Self-Checking Carry-Select Adder Design Based on Two-Rail Encoding

Dilip P. Vasudevan, Parag K. Lala, and James Patrick Parkerson

***Abstract*—Carry-select adders are one of the faster types of adders. This paper proposes a scheme that encodes the sum bits using two-rail codes; the encoded sum bits are then checked by self-checking checkers. The multiplexers used in the adder are also totally self-checking. The scheme is illustrated with the implementation of a 2-bit carry select adder that can detect all single stuck-at faults on-line; the detection of double faults is not guaranteed. Adders of arbitrary size can be constructed by cascading the appropriate number of such 2-bit adders. A range of adders from 4 to 128 bits is designed using this approach employing a 0.5-μm CMOS technology. The transistor overhead in implementing these self-checking adders varies from 19.51% to 20.94%, and the area overhead varies from 16.07% to 20.67% compared to adders without built-in self-checking capability.**

***Index Terms*—On-line fault detection, self-checking checker, transient faults, two-rail code.**

I. Introduction

ARITHMETIC operations are frequently used in many VLSI-based systems [1]. The design of faster and highly reliable adders is of major importance in such systems. Over the years several types of adders have been designed, for example ripple-carry, carry-skip, carry-select adder, etc. Each type of adder has different area and delay constraints, as shown in Table I [2].

As can be seen in Table I, the carry-select adder is one of the faster types of adders, and has smaller area overhead than all other types of adders except for the carry-skip adder. Fig. 1 shows the block diagram of an n-bit carry-select adder. The main difference between a carry-select adder and a ripple-carry adder is that in a ripple-carry adder the carry has to ripple through all full-adders, but in the case of a carry-select adder the carry has to pass through a single multiplexer.

As discussed above, the carry-select addition process results in faster addition. To guarantee reliable operation of such an adder the detection of faults in the adder, especially transient faults, is extremely important. The probability of transient faults occurring in modern VLSI systems has grown significantly because of the shrinkage in transistor dimensions [3]. In VLSI systems transient faults can only be detected by built-in on-line fault detection, i.e., self-checking capability. The design of self-checking adders based on arithmetic residue codes was first proposed in [4], [5]. However, arithmetic codes cannot detect the presence of all single faults in the circuit [6], [7]. In addition the checkers for such codes are complex and the overhead is high. An alternative method for implementing self-checking ripple carry adders is parity prediction [8]. This technique detects errors only at the output of an adder. In the presence of a fault in the carry, the fault gets propagated to other outputs and remains undetected [9]. In recent years, several techniques have been proposed for online fault detection in various types of adders [10]–[14], [16], [17]. A technique for an online testable carry-select adder based on time redundancy concepts is given in [15]. The drawback of this technique is that the computation time is more than double the execution time, thus making the addition operation extremely slow. In [18] a technique for designing self-checking carry-select adders based on a simplified design of such adders [16], [17] has been presented; the area overhead of the resulting adders is very high (40%). Another technique for an online testable carry-select adder using 4-bit self-checking carry-select adders as building blocks has been proposed in [19]. The disadvantage of this approach is the high area overhead resulting from the complex reconfiguration logic, and also the multiplexers and demultiplexers at the outputs are not self-checking.

This paper proposes a design scheme for implementing self-checking carry-select adders by cascading totally self-checking 2-bit adders. The resulting carry-select adders satisfy the following criteria [9].

1) They are totally self-checking for all single faults.
2) Have a built-in compact checker.

This paper is organized as follows. Section II presents the design of a self-checking 2-bit carry-select adder. Section III describes the configuration logic required by the 2-bit adder. Section IV provides the transistor level design of

TABLE I
Area and Speed of Different Adders (Taken From [2])

Type of adder	*Area (10^5 μm^2)*	*Delay (ns)*	*Overhead (%)*
Ripple-carry	2.65	28.3	34
Manchester	2.44	14.7	35
Carry-skip	3.46	11.3	27
Carry-select	5.97	9.2	30
Conditional-sum	6.07	10.6	31

Manuscript received February 2, 2005; revised February 21, 2006. This paper was recommended by Associate Editor S.-G. Chen.

D. P. Vasudevan is with the University of Edinburgh, Edinburgh, EH9 3JZ, U.K. (e-mail: D.P.Vasudevan@sms.ed.ac.uk).

P. K. Lala is the with the Electrical at the Texas A&M University at Texarkana, Texarkana, TX 75505 USA (e-mail: parag.lala@tamut.edu).

J. P. Parkerson is with the Department of Computer Science and Computer Engineering, University of Arkansas, Fayetteville, AR 72701 USA (e-mail: jparkers@uark.edu).

Digital Object Identifier 10.1109/TCSI.2007.910537

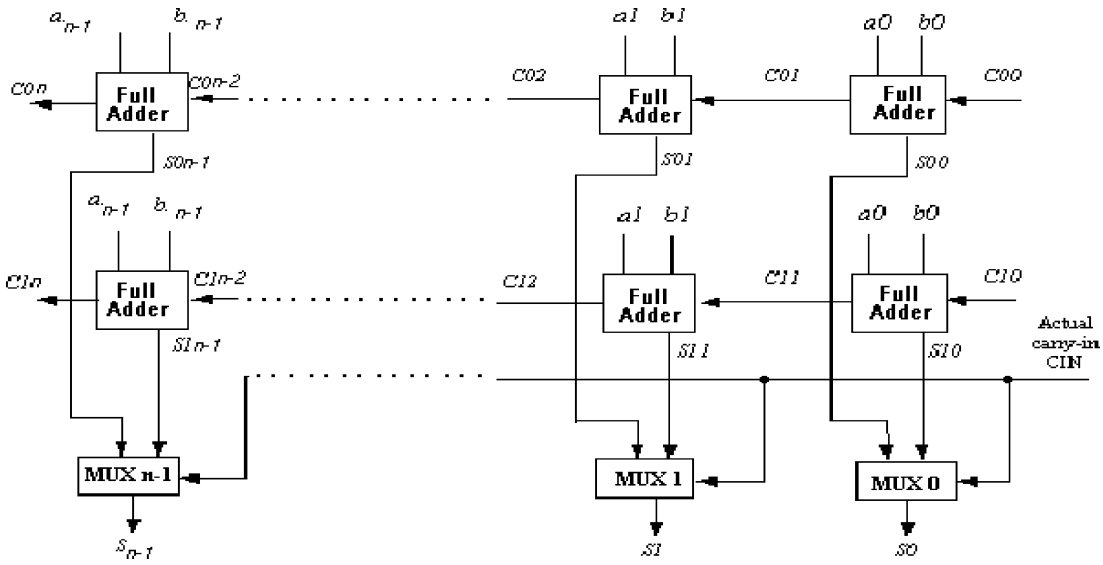

Fig. 1. n-bit carry-select adder.

