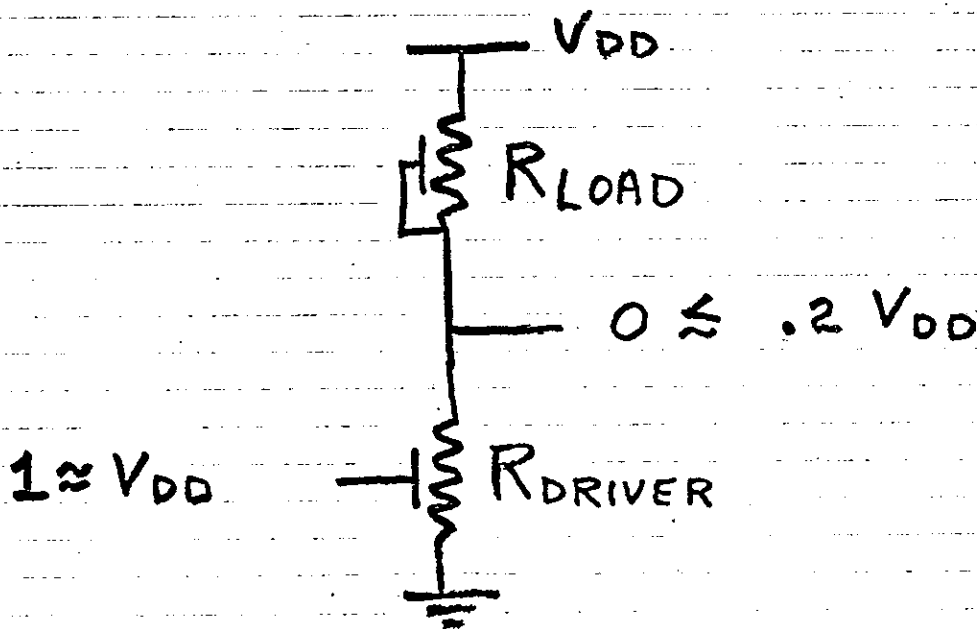


Fig.5b. Shift-Up Register Array

Gates: Ratio Logic

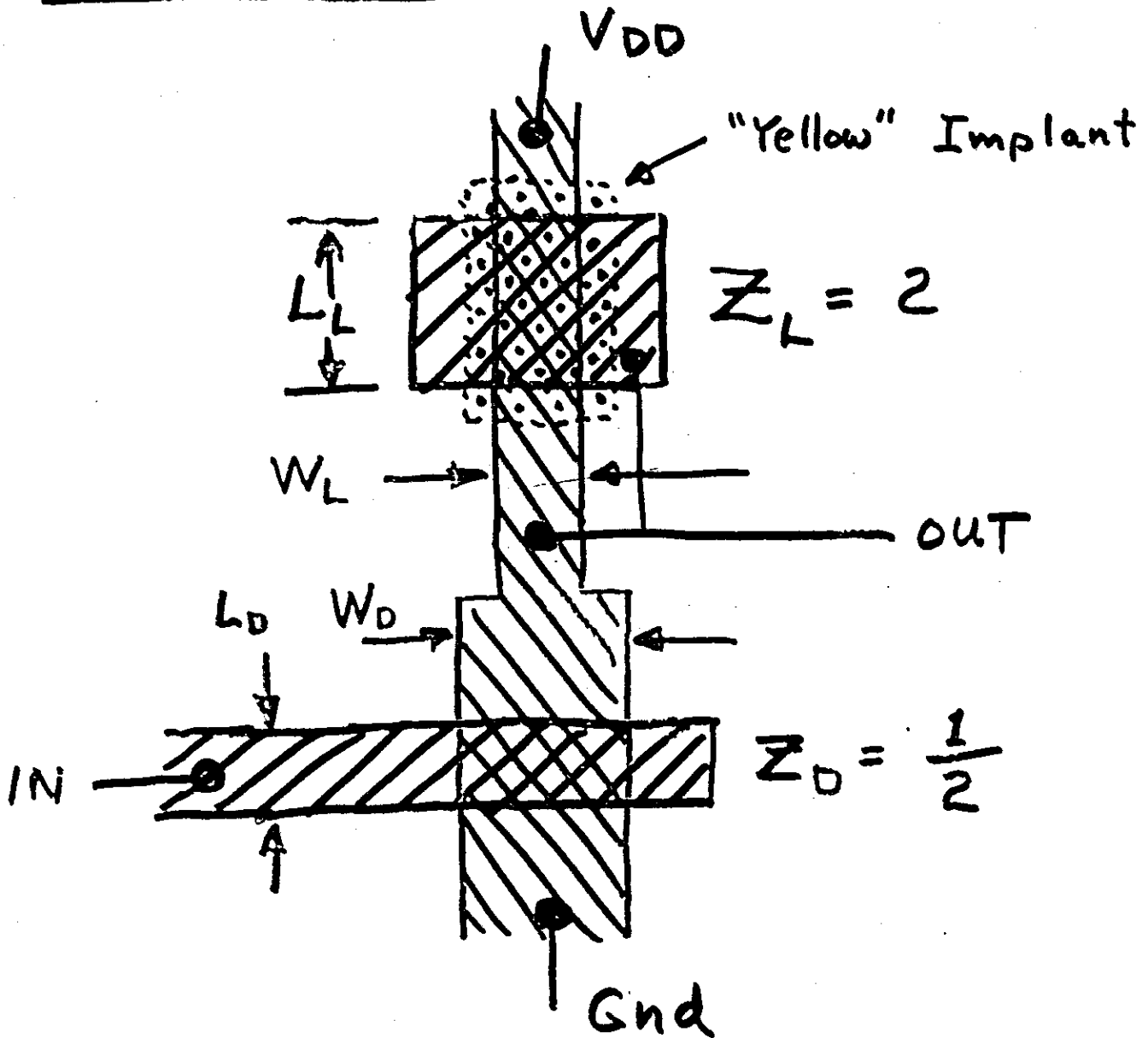
The switches are really resistors when ON, just like the load, which is always ON (a yellow transistor).



$$\Rightarrow R_{LOAD} \approx \boxed{4}^k \cdot R_{DRIVER}$$

Actually, the load current saturates, so $V_{OL} \approx .1 V_{DD}$.
Use 4 as geometric \approx ratio, k .

Ratio Logic: Inverter



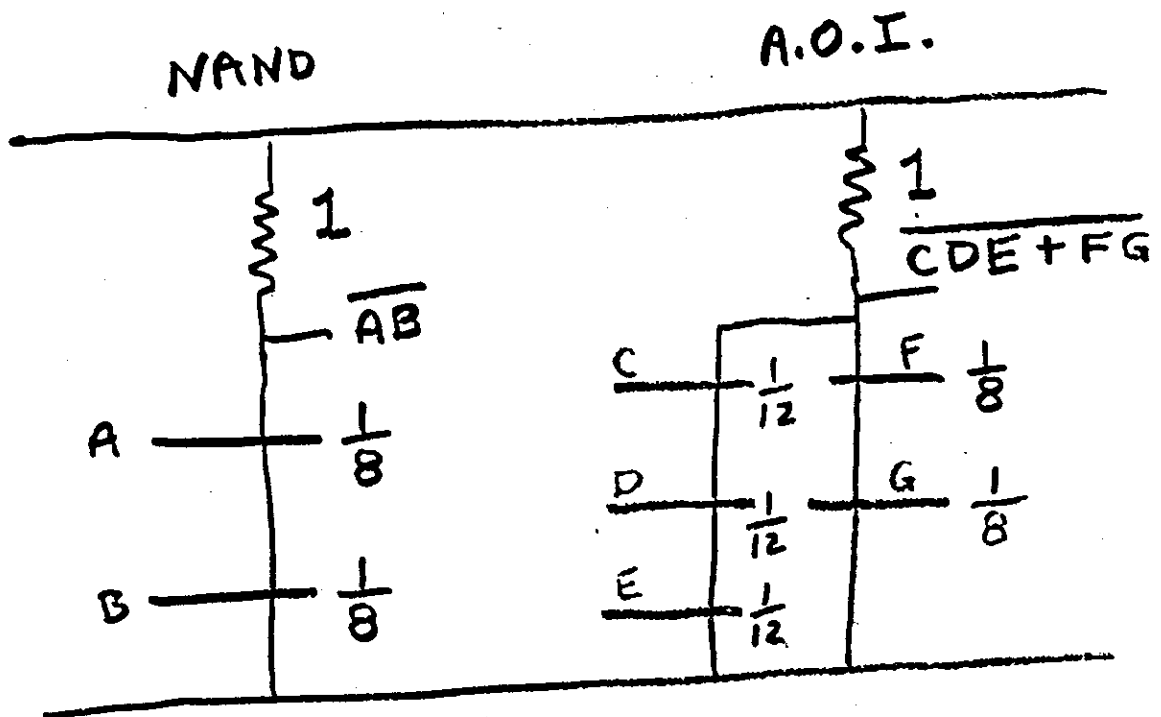
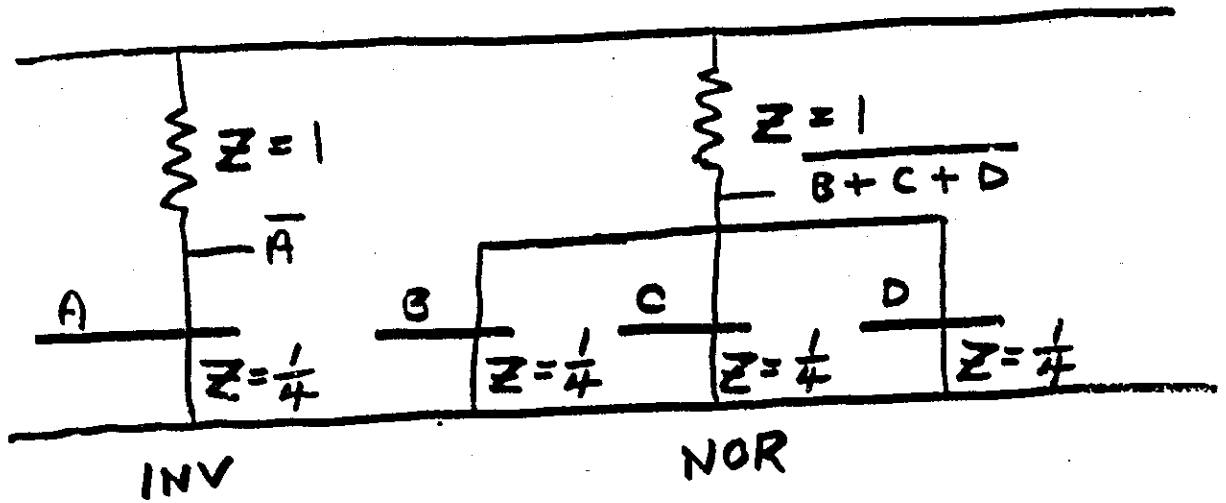
Example:

$$K = 4$$

$$K = \frac{Z_L}{Z_D} = \frac{L_L/W_L}{L_D/W_D}$$

Gates.

$K=4$ Examples:



\Rightarrow NOR is easy.

NAND requires big driver transistors.

(don't exceed 4 inputs to NAND)

Another Rule:

$K=4$ is good if the input signal comes from a level-restoring stage (a gate), or has logic 1 = V_{DD} .

$K=8$ is needed if the input comes through a pass transistor, or has logic 1 = $V_{DD} - V_{TH}$.

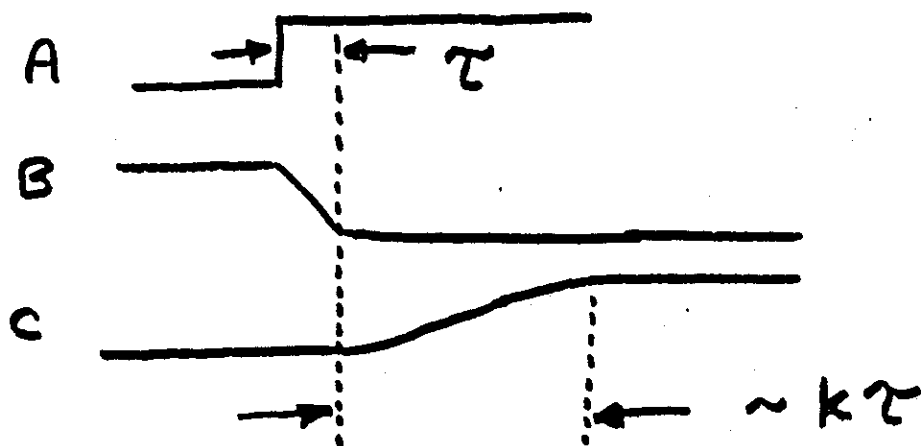
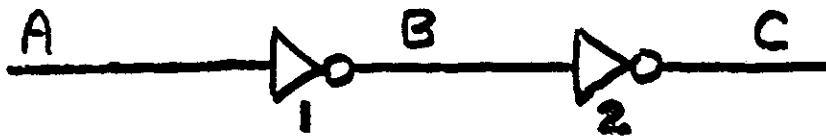
$K=8$ will work everywhere, but may be slower.

see Ch. 1 p. 21.

Timing & Delay:

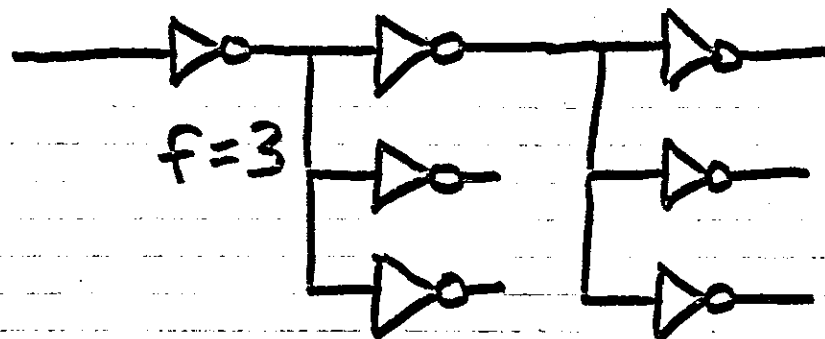
τ as the prime process parameter.

$\tau \equiv$ the time required for a turned-ON driver device⁽¹⁾ to sink enough charge to turn OFF another driver like itself.⁽²⁾



The load pulls up K times slower than the driver pulls down.

Delay with Fanout f :



Delay is really associated with the rise & fall of a node.

\boxed{f} is the load on the node (area relative to the driver).

\boxed{k} is the impedance ratio of the gate driving the node.

$$t_{\text{fall}} = f \cdot \tau$$

$$t_{\text{rise}} = k \cdot f \cdot \tau$$

$$t_{\text{pair}} = (k+1) \cdot f \cdot \tau$$

e.g. $k=8$, $f=4$, $\tau = .3 \text{ nsec}$

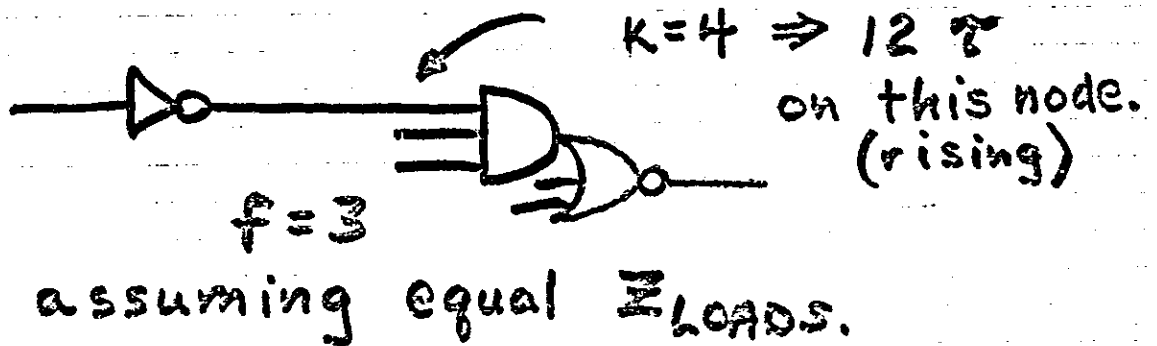
$$t_{\text{pair}} = 10.8 \text{ nsec}$$

but f must count stray C !

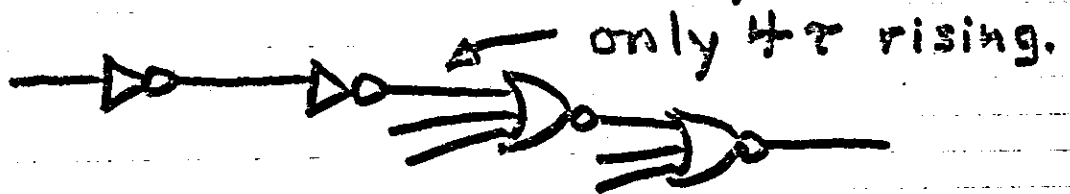
NAND Gates

A.O.I. $\&$ Gates

May be slow due to greater fanout caused by series driver devices.



NOR-NOR logic may be faster:



but more loads \Rightarrow more power.

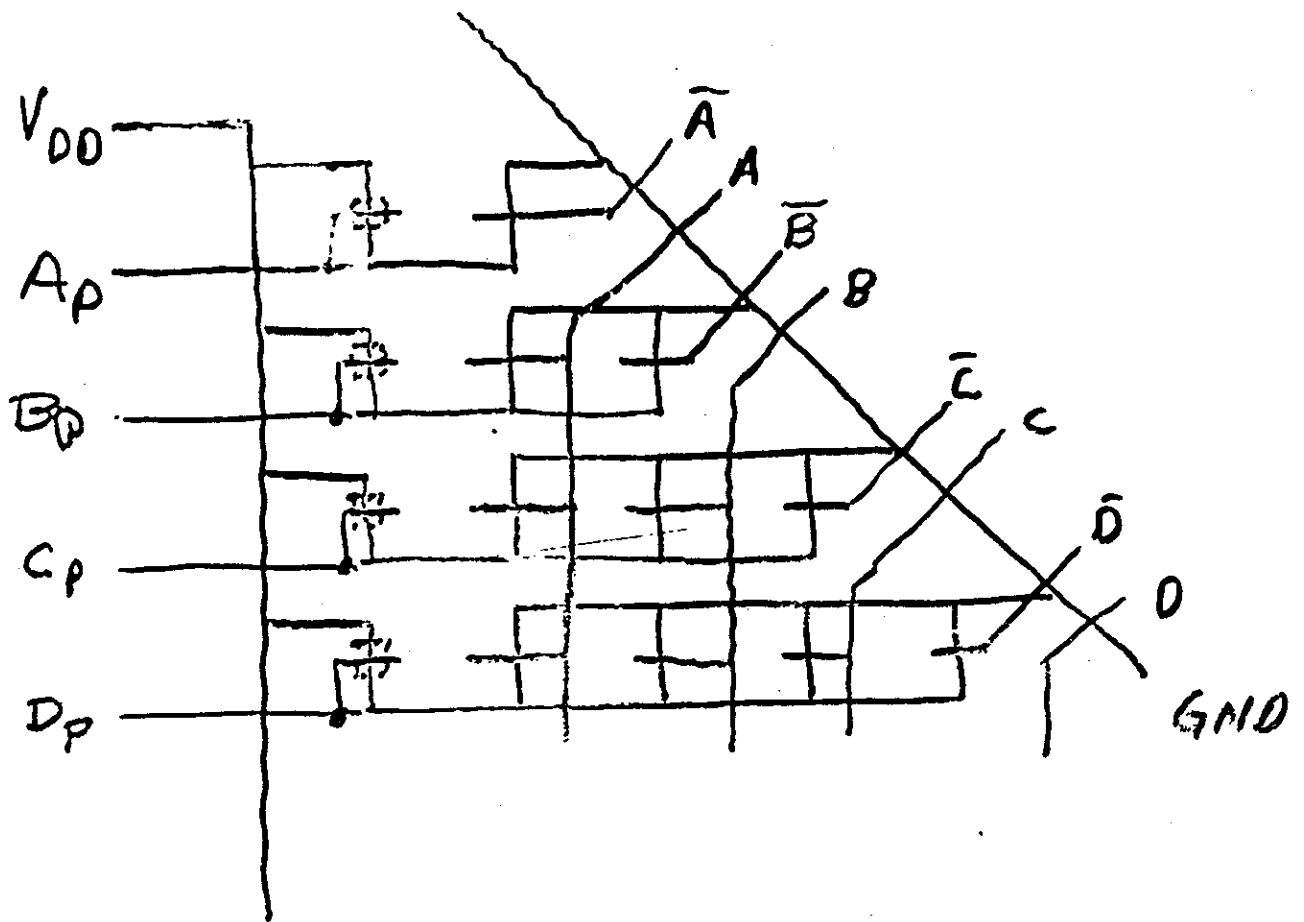
Fanout of all sorts
is a big source of delay!

Other Delays:

1. In wires
 - a. due to strays - fanout.
 - b. due to distributed RC.

2. In pass transistors:
ultimately proportional to N^2
(for large N)
for N in series.

3. In driving off-chip:
through multi-stage
medium-fanout buffers
(or super buffers)
(also for large loads
on-chip)

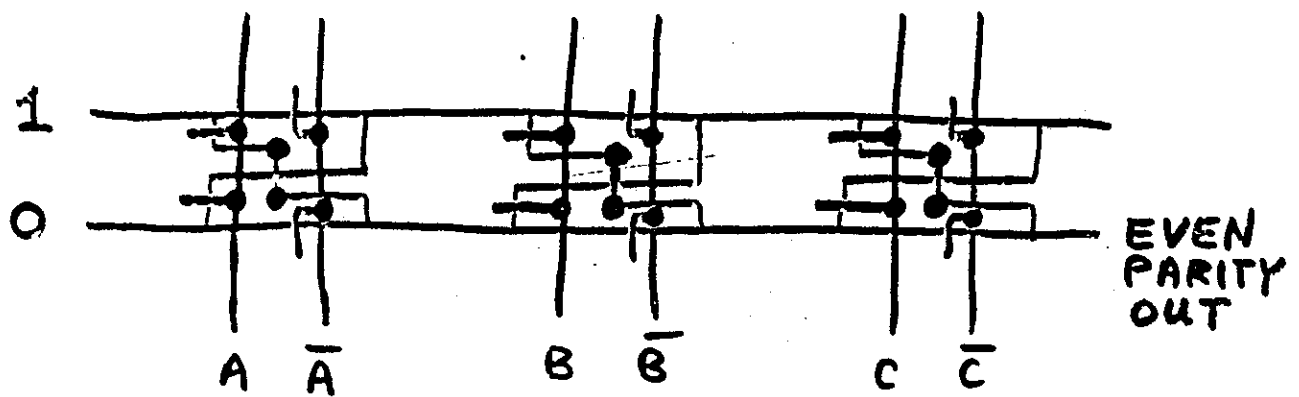


Problem #1

A NOR-gate solution in N^2 area.

- Ted Kachler

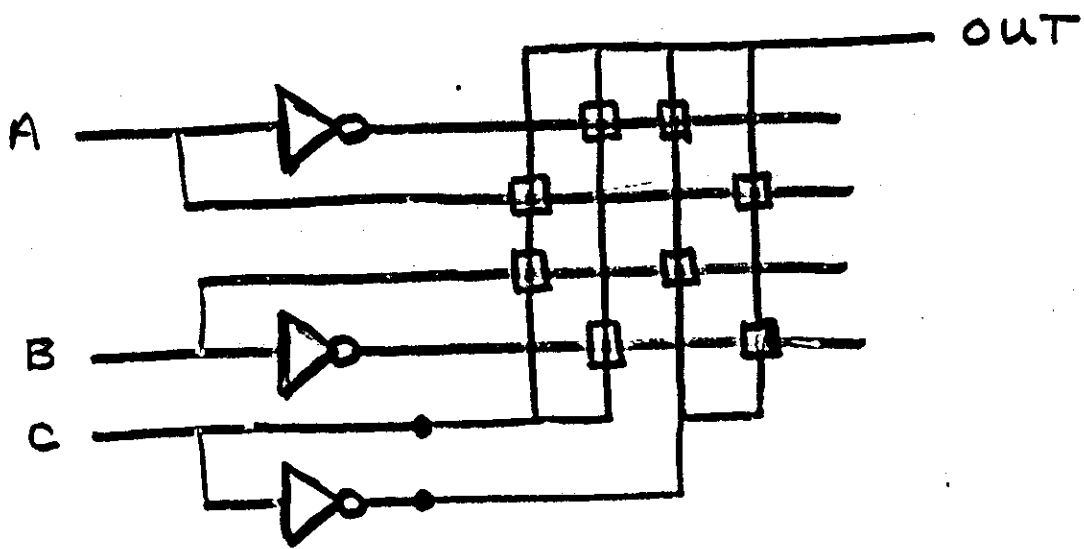
N -input linear XOR array:
"conditional swap"



(a popular approach)

3-input XOR (not expandable)

based on function block:



A	B	OUT
0	0	0
0	1	1
1	0	1
1	1	0

Today's topic:
More Digital Structures
for LSI Systems.

Arrays.

Topology of Power, Control,
and Data Routing.

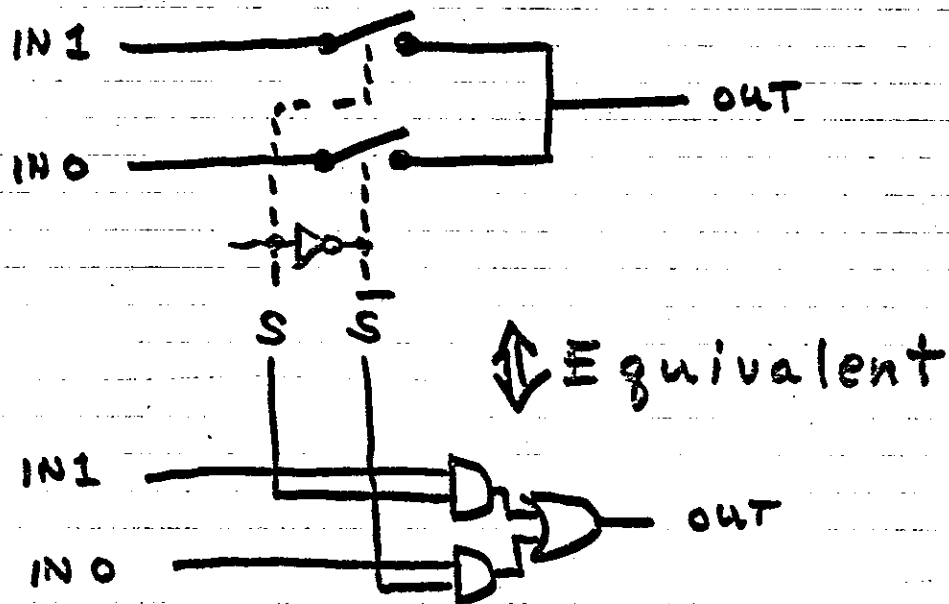
Regular Structures for
Random Logic.

Special Structures for
Highly Functional Arrays.

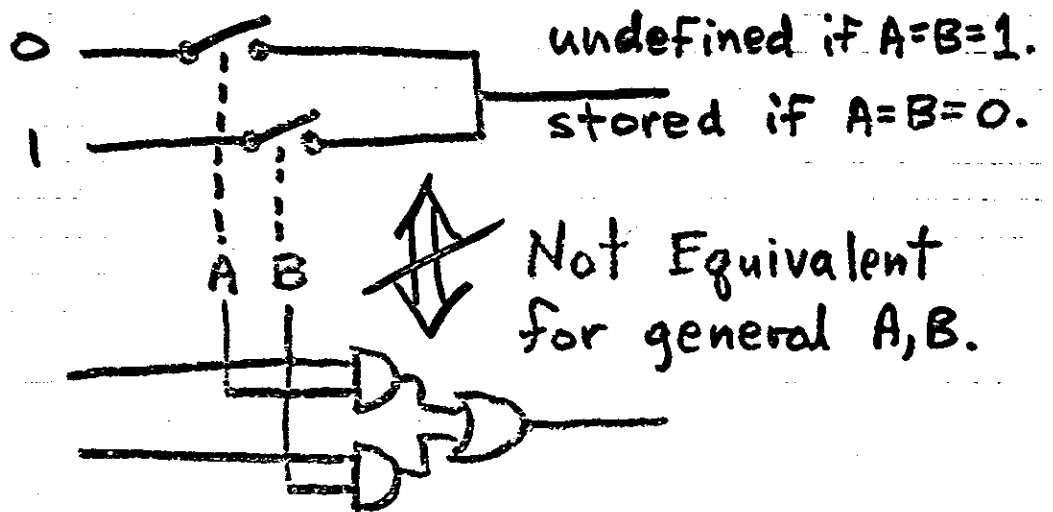
More Structured Design Methodology.

leftovers

Switch Logic Pitfalls

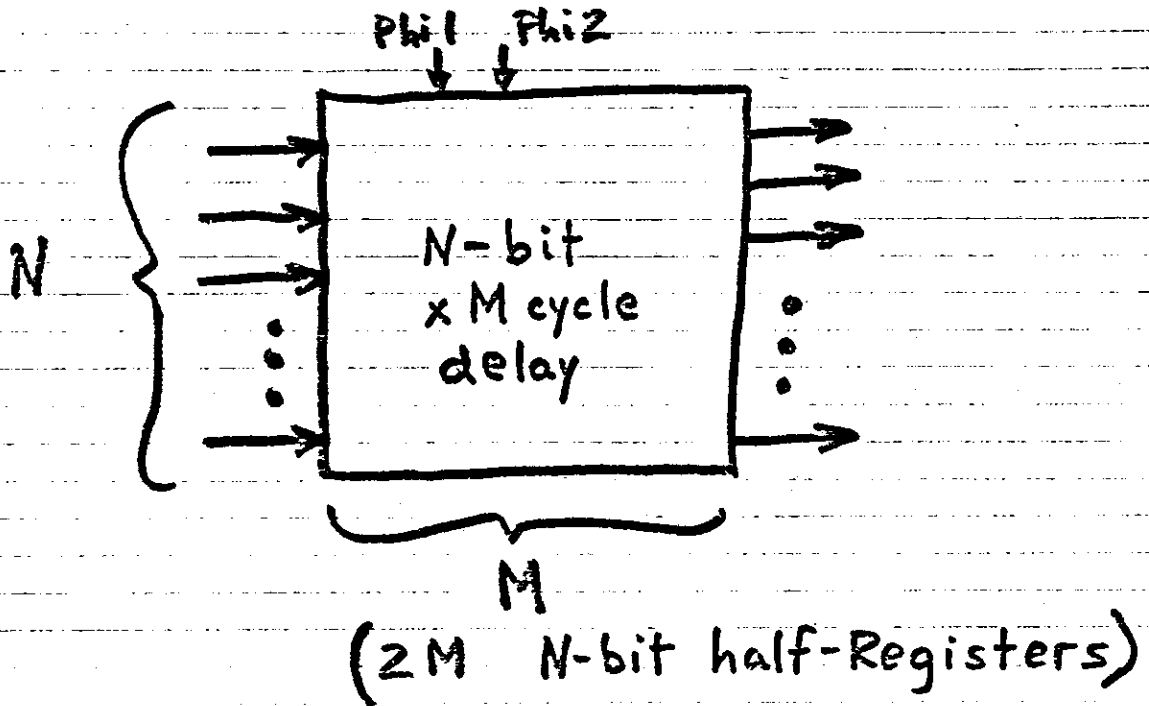


Notice the equivalence only works because the control inputs are complementary.



Arrays - basic structures

Example: Shift Register Array



Array Advantages:

Duplication of simple cells.

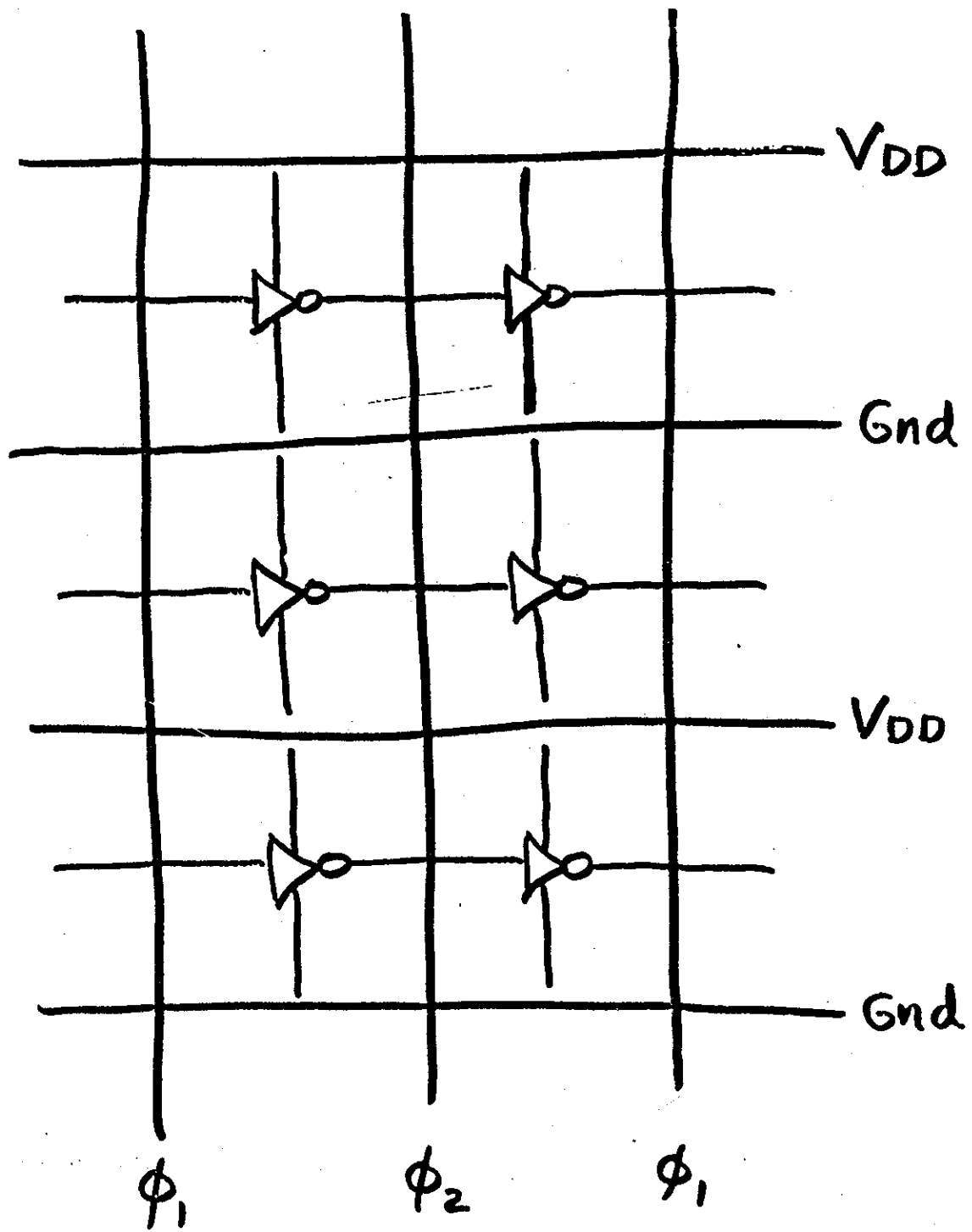
Requires connections only at edges.

