Structured Logic Design of Integrated Circuits Using the Storage/Logic Array (SLA)

KENT F. SMITH, TONY M. CARTER, STUDENT MEMBER, IEEE, AND CHARLES E. HUNT, STUDENT MEMBER, IEEE

Abstract—**The Storage/Logic Array (SLA), a form of structured logic derived from PLA's, will allow development of sophisticated computer aids for VLSI design. The AND and OR planes of PLA's are folded into a single AND/OR plane. The SLA is described and comparisons with programmable logic arrays (PLA's) are made. Segmenting SLA's with arbitrary row and column breaks results in functional duality of SLA columns and allows embedded memory elements. Arbitrary SLA cell placement permits topological optimization of modules and interconnect. SLA program logic symbols map directly to IC layouts.**

Cell set realizations of SLA's in I^2L, NMOS, and CMOS are described and compared. I^2L designs are not very practical, suffering from poor fanout. Static NMOS SLA circuits provide excellent fanout, but result in high power consumption. CMOS SLA circuits use single, identical Schottky diodes for both AND and OR planes, giving dense circuits with good potential for VLSI. Programming techniques and examples are given.

INTRODUCTION

THE POSSIBILITY of fabricating integrated circuits (IC's), which have in excess of 1 000 000 gates, has created the need for the development of a methodology for IC design which focuses on the complexity problem rather than on the optimized layout problem which presently faces most IC designers. One way to solve this problem is through the use of extensive computer aids. In the past, IC circuit designers have always been faced with a limited number of gates on any single chip and with circuit requirements which could only be met by custom layout of the composites. With the dramatic reduction of device dimensions over the past few years, the designer's emphasis has been moving steadily from optimizing layout to managing the complexity problems inherent with large numbers of transistors [11].

An analogous need developed over a decade ago in the design of computer programming languages. Most computer programs were then written in machine language on minicomputers which contained very small memories and had very limited instruction sets. The PDP8 computer, for example, contained only 4K of memory and had a total of 8 instructions. Machine language coding was a necessity because of these limited resources. As larger memories and more complex instruction sets became available, it became necessary to develop programming techniques using high level languages. These early high level languages permitted and facilitated the development of large and complex programs, but were less efficient in their use of memory.

This same pattern is developing with the design of the IC today. Most IC's which are currently being designed are limited by the total number of transistors and interconnecting wires which can be placed on any given chip. This is analogous to the limited memory and instruction sets available on early computers. Economic reasons forced designers to utilize every bit of available space on the IC and thus only custom designs could be used. The most important consideration in Very Large Scale Integration (VLSI) is the development of very complex and error-free designs in practical amounts of time, rather than perfect chip space utilization. This is analogous to the need for high level languages, such as Fortran and PASCAL, in writing complex computer programs.

Present techniques for the design of complex IC's require large amounts of engineering talent. For example, typical engineering manpower required for the design of a random logic IC with 10 000 transistors might be about 2 man-years [8]. As the number of transistors in any given IC increases, the man-years of engineering design time increases exponentially. Thus VLSI circuits which contain 100 000 transistors would take significantly more man-hours of effort than the linear increase of a factor of ten from the 10 000 transistor circuit would predict. The typical 100 000-transistor part composed of random logic could easily require a 60-man-year effort. This is completely unacceptable for the IC industry and new and better techniques must be developed.

Presented here is a technique for the realization of a structured logic array which lends itself extensively to the use of computer aids. The methods which have been investigated are for the implementation of the array in three different integrated circuit technologies, I^2L, NMOS, and CMOS. These efforts have included the circuit design, composite layout, analysis, and where practical, the actual fabrication of the circuits in each of these three technologies. Concurrent investigations have also been carried out for the development of software tools to facilitate VLSI circuit design using these implementations. Some of the software tools which are being developed include a database which can adequately represent the structured logic array, a symbolic editor for generating structured logic array programs, a logic simulator, and a program which will process the symbolic database for mask making.

Manuscript received September 21, 1981; revised January 4, 1982. This work was supported by the National Science Foundation under Contract MCS78-04853, by General Instrument Corp. Research and Development Center, Chandler, AZ, and by Boeing Aerospace Company under Air Force Contract F33615-80-C-1196.

The authors are with the Department of Computer Science, University of Utah, Salt Lake City, UT 84112.

0018-9200/82/0400-0395$00.75

PART I: THE DEFINITION OF THE SLA

WHAT IS AN SLA?

The form of structured logic which has been implemented is known as the Storage/Logic Array (SLA). The SLA was originally conceived by S. Patil [13] and further elaborated on by Patil and Welch [14]. Patil's idea was to build structured logic arrays using a "folded" programmable logic array (PLA), containing memory distributed throughout the array, with arbitrary column and row breaks to give multiple, independent, finite state machines and data modules on a single IC. Circuit design is performed by placing logic symbols on a grid which, when completely populated, is the logical representation of the actual physical chip. This placement performs both the logic description and the interconnection of the integrated circuit while simultaneously giving all of the necessary information for the generation of the composite. Thus the system designer can have "visual perception" of the physical layout of the IC in terms of the logical design. This technique can simplify VLSI design.

SLA techniques are similar to those found in other structured logic layout techniques. For example, the PLA [18], the gate array [12], [5], macro cells [1], the Mead-Conway chip floor plan [10], Bristle Blocks [7], and Associative Logic [3] all have common goals and attempt to specify designs at a level of abstraction higher than the transistor or gate levels. The use of symbols to represent functions in each of these techniques significantly reduces the amount of information required to describe any particular design and thus helps to solve the complexity issue.

The structured layout of SLA's closely approximates that of the PLA where the AND and the OR planes are folded on top of each other. The logical OR is achieved by combining data along a single column and the logical AND is achieved by combining data along a single row. Memory elements are placed on the grid itself and can be randomly distributed within the SLA logic. Columns may also contain Boolean combinations of state values. Using columns to generate these Boolean expressions permits multiple levels of logic. All columns and rows can be broken at arbitrary locations. This allows the specification of independent finite-state machine and datapath modules. Row and column connections link these different data and control modules.

These features provide several advantages over conventional PLA designs. For example, data and control modules can be custom made to conform to the requirements for interconnect, thereby reducing the overall interconnection area. Efficient design changes can be made, maintaining minimum circuit area, because the designer has simultaneous perception of the circuit function and the layout. The interface between data paths and control modules is optimized by topological alignment of control signals.

The SLA should ideally be technology independent. That is, one SLA program should be portable, without change, between different process technologies. Initial experience with SLA's has shown that, in practice, this is not the case. Different process technologies not only have different SLA programming design rules, but have radically different advantages and disadvantages. It becomes necessary to select a particular process and implementation based on the individual circuit needs. Specifically, it was shown [9], [15] that I^2L is extremely limited in allowing large numbers of actions controlled by a row or column because of poor fanout. NMOS proved to be able to handle large, heavily loaded circuits, but at the price of high power consumption and relatively low speed [16]. CMOS largely overcomes most of the objections to I^2L and NMOS at the expense of a more complex process and significantly larger inverters and flip-flops [17].

SIMPLE SLA CELL SET

An SLA program is a two-dimensional array of symbols that specifies the placement of cells for a given circuit. The elements of the SLA program are taken from an SLA cell set which is predefined and dependent upon the specific technology which has been chosen. The SLA cell set will include memory elements, inverters, combinational elements for the folded AND and OR plane, row and column breaks, and row and column connection cells. The memory elements might be as sophisticated as a set/reset read/write-enabled master/slave flip-flop or as simple as a set/reset latch. The inverter is always a simple inverting buffer. The combinational elements may be as complicated as multiple clocked transistors or as simple as single diodes. Row and column breaks simply indicate the absence of row and column connections. The SLA designer assumes the placement of the connection cells between adjacent columns or rows unless he specifies a row or column break.