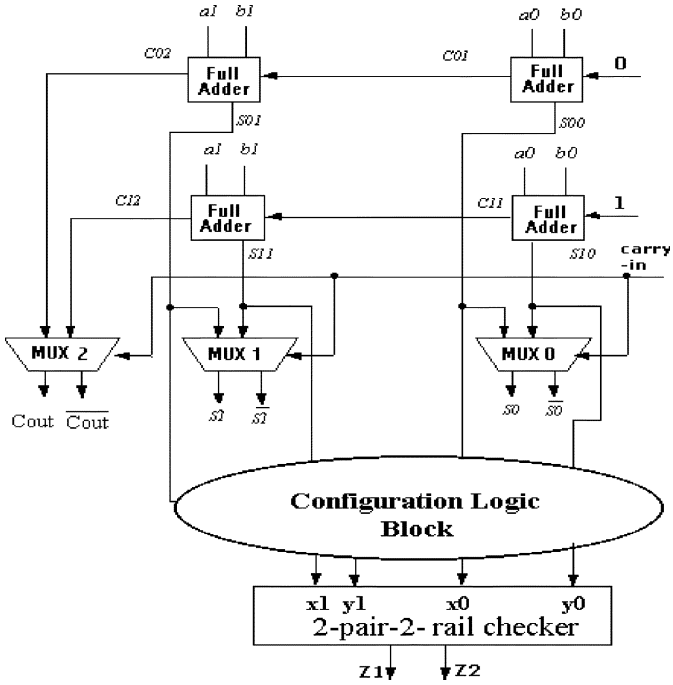

Fig. 2. Proposed design of a 2-bit self-checking carry–select adder.

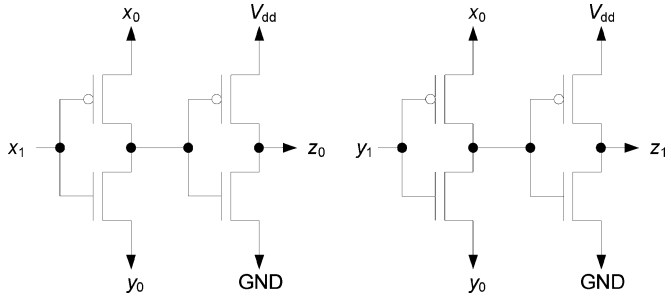

Fig. 3. Two-pair two-rail checker (taken from [20]).

the self-checking 2-to-1 multiplexer. Fault coverage is discussed in Section V. Section VI gives the overhead calculation of the self-checking adder design from 4 to 128 bits. The implementation of 64- and 128-bit adders in CMOS is discussed in Section VII. Simulation results of the implemented design are presented in Section VIII. The discussion of the results is given in Section IX, and the paper is concluded in Section X.

II. Design of a Self-Checking 2-Bit Carry-Select Adder

The design of a totally self-checking 2-bit carry-select adder is shown in Fig. 2. The 2-pair-2-rail checker can detect the presence of any single stuck-at fault in the circuit on-line. In addition to the checker, three self-checking multiplexers and an EX-NOR gate are used for detecting these faults. Any single stuck-at fault in the modified adder can be detected online. The self-checking multiplexers detect faults at the primary outputs; faults internal to the circuit are detected by allowing the effect of the fault to propagate to the output stage. A totally self-checking checker at the output stage as in Fig. 2 determines the presence of the fault. The checker has two outputs; two of the output combinations 01 and 10 (1-out-of-2 code) are considered valid code words. The sum bits with carry-in 0 and 1 generated internally by the adder are used as inputs to the checker. A nonvalid checker output, i.e., 00 or 11 indicates the presence of a fault in the circuit or in the checker. A valid code word is generated only when every input pair of (Xj, Yj) is complementary for all $j = 1$ to n. In Fig. 2 one of the inputs $(x0)$ in the complementary input pair to the checker is produced by the output of the adder with carry-in 0 and the second one $(y0)$ by the adder with carry-in 1. The input $x1$ is from the output of the adder with carry-in equal to $C01$. The input $y1$ is from the output of the EX-NOR gate whose inputs are $S00$ and $S11$.

The two-pair two-rail checker receives pairs of inputs $(x0, y0)$ and $(x1, y1)$ where $x0 = y0'$ and $x1 = y1'$. The outputs of the checker are also in two-rail form $z0 = z1'$. It has been shown in [20] that a totally self-checking 2-pair-2-rail

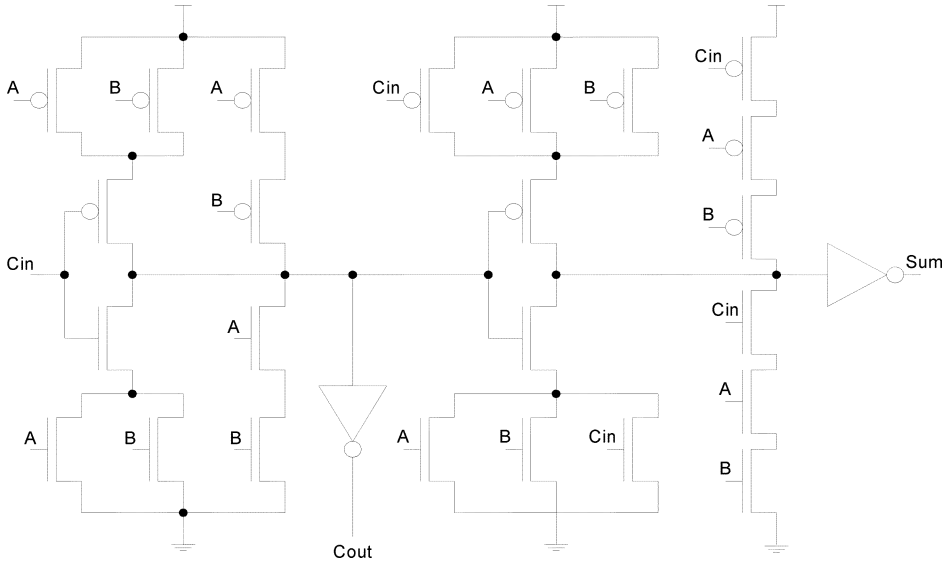

Fig. 4. Full adder.

checker can be implemented using eight transistors as shown in Fig. 3.

It should be clear from the above discussion that a carry-select adder could be tested on-line if the checker is provided with complementary inputs, i.e., valid input code words. Hence, a configuration block that produces valid input code words for the checker from the adder's outputs is needed as shown in the Fig. 2. The design of the configuration logic is discussed in Section III.

The schematic of the full adder is implemented with 28 transistors as shown in Fig. 4.

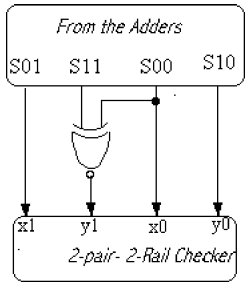

Fig. 5. Configuration logic circuit.