Easy to express design intent.

i.e. Regular.

much function with
little complexity.

Array Topology Example:
Horizontal Power ————
Horizontal Data ————
Vertical Control (clock) |



SRCELL - see Ch. 4, Figs. 8-11.

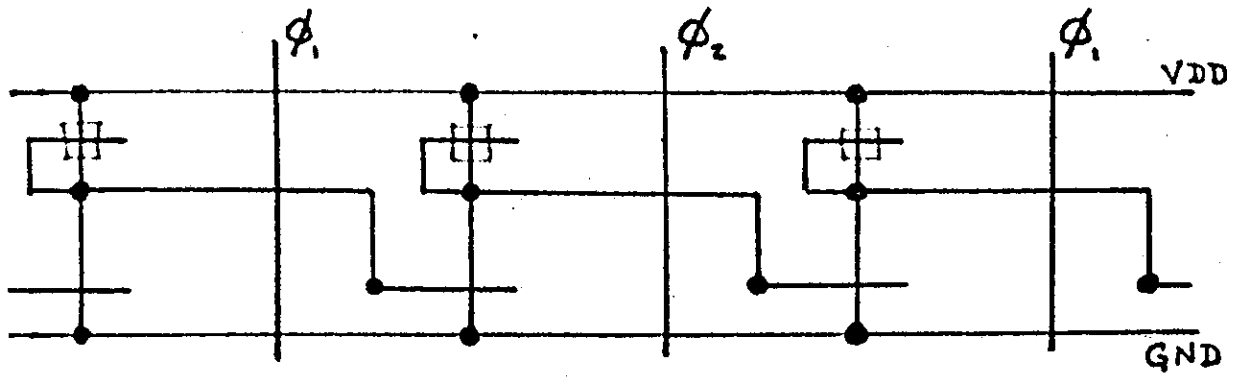


Fig. 8a. Stick Diagram of One Row of Shift Register Array

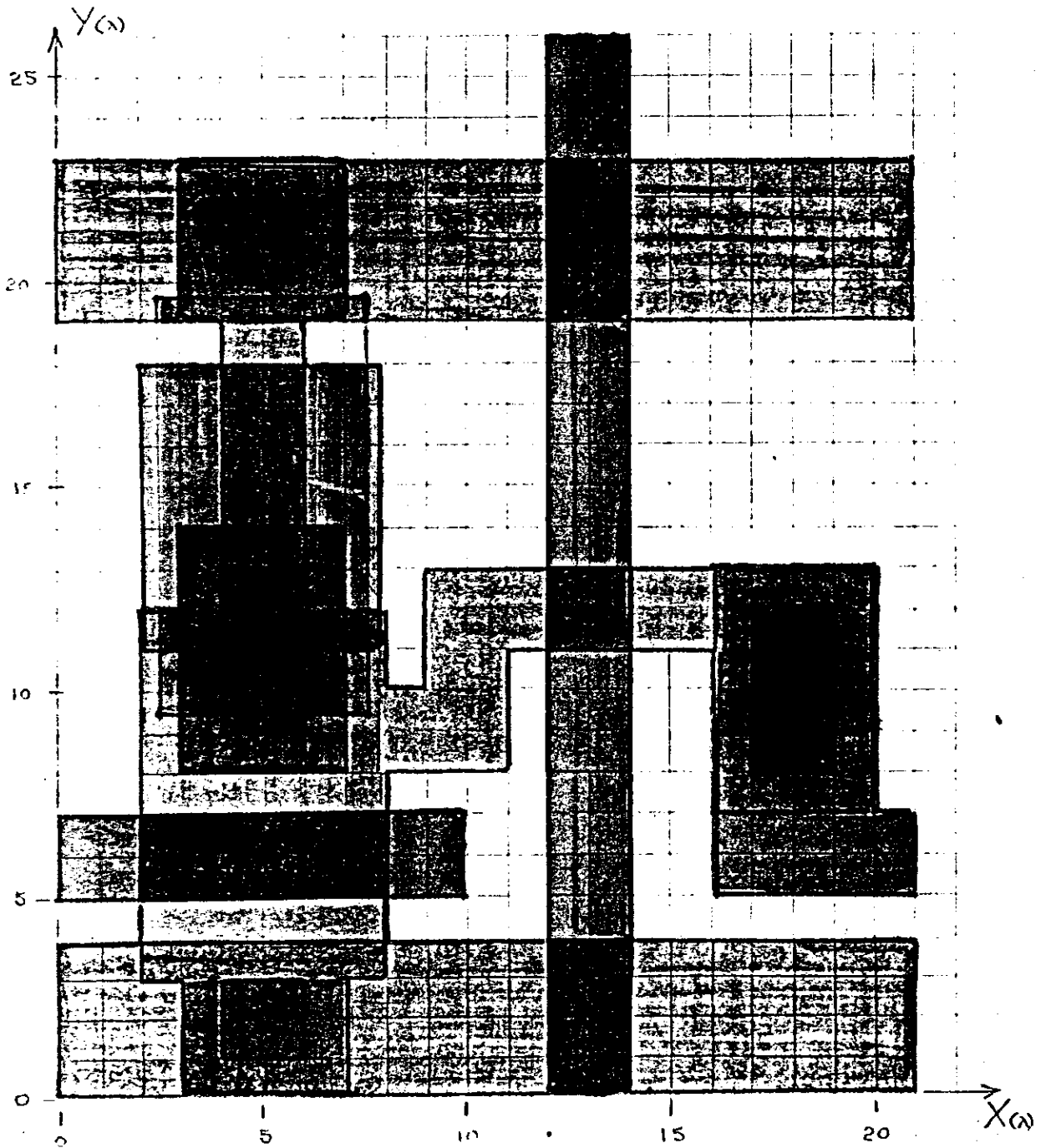
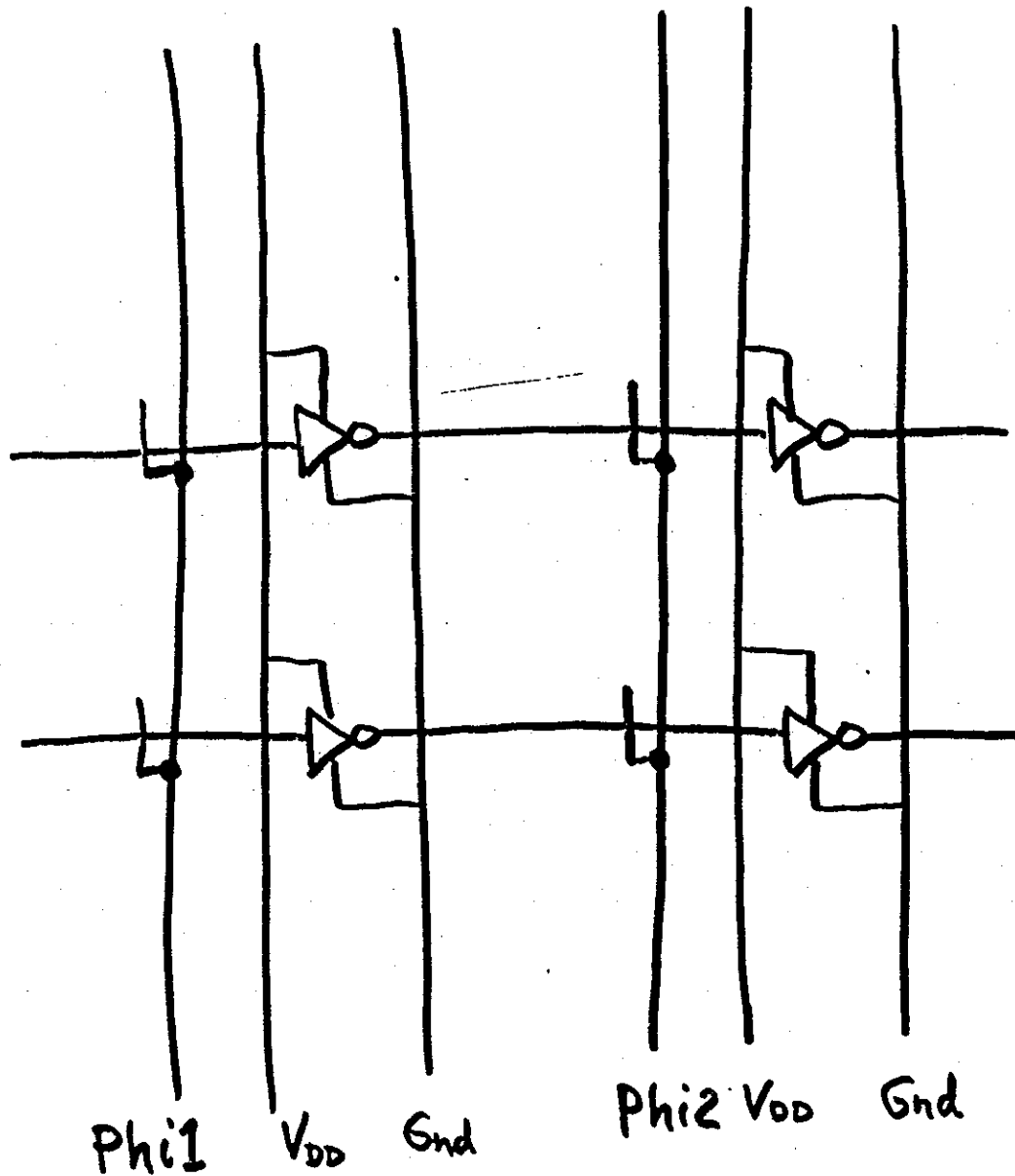


Fig. 8b. Hand Sketch of Layout of One Shift Register Cell

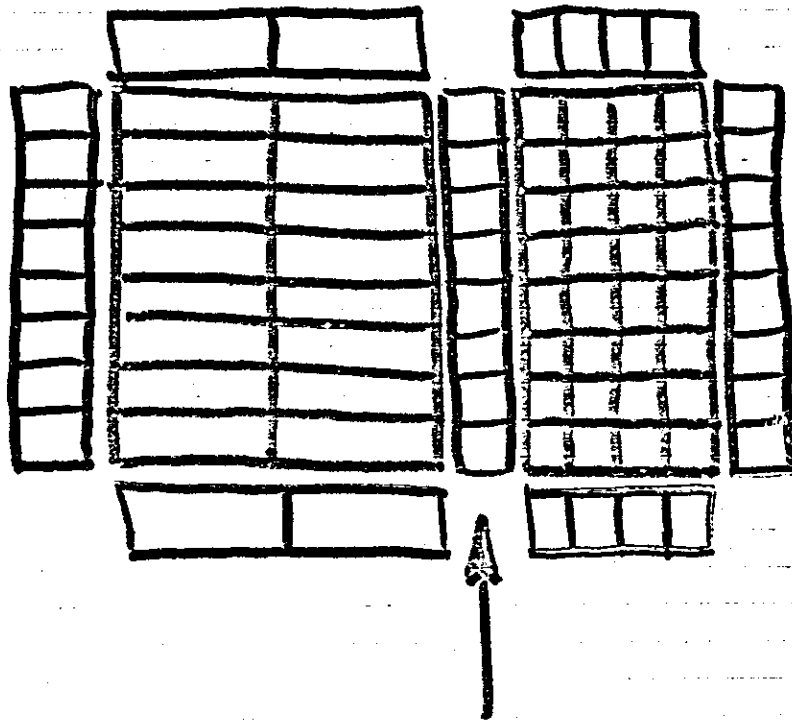
Another Good Topology:
Vertical Power and Control
(all Metal \Rightarrow fast)
Horizontal Data
Control Tabs \rightarrow



Note: Data \perp Control
is usually good.
Data usually green.

Linear Arrays:

Rule of thumb: For every "rectangular" array there are several linear arrays.



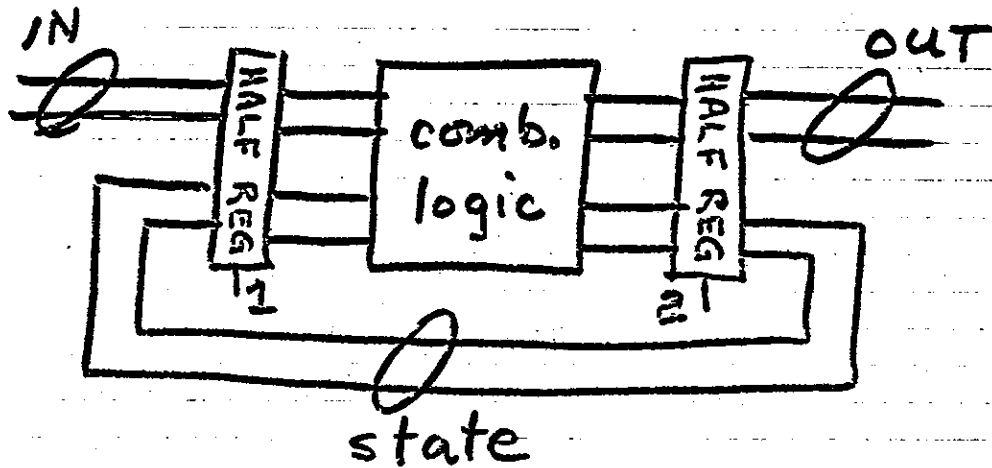
Notice: Pitch should match so a linear array can be used to couple the rectangular arrays.

OM2 Data Path is an example.

PLA is an example.

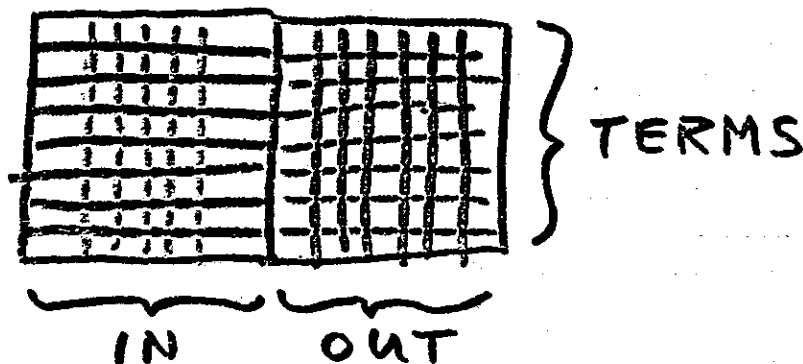
Finite-State Machines

for controllers and many other functions, including serial arithmetic.



Combinatorial logic is arbitrary, and may be built "randomly," but a single logic structure, the PLA, is often better.

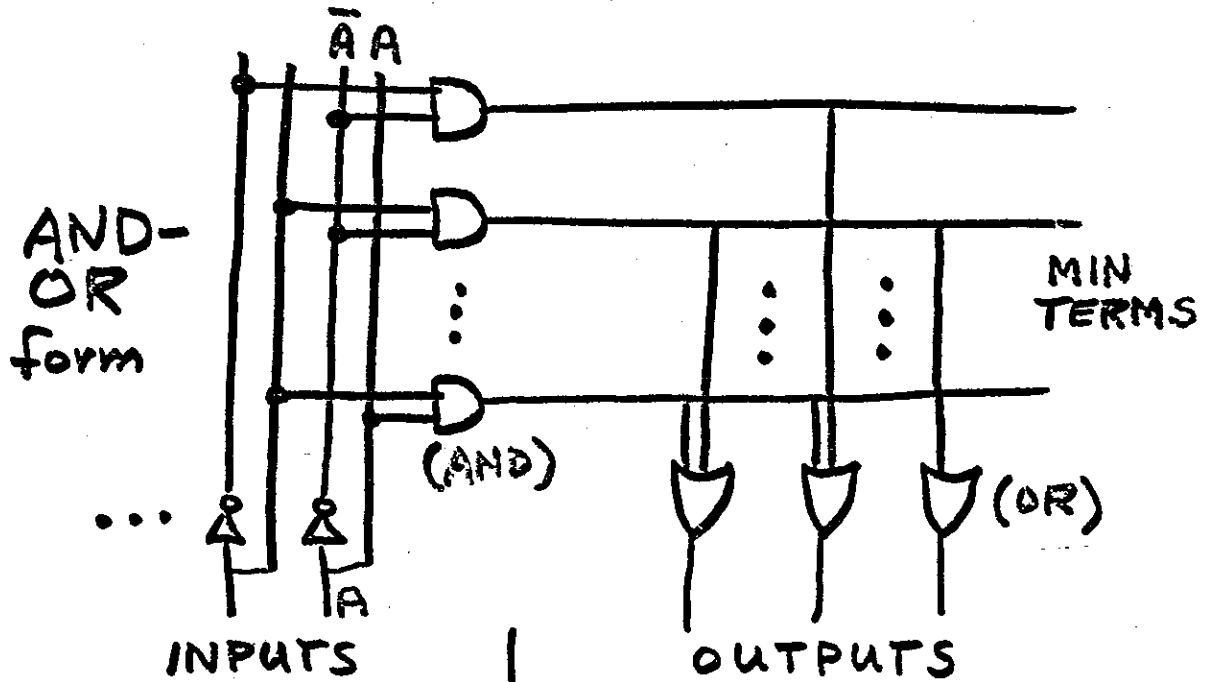
A PLA is measured by inputs, outputs, & product terms:



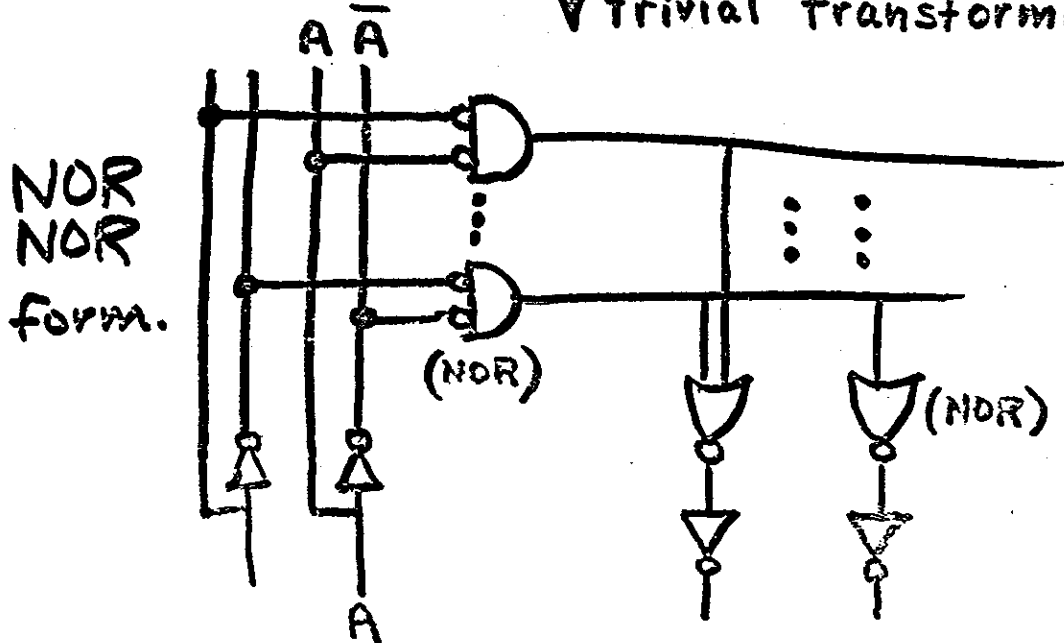
The PLA

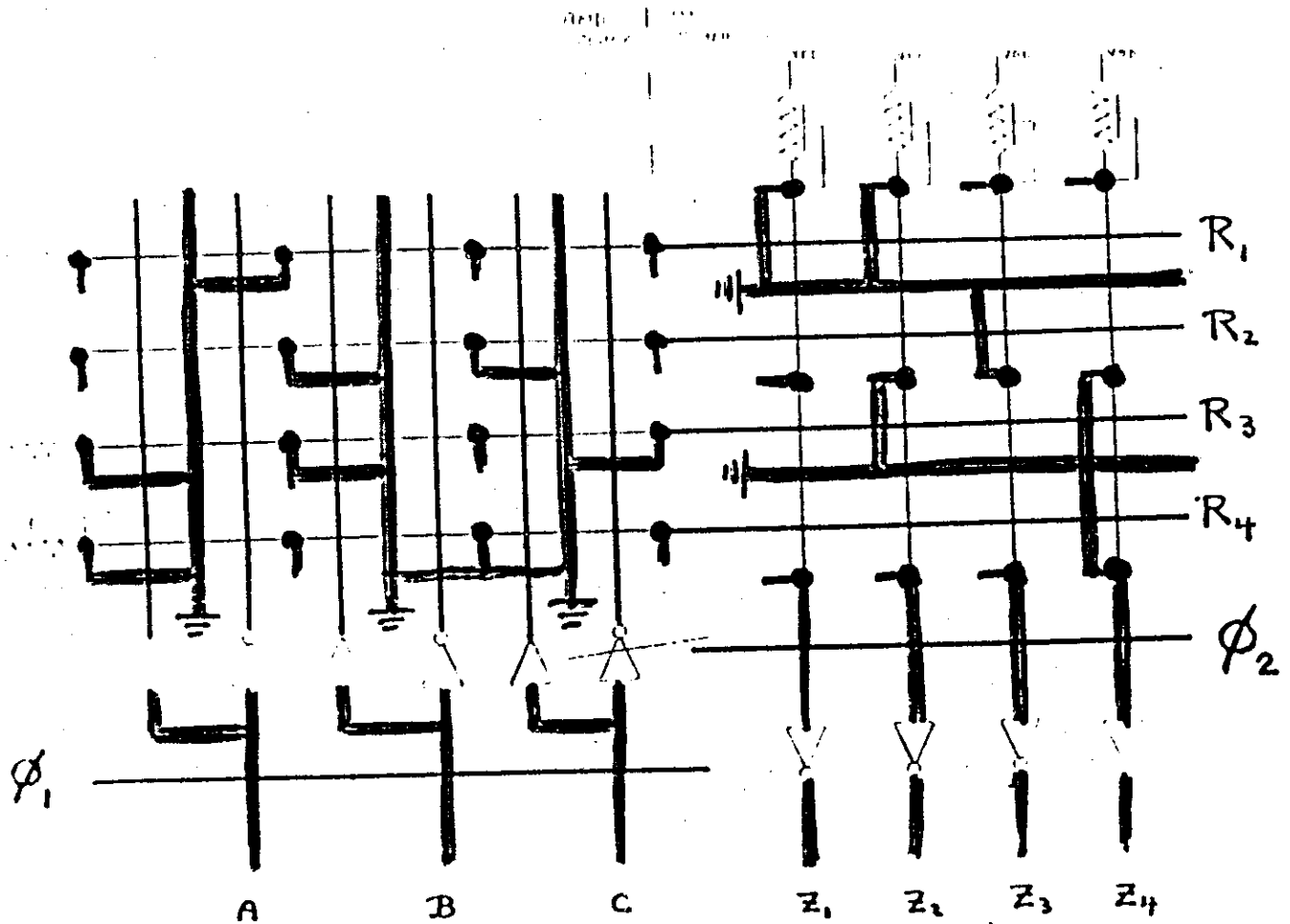
(Programmed Logic Array)

NOR-NOR logic



↓ trivial transformation





Green lines enhanced to show programming paths to ground.

Fig. 1.3c. Stick Diagram of PLA Example.

Can you find the gates?

Product Terms:

$$R_1 = (A')' = A$$

$$R_2 = (B+C)' = B'C'$$

$$R_3 = (A+B+C)' = A'B'C'$$

$$R_4 = (A+B'+C)' = A'BC'$$

Outputs:

$$Z_1 = A$$

$$Z_2 = A + A'B'C'$$

$$Z_3 = B'C'$$

$$Z_4 = A'BC' + A'BC'$$

Stored during ϕ_1 in INREG				Stored during ϕ_2 in OUTREG					Product terms:	
Inputs:		Present State:	Next State:	Outputs:						
C	TL	TS	Y_{p0}, Y_{p1}	Y_{n0}, Y_{n1}	ST	HL ₀	HL ₁	FL ₀	FL ₁	
0	X	X	0, 0 (HG)	0, 0 (HG)	0	0	0	1	0	R1
X	0	X	0, 0 (HG)	0, 0 (HG)	0	0	0	1	0	R2
1	1	X	0, 0 (HG)	0, 1 (HY)	1	0	0	1	0	R3
X	X	0	0, 1 (HY)	0, 1 (HY)	0	0	1	1	0	R4
X	X	1	0, 1 (HY)	1, 1 (FG)	1	0	1	1	0	R5
1	0	X	1, 1 (FG)	1, 1 (FG)	0	1	0	0	0	R6
0	X	X	1, 1 (FG)	1, 0 (FY)	1	1	0	0	0	R7
X	1	X	1, 1 (FG)	1, 0 (FY)	1	1	0	0	0	R8
X	X	0	1, 0 (FY)	1, 0 (FY)	0	1	0	0	1	R9
X	X	1	1, 0 (FY)	0, 0 (HG)	1	1	0	0	1	R10

Fig.15e. Encoded State Transition Table for the Light Controller

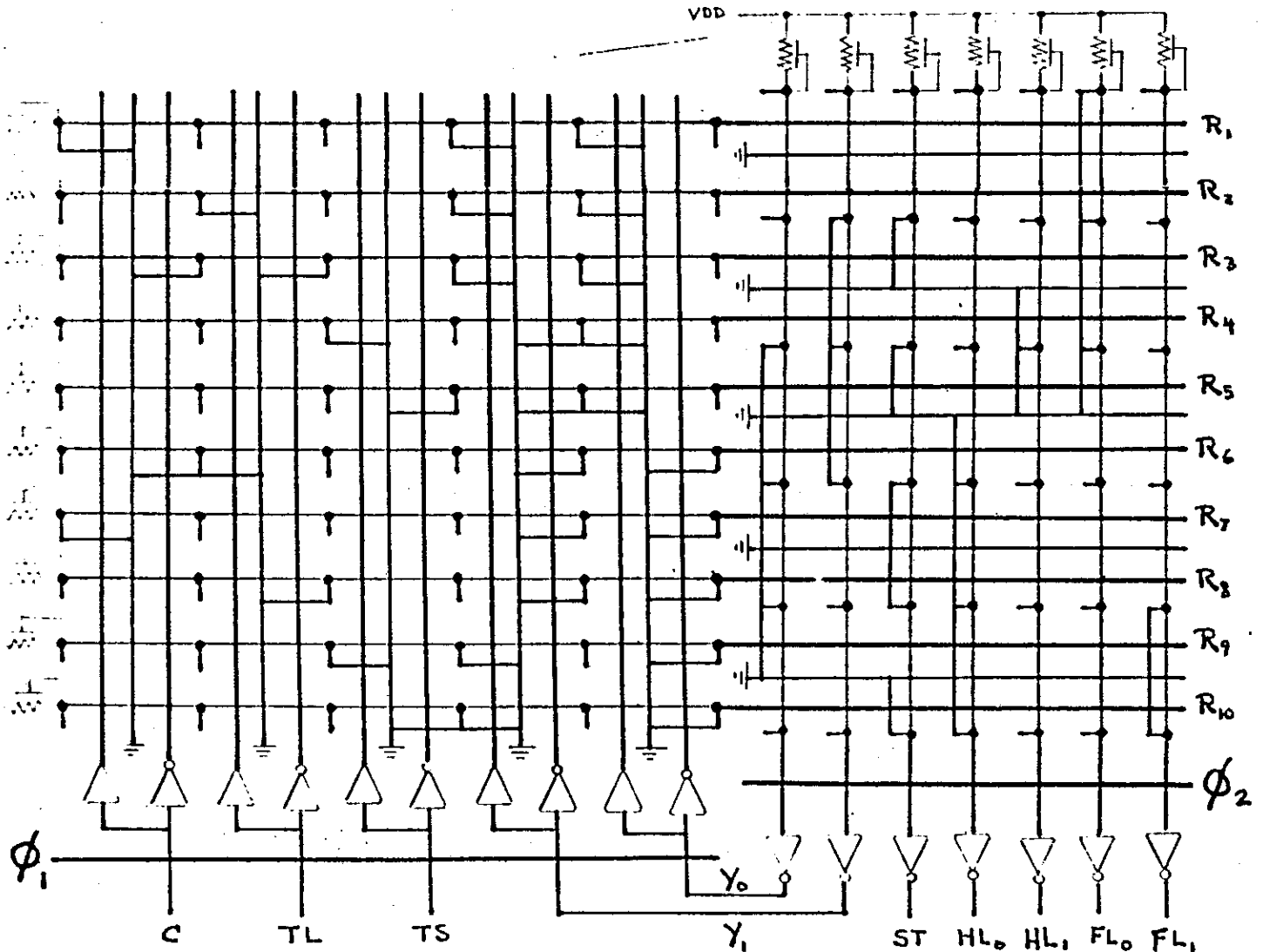
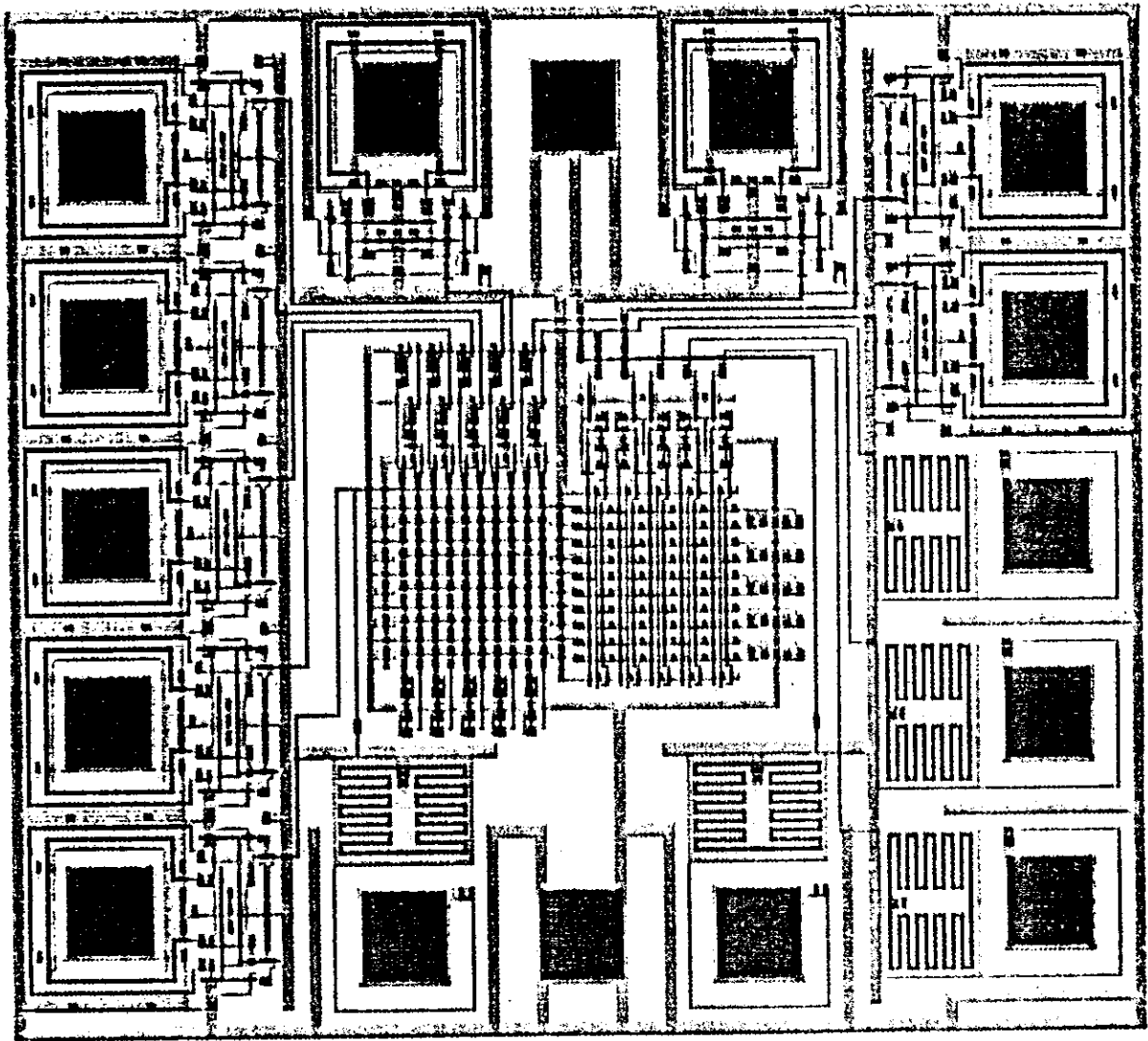


Fig.15f. PLA Sequential Circuit Implementing the Light Controller



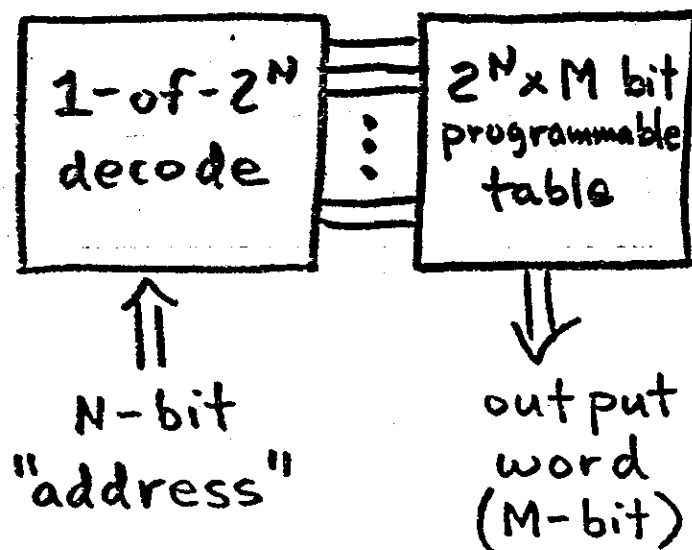
How big can a PLA get?

Inputs & Outputs are
easy to count.

Product terms:
worst case 2^N
For N inputs.

In this case the "AND"-plane
is a 1-of- 2^N decoder.

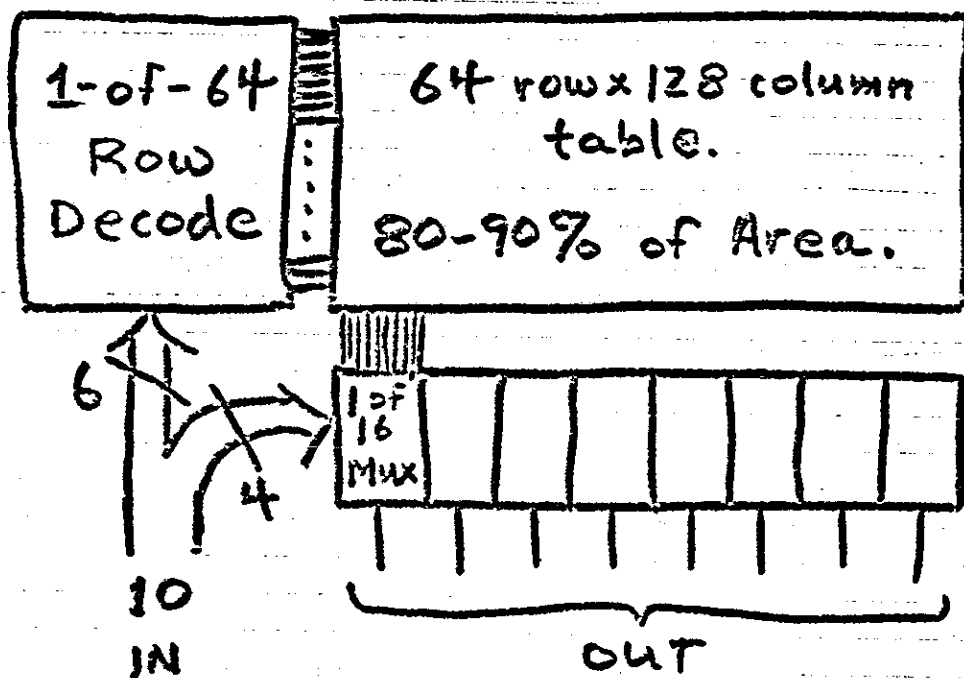
This "Full" PLA is exactly
a ROM - Read Only Memory.



