

# Top-Down Pass-Transistor Logic Design

Kazuo Yano, *Associate Member, IEEE*, Yasuhiko Sasaki, *Member, IEEE*, Kunihiro Rikino,  
and Koichi Seki, *Member, IEEE*

**Abstract**—The pass-transistor based cell library and synthesis tool are constructed, for the first time, to clarify the potential of top-down pass-transistor logic. The entire scheme is called LEAP (*Lean Integration with Pass-Transistors*). The feature of a pass-transistor based cell is its multiplexer function and the open-drain structure. This cell has the flexibility of transistor-level circuit design and compatibility with conventional cell-based design. An extremely simple cell library with only seven cells combined with a synthesis tool called “circuit inventor” is compared with the conventional CMOS library that has over 60 cells combined with the state-of-the-art logic synthesis. The results show that the area, delay, and power dissipation are improved by LEAP and that the value-cost ratio is improved by a factor of three. This demonstrates that LEAP has the potential to achieve a quantum leap in value of LSI’s while reducing the cost. Key issues which have to be cleared before pass-transistor logic is used as the generic logic scheme replacing CMOS are also discussed.

## I. INTRODUCTION

UNDER intensive competition in the microprocessor, microcontroller, ASIC’s, and related logic marketplace, differentiation and gaining competitive advantages in performance and cost is crucial for a chip (or even a firm) to survive. To gain such advantages, new circuit techniques which go beyond conventional CMOS circuitry have been given a great deal of attention. These include pass-transistor based logics [1]–[4], domino-like dynamic logics [5], [6], and dual-rail logic circuits [1]–[4], [6].

Among these, nMOS-based pass-transistor circuits have a great potential to surpass the CMOS not only in performance but also in area and power [1]–[4], [7]. The fact that such an nMOS pass-transistor has long been an indispensable element in the memory cells of DRAM and SRAM, acting to keep the cost of memory chips low, is strong evidence for the advantages of pass transistors in cost reduction. Therefore, one is naturally tempted to apply them to logics to gain similar advantages.

However, in real logic LSI’s the pass transistors have not succeeded in capturing a major role, but they are used only in a small portion of arithmetic macros or XOR logic. The reason for this may be as follows. First, there is no established design methodology for forming a general logic function. Indeed, one can arrange a pass-transistor

circuit which functions as (N)AND or (N)OR, and therefore, general logic functions can be achieved by combining these gates [1], [8]. However, this simple approach does not exploit the full functionality of pass transistors and gives inferior results to conventional CMOS logics. So far, no method which fundamentally changes this situation has been found. There are some exceptional logic functions, for which a clever pass-transistor configuration has already been devised (e.g., the full-adder circuit described in [1]). However, such a panacea should not be anticipated in a time-constrained real logic-design process. The second important point, which is related to the first, is that neither a synthesis tool nor a cell library are available for pass-transistor based design. Because recently, so-called top-down design, which is based on a hardware description language (HDL) and logic synthesis, has been used extensively in logic design, particularly in control logic blocks, the lack of such tool support is a fatal disadvantage. The third point is that nMOS pass-transistor logics are generally believed to have poorer low-voltage performance than full static CMOS circuits, which may have raised doubts about future applicability.

This paper attempts to re-examine the possibility of using pass transistors in light of the top-down design—the new rule of the game. Although top-down design poses a hurdle for new circuits entering the game because of the necessity for various design tools and libraries, it also has positive aspects. Because the logic circuits are designed with a synthesis tool, not by a human designer, logic generation algorithms and circuits which might otherwise have been difficult to get logic designers acquainted with (particularly, under intense time-to-market pressure) are hidden in a design automation tool. This allows new ideas in both algorithms and circuits to be easily incorporated in a real logic design process once the tool is developed. In trying to fully utilize this degree of freedom, we developed the experimental top-down environment based on pass transistors (the preliminary results are reported in [9]). This new logic design scheme is called LEAP (*Lean Integration with Pass Transistors*). Although the final goal of this work is directed toward replacing most CMOS circuits with pass transistors in both arithmetic and control logics, this is a too ambitious objective for one paper to firmly confirm because too many issues are interrelated and they have to be clarified. Instead, we try to clarify the potential and limitations of this top-down pass-transistor logic design and to discuss key issues which one inevitably encountered when one exploits similar directions. We perform quantitative comparison on relatively

Manuscript received January 19, 1995; revised June 20, 1995.

K. Yano, Y. Sasaki, and K. Seki are with ULSI Research Department, Central Research Laboratory, Hitachi Ltd., Kokubunji, Tokyo 185, Japan. K. Rikino is with Hitachi Device Engineering, Kokubunji, Tokyo 185, Japan.

Publisher Item Identifier S 0018-9200(96)04205-9.

small benchmark circuits between our new design scheme and the conventional CMOS design scheme.

New cell-library architecture based on pass transistors is proposed in Section II, and the logic synthesis tool using the new library is introduced in Section III. Detailed benchmark test results are remaining issues that are discussed in Section IV, followed by conclusions in Section V.

## II. PASS-TRANSISTOR CELL LIBRARY

### A. Proposed Cell Scheme

The proposed top-down design scheme LEAP is compared with conventional CMOS in Fig. 1. A logic function is given in HDL form, and the corresponding pass-transistor circuits are synthesized. Then the mask layout pattern is generated by an automatic place-and-router. The cell library which supports logic synthesis is the key element in this scheme.

The proposed pass-transistor cell library is compared with a conventional CMOS library in Table I. Because the conventional CMOS library has more than 60 cells even if we exclude the sequential circuits, substantial engineering efforts are required to create and update the cell library [10]. The pass-transistor-cell library has only seven cells, far smaller than the conventional standard-cell library. The three of these are essential logic cells, Y1, Y2, and Y3 (Fig. 2). The other four are simple inverters with various drive capabilities.

Cell “Y1” has a transistor arrangement just like the letter “Y.” Its function is to select one of two inputs connected to the drain [data, A and B in Fig. 2(a)] depending on an input signal connected to the gate [control, C in Fig. 2(a)] and then to invert the selected signal. This is an inverted multiplexer. The two inputs, which are connected to the MOSFET gates, are designed to be complementary so as to avoid conflict between the signals. This is not a dual-rail logic, such as complementary pass-transistor logic (CPL [1]), in which any logic signal line is accompanied by a complemented signal line. Instead, a simple inverter cell is used to generate the complemented signal in this case. Therefore, complemented signal exists only in a local region between the inverter and the Y1. This complemented signal can be sometimes shared by multiple cells, which minimizes the overhead related to this addition of an inverter. Cells “Y2” and “Y3” have two or three stacked Y1 cells and process more complex logic functions.

### B. Comparison to Other Cell-Based Design Schemes

Although the new library does not include logic functions like NAND or NOR, which have played the central roles in the conventional libraries, combinations of multiplexers are enough to generate arbitrary logic functions. However, the way a given logic function is represented by a cell network is very different from the one using conventional NAND, NOR based libraries as illustrated in Fig. 3. To obtain NAND function, a Y2 cell and two in-

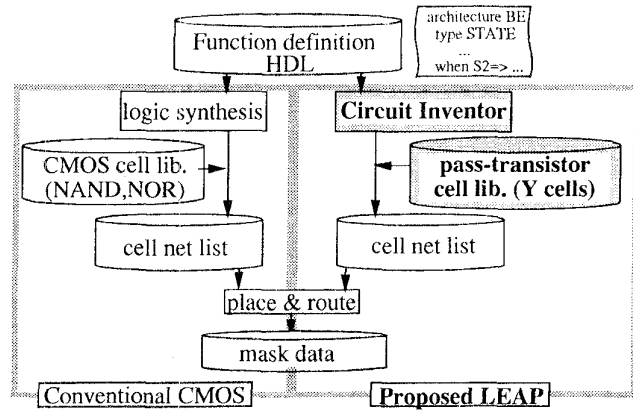
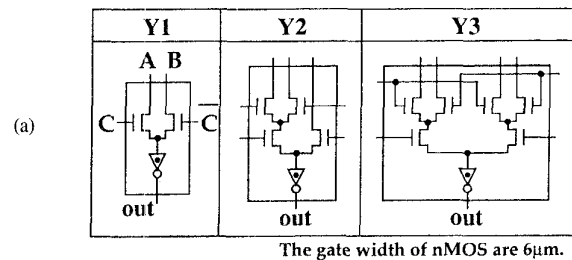


Fig. 1. Proposed top-down design flow, LEAP is compared with that of conventional CMOS design.