III. Configuration Logic for the Input of the Checker

A close observation of the sum bits generated with a carry-in of "0" and "1" shows the following property of addition:

In an addition of two n-bit numbers with the least significant bit being "0" for both the numbers, if the carry-in bit is changed from one value to another the LSB of the sum is complemented, the other bits remain unchanged.

1) Example: Note that when two binary numbers 1100 and 0100 are added together with "0" as carry-in the resultant sum ($S0$) has "0" as the least significant bit. When the same binary numbers are added with a "1" as carry-in, the resultant sum ($S1$) has "1" as the LSB

	$0 \leftarrow$ carry-in	$1 \leftarrow$ carry-in
(12)	1100	1100
(4)	0100	0100
	(16) $10000\mathbf{0}$ ($S0$)	(17) $10001\mathbf{1}$ ($S1$)

Comparing $S0$ and $S1$ it can be observed that the LSB changes from "0" to "1" whereas the remaining bits remain the same. In general the first sum bit in two full adders with the same input patterns ($A_k\ A_{k-1} \ldots .\ A_1$ and $B_k\ B_{k-1} \ldots .\ B_1$) and different carry-ins will be complementary. Any subsequent sum bit, e.g., the kth bit, will be complementary if

$$r_k \cdot r_{k-1} \cdots r_1 = 1$$

where $r_k = A_k - 1$ EX-OR B_{k-1}. Table II shows the possible input combinations for the 2-bit self-checking adder and the corresponding carry and sum outputs. $S00$ and $S01$ are the sum outputs of the two full adders with inputs ($A0$, $B0$) and ($A1$, $B1$) respectively, and carry-in "0." The corresponding carry-out signals are $C01$ and $C02$. $S10$ and $S11$ are the sum outputs of the two full adders with inputs ($A0$, $B0$) and ($A1$, $B1$) , respectively, and carry-in "1." The carry-out signals are $C11$ and $C12$.

It can be observed from the output patterns of four sum bits that the bits $S10$ and $S00$ are always complementary to each other. Thus, these lines can be directly fed to the first two inputs of the checker $x0$ and $y0$. The sum bit signals "$S11$" and "$S01$" have to be connected to the inputs $x1$ and $y1$ of the checker in such a way that their combination will produce only complementary signals. It can be observed from Table II that

$$S00 = S11 \cdot S01' + S11' \cdot S01 = S11 \text{ EX-OR } S01.$$

Thus $S01 = S00$ EX-OR $S11$; hence $S01$ and ($S00$ EX-NOR $S11$) form the complementary input pair to the checker. Fig. 5

TABLE II
ALL POSSIBLE COMBINATIONS OF BIT PATTERNS TO THE PROPOSED SELF-CHECKING ADDER

				Carry-in = 1				Carry-in = 0			
A1	B1	A0	B0	S11	C12	S10	C11	S01	C02	S00	C01
0	0	0	0	0	0	**1**	0	**0**	0	**0**	0
0	0	0	1	1	0	**0**	1	0	0	**1**	0
0	0	1	0	1	0	**0**	1	0	**0**	**1**	0
0	0	1	1	1	0	**1**	1	1	0	**0**	1
0	1	0	0	1	0	**1**	0	1	0	**0**	0
0	1	0	1	0	1	**0**	1	1	0	**1**	0
0	1	1	0	0	1	**0**	1	1	0	**1**	0
0	1	1	1	0	1	**1**	1	0	1	**0**	1
1	0	0	0	1	0	**1**	0	1	0	**0**	0
1	0	0	1	0	1	**0**	1	1	0	**1**	0
1	0	1	0	0	1	**0**	1	1	0	**1**	0
1	0	1	1	0	1	**1**	1	0	1	**0**	1
1	1	0	0	0	1	**1**	0	0	1	**0**	0
1	1	0	1	1	1	**0**	1	0	1	**1**	0
1	1	1	0	1	1	**0**	1	0	1	**1**	0
1	1	1	1	1	1	**1**	1	1	1	**0**	1

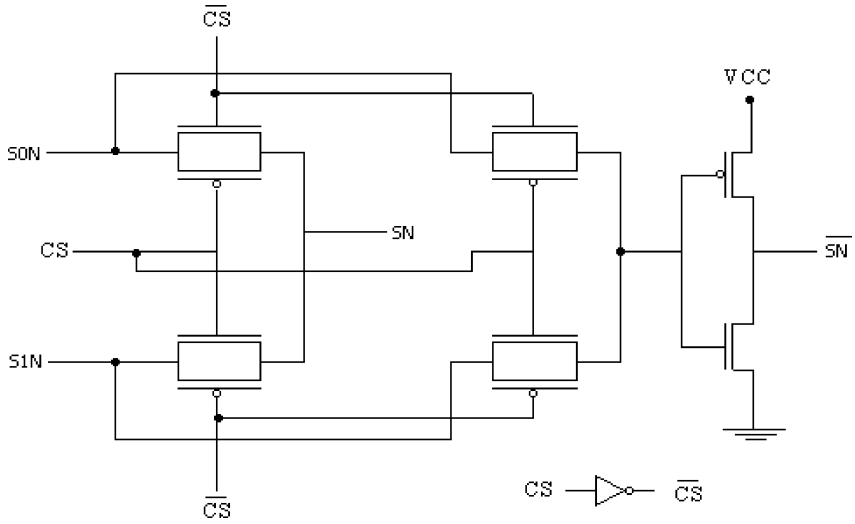

Fig. 6. Self-checking multiplexer.

shows the resulting configuration circuit with the adder outputs as its inputs. The outputs of the configuration logic are inputs $x1, y1, x0, y0$ to the checker. It only needs one EX-NOR gate and appropriate connections of the four output lines of the adder for implementing the required logic.

IV. DESIGN OF SELF-CHECKING MULTIPLEXER

In the proposed self-checking adder the checker circuit cannot detect on-line the presence of faults in the multiplexer. Thus, it is necessary to make the multiplexer self-checking in order to increase the fault coverage of the adder. In the presence of faults the outputs of the multiplexer generate nonvalid code words and can either be transferred to a checker or can be used to raise an error flag.

A self-checking multiplexer is designed using four transmission gates and an inverter as shown in Fig. 6. The multiplexer in the proposed design has two outputs SN and SN' that are complementary to each other. In the presence of a fault the outputs are identical indicating the presence of the fault. The inputs are $S0N, S1N$, and the control signal CS. When the control signal CS in the multiplexer is logic "0" input $S0N$ is transferred as the normal output and $S0N$' is transferred as the complemented output.