But the address decoder
consumes half the area!

More Elaborate ROM:
Row & Column Decoding.

Example: $1K \times 8$
($N=10, M=8$)



Each Row reads out 16 words,
bit interleaved,
and the column multiplexor
selects one as output.

Summary

Combinatorial Logic & FSM Techniques

small - Gates, Function Blocks,
small PLAs.

medium - PLA's, small ROMs.

large -

unstructured "Random" logic:
big ROM.

structured logic:

a.) decompose into medium
size pieces.

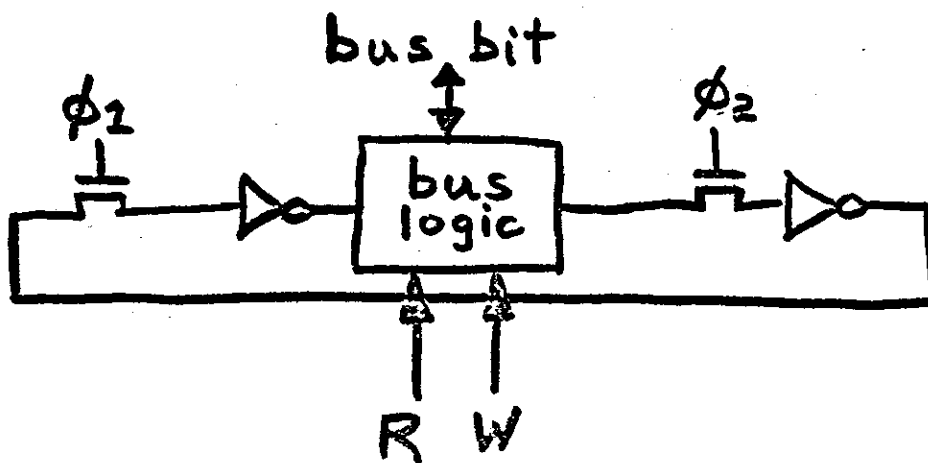
b.) design a special structure
to match the logic.

e.g. highly functional array
of special cells.

Example: LCC (Xerox micro)
combines these.

Memories

Small RAM — Register File
semi-static cell refreshes
on every clock cycle.



Ram Cell as a small FSM.

Use pass transistor logic
for bus read/write connection.

OM2 has a 2-port version.

(two busses with separate
read and write controls)

Area dominated by lines.

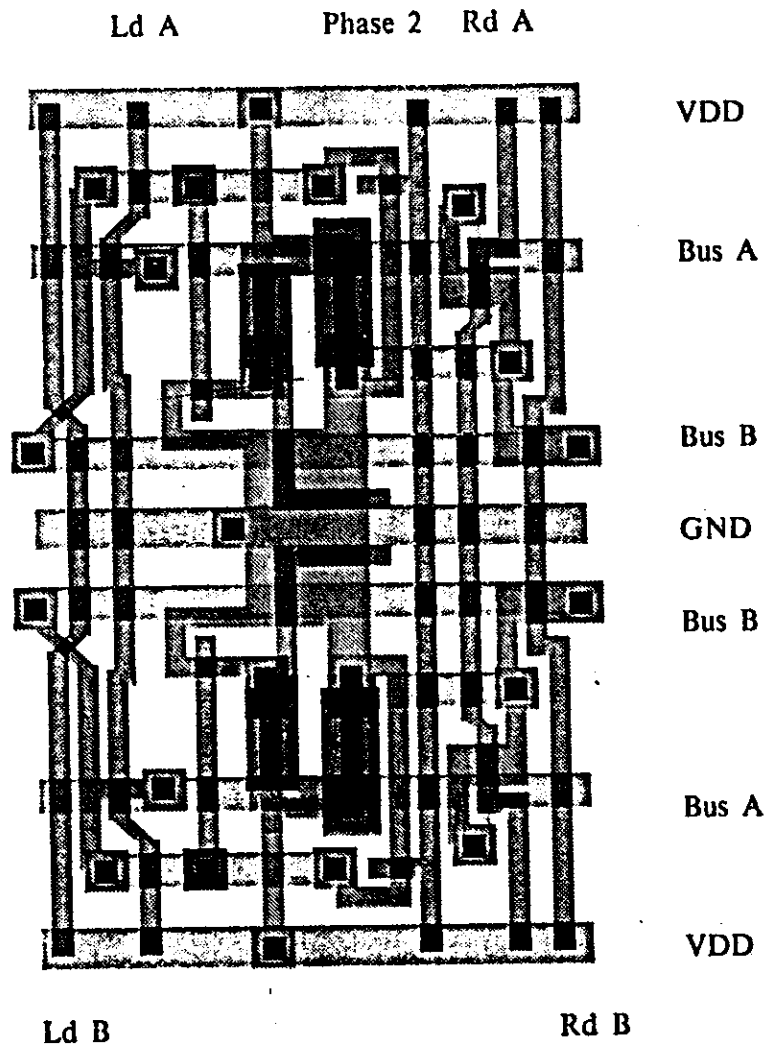
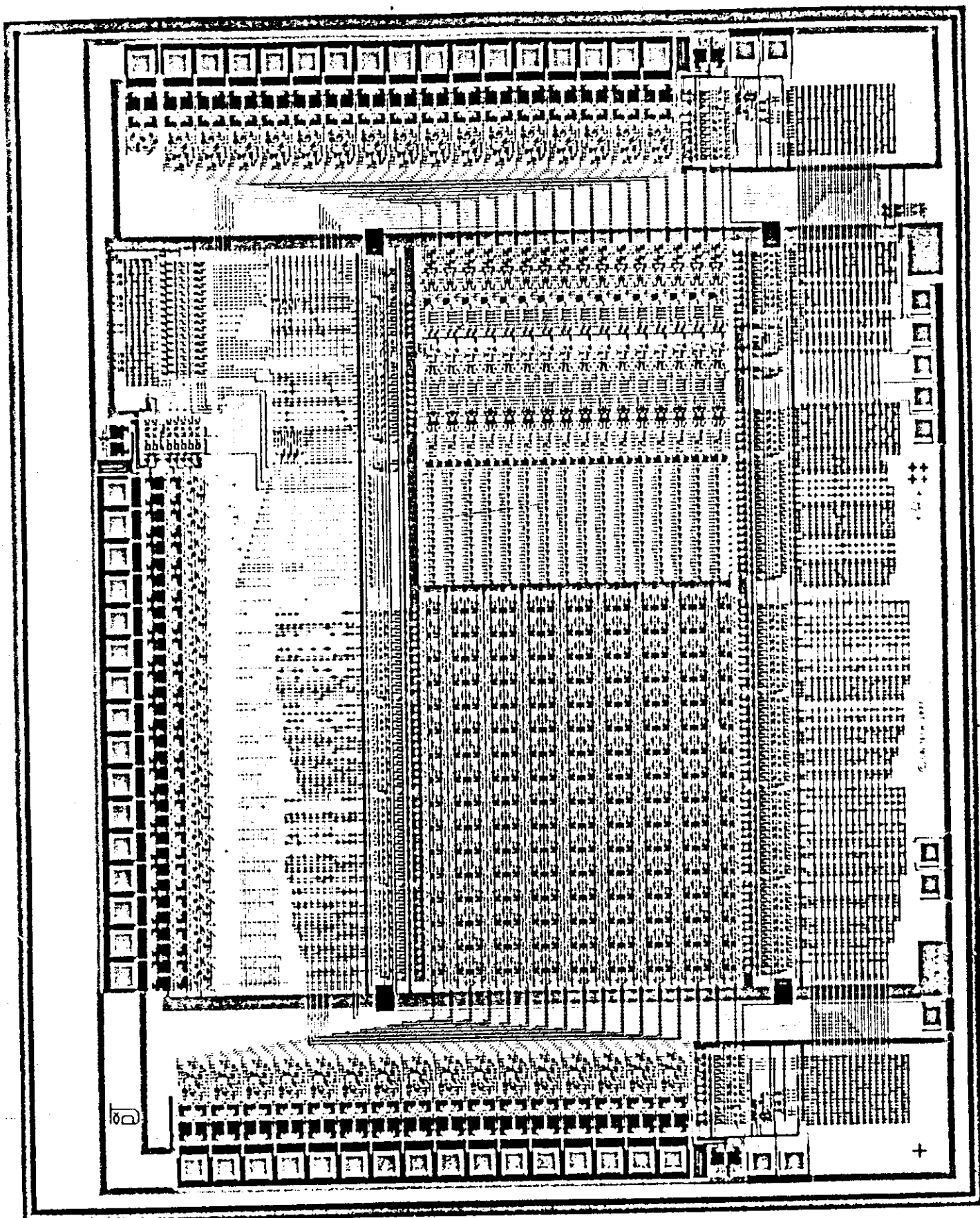


Fig. 22a. Layout of Two Dual-Port Register Cells



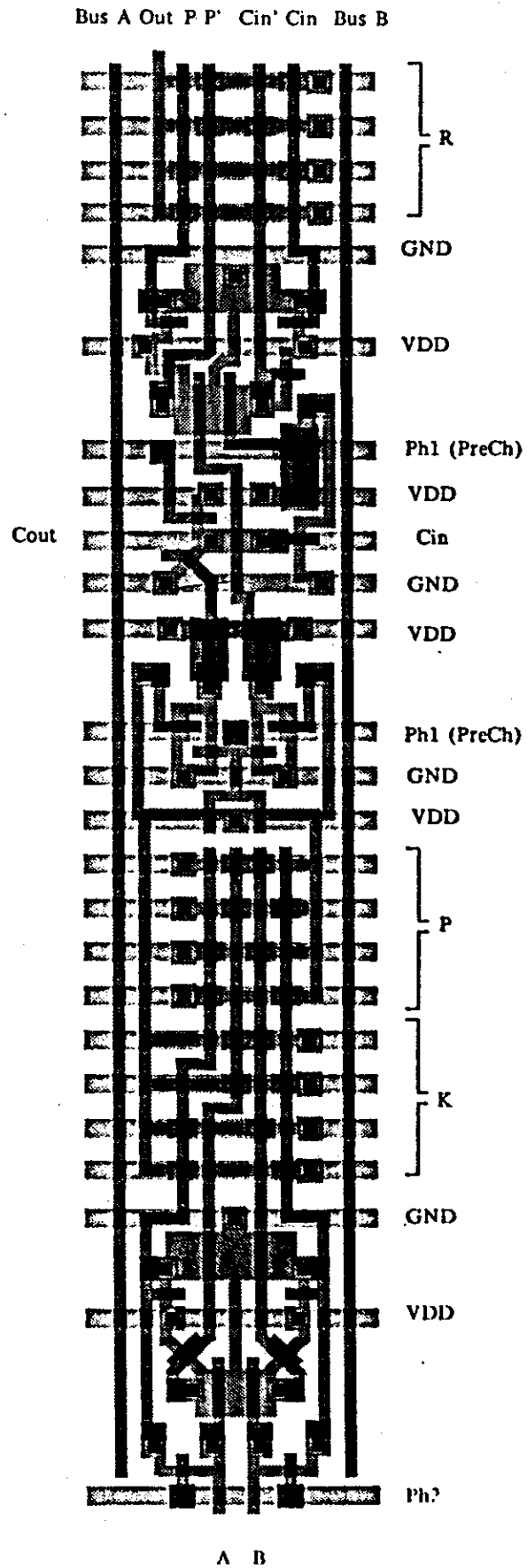


Figure 6a. Layout of ALU Bit Slice and Input Registers

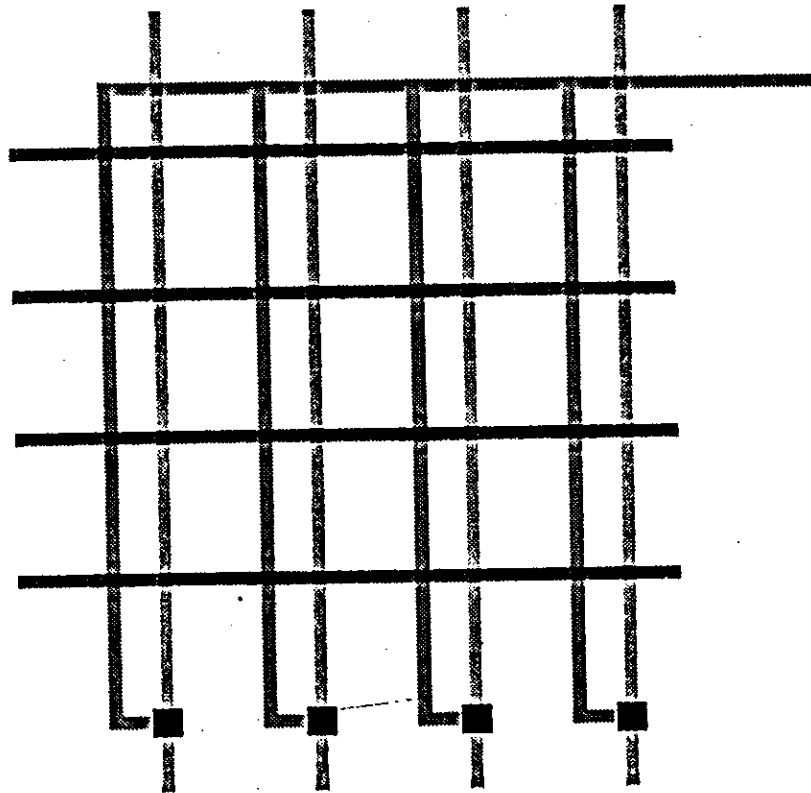


Fig 4a. Stick Diagram of the Function Block

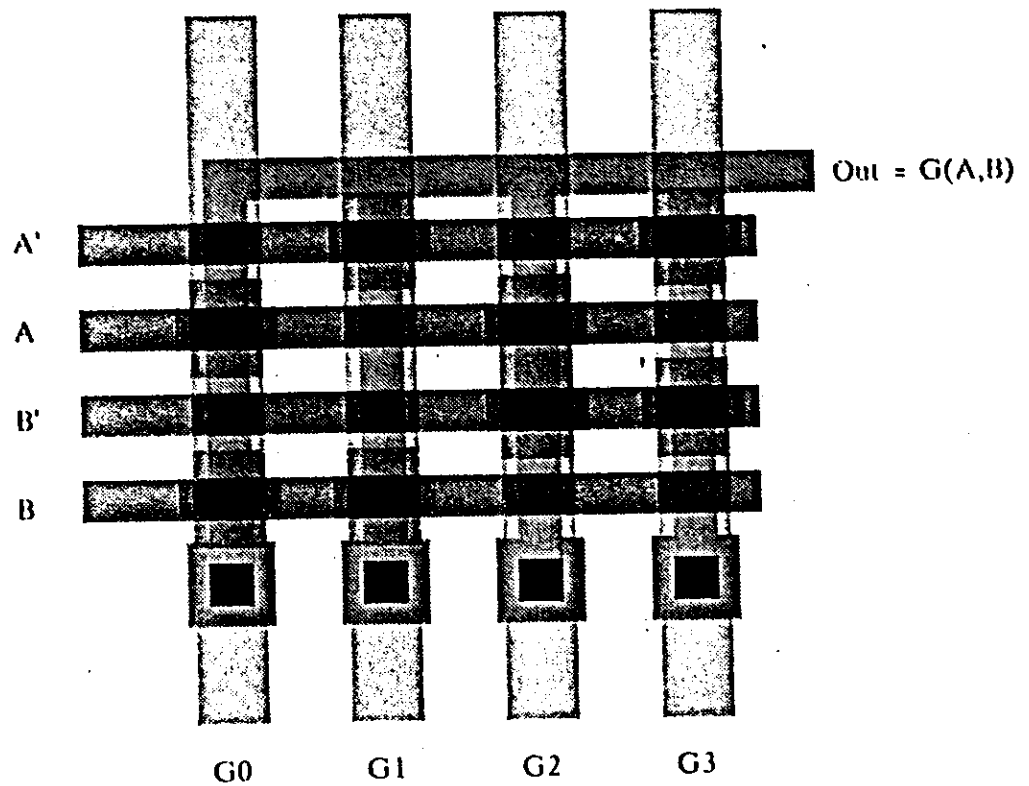
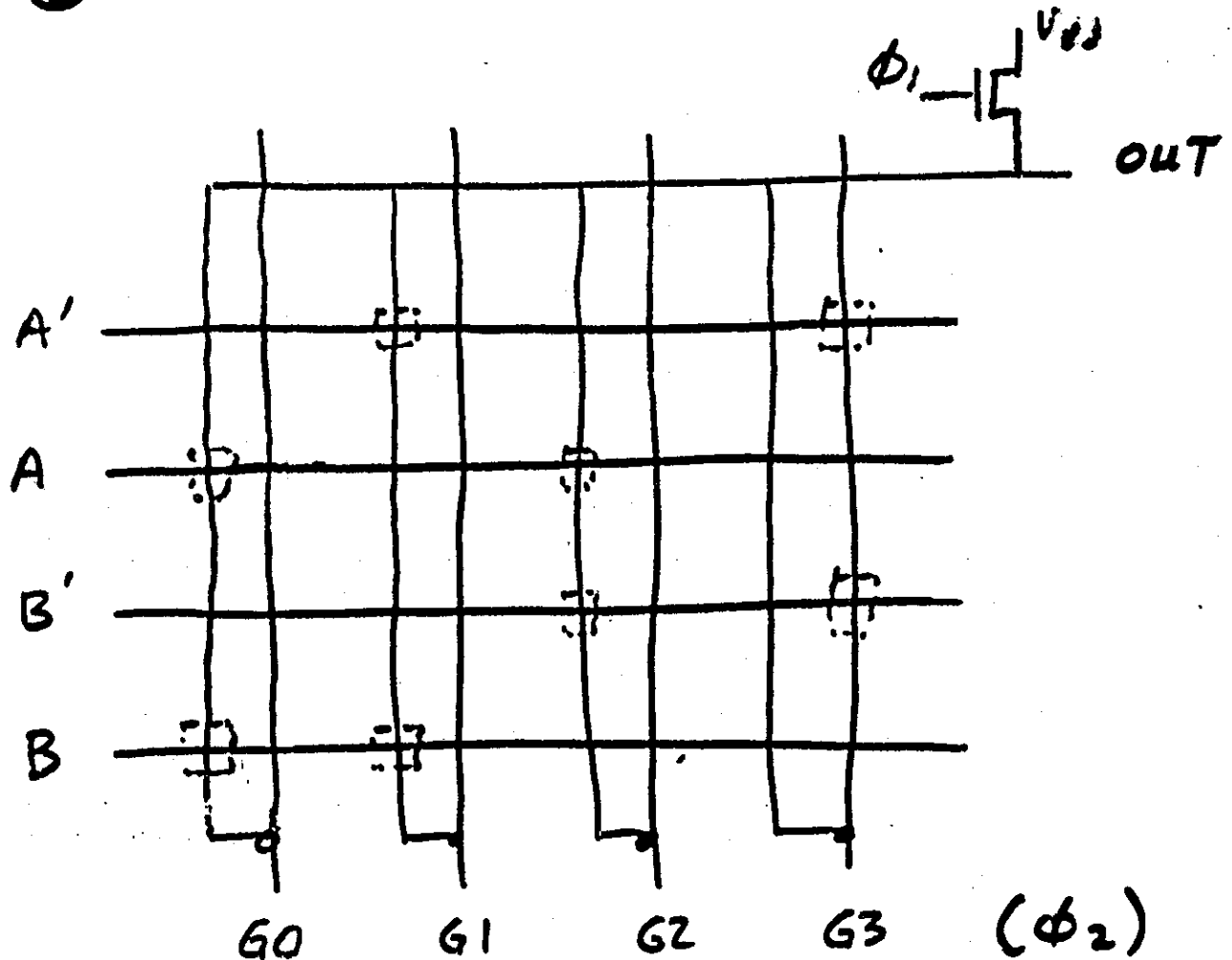


Fig 4b. Actual Layout of the Function Block

CLOCKED LOADS : PRE-CHARGING

- ① USES ENHANCEMENT MODE LOAD
LOAD
(PULLS UP TO $V_{DD} - V_{TH}$)
- ② CAN INCREASE SPEED
- ③ NO DC POWER CONSUMPTION



$OUT = XNOR \text{ IF } G1 = G2 = LOW$
 $G0 = G3 = HIGH$

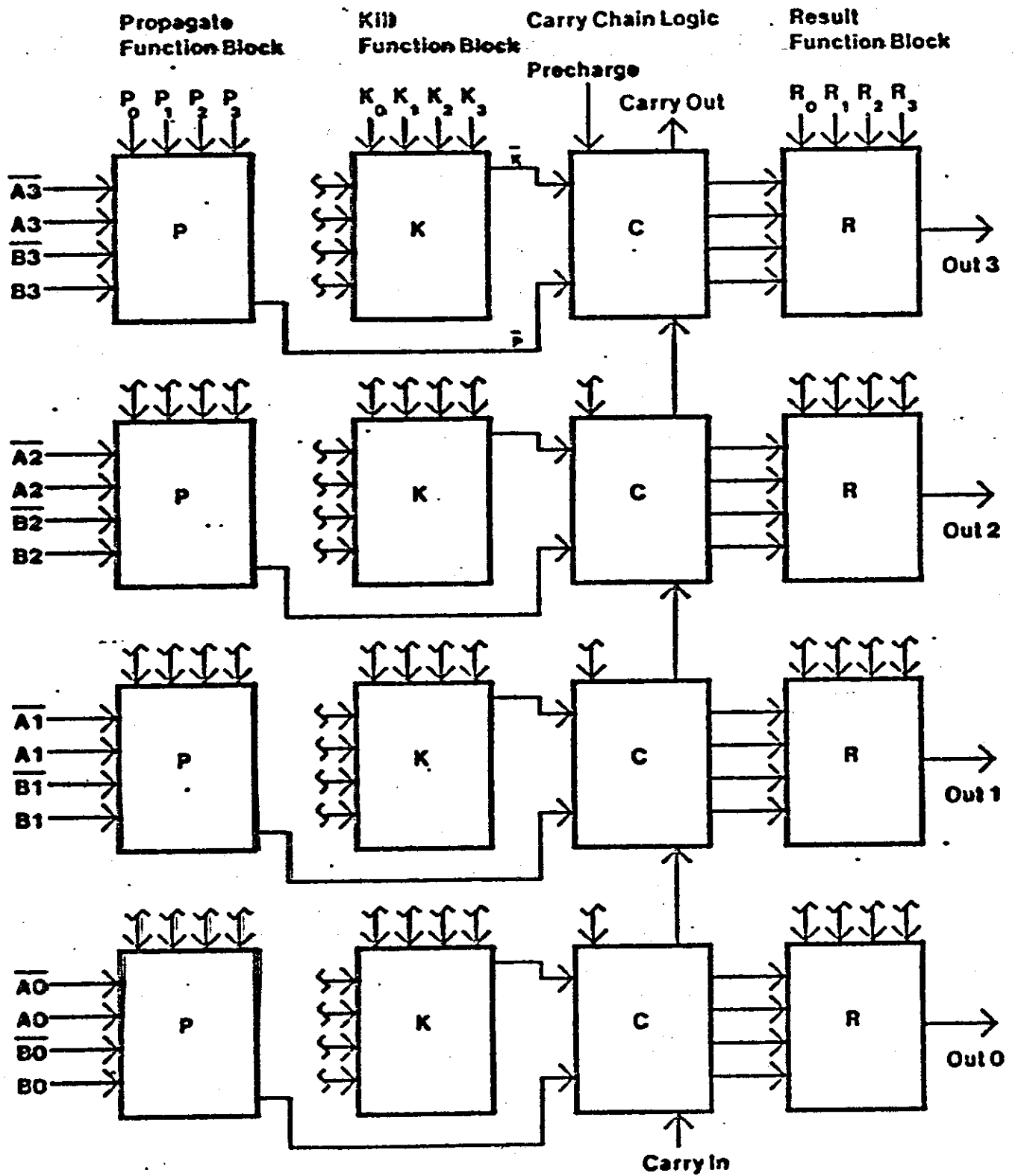


Figure 6. Block Diagram of a 4-Bit ALU.

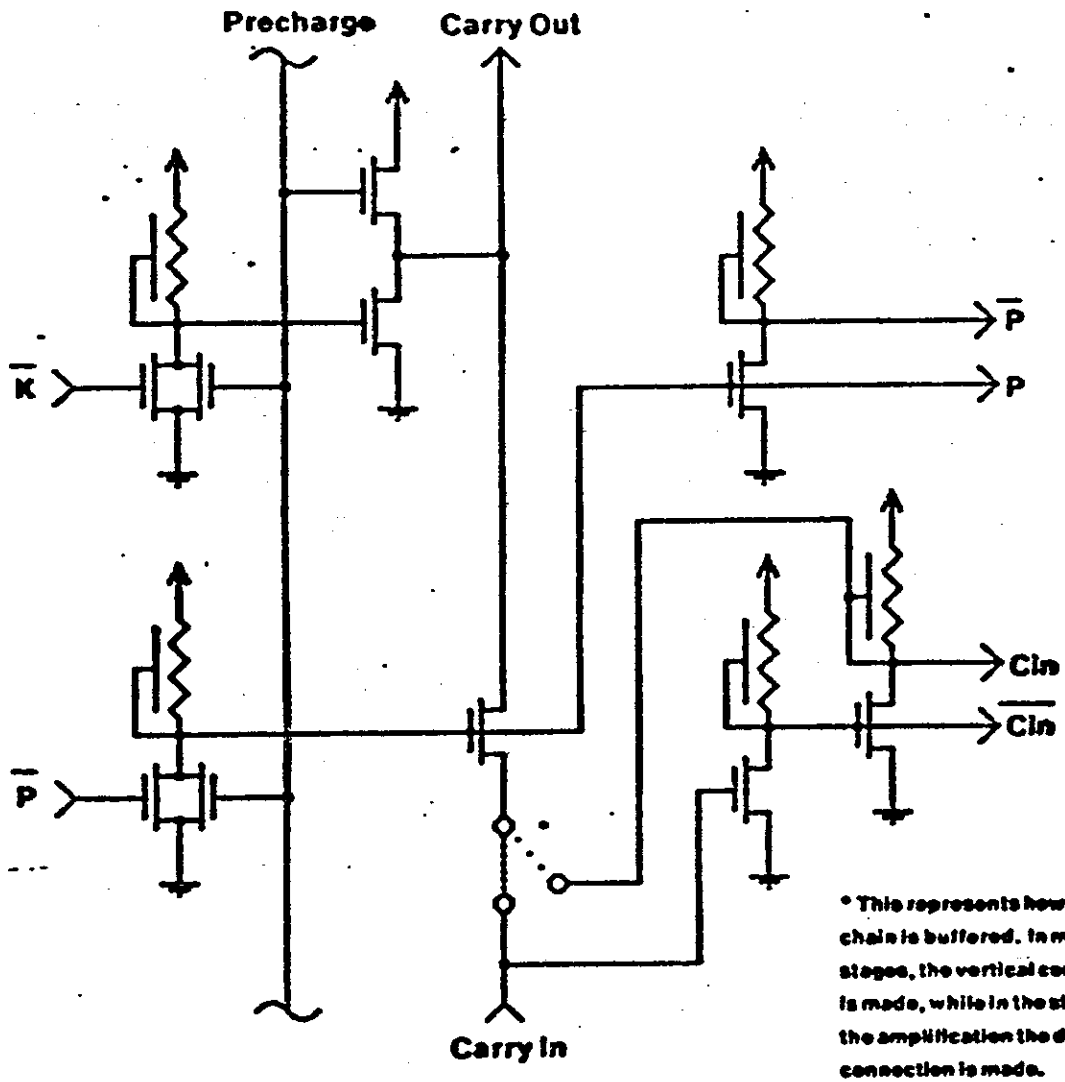


Figure 2. Carry Chain Circuit for the Arithmetic Logic Unit.

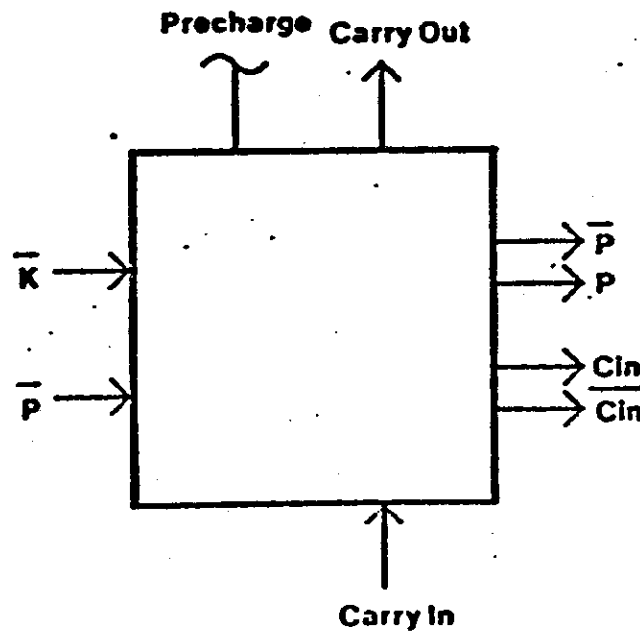


Figure 3. Abstraction of the Carry Chain Circuit.