TABLE I  
CELL LIST OF THE CONVENTIONAL CMOS LIBRARY COMPARED TO THE PROPOSED PASS-TRANSISTOR-CELL LIBRARY

Conventional CMOS Library (61 cells)			Pass-Transistor Cell Library (7 cells)
INVERTER	4AND	2OR/2AND	Y1
INVERTER_P2	4AND_P2	2OR/2AND_P2	Y2
INVERTER_P4	2OR	3OR/2AND	Y3
INVERTER_P8	2OR_P2	3OR/2AND_P2	INVERTER
2NAND	3OR	2ANDx2/2OR	INVERTER_P2
3NAND	3OR_P2	2ANDx2/2OR_P2	INVERTER_P4
2NOR	4OR	3ANDx2/2OR	INVERTER_P8
2NOR_P2	4OR_P2	3ANDx2/2OR_P2	
3NOR	2AND/2NOR	2ANDx3/3OR	
3NOR_P2	2ANDx2/2NOR	2ANDx3/3OR_P2	
4NOR	3AND/3NOR	2ANDx2/3OR	
4NOR_P2	2OR/2NAND	2ANDx2/3OR_P2	
2XOR	2OR/3NAND	2ORx2/2AND	
2XOR_P2	3OR/2NAND	2ORx2/2AND_P2	
2XNOR	2AND/2OR	2ANDx4/4OR	
2XNOR_P2	2AND/2OR_P2	2ANDx4/4OR_P2	
2AND	3AND/2OR	8NAND	
2AND_P2	3AND/2OR_P2	8NAND_P2	
3AND	2AND/3OR	8AND	
3AND_P2	2AND/3OR_P2	8AND_P2	
		8OR	

(\_P2, \_P4, \_P8 represent x2, x4, x8 powered cells)



The gate width of nMOS are 6μm.

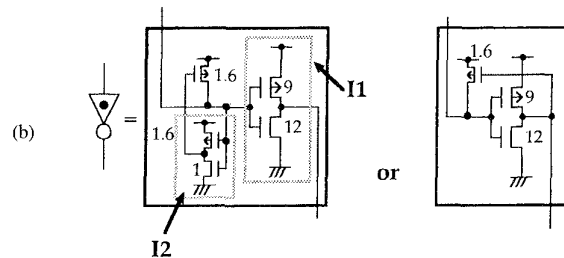


Fig. 2. (a) Circuit diagram of pass-transistor cells. (b) The output inverters. The left circuit of (b) is used when the output load can be very large, and the right is used when the output load capacitance is relatively small.

Typical Cell	Conventional CMOS Cell	Y-Cell (This Work)	Alternatives		
			Boolean-Function-Based Pass-Transistor Cell (Pasternak, et al [8])	CPL Based Y-Cell	MUX-Based CMOS Cell
1-input NAND $S = \overline{A \cdot B \cdot C}$					
	Function	$S = \overline{A \cdot B \cdot C}$	$S = \overline{A \cdot B \cdot C}$	$S = \overline{C \cdot (C \cdot D1 + \overline{C1} \cdot D2) + \overline{C2} \cdot D3}$	$S = \overline{C \cdot D1 + \overline{C} \cdot D2}$
	Circuit Diagram				
	Tr. Count	6 (1)	13 (2,17)	14 (2,33)	22 (3,67)
	Cell Count	1	3	1	3
$S = \overline{A \cdot B} + \overline{B \cdot C} + \overline{C \cdot A}$	Circuit Diagram				
	Tr. Count	16 (1)	13 (0.81)	30 (1.88)	14 (0.88)
	Cell Count	4	3	4	1
	Area	852 $\mu\text{m}^2$ (1)	579 $\mu\text{m}^2$ (0.68)	1158 $\mu\text{m}^2$ (1.36)	774 $\mu\text{m}^2$ (0.91)
	Delay	652 ps (1)	465 ps (0.71)	877 ps (1.35)	371 ps (0.57)
	Power	1.81 $\mu\text{W}/\text{MHz}$ (1)	0.96 $\mu\text{W}/\text{MHz}$ (0.53)	2.17 $\mu\text{W}/\text{MHz}$ (1.20)	1.35 $\mu\text{W}/\text{MHz}$ (0.75)
3-input NAND $S = \overline{A \cdot B \cdot C}$	Circuit Diagram				
	Tr. Count	6 (1)	13 (2,17)	14 (2,33)	22 (3,67)
	Cell Count	1	3	1	3
	Area	329 $\mu\text{m}^2$ (1)	579 $\mu\text{m}^2$ (1.75)	483 $\mu\text{m}^2$ (1.47)	774 $\mu\text{m}^2$ (2.35)
	Delay	295 ps (1)	465 ps (1.58)	465 ps (1.58)	371 ps (1.25)
	Power	0.91 $\mu\text{W}/\text{MHz}$ (1)	0.96 $\mu\text{W}/\text{MHz}$ (1.05)	0.96 $\mu\text{W}/\text{MHz}$ (1.05)	1.35 $\mu\text{W}/\text{MHz}$ (1.41)
$S = \overline{A \cdot B} + \overline{B \cdot C} + \overline{C \cdot A}$	Circuit Diagram				
	Tr. Count	16 (1)	13 (0.81)	30 (1.88)	14 (0.88)
	Cell Count	4	3	4	1
	Area	852 $\mu\text{m}^2$ (1)	579 $\mu\text{m}^2$ (0.68)	1158 $\mu\text{m}^2$ (1.36)	774 $\mu\text{m}^2$ (0.91)
	Delay	652 ps (1)	465 ps (0.71)	877 ps (1.35)	371 ps (0.57)
	Power	1.81 $\mu\text{W}/\text{MHz}$ (1)	0.96 $\mu\text{W}/\text{MHz}$ (0.53)	2.17 $\mu\text{W}/\text{MHz}$ (1.20)	1.35 $\mu\text{W}/\text{MHz}$ (0.75)

Fig. 3. Comparison among various cell architectures. The area, delay, and power are evaluated based on our 0.6  $\mu\text{m}$  CMOS technology. The gate width of nMOS's and pMOS's used in the CMOS NAND are 10.5  $\mu\text{m}$ . The gate width of the nMOS and pMOS in the CMOS inverters are 3.5  $\mu\text{m}$  and 5  $\mu\text{m}$ , respectively. The gate width of the other nMOS's and pMOS's used in the other CMOS circuits is 6  $\mu\text{m}$  and 12  $\mu\text{m}$ , respectively. The gate width of the output inverters in Y-cells is shown in Fig. 2. The load capacitance is set to 100 fF in delay evaluations.

verters are required with our library, whereas a simple NAND cell is enough with the conventional CMOS library. The area and delay of the Y-cell based design is worse than the simple CMOS circuit for NAND. On the other hand, when one forms another function ( $S = \overline{A \cdot B} + \overline{B \cdot C} + \overline{C \cdot A}$ ) shown in Fig. 3, still a Y2 cell and two inverters are enough with our new library, whereas two OR/NAND cells and two inverters are required with the CMOS library. The area and delay of the Y-cell based design in this case is better than the CMOS based design. These results suggest that gate-level comparison between these two cell architectures is difficult.