When CS is logic "1" input $S1N$ is transferred as the normal output where as $S1N$' is transferred as the complemented output. These outputs can be transferred as input pairs to the checker, thus enabling online detection of faults in a multiplexer. Note that only in the first stage of an n-stage adder, an inverter is used to generate CS' from the externally applied CS input. Since the multiplexer in the carry-out of the first stage of the self-checking adder generates the complement of the selected carry-out signal, the next stage adder is provided with both the carry-select signal and its complement; this applies to all other adder stages. Thus, no inverters are required in the multiplexers to generate the complement of intermediate carry signals. Although the self-checking feature in a multiplexer can be obtained by duplicating the multiplexer and comparing their outputs, this configuration will require a comparator that

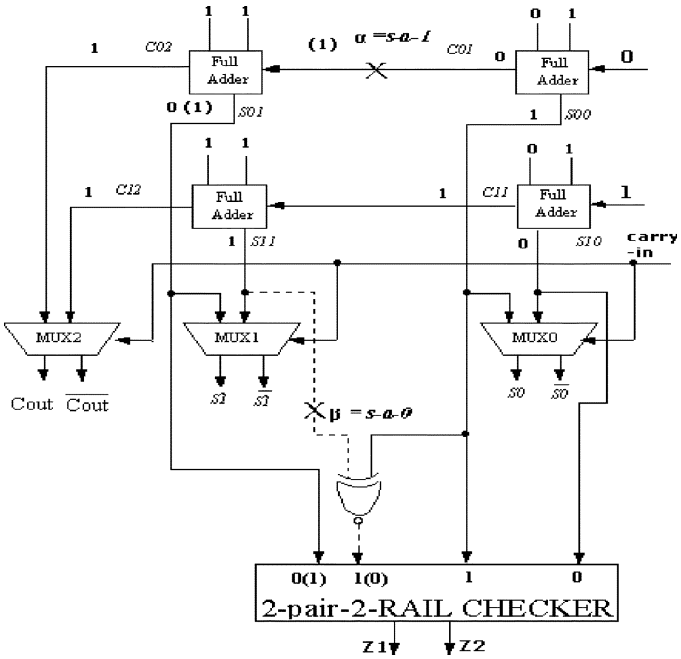

Fig. 7. Adder with a s-a-1 fault at the carry-out of the full adder.

also needs to be self-checking to guarantee the reliability of the comparison process. Consequently the overhead will be higher than the self-checking multiplexer proposed here.

V. Fault Coverage

The proposed self-checking adder design can detect the presence of a fault at all nodes in the circuit except at the primary inputs, or at the primary outputs of the adder circuit. Some of the potential nodes where the faults can be detected on-line are:

- Carry-out/carry-in of the adders
- Sum bits of the adders
- Inputs and outputs of the EX-NORs
- Inputs and outputs of the multiplexers

To illustrate let us consider the addition of two arbitrary patterns 10 and 11.

$0 \leftarrow$ carry-in	$1 \leftarrow$ carry-in
10	10
11	11
101 ($S0$)	**110** ($S1$)

Fig. 7 shows the implementation of the carry-select addition of the two numbers.

A. Fault-Free Condition

In the absence of any fault in the circuit, the checker generates valid code words. The input pairs to the 2-pair-2-rail checker are (1,0), (0,1), (0,1) and (1,0). The actual carry-in, CIN, selects the sum to be multiplexed and then propagates the final carry-out as the actual carry-in to the successive stage.

1) Case 1: A Fault at the Carry-In /Carry-Out of the Adders: If there is a fault at the carry-in or carry-out of the full adder there can be a change in the values of the sum and the carry generated. Either the sum bits leading to the checker or the carry-in leading to the successive adder is modified. This modified line will carry the faulty value in the circuit and is propagated to the checker where the fault is detected. In a few cases the change in the sum bits with the carry-in "0" changes the outputs of the configuration block.

Let us consider the case where the carry-out ($C01$) from the first bit in the first section, as shown in Fig. 7, is s-a-1 (fault α). The stuck-at fault forces the carry-out to an invalid state of 1 instead of the fault free state 0, thus forcing the full-adder corresponding to the second bit generates a sum ($S01$) of 1 instead of 0. This faulty value, propagated by the stuck-at fault is fed to the checker as one of the inputs. The other input, which is complementary in the fault free condition, is the output of the EX-NOR gate. The output of the EX-NOR gate being dependent on $S11$ and $S00$ will have the value of "1," as both $S11$ and $S00$ are 1 for the given input condition. This gives rise to an invalid condition where the input pairs are non complementary. Therefore, a nonvalid code word indicating the presence of fault α , is produced.

2) Case 2: A Fault at the Sum Bits With Carry-In "0" and Carry-In "1": In the presence of a stuck-at fault at the sum bits of the full adder, the fault is allowed to propagate to the checker inputs. The fault does not modify the values of the carry-out since the function evaluating the carry does not depend on the sum. Instead the fault might modify the logic determining the inputs to the EX-NOR and checker. Consider a fault β (s-a-0) on line $S11$ as in Fig. 7. This line is fed as input to the EX-NOR. Since this input ($S11$) is "0" and another input ($S00$) is "1," the output of the EX-NOR is "0" and hence the checker receives a noncomplementary input pair, and the fault is detected.

3) Case 3: A Fault at the Inputs/Output of the EX-NOR*:* All these faults lead to a similar type of error in the circuit. A fault at the input of the EX-NOR gate is propagated to its output, which is in turn transferred as a faulty input to the checker. Hence, any error in the EX-NOR is always reflected as a fault on the checker input. In the presence of such an error the situation becomes similar to the one discussed in case 2, where a fault on line $S11$ propagates an error on to one of the input lines of the EX-NOR. The fault modifies the line $S01$ and output of EX-NOR from (0,1) to (0,0), thus transferring a nonvalid input to the checker.

4) Case 4: Faults at the Inputs/Outputs of the Multiplexer and Actual Carry-In: Any fault present at the inputs or the outputs of the multiplexer and the actual carry-in is propagated to the primary outputs and can be detected at the output stage. As shown in Fig. 8, if $S0N = 0$, $S1N = 1$ and $CS = 0$, in the presence of a stuck-at 1 fault on one of the fan-out nodes of the input line $S0N$ the faulty value is propagated as input to the transmission gate C. This fault modifies the output (SN,SN') from (0,1) to (0,0) and hence can be detected. Similarly any fault on lines $S1N$ and CS can also be detected.