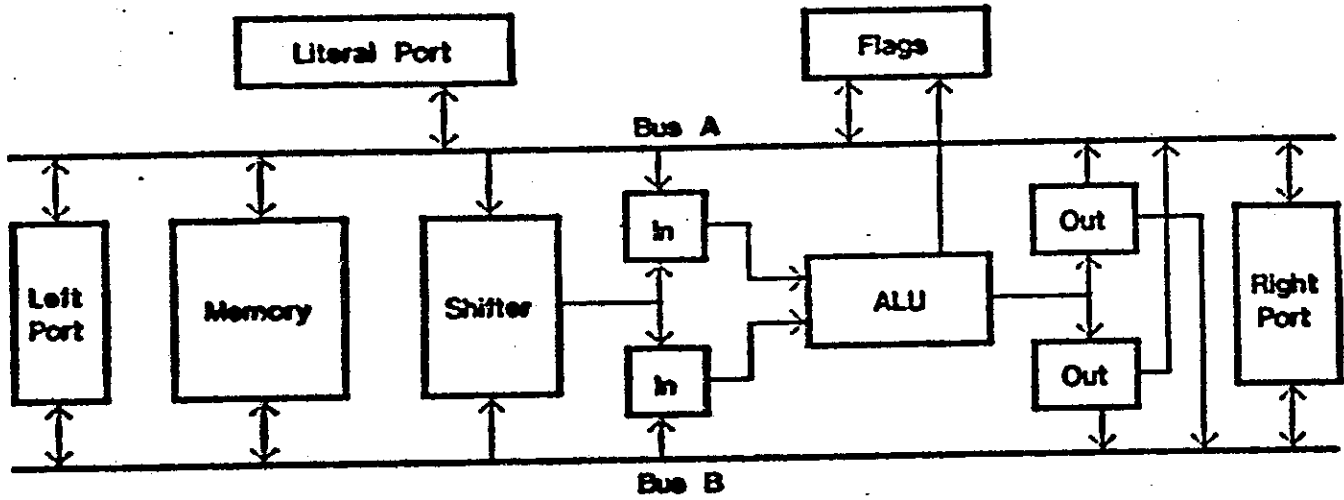


Figure 2. Block Diagram of OM2

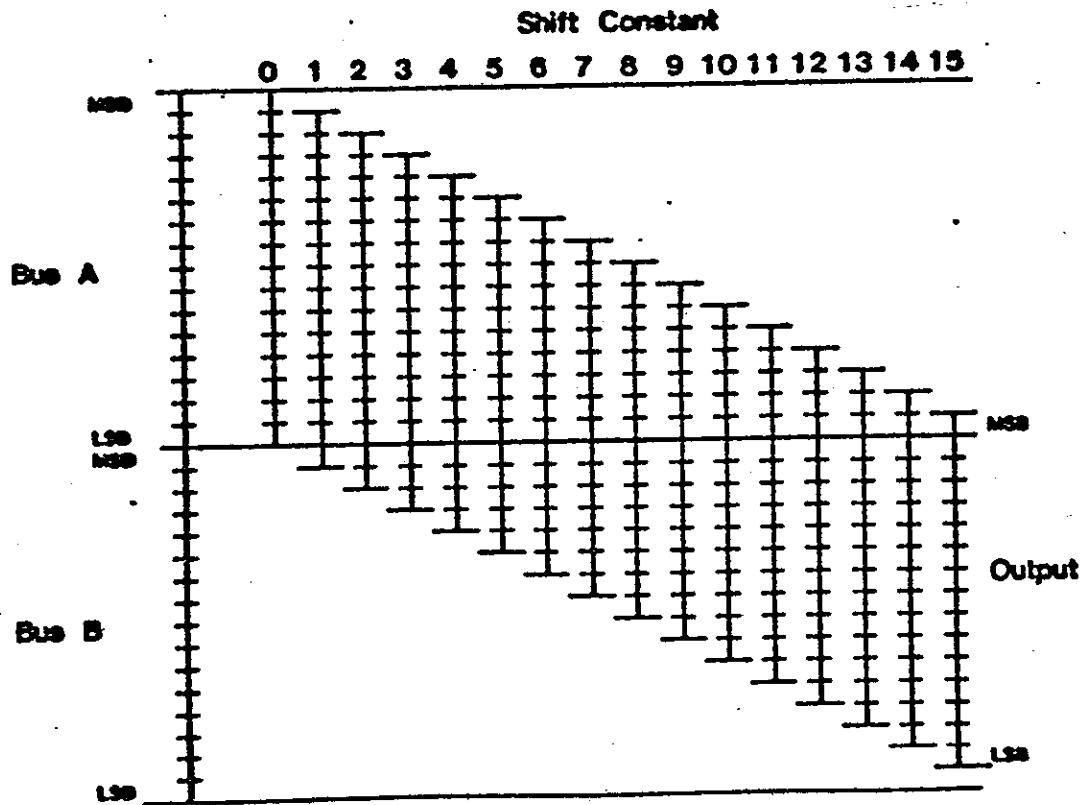
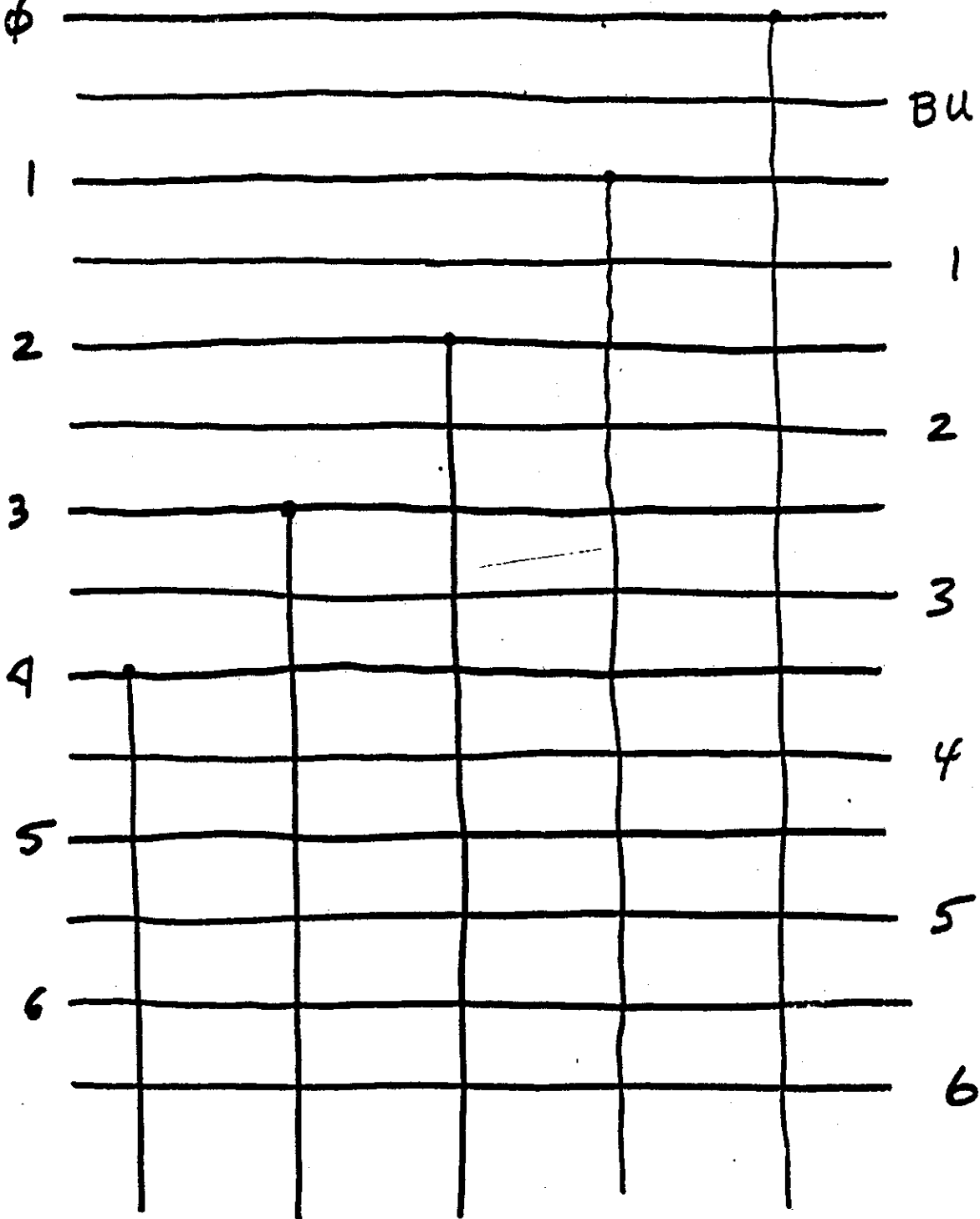
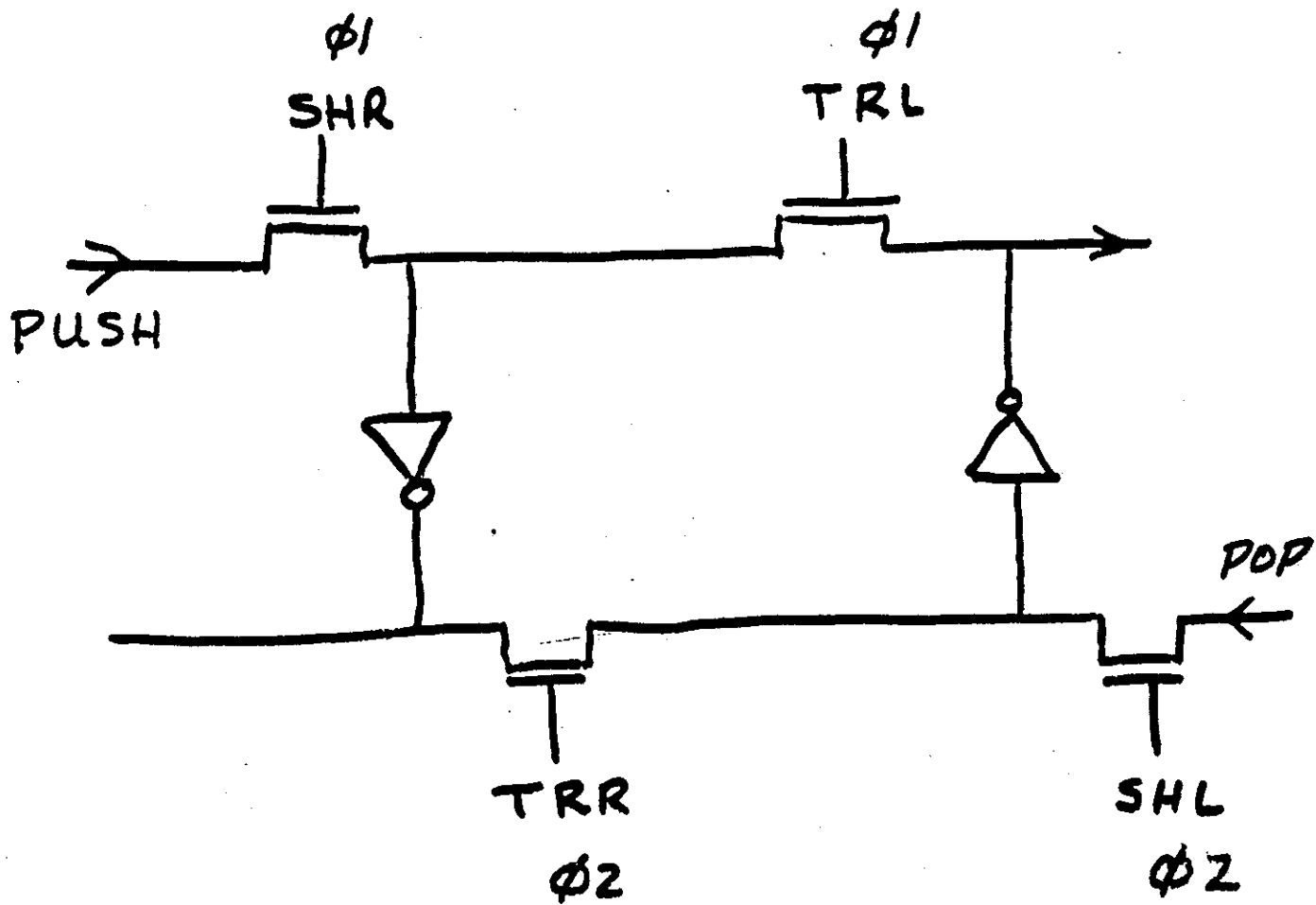


Figure 3. Shifter Operation.

LN
BUS ϕ

OUT



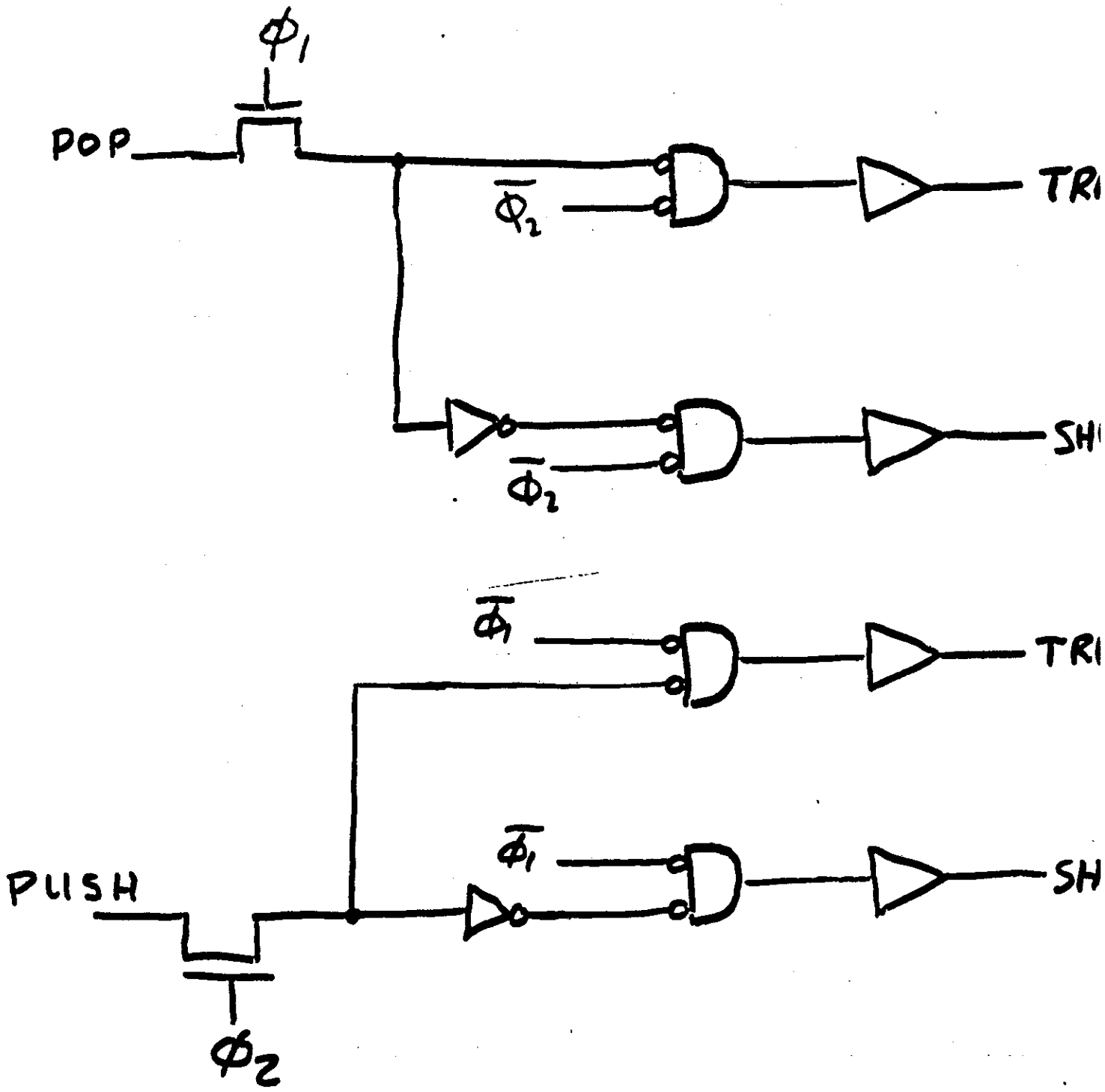


$$\text{SHR} = \text{PUSH} \cdot \phi 1$$

$$\text{SHL} = \text{POP} \cdot \phi 2$$

$$\text{TRL} = \overline{\text{PUSH}} \cdot \phi 1$$

$$\text{TRR} = \overline{\text{POP}} \cdot \phi 2$$



NOTE:  \Rightarrow 

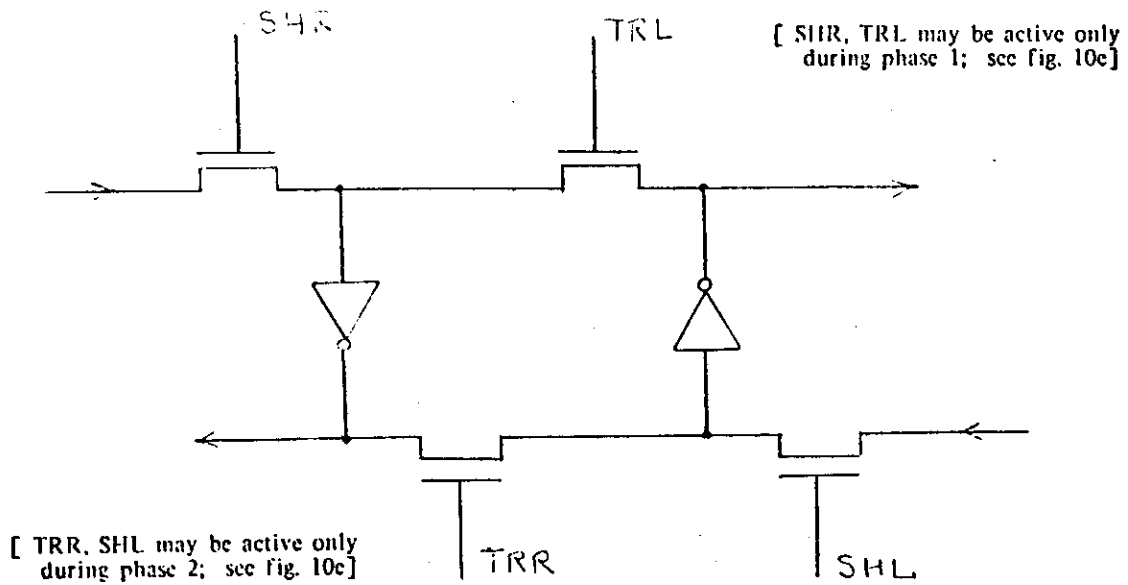


Fig.10a. One Horizontal Row of the Stack

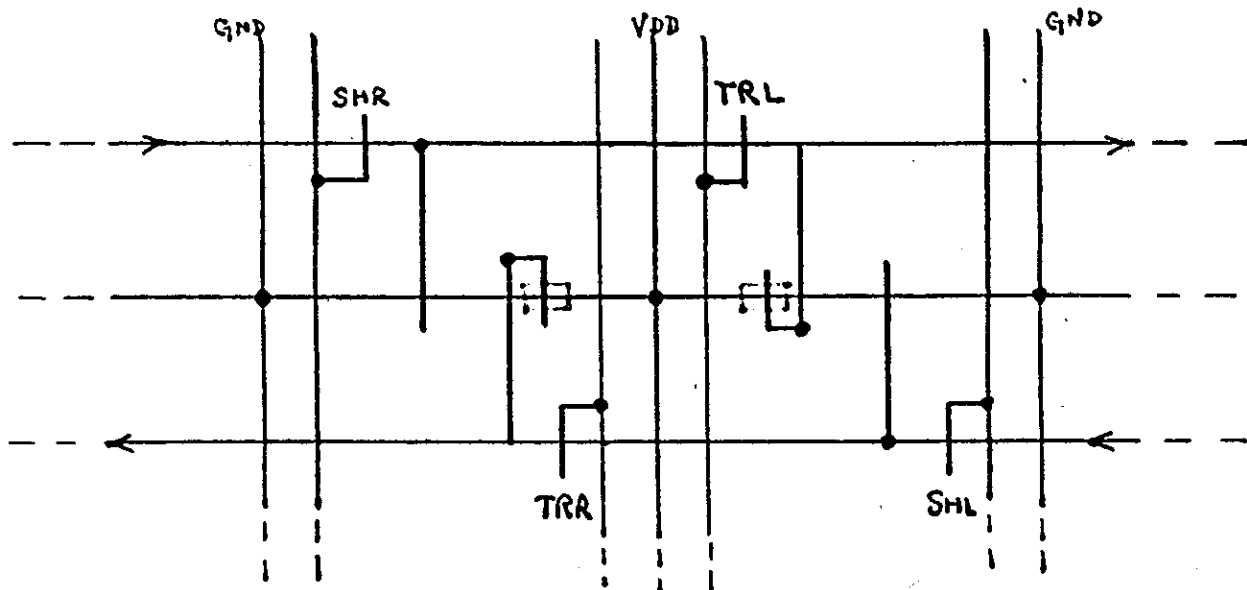
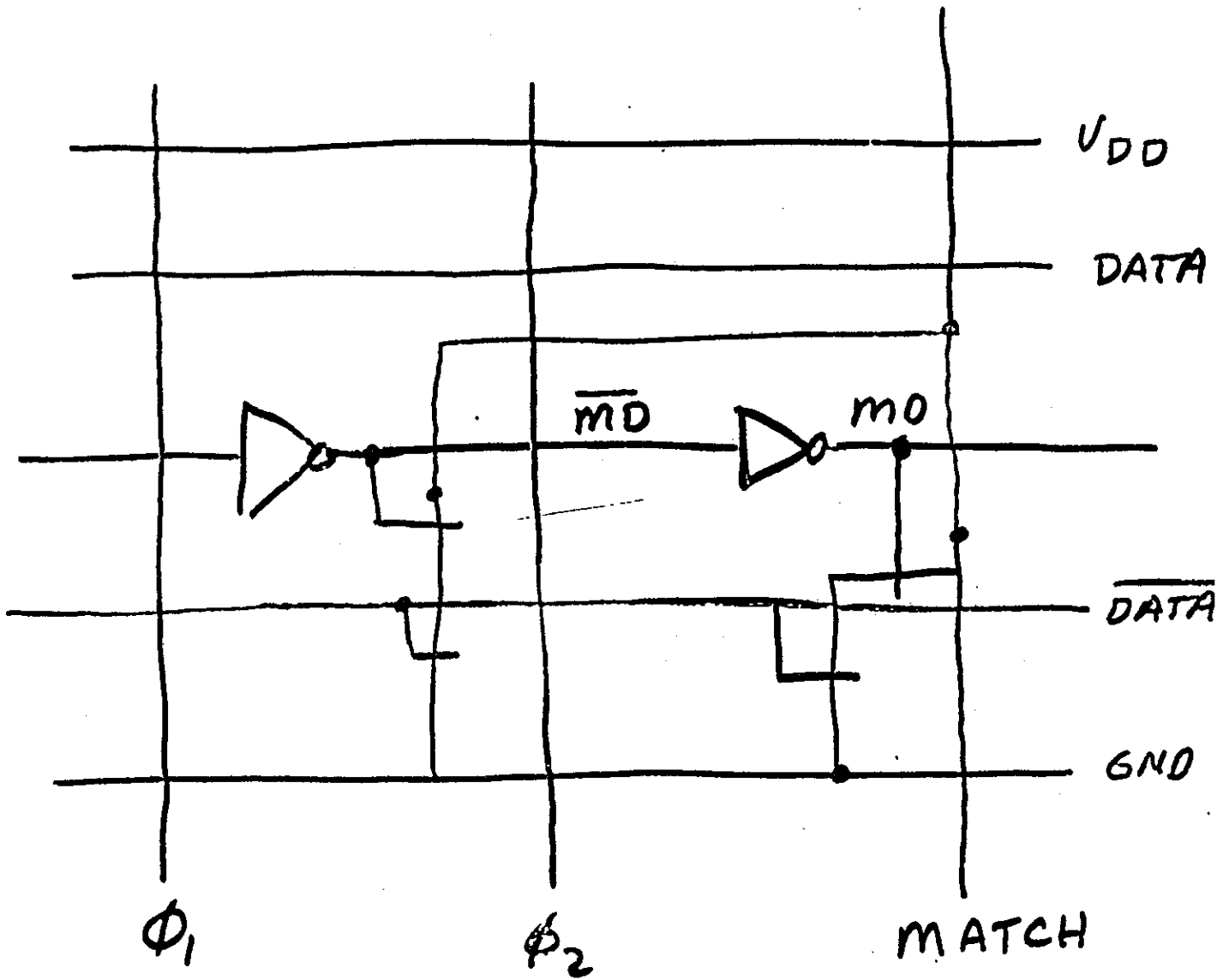


Fig.10b. Topology of One Horizontal Stack Row

POSSIBLE EXAMPLES OF SMART MEMORIES :

- ① MEMORIES WHICH SORT THEMSELVES
- ② MEMORIES WHICH DO GARBAGE COLLECTION
- ③ MEMORIES WHICH GENERATE A RASTER DISPLAY FROM DATA REPRESENTED AS RECTANGLES
- ④ CONTENT ADDRESSABLE MEMORIES.



$$\overline{MATCH} = DATA \cdot \overline{MD} + \overline{DATA} \cdot MD$$

An Introduction to VLSI Systems Design

LSI Systems Area - Xerox Palo Alto Research Center
Homework #2 - February 27, 1979

File: LsiHomework-2.bravo

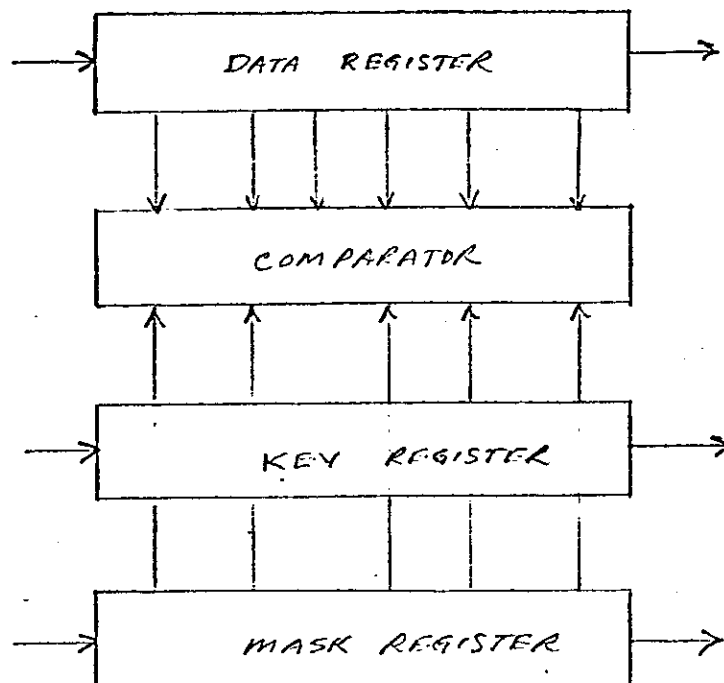
The basic goal of today's assignment is to explore the design of clocked structures. In tomorrow's assignment you will be asked to actually lay out the designs you do here. Having the stick diagrams prepared ahead of time will make learning the layout process easier.

1. Prepare a stick diagram of a four-bit synchronous counter. The implementation may take a variety of forms including a PLA or a less regular structure. How easily is your design expanded to 8 or 12 bits?

2. Figure 1 below sketches the block diagram of a serial bit string comparator. The function of this subsystem is as follows: A stream of data bits is clocked through a *data register*, which might be 64 bits long. Each clock cycle, the data bits in the register are compared with a previously loaded pattern of bits in a *key register*. However only a subset of the bits are actually compared, namely those in positions marked by a pattern of bits in a *mask register*. The comparison is made in a *comparator*, which has a *match line* running through it. If any of the data bits in positions marked by the mask bits are different from the key bits, then the Match Line is pulled low. Otherwise it is left high.

Prepare a stick diagram of one bit of this structure, including each of the registers and the matching logic.

FIG. 1



PROJECTS :

- 1) DUE APRIL 20
(LAYOUTS FINISHED AND CHECKED)
- 2) NOTHING IS TOO SMALL
- 3) IF YOU DON'T HAVE SPECIFIC IDEA , WORK WITH SOMEONE WHO DOES.
- 4) IF YOUR FAVORITE PROJECT IS TOO BIG FOR A FIRST TRY - DO A PIECE OF IT
OR ...
- 5) FIND SOMEONE TO WORK WITH YOU.

TOPICS FOR TODAY:

- 1) REVIEW HOMEWORK
- 2) IC PROCESSING FOR NMOS
LAYOUT → PACKAGED CHIP.
- 3) LAYOUT DESIGN RULES
- 4) ICARUS DEMONSTRATION

DICK LYON:

- 1) REAL NMOS TRANSISTORS
- 2) SUPER BUFFERS
- 3) PAD I/O
- 4) MORE CIRCUIT DETAILS

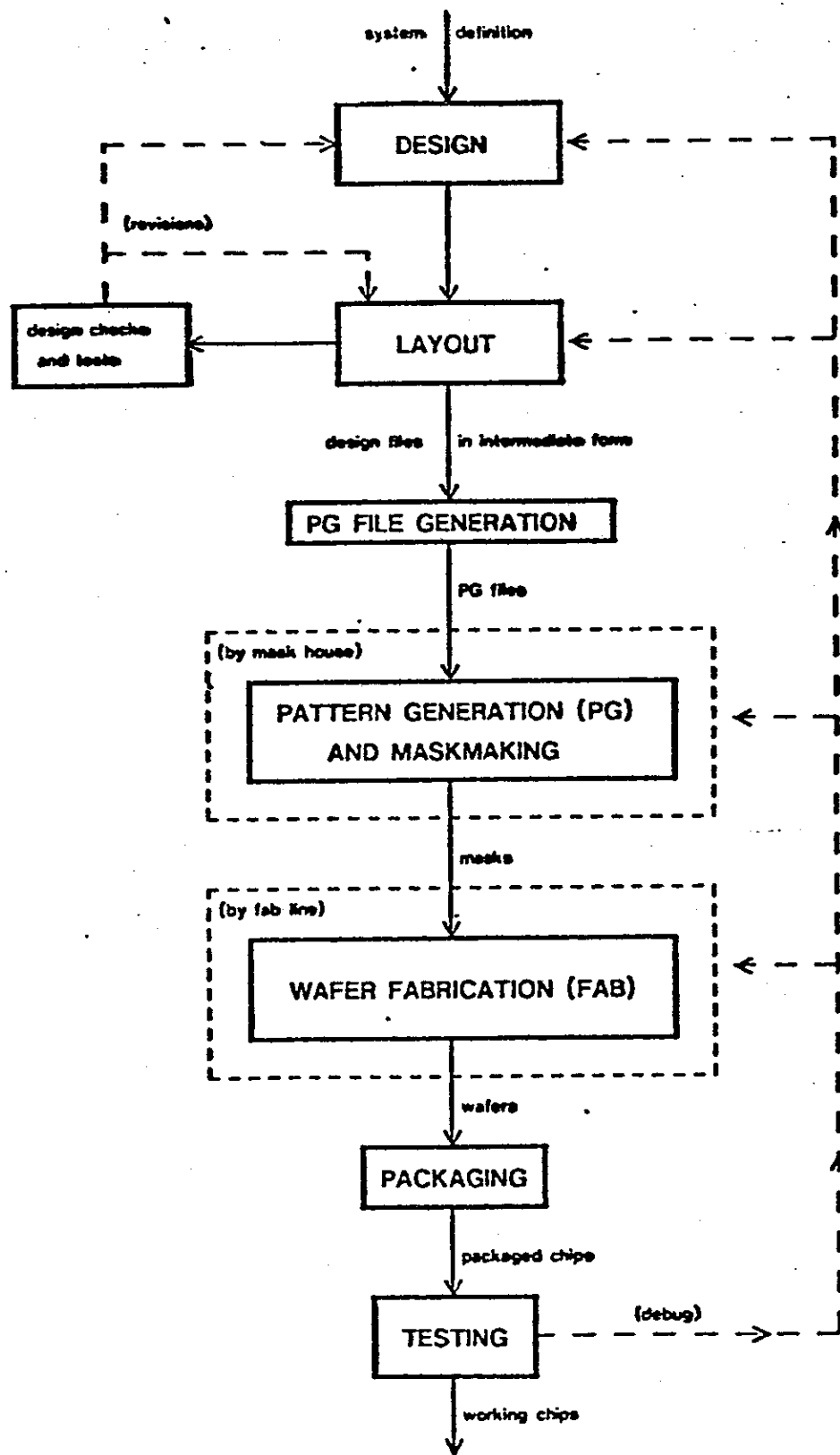


Fig. 1. Overview of Integrated System Implementation

INTEGRATED CIRCUIT FABRICATION

INVOLVES THREE MAJOR STAGES:

1) MASK FABRICATION

- OUTSIDE VENDOR
- WE TAKE THEM A LAYOUT DESCRIPTION ON MAS TAPE
- COST: \$3K - \$6K
- TURN-AROUND: 2-8 WEEKS
- RESULT: 1 MASK (GLASS PLATE) W/ ARRAY OF CHIP PATTERS FOR EACH LAYER.

2) WAFER PROCESSING:

- IN-HOUSE OR OUTSIDE VENDOR
- WE TAKE THEM A SET OF MASKS
- COST: ~ \$3K
(10-20 WAFERS)
- TURN AROUND:
2-8 WEEKS
- RESULT: FINISHED SILICON WAFERS

3) PACKAGING

- IN-HOUSE

- BASIC STEPS -

- SAW OR SCRIBE
WAFER

- BREAK WAFER
INTO CHIPS

- MOUNT CHIPS IN
PACKAGE

- BOND WIRES FROM
PADS TO PINS

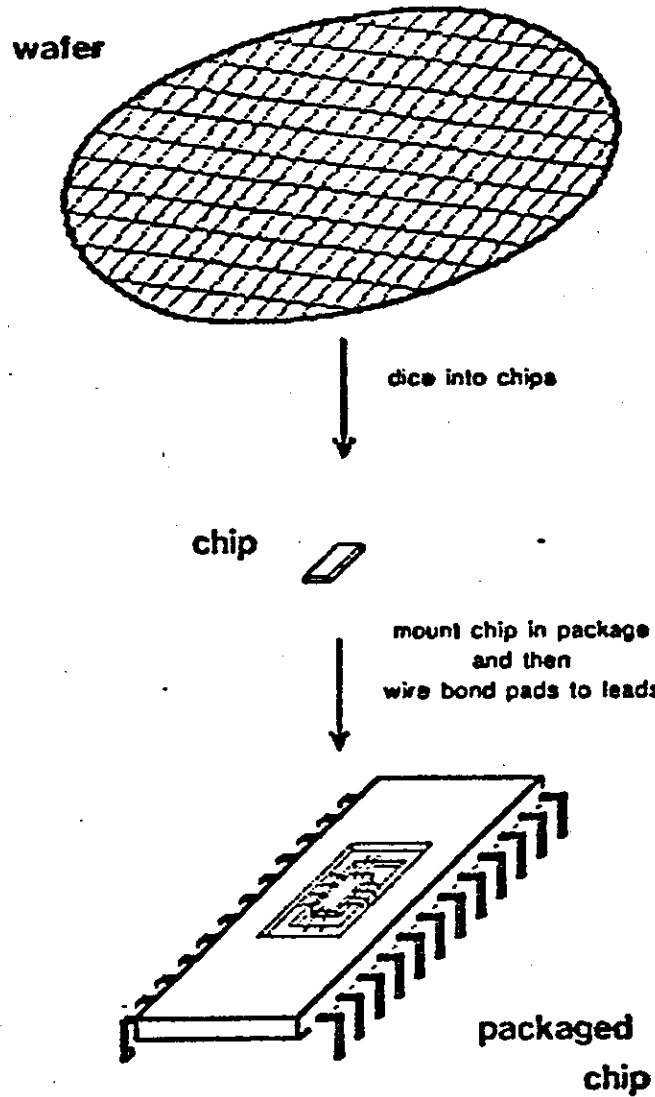


Fig. 7. Sketch Illustrating the Packaging Sequence

(wafer.press)

MASK MAKING PROCESS

PG TAPES



PATTERN GENERATION ON
RETICLE (10X)



STEP AND REPEAT RETICLES
ON MASK



MAKE SUBMASTERS AND
WORKING PLATES OF MASK.

DIRECT E-BEAM MASKS ARE AN
ALTERNATIVE.

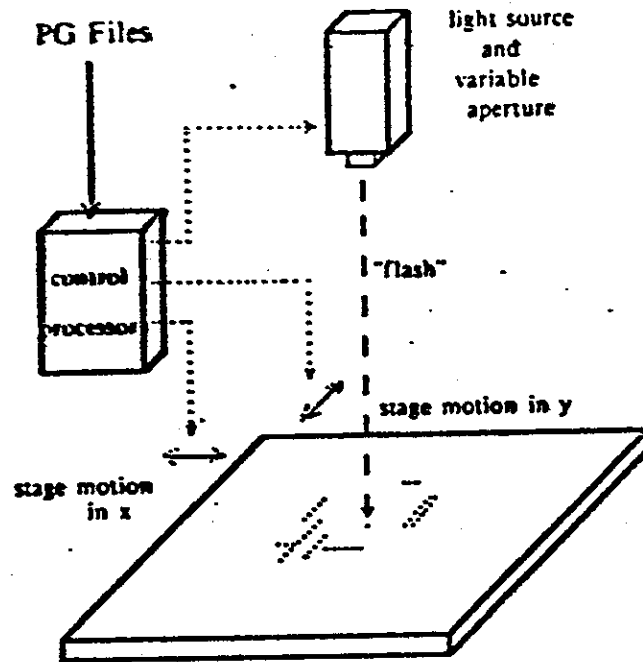


Fig. 3. Sketch Illustrating Function of Pattern Generator

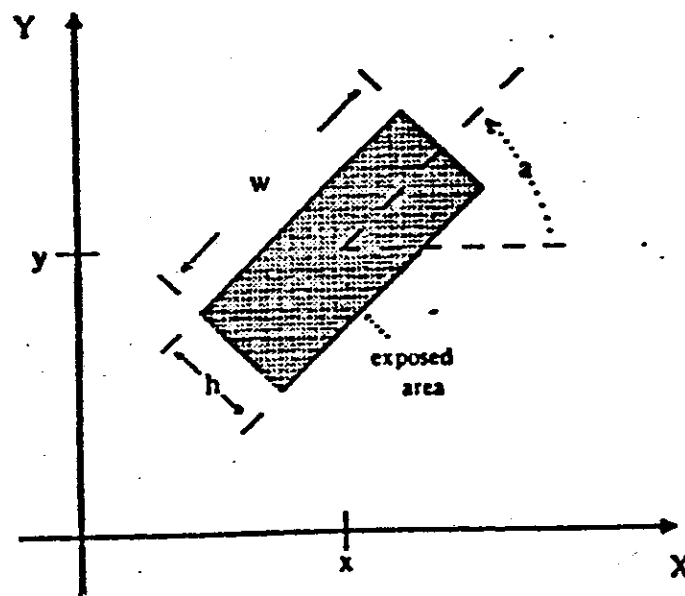


Fig. 4. Parameters of One Flash of Pattern Generator

(pg.press)

WAFER PROCESSING

1. PATTERN INDEPENDENT
2. IMPORTANT TO UNDERSTAND HOW PATTERN IS TRANSFERRED FROM MASK TO WAFER
3. PROCESSING INVOLVES BUILDING A "SANDWICH" OF ALTERNATING LAYERS OF CONDUCTORS AND INSULATORS
4. THE PROCESS DESCRIBED WILL BE A SIMPLIFIED VERSION OF WHAT HAPPENS - ADEQUATE FOR DESIGNERS HOWEVER.

