A 1.25 GHz 8-bit Tree-Structured Carry Lookahead Adder*

Chua-Chin Wang†, Po-Ming Lee, Chenn-Jung Huang‡, Rong-Chin Lee

Department of Electrical Engineering
National Sun Yat-Sen University
Kaohsiung, Taiwan 80424
Tel : 886-7-525-2000 ext. 4144
Fax : 886-7-5254199
E-Mail: ccwang@ee.nsysu.edu.tw

ABSTRACT

By modifying the so-called all-N-transistor (ANT) design, several small but novel cells are proposed which can rapidly compute the required generate and propagate function, respectively. We utilize these cells to design a 8-bit tree-structured carry lookahead adder (CLA). The 8-bit CLA not noly possesses few transistor count, but also occupies small area size. Moreover, the post-layout simulation results given by TimeMill show that the clock used in this 8-bit CLA can run up to 1.25 GHz. The proposed architecture is also easy to be expanded for long data additions.

Keywords: tree-structured CLA, "*o*" cell, all-N-transistor(ANT)

1. Introduction

Improving adder designs has received considerable attention [1]-[6]. CMOS dynamic logic is one of the promising options to challenge GHz operation regarding adder design [5]. However, domino logic [1] cannot be noninverting; an adder in [4] can only process short data words; all-N-logic [5] and robust single phase clocking [6] cannot operate correctly under clocks with short rise time or fall time; single-phase logic [2] and Zipper CMOS [3] contain slow P-logic blocks. An all-N-transistor (ANT) noninverting function block was proposed to resolve the mationed difficulties [7], which can be used to design a high-speed CLA. In this paper, we, furthermore, enhance the ANT logic by proposing an "*o*" cell and a tree-structured scheme for CLAs. The simulation result of the proposed design is also given to prove its high-speed performance.

*This research was partially supported by Nation Science Council under grant NSC 88-2219-E-110-001

†Contact author

‡ Dr. Chenn-Jung Huang is currently an assistant professor with Department of Information Education, National Taitung Teachers College.

2. 8-bit Tree-Structured CLA

2.1 Prior All-N-Transistor (ANT) Function Unit

An ANT logic is presented in Figure 1. The main feature of this design is the presence of feedback transistor pair, P3 and N3, between the evaluation block and the output. P3 and N3 respectively provide an extra charging path and discharging path, thus accelerating the evaluation. Notably, the ANT in Figure 1 is non-inverting. The detailed operations of the ANT are described in [7].

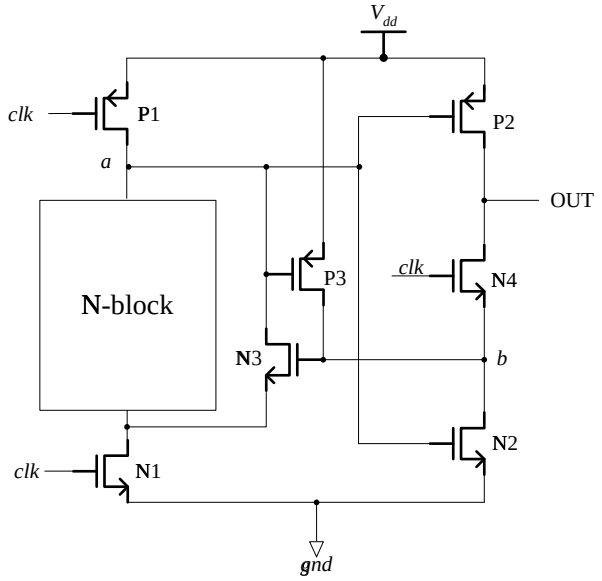

Figure 1: The schematic diagram of the prior ANT logic.