VI. Overhead

The overhead in the proposed design for a self-checking 2-bit adder is the 2-pair-2-rail checker, and the EX-NOR gates. The

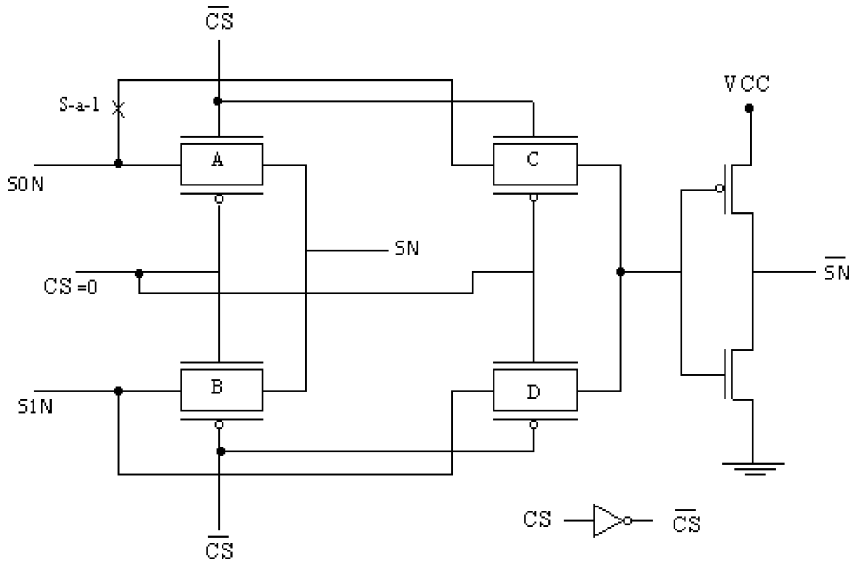

Fig. 8. Self-checking multiplexer with faults.

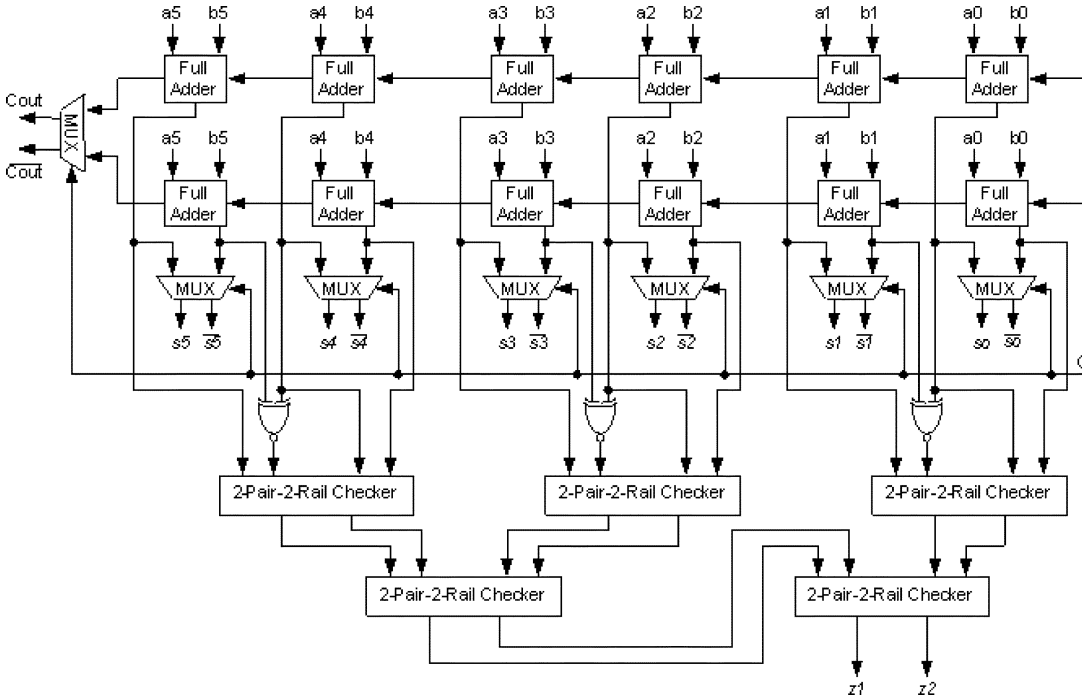

Fig. 9. 6-bit self-checking adder.

number of transistors needed to implement these units for a 2-bit addition is given below:

- full adder – 28 transistors;
- 2-pair-2 Rail checker- 8 transistors as proposed in [20];
- EX-NOR – 10 transistors;
- self-checking multiplexer- 12 transistors.

Self-checking adders of arbitrary size can be implemented by cascading self-checking 2-bit adders. The block diagram of a 6-bit self-checking adder is shown in Fig. 9. Note that MUX2 in a 2-bit adder (Fig. 2) is not needed when cascading a number of 2-bit adders except to generate the carry-out at the final stage of the larger adder. In other words in all intermediate 2-bit adders MUX2s are eliminated. Thus, the inclusion of the self-checking feature does not result in any additional delays during the normal operation of a carry-select adder implemented using the 2-bit adder cells. The 6-bit self-checking adder is shown in Fig. 9. is composed of twelve full adder blocks, seven self-checking multiplexers, three EX-NOR gates and five 2-pair-2 rail checkers.

Table III-A shows the number of transistors needed for implementing various size adders without the self-checking feature. Table III-B shows the transistor overhead when self-checking feature is incorporated; the overhead ranges from 19.51% to 20.94%. The adders were implemented in Mentor graphics-IC station using 0.5-μm CMOS technology with three metal layers. The area overhead shown in Table III-C, ranges from 16.07% to 20.67%.

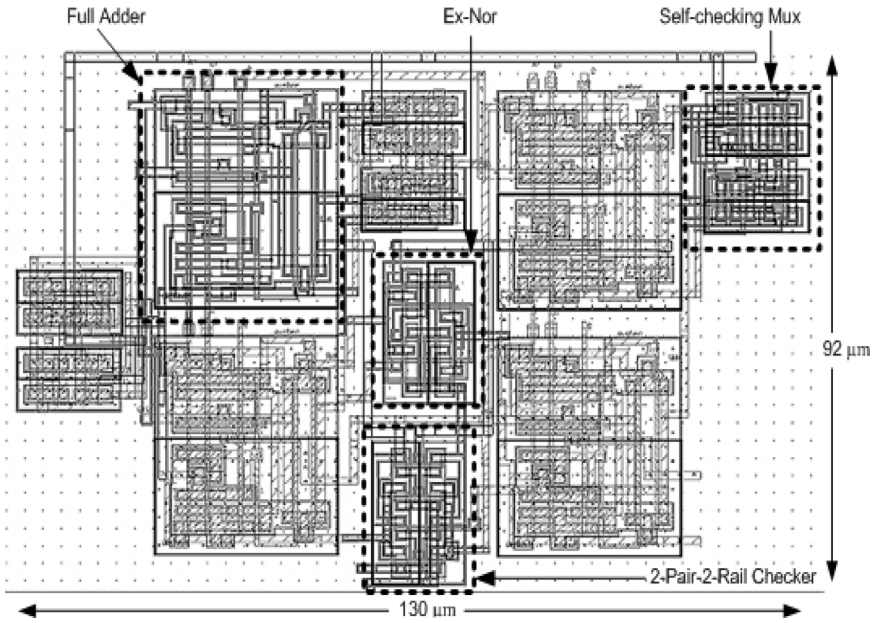

Fig. 10. Layout of 2-bit adder with checker.