The reason we did not use the CPL [1] in this work is simply to minimize the area (as shown in Fig. 3), which, we believe, is important for pass-transistor logics to be widely used. However, the high-performance aspect of the CPL shown in Fig. 3 is attractive and may be useful in some high-performance applications, such as in a multiplier.

An alternative is to prepare a usual Boolean "standard-cell" function using pass transistors and construct a given logic function using these cells just like CMOS (Fig. 3). Similar idea based on CPL is shown in [1] and one based on single-ended pass transistors is suggested in [8] and [22]. However, as is clear in the figure, this always gives

inferior results to CMOS and is not practical. The reason is that the input inverters and the output inverters makes the area large.

We also examined another approach, in which the multiplexer is configured by a clocked CMOS inverter circuit. However, as is shown in Fig. 3, the area and the delay is much worse than the other approaches and this approach is not practical. This result also shows that the advantages of the present Y-cell library comes from the transistor-level design, not from the multiplexer-based cell function. This is the reason why we did not refer to our library as "multiplexer-based cell library."

However, this shift of the logic primitive from the NAND (or NOR) to the multiplexer poses a major challenge (or hurdle) against logic synthesis because conventional logic syntheses have been based on Boolean operations. Fig. 4 shows how a simple change of the input configuration of the simple Y2 cell corresponds to very different Boolean functions. All these functions, which is complex in the Boolean-based representation, rarely appear in synthesized Boolean expressions of a real logic block. This is because there is a fundamental difference between a multiplexer and Boolean functions such as NAND. This can be illustrated by the fact that at least three input variables are related to form the simplest mul-

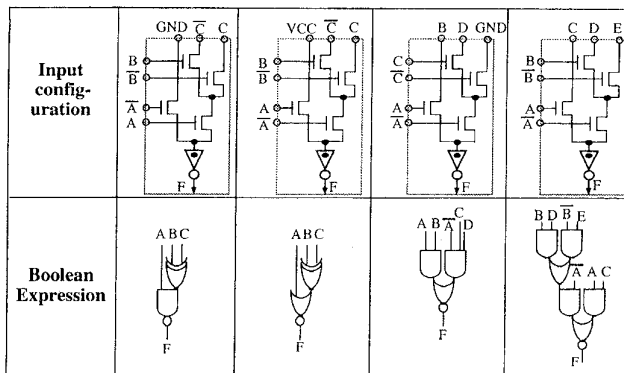


Fig. 4. Various logic functions of the cell "Y2" in Boolean representation.

tiplexing function (one for gate and two for drains), whereas two input variables are enough to form the simplest Boolean logic (more complex functions are constructed by combining these gates). These facts imply that a completely new logic synthesis methodology has to be established if one wants to extract the full potential of pass-transistor cells. Although we have this major difference, we would like to claim in this paper that the synthesis based on multiplexers is possible and the generated circuits have quite competitive figures compared to the state-of-the-art CMOS/Boolean based logic synthesis. This issue will be discussed in Section III again.

### C. Other Features of New Cells

The other feature of Y-cells is that it is compatible with the conventional framework of cell-based design. Most design tools except logic synthesizer can be used without a change or with a minor change. The delay of a Y-cell is specified as a function of the load capacitance just as in conventional CMOS cells as shown in Fig. 5. Therefore, logic simulation and delay check is performed just as the conventional standard-cell design. This compatibility is brought about by the output inverter which separates the inputs from the output. A feedback inverter and pull-up pMOS, both consisting of minimum-size MOSFET's, are added to avoid DC leakage current in the CMOS inverter (Fig. 2). Although the feedback to the pull-up pMOS can be taken from the output of the main inverter I1 [the right-hand side of Fig. 2(b)], we have decided to add the minimum-sized inverter I2 in the standard cell based comparison shown later. This is because, in this configuration, the timing of the feedback signal is stable no matter how large the load capacitance is. Therefore, this circuit works stably even if the load capacitance of a cell is extremely large in an automatically routed very large random-logic block. However, when the design process is such that one can control the load capacitance of a cell within an allowable range (this is relatively easy in a regularly structured arithmetic module), it is better to delete I2 to reduce area.

Another advantage of the pass-transistor cells is its small number of cells in the library. Because preparing

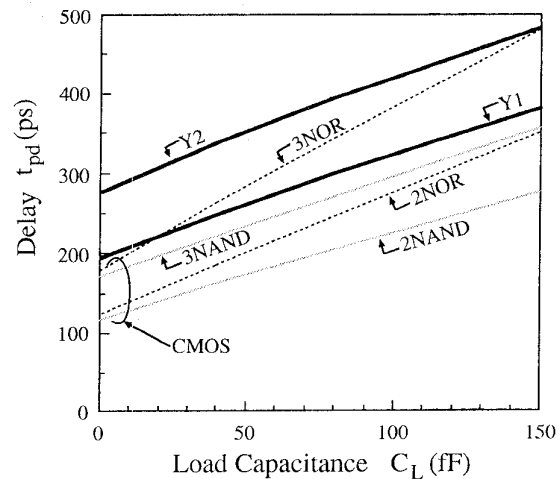


Fig. 5. The dependence of the delay time on load capacitance for pass-transistor cell Y2, Y3, CMOS two-input NAND, and three-input NAND. The delay is the average of the pull-down delay and the pull-up delay.

and updating the pass-transistor-cell library requires small engineering costs, it is easier to adopt state-of-the-art process technology even under a tight schedule constraint. It should be noted that the integration capacity of a ULSI is improving at a rate of doubling per two years. In this way, the pass-transistor cell is expected to encourage concurrent interaction between a logic design and the state-of-the-art process and devices. The cost of preparing and updating precise cell-library data has recently become a serious burden in LSI design, but creating opportunities for companies to provide library-design service [10]. Major revisions in the conventional library, which includes cell-layout data, logic-synthesizer data, and automatic place and router data for more than 60 cells, requires much more engineering effort and is sometimes unrealistic. Pass-transistor cells, with their smaller library, are favorable in this respect.

### III. CIRCUIT INVENTOR

We have developed a logic/circuit synthesis tool called "circuit inventor," which fully utilizes the pass-transistor cell characteristics. Finding a good pass-transistor circuit for a given logic function has been just like an invention. We chose the name circuit inventor because it is anticipated to perform the job previously done by the inventor. The circuit inventor accepts an HDL description, creates net lists based on pass-transistor cells, and gives those data to the layout tool. Automatic placing and routing are done with conventional tools.

Constructing a logic function using pass-transistor cells can be accomplished by constructing logics based on two-input multiplexers. A simple idea is to use the Boolean expression generated by the conventional multilevel Boolean-equation based synthesis and maps to the multiplexers. This mapping of a Boolean equation to multiplexer cells has recently been a subject of extensive study [20], [21], because of the emergence of field-programmable

gate arrays based on multiplexer primitive cell. However, this approach gives a much larger area than the use of conventional standard cell as is illustrated in Fig. 3. This is natural because literal count, which has been minimized in a Boolean-expression based synthesis, is not a good metric for area estimation with pass-transistor circuits. So we have abandoned this approach. Historically, there have been some other attempts to use multiplexers to form a general logic [12], [13]. This is based on the well-known fact that the  $k$ -control-signal multiplexer functions as arbitrary  $(k + 1)$ -input logic. This is based on Shannon expansion [14], which corresponds to a binary tree of the multiplexers [13]. Manipulation of this binary tree, called the binary decision diagram (BDD) [15], [16], is attracting attention due to other motivations in the design automation community: BDD has been known to be an efficient data structure in DA program to handle logic functions and proven useful in formal verification applications. The techniques used to manipulate binary trees have been intensively studied [16], [17], revealing that the size of a binary tree is reduced by avoiding redundant nodes of BDD (this corresponds to multiplexers) or by sharing an equivalent binary tree. The cases where redundant pass-transistor reduction is possible are summarized in Fig. 6(a). These techniques are incorporated in Circuit Inventor.