A minimum set of cells which is common to all technologies and implementations includes a set/reset flip-flop for the memory element, inverters for generation of true and false column signals, 0, 1, S, R, and + combinational elements, and row and column breaks. The 0 and 1 are the combinational elements of the AND plane and detect, respectively, the false and true states of a flip-flop or the false or true outputs from an inverter. The S and R are among the combinational elements of the OR plane and, respectively, set or reset a flip-flop. The + is also a combinational element of the OR plane and serves as the input to an inverter. Its circuit description is identical to that of the S combinational element.

The original SLA concept [13] assumed that the SLA would be implemented for asynchronous design. The first designs which were made for the SLA were asynchronous structures in I^2L [15], but only limited success was achieved with these structures. As a result of this experience, synchronous as well as asynchronous structures were developed for I^2L. Designs in NMOS [16] were done strictly for synchronous structures. The reasons for this choice were: 1) typical NMOS circuits are almost exclusively synchronous designs, 2) most existing NMOS systems require the use of a synchronizing system clock, 3) synchronous systems are much better understood by the majority of IC designers, and 4) synchronous designs were generally assumed to be physically smaller than their equivalent asynchronous designs. SLA structures in CMOS [17]

were made for both synchronous and asynchronous designs. This need is recognized in part because of many new advances in understanding asynchronous design [4], [6]. Thus the CMOS SLA cell set will allow both synchronous and asynchronous designs.

A synchronous set of elements is implied in the explanation below but the principles of SLA operation, either synchronous or asynchronous, are very similar. The relative advantages and disadvantages of the synchronous and asynchronous designs will be discussed in the description of the I^2L implementation of the SLA. The synchronous elements used in this illustration require a two-phase nonoverlapping clock scheme. The true and false outputs from the flip-flops or inverters are detected during clock phase one and the *S*, *R*, and + actions take place during clock phase two.

SLA Concepts

Several design features of SLA's produce circuits with both performance and size that approach custom designs. Four features which are discussed below are: 1) symbolic representation of circuits; 2) optimization of a single SLA program to give minimum layout; 3) optimization of SLA blocks to minimize interconnect between modules; and 4) embedding of logic on bus lines to produce tightly packed data and control path modules. Each of these design features is completely controlled by the logic designer rather than by the combination of a logic designer and a composite layout designer. Most circuits being designed today involve specification by the logic designer who then gives the design to an experienced layout designer. The layout designer does a physical layout and then gives it back to the logic designer. The logic designer makes changes which are appropriate for that layout and then gives the design back to the layout designer again. This process repeats until an acceptable layout and logic combination is achieved. SLA design does not require this kind of interaction because only the logic designer is required. As a result, the finished composite has characteristics which are similar to that of an optimized hand layout.

One of the simplest examples demonstrating the symbolic design used in the SLA is the design of an oscillator. The SLA program and the physical realization for an oscillator is shown in Fig. 1.

In this case, the set/reset flip-flop is placed at the top of a column of combinational elements. When the flip-flop is in the reset or "0" state, it is detected by the 0 in row 1 during clock phase one. The flip-flop is set to the "1" state during clock phase two by the *S* in row 1. The set or "1" state of the flip-flop is then detected by the 1 in row 2 during clock phase one and the flip-flop is reset to the "0" state by the *R* in row 2 during clock phase two. Thus the flip-flop will oscillate at one half the clock frequency.

The flip-flop is a master/slave set/reset flip-flop with true and false outputs and set and reset inputs. The set and reset inputs are clocked into the master flip-flop during phase one and the true and false outputs are clocked to the slave flip-flop during phase two. The AND plane is represented by the transistors marked with the 1 and 0 symbols. The OR plane is represented by the transistors marked with the *S* and *R* symbols. The row pull-up resistors, *R*1 and *R*2, are used for loads on the AND gates formed on the rows. The column pull-up resistors, *R*3 and *R*4, are used for loads on the OR gates formed on the columns.

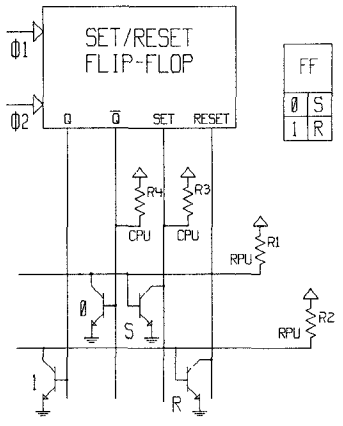


Fig. 1. SLA program and physical realization of an oscillator.

ROW	MSB			LSB	RESET	COUNT	ALPHA
	FF	FF	FF	FF	INV	INV	FF
1	R	R	R	R	1		
2				ØS	Ø	1	1R
3	Ø		ØS	1R	Ø	1	1R
4		ØS	1R	1R	Ø	1	1R
5	ØS	1R	1R	1R	Ø	1	1R
6	1R	Ø	Ø	1R	Ø	1	1R
7						Ø	ØS
COLUMN	1	2	3	4	5	6	7

Fig. 2. SLA program for a modulo 10 BCD counter.

A second example of symbolic SLA programming is shown in Fig. 2. This is the SLA program for a BCD (modulo 10) counter. There are four flip-flops which define the counter states located in columns 1 through 4. The reset and count inputs are located in columns 5 and 6. Column 7 contains an alpha flip-flop which is used as an edge detector to force a single count whenever the count input goes to a "1." Counting occurs whenever the alpha flip-flop and the count input are both a "1." The alpha flip-flop is reset to the "0" state immediately after the count and will remain in that state until the count input returns to a "0."

The sequence of row activations counting from 0 through 9 is shown below. Only six SLA rows are required to count from 0 to 9 because of shared states in the counter. For example, the required actions for changing the count from 0000 to 0001 is identical to changing the count from 0010 to 0011 since only the least significant bit (LSB) changes during this counting sequence.

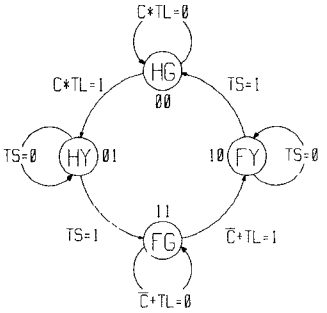
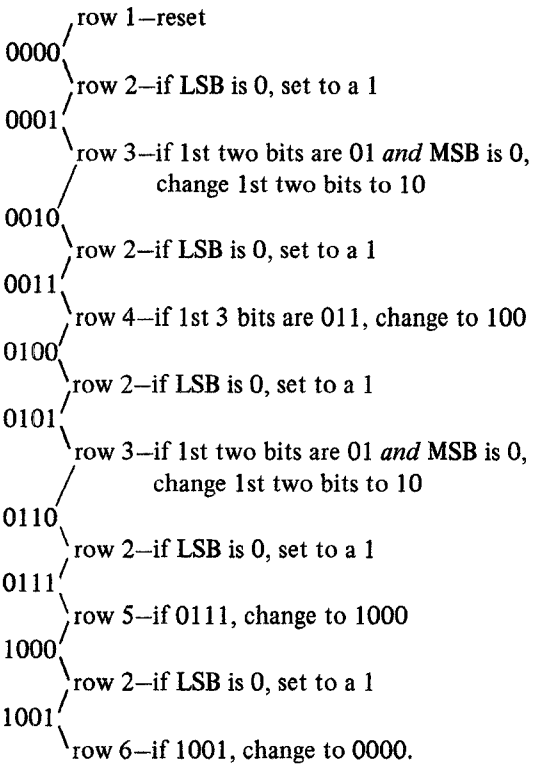


Fig. 3. State machine for SLA example.



A third example which illustrates the symbolic representation for the translation from a conventional state machine description to a PLA structure and then to an SLA program is shown in Figs. 3-5. The state machine is taken from Mead and Conway's text [10, pp. 85-88], where a complete description of the machine is given. The state machine itself is shown in Fig. 3. The PLA realization of this state machine using set/reset master/slave flip-flops is shown in Fig. 4, and the SLA realization of this same state machine is shown in Fig. 5.