Figure 1. Bare Wafer

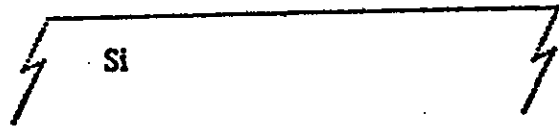


Figure 2. Oxidation



Figure 3. Coat w/Resist

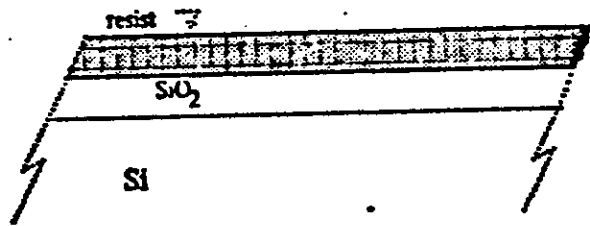


Figure 4. Mask & Expose

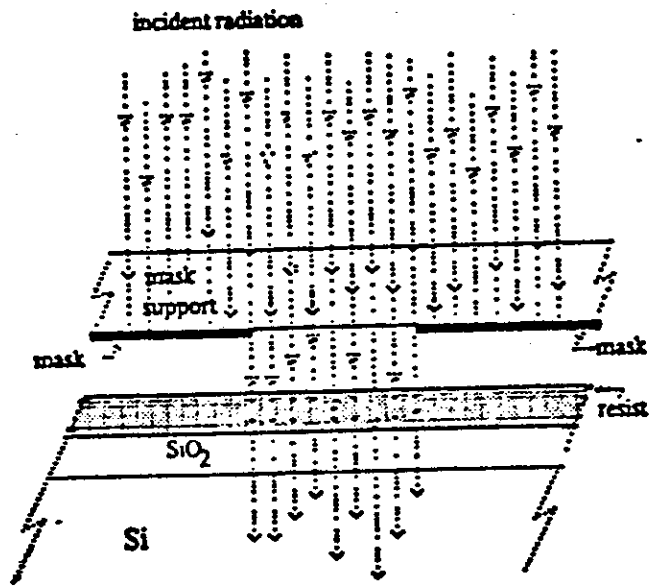


Figure 5. Exposed Resist

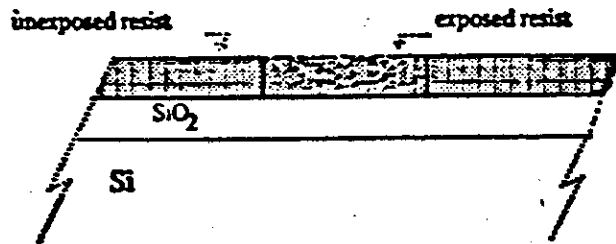


Figure 6. Develop Resist

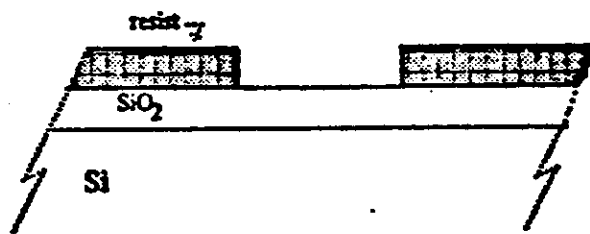


Figure 7. Etch

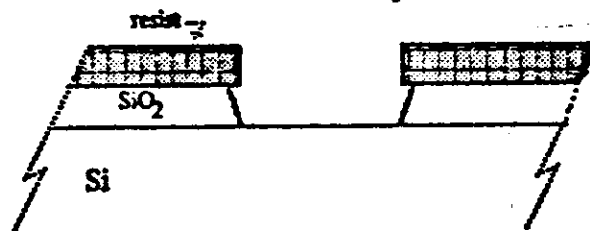
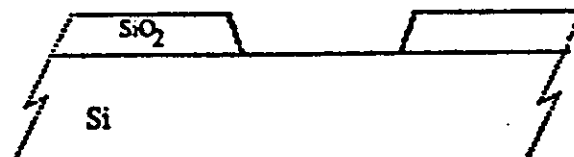


Figure 8. Remove Resist



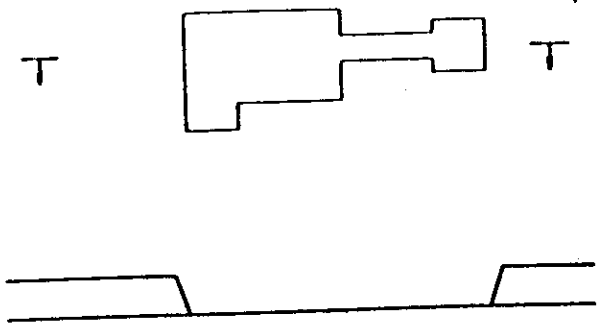


Fig.9. Patterning SiO₂

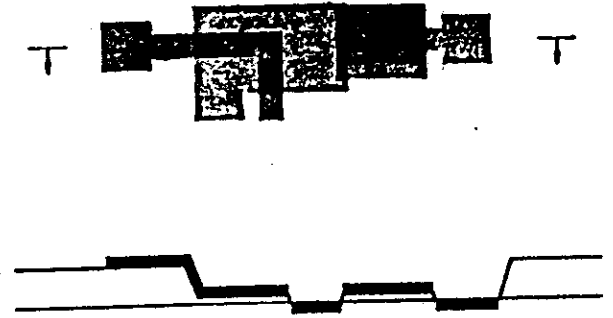


Fig.12. Placing Diffused Region

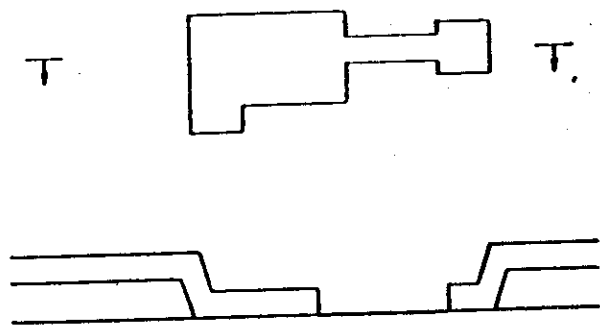


Fig.10. Patterning Ion Implantation

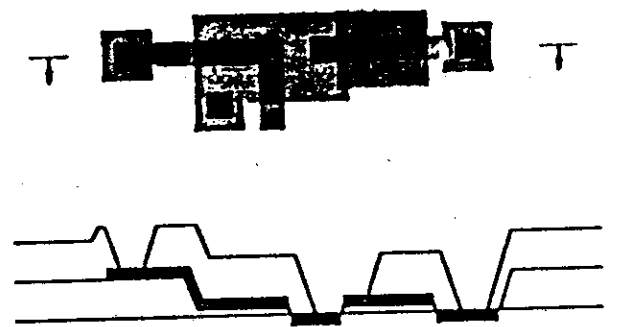


Fig.13. Placing Contact Cuts

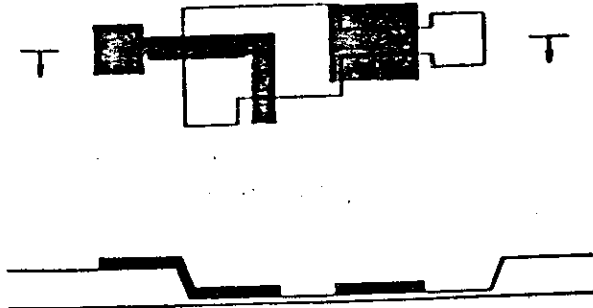


Fig.11. Patterning Polysilicon

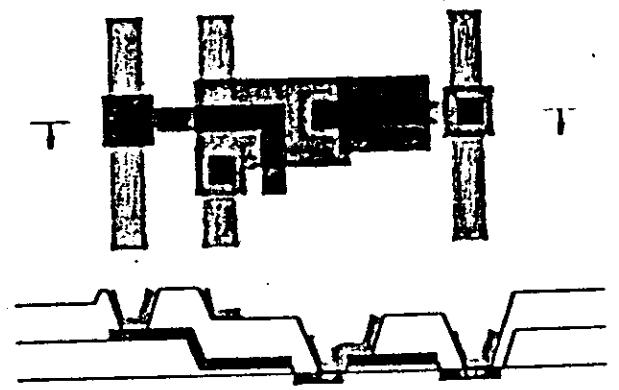


Fig.14. Patterning the Metal Layer

NOTE:

1. THIN OXIDE REGROWTH
UNDER POLY (GATE OXIDE)
2. POLY BLOCKS DIFFUSION
(SELF ALIGNED GATE)
3. CONTACT CUTS GO ONLY SO
FAR (TO CLOSEST SILICON)
4. BUTTING CONTACT
(BURIED CONTACT IS
ALTERNATE METHOD TO
CONNECT POLY AND
DIF)

NOW USE WHAT WE'VE LEARNED
ABOUT PROCESS TO DETERMINE
A SET OF DESIGN RULES.

MAJOR ISSUES TO CONSIDER

1) WHAT SHOULD BE MIN.
LINE WIDTH?

2) WHAT SHOULD BE MINIMUM
DISTANCE BETWEEN LINES?

3) HOW WELL CAN ONE LAYER
BE REGISTERED TO ANOTHER?

⇒ WE WILL DEFINE A METRIC (λ)
WHICH CHARACTERIZES THE
RESOLUTION OF THE PROCESS
AND EXPRESS ALL DESIGN RULE
IN TERMS OF λ .



Fig.15. $W_d/\lambda \geq 2$

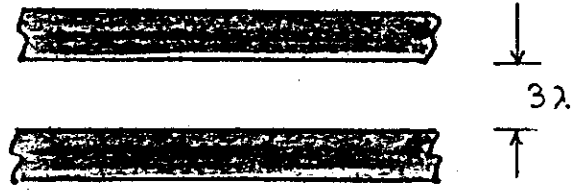


Fig.16. $S_{dd}/\lambda \geq 3$



Fig.17. $W_p/\lambda \geq 2$

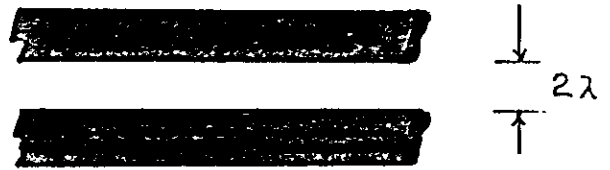


Fig.18. $S_{pp}/\lambda \geq 2$

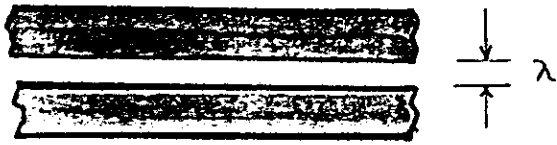


Fig.19. $S_{pd}/\lambda \geq 1$

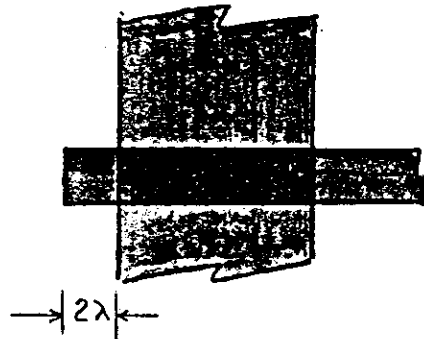


Fig.20. $F_{pd}/\lambda \geq 2$

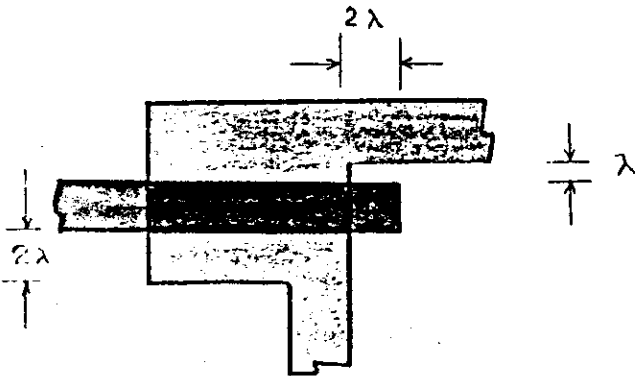


Fig.21. Example of Several Rules

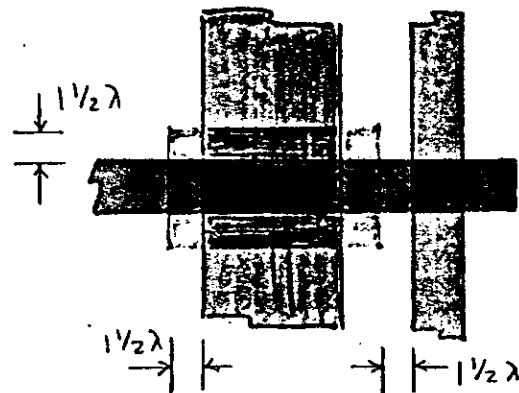


Fig.22. $S_{ig}/\lambda \geq 1\frac{1}{2}$ $F_{ig}/\lambda \geq 1\frac{1}{2}$

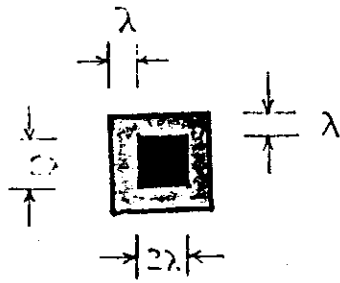


Fig. 23. $W_c/\lambda \geq 2$, $E_{dc}/\lambda \geq 1$

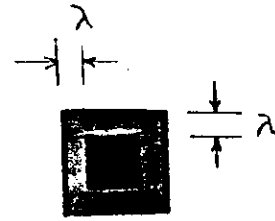


Fig. 24. $E_{pc}/\lambda \geq 1$

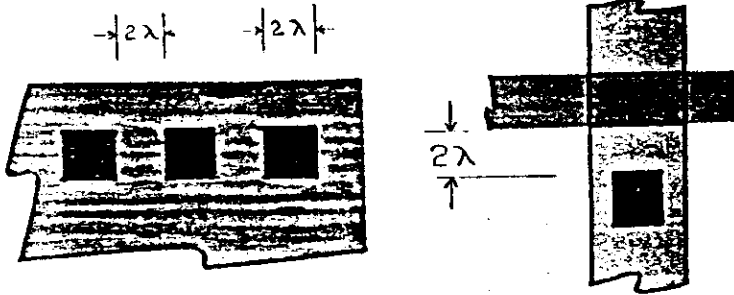


Fig. 25. $S_{cc}/\lambda \geq 2$, $S_{cg}/\lambda \geq 2$

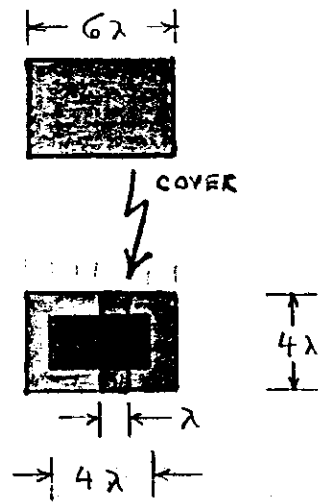


Fig. 26. $O_{pd}/\lambda = 1$,
and details of butting contact

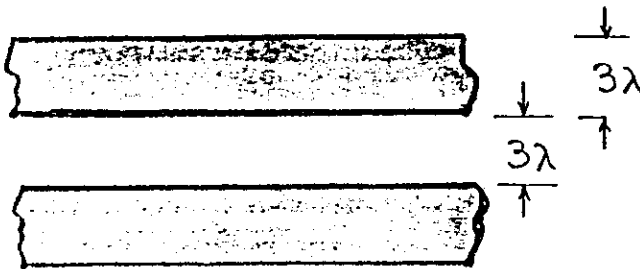


Fig. 27. $W_m/\lambda \geq 3$, $S_{mm}/\lambda \geq 3$

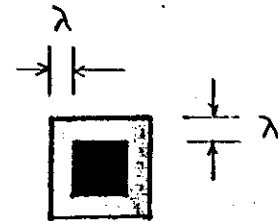
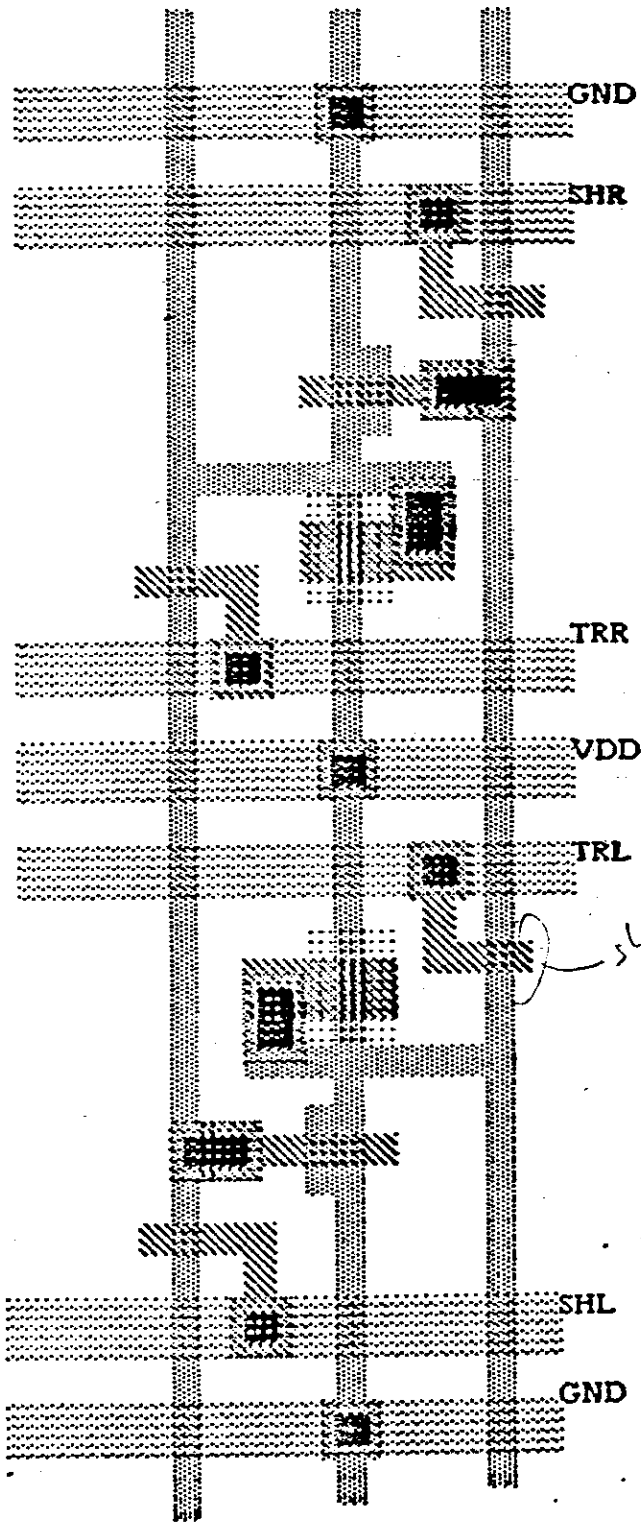
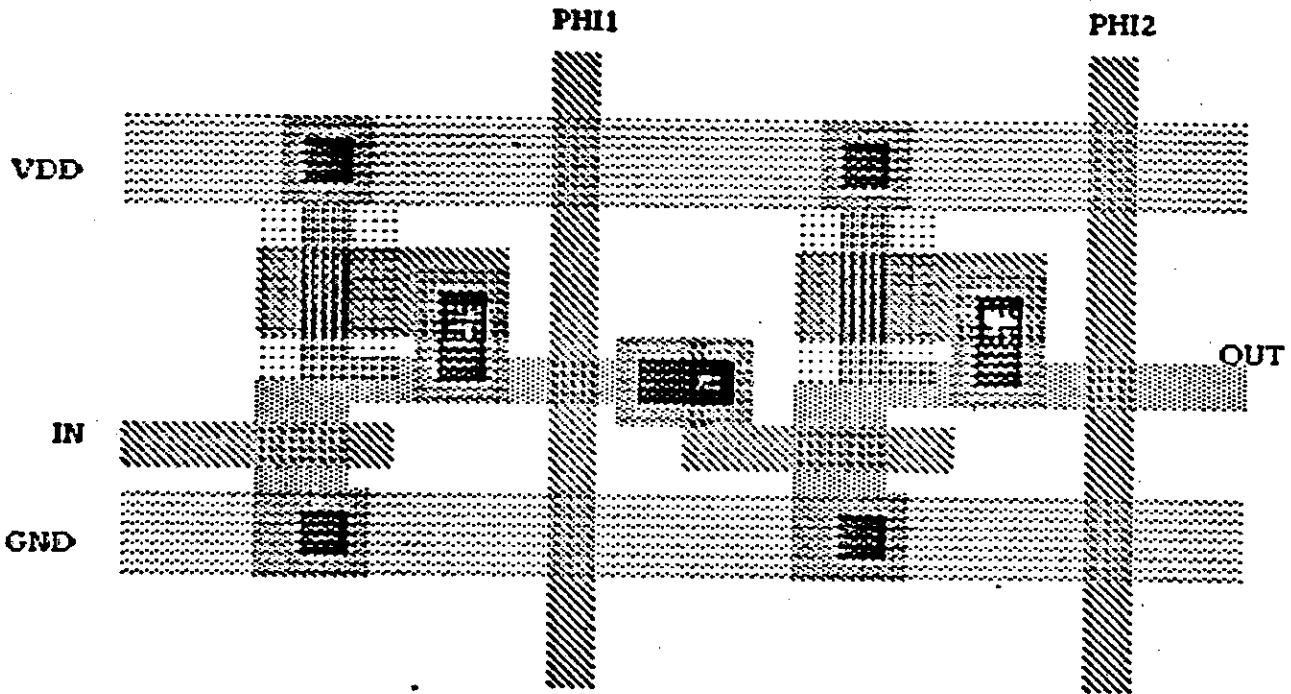


Fig. 28. $E_{mc}/\lambda \geq 1$

Stack: left-Right shift Reg



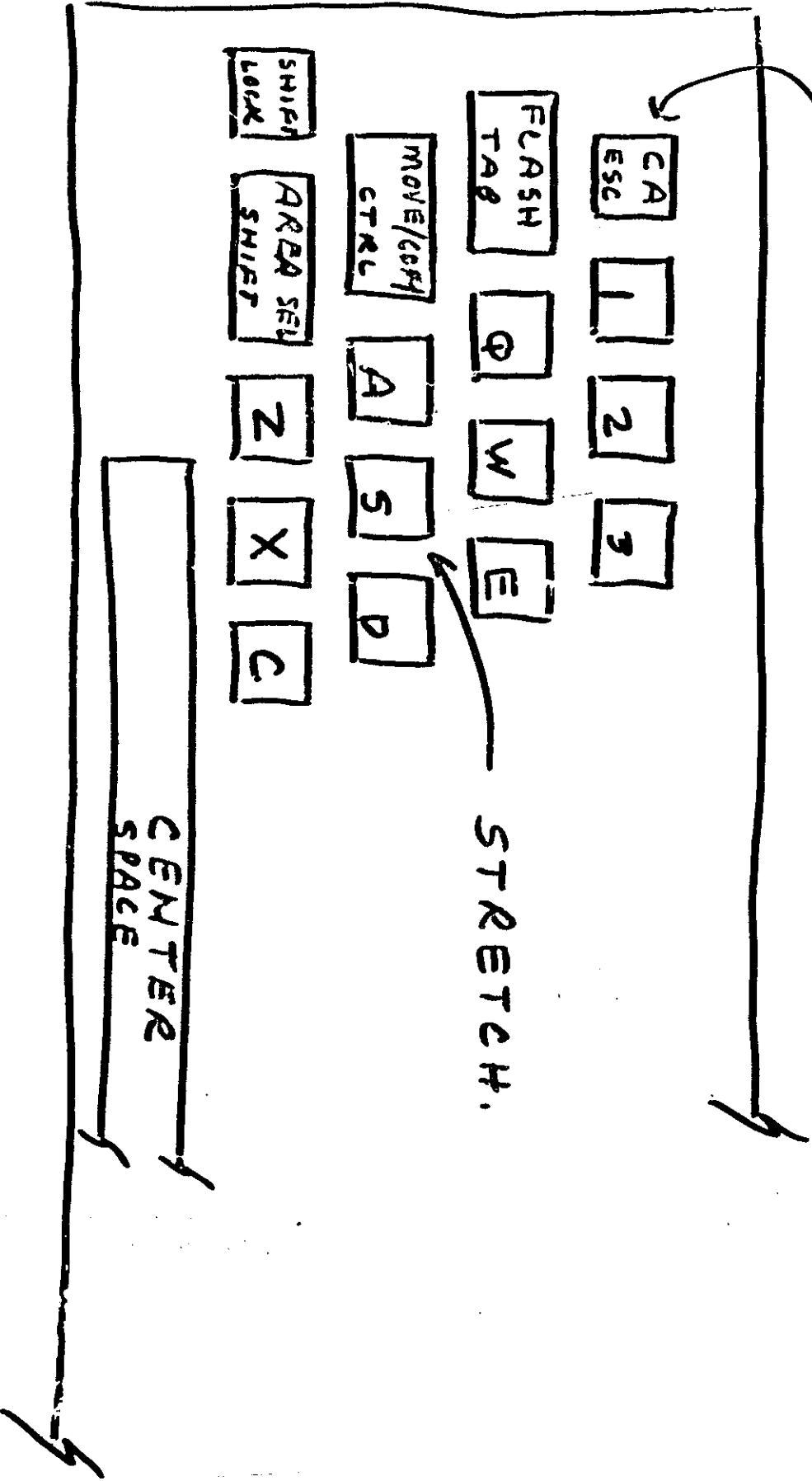
should be 27 overlap



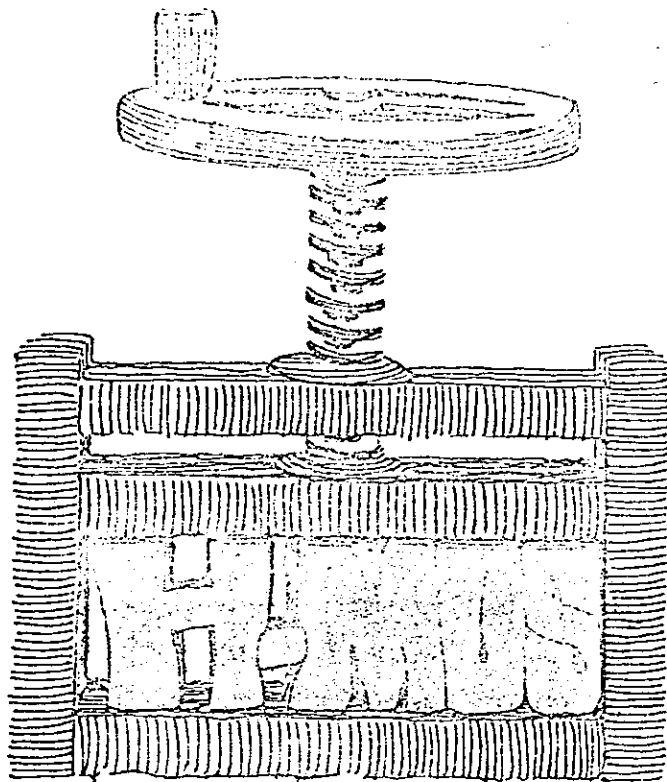
Design rule violation.
In shift register, inverters must be 8:1
because of voltage drop in pass transistors.

I CARUS COMMAND KEYS

COMMAND ACCEPT







H-MOS scales traditional devices to higher performance level

by Richard Pashley, Kim Kokonnen, Edward Boleky, Robert Jecmen, Samuel Liu, and William Owen
Intel Corp., Santa Clara, Calif.

□ It has almost become a law of nature, the way metal-oxide-semiconductor devices double in density or performance every year. Over the last decade, MOS chips have gone from being low-density shift registers, gates, and flip-flops operating at millisecond speeds to being entire memories, microprocessors, and dedicated systems and subsystems packing tens of thousands of electronic functions into a single device that is capable of nanosecond operation.

Fueling this astonishing progress is the tremendous versatility of metal-oxide-semiconductor technology. Starting out as a high-threshold p-channel multiple-supply circuit technique capable at best of simple calculator and serial-storage functions, MOS moved to n-channel single- and double-layer structures that use a single 5-volt power supply to perform complex computer instructions in less than 100-ns cycles and static and dynamic memory operations in less than 50 ns—all at ever lower power dissipations.

Now MOS circuit technology stands at a still higher level. For the first time it can challenge the performance of bipolar circuits, while continuing to set new records in complexity and low cost. The techniques that have proved capable of achieving this breakthrough are various but share a crucial characteristic in that they all shorten the effective channel length, or drain-source spacing, of the fundamental MOS transistor.

Two approaches are possible. One relies on a double-diffused process: a depletion-mode device with a relatively long, 5-micrometer channel under the MOS gate is integrated in series with a 1- μm enhancement-mode channel, which is formed by the outdiffusion of boron through the self-aligned source-junction opening. The process is called D-MOS when the double-diffused structure has a planar configuration, but V-MOS when the structure has a vertical configuration, with the surface of the MOS transistor laid on the face of a V-shaped groove etched anisotropically into the silicon substrate. In either

case, the double-diffused structure requires new process technology and circuit structures that differ markedly from standard silicon-gate techniques.

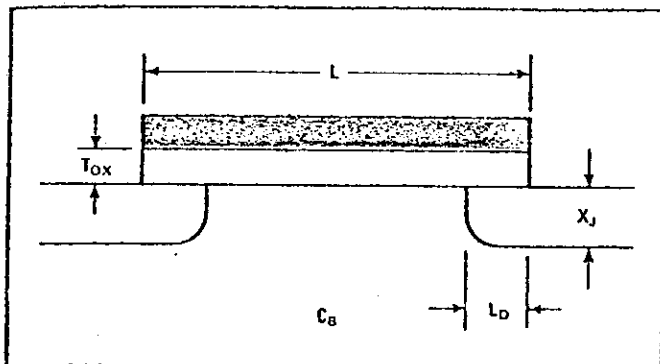
The other approach relies on scaling down the size and parameters of MOS devices directly—in other words, device-scaling conventional n-channel silicon-gate structures. This is nothing new; right from the start MOS designers knew that by trimming down the size of their devices they could achieve higher speed, higher density, and lower power dissipation.

To achieve their new high-performance process called H-MOS, Intel has chosen the direct device-scaling method for two reasons. First, it evolves directly out of standard silicon-gate processing and so requires neither new device structures nor complex circuit schemes (either requirement would make yields and fabricating costs too unpredictable to guarantee their usefulness over a wide range of semiconductor products). Second, it fits in with the trend to smaller and smaller circuit patterns, as photolithographic methods grow more refined and electron-beam wafer-fabrication techniques stand ready to take over.

Further, double-diffused structures have a limited future. They may have been appropriate two to three years ago, when the industry was unable to build channels less than 5 or 6 μm long. But now that 4- μm (and soon 3- and 2- μm) channel lengths are possible, the need for new structures like D-MOS and V-MOS may simply be in the process of vanishing.

How to scale an MOS device

Figure 1 shows the cross section of a silicon-gate n-channel device, where L is the channel length, T_{ox} is the gate-oxide thickness, X_j is the junction depth, L_D is the lateral diffusion, and C_s is the substrate doping level. Now, first-order scaling theory says that the characteristics of an MOS device can be maintained and the desired operation assured if the parameters of the device are



1. Scaling down. To reduce the size of an MOS device, all physical parameters must be scaled down proportionally. If the channel length L is shortened by $1/S$ where S is the scaling factor, then the oxide thickness, T_{ox} , the lateral underdiffusion, L_D , and the junction depth, X_j , must also be scaled down by $1/S$. Meanwhile, the substrate doping constant, C_B must be increased by S .

scaled as shown in Table 1. When S is the scaling factor and the channel length L is scaled by a factor of $1/S$, then the other device dimensions—the thicknesses of the gate oxide and the lateral underdiffusion, the device width and junction depth—must also be scaled $1/S$. Moreover, to maintain adequate threshold voltage and drain-source breakdown voltage, the scaling theory also states that the substrate doping concentration must be increased by S , while the supply voltage and current decrease by $1/S$.

The effect on performance

When this is done properly, the increase in the performance of the device is dramatic, as Table 1 also shows. The parasitic capacitance, gate delay, power dissipation, and power-delay product all improve markedly. Since the parasitic capacitance goes down roughly as the junction depth decreases, it too scales by $1/S$; this means that since gate delay is roughly proportional to parasitic capacitance, it is scaled by $1/S$ as well. Moreover, since the device's power dissipation is proportional to the supply voltage and current, it scales by the still stronger factor of $1/S^2$. Finally, the bottom line of all this is the power-delay product, or figure of merit, of the MOS device; and since it is the product of the gate delay and power dissipation, it is scaled down by a very significant factor of $1/S^3$. Thus, scaling the dimensions of an MOS device improves its performance by the cube of its scaling factor.

In short, by reducing the dimensions of a circuit, the MOS designer gains enormous leverage on its density and performance—a statement that happens also to describe a recurring event in MOS history. Table 2 places the move to H-MOS in this perspective. Notice the sharp reduction in circuit parameters that occurred between 1976 and 1977 when Intel went to H-MOS from standard n-channel silicon-gate processing. By reducing the channel length from 6 to 3.5 μm and decreasing the other parameters appropriately, it was possible to quarter the speed-power product. This improvement would have been even larger had the supply voltage been scaled as required by a first-order device-scaling theory, instead of being kept at the more acceptable 5-V system

Device/circuit parameter	Scaling factor
Device dimension, T_{ox} , L , L_D , W , X_j	$1/S$
Substrate doping, C_B	S
Supply voltage, V	$1/S$
Supply current, I	$1/S$
Parasitic capacitance, WL/T_{ox}	$1/S$
Gate delay, VC/I (τ)	$1/S$
Power dissipation, VI	$1/S^2$
Power-delay product	$1/S^3$

Device/circuit parameter	Enhancement-mode n-MOS 1972	Depletion-mode n-MOS 1976	H-MOS 1977	MOS 1980
Channel length, L (μm)	6	6	3.5	2
Lateral diffusion, L_D (μm)	1.4	1.4	0.6	0.4
Junction depth, X_j (μm)	2.0	2.0	0.8	0.8
Gate-oxide thickness, T_{ox} (\AA)	1,200	1,200	700	400
Power supply voltage, V_{CC} (V)	4-15	4-8	3-7	2-4
Shortest gate delay, τ (ns)	12-15	4	1	0.5
Gate power, P_D (mW)	1.5	1	1	0.4
Speed-power product (pJ)	18	4	1	0.2

level. However, by 1980, as channel length shrinks to 2 μm and the supply voltage to 3 v, performance will improve even more dramatically, this time by a factor of five, to become altogether 20 times better than that of 1976 MOS devices.

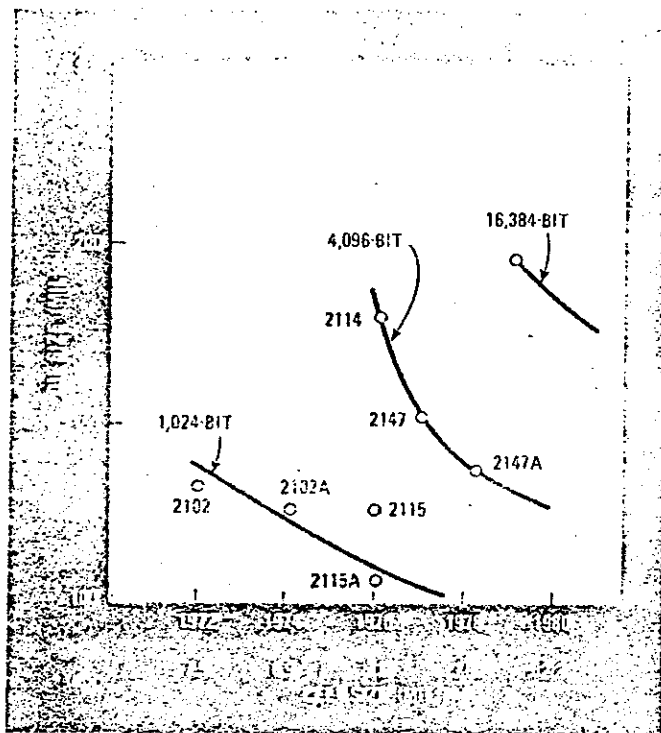
An example of what device scaling means to the user of integrated circuits is given in Figs. 2 and 3, which chart the progress made over the years in static random-access memories. In 1972, the standard static MOS RAM was the 500-ns 2102, built with a 6- μm channel length and 1,200-angstrom gate-oxide thickness. Its resulting speed-power product was 18 picojoules. It occupied a silicon chip nearly 140 mils on a side and had a cell size of almost 8 square mils.

