Evaluation of Three 32-Bit CMOS Adders in DCVS Logic for Self-Timed Circuits

Gustavo A. Ruiz

***Abstract*—The efficient implementation of adders in differential logic can be carried out using a new generate signal (N) presented in this paper. This signal enables iterative shared transistor structures to be built with a better speed/area performance than a conventional implementation. It also allows adders developed in domino logic to be easily adapted to differential logic. Based on this signal, three 32-b adders in differential cascode switch voltage (DCVS) logic with completion circuit for applications in self-timed circuits have been fabricated in a standard 1.0-μm two-level metal CMOS technology. The adders are: a ripple-carry (RC) adder, a carry look-ahead (CLA) adder, and a binary carry look-ahead (BCL) adder. The RC adder has the best levels of performance for random input data, but its delay is significantly influenced by the length of the carry propagation path, and thus is not recommended in circuits with nonrandom input operands. The BCL adder is the fastest but has a high cost in chip area. The CLA adder provides an intermediate option, with an area which is 20% greater than that of the RC adder. Its average delay is slightly greater than that of the other two adders, with an addition time which increases slowly with the carry propagate length even for adders with a high number of bits.**

***Index Terms*— Adders, asynchronous addition, asynchronous design, DCVS logic, dynamic-logic-circuit.**

I. Introduction

SELF-timed circuits have recently received a great deal of attention as they offer a possible solution to clock distribution problems, since they do not require a global clock, simplifying global chip routing and eliminating clock skew problems [1]. Self-timed circuits avoid timing problems by replacing the clock with a request-acknowledge communications protocol between modules of the circuit. These systems show increased robustness to processing and environmental variables, provide component modularity, can exhibit low system power requirements, and are capable of operating at the maximum speed determined by the intrinsic hardware delays (they can be designed for average case rather than worst-case performance). Unfortunately, self-timed circuits are more difficult to design than synchronous circuits and, in many cases, lack the required supporting design tools [2].

The most popular form of self-timed approach consists of computation blocks interconnected by handshaking interface circuits operating with completion signals for data transfer control. In self-timed circuits, the completion signals involved in the asynchronous local communication can be generated in a general way with dynamic differential cascode voltage switch (DCVS) logic. The key feature of this logic family, made up of two complementary logic trees similar to domino logic, is the dual-rail coded nature. This characteristic is used to generate completion signals which indicate when a computation operation has finished. This logic has potential advantages over standard CMOS logic [3], [4] and other differential CMOS logic families [5] in terms of speed, power, and logic flexibility. DCVS logic also has an inherent self-testing property to implement fault tolerance circuits at low cost [6]. Its major drawback is the need for a greater area for routing, since it requires twice as many interconnection lines.

The self-timed approach has been used to implement various asynchronous processors [7], [8] whose performance is influenced to some extent by the adder speed. Most of the asynchronous adders implemented up to now, and those which have recently been proposed [9], are based on random operand models with an average delay which is far lower than the worst-case delay. Burks *et al.* [10] have shown that for an n-bit ripple-carry addition of randomly distributed input operands, the average length of the longest carry sequence is bounded from above by $\log_2 n$. Reitwiesner [11] and Hendrickson [12] demonstrated that the average longest carry in an asynchronous addition can be approximated by $\log_2(5n/4)$. However, Garside [13] has shown that addition performed in the asynchronous AMULET processor [8] had an average maximum carry propagate path almost twice as long as that expected from purely random input data, with typical data processing operations of up to three times greater than this value. In these circumstances, asynchronous addition takes a long time unless the operands are random in nature.

Recent literature contains some implementations and evaluations of asynchronous adders. Franklin and Pan [14] have simulated the suitability of various adders (implemented in DCVS logic, although circuit details are not given) operating in an asynchronous environment. The results show that hybrid structures run faster than simple ripple-carry adders although the cost in area may be high. Kinniment [15] compared various adders and observed that asynchronous adders only give a performance improvement over conventional synchronous adders in very limited conditions. This comparison was made in terms of time and cost, using designs carried out in standard gates. In both studies, the asynchronous adders evaluated are adaptations of conventional structures which do not incorporate any modification which might enhance its asynchronous performance. In fact, some high-speed adders [16], [17] in

Manuscript received June 9, 1997; revised September 19, 1997. This work was supported in part by the Spanish Government Research Commission (CICYT) and the Caja Cantabria.

The author is with the Departamento de Electrónica y Computadores, Facultad de Ciencias, 39005-Santander, Spain.

Publisher Item Identifier S 0018-9200(98)02332-4.

domino logic cannot be implemented directly in DCVS logic because of their complementary nature, despite the similarities between both logics.

In this paper, new logic equations for binary addition enabling the efficient implementation of adders in differential logic are described. For this purpose, a new generate signal (N) is introduced, associated with the definition of the complementary carry which has similar characteristics to the carry generate signal G of conventional adders. The symmetry properties of G and N enable new iterative shared transistor structures to be built which reduce the number of devices and interconnection lines, providing area and speed improvements over conventional implementations. Based on these signals, a 32-b ripple-carry (RC) adder, a 32-b carry look-ahead (CLA) adder, and a 32-b binary carry look-ahead (BCL) adder in DCVS logic with completion circuit have been fabricated in a standard 1.0-μm CMOS technology. The area and performance parameters of each adder have been measured and compared. The results show that the BCL adder is the fastest but has extremely high area requirements. The RC adder offers the best speed-area ratio for random input data but is slower for long carry propagation paths. The CLA adder provides the best option. It has an area increase of 20% over the RC adder but maintains a high average throughput for any addition operation.

II. Mathematical Formulation

The most widely used technique in the design of adders in differential logic consists in defining both the logic function, which is generally taken from conventional addition logic equations, and its complement, with the aim of maximizing the sharing of terms [3]. This technique has the problem that many of the structures developed for conventional logic cannot be efficiently translated into differential logic, since there is no duality between the two complementary network trees. This section presents some new logic equations for binary adders which solve this problem.

Let A_i and B_i be the i bits of the input data and C_{i-1} the carry-in for stage i. The usual method for computing the carry-out C_i is

$$C_i = G_i + P_iC_{i-1} \tag{1}$$

where

$$G_i = A_iB_i \tag{2}$$

and

$$P_i = A_i \oplus B_i \tag{3}$$

where G_i is the carry generate signal and P_i the carry propagate signal. Expanding (1) yields

$$C_i = G_i+P_iG_{i-1}+P_iP_{i-1}G_{i-2}+\cdots+P_iP_{i-1}\cdots P_1C_0. \tag{4}$$

The sum is generated by

$$S_i = C_{i-1} \oplus A_i \oplus B_i = C_{i-1} \oplus P_i. \tag{5}$$

TABLE I
Truth Table of the P_i, G_i, and N_i

A_i	B_i	P_i	G_i	N_i
0	0	0	0	1
0	1	1	0	0
1	0	1	0	0
1	1	0	1	0

In adapting (1) and (5) to differential logic, it is necessary to define their complements, which are expressed as

$$\overline{C}_i = \overline{G}_i(\overline{P}_i + \overline{C}_{i-1}) \tag{6}$$
$$\overline{S}_i = \overline{C}_{i-1} \oplus P_i. \tag{7}$$