The machine contains four states named *HG*, *FY*, *FG*, and *HY*. The external inputs to the machine are *C*, *TL*, and *TS*. The external outputs are *ST*, *HL*0, *HL*1, *FL*0, and *FL*1. Two memory elements, *Y*0 and *Y*1, are used to define the four states. The AND and OR planes are conventional representations for the PLA structure. The inputs to the AND plane are actually double wires so either a true or a false input can be detected by the placement of a 1 or a 0 in the AND plane. The OR plane outputs are all single wires which are activated by the + in any given row of the OR plane.

The SLA realization of this same state machine is shown in Fig. 5. The flip-flops and inverters for the inputs and outputs have been imbedded into the structure. Close examination of

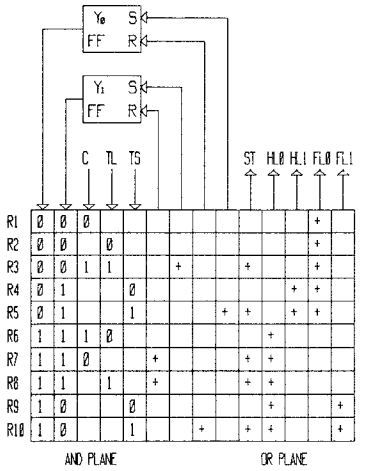


Fig. 4. PLA realization of state machine.

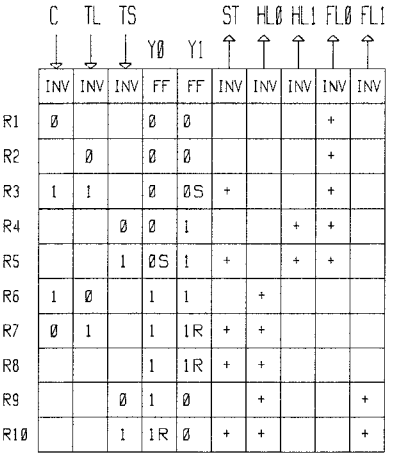


Fig. 5. SLA realization of state machine.

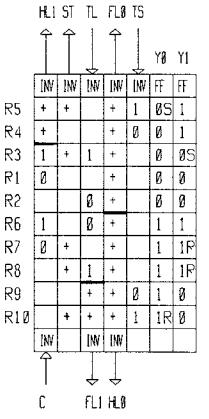


Fig. 6. Optimized SLA state machine.

the program will show that the PLA design and the SLA design are logically identical.

The optimization of a given SLA program is illustrated in Fig. 6. This SLA program will perform exactly as that shown in Fig. 5. The design has taken advantage of the ability to break and interleave columns to reduce the physical dimensions of the design. There are three column breaks, which are represented as heavy solid lines in columns 1, 3, and 4 in this program, giving six columns in the space of three. The pro-

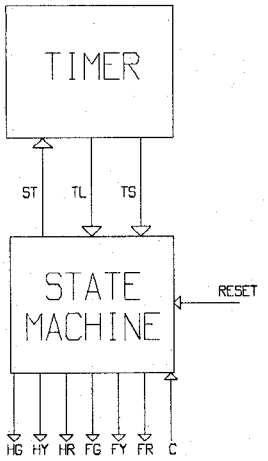


Fig. 7. Block diagram of system for SLA example.

gram has thus been reduced from 10 columns in Fig. 5 to 7 columns in Fig. 6. Outputs *FL*1 and *HL*0 and input *C* are now at the bottom of the machine rather than at the top. This may or may not be an advantage in a particular design, but the logic designer does have a choice of the actual placement of the logic elements.

A powerful technique which is available to SLA designers is the ability to control the physical layout of any given set of SLA modules to give an optimum arrangement to minimize the interconnect and area. The logic designer has complete control of the physical layout, making an optimum layout more attainable. An example demonstrating this type of design is illustrated by the simple system shown in Fig. 7. The state machine in Fig. 7 is identical to that shown in Figs. 4 and 5, but has six completely decoded outputs instead of four encoded outputs. The timer is added as a component part of the system. The combination of the timer and the state machine form a complete system and the implementation of this system in SLA notation is shown in Fig. 8.

The timer itself is shown in the upper left-hand corner of the SLA program. The timer is simply a 12-bit binary counter which has been implemented in a fashion similar to that shown for the BCD counter in Fig. 2. The state machine is shown in the lower right-hand corner of the program and has been arranged to conform to the layout of the timer. These two independent SLA modules are separated by row and column breaks. The communication paths within the system are embedded inside the SLA itself. The *TS* and *TL* signals from the timer to the state machine are located in columns 17 and 16, rows 6 and 3. The *ST* output of the state machine is located on row 17 in columns 1 to 11. This compact arrangement of an independent SLA module for the state machine, an independent SLA module for the timer, and SLA cells for the interconnect illustrates the control a logic designer can exercise over the topology of the integrated circuit.

The tight coupling of the SLA modules and the interconnect allows the designer to place data modules directly on signal buses. A block diagram of a logic module which illustrates this concept is shown in Fig. 9(a). Bus lines *Z*0, *Z*1, *Z*2, and *Z*3 are connected to a 4-bit storage register *A*0, *A*1, *A*2, and *A*3. The control for loading the bus to the register and for loading the register to the bus are located on the side of the

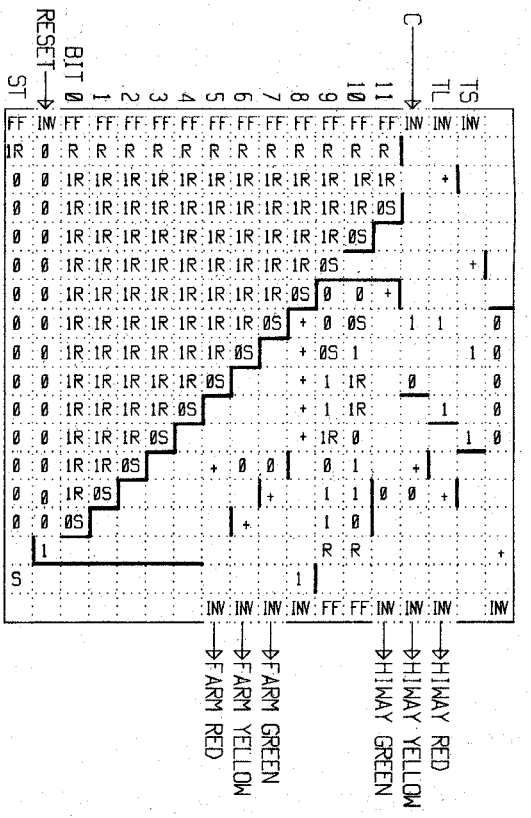


Fig. 8. SLA program for system in Fig. 7.

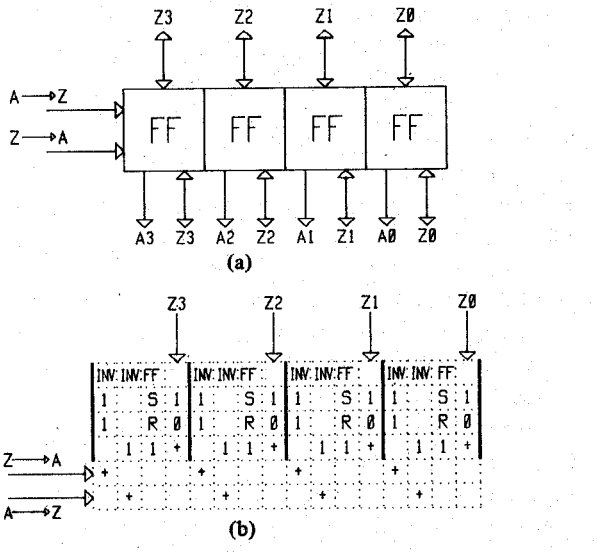


Fig. 9. (a) Block diagram and (b) SLA program of embedded modules on a data bus.

register and enter as row signals. The SLA program which implements this configuration is shown in Fig. 9(b).