In 1974, the 2102 was redesigned around a depletion-load n-channel technology that shrank its die area by 15% and its access time to 200 ns. To this process oxide isolation and built-in substrate bias were added in 1976, to create the 2115 static RAM that accessed the same 1,024 bits in less than 70 ns. Today, the impact of device scaling is even more apparent with H-MOS, which fits the 2115 RAM (now called the 2115A) onto a chip slightly larger than 100 mils on a side, while improving access time typically to 25 ns.

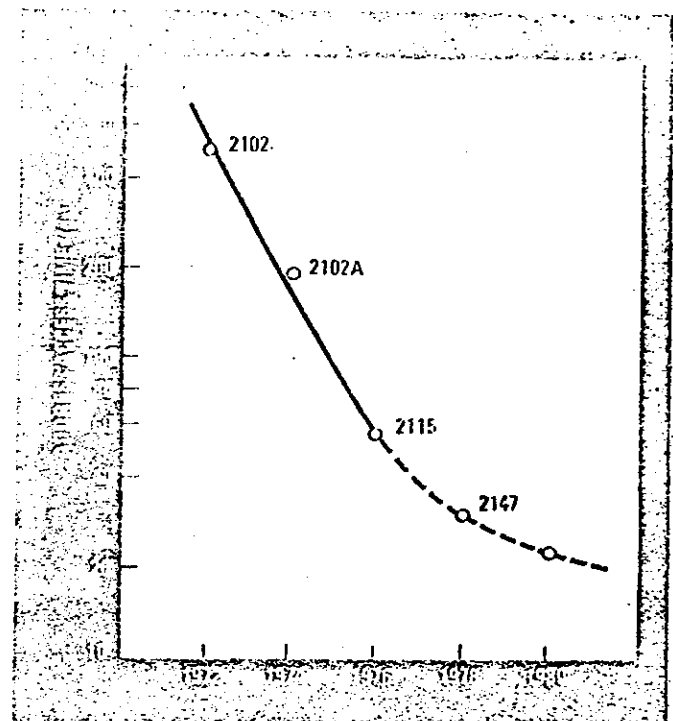
More to come

Moreover, applied to a 4,096-bit static memory design, H-MOS results in a chip a little larger than the original 2102, yet pushes access times typically below 50 ns. Finally, as the MOS process evolves and scaling continues, a 16,384-bit fully static RAM will fit on a chip no larger than 200 mils on a side and offer system designers access times in the 50-ns range.

The high speed and high density of H-MOS are achieved through five major improvements in MOS technology, four of which are directly related to device



2. Better and better. Thanks to the vigorous development of n-channel silicon-gate MOS technology, static RAMs continually improve in density. In comparison with earlier processes, today's H-MOS increases device packing density by a factor of 4.



3. Faster, too. Process improvements and device scaling, as embodied in H-MOS, are also making MOS RAMs faster. In 1972, a typical 2102 1,024-bit static RAM had an access time of 600 ns; today's 2147 4,096-bit parts can be accessed typically in 45 ns.

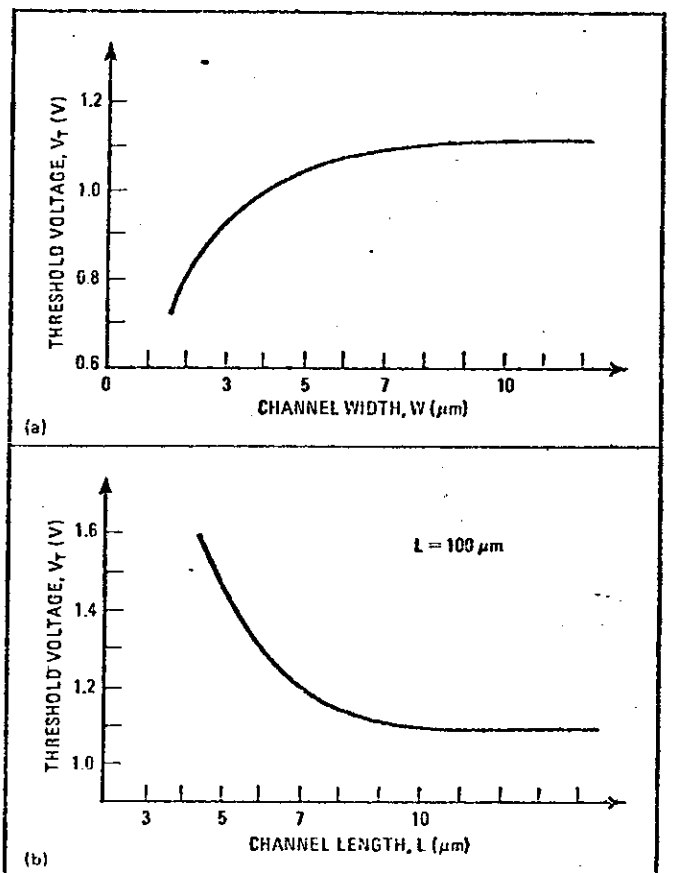
scaling. A high-resistivity substrate is used, while device-scaling theory is applied to gate-oxide thickness, junction depth, gate length, and threshold-modifying ion implants.

The high-resistivity substrate made of 50-ohm-cm, p-type material is used to lower junction capacitance, reduce the substrate body effects that degrade performance, and increase the device's effective carrier mobility. All three factors result in faster, lower-power devices.

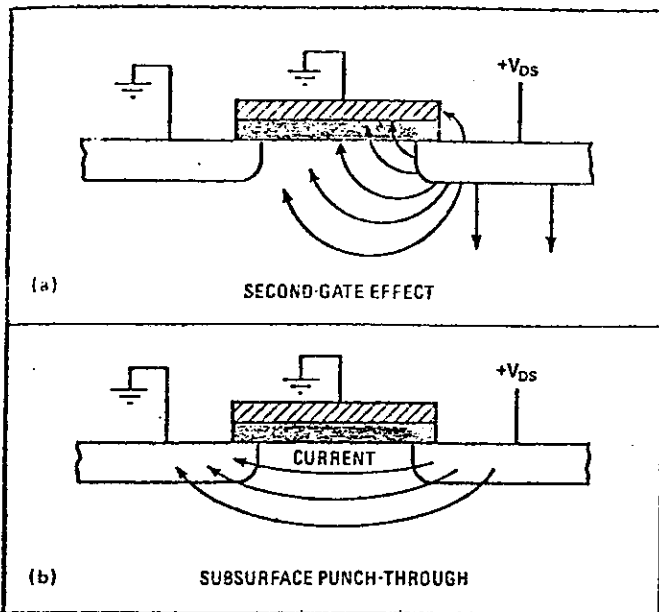
Scaling the H-MOS gate oxide down to 700 angstroms increases device gains and punch-through voltages and reduces body and short-channel effects, thereby increasing performance and reliability. The junction depth is scaled to approximately 0.75 μm by using slow-diffusing arsenic as the source-drain dopant. The shallowness of the junctions increases both speed, by reducing peripheral junction capacitance and gate-drain Miller capacitance, and density, by allowing smaller diffusion-to-diffusion spacing.

Scaling principles are also applied to the polysilicon-gate electrodes, which form the self-aligned source and drain diffusion regions. The narrow 3.5- μm polysilicon gates of H-MOS increase the device gain and still further increase circuit speed and density. Narrow gates, however, come at the expense of more severe photolithographic and etch control requirements, which are needed to avoid a wide variation in the electrical channel lengths of the device.

Finally, H-MOS threshold-voltage stability is maintained for both enhancement-mode and depletion-mode devices by using an ion-implanted channel region in conjunction with the high-resistivity substrate. This implant procedure controls threshold voltage with great



4. Maintaining that threshold. Decreasing channel length; and width in small devices has a strong effect on threshold voltage. When the channel length goes below about 5 micrometers, V_t begins to decline (a); while for widths below 7 μm , V_t begins to climb (b).



5. Second-order problems. Small devices are vulnerable to two second-order effects. One is the second-gate effect (a), where the electric field lines emanating from the drain junction end up on the oxide-silicon interface. The other is punch-through (b), which can be relieved by careful choice of the substrate impurity profile through ion implantation combined with a thin gate oxide.

precision and allows the MOS threshold voltage to be optimized independently of the substrate doping.

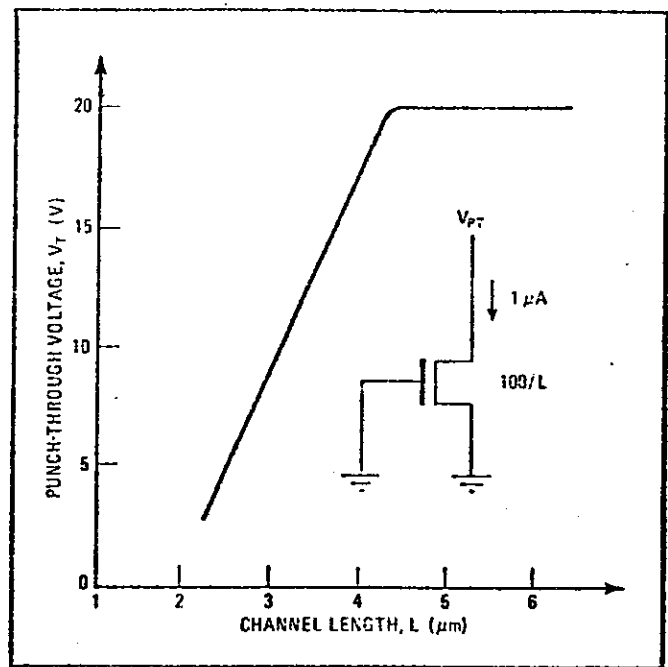
Happily, it proved possible to make all of these H-MOS technology advances within a relatively short time (about one year) without affecting the ability to manufacture devices at reasonable costs. H-MOS also is flexible enough to be applied over a broad range of circuit designs while maintaining its inherent high speed, small size, and low power. Unlike V-MOS and integrated injection logic, which require new circuit techniques to make them applicable to dynamic-memory and large-scale-integrated logic designs, H-MOS can be directly applied across the entire product spectrum.

Already the process has resulted in a family of static RAMs (the 1-k 2115A and 4-k 2147), which offer the industry the best speed-power performance of any memory. Moreover, work is under way on the application of H-MOS to a high-performance, 16-bit microprocessor family, a large variety of complex peripheral chips, 16-k and 65-k dynamic RAMs, high-density ready-only memories, and erasable programmable ROMs.

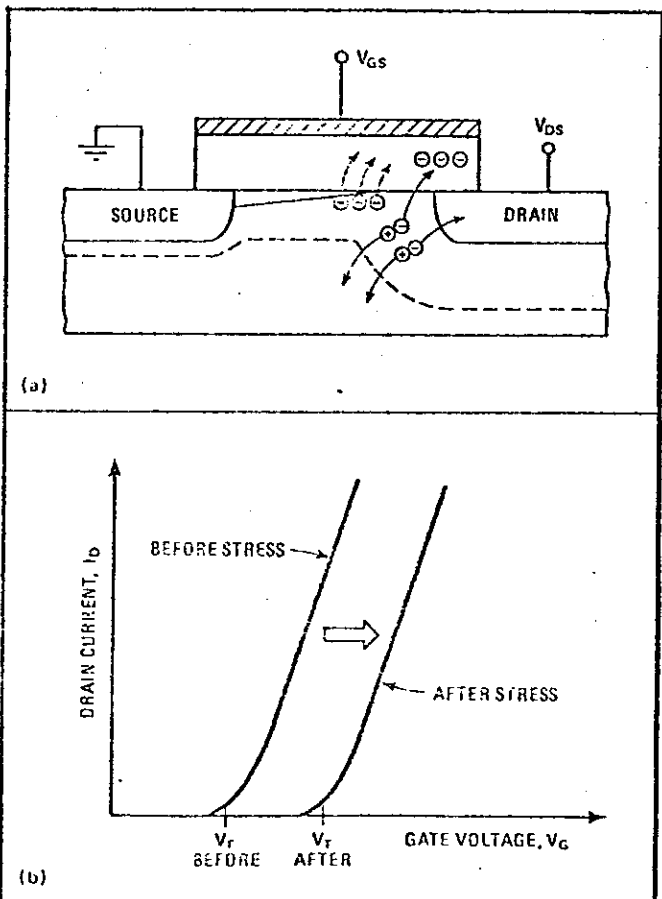
Its ability to be upgraded is a final and very important feature of H-MOS. Indeed, H-MOS is only the first step in that direction. As advances in photolithography occur, direct scaling can be applied to improve speed-power product and density even further.

Beyond first-order theory

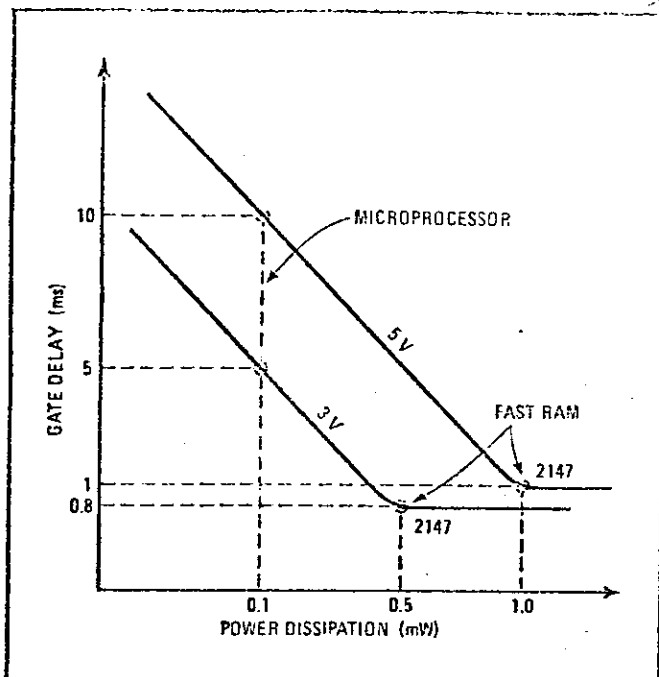
As devices are shrunk, scaling theory says ideally they should maintain the same qualitative characteristics. But in reality, second-order phenomena become quite significant. Some of these phenomena affect the circuit design, while others relate to reliability, but all have to be considered and understood to assure that H-MOS is a useful and safe process. Basically, all of the second-order



6. Guaranteeing punch-through. In short-channel MOS devices, the punch-through voltage falls to levels that could cause high leakage and circuit problems. The answer is to keep the channel length in the 4- μm region and to see that the gate oxide is thin.



7. Effect of trapped electrons. So much charge can be trapped in the gate oxide of short-channel devices (a) that it may cause a permanent shift in the threshold voltage (b). This shift could cause a reliability problem in these devices, but it can be minimized by very careful oxide processing and by reducing the supply voltage.

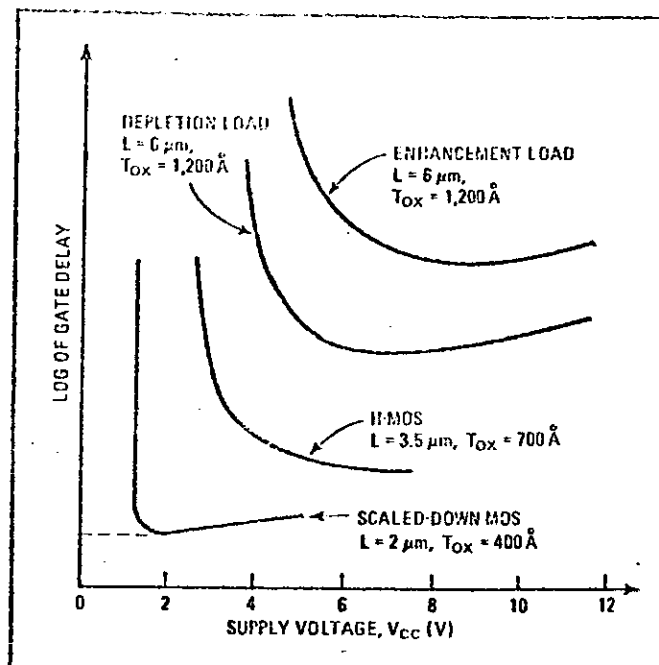


8. A change in supply voltage. Lowering the supply voltage from 5 to 3 V significantly enhances the speed and power dissipation of MOS circuits, especially scaled-down devices. The effect is most noticeable for microprocessors, where a 2-V reduction in supply voltage causes a doubling in performance.

effects arise for two reasons. First, as dimensions are shrunk while a constant (5-v) supply voltage is maintained, the average electric field is increased and this field activates many second-order effects. Second, the edges of a small device are so close together that the nonideal electric fields at these edges significantly affect its performance.

One second-order effect affects threshold voltage, which for the smallest geometries becomes a noticeable function of device size. As Fig. 4a shows, for the shortest channels, those less than $5 \mu\text{m}$ long, the junction depletion regions around the source and drain tend to support part of the ionized impurity charge that the gate voltage would otherwise maintain, and this reduces the threshold voltage. On the other hand, for the narrowest channels (Fig. 4b), additional ionized charge created by the fringing electric fields near the device edges tends to increase the threshold voltage. These effects will make small devices somewhat more sensitive to process control.

Moreover, in short-channel devices, the isolation characteristics between source and drain could also affect operation. With the gate and source of an enhancement device grounded, the drain must be able to stand off a certain positive voltage and still maintain a nominal amount of leakage. In high-performance transistors, this leakage current may have several causes, but for H-MOS the limiting factor is the phenomenon called second-gate punch-through (Fig. 5). In it, the electric field lines emanating from the drain junction terminate at the oxide-silicon interface of the channel. There the drain acts as an unwanted second gate and inverts the channel from the back, making the device more sensitive to punch-through effects.



9. Benefiting performance. As MOS technology becomes better and devices smaller, the need for lower supply voltages becomes more urgent. For 2-micrometer channel lengths, a 3-V supply yields the best gate-delay performance and process reliability.

The problem for short-channel devices is that the punch-through voltage arising from this effect is a linear function of the channel length (Fig. 6). The shorter the channel, the lower the voltage causing punch-through and therefore the more susceptible is the device to leakage. In H-MOS however, this is overcome by maintaining a long enough channel length and reducing the oxide thickness, since a thinner oxide prevents unwanted inversions by capacitively coupling the surface potential more tightly to the grounded gate electrode. A second punch-through effect occurs, as shown in Fig. 4b, when the electric field from the drain reaches through to the source and forward-biases the junction, causing current to flow—it is similar to that in a bipolar transistor; but again this punch-through voltage, which is proportional to L^2 , is a limiting factor only for devices smaller than those that are being used at present in H-MOS designs.

Impact ionization

Another source of leakage is impact ionization, the effects of which are illustrated in Fig. 7a. At a very large drain voltage of around 20 v, the junctions avalanche for all channel lengths greater than about $4 \mu\text{m}$. But even at the significantly lower (5-v) drain voltages of H-MOS, weak impact ionization can occur when current is flowing through the device channel. Activated by the high electric fields, impact ionization creates a population of electrons and holes with energies much higher than the normal channel electrons. The holes flow into the substrate and place a small load on the back-bias supply. Some of the electrons have enough energy to be injected into the gate oxide, as shown, where they can cause a gate current or be trapped. These trapped electrons cause a shift in the threshold voltage (Fig. 7b)—a

Where H-MOS excels

Three metal-oxide-semiconductor approaches to high performance are H-MOS, V-grooved MOS, and silicon on sapphire. As the accompanying table shows, the current versions of H-MOS and V-MOS both yield a speed-power product of about 1 picojoule.

V-MOS, in principle, has a slightly better packing density but pays for this compactness with a more complex process. Also, V-MOS yields an asymmetric device that must be used in one direction only, so that large-scale-integrated logic configurations are much more difficult to achieve than with H-MOS.

SOS, on the other hand, has the best speed-power product. But it requires a substrate five to seven times more costly and seems justified only for microprocessor applications, which do not require operation at the high-speed end of the speed-power curve.

The main advantage H-MOS has over V-MOS today is the fact that the scaling-down process moves it directly to higher performance and greater density at lower cost. The performance for 1980 scaled-down MOS (2-micrometer channels) is shown—it is about five times better than today's technology.

Parameter	H-MOS 1977	Scaled-down H-MOS 1980+	V-MOS 1977	SOS 1977
Layout density (gates/mm ²)	170	200	~220	150
Speed-power product (pJ)	1	0.2	~1	0.2
Gate delay (ns)	1	0.4	~1	0.5
Number of thin films	2	2	3	3
Number of implants	3	3	3	2

shift that could pose a reliability problem with channels less than 4 μm long.

Finally, there is the increase in interconnect capacitances induced by fringing fields. This parasitic effect occurs for some of the same reasons as the increase in threshold voltage associated with narrow channels—a narrow metal line over the large silicon ground plane has a larger effective area and therefore a larger parasitic capacitance from the fringing fields near its edges.

Happily, the only potential reliability problem brought out by the second-order theory—the trapping of injected electrons in the gate oxide—turns out not to affect H-MOS in its present form. True, trapped electrons tend to increase the threshold voltage, and an increase in threshold voltage could degrade circuit speed or totally stop it from functioning. But accelerated stress tests on H-MOS memory circuits reveal no signs of degradation [*Electronics*, Aug. 4, p. 103]. In fact, additional measurements on individual transistors, plus a physical model for electron injection, show that H-MOS devices will have a total threshold shift of less than 0.1 v after 10 years of continuous stress at worst-case conditions (at 0°C and a V_{DS} of 5.5 v). Careful processing of the gate oxide partly

accounts for these excellent results, for it minimizes the density of trapping sites for electrons. Moreover, it is worth noting that the gate oxides of these 5-v H-MOS devices are subjected to less stress than are the oxides of today's 12-v dynamic RAMs, since 5 v across an oxide 700 angstroms thick is less of an electric field than 12 v across the standard 1,000-angstrom oxides.

While the scaling down of devices as used in H-MOS for boosting MOS performance has a bright future, one condition must be met if its full potential is to be realized. That condition is a reduction in power supply voltage. Table 2 shows that if the technique is to work at all, the supply voltages for 1980 2- μm devices must be scaled down to the 2-to-4-v range. Since all supply voltages are now maintained at least at the 5-v TTL level, this lower supply-voltage requirement for future MOS devices must be accepted by integrated-circuit users.

There is, of course, an alternative for users who simply refuse to accept low system supply voltages. They could use converters to translate between the lower chip-voltage levels and the higher TTL input/output levels, or they could use two power supplies, one providing the chip's internal circuitry with 2 to 4 v and the second supplying 5 v to their I/O circuits. But either procedure is makeshift at best.

Key to the future: lower supply voltage

The fact is, a lower power supply voltage significantly increases the reliability of small-pattern devices, while at the same time increasing their performance remarkably. Reliability goes up because lower supply voltages entail greater tolerance to lower punch-through voltage and at the same time yield weaker electric fields in the channel region. This second effect reduces the risk that charge will be trapped in the gate oxide and alters the long-term stability of the device.

The increase in performance is even more striking. For RAMs and microprocessors (Fig. 8), the impact on the speed-power product would alone make it worth while going from a 5-v to a 3-v supply voltage. For RAMs, which operate at the saturation point of the speed-power curve, the lower power-supply voltage reduces power dissipation by about 60%, while maintaining the speed at the same high value. This reduced power dissipation becomes extremely important as the chip density goes up—65,536 bits and 262,144 bits—since it is generally agreed that for reliable operation power dissipation per package must be kept below a watt.

As for microprocessors, since they operate in a region that is well removed from the saturation point of the speed-power curve, they can take full advantage of a significant speed increase for a given power dissipation. A 3-v microprocessor chip, for example, will operate at twice the speed of a 5-v device.

How the supply voltage affects the various MOS processes that have evolved over the years is shown in Fig. 9, and again the desirability of lower voltages becomes evident. Indeed, for the 1980 scaled version of MOS devices—channel lengths of 2 μm and oxide thicknesses of 400 angstroms—a power supply of 2 to 4 v will give half the gate delay of today's H-MOS process operating at 5 v.

1978 Process Parameters (old data)

$$\lambda = 3 \mu\text{m}$$

$$L = 2\lambda = 6 \mu\text{m} \text{ gate length}$$

$$\tau = 0.3 \text{ nsec (or slower)}$$

Resistance, Ω/\square

Metal 0.1

Diff 10

Poly 40

Channel 20K ($V_{GS} - V_{TH} \approx 4V$)

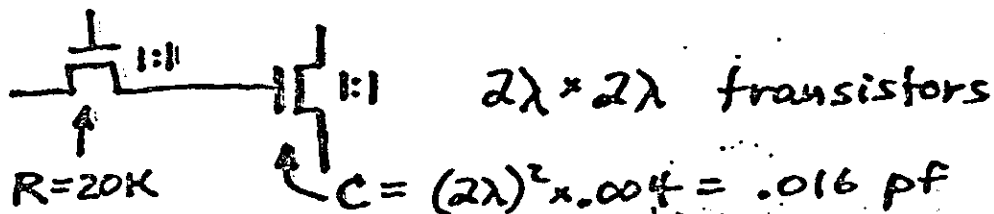
Capacitance pF/λ^2 ($\lambda^2 \approx 10 \mu\text{m}^2$)

Metal .0003

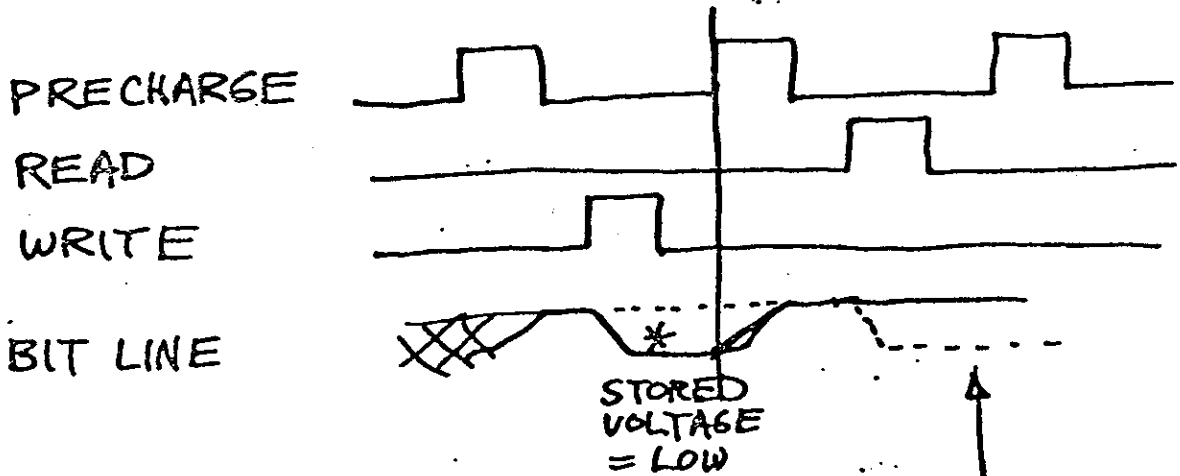
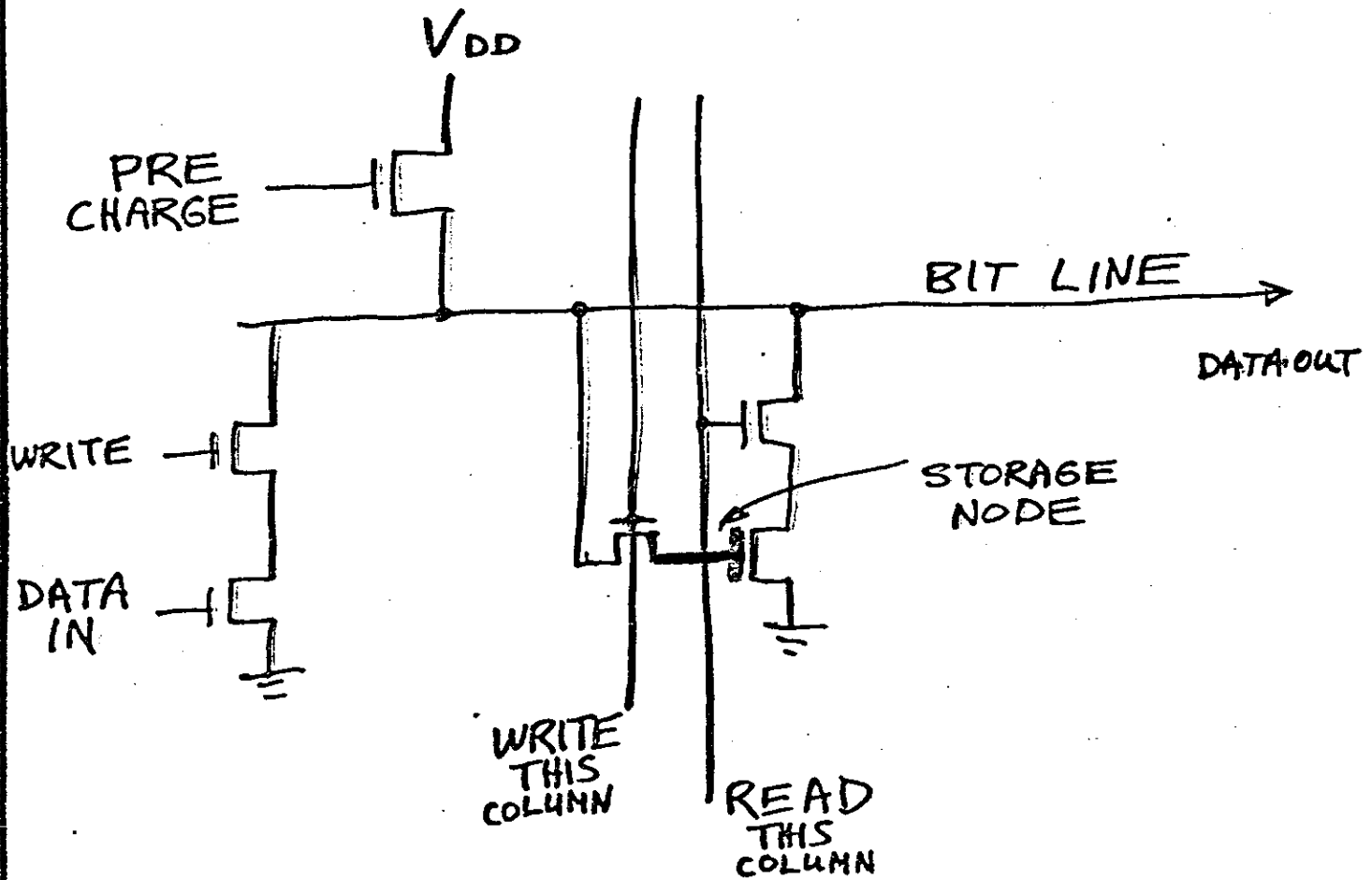
Diff .0008

Poly .0004

Gate .004



$$RC = \tau = 20K \times 0.16 \text{ pF} = 0.32 \text{ nsec.}$$



EXAMPLE:
DATA IN = 1
DATA OUT = 1

DATA IN = 0
DATA OUT = 0

Write

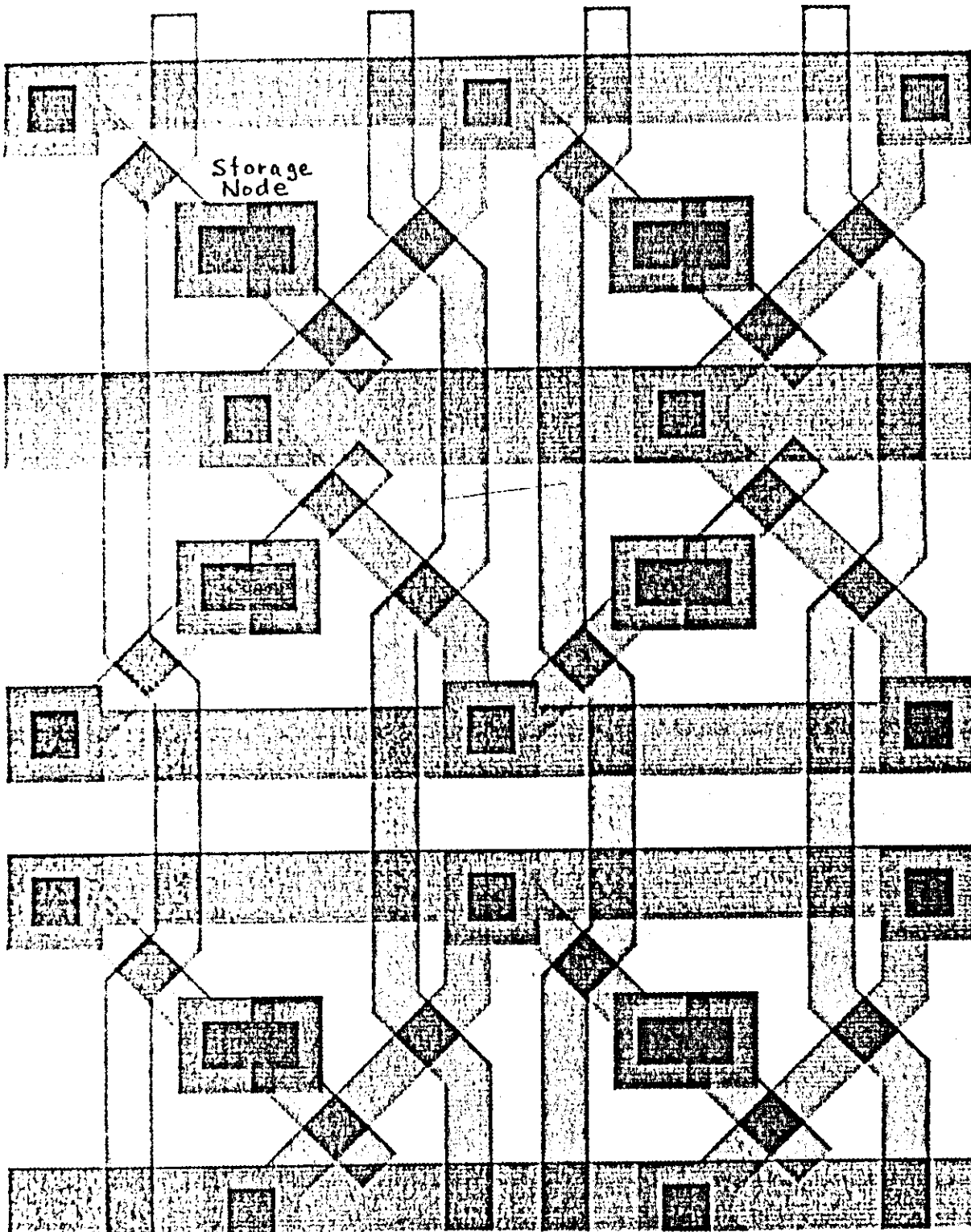
Read

Top
Bit

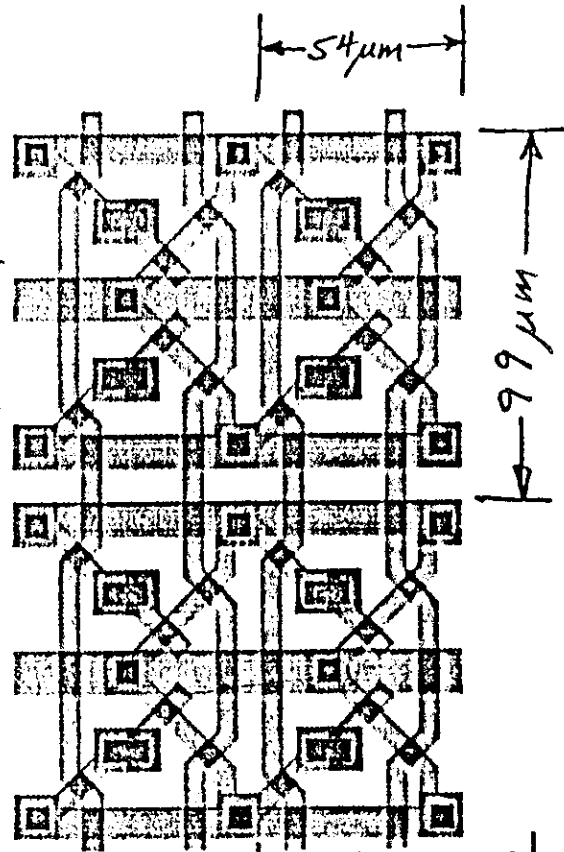
Storage
Node

Ground

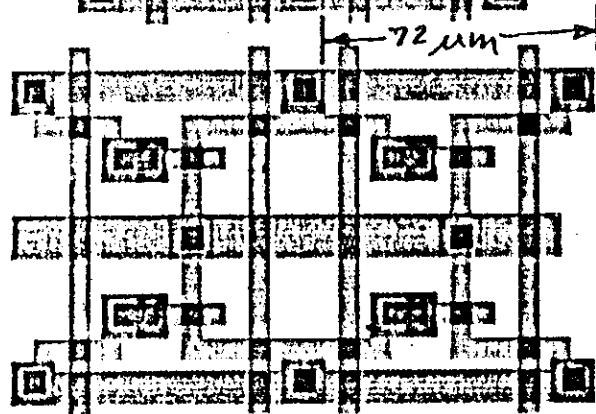
Bottom
Bit



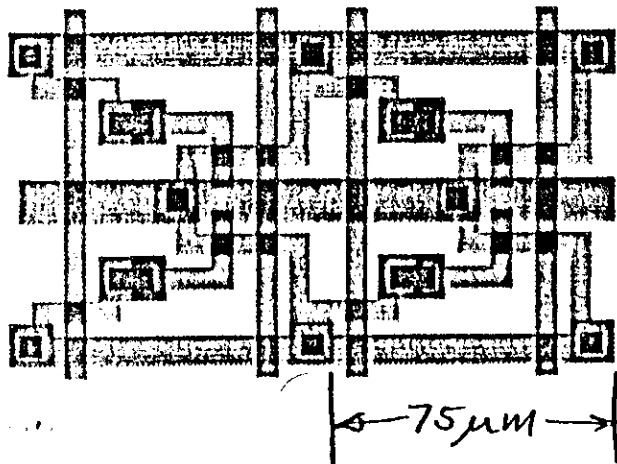
The central cell:
A pair of 3-transistor
Ram Cells, optimized
with 45° lines.