The basic idea of the way the Circuit Inventor works is shown in Fig. 6(b). First, it expresses the required logic function in a compact form using a reduced and shared BDD [2]. Then the BDD or binary tree is partitioned into smaller trees delimited by buffers. The node where the buffer is inserted is determined based on the following considerations. The height of the partitioned tree should not exceed the given maximum allowable tree height. In the cell-based design shown in Figs. 1 and 2, the height is given by the maximum pass-transistor height of a cell. In the case of the library shown in Fig. 2, the maximum height is two. The maximum tree height can be increased by increasing the number of cells. However, it should be considered that there is a considerable delay of long series-pass-transistor chain because of the quadratic dependence of delay on the number of series stages. After buffer insertion, the binary tree is mapped to the cells. In this straightforward mapping the tree has the same network topology as the pass-transistor cells. However, the inverters in the network reverse the polarity of the logic value. So the polarity is propagated from outputs to inputs. At heavily loaded nodes, larger inverters are inserted to improve speed. Inverters which provide complemented signals to one of a pass-transistor pair are also added.

This circuit inventor is a very simple experimental version and clearly has a limitation. The number of pass transistors for a signal to pass, in the worst case, reaches the number of the inputs of the given logic function. Taking an adder as an example, this corresponds to a ripple-carry adder, which is known to be slower than speed-oriented architectures, such as a carry-look-ahead or carry-select

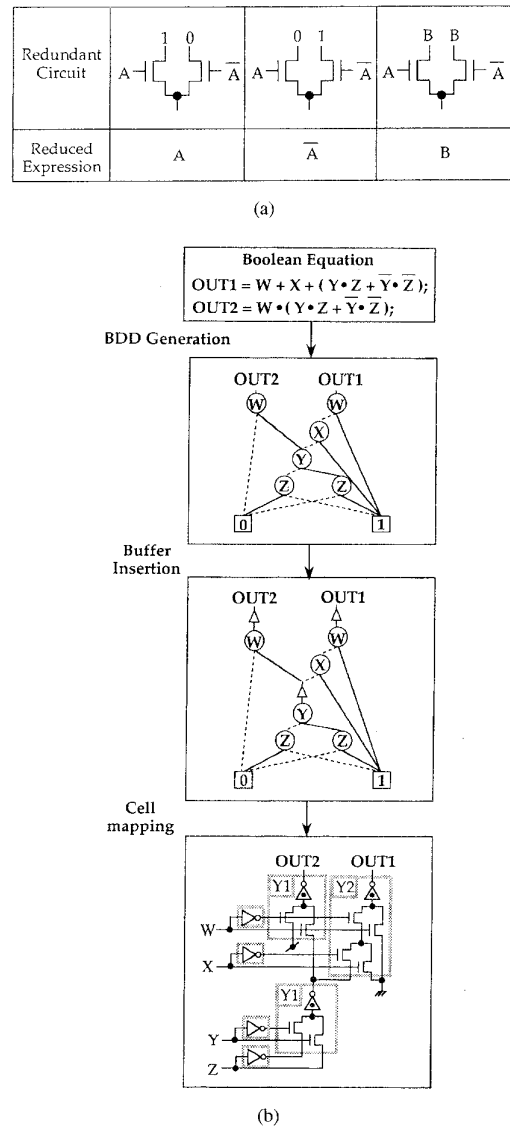


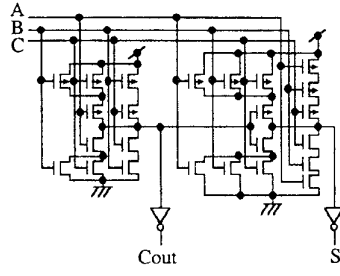
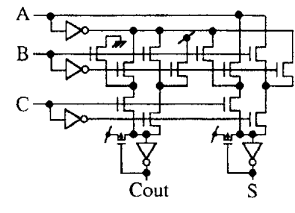
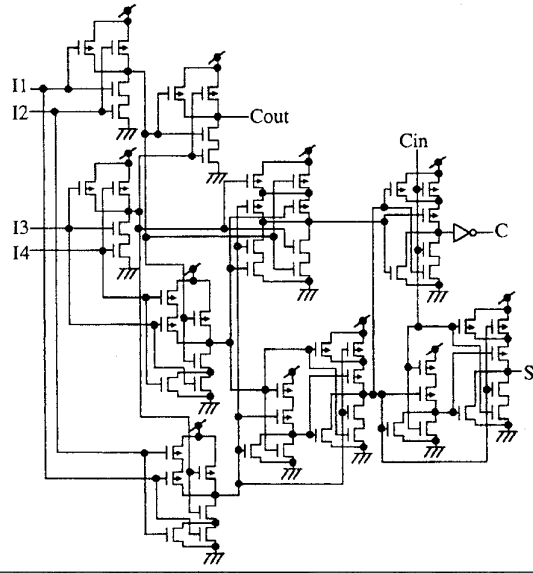
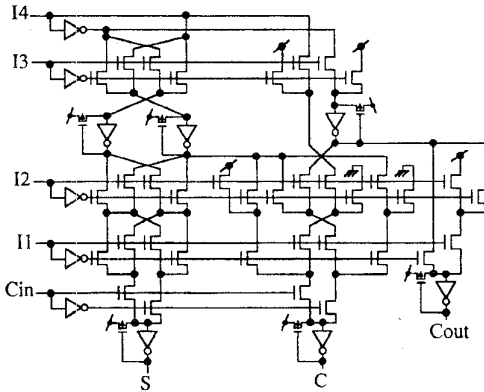
Fig. 6. (a) The basic elimination process of redundant pass transistors. (b) The synthesis flow of the circuit inventor.

adder, in which the delay is proportional to the logarithm of the number of the inputs. We confine our following evaluations to small circuits having less than ten inputs to avoid this effect. This limitation may be removed in the future by adding optimization capabilities, which as been commonly incorporated in conventional logic syntheses.

#### IV. EXPERIMENTS AND DISCUSSION

##### A. Comparison with Reported Circuits

First, we compared circuits generated by the circuit inventor with the circuits invented by a human to check the synthesis quality. We chose a full adder and a 4-2 compressor as the test vehicle—both are important components for fast arithmetic blocks, as evidenced by the many circuit configurations reported for these functions. We chose the full adder proposed by Montoye [18] and the

		Reported Circuits [18, 19]	Circuit Inventor (This Work)
Full Adder	Circuit Diagram		
	Tr. Count	28 (1)	24 (0.86)
	Area	1810 $\mu\text{m}^2$ (1)	1161 $\mu\text{m}^2$ (0.64)
	Delay	0.73 ns (1)	0.32 ns (0.44)
	Power	3.65 $\mu\text{W}/\text{MHz}$ (1)	2.35 $\mu\text{W}/\text{MHz}$ (0.64)
	FOM*	1	6.5
4-2 Compressor	Circuit Diagram		
	Tr. Count	58 (1)	64 (1.10)
	Area	3620 $\mu\text{m}^2$ (1)	3340 $\mu\text{m}^2$ (0.92)
	Delay	1.08 ns (1)	0.81 ns (0.75)
	Power	7.83 $\mu\text{W}/\text{MHz}$ (1)	6.19 $\mu\text{W}/\text{MHz}$ (0.79)
	FOM	1	1.83

\*Figure of merit is defined as the inverse of the product of area, delay and power.