Part II: The Physical Implementation of the SLA

Realization of the SLA Concept in I^2L

The physical implementation of the SLA was first realized using I^2L technology [15]. This technology has several distinct advantages over other technologies for the implementation of the SLA. One of the most important advantages is the presence of n-p-n transistors whose emitters (collectors) are tied to the substrate, eliminating the necessity to run a ground

wire throughout the integrated circuit. These grounded emitter transistors can be used for the combinational elements of the SLA and excellent packing density in the AND/OR plane can be achieved. I^2L has additional advantages of excellent speed-power product, linear and digital compatibility, and the ability to withstand very severe environmental conditions. The fabrication process is a relatively simple bipolar process and conventional bipolar circuits can be added to the I^2L integrated circuit. Some of the disadvantages are poor fanout, no pass transistor, and excessive voltage drops which can occur along signal paths.

The basic SLA cell set which was developed for the I^2L technology includes memory elements and inverters, combinational logic elements, row and column loads, and miscellaneous interconnect cells. The cell set was originally designed to perform asynchronous logic but practical limitations dictated the design of additonal cells to perform synchronous logic.

The asynchronous cells include: 1) a set/reset latch for the memory element; 2) an inverter; 3) 1, 0, S, R, and + combinational cells; 4) row and column pull-up cells (these pull-up cells are the dc loads for the AND and OR plane gates); and 5) miscellaneous cells including *, #, and blank cells. The * and the # cells are ohmic contacts between rows and columns and are used to hard-wire signals from one position to another. Row and column interconnects are assumed to always be present between cells unless the programmer deletes them, creating a row or column break. The blank cell is used to carry row and column signals through nonfunctional areas.

The smallest building blocks in the SLA layout are the combinational row cells. These cells have been kept as simple and small as possible to establish the minimum grid spacing in the SLA and they occupy a space of one row by one column. The flip-flop cells occupy a space of two columns by seven rows. The inverter cell is one row by one column. All of the SLA cells may be placed in any arbitrary location on this minimum grid without regard to what cell lies next to it.

A schematic containing all the combinational logic cell configurations as well as a flip-flop cell is shown in Fig. 10. Transistors $Q3$ thru $Q6$ are four combinational cells which represent the 1, R, S, and 0 functions, respectively. The + cell is an input to the OR gate formed on the column which is connected to an inverter cell. The + cell is electrically identical to the S cell (transistor $Q5$). The p-n-p pull-up transistors, $Q11$ and $Q12$, are the row pull-up cells. Although the actual size of the p-n-p pull-up transistor cell is fixed, the transistor size varies depending upon the different loading conditions which occur on the rows.

The design of the inverter cell for the asynchronous I^2L SLA is represented in Fig. 10(b). This is a simple inverter composed of an n-p-n transistor, $Q1$, with two p-n-p load transistors, $Q2$ and $Q3$. The pull-up transistors sizes can be varied for a specific design depending upon the load to be driven.

The asynchronous design shown in Fig. 10(a) has the distinct advantage of having a minimum number of components and potentially the highest speed of any design which might be devised. This asynchronous design does have a potential race condition, however, which will cause errors if the delays within the circuit become excessive. The asynchronous circuit operates with a clear timing separation between the detection

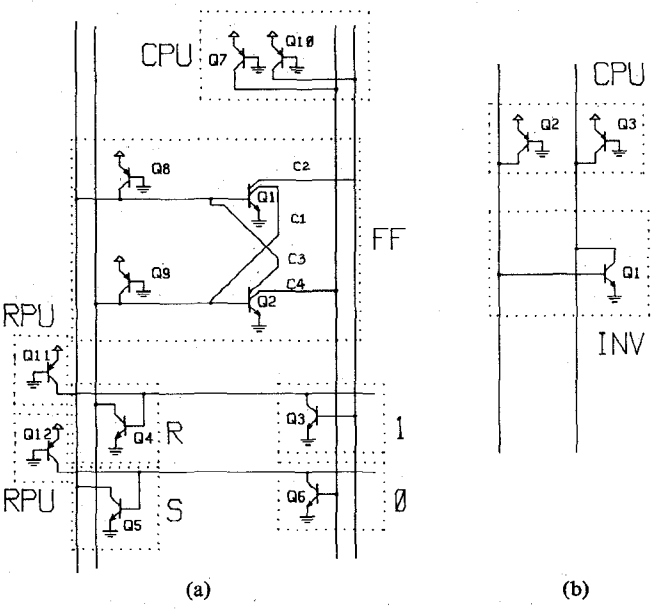


Fig. 10. (a) Asynchronous I^2L SLA flip-flop with two associated rows. (b) Schematic of asynchronous I^2L inverter and pull-up cells.

of the "1's" and "0's" in the AND plane and the set and reset actions of the OR plane, provided that both collectors $C2$ and $C4$ of transistors $Q1$ and $Q2$ are pulled simultaneously to ground upon a set or a reset condition. This problem, along with the constraints which must be placed upon the I^2L SLA design, has been studied previously [9]. It was shown that the injectors in both the columns and rows had to be custom fit to the situation, and an algorithm for determining the relative size of each of the injectors was derived. The conclusion was that it would only be possible to have a very limited number of 1's and 0's or S's and R's on any row in a practical asynchronous circuit.

Synchronous SLA design in I^2L eliminates many of the loading problems and design restrictions which were present in the asynchronous design. The row cells of the synchronous structure are the same as those used in asynchronous designs with the substitution of a clocked load device on each row rather than a dc load device. These load devices become active when the clock is high. The synchronous design uses a master/slave flip-flop (MSFF) rather than the set/reset latch which was used in the asynchronous design. The MSFF consists of two set/reset latches with appropriate circuitry for transferring the data between the master flip-flop and the slave flip-flop when the clock is low. Fig. 11 is a schematic drawing of the synchronous equivalent to the circuit shown in Fig. 10(a).

Fig. 12 shows a representative SLA program which utilizes most of the cells which have been described, including the synchronous flip-flop, column and row pull-ups, inverter cells, row and column interconnects, and blank cells. The program occupies 12 columns and 13 rows. The flip-flop occupies columns 1 and 2 and the inverter cells occupy columns 3, 4, 7, 8, 11, and 12. Columns 5, 6, 9, and 10 are blank and are used to represent the use of the * and # cells. The heavy lines represent breaks in the rows and columns. Fig. 12(b) shows the practical realization of the SLA program shown in Fig. 12(a). The blank columns which were shown in Fig. 12(a) have been eliminated. The row and column pull-up cells (shown with the symbols *RPU* and *CPU*) have been added to Fig. 12(b). These

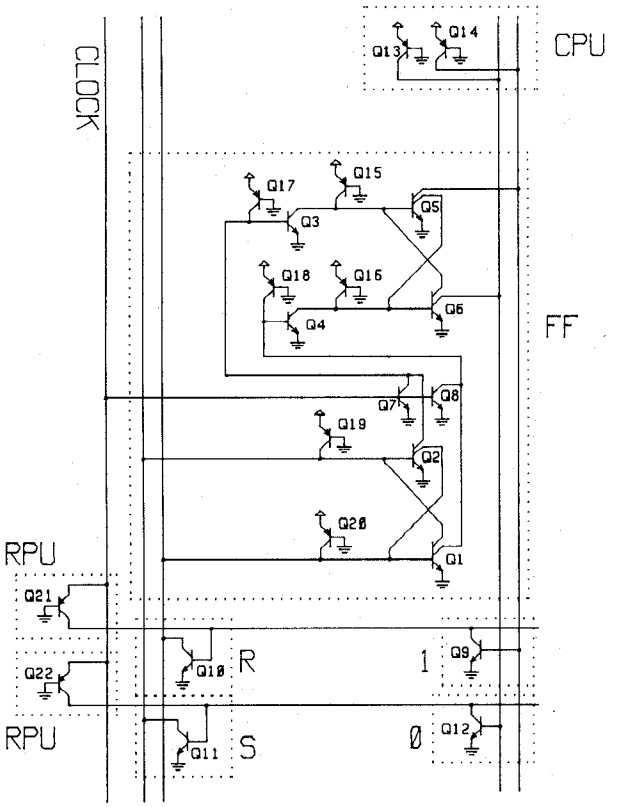


Fig. 11. Schematic of I^2L synchronous master-slave flip-flop and two row cells.