Equation (6) produces rather inefficient structures [14], [23], [24] since it breaks the duality between the complementary network trees. This problem can be solved by redefining $\overline{C}_i$ as follows:

$$\overline{C}_i = \overline{G}_i(\overline{P}_i + \overline{C}_{i-1})(\overline{P}_i + P_i) = \overline{G}_i\overline{P}_i + \overline{G}_iP_i\overline{C}_{i-1} \tag{8}$$

but

$$\overline{G}_iP_i = P_i \tag{9}$$

and

$$\overline{G}_i\overline{P}_i = \overline{A_i + B_i} = N_i \tag{10}$$

resulting in

$$\overline{C}_i = N_i + P_i\overline{C}_{i-1}. \tag{11}$$

N_i being the complement carry generate signal. Expanding this yields

$$\overline{C}_i = N_i + P_iN_{i-1} + P_iP_{i-1}N_{i-2} + \cdots + P_iP_{i-1}\cdots P_1\overline{C}_0. \tag{12}$$

It can be observed that the definition of $\overline{C}_i$ is similar to that of C_i, replacing N_i with G_i and C_{i-1} with $\overline{C}_{i-1}$. Moreover, P_i can propagate indistinctly the carry-in and its complement. Table I shows the definition of the generate and propagate signals. It can be observed that the following properties are verified:

$$P_iG_i = P_iN_i = G_iN_i = 0 \tag{13}$$
$$P_i + G_i + N_i = 1. \tag{14}$$

Equation (13) indicates the mutually exclusive property and (14) indicates that permanently one of these signals is high.

This definition of $\overline{C}_i$ opens up new possibilities for the implementation of adders in differential logic. A specific case is the operator "o," developed by Brent and Kung [18], which enables the binary tree addition to be transformed into a parallel computation. This can easily be adapted to differential logic. The new concatenation operator "o" is defined by

$$(g_{\text{out}}, n_{\text{out}}, p_{\text{out}}) = (g_{\text{in}}, n_{\text{in}}, p_{\text{in}}) \,\text{o}\, (\hat{g}_{\text{in}}, \hat{n}_{\text{in}}, \hat{p}_{\text{in}}) \tag{15}$$

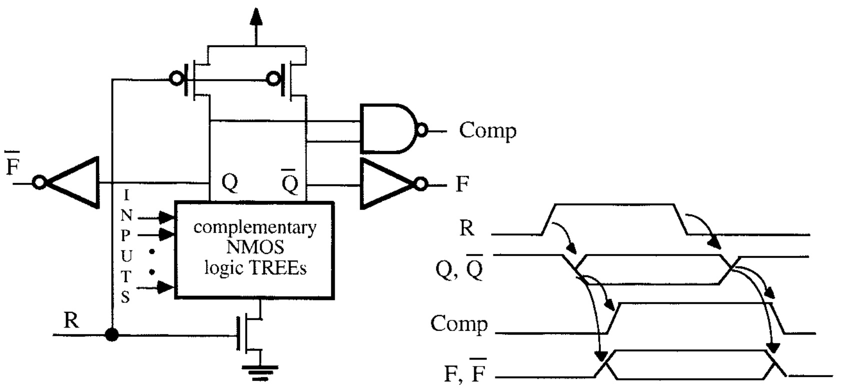


Fig. 1. DCVS logic with completion circuit and timing.

where

$$\begin{aligned} g_{\text{out}} &= g_{\text{in}} + \hat{g}_{\text{in}} p_{\text{in}} \\ n_{\text{out}} &= n_{\text{in}} + \hat{n}_{\text{in}} p_{\text{in}} \\ p_{\text{out}} &= p_{\text{in}} \hat{p}_{\text{in}}. \end{aligned} \tag{16}$$

The carry-out can be determined by

$$\begin{cases} C_i = G_i \\ \overline{C}_i = N_i \quad \text{for } i = 1, 2, \cdots, n \end{cases} \tag{17}$$

where

$$(G_i, N_i, P_i) = \begin{cases} (g_1, n_1, p_1), & \text{if } i = 1 \\ (g_i, n_i, p_i) \, \text{o} \, (G_{i-1}, N_{i-1}, P_{i-1}), & \text{if } 2 \le i \le n \\ = (g_i, n_i, p_i) \, \text{o} \, (g_{i-1}, n_{i-1}, p_{i-1}) & \\ \quad \text{o} \cdots \text{o} \, (g_1, n_1, p_1). & \end{cases} \tag{18}$$

It is easy to see that the operator described in (15) is associative and verifies the properties shown in [18] through their similitude with the original operator. Similarly, the “fo” operator presented by Lynch and Swartzlander [16] can easily be extended to differential logic to develop higher radix versions of the binary adder.

III. Implementation of Adders in DCVS Logic

DCVS logic is a differential logic style derived from domino logic made up of two complementary NMOS logic trees. This logic requires true and complementary input signals to switch the two outputs to different logic states. In self-timed circuits, its dual-rail property can be used to generate completion signals for combinational logic in a general way. Fig. 1 shows the basic DCVS circuit and its timing. When the signal control R is low (precharging phase), nodes Q and $\overline{Q}$ are precharged to high (outputs F and $\overline{F}$ to low) by the PMOS transistors. When R is high (evaluation phase) the input lines in the NMOS tree are evaluated by switching one of the outputs to low. The completion signal (Comp) for handshaking requirements in self-timed circuits can be generated by a NAND gate connected to nodes Q and $\overline{Q}$. When R is low, then Comp is low, and when R is high, Comp is high after one output is switched to low. Note that in circuits with a series of DCVS gates, the output of a DCVS gate directly drives the input of a succeeding gate; completion circuitry is only needed on the last gate in the chain.

The differential nature of DCVS logic does not allow for the efficient implementation of conventional adders. The introduction of a new complementary carry generate signal N provides C_i and $\overline{C}_i$ with symmetry properties which can be easily implemented in compact transistor sharing structures. This section describes three types of adders in DCVS logic which use the generate signal N: the RC adder, the CLA adder, and the BCL adder. All of the DCVS gates described below can be extended to other differential logic families and accept the alternative charge compensation scheme derived from the domino logic based on PMOS transistors fed back from the output, which solves the charge sharing and current leakage problems [5].

A. Ripple-Carry (RC) Adder

The RC adder has characteristics in terms of worst case delay and size which are midway between series and parallel structures. Two structures are proposed for this RC adder: one based on a conventional scheme and another more compact one with fewer devices.

Fig. 2 shows the first n-bit RC adder. Each slice of 1 b is made up of a carry generate and propagate block (GP) which computes the signals P_i, $\overline{P}_i$, G_i, and N_i in parallel, a carry bypass block (CB), and an EXOR output gate. The carry propagation is carried out through the CB blocks in a serial fashion with a high worst case delay. Fig. 3(a) shows an implementation in DCVS logic of the GP block which uses the redundance existing between the generate and propagate signals to generate all the signals in a compact structure. The CB block in Fig. 3(b) computes the carry-out using (1) and (11), and the EXOR gate in Fig. 3(c) implements (5) and (7) and generates the completion signal (Comp$_i$) by means of the NAND gate.