Fig. 7. Comparison of the circuits synthesized by the circuit inventor with the circuits reported in [18] and [19].

4-2 compressor proposed by Goto [19] for this comparison, because these circuits offer good balance between transistor count and delay. The delay is compared using circuit simulation based on half-micron CMOS technology at a 3.3-V supply. In the output inverter, the sub-inverter (I2 in Fig. 2) was omitted and the feedback signal to the pMOS was taken from the main inverter (I1). This is because the load capacitance of these circuits in an arithmetic module is easily controlled within a range, in which I2 is unnecessary. In this comparison, the cell-based design procedure was not used for both CMOS and LEAP; instead a transistor-based design was used. Here we used an option of the circuit inventor for generating transistor-based net list. The maximum tree height of LEAP is set to four in this comparison.

The results (shown in Fig. 7) clearly demonstrate the excellent quality of tool synthesized circuits for these relatively small arithmetic circuits. For both functions, the Circuit Inventor generates better circuits in terms of area, delay time, and power dissipation. This result is somewhat surprising when one recalls that this version of Circuit Inventor does not have any optimization capability, and in this sense the synthesis itself is far less sophisticated than the state-of-the-art synthesis program. This is encouraging for one who tries to use the circuit inventor to aid his/her design of hand-crafted modules. As will be discussed in the next section, the larger arithmetic circuits generated by the present circuit inventor are still inferior to the best human-designed circuits (the reason may be the hierarchical partitioning of a logic is impossible in this

present version). Even so, it should be noted that in real arithmetic design process, an entire arithmetic block (ALU, multiplier, or any new functions) can usually be divided into smaller components whose sizes are similar to that of a full-adder or a 4-2 compressor. So the present simple version is quite helpful for a designer, who intends to have a quality circuit for such components in a short time.

If we look more closely at the synthesized circuit structure, we get a feeling as to how reaching a circuit for pass-transistor cells is difficult if one is based on Boolean representation. The carry output of the full adder ( $C_{out}$ ) in the synthesized circuit can be directly translated to Boolean expression

$$C_{out} = C(B1 + \bar{B}A) + \bar{C}(BA + \bar{B}0) \quad (1)$$

which is simplified to

$$C_{out} = C(A + B) + AB\bar{C}. \quad (2)$$

However, this is different from a natural expression of the majority-rule logic

$$C_{out} = AB + BC + CA. \quad (3)$$

Transforming (2) to (3) requires some tricks as follows. By using the relation  $A + B = A + B + AB$ , we obtain

$$C_{out} = C(A + B + AB) + \bar{C}AB. \quad (4)$$

This is simplified to (3) if we use  $C + \bar{C} = 1$ .

### B. Comparison with Conventional Top-Down Design

Currently, logic synthesis is mainly used for blocks, in which turn-around time of a design is critical, e.g., ASIC's and control blocks of MPU's. However, pass-transistors have not been widely used in these blocks. If the top-down pass-transistor design is to be used in such logics, the LEAP scheme should have an advantage over the state-of-the-art CMOS-based logic synthesis scheme. To exploit this possibility, we compared LEAP to the state-of-the-art CMOS synthesis in terms of cell-library based design.

Two types of benchmark logic functions are chosen. One is a 4-b adder/subtractor in absolute-value representation (both the delay and the area are about 50% larger than the simple signed-value subtracter). This is a typical arithmetic logic and the number of inputs is ten and the number of outputs is five. The other is seven-input four-output random logic, created by assigning random numbers to the output of the truth table. CMOS logic is synthesized by using the most widely used state-of-the-art commercial logic synthesizer with area minimization option. A 0.5- $\mu\text{m}$  fabrication with three-level metal is assumed and a poly-cell-type layout style is used. Metal 1 is assigned to the intra-cell wiring, Metal 2 to Y-direction inter-cell wiring, and Metal 3 to the X-direction inter-cell wiring. The cell input/output ports are formed as through-holes between Metal 1 and Metal 2. The transistor size of the pass-transistor cell is as shown in Fig. 2. Here the

output inverter with subinverter (I2) is used. The typical transistor size of CMOS cells is as follows. The minimum gate width of nMOS is 3.5  $\mu\text{m}$ , whereas that of the pMOS is 5  $\mu\text{m}$ . The gate width of the two-input NAND is 10.5  $\mu\text{m}$  for nMOS and 10.5  $\mu\text{m}$  for pMOS. The gate width of the two-input NOR is 6  $\mu\text{m}$  for nMOS and 12  $\mu\text{m}$  for pMOS. All these sizes are comparable to those used in the pass-transistor cells, making the following comparison fair in this respect.

The critical path and wiring load was extracted from the layout data and the delay was obtained by circuit simulation. The power consumption was determined by circuit simulation of the total circuit without wiring capacitance. Because the power consumption of these small logic blocks is dominated by the gate and source/drain capacitances, this neglect of wire capacitance gives very minor modification but does not change the conclusion.

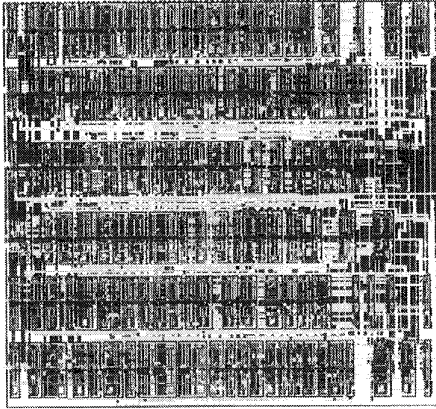
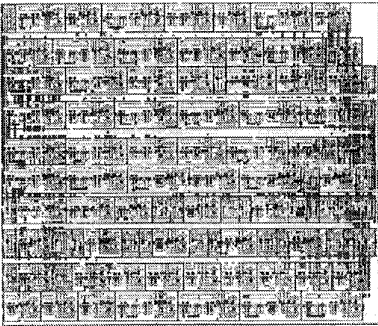
Dramatic results can be seen in Fig. 8. The synthesized critical path circuits are shown in Fig. 9. The LEAP is superior in all respects including area, delay, and power consumption for both benchmarks. If we define the figure-of-merit of a circuit by using the product of the area, delay, and power, the LEAP gives a three to four times higher figure. This is again surprising as one recalls that the current version of the circuit inventor does not have an optimization capability, whereas the compared CMOS synthesis is fully equipped with such a capability. In a conventional logic synthesis, the basic two-level logic optimization (such as sum-of-product form) is followed by decomposition and factorization, or multilevel logic optimization, which tunes the logic to the design constraints. In this sense the current version of Circuit Inventor (based on a simple BDD) is just like being limited to the conventional two-level optimization level, which implies that there remains much to be improved in optimizing the logic forms.