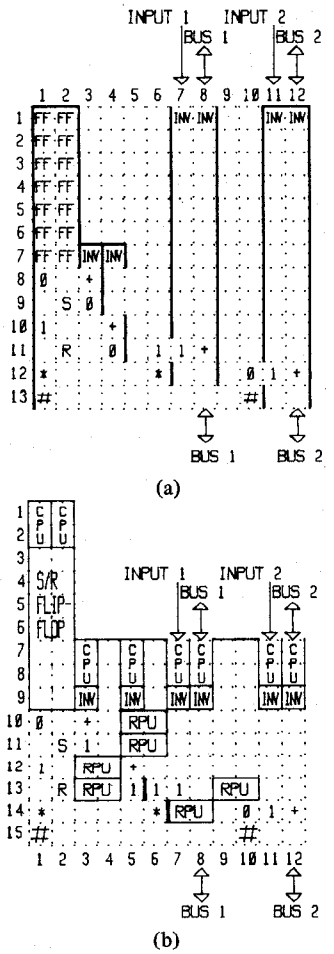


Fig. 12. (a) Representative I^2L SLA program illustrating use of symbols. (b) Synchronous I^2L cell placement for program in (a).

cells do not show up in Fig. 12(a) because it is simply a logical description of the circuit, but they are an essential part of the design and must be included.

A composite layout of the I^2L SLA cell placement shown in Fig. 12(b) was made. The process was a standard, 7-mask, non-buried-layer, two-layer metal process with a minimum feature size of 5 μm. A single row was 18 μm high and a column was 110 μm. The active area of the circuit was 270 X 660 μm.

SLA Design Using NMOS Technology

The majority of IC designers are experienced in MOS design and hence more emphasis will be placed on SLA designs which are done in an MOS technology. An SLA cell set was designed using a conventional 5-μm NMOS silicon-gate process. Both enhancement and depletion transistors were used as well as buried contacts for connections between poly and N diffusions. Unlike the previous sets in I^2L, only synchronous cells were designed. Two different synchronous schemes were explored for the implementation of the NMOS SLA. These were 1) dynamic multiphase (four or more) clock schemes which used very little ratioed logic and 2) a static two-phase clock scheme using clocked pass transistors in memory elements and ratioed logic for all inverters.

An investigation [16] showed that a dynamic implementation of NMOS SLA's would be difficult and would entail certain crucial tradeoffs in size and performance. For example, a dynamic circuit would require an additional clock for each multiple level of inverter logic in the program. The space needed to route these additional clocks would quickly deplete any space savings in other parts of the circuit. Thus the investigation of a dynamic implementation of the SLA was delayed and a static two-phase clock scheme was pursued. The discussion below will concentrate on this static implementation.

The static NMOS circuit uses the same four-column wires for the set, reset, true, and false signals from the flip-flop as in the I^2L designs and two-column wires for the inverter. The row logic functions are accomplished by the use of single NMOS transistors, similar to that used in the I^2L SLA. The set of cells designed in NMOS included the following: 1) four combinations of read/write-enabled set/reset flip-flops; 2) four combinations of read/write-enabled D flip-flops; 3) two inverter cells (for odd and even columns); 4) eighteen row and column load cells; 5) seven odd- and even-column 1, 0, *R*, *S*, and + cells; and 6) fifteen miscellaneous interconnect cells for column connect, row connect, end caps, and blank rows.

As in I^2L, the minimum grid spacing of the SLA is dictated by the dimensions of the combinational logic row cells. These cells occupy a single column width and a single row height. Composite layouts of these row cells were found to be 35 μm high and 75 μm wide. The set/reset and D flip-flop cells are five rows high and two columns wide. The inverter cell occupies a single row and column. Each row and column segment is terminated by a row or column load cell. Row and column load cells are selected from a group of cells which have different width and length ratios. The selection is dependent on the loading.

The relationship between the SLA symbols and the actual circuit, using the static, two-phase cell set is best understood by examination of the simple SLA shown in Fig. 13. This fig-

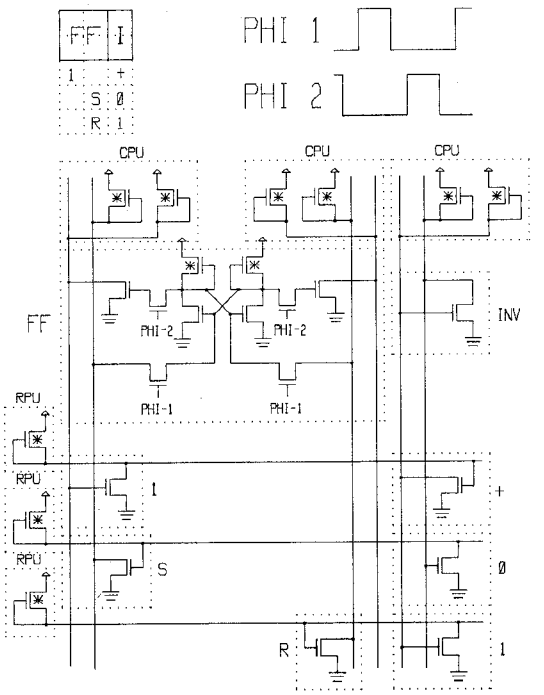


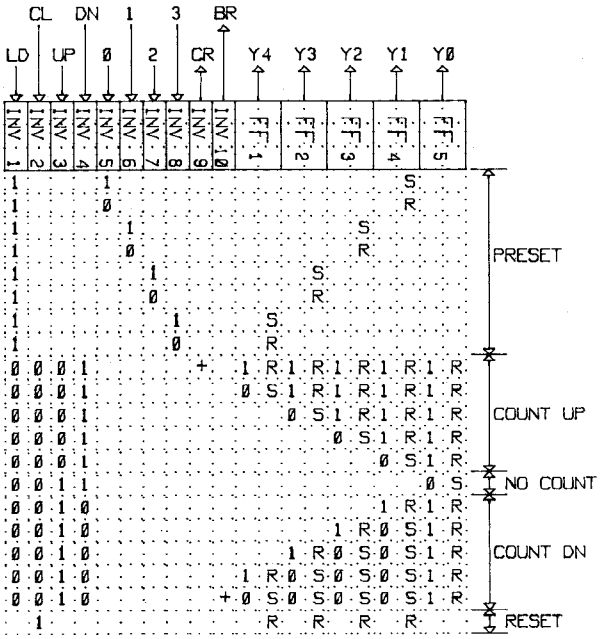
Fig. 13. Static NMOS SLA showing SLA program, circuit, and two-phase clock waveform.

ure contains the SLA program, the nonoverlapping two-phase clock waveforms, and the circuit schematic.

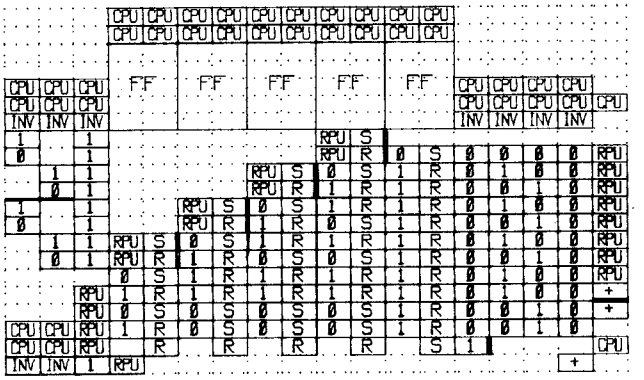
An SLA program for a presettable 4-bit up-down counter, implemented using the two-phase static scheme, is shown in Fig. 14(a). The counter is composed of four flip-flops, *FF*1-*FF*4 in columns 11-18. A fifth flip-flop, *FF*5 in columns 19-20, is used for edge triggering of the counter. There are 10 inverters, *I*1-*I*10 in columns 1-10 which are used to generate the true and false values of the input data and the internally generated data. The counter can be preset by the action represented in rows 4-11. Each of the four inputs, 0, 1, 2, and 3, columns 5-8, are loaded into *FF*4, *FF*3, *FF*2, and *FF*1, respectively, by means of the 1-*S* and 0-*R* in rows 4-11. The load action can take place only when the load input is a 1.