Fig. 4 shows the structure of the second RC adder which simplifies the generation of the carry-out by substituting the G_i and N_i signals in the CB block with input data. Each slice of the adder is made up of two EXOR gates—one input gate to generate the propagate signals (P_i, $\overline{P}_i$) and one output gate to obtain the addition (S_i, $\overline{S}_i$) and the completion signal

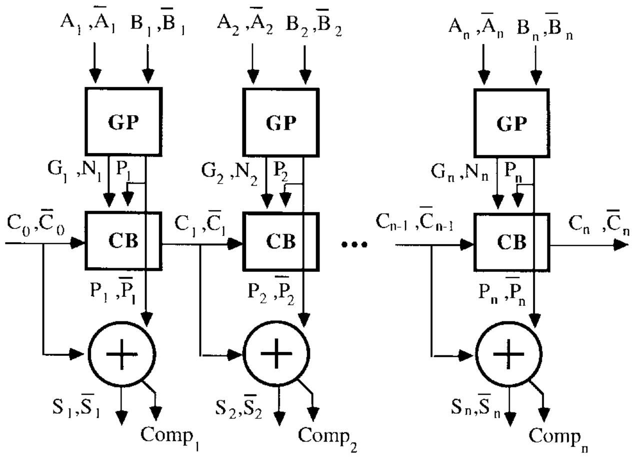


Fig. 2. Structure of the first RC adder.

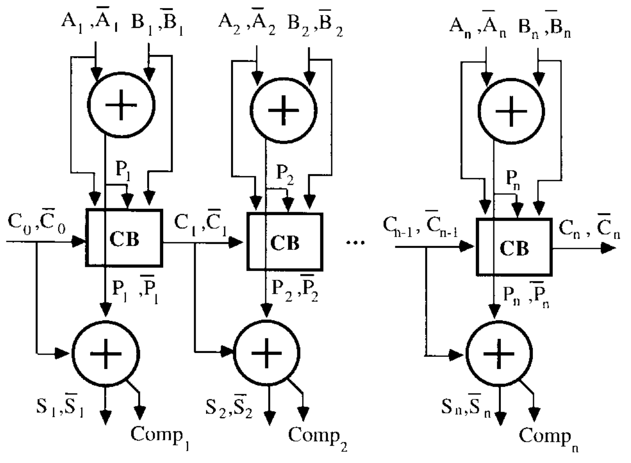


Fig. 4. Structure of the second RC adder.

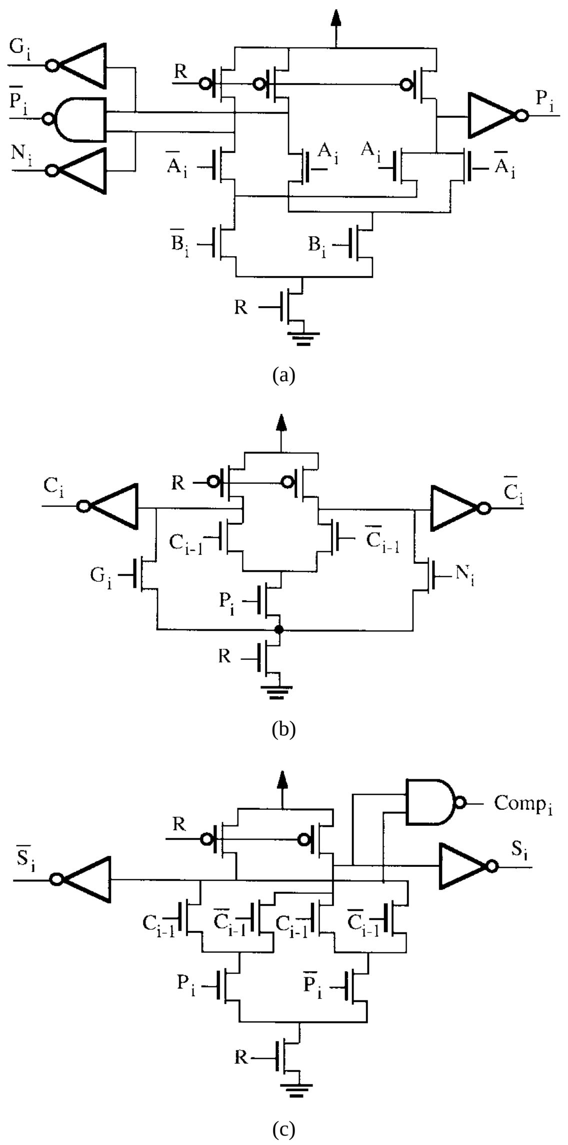


Fig. 3. (a) GP block, (b) CB block, and (c) EXOR gate of the RC adder of Fig. 2.

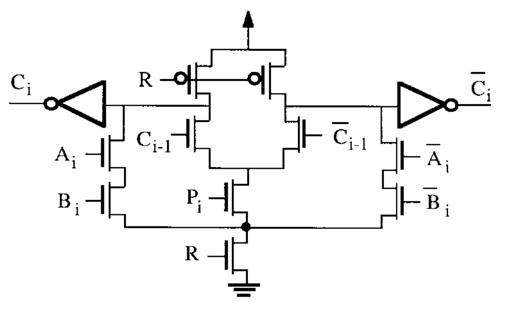


Fig. 5. Block CB of the second RC adder.

(Comp$_i$), and one carry bypass block (CB). Fig. 5 shows this CB block which combines (1), (2), (10), and (11) to generate the carry-out defined as follows:

$$\begin{aligned} C_i &= A_i B_i + P_i C_{i-1} \\ \overline{C}_i &= \overline{A}_i \overline{B}_i + P_i \overline{C}_{i-1}. \end{aligned} \qquad (19)$$

This new RC adder reduces each slice by five transistors with respect to the previous adder with a slight increase in the capacitance of the input lines. The computing time of the carry-out is also reduced when $A_i B_i = 11$ or $\overline{A_i} \overline{B_i} = 11$, as the generation time of the G_i and N_i signals in the previous GP block is eliminated. Consequently, there is a slight improvement in the average-case delay, the worst-case delay remaining similar in both adders.

B. Carry Look-Ahead (CLA) Adder

Adders based on the carry look-ahead principle are the dominant trend at the moment, since this structure allows the propagation delay of the carry-out to be reduced by calculating the carries in each stage in parallel. The Manchester carry chain (MCC) is the most popular dynamic (domino) CLA with a regular, fast, and simple structure suitable for implementation in very large scale integration (VLSI) [19]. Moreover, the recursive properties of the definition carries in the MCC have enabled the development of multi-output domino gates which have shown area and speed improvement with respect to single output ones [17]. In [20], a new enhanced CLA in multi-output DCVS logic was proposed, based on the N signal which reduces silicon area, improves circuit performance, and decreases power. Fig. 6 shows the structure of this CLA which