Next, let us more quantitatively analyze the key factors which have brought about the above results. The total area can be decomposed into the following three factors:

$$\text{Logic area} = \text{Net count} \times \frac{1}{\text{Nets per cell}} \times \text{Cell area} \quad (5)$$

where the "net count" is the total number of nets in a cell net list, "nets per cell" (NPC) is the average number of input ports per cell, and "cell area" is the average area per cell. In (5) a cell is considered to be a box (or compressor) which reduces the number of nets (inputs) to one (the output). Although these three factors are interrelated, the net count is mainly determined by the logic synthesis algorithm, the NPC by the cell architecture, and the cell area by the technology. These factors for the synthesized benchmark circuits are shown in Fig. 1 and Table II. The net count for LEAP is larger than that of the CMOS. This is mainly because complementary inputs are required for control input ports as well as that the connection to VCC and GND is done as an inter-cell wiring. However, the pass-transistor cell, which has a large NPC without in-

7-Input, 4-Output Random Logic

	CMOS	LEAP
Layout		
Area	86786.04 $\mu\text{m}^2$ (1.0)	60819.44 $\mu\text{m}^2$ (0.70)
Delay Time	2.284ns (1.0)	1.590ns (0.70)
Tr. Count	800 (1.0)	644 (0.81)
Gate Width	7530 $\mu\text{m}$ (1.0)	3741 $\mu\text{m}$ (0.50)
Net Count	385 (1.0)	473 (1.23)
Power	5.87mW/MHz (1.0)	3.58mW/MHz (0.61)
Cell Count	136 (1.0)	98 (0.72)
Critical Path	10 (1.0)	10 (1.0)

4-b Adder/Subtractor

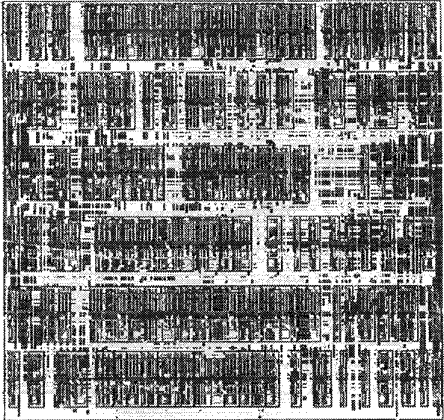
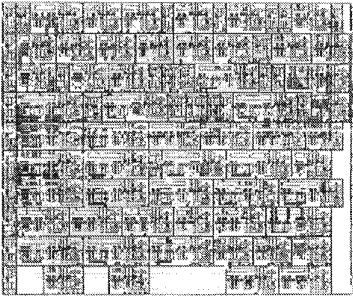
	CMOS	LEAP
Layout		
Area	84288.6 $\mu\text{m}^2$ (1.0)	48589.2 $\mu\text{m}^2$ (0.55)
Delay Time	3.620ns (1.0)	2.691ns (0.74)
Tr. Count	828 (1.0)	545 (0.66)
Gate Width	7583 $\mu\text{m}$ (1.0)	3091 $\mu\text{m}$ (0.41)
Net Count	386 (1.0)	400 (1.04)
Power	6.08mW/MHz (1.0)	3.84mW/MHz (0.63)
Cell Count	133 (1.0)	85 (0.64)
Critical Path	12 (1.0)	10 (0.83)

Fig. 8. Synthesized results of the benchmark circuits.



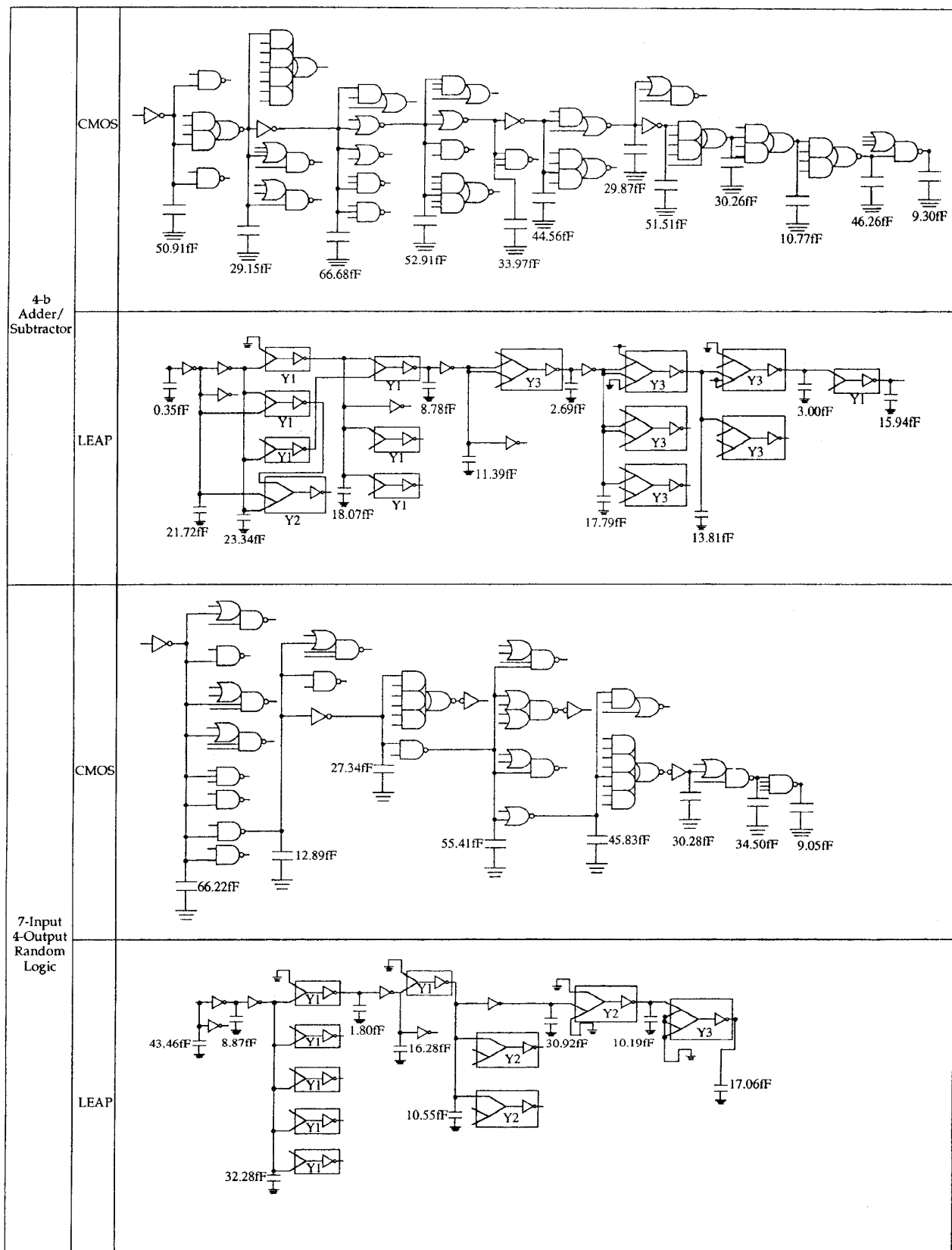


Fig. 9. Critical paths of synthesized circuits.

TABLE II  
SUMMARY OF THE COMPARISON RESULTS OF CELLS

	4-b Adder/Subtractor		7-In, 4-Out Random Logic	
	CMOS	LEAP	CMOS	LEAP
Tr. Count / Cell	6.22	6.41	5.88	6.57
Gate Width / Cell	57 $\mu\text{m}$	36 $\mu\text{m}$	55 $\mu\text{m}$	38.2 $\mu\text{m}$
Net Count / Cell	2.9	4.7	2.8	4.8
Area / Cell	633 $\mu\text{m}^2$	571 $\mu\text{m}^2$	638 $\mu\text{m}^2$	620 $\mu\text{m}^2$
Delay Time / Cell	0.302ns	0.269ns	0.286ns	0.197ns
Power / Cell	45.7 $\mu\text{W}/\text{MHz}$	45.2 $\mu\text{W}/\text{MHz}$	43.2 $\mu\text{W}/\text{MHz}$	36.5 $\mu\text{W}/\text{MHz}$

creasing the cell area, has a high capability for reducing nets. The benchmark test results reveal that the gain due to the enhanced functionality of Y cells has a larger impact on the total area than the above mentioned increased net count.