The controls for counting up are contained in rows 12-16. The controls for counting down are contained in rows 18-22. Counting takes place when the *LD* (load) and the *CL* (clear) inputs, columns 1 and 2, are low, "0." Up counting occurs when the *DN* input, column 4, is held at a "1" and the *UP* input, column 3, is toggled from a "1" to a "0." Likewise, down counting is accomplished when the *UP* input is held at a "1" and the *DN* (down) input is toggled from a "1" to a "0." No counting occurs when both the *UP* and the *DN* inputs are a "1." *FF*5 is used to force a single count whenever the toggling of either the *UP* or the *DN* input occurs.

Segmenting the rows and columns significantly reduces the space occupied by this counter scheme. Fig. 14(b) represents the same counter as that shown in Fig. 14(a) using 18 columns and 21 rows as compared to the 20 columns and 23 rows in Fig. 14(a). Row and column pull-up cells and blank cells for carrying signals to various sections of the SLA have also been added. The flip-flops were also changed to be 5 rows high.



(a)



(b)

Fig. 14. (a) SLA program for NMOS 4-bit up-down counter. (b) NMOS SLA cell placement for program in (a).

The CMOS SLA Cell Set

The CMOS SLA cell set contains elements [17] which have been present in all previous SLA implementations. These elements are not implemented with exactly the same functionallity in CMOS as in I^2L and NMOS, but can be used to write SLA programs which are functionally equivalent. The basic CMOS SLA elements were designed at a more primitive level and include such elements as a single inverter, a double inverter, a simple NAND gate latch, a pass transistor, and the standard 0, 1, *S*, and *R* row combinational elements. The higher level SLA elements such as the master/slave set/reset flip-flop are made by combining these lower level elements. This concept of primitive SLA cells allows a designer the opportunity of building extended SLA cells which are optimized for a particular design as well as the more common cells which have been available in the I^2L and NMOS circuits. These extended SLA cells can be constructed from the primitive elements to have both a desired physical aspect ratio and logical arrangement to fit the particular needs of a design. In addition to the SLA elements, a collection of miscellaneous cells is re-

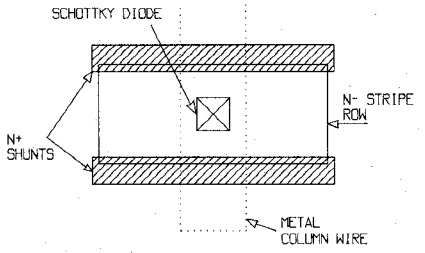


Fig. 15. Schottky-diode composite.

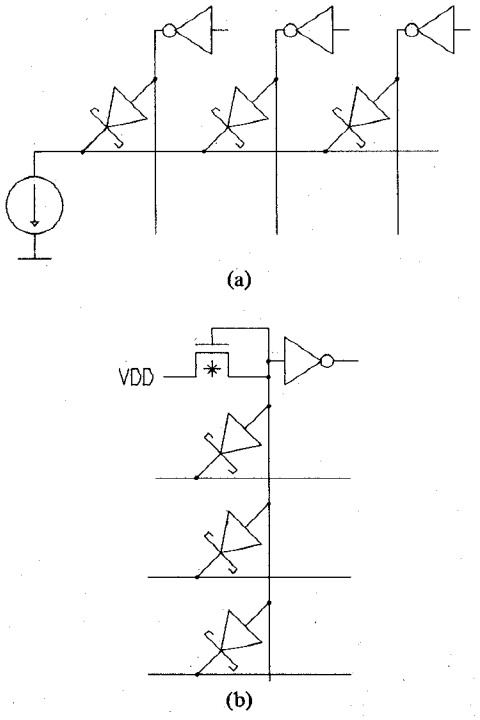


Fig. 16. (a) CMOS AND plane. (b) CMOS OR plane.

quired. These cells included power buses, row and column loading cells, and blank cells.

The CMOS SLA cell set was designed around a 4-μm n-well process with n- and p-type enhancement transistors and n-type depletion-mode transistors. The n-regions made possible the inclusion of Schottky diodes as the key elements of the combinational logic. The n-regions used for p-channel devices were encircled by an n^+ high bias ring, which in turn was surrounded by a p^+ grounded guard ring.

Identical Schottky diodes were used for the combinational elements of both the AND and the OR planes. These diodes are made with no additional processing steps by simply contacting metal column wires to horizontal n-stripes. The AND and the OR planes can thus be merged together with no space penalty. This was not true in the I^2L and the NMOS technologies which required a differently configured transistor in the AND plane than in the OR plane. A composite representing a single Schottky diode in a row cell is shown in Fig. 15. The row of the SLA is an n-stripe with low-resistance n^+ shunts on each side.

Representative AND and OR planes are shown in Fig. 16(a) and (b) and are implemented using negative true logic. The inputs to the AND plane in Fig. 16(a) are the three columns

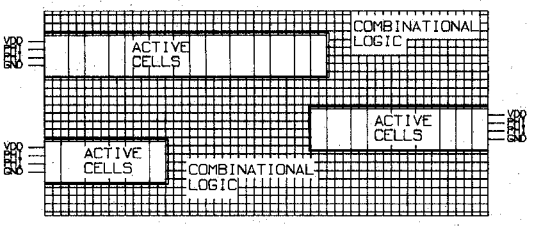


Fig. 17. Block structure of a CMOS SLA program.

which are connected to the anodes of the diodes. When an active element (such as a latch, flip-flop, or inverter) drives through a diode to the row, the current drive of the active element exceeds the pull-down capability of the row current source and the row is forced high. The output of the AND gate is the horizontal row which is pulled low by the current source. This current source is actually a depletion-mode n-channel transistor with its gate and source grounded. The row will become true (low) when all of the inputs (columns) are true (low). Thus this row is the logical AND of the three columns. The OR plane in Fig. 16(b) has inputs (rows) at the cathodes of the three Schottky diodes and has its output on the column. The column will become true (low) when any of the three rows are true (low). Thus this column is the logical OR of the three rows. Therefore, the SLA rows are both the outputs of the AND plane and the inputs to the OR plane and the SLA columns are both the inputs to the AND plane and the outputs from the OR plane.

The proper operation of the OR plane is dependent upon the ratio of the current being sourced by the column pull-up transistor and the current being sunk by the row pull-down transistor. Any one of the three row pull-down transistors must be capable of sinking the current from all of the depletion-mode pull-up transistors which are connected to that row. In all cases, the sinking current must be greater than the sum of all sourcing currents.

Since the combinational logic cells (1, 0, S, and R) are identical diodes, the only difference between these cells is their notation in the program itself, and this difference is maintained strictly for clarity in the SLA file. With a logic array cell size of 14.5 μm wide by 27 μm high, the packing density of the CMOS combinational logic is about three times that of NMOS. An obvious advantage of CMOS over NMOS for SLA designs would be in implementing circuits with a high ratio of combinational logic to active elements.

CMOS circuits produced a new cell design problem. The inclusion of guard rings causes the well size of the individual active elements to become prohibitively large if a cell were able to stand alone within the SLA grid. The problem is solved by requiring that active elements be programmed in the SLA as horizontal "bands" with shared n-wells, as shown in Fig. 17. At each end of the band is an "end cap" for the well, which closes the guard rings and is two column wires wide. SLA designs in other process types which did not require horizontal "banding" of active cells typically already had active cells programmed in horizontal alignment for convenience, hence this design constraint is considered to be a reasonable tradeoff and is not likely to become a serious restriction in SLA programming. The cell size is further improved over pre-

vious implementations by not carrying power, ground, and clocks in each column. All power, ground, and clock lines are brought in horizontally in the active cell bands, with column wires tunneled underneath in the vertical direction, permitting a much smaller minimum size for the array cells. The location of horizontal bands and their frequency is left totally to the SLA programmer, as required by the circuit being designed.