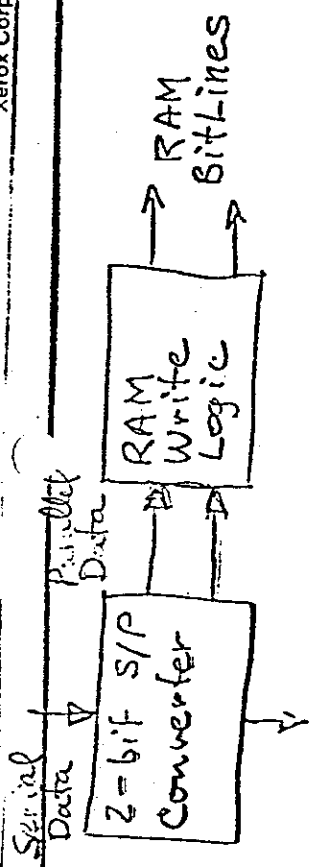
The best layout
of the same cell
with orthogonal
features.



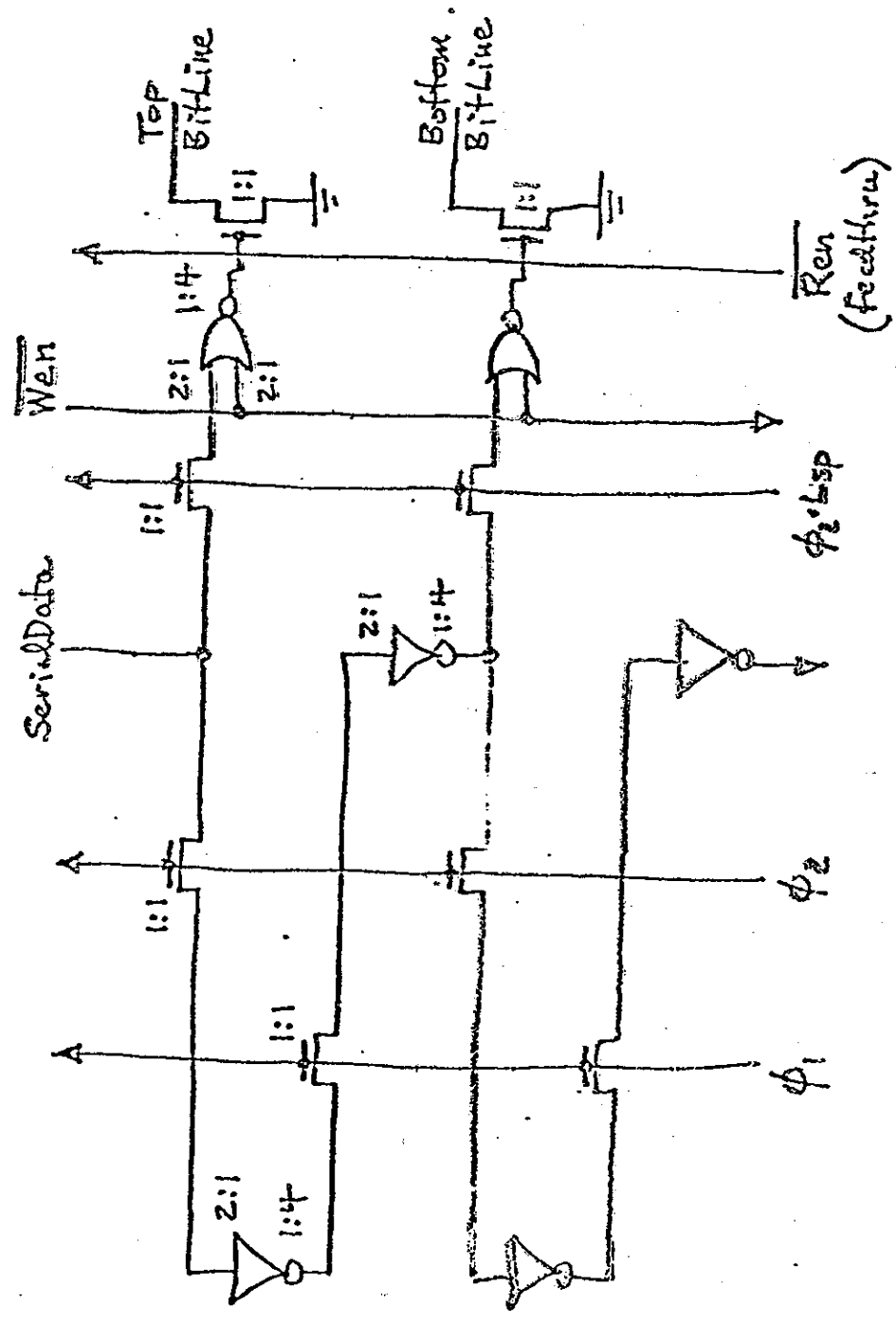
Another attempt



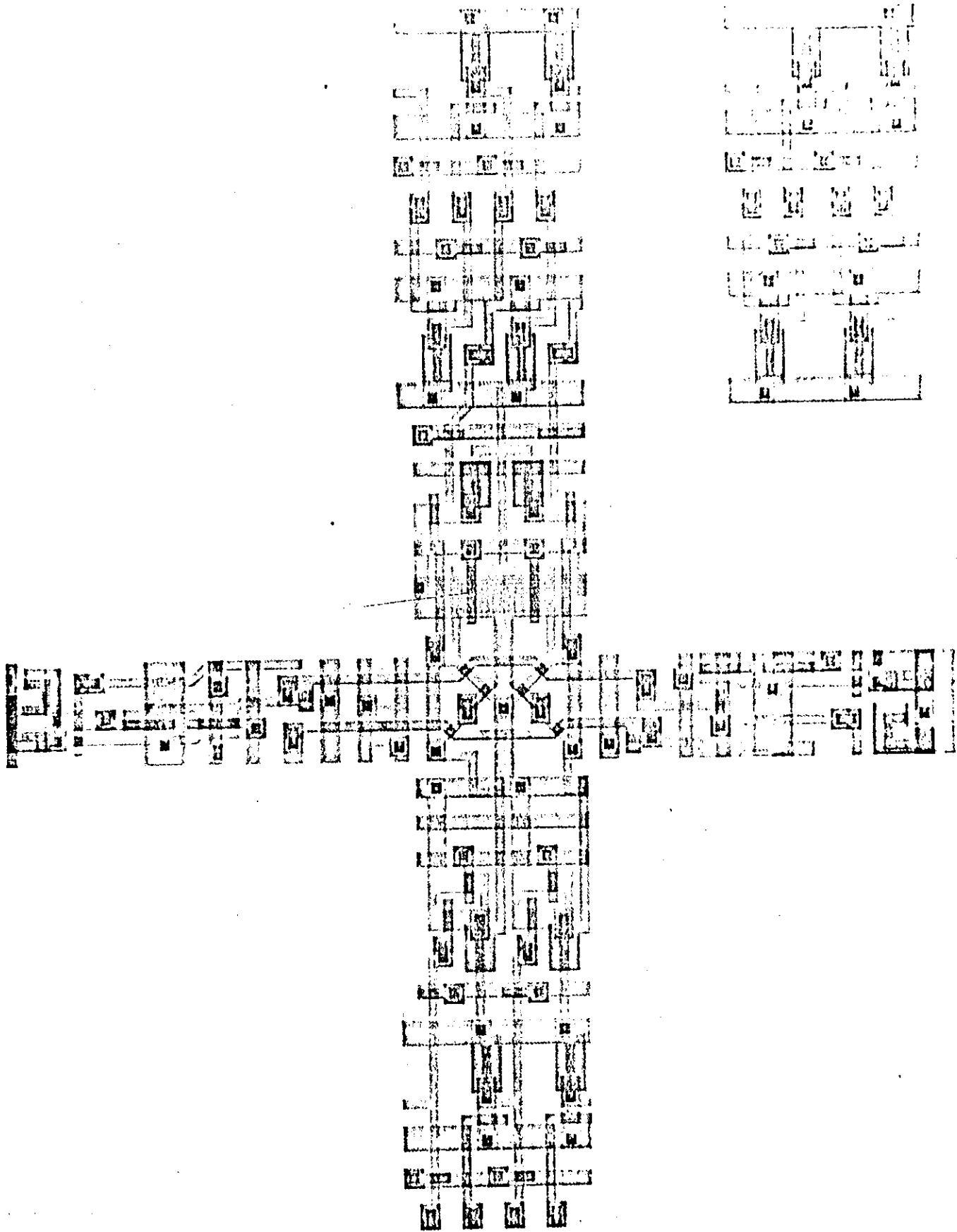
Block Diagram:

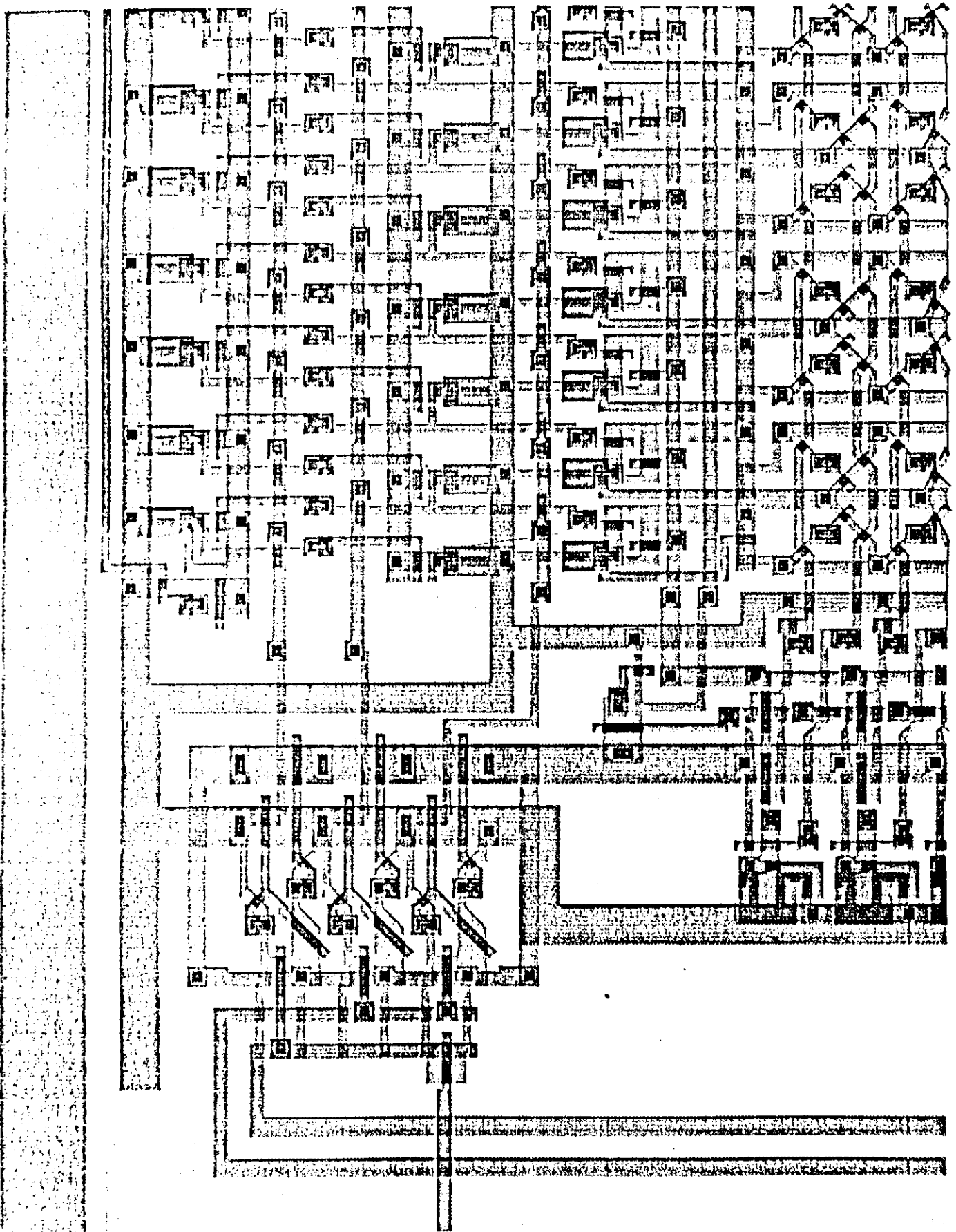


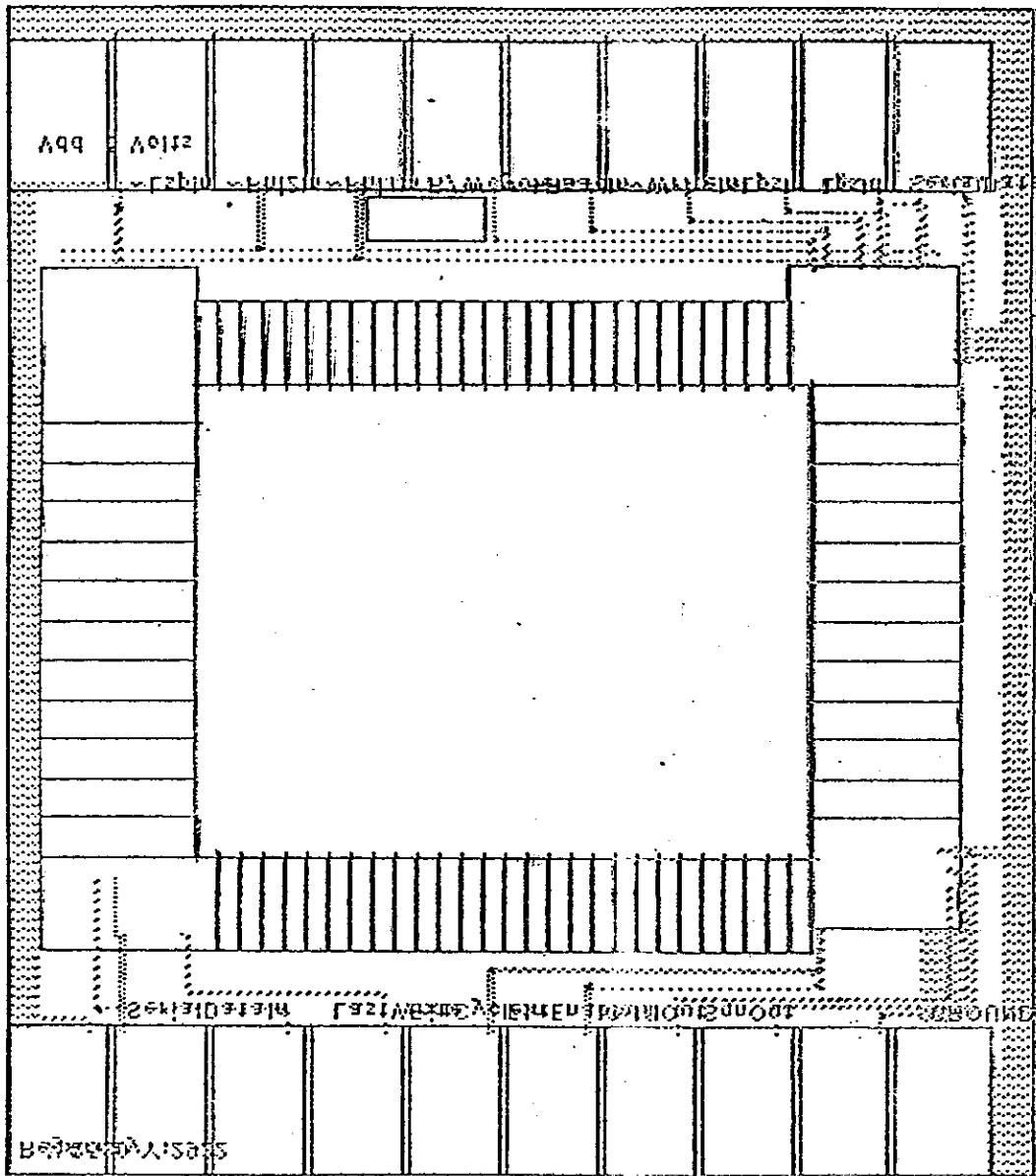
Logic Diagram:

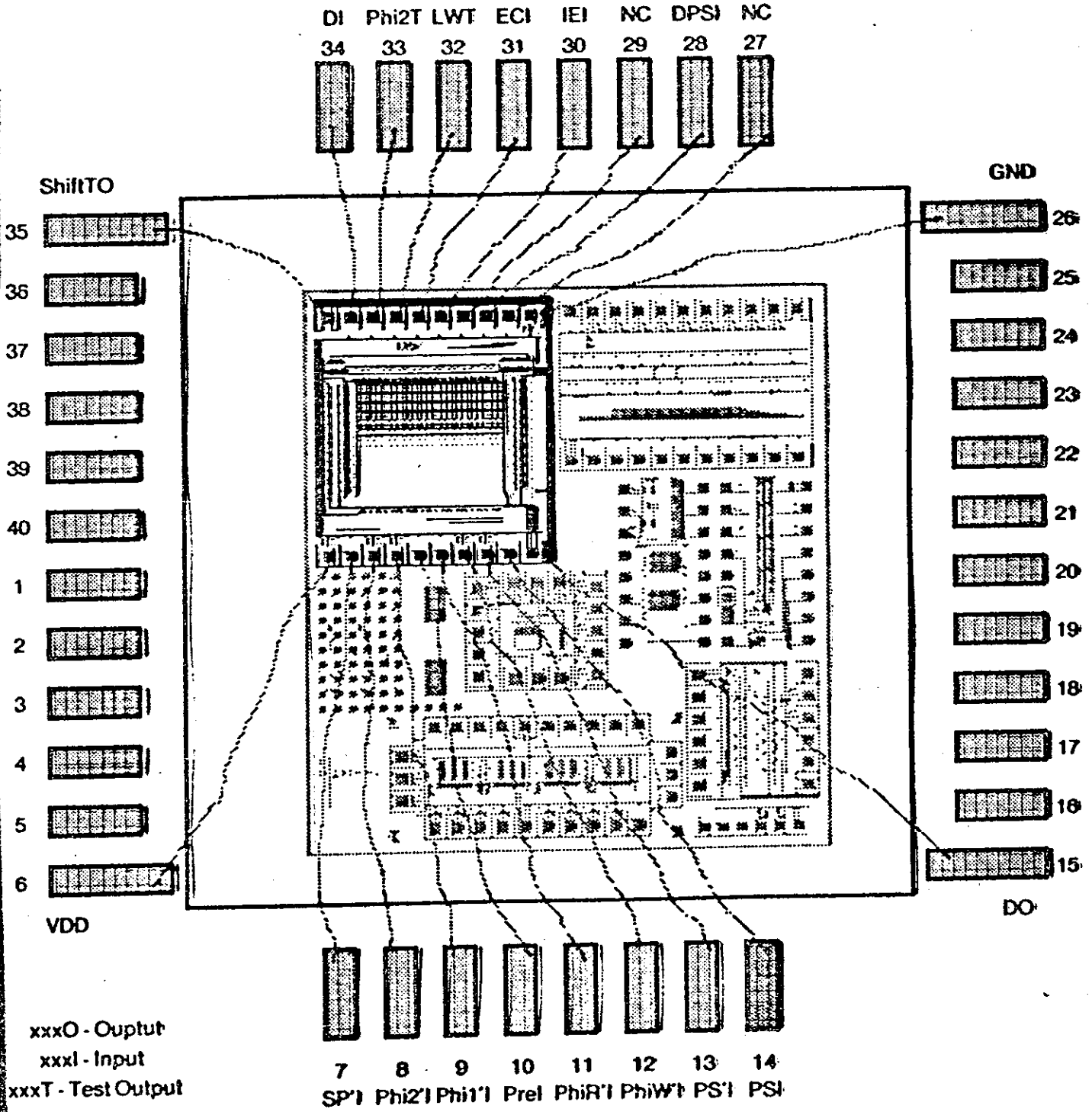


A typical cell design: Ram Reg In.
 Note: Mixed notations, topologically correct.









TODAY'S SCHEDULE

1. QUESTIONS
2. FOLLOW-ON SEMINAR SERIES
3. SYMBOLS IN ICARUS
4. PROJECT SCHEDULE
5. MORE SOFTWARE TOOLS
6. IC TESTING AND TESTABILITY
7. SCALING
8. CIRCUIT SIMULATION
9. ADVANCED DESIGN FACILITY
11. COURSE SUMMARY - LYNN CONWAY

ADVANCED LSI DESIGN SEMINAR

1. WEDNESDAY MORNINGS OK?

2. SPEAKERS FROM

- PARC (ALL OF YOU)
- XEROX / EL SEGUNDO
- UNIVERSITIES
- INDUSTRY

SYMBOLS IN ICARUS

1. ALLOW CELLS TO BE REPEATED EASILY IN X, Y ARRAY
2. SAVES MEMORY SPACE
3. KEEPS YOU FROM UNINTENTIONALLY MODIFYING A DESIGN
4. EASES MANIPULATION OF LARGE AMOUNTS OF DATA.

NOTE:

- CAN BE NESTED TO ANY LEVEL
- CAN BE EXPANDED, MODIFIED AND REDEFINED WITH SAME NAME OR DIFFERENT NAME.

PROJECT NOTES:

1) BASIC PROJECT STEPS:

- PICK PROJECT
- OVERALL SYSTEM PLAN
- DEFINE CELL TYPES AND FLOOR PLAN.
- STICK DIAGRAM CELLS
- CHECK!
- LAYOUT CELLS W/ ICARUS
- CHECK CELL DESIGN RULES AND FUNCTION!
- ASSEMBLE FINAL DESIGN
- CHECK!
- SEND DESIGN TO U.S.

WHAT OTHER TOOLS ARE AVAILABLE?

- 1) STANDARD I/O PADS AND PLAS ARE ON [IVY] <ICARUS>
- 2) DESIGN RULE CHECKING PROGRAMS
 - DRC.IMAGE (CHECKS SMALL CELL)
 - DRC RUNNING ON SIGMA IN EL SEGUNDO FOR LARGE DESIGNS
- 3) ICL - A GROUP OF BCPL ROUTINES WHICH CREATE ITEMS AND CELLS - CAN MAKE USE OF CELLS PREVIOUSLY DEFINED IN ICARUS (NEEDS DOC. AND CLEAN UP)
- 4) SCALING PROGRAM.

TIPS ON TESTING:

- 1) ALLOW BONDING (PROBING) OF TEST POINTS
- 2) COUNT ON DOING TESTING ONLY FROM PACKAGE PINS
- 3) SHIFT REGISTERS?
- 4) PLACE TEST CELLS WHERE THEY CAN BE TESTED INDEPENDENTLY OF OTHER CELLS
- 5) REGULAR STRUCTURES ARE EASIER TO TEST.

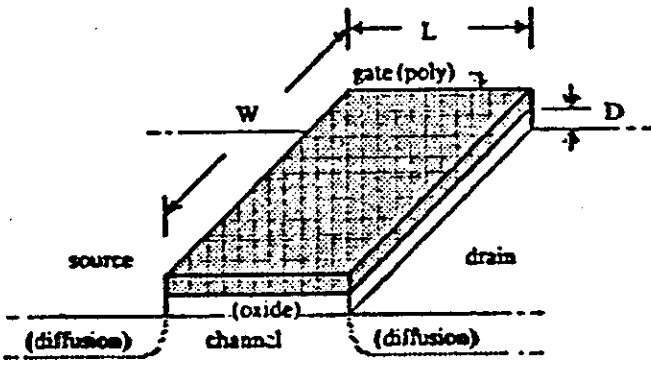


Fig. 14a. MOSFET, 1978

$$W' = W/\alpha$$

$$L' = L/\alpha$$

$$D' = D/\alpha$$

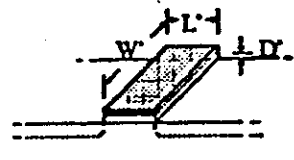


Fig. 14b. MOSFET Scaled
Down by Alpha, 19XX

SCALING:

WHY? - FASTER, DENSER, LOWER POWER

WHAT? - SCALE λ BY $1/\alpha$

$$1) \quad W' = W/\alpha \quad ; \quad L' = L/\alpha \quad ; \quad D' = D/\alpha$$

$$2) \quad V_{DD}' = V_{DD}/\alpha \quad (\text{SCALE P.S.})$$

$$V_{TH}' = V_{TH}/\alpha \quad \text{THRESHOLD VOLTAGE}$$

WHAT HAPPENS TO τ , C_g , I_{ds} ...

$$\tau' = RC'_g = \frac{L'^2}{\mu (V'_{gs} - V'_{TH})}$$

$$= \frac{(L/\alpha)^2}{\mu \frac{(V_{gs} - V_{TH})}{\alpha}} = \tau/\alpha$$

\Rightarrow

DELAYS DECREASE LINEARLY

W/α .

• GATE CAPACITANCE

$$C_g = \frac{\epsilon W L}{D}$$

$$C'_g = \frac{\epsilon (W/\alpha) (L/\alpha)}{D/\alpha}$$

$$= \frac{1}{\alpha} C_g$$

- ⇒ GATE CAP DECLINES BY $1/\alpha$
- ⇒ PARASITIC CAP " " $\sim 1/\alpha$
- ⇒ PARASITIC R INCREASES

• DRAW CURRENT

$$I_{DS} = \frac{\mu \epsilon W}{L D} (V_{GS} - V_{TH})(V_{DS})$$

$$I_{DS} \propto \frac{W V^2}{L D}$$

$$I'_{DS} \propto \frac{W V^2 / \alpha^3}{L D / \alpha^3} = \frac{1}{\alpha} I_{DS}$$

- ⇒ CURRENT / TRANS DECLINES BY $1/\alpha$.

SUMMARY

$$\text{DELAY} = 1/\alpha$$

$$\text{AREA/FUNCTION} = 1/\alpha^2$$

$$\text{POWER/FUNCTION} = 1/\alpha^2$$

NOTE :

CURRENT / CHIP SCALES

UP BY α . THEREFORE

HIGHER CURRENT DENSITIES

CIRCUIT SIMULATION : SPICE

- NORMALLY USED TO CHECK OR VERIFY THE SPEED PERFORMANCE OF A SMALL PART OF DESIGN
- PROCESS PARAMETERS ARE PRE-SPECIFIED FOR AN INDIVIDUAL PROCESS - USER NEED NOT SPECIFY THEM
- USER DOES FURNISH
 - CIRCUIT DESCRIPTION
 - TRANSISTOR RATIOS
 - STRAY CAPACITANCE
 - EXCITATION AND PLOTTING INFO.

SPICE SIMULATION OF 4-INPUT NAND GATE :

- SIMULATED UNDER 3 CONDITIONS

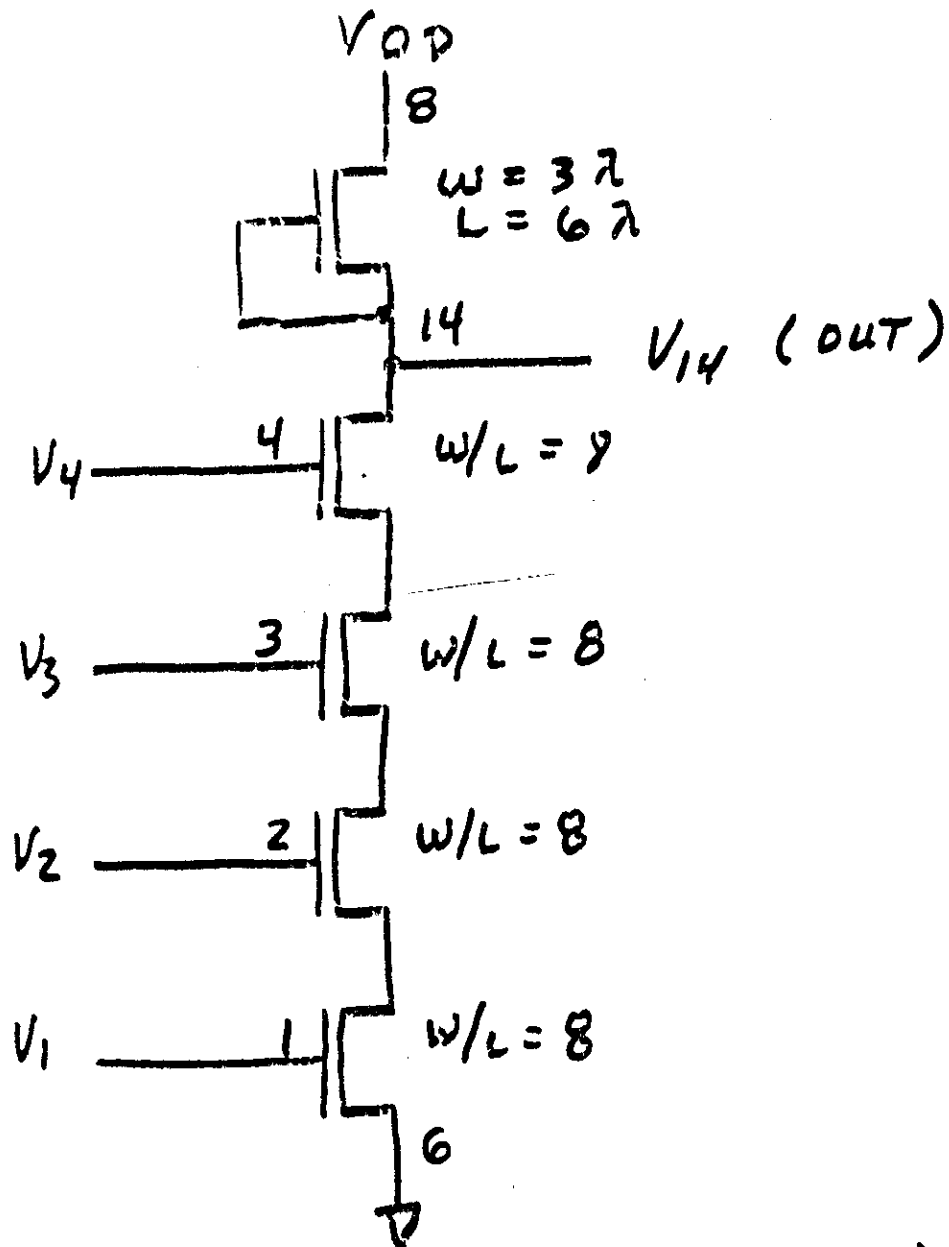
$\lambda = 3$ (NSIL I)

$\lambda = 2$ (NSIL II) PARTIAL SCALING

$\lambda = 2$ TRUE SCALING

- NSIL II DOES NOT SCALE V_{TH} OF DEP. LOAD (YELLOW) AND DOES IT SCALE POWER SUPPLY
- ABOVE FACT MEANS POWER/DEVICE IS INCREASING INSTEAD OF DECREASING BY α

CIRCUIT SIMULATED BY SPICE



	VDD	I_{D3}	P(mW)	T_R (V1)
NSIL I	5	39 μ a	.19	38 ns
NSIL II	5	57 μ a	.28	18 ns
TRU SCALING	3.33	27 μ a	.09	28 ns

***** 2 Mar79 ***** XEROX PARC SPICE 2.1 *****00:03:50 *****

ANDGATE

INPUT LISTING TEMPERATURE = 25.000 DEG C

*** PRELIMINARY VERSION ***** REPORT PROBLEMS TO <DAVIES> ****

VDD 8 6 5.0

V1 1 6 PULSE 3.33 0 1N 4N 4N 150N 800N
 V2 2 6 PULSE 3.33 0 200N 4N 4N 150N 800N
 V3 3 6 PULSE 3.33 0 400N 4N 4N 150N 800N
 V4 4 6 PULSE 3.33 0 600N 4N 4N 150N 800N

M1 11 1 6 0 ENH1NMOS WOVERL=8
 M2 12 2 11 0 ENH1NMOS WOVERL=8
 M3 13 3 12 0 ENH1NMOS WOVERL=8
 M4 14 4 13 0 ENH1NMOS WOVERL=8
 M5 8 14 14 0 DEP1NMOS W=3 L=6