2.1.1 *Sizing Problem for 0.35 μm 1P4M CMOS Technology*

A reason why other high-speed logics can not run correctly given clocks with short rise time or fall time is that the size of each transistor can not be tuned properly. Both [1] and [4] possess this inherent shortcoming, accounting for why they can not use normal square-wave clocks in the GHz range. Thus, the sizing problem of the transistors in the ANT, let alone those in the N-block, significantly affects the speed. Theoretically, wider transistors have lower resistance. Herein, we conducted several simulations to obtain the optimal figure for the sizing of each transistor given in Figure 1 using TSMC (Taiwan Semiconductor Manufacturing Company) 0.35 μm 1P4M technology. Those results are summarized in Table 1.

2.2 *Element Cells for Sum, Generate and Propagate Functions*

A CLA needs *propagate* signals as well as *generate* signals to calculate sums and the carry signals. The circuits to produce the *propagate* signals and the *generate* signals are described detailedly in the following subsections.

Transistor	L (μm)	W (μm)
N1	0.35	10
N2	0.35	10
N3	0.35	3
N4	0.35	10
P1	0.35	15
P2	0.35	15
P3	0.35	6
N-block	0.35	5

Table 1: The sizes of ANT logic block.

2.2.1 *Generate Signals*

The *generate* signal at stage i is formulated as $g_i = A_i \cdot B_i$, where A_i and B_i are input signals, $\forall i, i = 1...7$. Figure 2 shows the schematic of the generation of $g_i, i = 1...7$. Notably, the g_0 used in our CLA is particularly formulated as $g_0 = A_0 \cdot B_0 + C_{in}(A_0 + B_0)$. The schematic view of such a g_0 function is given in Figure 3. The reason why the generation of g_0 is different from the rest of g_i, $\forall i, i = 1...7$, is that the carry can be produced by C_{in} and either one of A_0 and B_0.

2.2.2 *Propagate Signals*

Propagate signals are resulted from the definition of $p_i = A_i \bigoplus B_i$. Notably, the *propagate* signals are defined differently from the traditional $p_i = A_i + B_i$, because $p_i = A_i \bigoplus B_i$ can be reused to compute the sum terms, S_i, $\forall i, i = 0...7$. The *propagate* signal generation schematic is shown Figure 4.

2.2.3 *Sum Signals*

The sum term is defined as $S_i = C_{i-1} \bigoplus p_i$, where C_{i-1} means $(i-1)$th carry signal, $\forall i, i = 1...7$. The schematic diagram of the sum signal S_i is shown in Figure 5. Meanwhile, it should be noted that

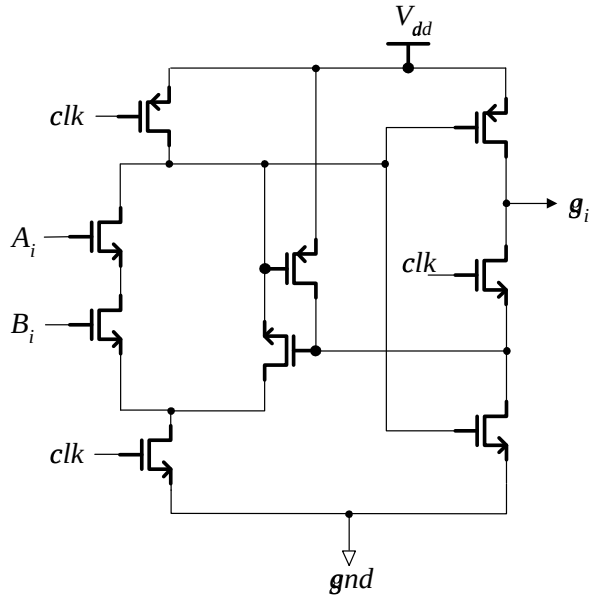

Figure 2: The schematic diagram of g_i generation, $\forall i, i = 1...7$.