Active elements in the CMOS implementation are static and asynchronous. The logic being implemented may, however, be synchronous. To provide the possibility of using a system clock, two clock lines are bused in with the power and ground in the horizontal bands of active cells. Clocking schemes are included as a part of the SLA program using the array cells. (This is done to take advantage of the greater density of combinational logic in the array.) The clocks must be tapped using a "bender" cell, which connects a power bus wire to a column wire, and detected on a row in order to provide clocked synchronization. Similarly, the ground must be tapped by a bender cell to supply the necessary grounding for the row pull-down current sources.

Building complex SLA cells from the primitive cell set results in several advantages over prior SLA implementations. Complex elements (macros) can be custom designed so they fit well into the overall circuit. Standard circuit elements, such as dynamic shift registers, register stacks, multiplexers, etc., can be easily implemented using the cell set. Unnecessary functionality and/or circuit elements are not inserted into the SLA. Synchronous and asynchronous circuits can be implemented with the same cell set.

The construction of macro elements is best illustrated by the examination of an SLA program for a set/reset MSFF. This is one of the more complex SLA cells which was used in the I^2L and the NMOS technologies. Two nonoverlapping clocks, Phi-A and Phi-B, are used to gate the set/reset inputs and true/false outputs of the flip-flop. Fig. 18(a) and (b) shows two different SLA program implementations of an MSFF using the primitive CMOS SLA cell set. The two cells perform identical logic functions but are very different both in shape and in the cells used to implement them. The version of the MSFF illustrated in Fig. 18(a) uses pass transistors (PTA) enabled while clock Phi-A is high to transfer the set/reset inputs into the master flip-flop, and pass transistors (PTB) enabled while clock Phi-B is high to transfer the outputs from the master into the slave.

The version of the MSFF in Fig. 18(b) achieves the same effect as that in Fig. 18(a), but through a very different technique. Phi-A is routed through a single inverter because the clocking scheme used requires that the clocks be inverted before being sensed by SLA rows (the 0's in column 10). The left segment of row 7 AND's the buffered set input (the 1 in column 3) with an inverted Phi-A clock to generate the set signal for the master flip-flop and row 8 AND's the buffered reset input (the 1 in column 6) with an inverted Phi-A clock to generate the reset signal. Phi-B is similarly inverted to clock the transfer of data from the master to the slave (the right segment of row 7 and row 9).

The CMOS SLA cell set will allow the generation of macros with a variety of different physical and logical configurations. For example, an MSFF macro which included separate read and write enable inputs could easily implement a hardware stack. Other circuit elements, such as dynamic shift registers, would be built using the same techniques used in constructing master/slave flip-flops.

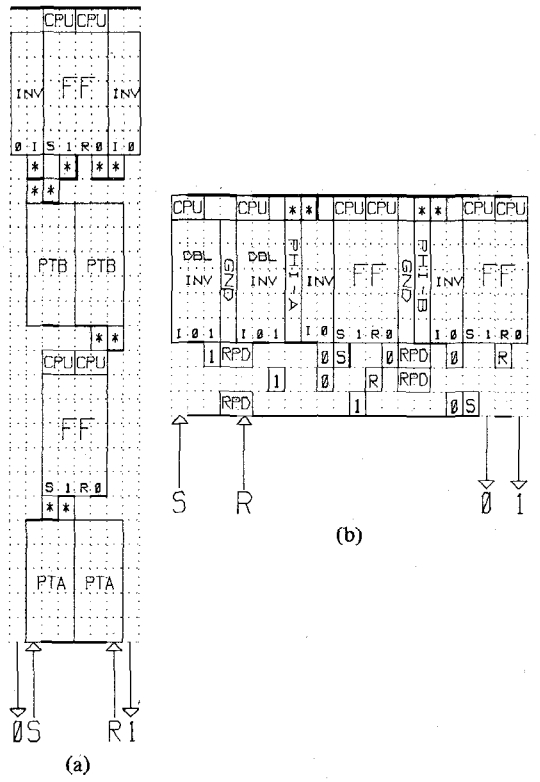


Fig. 18. (a) and (b) Two different CMOS SLA macros of functionally equivalent MSFF's.

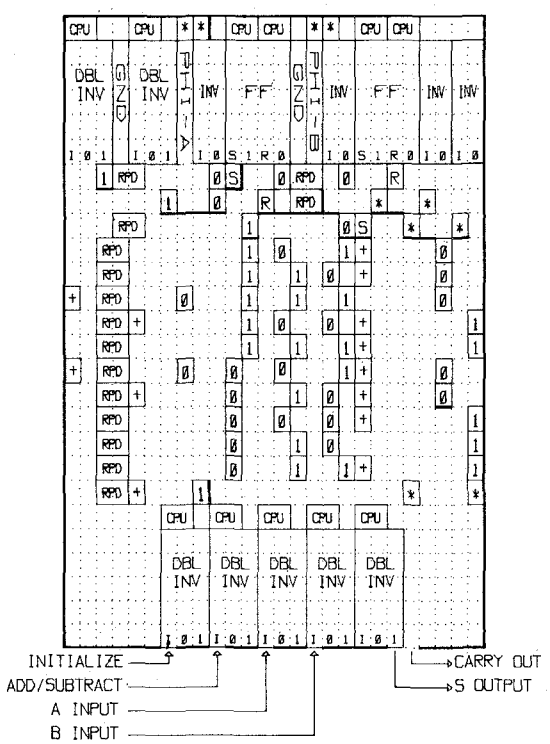


Fig. 19. The USCM adder/subtractor.

Other larger macros can also be built. Fig. 19 contains the SLA program for a bit-serial adder/subtractor of the type used

in designing a much larger machine, the Utah Serial CORDIC Machine (USCM) [2]. Three internally identical bit-serial adder/subtractors were required in the CMOS SLA implementation of the CORDIC. The top eight rows of the SLA program for the bit-serial adder/subtractor contain an MSFF of the type illustrated in Fig. 18(b). This MSFF is used to store the serial carry generated by each successive bit-add or bit-subtract. Macro elements were used to construct 16-bit dynamic shift registers, adder/subtractors, set/reset MSFF's, and parallel loaders for dynamic shift-registers.

Conclusions

The storage/logic array (SLA) is a viable solution to the current complexity problem facing most custom IC designers who are attempting to use small devices in order to attain VLSI. Memory elements can also be placed within the array itself. This improves the space utilization of programmed elements within the array. By using a predefined set of cells, a circuit can be specified by the logic designer. Software tools (i.e., an editor, logic simulator, etc.) can work effectively at the cell level. A direct translation can be made between the logic description of the circuit (the SLA program) and the composite level description used to generate masks. Design times can be lowered by eliminating the necessity of a custom composite level design.

Programming techniques and examples were given. Variations of programming methodology can minimize interconnect within the array to other modules, thereby improving the program density. Design examples in I^2L, NMOS, and CMOS were completed to the composite level and each implementation of the SLA was examined and simulated. Several designs have been submitted for manufacture, but measured results are not available as yet.

Design of SLA circuits using I^2L has the potential for large circuits if the loading and timing problems are appropriately considered. The asynchronous structure has the advantage of higher density but suffers through considerable timing problems. The timing problems can be solved by using a synchronous structure, but the solution causes reduced density. A major disadvantage of the I^2L SLA is the poor fanout which limits the number of practical AND and OR plane terms. This can be overcome only at the expense of difficult design procedures. In addition, the relatively simple processing which was originally envisioned has to be further complicated by the addition of at least two-layer metal for the large AND and OR planes which are required. A further, and perhaps more important disadvantage, is that I^2L technology is not in the mainstream of the IC industry.

The NMOS SLA cell set solved the major problems of the I^2L SLA. Dynamic and static cell types were investigated, but static ratioed logic was found to be more versatile and useful. The result was higher power consumption, restricting the ultimate size of the circuit. The widespread use of NMOS processing in the IC industry would justify the development of a low-power NMOS SLA cell set.