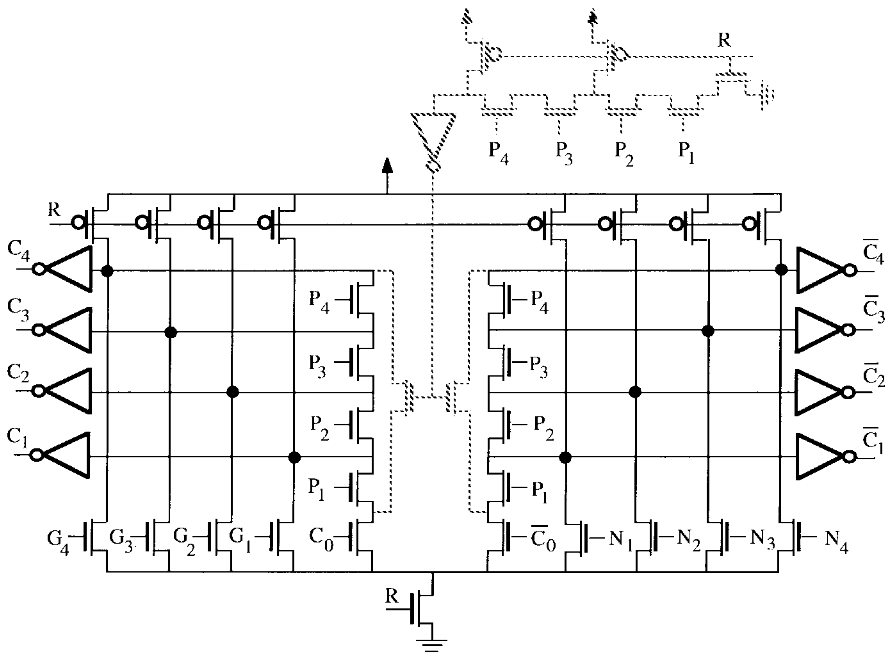
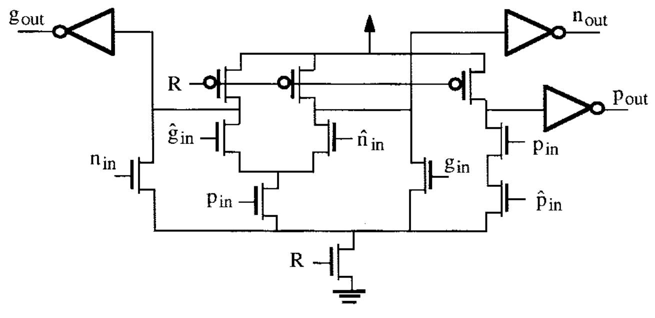


Fig. 6. A 4-b CLA gate in multi-output DCVS logic with additional reduced carry propagation circuit.

has an additional circuit (dashed lines) based on a dynamic AND gate to reduce the worst case propagation time. This gate turns on the pass transistors if all propagate signals are true, improving the overall speed of the adder. The structure of the CLA adder in DCVS logic is identical to that indicated in Fig. 2, substituting all the CB's with a CLA unit, as shown in Fig. 6. The optimum number of cascaded stages may be calculated for a given technology by simulation. Moreover, the compact structure of this new CLA and the symmetry existing between the two complementary network trees enables the application of some of the extensions developed for the MCC, such as the spanning tree carry look-ahead adder [16] and the carry-skip adder [21], among others.

C. Binary Carry Look-Ahead (BCL) Adder

The BCL adder presented by Brent and Kung [18] is an area-time optimal parallel adder with a regular, simple structure which is adequate for VLSI high-speed addition. It is made up of three functional stages. The first stage computes the generate and propagate signals. These signals act on a second stage formed by a BCL with an inverse binary tree structure. The output stage is the sum circuit and operates on the propagate signals generated in the first stage with the carry-out generated in the BCL. The BCL is a fast carry generator made up of elementary "black" and "white" processors. The "black" processor performs the function defined by the operator "o" and the "white" processor simply transmits data. Fig. 7 shows the compact structure in differential logic of the "black" processor according to the operator "o" defined in (15). This processor reduces the number of interconnection lines and the number of transistors with respect to a standard DCVS logic implementation, conferring substantial advantages in terms of speed, area, and power consumption.



$(g_{out}, n_{out}, p_{out}) = (g_{in}, n_{in}, p_{in}) o (\hat{g}_{in}, \hat{n}_{in}, \hat{p}_{in})$

Fig. 7. Compact "black" processor of a BLC adder in DCVS logic.

The average-case delay of this adder can be improved by speeding up the transmission process of the carry-out to the output stage once the BCL has been generated. This brings forward the computing of the output sum and the completion signal and, consequently, the addition time in the overall adder is reduced. Fig. 8 presents the structure of an 8-b BCL adder with a reduced average delay formed by EXOR gates and a BCL. The BCL is made up of four elementary processors (P, A, B, and C) with a tree structure. The interconnection in one same row of these processors is carried out through two common dynamic lines $(\overline{G_i}, \overline{N_i})$ as shown in Fig. 9. This interconnection is made possible by the properties of the generate and propagate signals represented in (13) and (14). With this new structure, the carry-out generated in any of these processors is transmitted directly to the output stage, unlike conventional schemes in which the carry-out generated in a "black" processor has to run through all of the processors in the row.

The processor P carries out the overall precharging of the dynamic lines $(\overline{G_i}, \overline{N_i})$ when R is low; the weak transistors labeled * avoid the problem of sharing charge. When R is

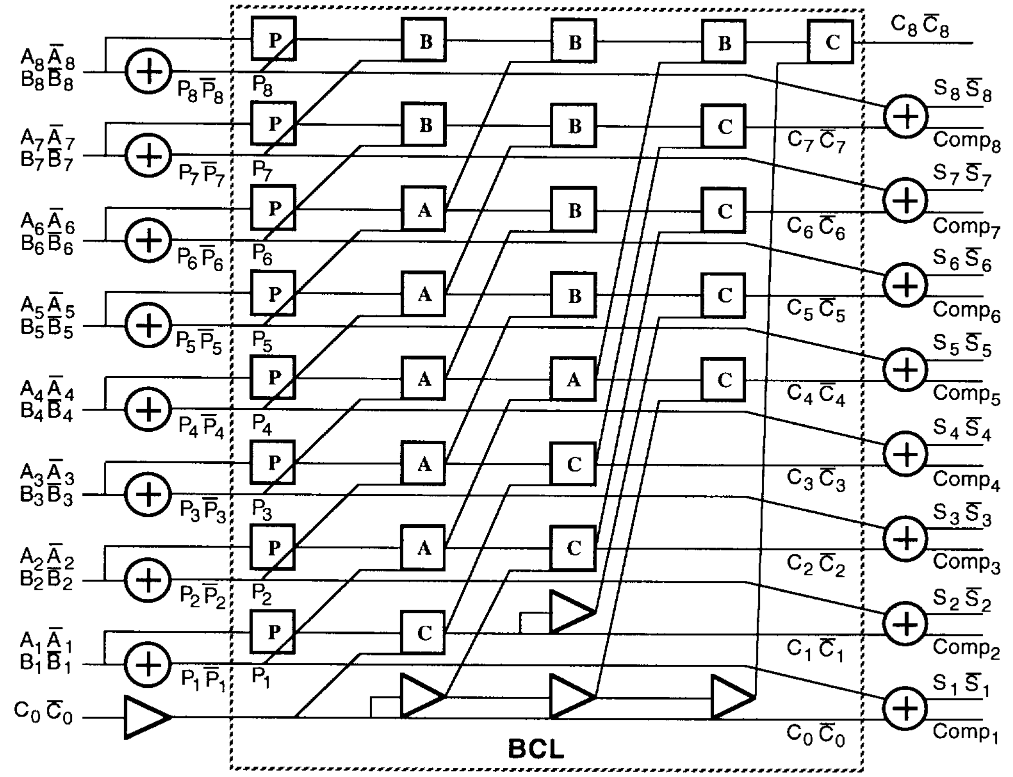


Fig. 8. An 8-b BCL adder in DCVS logic.

