Lean Integration:
Achieving a Quantum Leap in Performance and Cost of Logic LSIs

Kazuo Yano, Yasuhiko Sasaki, Kunihito Rikino* and Koichi Seki

ULSI Research Center, Hitachi Central Research Laboratory, Kokubunji, Tokyo 185, Japan
*Hitachi Device Engineering, Kokubunji, Tokyo 185, Japan

ABSTRACT

Lean integration aims at a fundamental change in top-down design by following the path from CISC to RISC. The central idea is a lean cell, which has a tree-shaped nMOS network with input ports placed at the end of an every branch of the tree. A lean cell has flexibility of transistor-level circuit design and full compatibility with conventional cell-based design. An extremely simple lean-cell library with only 7 cells and a synthesis tool called "Circuit Inventor," which uses the lean cells, are developed and they are compared with the conventional "complex" CMOS library that has over 60 cells. The results show that the area, the delay, and the power dissipation are improved by lean integration and performance-cost ratio is improved by a factor of three.

INTRODUCTION

Due to recent progress in top-down LSI design using logic synthesis and HDL, the cell-library design is becoming a key factor in achieving high performance ASICs and MPUs. This is quite natural if we recall that the cell library corresponds to the "instruction sets" if we compare the LSI design to the CPU design. Recently even the evaluation method of cell-library quality has been seriously considered [1]. A conventional CMOS cell library usually has over 60 cells even if we limit the cells to the combinational logic. We postulate that this complex library is, compared to a "CISC," causing unnecessary silicon and engineering costs, and a much leaner LSI design method just like RISC should be conceived. In fact, creating and maintaining the library is becoming such a serious burden in LSI design that it is creating opportunities for companies to provide library-design services.

In this work, we propose lean integration, a completely new cell-library architecture which achieves a quantum leap in performance and cost for logic LSIs. We investigate the impact of this method by comparing overall figures with those of CMOS.

LEAN CELL

The proposed lean-cell library is compared with a conventional CMOS library in Table 1 and Fig. 1. The lean-cell library has only 7 cells, which is far smaller than a conventional library. The number of essential logic cells is also less and it is only 3, Y1, Y2, Y3 (Fig. 1). The other 4 cells are simple inverters.

Another feature of lean cells is that they are defined by the transistor network topology, a binary tree-shaped nMOS network, rather than the Boolean function. The name "lean" came from the fact that the situation is just like transistors are directly connected to the cell ports of the cell. Although the cell works as a multiplexer, the essential advantage of the lean cell is that transistor-level circuit engineering is possible by using this cell. In fact the transistors act as pass-transistors, or source-grounded configuration, or source-follower configuration depending on the input configuration. By contrast, the conventional cell is defined by its Boolean function, which is often chosen based on previous case studies, and the inner circuit configuration is simply the means to meet the function requirement.

The advantage of a lean cell is that it changes the function by changing the configuration of the cell input ports (Fig. 3). The drain port, the end of a branch of the tree, has the freedom to be connected with the output of another cell, or a power supply line, or a ground line. Different input configurations correspond to different Boolean functions. Note that a very complex logic function is achieved by a single cell. This functionality came from that the transistor-level circuit engineering already described.

Despite this transistor-level flexibility of the lean cell, it is fully compatible with the framework of cell-based design. As a result the delay of the cell can be defined as a function of the load capacitance. This is made possible by the output inverter, which separates the inputs from the output. The feedback inverter and the pull-up pMOS, both consisting of minimum-size MOSFETs, are added to avoid DC leakage current in the CMOS inverter.

Because preparing and updating the lean-cell library requires only small engineering cost, it is much easier to adopt the state-of-the-art process technology even in a tight schedule constraint. Therefore, the lean cell encourages concurrent interaction between logic designers and process engineers. By contrast, major revision of 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.

The area of a logic block is reduced by using lean cells. The logic area is given by the following equation.