Five characteristics of the cell set summarize the functionality of the CMOS SLA: 1) the S, R, I, 0, and 1 combinational elements are all identical Schottky diodes which result in very dense structures, 2) the memory elements are simple cross-coupled set/reset NAND latches without read/write-enables, 3) more complex SLA structures are built from these primitive SLA cells rather than being predefined as was the case in SLA implementations in other technologies, 4) the elements are basically asynchronous and clocking is accomplished by ANDing clock signals on array rows, and 5) the power bus structure is arranged in horizontal bands rather than in global sets of vertical wires.

The generation of a medium to large size subsystem (the USCM) demonstrated the advantages of the CMOS SLA cell set over previous SLA implementations. It showed that macro elements could be effectively used in the SLA context. Flexibility was noted in different configurations of the same functional macro (MSFF's) and in implementing subcircuits which were difficult if not impossible to build in previous SLA implementations (i.e., dynamic shift registers). Using the lower level CMOS SLA cell set could give the SLA circuit designer (programmer) greater flexibility in writing programs. Working typically at a high level of abstraction with macros such as MSFF's, he retains the ability to drop down to a more primitive level when desirable.

Acknowledgment

The authors gratefully acknowledge the contributions of Dr. S. S. Patil, a former faculty member at the University of Utah and presently with Patil Systems, Inc., L. Hill and W. Knapp at General Instrument Corporate Research and Development Center, and H. Waldron and G. B. Goates at Boeing Aerospace Co.

References

[1] J. Carey and B. Blood, "Macrocell arrays—An alternative to custom LSI," in *Proc. Semi-Custom Integrated Circuit Technology Symp.* (Institute for Defense Analysis, Science and Technology Division), pp. 19-37, May 1981.

[2] G. B. Goates, H. M. Waldron III, S. S. Patil, K. F. Smith, and J. A. Tatman, "Storage/logic arrays for VHSIC," in *Proc. of Semi-Custom Integrated Circuit Technology Symp.* (Institute for Defense Analysis, Science and Technology Division), pp. 191-207, May 1981.

[3] D. L. Greer, "An associative logic matrix," *IEEE J. Solid-State Circuits*, vol. SC-11, pp. 679-691, Oct. 1976.

[4] A. B. Hayes, "Stored state asynchronous sequential circuits," *IEEE Trans. Comput.*, vol. C-30, pp. 596-600, Aug. 1981.

[5] H. K. Hingarh and B. Marshall, "Advanced gate arrays offering power/delay tradeoffs," in *Proc. Semi-Custom Integrated Circuit Technology Symp.* (Institute for Defense Analysis, Science and Technology Division), pp. 39-53, May 1981.

[6] L. A. Hollaar, "Direct implementation of asynchronous control units, submitted to *IEEE Trans. Comput.*

[7] D. Johannsen, "Bristle blocks: A silicon compiler," in *Proc. 16th Annual Design Automation Conf.* (ACM Special Interest Group on Design Automation and IEEE Computer Society), pp. 310-313, June, 1979. Also published in *Proc. Caltech Conf. on Very Large Scale Integration*, pp. 303-310. Also available as Silicon Structures Project File 2587, Computer Science Department, California Institute of Technology, Jan. 23, 1978.

[8] B. Lattin, "VLSI design methodology: The problem of the 80's for microprocessor design," in *Proc. 1979 Caltech Conf. on Very Large Scale Integration* (Caltech Computer Science Dept.), pp. 247-252, Jan. 1979.

[9] E. S. Lin, "A study of loading constraints of existing integrated injection logic realizations of storage/logic arrays," M.Sc. thesis, Department of Computer Science, Univ. of Utah, Salt Lake City, Aug. 1980.

[10] C. A. Mead and L. A. Conway, *Introduction to VLSI Systems.* Boston, MA: Addison Wesley, 1980.

[11] C. A. Mead, "VLSI and technological innovation," a testimony

on June 27, 1979, before the Senate Subcommittee on Science, Technology and Space in Washington, D.C. Refer to Senate bill 1250, The National Technical Innovation Act of 1979.

[12] R. M. Orndorff, M. M. Spanish, G. D. Gebhard, and D. M. Mitchell, "Advanced CMOS/SOS gate array technology," in *Proc. Semi-Custom Integrated Circuit Technology Symp.* (Institute for Defense Analysis, Science and Technology Division), pp. 221-240, May 1981.

[13] S. S. Patil, "An asynchronous logic array," Project MAC Tech. Memo TM-62, Massachusetts Institute of Technology, Cambridge, May 1975.

[14] S. S. Patil and T. A. Welch, "A programmable logic approach for VLSI," *IEEE Trans. Comput.*, vol. C-28, pp. 594-601, Sept. 1979.

[15] K. F. Smith, "Design of stored logic arrays in I^2L," in *Proc. 1981 IEEE International Symp. on Circuits and Systems* (IEEE Circuits and Systems Society), pp. 105-110, Apr. 1981.

[16] —, "Implementation of SLA's in NMOS technology," in *Proc. VLSI 81 Internat. Conf.* (Edinburgh, UK), pp. 247-256, Aug. 1981.

[17] K. F. Smith, T. M. Carter, and C. E. Hunt, "The CMOS SLA and SLA program structures, in H. T. Kung, B. Sproull, and G. Steele, Eds. *Proceedings of the 1981 CMU Conference on VLSI Systems and Computations* (Computer Science Dep., Carnegie-Mellon Univ). Pittsburgh, PA: Computer Science Press, Oct. 1981. pp. 396-407.

[18] R. A. Wood, "A high density programmable logic array chip," *IEEE Trans. Comput.*, vol. C-28, pp. 602-608, Sept. 1979.

Briefs

A Closed-Form Threshold Voltage Expression for a Small-Geometry MOSFET

L. A. AKERS AND C. S. CHAO

Abstract—**An analytical expression is developed to predict the threshold voltage of a small-geometry MOSFET. The expression includes the effects of field doping encroachment at the channel edges, and charge sharing with the source and drain regions.**

I. Introduction

As MOS devices are scaled down to near and submicrometer dimensions, geometry effects resulting from this scaling are a source of device parameter variation. Geometry effects include the short-channel, narrow-width, and small-geometry effects. These effects are modeled to allow an accurate prediction of the threshold voltage, an important design parameter.

The behavior of short-channel and narrow-width MOS devices has been investigated by several authors [1]-[7]. Simple yet elegant geometrical arguments to complete two-dimensional exact computer models have been used to simulate their behavior.

A small-geometry MOSFET is defined as a device with a channel length the same order of magnitude as the junction depletion depth, and the channel width the same order of magnitude as the channel depletion depth. For such small devices, neither just the short-channel nor the narrow-width expression is sufficient to accurately predict the threshold voltage due to the coupling of these effects [8]. Merckel [9] and Akers [10] have developed small-geometry MOSFET models which include the length-width coupling but these models do not include the effects of channel doping encroachment in the width direction or a tapered oxide.

This brief discusses the derivation of a closed-form analytical expression for the threshold voltage of a small-geometry MOSFET. The expression includes the effects of a tapered oxide, field doping encroachment at the channel edges, and charge sharing with the source and drain regions.

Manuscript received July 1, 1981; revised December 4, 1981. This work was supported in part by the National Science Foundation under Grant ECS-8109198.

The authors are with the Department of Electrical and Computer Engineering, Arizona State University, Tempe, AZ 85281.

II. Derivation of Model

The threshold voltage for a large-geometry MOSFET may be expressed as

$$V_T = V_{FB} + 2\,\phi_B + Q_B/C_{ox} \tag{1}$$

where the symbols have their normal meaning.

Equation (1) is valid as long as the channel length is long compared to the source and drain junction depletion depths and the width is wide compared to the depth of the channel depletion region. If these conditions are not valid, the bulk charge in the depletion region must be appropriately modified.

The short-channel effect, the decrease in the threshold voltage V_T as the channel length L is reduced, results from the charge shared by the drain and source depletion regions and the channel depletion region. The amount of bulk charge associated with the gate electrode for a drain voltage equal zero was modeled following Yau [2].

Fig. 1 illustrates a width cross section in the middle of the length of a MOSFET. The thick oxide on both sides does not change abruptly to the thin gate oxide, but is tapered and re-