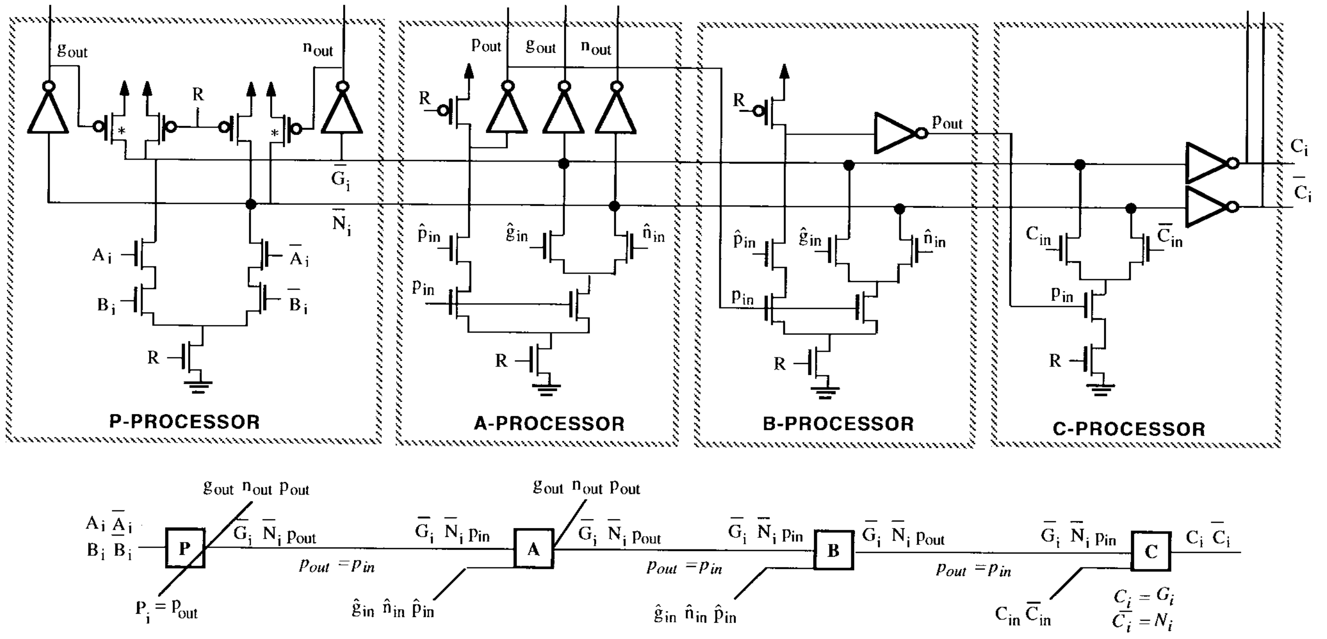


Fig. 9. Interconnection between the various processors in one row of a BCL adder.

high, if $A_iB_i = 11$ ($\overline{G_i} = 0$) or $\overline{A_i}\overline{B_i} = 11$ ($\overline{N_i} = 0$), then the carry-out generated in this processor is transmitted to the output stage through processor C and to other processors through the $g_{\rm out}$ or $n_{\rm out}$. This carry-out can also be generated in processors A, B, and C. Fig. 9 verifies that

$$\begin{aligned} G_i &= g_{\rm out} = C_i \\ N_i &= n_{\rm out} = \overline{C}_i. \end{aligned} \tag{20}$$

The main drawback of this structure is related to the relatively high capacitance value associated with the lines ($\overline{G_i}$, $\overline{N_i}$) which share the processors in the same row; in the worst case, an adder of n bits has $(\log_2 n) + 2$ processors in a row. Simulations carried out to compare the performance of this structure with that of a conventional scheme based on the processor of Fig. 7 have shown that the proposed adder reduces the average-case delay, without any significant

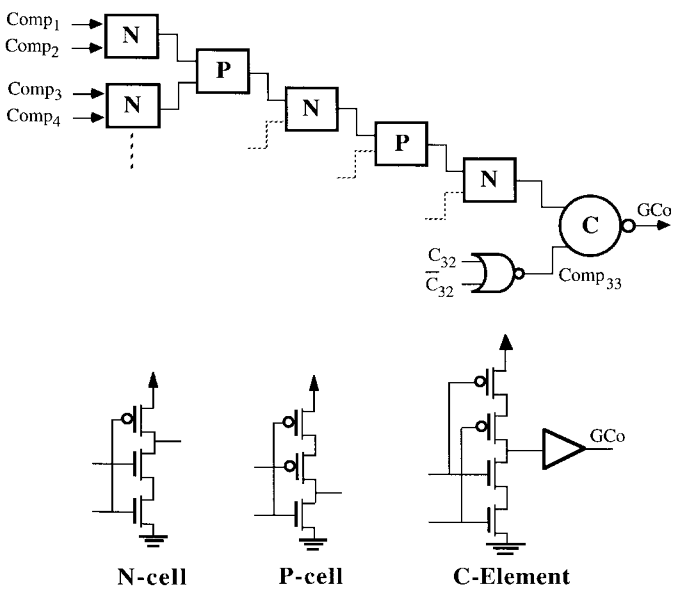


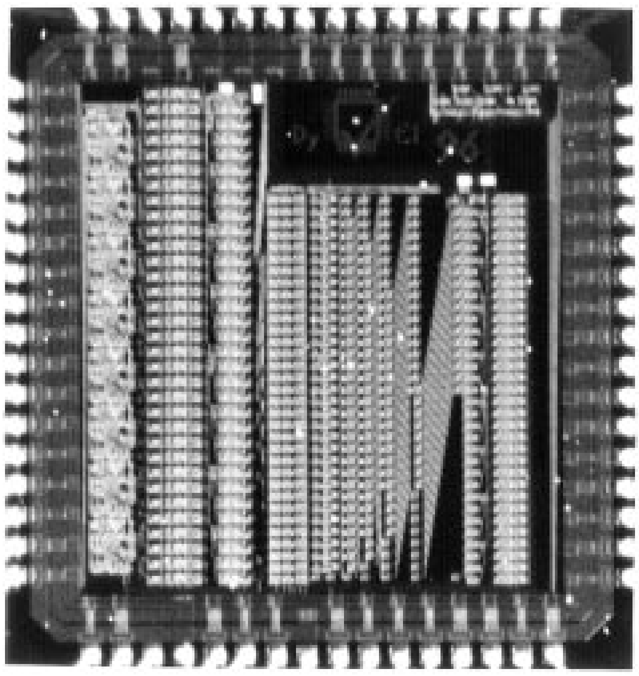
Fig. 10. Global completion circuit for a 32-b adder.

increase in the worst-case delay. This characteristic makes it more suitable in a self-timed approach where the average function delays principally govern overall throughput.