The advantage of the pass-transistor-cell library over the CMOS library can be understood in a different way by comparing LSI design to the system (software/hardware) design (Fig. 10). This analogy is well known—terminology like “silicon compiler” comes from here. Recently, the change in the instruction set from RISC (reduced instruction set computer) to CISC (complex instruction set computer) has dramatically changed the cost and performance of CPU's. This prods us to explore new cell sets, because the cell library is the counterpart of the instruction set in CPU design, as shown in Table III. A pass-transistor-cell library corresponds to the small instruction set of RISC's. Equation (5) corresponds to the well-known relation used in CPU design [11]

CPU time = Instruction count

$$\times \frac{1}{\text{Instructions per cycle}} \times \text{Cycle time} \quad (6)$$

where instruction count is mainly determined by the algorithm, instruction per cycle by the instruction architecture, and cycle time by the technology. The high performance of RISC's is explained by the large number of instructions per cycle. The high performance-cost ratio of LEAP is explained by its larger nets per cell, as described above.

Although the synthesis result of the 4-bit adder/subtractor is excellent compared to the conventional synthesis results, there still is a considerable distance until it catches up the man-designed circuit quality. One of the authors, who is well experienced in designing fast arithmetic modules [1], actually designed the same logic function based on his heuristics using the pass-transistor cells. The obtained area is 25 967  $\mu\text{m}^2$ , which is about half of the LEAP synthesized circuit (48 589  $\mu\text{m}^2$ ). The delay is 1.7 ns, which is also better than the figures of the synthesized circuit (2.69 ns). If we take the excellent results in the full-adder and 4-2 compressor as shown in Fig. 7 into consideration, the major reason for this may be the lack of the partitioning capability of the entire logic block, which is naturally done in the man-designed circuit.

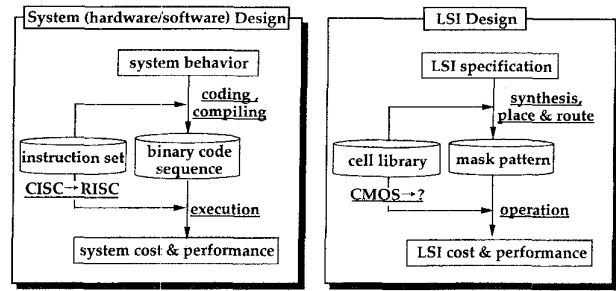


Fig. 10. A comparison between system design, which includes software/hardware design, and LSI design. The cell library is the key for LSI cost and performance in both the chip-design phase and the operation phase just as the instruction set is the key for system cost and performance in both the software-design phase and execution phase.

TABLE III  
THE TRANSITION FROM CMOS TO LEAP IS COMPARED TO THE TRANSITION FROM CISC TO RISC

	CISC	RISC		CMOS	LEAP
No. instructions	many	fewer	No. cells	many	fewer
Instruction Function	orthogonal self-contained	Load/Store	Cell Function	self-contained (NAND,NOR)	Select/Amplify
Instructions/Cycle	low( $\sim 0.3$ )	high( $\sim 1$ )	Nets/Cell	low( $\sim 2.8$ )	high( $\sim 4.7$ )
Programming	assembly	high-level	Design	schematic	HDL
CPU Time = Inst. count $\times$ $\frac{\text{Cycle time}}{\text{Inst. per cycle}}$			Area = Net count $\times$ $\frac{\text{Cell area}}{\text{Nets per cell}}$		

Therefore, the optimized partitioning of a logic into the blocks, which is small enough for a BDD to express efficiently, may be the next challenge if one wants to apply this LEAP scheme to large-block logic synthesis.

### C. Low-Voltage Performance

Because LEAP uses nMOS pass transistors, the slope of the transient pass-transistor output waveform is degraded above the point representing supply voltage minus threshold voltage. Therefore, minimum voltage that can be used for such a circuit is an important issue. This is evaluated by using the synthesized circuit. The inset in Fig. 11 shows the simulated dependence of delay on supply voltage. Here, the threshold voltage is set at 0.67 V. LEAP becomes slower than CMOS in the supply voltage range below the critical value (in this case 1.9 V) due to the influence of the threshold voltage of the nMOS. We performed similar delay simulation for various nMOS threshold voltages (Fig. 11) and found that LEAP is faster than CMOS when the following condition is satisfied:

$$V_{CC} > 2.7V_{th}. \quad (7)$$

In practical applications, this condition has always been satisfied. There is no reason to expect this trend to change in the future. This confirms that LEAP can be applied down to about 1-V supply, where the critical threshold voltage is about 0.4 V. One reason for this result is that the body-effect, which affects the performance of pass-transistors more than that of CMOS's, gets weakened as the supply is lowered. With the threshold of 0.4 V, the

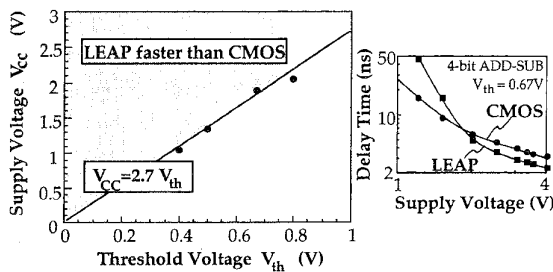


Fig. 11. The condition that LEAP is faster than CMOS as a function of the supply voltage and the threshold voltage. The inset shows the dependence of the delay on supply voltage when the threshold voltage is fixed.

leakage current is still much smaller than the charging/discharging dynamic current; however, if one tries to use lower threshold, subthreshold leakage may be a severe constraint against low-power design for either LEAP or CMOS.

#### D. Remaining Issues

Although LEAP generates excellent results compared to the CMOS for these relatively small blocks, there are many issues to be cleared before general superiority of LEAP over CMOS is claimed. These issues will be discussed here.

As is addressed above, the partitioning of a given logic into smaller blocks may be the key issue for this LEAP system to extend the application. If one wants to use the LEAP system to fully automated large-scale synthesis in an ASIC or a control block of a microprocessor, this condition has to be satisfied, although the current version is quite useful for accelerating hand-crafted logic design process, e.g., design of an arithmetic module of a fast microprocessor. Because of the positive results for relatively small blocks shown in Sections IV-A and IV-B, this issue might be an attracting challenge for the design-automation community to extend the current logic synthesis horizon.

The optimization of the circuits for speed, area, and power constraints will also be important. The next natural question may be whether the optimization techniques which have been used in the conventional logic synthesis can be applied to this pass-transistor based logic design and how they should be modified if this is possible. This is an open question.

The testing of a synthesized logic circuit is also important. As is pointed out by Akers [15], the BDD has a structure which is convenient for the testing of a stack-at-fault. Because the circuit synthesized by the LEAP system has the same structure as the BDD, the synthesized circuit may have an advantage in this respect.

#### V. CONCLUSIONS

Our primary goal in this paper is to clarify the possibilities of top-down pass-transistor design. For this purpose, we constructed a cell library and synthesis tool utilizing the functionality of pass-transistors. The benchmark

test results of our scheme are quite positive, although the size of the benchmarks are still relatively small, and they provide evidence so that this approach has the potential to provide a new generic ULSI logic design scheme which might replace the present CMOS circuitry, our final goal. Because there remains much to be improved in the current simple version of the synthesis tool, we hope this result encourages research activities toward a more advanced version of the synthesis. The key issue which should be cleared before the final goal is reached is how a given logic function can be decomposed into small sub-blocks. We also have found that the low-voltage performance of nMOS-based pass-transistor logic is better than the conventional CMOS circuits down to 1 V, when the threshold voltage is 0.4 V.

#### ACKNOWLEDGMENT

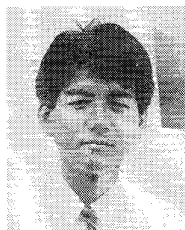
The authors would like to thank Dr. E. Takeda, K. Uchiyama, S. Narita, K. Noguchi, N. Kageyama, M. Tonomura, O. Nishii, and S. Yamashita of Hitachi Central Research Laboratory for their valuable discussions.