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

where net count is mainly determined by the logic synthesis algorithm, nets per cell (NPC) is mainly determined by the cell architecture, and cell area is mainly determined by the technology level. This relation corresponds to the well-known relation used in CPU design:

$$\text{CPU time} = \text{Instruction count} \times \frac{1}{\text{Instructions per cycle}} \times \text{Cycle time} \quad (2)$$

In Eq.(1) a logic function is considered to be a box which reduces the number of nets, or nodes. The lean cell, which has a large NPC without increasing the cell area, has a high capability of reducing nets. Therefore the number of cells required to build a logic block is smaller than that of the CMOS, resulting in smaller block area. A similar argument holds for power consumption, which leads to lower power consumption.

The delay of the logic block is also reduced by the lean cells. Very complex logic functions, which are not included in conventional CMOS libraries, can be achieved by using only a single lean cell. Therefore, the number of critical-path cells is reduced. In addition, complex CMOS gates with large parasitic capacitance are slow and have low current drive capability. By contrast, parasitic capacitance of lean cells is small and high current drive capability is possible due to the output inverter.

The transition from a CMOS library to the lean-cell library is just like the transition from CISC to RISC (Table. 2). A lean-cell library corresponds to the small instruction set of RISCs. The instructions of the RISC were not convenient for designers who were accustomed to the conventional orthogonal instructions of CISCs. However, this has been overcome by using an optimized compiler. A lean cell has a similar characteristic. Because it is somewhat like directly controlling the transistor behavior from outside of the cell, logic designers, who are not familiar with the details of individual circuit may become reluctant to deal with them. However, logic synthesis based on lean cells solves this problem. The high performance of RISCs is explained by the large number of instructions per cycle. The high performance-cost ratio of lean integration is explained by its larger nets per cell. The simple instruction set of RISCs and the especially good expectation of a relation between the performance and the instruction sequence help the compiler to provide highly efficient instructions. The lean cells also help the synthesizer to provide area- and delay-effective net lists.

We also developed a logic synthesis tool called "Circuit Inventor", which fully utilizes the lean cell characteristics. Circuit Inventor accepts an HDL description, creates net lists based on lean cells and gives those data to the layout tool. It expresses the required logic function in a compact form by using a reduced BDD (Binary Decision Diagram) [2] and conducts various optimizations. BDD has the same network topology as lean cells and efficient mapping to cells is possible. Automatic insertion of optimized inverters into heavily-loaded nodes is possible. The details of the internal algorithm of Circuit Inventor will be described elsewhere.

EXPERIMENTS AND DISCUSSION

The performance of lean cells is compared with that of CMOS cells. Two types of benchmark logic are chosen. One is a 4-b adder/subtracter, which represents arithmetic logic, and the other is 7-input 4-output random logic, which is created by assigning random numbers to the output of the truth table. CMOS logic is synthesized by using a popular commercial logic synthesis tool. 0.5-μm process with 3-level metal is assumed and poly-cell-type layout style is used. Metal 1 is assigned to the intra-cell wiring, metal 2 is assigned to Y-direction inter-cell wiring, and metal 3 is assigned to X-direction inter-cell wiring. The cell input/output ports are formed as through-holes between metal 1 and metal 2. The critical path and the wiring load was extracted from the layout data (Fig. 4) and the delay was obtained by using circuit simulation. The power consumption was determined by using circuit simulation of the total circuit.

The results are dramatic (Table. 3). The lean cells show higher figures in all respects including area, delay, and power consumption for either benchmark. If we define the performance-cost ratio of a cell architecture by using the product of the area, delay, and power, the lean cells has 3-4 times learger ratio.

The average NPC is actually boosted in the lean integration from 2.8 to 4.7 (Table 4), which is the major contributor of the area reduction. The delay per cell is smaller in the lean cells, which contributes the delay reduction.

The dependence of the delay on supply-voltage is another important aspect of the technology choice. The lean cells become slower than CMOS cells at supply voltages below the critical value, because of the influence of threshold voltage of the nMOS. However, under practical conditions, where the extrapolated threshold voltage is smaller than $V_{CC}/2.7$, the lean cells are always faster than the CMOS cells as shown in Fig. 5.

CONCLUSIONS

Lean integration, which provides a quantum leap in performance and cost of ASICs and MPUs is proposed. This new design method goes far beyond marginal improvements and reaches 3-4 times improvement in performance-cost ratio and gives much higher competitiveness in "lean LSIs" and in systems that use LSI chips.