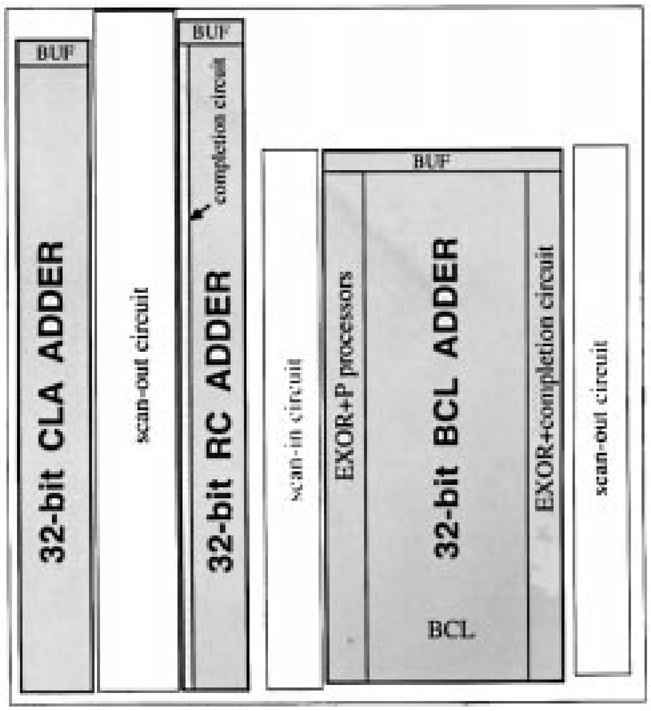
IV. Generation of a Global Completion Signal

The self-timed approach consists of computation blocks which generate completion signals that are interconnected by handshaking circuits. This completion information is used to control the movement of data between stages via handshake signals usually named *request* and *acknowledge* signals. These signals follow a standard protocol which ensures the correct data transfers between stages. In a 32-b adder, a global completion-signal (GCo) is produced by observing the 33 completion signals corresponding to 32 completion signals of the sum output (Comp_i) plus one completion signal of the carry-out (Comp_{33}). This GCo signal is generated in a completion circuit whose traditional implementation is carried out by means of a 33-input Muller C-element [22]. The output of this C-element goes high only when all its inputs are high (evaluation phase) and goes low only when all its inputs are low (precharge phase). However, the delay introduced by this C-element is of the same order or greater than the delay of the adder itself, thus considerably reducing the overall circuit performance. Hence, a high-speed completion circuit which reduces the GCo generation time has been developed. This circuit is based on the assumption that precharge delay is approximately a constant for all sum outputs (S_i, $\overline{S}_i$), since they use the same output stage (EXOR circuit) to generate the completion signals.

Fig. 10 shows the structure of the completion circuit used in 32-b adders. This circuit is made up of a binary tree of five levels of type N and P dynamic cells connected in cascade, one static NOR gate to generate the carry-out completion signal (Comp_{33}), and a 2-b dynamic C-element which generates the global completion signal GCo. During the precharging phase, the carry-out and Comp_i signals are low and the precharging



(a)



(b)

Fig. 11. (a) Micrograph of the chip and (b) arrangement in the core chip of the adders.

of all of the dynamic cells takes place: N cells are precharged to high and P to low. GCo goes to low. During the evaluation phase, the binary tree output goes to high after all the Comp_i signals are high and Comp_{33} goes to high after the carry-out is generated. CGo goes to high. The increase in binary tree speed has been achieved by limiting the number of series transistors in each dynamic cell to two and eliminating all redundancy in the circuit. The 2-b output C-element detects

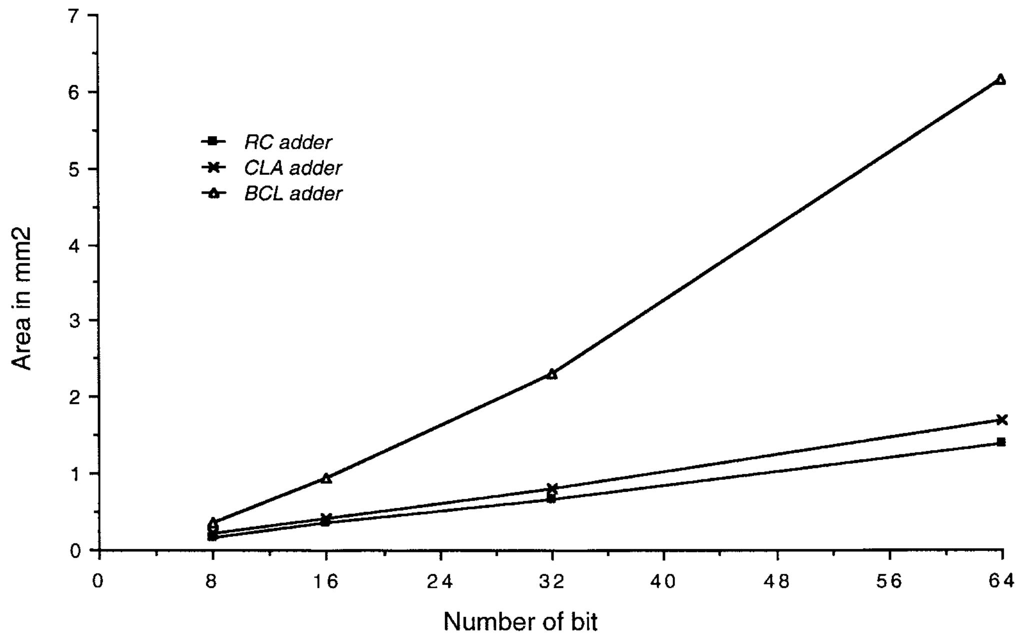


Fig. 12. Area of the adders in mm^2 for different number of bits.

both the carry-out (Comp$_{33}$) and the binary tree completion signal to generate GCo.

V. Chip Implementation and Measurement Results

Three 32-b adders for self-timed circuits were implemented in a chip to evaluate and compare their performance. The adders selected were: a 32-b RC adder as shown in Fig. 4, a 32-b CLA adder made up of eight 4-b CLA adders from Fig. 6 connected in series, and a 32-b BCL adder which is an extension of the BCL adder shown in Fig. 8. The experimental chip was fabricated using a 1.0-μm two-level metal n-well CMOS process. The micrograph of the chip is shown in Fig. 11(a) and the arrangement of the core in Fig. 11(b). All of the adders have distributed control buffers [labeled with BUF in the Fig. 11(b)] to drive the R signal of the DCVS gates. The limitation in the number of pads has made it necessary to use scan-in and scan-out circuits in order to perform the chip test (the chip is clearly I/O pad limited).

Table II lists the area and number of transistors of each adder. The RC adder occupies the smallest area, followed by the CLA adder with a factor of 1.2×, and the BCL adder with a factor of 3.5×. The relatively large size of the BCL adder is a consequence of the high number of transistors (3271 transistors as opposed to 1745 for the CLA adder and 1525 for the RC adder) and the high number of interconnection lines between processors which require a routing area of approximately 35% of the total area. This problem of the large area required for routing resulting from the need to duplicate the interconnection lines is one which is inherent in any differential logic. It can be more easily observed in Fig. 12 which represents an estimation of the sizes of the adders according to the number of bits. The RC adder and the CLA adder have an approximately linear area increase, since the number of interconnection lines is reduced, whereas the BCL adder presents a quadratic growth with a high cost in routing area in large size adders.