#### REFERENCES

- [1] K. Yano, T. Yamanaka, T. Nishida, M. Saito, K. Shimohigashi, and A. Shimizu, "A 3.8 ns CMOS  $16 \times 16$  multiplier using complementary pass-transistor logic," *IEEE J. Solid State Circuits*, vol. 25, p. 388, 1990.
- [2] J. H. Pasternak, A. S. Shubat, and C. A. T. Salama, "CMOS differential pass-transistor logic design," *IEEE J. Solid State Circuits*, vol. 22, p. 216, 1987.
- [3] M. Suzuki, N. Ohkubo, T. Yamanaka, A. Shimizu, and K. Sasaki, "A 1.5 ns 32 b CMOS ALU in double pass-transistor logic," *1993 IEEE Int. Solid-State Circuits Conf. Dig. Tech. Papers*, p. 90, 1993.
- [4] H. Partovi and D. Draper, "A regenerative push-pull differential logic family," *1994 IEEE Int. Solid-State Circuits Conf. Dig. Tech. Papers*, p. 294, 1994.
- [5] R. H. Krambeck, C. M. Lee, and H-F. S. Law, "High-speed compact circuits with CMOS," *IEEE J. Solid State Circuits*, vol. SC-17, p. 614, 1982.
- [6] C. Heikes, "A  $4.5 \text{ mm}^2$  multiplier array for a 200 MFLOP pipelined coprocessor," *1994 IEEE Int. Solid-State Circuits Conf. Dig. Tech. Papers*, p. 290, 1994.
- [7] A. P. Chandrakasan, S. Sheng, and R. W. Brodersen, "Low-power CMOS digital design," *IEEE J. Solid State Circuits*, vol. 27, p. 473, 1992.
- [8] J. H. Pasternak and C. A. T. Salama, "Differential pass-transistor logic," *IEEE Circuits and Devices*, July, p. 23, 1993.
- [9] K. Yano, Y. Sasaki, K. Rikino, and K. Seki, "Lean integration: achieving a quantum leap in performance and cost of logic LSI's," in *Proc. IEEE 1994 Custom Integrated Circuit Conference*, p. 603, 1994.
- [10] H. Harvey-Horn, "User-defined benchmarks help evaluate IC physical libraries," *Electronics Design*, p. 80, Oct. 1993.
- [11] J. L. Hennessy and D. A. Paterson, *Computer Architecture: A Quantitative Approach*. Los Altos, CA: Morgan Kaufmann, 1990.
- [12] S. S. Yau and C. K. Tang, "Universal logic modules and their applications," *IEEE Trans. Comput.*, vol. C-19, p. 141, 1970.
- [13] T. F. Tabloski and F. J. Mowle, "A numerical technique and its application to minimum multiplexer logic circuits," *IEEE Trans. Comput.*, vol. C-25, p. 684, 1976.
- [14] C. E. Shannon, "The synthesis of two-terminal switching functions," *Bell Syst. Tech. J.*, vol. 28, p. 59, 1949.
- [15] S. B. Akers, "Binary decision diagram," *IEEE Trans. Comput.*, vol. C-27, p. 509, 1978.
- [16] R. E. Bryant, "Graph-based algorithm for Boolean function manipulation," *IEEE Comput.*, vol. C-35, p. 677, 1986.
- [17] S. Minato, N. Ishiura, and S. Yajima, "Shared binary decision dia-

grams with attribute edges for efficient Boolean function manipulation," *Proc. 27th ACM/IEEE Design Automation Conf.*, p. 52, 1990.

- [18] R. K. Montoye, P. W. Cook, E. Hokenek, and R. P. Havreik, "An 18 ns 56-bit multiply-adder circuit," *1990 IEEE International Solid-State Circuits Conf. Dig. Tech. Papers*, p. 46.
- [19] G. Goto, T. Sato, M. Nakajima, and T. Sukemura, "A  $54 \times 54$ -b regularly structured tree multiplier," *IEEE J. Solid State Circuits*, vol. 27, p. 1229, 1992.
- [20] R. Murgai, R. Brayton, and A. Sangiovanni-Vincentelli, "An improved synthesis algorithm for multiplexor-based PGA's," *29th ACM/IEEE Design Automation Conf.*, p. 380, 1992.
- [21] K. Karplus, "Amap: A technology mapper for selector-based field-programmable gate arrays," in *Proc. 28th ACM/IEEE Design Automation Conf.*, 1991, p. 244.
- [22] R. J. Landers and S. S. Mahant-Shetti, "Multiplexer-based architecture for high-density, low-power gate arrays," *Dig. 1994 Symposium on VLSI Circuits*, p. 33, 1994.

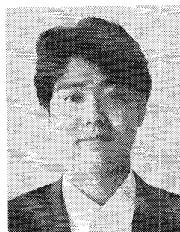


**Kazuo Yano** (A'87) was born in Sakata, Japan, on October 18, 1959. He received the B.S., M.S., and Ph.D. degrees in physics from Waseda University, Tokyo, Japan, in 1982, 1984, and 1993, respectively.

He joined the Central Research Laboratory, Hitachi Ltd., Tokyo, Japan, in 1984. From 1984 to 1990 he was engaged in research on MOS/bipolar devices for low-temperature computers. Since 1987 he has been engaged also in CMOS/BiCMOS logic circuits for ULSI processors and

ASIC's. Since 1990 he has been interested in single-electron devices for future integrated circuits. From March 1991 to February 1992 he was a Visiting Scientist at the Center for Solid-State Electronics Research, Arizona State University, Tempe. He is a co-author of the book *Silicon-Based Heterojunction Devices* (Maruzen, 1991).

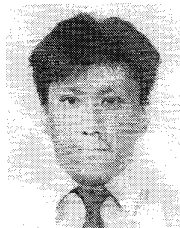
Dr. Yano is a member of the IEEE Electron Devices Society, the American Physical Society, the Japan Society of Applied Physics, and the Institute of Electronics, Information, and Communication Engineers of Japan. He received Best Paper Award at the IEEE International Conference on Computer Design in 1991, and the IEEE 1994 Paul Rappaport Award.



**Yasuhiko Sasaki** (M'95) received the B.S. and M.S. degrees in electronic engineering from the University of Tokyo, Japan, in 1989 and 1991, respectively.

He joined the Central Research Laboratory, Hitachi Ltd., Tokyo, Japan, in 1991. Since then he has been engaged in the research and development of high-speed, low-power logic circuits.

Mr. Sasaki is a member of the Institute of Electronics, Information, and Communication Engineers of Japan.



**Kunihiro Rikino** was born in Tokyo, Japan, on September 21, 1967. He received the B.S. degree in electrical engineering from Tokyo University of Agriculture and Technology, Tokyo, Japan, in 1992.

In 1992 he joined Hitachi Device Engineering Co., Ltd., Chiba, Japan. Since then he has been engaged in the research and development of low-voltage ASIC technology and top-down pass-transistor logic design (LEAP).

Mr. Rikino is a member of the Institute of Applied Physics of Japan.



**Koichi Seki** (S'76-M'81) was born in Tokyo, Japan, in 1954. He received the B.S., M.S., and Ph.D. degrees in electronic engineering from the University of Tokyo, Japan, in 1976, 1978, and 1981, respectively.

Since he joined the Central Research Laboratory, Hitachi, Ltd., he has been engaged in the research and development of amorphous silicon devices, nonvolatile memories, and CMOS logic circuits. Currently he is a Senior Researcher at the laboratory. From 1986 to 1987, he was a Visiting

Industrial Fellow at the Department of Electrical Engineering and Computer Sciences, University of California, Berkeley. His current interests include single electron devices and low power Si LSI's.

Dr. Seki received the Young Engineer Award from the Institute of Electronics and Communication Engineers of Japan in 1985. He is a member of the Institute of Electronics, Information, and Communication Engineers of Japan and the Japan Society of Applied Physics.