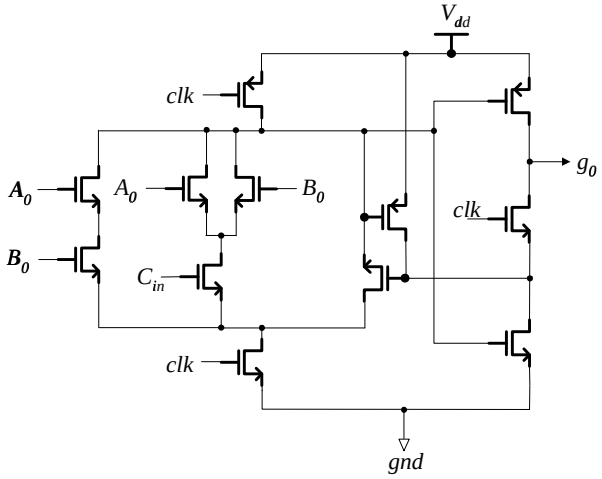

Figure 3: The schematic diagram of g_0 generation.

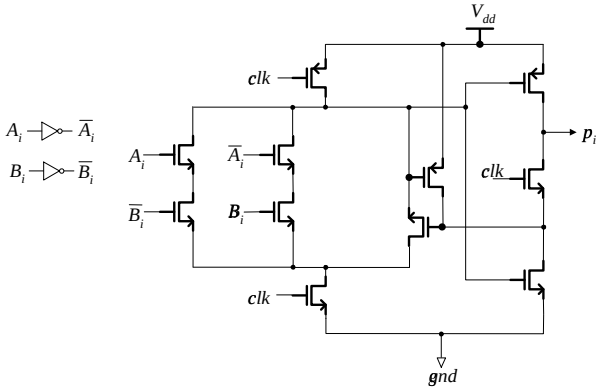

Figure 4: The schematic diagram of p_i generation, $\forall i, i = 0...7$.

$S_0 = C_{in} \bigoplus P_0$ of which the schematic is shown in Figure 6.

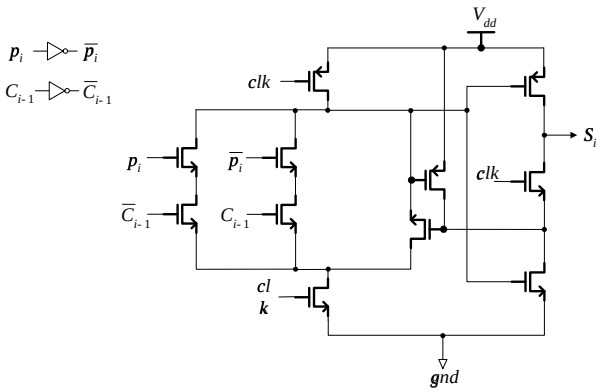

Figure 5: The schematic diagram of S_i generation, $\forall i, i = 1...7$.

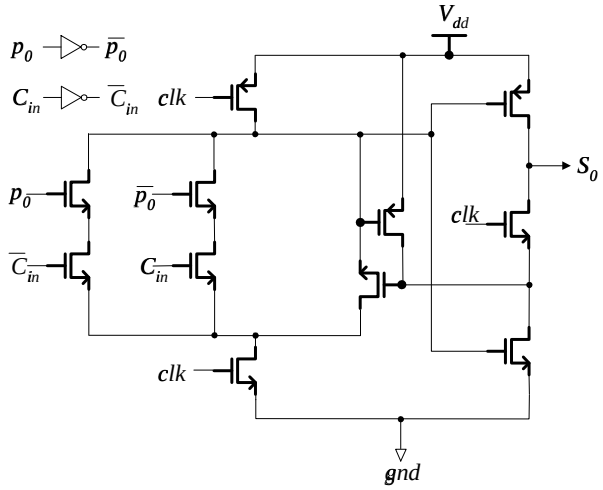

Figure 6: The schematic diagram of S_0 generation.

2.3 Tree-Structured CLA

The 64-b PLA-styled hierarchical CLA presented in [7] can not be pipelined because the carry-in signal for the first-level 8-bit CLA unit has to be fed by the second-level 8-bit CLA. This limitation diminishes the effectiveness of this circuit when large streams of data are operated such that higher data rates are required. Therefore, we propose an alternative architecture of CLAs based on a binary-tree structure in this subsection, which eliminates the generation

of back propagation signals demanded in the PLA-styled hierarchical architecture.