TABLE II
Area and Number of Transistors of the 32-B Adders

	RC adder	CLA adder	BCL adder
Area	274x2430μm^2	304x2667μm^2	1020x2265μm^2
N. of transistors	1525	1745	3271

The generation time of the global completion signal GCo, or addition time, provides an accurate measurement of the speed of these adders, as it indicates when the addition has finished. This time is measured from when signal control R goes to high until GCo goes to high and varies with the speed of the propagation of the carry signal across the adder. Experimental measurements with different carry propagation lengths were made in order to determine the dependence of the adder speed on the input data. To this end, a calibration for the I/O pad delay was obtained by connecting, in series, the circuitry for both an input and output pad. Figs. 13 and 14 show in graphic form the addition times for different carry propagate lengths at supply voltage $V_{\rm DD} = 5$ V and $V_{\rm DD} = 3.5$ V; the minimum voltage of the operation is 2.5 V. The best results for any carry propagation condition are obtained in the BCL adder. With a short carry propagation chain (<5) the RC adder obtains similar times, and with a long carry propagation chain (>12), the CLA adder is slightly slower. Assuming random input data, the average length of the carry propagation can be approximated by $\log_2(5n/4) = 5.32$. This means that the RC adder has the best speed–area characteristics under random operand conditions. However, it is slow for long carry propagation paths.

The addition time can be broken down into three components: delay time of control buffers (t_b), computation time of addition (t_a), and delay time of completion circuit (t_d). The t_b corresponds to the delay in the distributed control buffers. The HSPICE simulation results of t_b are shown in Table III. The t_d

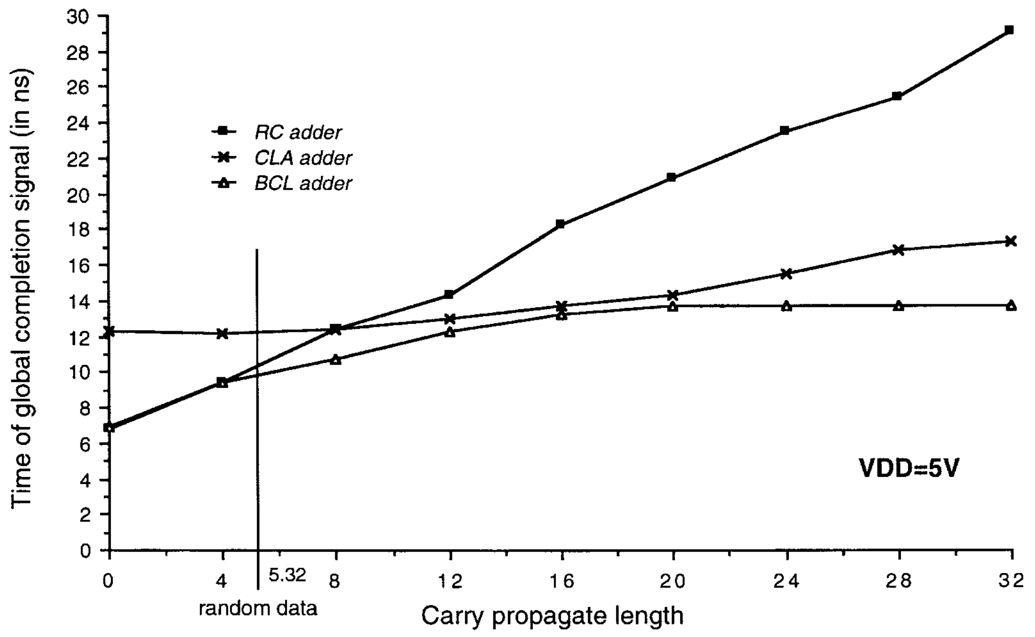


Fig. 13. Addition time for different carry propagate lengths at $V_{\rm DD} = 5$ V.

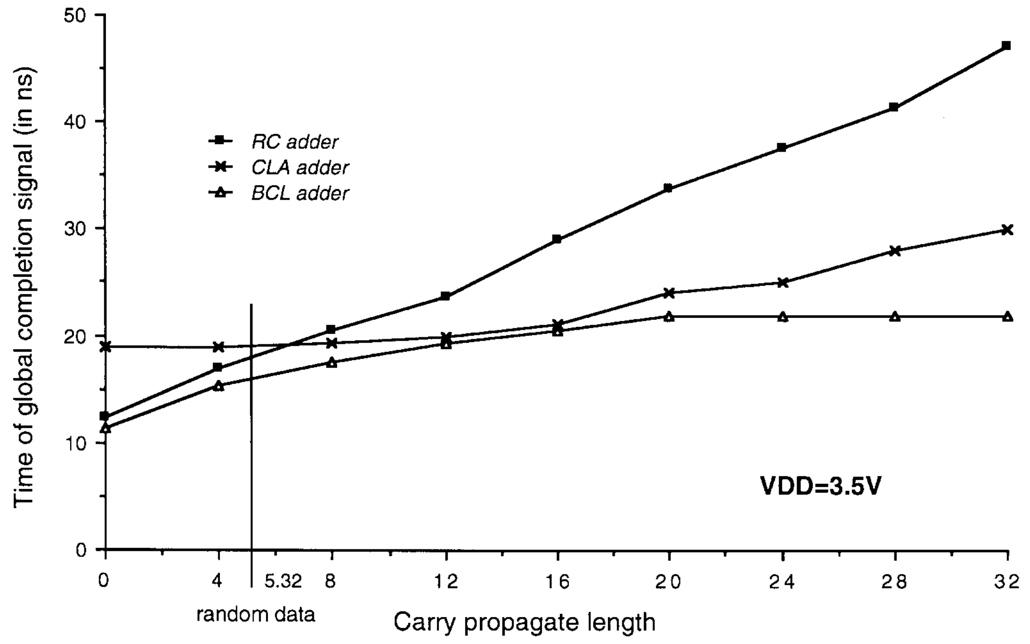


Fig. 14. Addition time for different carry propagate lengths at $V_{\rm DD} = 3.5$ V.

TABLE III
SIMULATED DELAY TIME (t_b) OF BUFFER DRIVERS

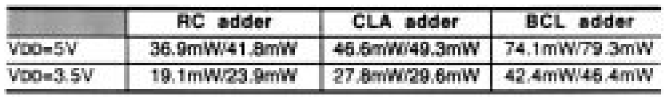
	RC adder	CLA adder	BCL adder
VDD=5V	0.7ns	0.8ns	0.8ns
VDD=3.5V	1.0n.s	1.3n.s	1.4n.s

varies slightly with the generation sequence of the completion signals (Comp$_i$) and is similar in the three adders, as they have the same completion circuit. The HSPICE simulation results show a min/max delay time of 3.2 ns/4.1 ns at $V_{\rm DD} = 5$ V and of 4.2 ns/5.3 ns at $V_{\rm DD} = 3.5$ V. The t_a indicates the addition time which is variable with the number of stages through which the carry must propagate during an addition; t_a can be estimated by subtracting the t_b and t_d times from the addition time.

The precharge time is the delay time from when signal control R goes low until GCo goes low. During this time, the precharging phase of the DCVS gates and the precharging of the completion circuit are carried out. The precharge time is identical in all the adders as they have the same output stage (EXOR circuit) to generate the completion signals. Its value is 4.6 ns at $V_{\rm DD} = 5$ V and 7.3 ns at $V_{\rm DD} = 3.5$ V.

