The CMOS SLA Implementation and SLA Program Structures

K.F. Smith, T.M. Carter, and C.E. Hunt
University of Utah
Department of Computer Science
Salt Lake City, Utah 84112

INTRODUCTION - THE SLA CONCEPT

The storage/logic array (SLA) is a form of structured logic which is well suited to VLSI design. The SLA concept, which was derived from the PLA, was originally conceived by Patil [3] and later elaborated upon by Patil and Welch [4]. The SLA differs from the PLA in several major respects. The SLA has both the AND and the OR planes from the PLA, but these planes are superimposed or folded on top of each other. This folding of the AND and OR planes generates a structure in which AND terms are generated on the rows of the SLA and the OR terms are generated on the columns. The single AND/OR plane of the SLA contains column wires which can serve as inputs to the SLA rows (AND plane) or as outputs from the OR plane. This functional duality of SLA columns means not only that the SLA can be arbitrarily segmented, but that inputs to and outputs from segments of the SLA can be arbitrarily interleaved.

Also due to the functional duality of the SLA columns, the SLA can contain memory elements imbedded within its structure which merges feedback loops into the array itself. This allows for the specification and implementation of independent finite state machines and data path modules within a single integrated structure. In addition to memory, inverters or other standard logic gates can be placed in the SLA to provide multiple levels of logic, whereas the conventional PLA can only generate one level of AND data and one level of OR data.

Adding row breaks placed between adjacent columns and column breaks placed between adjacent rows allows great flexibility in segmenting the array. Segments of the array need not be rectangular but may be polygonal (where the polygon has orthogonal sides).

Aside from the physical advantages and flexibility of the SLA, it has several logical and design automation advantages. The symbolic nature of the SLA program specification gives the circuit designer an immediate perception of the logical function of the circuit being designed. Each SLA logic symbol maps directly onto a member of the SLA cell set, giving the SLA designer a simultaneous perception of both the logical function of the SLA and its physical layout. Given a set of established SLA cells and rules for using them, a circuit designer can, for the most part, ignore both the electronics of the circuit and the layout while concentrating on the logical function.

The SLA should ideally be technology independent. That is, one program should be transferable, without change, between different

H. T. Kung et al. (eds.), *VLSI Systems and Computations*

processes. Initial experience with I2L and NMOS SLA implementations [5, 6] 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. Specifically, it was shown that I2L [2] cannot adequately implement large gates on the SLA rows or columns. An NMOS implementation proved to be able to handle large row and column gates, but at the price of high power consumption. Folding the AND and OR planes in the I2L and NMOS SLA implementations has resulted in relatively poor space utilization. The new CMOS SLA overcomes the gate size, power, and space problems encountered in I2L and NMOS. The CMOS SLA uses Schottky diodes as the combinational logic elements in both the AND and OR planes and thus significantly increases packing density of combinational logic over both the I2L and the NMOS SLAs. The CMOS SLA will also have the speed and driving capability of a conventional CMOS circuit but will consume only about one-fourth of the power consumed in NMOS SLAs. The packing density will be comparable to that of an NMOS ROM.

THE CMOS SLA CELL SET

The CMOS SLA Cell Set contains elements which have been present in all previous SLA implementations. These elements may not be implemented with exactly the same functionality in CMOS, but can be used to write SLA programs which are functionally equivalent to those in NMOS or I2L. In addition to the "standard" SLA elements, new ones have been added which greatly increase the flexibility and power of the CMOS SLA Cell Set. The CMOS SLA Cell Set includes:

- memory elements (flip-flops [latches] composed of cross-coupled NAND gates),
- inverters (both a single inverter and two inverters in series, generating both the true and not-true of the input signal),
- elements which act on memory elements and inverters:
  - S (set a flip-flop)
  - R (reset a flip-flop)
  - I (inverter input),
- elements which detect the state of memory elements and inverters which are driving onto a column:
  - 0 (detect the reset state of a flip-flop or the false output from an inverter)
  - 1 (detect the set state of a flip-flop or the true output from two inverters in series)