2.3.1 Carry-Lookahead Generator Cell ("o" cell)

Brent and Kung [9] defined a so-called operator "o" as follows:

$$\begin{aligned}(g_{out}, p_{out}) &= (\hat{g}_{in}, \hat{p}_{in}) o (g_{in}, p_{in}) \\ &= (\hat{g}_{in} + g_{in} \cdot \hat{p}_{in}, p_{in} \cdot \hat{p}_{in}) \quad (1)\end{aligned}$$

Based on Eqn. (1), the carry C_i can be expressed as:

$$\begin{aligned}(G_0, P_0) &= (g_0, p_0), \\ (G_i, P_i) &= (g_i, p_i) o (G_{i-1}, P_{i-1}) o ... o (g_0, p_0), \\ C_i &= G_i, \quad (2)\end{aligned}$$

where p_i's and g_i's are the *propagate* and *generate* signals, respectively. Note that $g_0 = A_0B_0 + A_0C_{in} + B_0C_{in}$ is allowed to be used in both addition and subtraction. Figure 7 shows the implementation of the "o" operator using ANT logic.

Due to the associativity property of the "o" operator, (G_i, P_i) can be evaluated in any order from the given p_i's and g_i's. Accordingly, Brent and Kung [9] proposed a tree structure to compute (G_i, P_i). Dozza *et al.* [8] reduced the critical delay of C_i's generations from $2(\log_2 n - 1)$ to $\log_2 n$ levels of the "o" cells by introducing more "o" cells in the binary tree. An example of a 32-bit tree-structured CLA is illustrated in Figure 8. Since this architecture is very regular and modular, this approach can be expanded to design an alternative 64-bit ANT CLA to fix the problem of the PLA-styled CLA [7].

2.3.2 Speed and Area Analysis

Speed: The critical path of an adder resides in the generation of carry signals in the tree-structured adder. When the binary data are ready, the generation of p_i's and g_i's using ANT logic requires the high half of a full cycle. The p_i's and g_i's are then fed into the array of "o" cells. The final C_i results are then ready after $\frac{\log_2 n}{2}$ cycles. As soon as it is generated, each C_i is inverted and fed into the S_i's function blocks as shown in Figure 5. Another half cycle is then required to produce all of the S_i's. The final result is available after $\frac{\log_2 n}{2} + 1$ cycles.

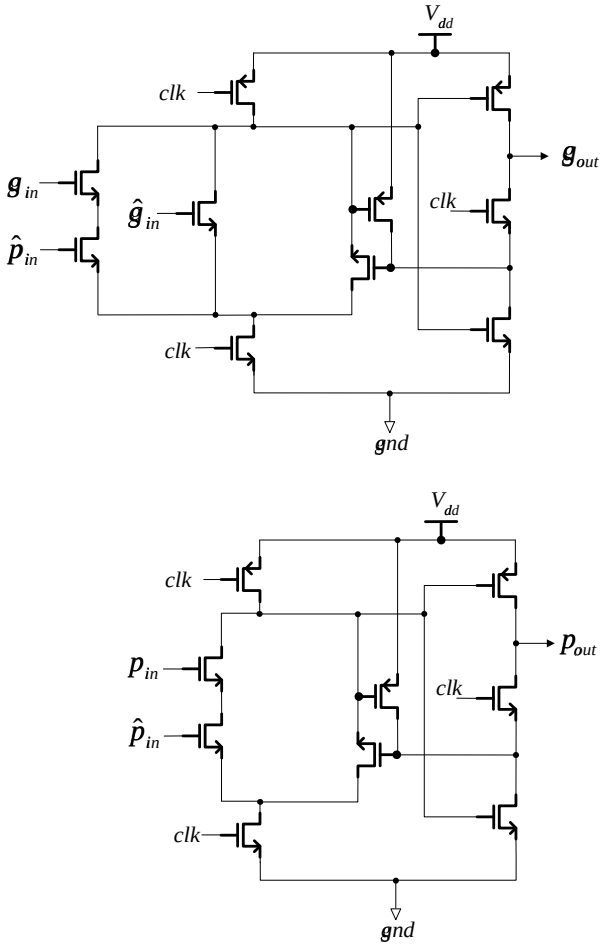

Figure 7: The schematic diagram of the "o" cell.