Finally, Table IV shows the dynamic power consumption at 10 MHz without carry propagation (minimum value) and with

TABLE IV
DYNAMIC POWER CONSUMPTION (MIN/MAX) AT 10 MHz



	RC adder	CLA adder	BCL adder
VDD=5V	36.9mW/41.8mW	46.6mW/49.3mW	74.1mW/79.3mW
VDD=3.5V	19.1mW/23.9mW	27.8mW/29.6mW	42.4mW/46.4mW

longest carry propagation path (maximum value). Regarding dynamic power, the BCL is the worst structure, followed by the CLA adder, the best results being obtained by the RC adder.

VI. CONCLUSION

The use of the generate signal (N) enables the efficient implementation of adders in differential logic with a better speed/area performance than conventional implementations. Three 32-b self-timed adders in DCVS logic have been fabricated and the area, speed, and power consumption have been measured and compared. The RC adder has the best performance for random input data. Its design regularity and compact structure are suitable for implementation in VLSI architecture. However, the delay is significantly influenced by the length of the carry propagation path, and it is not recommended in asynchronous circuits with nonrandom input operands. The BCL adder is the fastest for any input data condition, but it has a high cost in chip area as it is based on elementary processors which require a large routing area. Moreover, its high number of transistors results in a high dynamic power consumption. The CLA adder offers an intermediate option. The compact shared transistor structure of the CLA unit reduces the number of interconnection lines with an area increase of 20% and an increase in transistors of 12% in comparison with the RC adder. The average case delay is slightly greater than that of the above adders with an addition time which increases slowly with the carry propagate length even for adders with a high number of bits. Moreover, this CLA unit is compatible with other high-speed adders in domino logic, such as the spanning tree CLA [16] and the carry-skip adder [21].

ACKNOWLEDGMENT

The author would like to thank M. A. Manzano for the comments which resulted in an improved manuscript.

REFERENCES

[1] T. H. Meng, *Synchronization Design for Digital Systems.* Boston/Dordrecht/London: Kluwer, 1991.
[2] S. Hauck, "Asynchronous design methodologies: An overview," *Proc. IEEE,* vol. 83, pp. 69–93, Jan. 1995.
[3] K. M. Chu and D. L. Pulfrey, "Design procedures for differential cascode voltage switch circuits," *IEEE J. Solid-State Circuits,* vol. SC-21, pp. 1082–1087, Dec. 1986.
[4] ——, "A comparison of CMOS circuit techniques: Differential cascode switch logic versus conventional logic," *IEEE J. Solid-State Circuits,* vol. SC-22, pp. 528–532, Aug. 1987.
[5] P. Ng, P. T. Balsara, and D. Steiss, "Performance of CMOS differential circuits," *IEEE J. Solid-State Circuits,* vol. 31, pp. 841–846, June 1996.
[6] N. Kanopoulus, D. Pantzartzis, and F. R. Bartram, "Design of self-checking circuits using DCVS logic: A case study," *IEEE Trans. Comput.,* vol. 41, pp. 891–896, July 1992.
[7] G. M. Jacobs and R. W. Brodersen, "A fully asynchronous digital signal processor using self-timed circuits," *IEEE J. Solid-State Circuits,* vol. 25, pp. 1526–1537, Dec. 1990.
[8] J. V. Woods, P. Day, S. B. Further, J. D. Garside, N. C. Paver, and S. Temple, "AMULET1: An asynchronous ARM microprocessor," *IEEE Trans. Comput.,* vol. 46, pp. 385–398, Apr. 1997.
[9] R. Ramachandran and S. L. Lu, "Efficient arithmetic using self-timing," *IEEE Trans. VLSI Syst.,* vol. 4, pp. 445–454, Dec. 1996.
[10] A. W. Burks, H. H. Goldstine, and J. von Neumann, "Preliminary discussion of the logical design of an electronic computing instrument," in *Inst. Advanced Study,* Princeton, NJ, June 28, 1947.
[11] G. W. Reitwiesner, "The determination of carry propagation length for binary addition," *IRE Trans. Electron. Comput.,* vol. 9, pp. 35–38, Mar. 1960.
[12] H. C. Hendrickson, "Fast high-accuracy binary parallel addition," *IRE Trans. Electron. Comput.,* vol. 9, pp. 465–469, Dec. 1960.
[13] J. D. Garside, "A CMOS VLSI implementation of an asynchronous ALU," in *Proc. IFIP Conf. Asynchronous Design Methodologies,* Manchester, England, Apr. 1993, pp. 181–192.
[14] M. A. Franklin and T. Pan, "Performance comparation of asynchronous adders," in *Proc. Int. Symp. Advanced Res. Asynchronous Circuits and Syst.,* Salt Lake City, UT, Nov. 1994, pp. 117–125.
[15] D. J. Kinniment, "An evaluation of asynchronous addition," *IEEE Trans. VLSI Syst.,* vol. 4, pp. 137–140, Mar. 1996.
[16] T. Lynch and E. E. Swartzlander, "A spanning tree carry lookahead adder," *IEEE Trans. Comput.,* vol. 41, pp. 931–939, Aug. 1992.
[17] I. S. Hwang and A. L. Fisher, "Ultrafast compact 32-bit CMOS adders in multiple-output domino logic," *IEEE J. Solid-State Circuits,* vol. 24, pp. 358–369, Apr. 1989.
[18] R. P. Brent and H. T. Kung, "A regular layout for parallel adders," *IEEE Trans. Comput.,* vol. C-31, pp. 260–264, Mar. 1982.
[19] N. Weste and K. Eshraghian, *Principles of CMOS Design: A Systems Perspective.* Reading, MA: Addison-Wesley, 1985.
[20] G. A. Ruiz, "Compact four bit carry look-ahead CMOS adder in multi-output DCVS logic," *Electron. Lett.,* vol. 32, no. 17, pp. 1556–1557, Aug. 15, 1996.
[21] P. K. Chan and M. D. F. Schlag, "Analysis and design of CMOS Manchester adders with variable carry-skip," *IEEE Trans. Comput.,* vol. 39, pp. 983–992, Aug. 1990.
[22] T.-Y. Wuu and S. B. K. Vrudhula, "A design of a fast and area efficient multi-input Muller C-element," *IEEE Trans. VLSI Syst.,* vol. 1, pp. 215–219, June 1993.
[23] S. L. Lu and M. D. Ercegovac, "Evaluation of two-summand adders implemented in ECDL CMOS differential logic," *IEEE J. Solid-State Circuits,* vol. 26, pp. 1152–1160, Aug. 1991.
[24] M. Renaudin and B. El Hassan, "The design of fast asynchronous adders and their implementation using DCVS logic," in *Proc. ISCAS'94,* London, UK, May 1994, pp. IV.291–IV.294.

Gustavo A. Ruiz was born in Burgos, Spain, in 1962. He received the M.Sc. degree in physics in 1985 from the University of Navarra, Spain, and the Ph.D. degree in physical science in 1989 from the University of Cantabria, Santander, Spain.

Since 1985, he has been with the Department of Electronics and Computers at the University of Cantabria, where he is currently an Associate Professor. His current research interests are mainly focused on VLSI architectures for signal processing and high-speed arithmetic circuits.