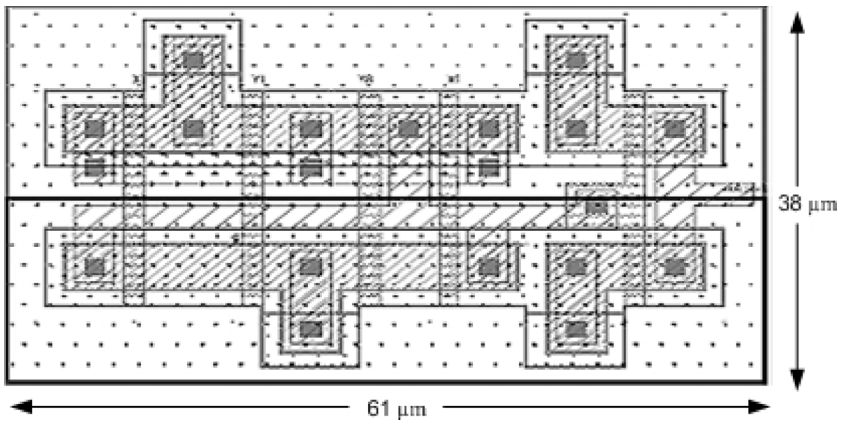

Fig. 11. Layout of the 2-pair-2-rail checker.

VII. Implementation of a 64 Bit and 128 Bit Self-Checking Adder

The layout of the self-checking 2-bit adder is as shown in Fig. 10. It consists of a group of four full adders connected individually, forming dual ripple-carry adders performing two 2-bit additions with "0" and "1" as carry-in, respectively. It occupies an area of $130 \times 92\ \mu m^2$.

The layout of 2-pair-2-rail checker is shown in Fig. 11. It has an area of $61 \times 38\ \mu m^2$. Fig. 12 shows the layout of the self-checking multiplexer. The multiplexer occupies an area of $21 \times 27\ \mu m^2$ and consists of two primary inputs, a control signal and two primary outputs. The primary outputs are complementary to each other in the absence of any fault in the circuit.

Using these blocks a 64-bit self-checking carry-select adder was implemented. The layout of the adder is shown in Fig. 13. The adder has a total area of $2152 \times 390\ \mu m^2$. In implementing the 64-bit carry-select adder, thirty-two blocks of 2-bit adders and sixty-five 2-pair-2-rail checkers were used. A 128-bit adder can be designed in a similar way; the layout of the 128-bit self-checking adder is shown in Fig. 14. The adder has a total area of $2152 \times 790\ \mu m^2$.

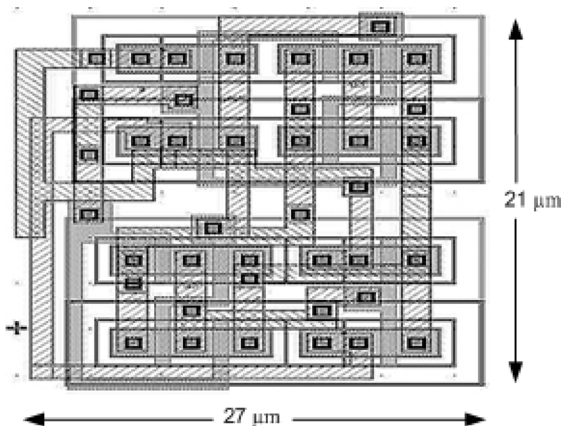

Fig. 12. Layout of the self-checking Multiplexer.

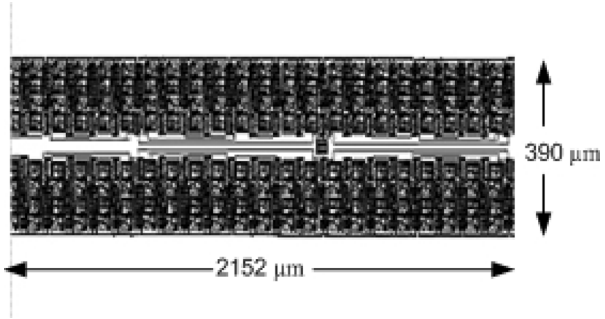

Fig. 13. Layout of 64-bit adder.

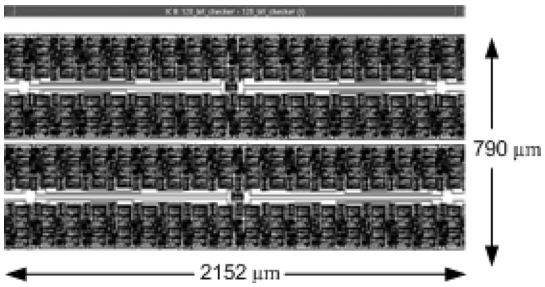

Fig. 14. Layout of 128-bit self-checking adder.

VIII. Simulation Results

The self-checking 64-bit carry-select adder was simulated in QuickSim II and the output waveforms are as shown in Figs. 15 and 16. For simplicity only the carry inputs and the least significant 4 bits, i.e., $S60 \ldots . S63$, and the 2-pair-2-rail checker outputs (Z1 and Z2) are shown in the figures. It is observed that the propagation delay for the sum is 4.3 ns and the delay for the checker is 2.9 ns.

Fig. 15 shows a simulation of the propagation delay for a 64-bit self-checking carry-select adder.

Fig. 16 shows the behavior of the self-checking 64-bit adder for two different cycles. In the first cycle, the outputs are simulated in the presence of a fault and in the second cycle the outputs are simulated without any faults. The fault as indicated in Fig. 16 is a s-a-0 fault on line C10. C10 is at logic "1" in the fault free case. During the first cycle, in the presence of a s-a-0 fault on line C10, the checker generates a nonvalid output code word detecting the presence of the fault, and in the second cycle i.e., the fault free case the checker produces valid a output code word.