Area: Area analysis includes the following aspects:

- Carry Out : The number of total "o" cells used in the generation of C_i's is summarized as:

$$T_o = n \cdot \log_2 n - (1 + 2 + .. + 2^{\log_2 n - 1})$$

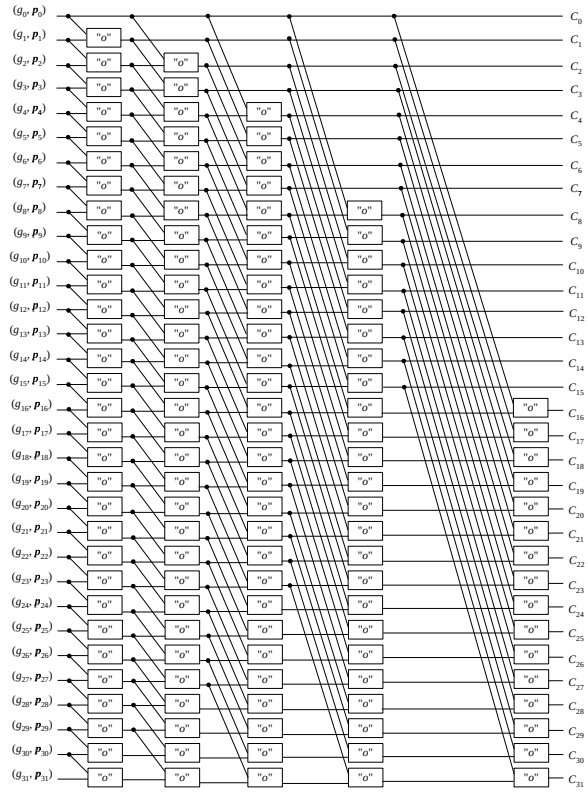

Figure 8: A 32-bit tree-structured CLA.

$$= \quad n \cdot \log_2 n - n + 1. \qquad (3)$$

The number of the transistors in the "o" cells becomes $19 \cdot (n \cdot \log_2 n - n + 1)$ since there are 19 transistors used in an "o" cell. However, all the *propagate* signals, P_i's, which reside in the "o" cells generating C_i's, are not involved in the computations. Consequently, they can be removed from the corresponding "o" cells for area saving. Hence, the total transistor count for the array of "o" cells is

$$\begin{aligned} T_{Tree} &= 10 \cdot (n \cdot \log_2 n - n + 1) + 9 \\ &\quad \cdot (n \cdot \log_2 n - 2n + 2) \qquad (4) \end{aligned}$$

- Inverters : Signals which must be inverted include the input data words, clock, C_i's, and p_i's. Therefore, the total number of transistors for the inverters is $T_I = 8n+2$.

- p_i's and g_i's generation : The total number of transistors is 20 n + 3.

- S_i's generation : The total transistor count is 11 n.

  In sum, the total number of transistors required to implement an n-bit CLA with tree-structured design using ANT logic is

$$\begin{aligned} T_{Total} &= 10 \cdot (n \cdot \log_2 n - n + 1) \\ &\quad +9 \cdot (n \cdot \log_2 n - 2n + 2) \\ &\quad +8n + 2 + 20n + 11n \\ &= 19n \cdot \log_2 n + 11n + 33 \quad (5) \end{aligned}$$

From the analysis given above, we can derive the number of transistors used in a 64-bit tree-structured CLA is 8034, which is less than that of 64-bit PLA-styled hierarchical CLA [7], 8352. The delay is also 4 cycles (3.2 ns if a 1.25 GHz clock is used). Moreover, the n-bit tree-structured CLA can be entirely pipelined by exploiting the 'half-cycle stealing' property of the ANT logic. The clock rate of such a circuit can be as high as 1.25 GHz, which is shown in Figure 9. The output of the consecutive additions of two n-bit binary numbers can be done in each cycle when the circuit is fully pipeline-operated.