ACKNOWLEDGEMENTS

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

[1] H. Harvey-Horn, "User-defined benchmarks help evaluate IC physical libraries," Electronics Design, Oct., 80 (1993)
[2] R. E. Bryant, "Graph-based algorithm for Boolean function manipulation," IEEE Computers, Vol-C35 (8), 677(1986)

Table 1 Cell lists of coventional and proposed lean-cell library

Conventional CMOS Library (61 cells)			Lean-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)

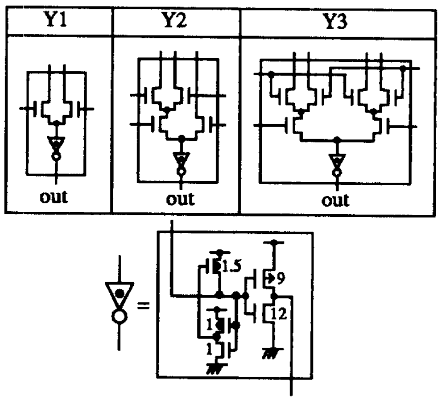


Fig. 1 Circuit diagram of lean cells and the output inverters

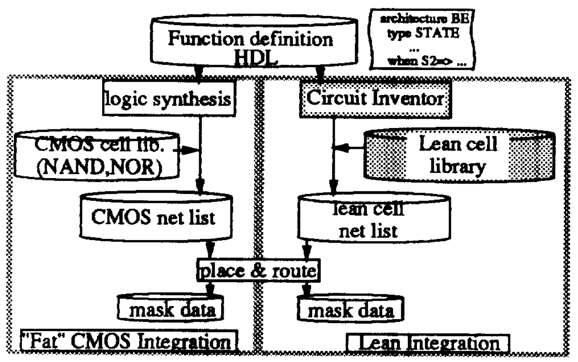


Fig. 2 Conventional "fat" CMOS integration vs. lean integration

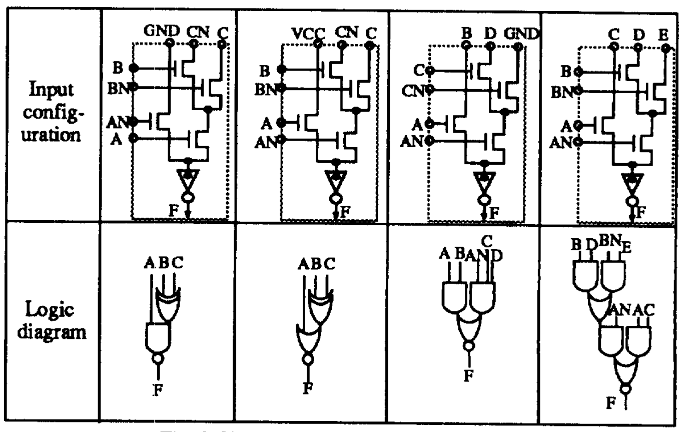


Fig. 3 Various logic fuctions of the lean cell "Y2"

Table 2 Transition from CMOS "fat" library to lean-cell library is compared to "CISC->RISC" transition

	CISC	RISC		"fat" CMOS	lean
No. instructions	many instructions ──▶	fewer instructions	No. cells	many cells ──▶	fewer cells
instruction fuction	orthogonal self-contained instructions ──▶	Load/Store architecture	Cell logic function	self-contained logic(NAND,NOR) ──▶	Select/Amplify
Instructions/cycle	low(~1/3) ──▶	high(~1)	Nets/cell	low(~2.8)	high(~4.7)
programming	assembler ──▶	high-level language	design	schematic ──▶	HDL