.TRAN 8NS 800NS
 .PLOT TRAN V(14,6) (-1,5)
 .WIDTH OUT=72
 .END

.....
 CIRCUIT ELEMENT SUMMARY TEMPERATURE = 25.000 DEG C

**** INDEPENDENT SOURCES

NAME	NODES	DC VALUE	AC VALUE	AC PHASE	TRANSIENT
VDD	8 6	5.00D+00	0.00D-01	0.00D-01	
V1	1 6	3.33D+00	0.00D-01	0.00D-01	PULSE
					INITIAL VALUE 3.33D+00
					PULSED VALUE 0.00D-01
					DELAY TIME... 1.00D-09
					RISETIME..... 4.00D-09
					FALLTIME..... 4.00D-09
					WIDTH..... 1.50D-07
					PERIOD..... 8.00D-07
V2	2 6	3.33D+00	0.00D-01	0.00D-01	PULSE
					INITIAL VALUE 3.33D+00
					PULSED VALUE 0.00D-01
					DELAY TIME... 2.00D-07
					RISETIME..... 4.00D-09
					FALLTIME..... 4.00D-09
					WIDTH..... 1.50D-07
					PERIOD..... 8.00D-07
V3	3 6	3.33D+00	0.00D-01	0.00D-01	PULSE

0 INITIAL VALUE 3.33D+00
 PULSED VALUE 0.00D-01
 DELAY TIME... 4.00D-07
 RISETIME..... 4.00D-09
 FALLTIME..... 4.00D-09
 WIDTH..... 1.50D-07
 PERIOD..... 8.00D-07

V4 4 6 3.33D+00 0.00D-01 0.00D-01 PULSE
 0 INITIAL VALUE 3.33D+00
 PULSED VALUE 0.00D-01
 DELAY TIME... 6.00D-07
 RISETIME..... 4.00D-09
 FALLTIME..... 4.00D-09
 WIDTH..... 1.50D-07
 PERIOD..... 8.00D-07

**** MOSFETS

NAME	D	G	S	B	MODEL	(>1--GRID UNITS		<1--METERS)		SQD	SQS
						L	W	DDPTH	SDPTH		
M1	11	1	6	0	ENH1NMOS	2.000000	16.000000	4.000000	4.000000	0.0	0.0
M2	12	2	11	0	ENH1NMOS	2.000000	16.000000	4.000000	4.000000	0.0	0.0
M3	13	3	12	0	ENH1NMOS	2.000000	16.000000	4.000000	4.000000	0.0	0.0
M4	14	4	13	0	ENH1NMOS	2.000000	16.000000	4.000000	4.000000	0.0	0.0
M5	8	14	14	0	DEP1NMOS	6.000000	3.000000	4.000000	4.000000	0.0	0.0

 MOSFET MODEL PARAMETERS: XSIM TEMPERATURE = 25.000 DEG C

---- LIBRARY WAS INSTALLED: 10FEB 79 VERSN 1 ----

TYPE	DIMENSION	ENH1NMOS DEP1NMOS	
		NMOS	NMOS
VTO	V	0.900	-3.000
KP	MHO/V	2.00D-05	2.40D-05
XK1	V**1/2	0.100	0.100
PHIF2	V	0.750	0.750
ETA0		0.950	0.950
RESGAT	OHM/SQ	20.000	20.000
RESSD	OHM/SQ	10.000	10.000
CGS	F/M	3.60D-10	3.60D-10
CGD	F/M	3.60D-10	3.60D-10
CGBLK	F/M**2	4.20D-05	4.20D-05
CBOTOM	F/M**2	7.70D-05	7.70D-05
CPERIF	F/M	7.87D-10	7.87D-10
TOX	M	9.70D-08	9.70D-08
VFB	V	0.063	-3.837
LD	M	3.00D-07	3.00D-07
WD	M	3.00D-07	3.00D-07
GRID	M	3.00D-06	3.00D-06
UOBRK	1/V	0.100	0.100

 INITIAL TRANSIENT SOLUTION TEMPERATURE = 25.000 DEG C

NODE	VOLTAGE	NODE	VOLTAGE	NODE	VOLTAGE	NODE	VOLTAGE
(1)	3.2097	(2)	3.2097	(3)	3.2097	(4)	3.2097
(6)	-0.1203	(8)	4.8797	(11)	-0.0037	(12)	0.1179
(13)	0.2457	(14)	0.3808				

VOLTAGE SOURCE CURRENTS

NAME	CURRENT
VDD	-3.905D-05
V1	0.000D-01
V2	0.000D-01
V3	0.000D-01
V4	0.000D-01

TOTAL POWER DISSIPATION 1.95D-04 WATTS

 OPERATING POINT INFORMATION TEMPERATURE = 25.000 DEG C

**** MOSFETS

	M1	M2	M3	M4	M5
MODEL	ENH1NMOS	ENH1NMOS	ENH1NMOS	ENH1NMOS	DEP1NMOS
ID	3.90D-05	3.90D-05	3.90D-05	3.90D-05	3.90D-05
VGS	3.330	3.213	3.092	2.964	0.000
VDS	0.117	0.122	0.128	0.135	4.499
VBS	0.120	0.004	-0.118	-0.246	-0.381

 TRANSIENT ANALYSIS TEMPERATURE = 25.000 DEG C

X	TIME	V(14.6)
X		-1.000D+00 5.000D-01 2.000D+00 3.500D+00 5.000D+00

0.0000-01 5.0110-01
 8.0000-09 4.3640-01
 1.6000-08 9.9790-01
 2.4000-08 1.4500+00
 3.2000-08 1.8580+00
 4.0000-08 2.6460+00
 4.8000-08 3.7950+00
 5.6000-08 4.5660+00
 6.4000-08 4.7720+00
 7.2000-08 4.8360+00
 8.0000-08 4.8810+00
 8.8000-08 4.9080+00
 9.6000-08 4.9270+00
 1.0400-07 4.9410+00
 1.1200-07 4.9510+00
 1.2000-07 4.9590+00
 1.2800-07 4.9650+00
 1.3600-07 4.9690+00
 1.4400-07 4.9730+00
 1.5200-07 4.9760+00
 1.6000-07 4.3760+00
 1.6800-07 6.6330-01
 1.7600-07 4.6820-01
 1.8400-07 4.8320-01
 1.9200-07 5.0820-01
 2.0000-07 4.6840-01
 2.0800-07 5.3670-01
 2.1600-07 1.3020+00
 2.2400-07 2.3760+00
 2.3200-07 4.0850+00
 2.4000-07 4.9870+00
 2.4800-07 4.9590+00
 2.5600-07 4.9780+00
 2.6400-07 4.9830+00
 2.7200-07 4.9870+00
 2.8000-07 4.9900+00
 2.8800-07 4.9920+00
 2.9600-07 4.9940+00
 3.0400-07 4.9950+00
 3.1200-07 4.9960+00
 3.2000-07 4.9960+00
 3.2800-07 4.9970+00
 3.3600-07 4.9970+00
 3.4400-07 4.9980+00
 3.5200-07 4.9980+00
 3.6000-07 4.2190+00
 3.6800-07 4.9290-01
 3.7600-07 5.3960-01
 3.8400-07 4.5760-01
 3.9200-07 5.3120-01
 4.0000-07 4.6490-01
 4.0800-07 6.0760-01
 4.1600-07 1.7970+00
 4.2400-07 3.6440+00
 4.3200-07 5.0140+00
 4.4000-07 4.9860+00
 4.4800-07 4.9970+00
 4.5600-07 4.9950+00
 4.6400-07 4.9980+00
 4.7200-07 4.9980+00
 4.8000-07 4.9980+00
 4.8800-07 4.9980+00
 4.9600-07 4.9990+00
 5.0400-07 4.9990+00
 5.1200-07 4.9990+00
 5.2000-07 4.9990+00
 5.2800-07 4.9990+00
 5.3600-07 4.9990+00
 5.4400-07 4.9990+00
 5.5200-07 5.0000+00
 5.6000-07 3.4800+00
 5.6800-07 3.7140-01
 5.7600-07 5.0770-01
 5.8400-07 4.8420-01
 5.9200-07 5.0760-01
 6.0000-07 4.8890-01
 6.0800-07 1.6610+00

5.1200-07 4.9990+00
 5.2000-07 4.9990+00
 5.2800-07 4.9990+00
 5.3600-07 4.9990+00
 5.4400-07 4.9990+00
 5.5200-07 5.0000+00
 5.6000-07 3.4800+00
 5.6800-07 3.7140-01
 5.7600-07 5.0770-01
 5.8400-07 4.8420-01
 5.9200-07 5.0760-01
 6.0000-07 4.8890-01
 6.0800-07 1.6610+00

6.160D-07	4.522D+00
6.240D-07	5.060D+00
6.320D-07	4.988D+00
6.400D-07	5.003D+00
6.480D-07	4.999D+00
6.560D-07	5.000D+00
6.640D-07	5.000D+00
6.720D-07	5.000D+00
6.800D-07	5.000D+00
6.880D-07	5.000D+00
6.960D-07	5.000D+00
7.040D-07	5.000D+00
7.120D-07	5.000D+00
7.200D-07	5.000D+00
7.280D-07	5.000D+00
7.360D-07	5.000D+00
7.440D-07	5.000D+00
7.520D-07	5.000D+00
7.600D-07	1.490D+00
7.680D-07	5.930D-01
7.760D-07	4.801D-01
7.840D-07	5.009D-01
7.920D-07	4.947D-01
8.000D-07	4.964D-01

Y TOTAL JOB TIME 199.00

***** 2 Mar79 ***** XEROX PARC SPICE 2.1 *****00:03:50 *****

.END

INPUT LISTING TEMPERATURE = 25.000 DEG C

*** PRELIMINARY VERSION ***** REPORT PROBLEMS TO <DAVIES> ****

ERROR: .END CARD MISSING

***** 2 Mar79 ***** XEROX PARC SPICE 2.1 *****00:08:23 *****

ANDGATE

INPUT LISTING TEMPERATURE = 25.000 DEG C

*** PRELIMINARY VERSION ***** REPORT PROBLEMS TO <DAVIES> ****

```

.
VDD 8 6 5.0
VBB 6 0 2.5
.
V1 1 6 PULSE 3.33 0 1N 4N 4N 150N 800N
V2 2 6 PULSE 3.33 0 200N 4N 4N 150N 800N
V3 3 6 PULSE 3.33 0 400N 4N 4N 150N 800N
V4 4 6 PULSE 3.33 0 600N 4N 4N 150N 800N
.
M1 11 1 6 0 ENH2NMOS WOVERL=8
M2 12 2 11 0 ENH2NMOS WOVERL=8
M3 13 3 12 0 ENH2NMOS WOVERL=8
M4 14 4 13 0 ENH2NMOS WOVERL=8
M5 8 14 14 0 DEP2NMOS W=3 L=6
.
. TRAN 8NS 800NS
. PLOT TRAN V(14,6) (-1.5)
. WIDTH OUT=72
. END
    
```

.....
 CIRCUIT ELEMENT SUMMARY TEMPERATURE = 25.000 DEG C

**** INDEPENDENT SOURCES

NAME	NODES	DC VALUE	AC VALUE	AC PHASE	TRANSIENT
VDD	8 6	5.00D+00	0.00D-01	0.00D-01	
VBB	6 0	2.50D+00	0.00D-01	0.00D-01	
V1	1 6	3.33D+00	0.00D-01	0.00D-01	PULSE
					INITIAL VALUE 3.33D+00
					PULSED VALUE 0.00D-01
					DELAY TIME... 1.00D-09
					RISETIME..... 4.00D-09
					FALLTIME..... 4.00D-09
					WIDTH..... 1.50D-07
					PERIOD..... 8.00D-07
V2	2 6	3.33D+00	0.00D-01	0.00D-01	PULSE
					INITIAL VALUE 3.33D+00
					PULSED VALUE 0.00D-01
					DELAY TIME... 2.00D-07
					RISETIME..... 4.00D-09
					FALLTIME..... 4.00D-09
					WIDTH..... 1.50D-07
					PERIOD..... 8.00D-07
V3	3 6	3.33D+00	0.00D-01	0.00D-01	PULSE

0 INITIAL VALUE 3.33D+00
 PULSED VALUE. 0.00D-01
 DELAY TIME... 4.00D-07
 RISETIME..... 4.00D-09
 FALLTIME..... 4.00D-09
 WIDTH..... 1.50D-07
 PERIOD..... 8.00D-07

0 V4 4 6 3.33D+00 0.00D-01 0.00D-01 PULSE
 INITIAL VALUE 3.33D+00
 PULSED VALUE. 0.00D-01
 DELAY TIME... 6.00D-07
 RISETIME..... 4.00D-09
 FALLTIME..... 4.00D-09
 WIDTH..... 1.50D-07
 PERIOD..... 8.00D-07

**** MOSFETS

NAME	D	G	S	B	MODEL	(>1--GRID UNITS		<1--METERS)		SQD	SQS
						L	W	DEPTH	SDPTH		
M1	11	1	6	0	ENH2NMOS	2.000000	16.000000	4.000000	4.000000	0.0	0.0
M2	12	2	11	0	ENH2NMOS	2.000000	16.000000	4.000000	4.000000	0.0	0.0
M3	13	3	12	0	ENH2NMOS	2.000000	16.000000	4.000000	4.000000	0.0	0.0
M4	14	4	13	0	ENH2NMOS	2.000000	16.000000	4.000000	4.000000	0.0	0.0
M5	8	14	14	0	DEP2NMOS	6.000000	3.000000	4.000000	4.000000	0.0	0.0

 MOSFET MODEL PARAMETERS: XSIM TEMPERATURE = 25.000 DEG C

----- LIBRARY WAS INSTALLED: 10FEB 79 VERSN 1 -----

TYPE	DIMENSION	ENH2NMOS DEP2NMOS	
		NMOS	NMOS
VTO	V	0.500	-3.400
KP	MHO/V	2.90D-05	3.00D-05
XK1	V**1/2	0.100	0.100
PHIF2	V	0.750	0.750
ETA0		0.950	0.950
RESGAT	OHM/SQ	33.000	33.000
RESSD	OHM/SQ	25.000	25.000
CGS	F/M	1.70D-10	1.70D-10
CGD	F/M	1.70D-10	1.70D-10
CGBLK	F/M**2	4.20D-05	4.20D-05
CBOTOM	F/M**2	2.11D-04	2.11D-04
CPERIF	F/M	3.50D-10	3.50D-10
TOX	M	7.30D-08	7.30D-08
VFB	V	-0.337	-4.237
LD	M	3.50D-07	3.50D-07
WD	M	3.50D-07	3.50D-07
GRID	M	2.00D-06	2.00D-06
UOBRK	1/V	0.100	0.100

NODE	VOLTAGE	NODE	VOLTAGE	NODE	VOLTAGE	NODE	VOLTAGE
(1)	5.8300	(2)	5.8300	(3)	5.8300	(4)	5.8300
(6)	2.5000	(8)	7.5000	(11)	2.5984	(12)	2.7000
(13)	2.8050	(14)	2.9139				

VOLTAGE SOURCE CURRENTS

NAME CURRENT

VDD	-5.714D-05
VBB	-3.226D-11
V1	0.000D-01
V2	0.000D-01
V3	0.000D-01
V4	0.000D-01

TOTAL POWER DISSIPATION 2.86D-04 WATTS

.....
 OPERATING POINT INFORMATION TEMPERATURE = 25.000 DEG C

**** MOSFETS

	M1	M2	M3	M4	M5
MODEL	ENH2NMOS	ENH2NMOS	ENH2NMOS	ENH2NMOS	DEP2NMOS
ID	5.71D-05	5.71D-05	5.71D-05	5.71D-05	5.71D-05
VGS	3.330	3.232	3.130	3.025	0.000
VDS	0.098	0.102	0.105	0.109	4.586
VBS	-2.500	-2.598	-2.700	-2.805	-2.914

.....
 TRANSIENT ANALYSIS TEMPERATURE = 25.000 DEG C

X
 TIME V(14,6)

X	-1.000D+00	5.000D-01	2.000D+00	3.500D+00	5.000D+00
0.000D-01	4.139D-01
8.000D-09	9.081D-01
1.600D-08	2.219D+00
2.400D-08	4.112D+00
3.200D-08	5.102D+00
4.000D-08	4.955D+00
4.800D-08	4.986D+00
5.600D-08	4.985D+00
6.400D-08	4.991D+00
7.200D-08	4.993D+00
8.000D-08	4.995D+00
8.800D-08	4.996D+00
9.600D-08	4.997D+00
1.040D-07	4.998D+00
1.120D-07	4.998D+00
1.200D-07	4.998D+00
1.280D-07	4.999D+00
1.360D-07	4.999D+00
1.440D-07	4.999D+00
1.520D-07	4.999D+00
1.600D-07	1.988D+00
1.680D-07	2.643D-01
1.760D-07	3.980D-01
1.840D-07	4.220D-01
1.920D-07	4.076D-01
2.000D-07	3.951D-01
2.080D-07	1.299D+00
2.160D-07	3.632D+00
2.240D-07	5.100D+00
2.320D-07	4.969D+00
2.400D-07	4.994D+00
2.480D-07	4.994D+00
2.560D-07	4.997D+00
2.640D-07	4.998D+00
2.720D-07	4.998D+00
2.800D-07	4.999D+00
2.880D-07	4.999D+00
2.960D-07	4.999D+00
3.040D-07	4.999D+00
3.120D-07	4.999D+00
3.200D-07	5.000D+00
3.280D-07	5.000D+00
3.360D-07	5.000D+00
3.440D-07	5.000D+00
3.520D-07	5.000D+00
3.600D-07	9.396D-01
3.680D-07	2.999D-01
3.760D-07	4.106D-01
3.840D-07	4.130D-01
3.920D-07	4.155D-01
4.000D-07	3.946D-01
4.080D-07	2.049D+00
4.160D-07	4.809D+00
4.240D-07	5.083D+00
4.320D-07	4.968D+00
4.400D-07	5.015D+00
4.480D-07	4.993D+00
4.560D-07	5.003D+00
4.640D-07	4.999D+00
4.720D-07	5.001D+00
4.800D-07	5.000D+00
4.880D-07	5.000D+00
4.960D-07	5.000D+00
5.040D-07	5.000D+00
5.120D-07	5.000D+00
5.200D-07	5.000D+00
5.280D-07	5.000D+00
5.360D-07	5.000D+00
5.440D-07	5.000D+00
5.520D-07	5.000D+00
5.600D-07	8.219D-01
5.680D-07	3.294D-01
5.760D-07	4.117D-01
5.840D-07	4.130D-01
5.920D-07	4.151D-01

6.000D-07	3.980D-01
6.080D-07	4.707D+00
6.160D-07	5.019D+00
6.240D-07	4.991D+00
6.320D-07	5.004D+00
6.400D-07	4.998D+00
6.480D-07	5.001D+00
6.560D-07	5.000D+00
6.640D-07	5.000D+00
6.720D-07	5.000D+00
6.800D-07	5.000D+00
6.880D-07	5.000D+00
6.960D-07	5.000D+00
7.040D-07	5.000D+00
7.120D-07	5.000D+00
7.200D-07	5.000D+00
7.280D-07	5.000D+00
7.360D-07	5.000D+00
7.440D-07	5.000D+00
7.520D-07	5.000D+00
7.600D-07	6.093D-01
7.680D-07	3.743D-01
7.760D-07	4.132D-01
7.840D-07	4.135D-01
7.920D-07	4.144D-01
8.000D-07	4.074D-01

Y TOTAL JOB TIME 168.00

***** 2 Mar79 ***** XEROX PARC SPICE 2.1 *****00:08:23 *****

.END

INPUT LISTING TEMPERATURE = 25.000 DEG C

*** PRELIMINARY VERSION ***** REPORT PROBLEMS TO <DAVIES> ****

ERROR: .END CARD MISSING

***** 2 Mar79 ***** XEROX PARC SPICE 2.1 *****00:11:40 *****

ANDGATE

INPUT LISTING TEMPERATURE = 25.000 DEG C

*** PRELIMINARY VERSION ***** REPORT PROBLEMS TO <DAVIES> ****

VDD 8 6 3.33
VBB 6 0 2.5

V1 1 6 PULSE 3.33 0 1N 4N 4N 150N 800N
V2 2 6 PULSE 3.33 0 200N 4N 4N 150N 800N
V3 3 6 PULSE 3.33 0 400N 4N 4N 150N 800N
V4 4 6 PULSE 3.33 0 600N 4N 4N 150N 800N

M1 11 1 6 0 ENH2NMOS WOVERL=8
M2 12 2 11 0 ENH2NMOS WOVERL=8
M3 13 3 12 0 ENH2NMOS WOVERL=8
M4 14 4 13 0 ENH2NMOS WOVERL=8
M5 8 14 14 0 DEP2 W=3 L=6

.MODEL DEP2 NMOS (VT0=-2.26 KP=3.0E-5 XK1=1.0E-1 PHIF2=.75 ETA0=.95
+RESGAT=33.0 RESSD=25.0 CGS=1.7E-10 CGD=1.7E-10 CGBLK=4.2E-5
+CBOTOM=2.11E-4 CPERIF=3.5E-10 TOX=7.3E-8 VFB=-4.237 LD=3.5E-7
+WD=3.5E-7 GRID=2.0E-6 UOBRK=0.1)
.TRAN 8NS 800NS
.PLOT TRAN V(14,6) (-1,5)
.WIDTH OUT=72
.END

CIRCUIT ELEMENT SUMMARY TEMPERATURE = 25.000 DEG C

**** INDEPENDENT SOURCES

NAME	NODES	DC VALUE	AC VALUE	AC PHASE	TRANSIENT
VDD	8 6	3.33D+00	0.00D-01	0.00D-01	
VBB	6 0	2.50D+00	0.00D-01	0.00D-01	
V1	1 6	3.33D+00	0.00D-01	0.00D-01	PULSE
					INITIAL VALUE 3.33D+00
					PULSED VALUE 0.00D-01
					DELAY TIME... 1.00D-09
					RISETIME..... 4.00D-09
					FALLTIME..... 4.00D-09
					WIDTH..... 1.50D-07
					PERIOD..... 8.00D-07
V2	2 6	3.33D+00	0.00D-01	0.00D-01	PULSE
					INITIAL VALUE 3.33D+00
					PULSED VALUE 0.00D-01
					DELAY TIME... 2.00D-07
					RISETIME..... 4.00D-09
					FALLTIME..... 4.00D-09
					WIDTH..... 1.50D-07
					PERIOD..... 8.00D-07
V3	3 6	3.33D+00	0.00D-01	0.00D-01	PULSE

0 INITIAL VALUE 3.33D+00
 PULSED VALUE. 0.00D-01
 DELAY TIME... 4.00D-07
 RISETIME..... 4.00D-09
 FALLTIME..... 4.00D-09
 WIDTH..... 1.50D-07
 PERIOD..... 8.00D-07

0 V4 4 6 3.33D+00 0.00D-01 0.00D-01 PULSE
 INITIAL VALUE 3.33D+00
 PULSED VALUE. 0.00D-01
 DELAY TIME... 6.00D-07
 RISETIME..... 4.00D-09
 FALLTIME..... 4.00D-09
 WIDTH..... 1.50D-07
 PERIOD..... 8.00D-07

**** MOSFETS

NAME	D	G	S	B	MODEL	>1--GRID UNITS		<1--METERS		SQD	SQS
						L	W	DDPTH	SDPTH		
M1	11	1	6	0	ENH2NMOS	2.000000	16.000000	4.000000	4.000000	0.0	0.0
M2	12	2	11	0	ENH2NMOS	2.000000	16.000000	4.000000	4.000000	0.0	0.0
M3	13	3	12	0	ENH2NMOS	2.000000	16.000000	4.000000	4.000000	0.0	0.0
M4	14	4	13	0	ENH2NMOS	2.000000	16.000000	4.000000	4.000000	0.0	0.0
M5	8	14	14	0	DEP2	6.000000	3.000000	4.000000	4.000000	0.0	0.0

 MOSFET MODEL PARAMETERS: XSIM TEMPERATURE = 25.000 DEG C

---- LIBRARY WAS INSTALLED: 10FEB 79 VERSN 1 ----

TYPE	DIMENSION	ENH2NMOS DEP2	
		NMOS	NMOS
VT0	V	0.500	-2.260
KP	MHO/V	2.90D-05	3.00D-05
XK1	V**1/2	0.100	0.100
PHIF2	V	0.750	0.750
ETA0		0.950	0.950
RESGAT	OHM/SQ	33.000	33.000
RESSD	OHM/SQ	25.000	25.000
CGS	F/M	1.70D-10	1.70D-10
CGD	F/M	1.70D-10	1.70D-10
CGBLK	F/M**2	4.20D-05	4.20D-05
CBOTOM	F/M**2	2.11D-04	2.11D-04
CPERIF	F/M	3.50D-10	3.50D-10
TOX	M	7.30D-08	7.30D-08
VFB	V	-0.337	-3.097
LD	M	3.50D-07	3.50D-07
WD	M	3.50D-07	3.50D-07
GRID	M	2.00D-06	2.00D-06
U0BRK	1/V	0.100	0.100

 INITIAL TRANSIENT SOLUTIONS TO BE DONE AT TEMPERATURE = 25.000 DEG C

NODE	VOLTAGE	NODE	VOLTAGE	NODE	VOLTAGE	NODE	VOLTAGE
(1)	5.8300	(2)	5.8300	(3)	5.8300	(4)	5.8300
(6)	2.5000	(8)	5.8300	(11)	2.5459	(12)	2.5924
(13)	2.6396	(14)	2.6876				

VOLTAGE SOURCE CURRENTS

NAME .CURRENT

VDD	-2.689D-05
VBB	-2.949D-11
V1	0.000D-01
V2	0.000D-01
V3	0.000D-01
V4	0.000D-01

TOTAL POWER DISSIPATION 8.96D-05 WATTS

 OPERATING POINT INFORMATION TEMPERATURE = 25.000 DEG C

**** MOSFETS

	M1	M2	M3	M4	M5
MODEL	ENH2NMOS	ENH2NMOS	ENH2NMOS	ENH2NMOS	DEP2
ID	2.69D-05	2.69D-05	2.69D-05	2.69D-05	2.69D-05
VGS	3.330	3.284	3.238	3.190	0.000
VDS	0.046	0.047	0.047	0.048	3.142
VBS	-2.500	-2.546	-2.592	-2.640	-2.688

 TRANSIENT ANALYSIS TEMPERATURE = 25.000 DEG C

X
 TIME V(14,6)

X	-1.000D+00	5.000D-01	2.000D+00	3.500D+00	5.000D+00
0.000D-01	1.876D-01
8.000D-09	3.095D-01
1.600D-08	8.319D-01
2.400D-08	1.364D+00
3.200D-08	1.887D+00
4.000D-08	2.360D+00
4.800D-08	2.860D+00
5.600D-08	3.207D+00
6.400D-08	3.281D+00
7.200D-08	3.297D+00
8.000D-08	3.308D+00
8.800D-08	3.314D+00
9.600D-08	3.318D+00
1.040D-07	3.321D+00
1.120D-07	3.323D+00
1.200D-07	3.324D+00
1.280D-07	3.325D+00
1.360D-07	3.326D+00
1.440D-07	3.326D+00
1.520D-07	3.327D+00
1.600D-07	1.388D+00
1.680D-07	4.536D-02
1.760D-07	1.839D-01
1.840D-07	1.896D-01
1.920D-07	1.858D-01
2.000D-07	1.822D-01
2.080D-07	4.712D-01
2.160D-07	1.214D+00
2.240D-07	1.938D+00
2.320D-07	2.567D+00
2.400D-07	3.132D+00
2.480D-07	3.335D+00
2.560D-07	3.303D+00
2.640D-07	3.324D+00
2.720D-07	3.322D+00
2.800D-07	3.326D+00
2.880D-07	3.326D+00
2.960D-07	3.327D+00
3.040D-07	3.328D+00
3.120D-07	3.328D+00
3.200D-07	3.329D+00
3.280D-07	3.329D+00
3.360D-07	3.329D+00
3.440D-07	3.329D+00
3.520D-07	3.329D+00
3.600D-07	7.097D-01
3.680D-07	7.830D-02
3.760D-07	1.873D-01
3.840D-07	1.873D-01
3.920D-07	1.879D-01
4.000D-07	1.838D-01
4.080D-07	6.928D-01
4.160D-07	1.910D+00
4.240D-07	2.927D+00
4.320D-07	3.328D+00
4.400D-07	3.329D+00
4.480D-07	3.329D+00
4.560D-07	3.329D+00
4.640D-07	3.330D+00
4.720D-07	3.330D+00
4.800D-07	3.330D+00
4.880D-07	3.330D+00
4.960D-07	3.330D+00
5.040D-07	3.330D+00
5.120D-07	3.330D+00
5.200D-07	3.330D+00
5.280D-07	3.330D+00
5.360D-07	3.330D+00
5.440D-07	3.330D+00
5.520D-07	3.330D+00
5.600D-07	5.078D-01
5.680D-07	1.204D-01
5.760D-07	1.877D-01
5.840D-07	1.873D-01
5.920D-07	1.878D-01

6.000D-07	1.850D-01
6.080D-07	2.119D+00
6.160D-07	3.389D+00
6.240D-07	3.313D+00
6.320D-07	3.334D+00
6.400D-07	3.329D+00
6.480D-07	3.330D+00
6.560D-07	3.330D+00
6.640D-07	3.330D+00
6.720D-07	3.330D+00
6.800D-07	3.330D+00
6.880D-07	3.330D+00
6.960D-07	3.330D+00
7.040D-07	3.330D+00
7.120D-07	3.330D+00
7.200D-07	3.330D+00
7.280D-07	3.330D+00
7.360D-07	3.330D+00
7.440D-07	3.330D+00
7.520D-07	3.330D+00
7.600D-07	3.424D-01
7.680D-07	1.553D-01
7.760D-07	1.878D-01
7.840D-07	1.876D-01
7.920D-07	1.874D-01
8.000D-07	1.888D-01

Y TOTAL JOB TIME 169.00

***** 2 Mar79 ***** XEROX PARC SPICE 2.1 *****00:11:40 *****

.END

INPUT LISTING TEMPERATURE = 25.000 DEG C

*** PRELIMINARY VERSION ***** REPORT PROBLEMS TO <DAVIES> ****

ERROR: .END CARD MISSING

XEROX

PALO ALTO RESEARCH CENTER
Systems Science Laboratory
LSI Systems Area
December 19, 1978

For Xerox Internal Use Only

To: Error prone Icarus designers
From: Jim Rowson
Subject: Design Rule Checking ICARUS Files

stored: [IVY] < rowson > drc.memo

There now exists a rudimentary design rule checker for Icarus based designs. The checker reads Icarus files, checks user specified cells for a restricted set of design rules, and produces an Icarus compatible error file. The specific numbers describing the design rules are specified in the user.cm file.

What

The design rule checker, called DRC.image, is a mesa program that specifically checks three design rules: minimum feature width, minimum feature separation, and metal surrounding contact cuts. **Note:** DRC does NOT check for gate overlap, buried contacts, any of the other design rules for normal and butting contacts, and a whole host of other design rules. The algorithm for checking the design rules is not at all general at the moment and will most certainly result in spurious errors, although an attempt has been made to filter out most of the common ones.

Some processing is done on the raw rectangles that are read from the Icarus file. In particular, rectangles whose union form another rectangle are combined into one and all those rectangles which transitively touch are gathered together into a Node. A node is, at the moment, merely a region in one layer. Eventually, the recognizer for the DRC program will be augmented to the point that the Node will be a valid NMOS electrical node.

The minimum feature width merely checks each individual rectangle, after the processing mentioned above, against a number provided in the user.cm for the minimum width of the rectangle's layer. **Note:** that individual features that have a dimension less than the minimum feature size for that layer but who abut against someone to form a legal shape will be flagged as errors. With the current design style in Icarus, wherein most of the drawn shapes have the default, minimum size, these kinds of problems should not happen, but beware.

The minimum feature separation checks between each rectangle and the rectangles in all the other nodes. This means that two rectangles belonging to the same node can be too close. The minimum separation checks are not restricted to items in the same layer, the separations can be given in the user.cm. **Caveat:** when two items fail the separation test and are of different layer, they are only reported if none of the other items in the two nodes touch. This reduces spurious errors caused by, for example, minimum separation from poly to contact cut. If the contact is between metal and poly, and a separate poly line connects to the poly flash in the contact, it might come too close to the cut without intersecting it. The extra test will filter these errors out.

The metal overlapping contact test is meant to catch the common fault of leaving off the metal flash on contacts. The test merely determines if the contact is completely surrounded on all sides by one metal item. The metal item must overlap by the number specified in the user.cm. **Note:** the contact must be surrounded by one metal item. This may produce extra errors.

How Big?

The DRC program must have all its data in core at the moment, and worse than that, it has to instantiate all the instances down to rectangles. This means that it won't check anything very big. The upper limit seems to be somewhere near 2700 rectangles after instantiation. Unfortunately, in Mesa there is no easy way to tell that you have run out of room, so DRC will happily gobble up your learus file until it runs out of room, at which point it will crash (into the Mesa Debugger if you have one on your disk). If you have the debugger on your disk, it will report an Uncaught Signal, at which point you will have to Swat and try a smaller cell.

If DRC can read in and recognize your cell (split it up into nodes) then it should have no problem doing the checking, since that doesn't generate any more data. However, a test run on a chip of about 2700 rectangles took ~1 hour, including the input, recognize and checking phases. In order to ease the strain of whether or not the thing is working, the last two phases, recognizing and checking, output periods to mark their progress. Each phase outputs twenty periods in all, independent of the amount of data.

How to use!

The user interface to DRC can best be described as pitiful. There are four commands: quit, file, list, and check. The commands must be typed in full.

The quit command does the obvious thing.

The file command takes a file name as an argument and assumes the extension ".ic" if not given (to give a file name with no extension, end with "."). DRC will respond to the file command with the name of the file where the errors will go (usually filename.er).

The list command will list the cells defined in the file. The list comes out 8 lines of 4 names each, asking the question "More?" every 8 lines. A lower case 'n' will terminate the list.

The check command takes a cell name, with no regard to case, and commences the inputting, recognition, and checking. Multiple check commands can be given per file, although with large things, it is discouraged (users may proceed at their own risk). If any errors are discovered in a cell, the offending items are written out into a cell in the output file. The error cell is called 'cellname.er' in order to distinguish it from the other check commands given. The top level items in the input learus file can be checked by giving the cell name "TopLevelItems".

Multiple file commands can be given although they are also discouraged.

Below is a commented example of a slice of the DRC section of the user.cm file. The comments are in italics and the keywords are in boldface.

```
[IcarusDesignRules]  --the header, no case is distinguished
minimumSeparation: --header for separation specifications
    layer 1 1 6      layer 2 2 6  --multiple specs per line are ok
    layer 1 2 3      --first two numbers are layer, third is separation
minimumWidth:    --header for minimum width
    layer 0 14
    layer 1 6        --first number is layer, second is width
    layer 4 9
metalOverCutBy: 3  --just the overlap number
    --these three categories of specifications can be in any order
```

Where?

The DRC program resides in [IVY]<Rowson>drc.dm which contains DRC.image and DRC.user-cm-slice. The image file needs nothing to run except RunMesa.Run. The user.cm slice that is included is a sample that approximates the NMOS design-rules that the SSL/LSI group has been using. If nothing is put into the user.cm, DRC will use its own defaults which are guaranteed to be nonsense.