3. Performance Simulations and Comparison

The performance of the 8-bit CLA using the ANT logic in a tree-structured design can be verified by EDA tools, e.g. HSPICE & TimeMill. The performance of several adder designs using different logics is revealed in Table 2. Note that the clock rate is

Logic	Delay	# transistors	Technology	Clock
8-bit PLA-ANT CLA [7]	2.0 ns	928	0.6 μm	1 GHz
64-bit PLA-ANT CLA [7]	2.0 ns	71908	0.6 μm	1 GHz
64-bit PLA-ANT hierarchical CLA [7]	4.0 ns	8352	0.6 μm	1 GHz
32-bit EMODL adder [1]	2.7 ns	1537(gates)	1.2 μm	1 GHz
8-bit TSPC adder (1 μm) [4]	7.5 ns	1832	1.0 μm	1 GHz
All-N-logic [5]	Failed	2062	0.8 μm	1 GHz
8-bit ANT tree-structured CLA	2.0 ns	577	0.35 μm	1.25 GHz
64-bit ANT tree-structured CLA	3.2 ns	8033	0.35 μm	1.25 GHz

Table 2: The performance comparison of different designs.

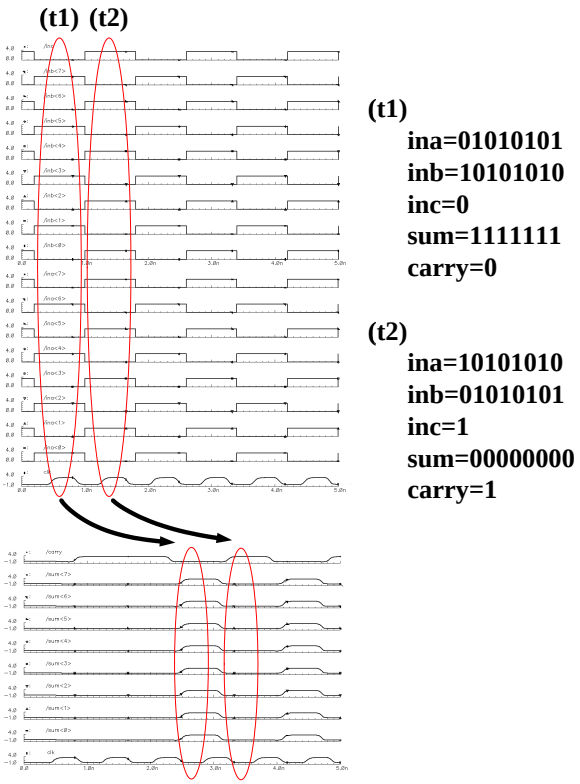

Figure 9: Simulation result of the 8-bit CLA.

1.25 GHz or 1 GHz with a 0.01 ns rise time and the same fall time.

The layout view of the 8-bit CLA using tree-structured ANT is given in Figure 10.