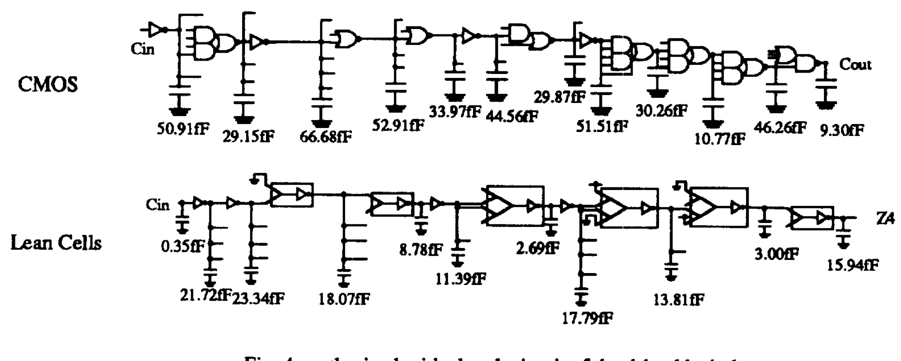


Fig. 4 synthesized critical path circuit of the 4-b adder/subtractor

Table 3 Summary of bencmark design

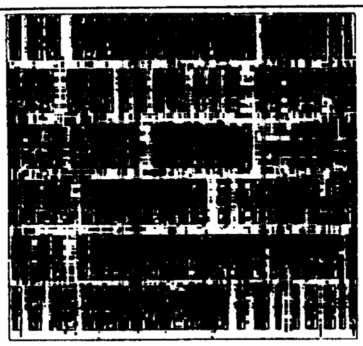
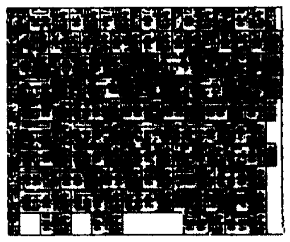
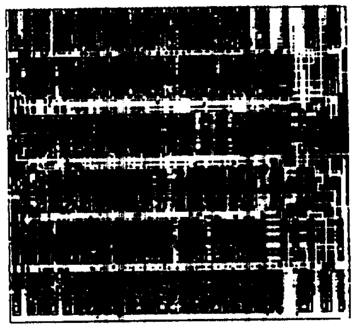
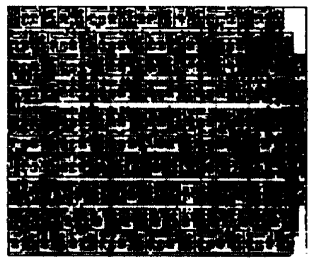
	CMOS	Lean
	4-bit ADD-SUB	
Layout		
AREA	84288.6μm^2(1.0)	48589.2μm^2(0.55)
Delay Time	3.620ns (1.0)	2.691ns (0.74)
Tr.Count	828 (1.0)	545 (0.66)
Gate Width	7583μm (1.0)	3091μm (0.41)
Net Count	386 (1.0)	400 (1.04)
Power	6.08mW/MHz (1.0)	3.84mW/MHz (0.63)
Cell Count	133 (1.0)	85 (0.64)
Critical Path	12 (1.0)	10 (0.83)
	7in Random Logic	
Layout		
AREA	86786.04μm^2(1.0)	60819.44μm^2(0.70)
Delay Time	2.284ns (1.0)	1.590ns (0.70)
Tr.Count	800 (1.0)	644 (0.81)
Gate Width	7530μm (1.0)	3741μm (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)

Table 4 Comparison of figures per cell

	4-bit ADD-SUB		7-in Random Logic	
	CMOS	Lean	CMOS	Lean
Tr. Count /Cell	6.22	6.41	5.88	6.57
Gate Width / Cell	57μm	36μm	55μm	38.2μm
Net Count / Cell	2.9	4.7	2.8	4.8
Area / Cell	633μm^2	571μm^2	638μm^2	620μm^2
Delay Time / Cell (Average wire length)	0.302ns (351μm)	0.269ns (133μm)	0.286ns (334μm)	0.197ns (206.5μm)
Power / Cell	45.7μW/MHz	45.2μW/MHz	43.2μW/MHz	36.5μW/MHz

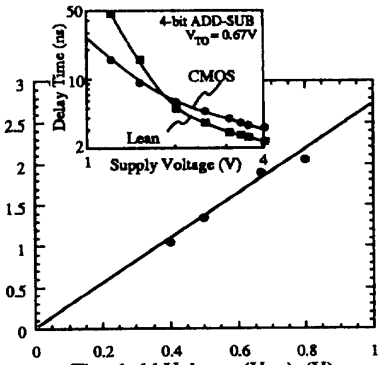


Fig. 5 Crossover supply voltage of delay between CMOS and lean cells