IX. Discussion of Results

Carry-select adders are one of the fastest types of adders, however they require a larger area overhead compared to other type of adders. This is because they contain duplicated blocks that are normally implemented using ripple-carry adders. The speed of operation of a carry-select adder is affected by the adder block sizes used to implement it [21]. To achieve the optimal speed, the lower bits have smaller sizes and the blocks for the higher bits have larger sizes. The rationale behind the strategy proposed in this paper is to incorporate self-checking features into carry-select adders with low area overhead. This can be achieved if a carry-select adder is built by cascading 2-bit adder cells because the associated logic consists of only two-rail checkers and very simple configuration logic that requires 8 and 10 transistors, respectively. On the other hand, to make optimal carry-select adders with larger adder sizes self-checking, the checker and the reconfiguration logic will be significantly more complex. Theoretically it is possible to design k-pair two-rail checkers for arbitrary number of input pairs but above 4-pair two-rail checkers the overhead due to checker and configuration circuit becomes unacceptably high. For example if a 64-bit carry-select adder is designed using 4-4-8-12-12-12-12 adder blocks for optimal operation, two 4-pair, one 8-pair and four 12-pair two-rail checkers in addition to complex reconfiguration circuits will be needed for incorporating self-checking in the adder. The self-checking checkers only monitor the output of the adder circuit for detecting the presence of any fault. They

TABLE III
A. Required Number of Transistors Without Self-Checking.
B. Required Number of Transistors With Self-Checking.
C. Area Overhead

Cell	# of FAs	# of Muxes	# of transistors
4-bit	8	5	284
6-bit	12	7	420
8-bit	16	9	556
16-bit	32	17	1100
32-bit	64	33	2188
64-bit	128	65	4364
128-bit	256	129	8716

Cell	# of FAs	# of Muxes	# of EXNORs	# of TRCs	# of Transistors	Transistor overhead	% transistor overhead
4-bit	8	5	2	3	328	44	19.51
6-bit	12	7	3	5	490	70	20.00
8-bit	16	9	4	7	652	96	20.25
16-bit	32	17	8	15	1300	200	20.62
32-bit	64	33	16	31	2596	408	20.80
64-bit	128	65	32	63	5188	824	20.89
128-bit	256	129	64	127	10372	1656	20.94

# of bits	Area =(width x height) μm^2 (Without Checker)	Area =(width x height) μm^2 (With Checker)	% Area overhead
4-bit addition	226x84	249x92	20.67%
6-bit addition	340x84	369x92	18.87%
8-bit addition	456x84	485x95	20.28%
16-bit addition	914x84	947x95	17.18%
32-bit addition	1830x84	1887x95	16.61%
64-bit addition	1830x186	1887x210	16.41%
128-bit addition	1830x382	1887x430	16.07%

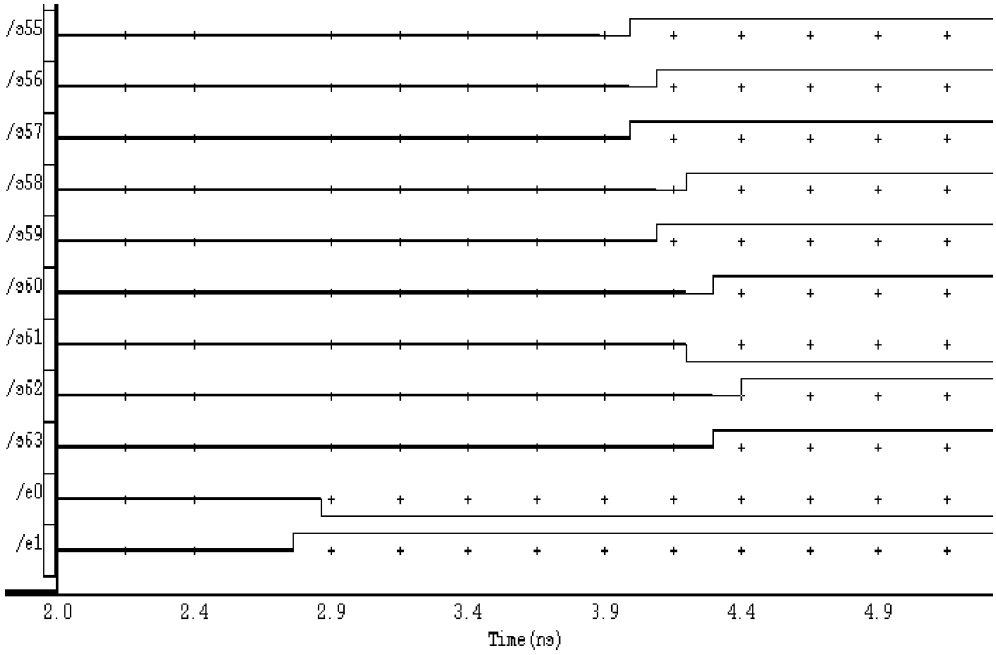

Fig. 15. Simulation showing the propagation delay for a 64-bit self-checking carry-select adder.

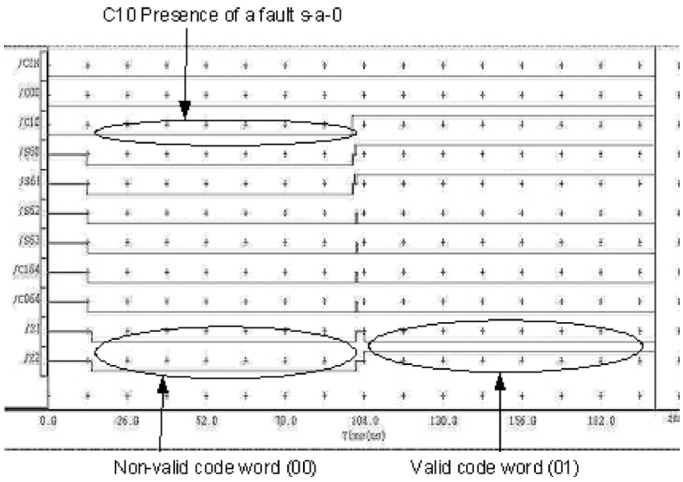

Fig. 16. Simulation results in the presence and absence of a fault.

do not affect the performance of an adder because they operate independently of the actual adder circuit. This strategy can in principle be extended to carry-select adders designed for optimal operation using bigger size adder blocks, but as mentioned previously the area overhead will be very high. Thus, there is a trade-off between optimal speed and self-checking operation.

An alternative approach to self-checking carry-select adder design is to make the ripple carry adders in a carry select adder *delay-insensitive*. This requires using dual-rail signals for input bits, sum bits and carry bits [22], and additional self-checking logic to check the dual-rail outputs. Thus, in delay-insensitive adders which are a subclass of asynchronous circuits, the number of connections is doubled; this in turn significantly increases the area overhead as well as the probability of fault occurrence. In the self-checking carry select adder proposed in this paper only two-rail signals generated by the multiplexers in the adder are used by a simple configuration circuit to drive the two-rail checkers, thus the area overhead is significantly lower.

X. Conclusion

A technique for implementing self-checking carry-select adders of arbitrary size using a 2-bit self-checking carry-select adder as the component is proposed. These adders are totally self-checking for both permanent and transient single stuck-at faults, however the detection of all double faults is not guaranteed. The amount of overhead in the proposed architecture ranges from 19.51% to 20.94% of the total number of transistors used in the design. A totally self-checking k-pair-two-rail checker indicates the presence of faults in the proposed design. An EX-NOR gate connected appropriately with the sum outputs provides valid input code words to the totally self-checking checker. In the presence of any fault a nonvalid code word is provided as input to the checker yielding a nonvalid output code word. Detailed implementations of a 64-bit and a 128-bit adder are provided to show the feasibility of the proposed design scheme.