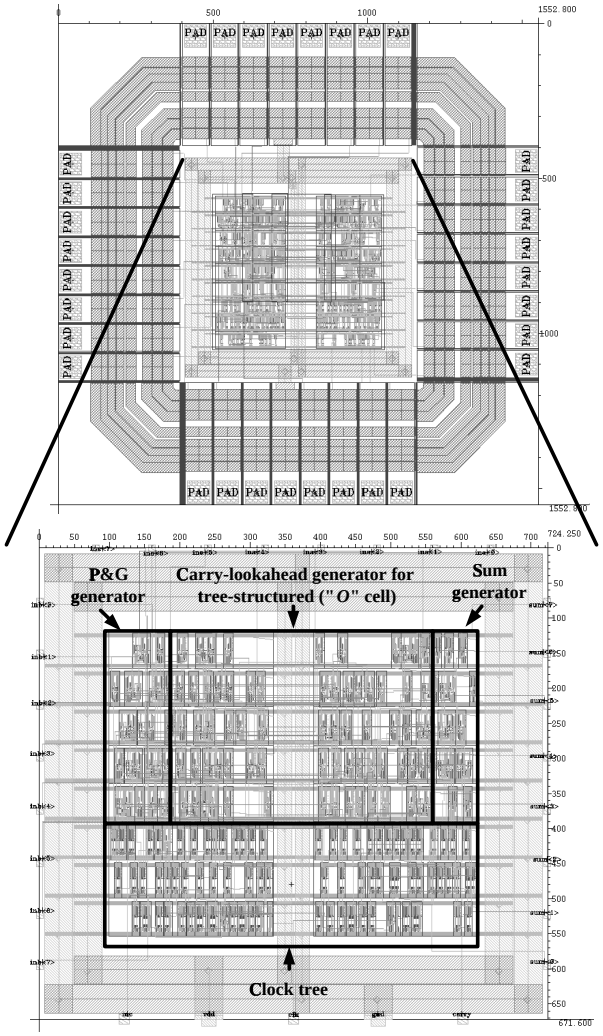

Figure 10: The layout view of the 8-bit CLA chip.

4. Conclusion

In this paper we have proposed a high-speed tree-structured ANT logic designs for the implementation of CLAs. Simulation results not only verify the accuracy of the function in the giga hertz range, but also indicate that the proper size of each transistor is adjusted such that a usual square-wave clock can be used. The tree-structured, using only one clock, causes the result of an 8-bit adder to be obtained in 2.0 ns when the 1.25 GHz clock is used. We also derived the number of transistors (area) for larger long

adders using the proposed approach. At last, the result of consecutive additions can appear in each cycle when the tree-structured CLA is pipeline-operated.

5. References

[1] Z. Wang, G. A. Jullien, W. C. Miller, J. Wang, and S. S. Bizzan, "Fast adders using enhanced multiple-output domino logic," *IEEE J. Solid-State Circuits,* vol. 32, no. 2, pp. 206-214, Feb. 1997.

[2] J. Yuan., and C. Svensson, "High-speed CMOS circuit technique" *IEEE J. Solid-State Circuits*, vol. 24, pp. 62-70, Feb. 1989.

[3] C. M. Lee, and E. W. Szeto, "Zipper CMOS," *IEEE Circuits Devices Mag.*, pp. 10-16, May 1986.

[4] R. Rogenmoser, and Q. Huang, "An 800-MHz 1mm CMOS pipelined 8-b adder using true single phase clocked logic-flip-flops," *IEEE J. Solid-State Circuits,* vol. 31, no. 3, pp. 401-409, Mar. 1996.

[5] R. X. Gu, and M. I. Elmasry, "All-N-logic high-speed true-single-phase dynamic CMOS logic," *IEEE J. Solid-State Circuits,* vol. 31, no. 2, pp. 221-229, Feb. 1996.

[6] M. Afghahi, "A robust single phase clocking for low power high-speed VLSI application," *IEEE J. Solid-State Circuits,* vol. 31, no. 2, pp. 247-253, Feb. 1996.

[7] C.-C. Wang, C.-J. Huang, and K.-C. Tsai, " A 1.0 GHz 0.6-μm 8-bit carry lookahead adder using PLA-styled all-N-transistor logic," *IEEE Trans. Circuits and Systems, Part II: Analog and Digital Signal Processing,* vol. 47, no. 2, pp. 133-135, Feb. 2000.

[8] D. Dozza, M. Gaddoni, and G. Baccarani, "A 3.5 ns, 64 bit, carry-lookahead adder," *1996 IEEE Inter. Symp. on Circuits and Systems,* vol. II, pp. 297-300, June 1996.

[9] R. P. Brent, and H. T. Kung, "A regular layout for parallel adders," *IEEE Trans. Computers,* vol. C-31, no. 3, pp. 260-264, Mar. 1982.