References

[1] N. Weste and D. Harris, *CMOS VLSI Design*. Reading, MA: Addison Wesley, 2004.
[2] Z. Chen and I. Koren, "Techniques for yield enhancement of VLSI adders," in *Proc. Int. Conf. Appl. Specific Array Process.*, Strasbourg, France, Jul. 24–26, 1995, pp. 222–229.
[3] Semiconductor Industry Assoc., The 2001 National Technology Roadmap for Semiconductors, San Jose, CA, 2001.
[4] W. W. Peterson, "On checking an adder," *IBM J. Res. Dev.*, vol. 2, pp. 166–168, Apr. 1958.
[5] W. W. Peterson and E. J. Weldon, *Error-Correcting Codes*, 2nd ed. Cambridge, MA: MIT Press, 1972.
[6] G. G. Langdon and C. K. Tang, "Concurrent error detection for group look-ahead binary adders," *IBM J. Res. Dev.*, pp. 563–573, Sep. 1970.
[7] F. F. Sellers, M. Y. Hsiao, and L. W. Bearnson, *Error Detecting Logic for Digital Computers*. New York: McGraw-Hill, 1968.

[8] E. Fujiwara and K. Haruta, "Fault-tolerant arithmetic logic unit using parity based codes," *Trans. IECE Jpn.*, vol. E64, no. 10, pp. 653–660, Oct. 1981.
[9] M. Nicolaidis, "Efficient implementations of self-checking adders and ALUs," in *Proc. 23rd Annu. Int. Symp. Fault-Tolerant Comput.*, Toulouse, France, Jun. 22–24, 1993, pp. 586–595.
[10] J. C. Lo, J. C. Daly, and M. Nicolaidis, "Design of static CMOS self-checking circuits using built-in current sensing," in *Proc. 1992 Fault Tolerant Comput. Symp.*, Boston, MA, Jul. 8–10, 1992, pp. 104–111.
[11] F. W. Shih, "High performance self-checking adder for VLSI processor," in *Proc. IEEE Custom Integr. Circuits Conf.*, San Diego, CA, May 12–15, 1991, pp. 15.7/1–15.7/3.
[12] M. Nicolaidis, "Carry checking/parity prediction adders and ALUs," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 11, no. 1, pp. 121–128, Feb. 2003.
[13] W. J. Townsend, J. A. Abraham, and P. K. Lala, "On-line error detecting constant delay adder," in *Proc. Int. On-Line Testing Symp.*, Rhodes, Greece, Jul. 7–9, 2003, pp. 17–22.
[14] P. K. Lala and A. Walker, "On-line error detectable carry-free adder design," in *Proc. IEEE Int. Symp. Defect Fault Toler. VLSI Syst.*, San Francisco, CA, Oct. 24–26, 2001, pp. 66–71.
[15] F.-H. W. Shih, High Performance Self-Checking Adder Having Small Circuit Area, US PS 5,018,093, 1991.
[16] T. Y. Chang and M. J. Hsiao, "Carry-select adder using single ripple-carry adder," *Electron. Lett.*, vol. 34, no. 22, pp. 2101–2103, Oct. 1998.
[17] K. Youngjoon and L.-S. Kim, "A low power carry-select adder with reduced area," in *Proc. IEEE Int. Symp. Circuits Syst.*, Sydney, NSW, Australia, May 6–9, 2001, vol. 4, pp. 218–221.
[18] V. Otscheretnij, D. Marienfield, E. S. Sogomonyan, and M. Goessel, "Self-checking code-disjoint carry select adder with low area overhead by use of Add1 circuits," in *Proc. Int. On-Line Testing Symp.*, Madeira Island, Portugal, Jul. 12–14, 2004, pp. 31–36.
[19] B. K. Kumar and P. K. Lala, "On-line detection of faults in carry-select adders," in *Proc. Int. Test Conf.*, Charlotte, NC, Sep.-Oct. 30-2, 2003, vol. 1, pp. 912–918.
[20] J. C. Lo, "Novel area-time efficient static cmos totally self-checking comparator," *IEEE J. Solid-State Circuits*, vol. 28, no. 2, pp. 165–168, Feb. 1993.
[21] B. Parhami, *Computer Arithmetic: Algorithms and Hardware Design*. Oxford, U.K.: Oxford University Press, 2000.
[22] F. C. Cheng, S. H. Unger, M. Theobald, and W.-C. Cho, "Delay- insensitive carry-lookahead adders," in *Proc. Int. Conf. VLSI Design*, 1997, pp. 322–328.

Dilip P. Vasudevan received the B.E. degree in electronics and communications engineering from the University of Madras, Chennai, India, in 2003, and the M.S. degree in computer engineering from the University of Arkansas at Fayetteville, in 2005. He is currently working toward the Ph.D. degree at the University of Edinburgh, Edinburgh, U.K.

Parag K. Lala (M'81–SM'89–F'01) received an M.Sc.(Eng.) degree from King's College, London, U.K., and the Ph.D. degree from The City University of London, London, U.K.

He is the Cary and Lois Patterson Professor and Chair of Electrical Engineering at Texas A&M University at Texarkana. Previous to his current position he was the Thomas Mullins Chair of Computer Engineering, University of Arkansas at Fayetteville. His current research interests are in On-line testable logic, Self-healing digital system design, hardware-based DNA sequence matching, and Nanocomputing system design. He has supervised more than thirty MS and Ph.D. theses, and authored or coauthored over 130 papers. He is also the author of six books including *Self-Checking and Fault-Tolerant Digital Design* (Morgan-Kaufmann, 2001) and Principles of Modern Digital Design (Wiley, 2007).

Dr. Lala was selected *Outstanding Educator* in 1994 by the Central North Carolina section of the IEEE. In 1998, he was awarded the D.Sc. degree (in electrical engineering) by the University of London for *contributions to digital hardware design and testing, and self-checking logic design*. He was named a Fellow of the IEEE in 2001 *for contributions to the development of self-checking logic and associated checker design*. He is also a Fellow of the Institution of Engineering and Technology (until recently Institution of Electrical Engineers) in U.K.

James Patrick Parkerson received the B.S.E.E. degree (honors), and the M.S.E.E. and Ph.D. degrees from the University of Arkansas at Fayetteville.

He is an Associate Professor of computer science and computer engineering at the University of Arkansas at Fayetteville. He worked in the semiconductor industry as a senior IC design engineer for Texas Instruments, Fairchild Semiconductor, National Semiconductor, and Applied Microcircuits Corporation. His research interests include ICs, application-specific ICs, complex programmable logic device (CPLD), field-programmable gate arrays, multichip module (MCM), and printed circuit board design, and aerospace electronics. While at the University of Arkansas he supported numerous research programs as the chief design engineer at the High Density Electronics Center.