Pass Transistor Based Self-Checking Full Adder

Hamdi Belgacem, Khedhiri Chiraz, and Tourki Rached,

Abstract—Self-checking designs will gain increasing interest in industrial applications if they satisfy the following requirements: high fault coverage and reduced hardware cost with reduced design effort. The aim of this work is to contribute to reach these requirements for the design of self-checking adders/ALUs. In this paper, we present efficient self-checking implementations for adder schemes using the dual duplication code. Among the known self-checking adder designs, the dual duplicated scheme has the advantage to be totally self-checking for single faults. The drawback of this scheme is that it requires generally the maximum hardware overhead. In this work, we propose a low cost implementation for self-checking adder. The proposed design is based on a novel differential XOR gate implemented in CMOS pass transistor logic, and performed with only four transistors.

Index Terms—Totally self-checking circuits; self-checking adder; differential XOR; CMOS Logic Styles; CMOS pass transistor logic.

I. INTRODUCTION

Interest in on-line error detection continues to grow as VLSI circuits increase in complexity [1]. Concurrent checking is increasingly becoming a desirable characteristic thanks to its ability to detect transient faults that may occur in a circuit during normal operation. Accordingly, Concurrent Error Detection (CED) techniques allow the detection of transient faults, which probably not be detected in off-line testing, since they may not occur in test mode. CED also provides an opportunity for self-diagnosis and self-correction within a circuit design, especially in specific applications domains requiring very high levels of reliability (fault-tolerant computers, safety critical applications, etc.) and eventually evolving in hostile environments (e.g. space).

On the other hand, addition is one of the most fundamental operations for digital computations. Thus, much effort has been invested in research that has led to faster and more efficient ways to perform this operation [2]–[8].

In addition, designing self-checking arithmetic units is a much more complex task than designing self-checking memory systems, register files, and shifters. Arithmetic units (i.e. adders, ALUs, multipliers and dividers) are an essential element of computers. Therefore, designing efficient self-checking arithmetic units is an important challenge in the area of self-checking and fault tolerant computers. That is why from the very early developments of fault tolerant computers, an important amount of effort has been done on designing self-checking arithmetic units. The first ones are based on arithmetic residue codes [9]–[11]. Then a parity prediction scheme has been proposed in [12] and [13]. A Berger code prediction scheme has been also developed in [14], and more recently self-checking fully differential design has been proposed [15].

Parity prediction self-checking arithmetic units [16]–[18] and logic units [12] have also been proposed. This scheme detects the single errors produced on the outputs of the arithmetic unit. Parity prediction arithmetic units require the lower hardware overhead among all known self-checking arithmetic unit schemes [14], [16]. This scheme is compatible with parity checked data paths (which requires the minimum hardware overhead) and with parity encoded self-checking memory systems. It also can be modified to be compatible with Hamming SEC/DED memory systems [16], [19].

However, a single fault in an arithmetic unit can produce an error on a carry signal, which can be propagated to several outputs of the arithmetic unit. Thus, the parity scheme does not ensure the fault secure property for single faults.

In this paper, we propose a self-checking full adder design based on two-rail encoding scheme. The area overhead of the proposed design is kept within an acceptable limit by the use of a novel differential XOR gate implemented in CMOS pass transistor logic, and realized with only four transistors.

II. BACKGROUND

A. Self-checking (SC) circuits

Self-checking circuits are increasingly becoming a suitable approach to the design of complex VLSI circuits, to cope with the growing difficulty of on-line and off-line testing [1]. Self-checking circuits are class of circuits in which occurrence of fault can be determined by observation of the outputs of the circuits. An important subclass of these self-checking circuits is known as totally self-checking (TSC) circuits.

B. Totally self-checking (TSC) circuits

TSC circuits are used to detect errors concurrently with normal operation. These circuits operate on encoded inputs to produce encoded outputs. TSC checkers are used to monitor the outputs to indicate error when a non-code word is detected. The concept of TSC circuits was first proposed in [20], and then generalized in [21], as follows:

• Definition 1: A circuit is fault-secure for a set of faults, F, if for any valid input code word; any single fault, the circuit
either produces an invalid code word on the output, or doesn’t produce the error on the output.
- **Definition 2**: A circuit is self-testing for a set of faults F, if for every fault f in F, the circuit produces a non-code output for at least one code input (i.e. any single fault is detectable by some valid input code word).
- **Definition 3**: A circuit is totally self-checking if it is fault-secure and self-testing.
- **Definition 4**: A circuit is code disjoint if it always maps code word inputs into code word outputs and noncode word inputs into noncode word outputs.
- **Definition 5**: A circuit is a totally self-checking checker if it is self-testing and code-disjoint.

Thus, a totally self-checking (TSC) functional block satisfies the two following properties:
1. For any valid input code word and any single fault, the circuit, either produces an invalid code word on the output, or does not produce the error on the output (fault secure property).
2. Any single fault is detectable by some valid input code word (self-testing property).

A circuit (functional block and checker) is TSC if the functional block and the checker are both TSC. Fig. 1 gives the basic structure of TSC circuits.

![Fig. 1. Basic structure of totally self-checking circuit](image)

The checker determines whether the output of the circuit is a valid code word or not. It also detects a fault occurring within itself [23]. Double-rail checker is based on the dual duplication code as shown in Fig. 2a. It compares two input words X and Y that should normally be complementary ($Y = \overline{X}$) and delivers a pair of outputs coded in dual-rail.

![Fig. 2. (a): Principle of dual-rail checker, (b): Dual-rail checker cell](image)

A self-testing dual-rail checker can be designed as a parity tree where each XOR gate is replaced by a dual rail checker cell. The dual-rail checker cell is shown in Fig. 2(b). The resulting checker is also an easily testable circuit since only four code inputs are needed to test a dual rail checker of any length [22]. This checker is important in self-checking design since it can be used to check dual blocks (and duplicated blocks by inverting the outputs of one of them). However, its more significant use consists on the compaction of the error indication signals delivered by the various checkers of a complex self-checking circuit. Each checker delivers a pair of outputs coded in dual-rail. Thus, the dual-rail checker can compact the dual-rail pairs delivered by the various checkers of the system into a single dual-rail pair. This pair delivers the global error indication of the system. The dual rail checker of the Fig. 2(b) is a totally self-checking checker given that it is self-testing and code-disjoint [16]. It can be implemented in static CMOS, with 16 transistors.

In self-checking adder/ALUs using double-rail code, the functional block receives dual inputs and generates dual outputs. Thus, self-checking functional block is performed with a dual complementation of the basic (not self-checking) function (e.g. addition). The proposed design is an improvement of the known differential self-checking adder presented in [14]. This target structure includes two differentials XOR and a differential carry gate as shown in Fig. 3.

![Fig. 3. Target structure SC adder](image)

The differential XOR gate and the dual carry gate of the Fig. 3 can be implemented using AND, OR and NOT CMOS symmetric gates. However, this solution requires large hardware overhead. An optimized design of these gates will certainly, improve the performance and reduce the hardware overhead of the self-checking adders and ALU data paths. Consequently, a great effort had been done in order to propose static CMOS self-checking implementations [14], [15], [19].

On the other hand, new circuit techniques that go beyond full-CMOS circuitry have been given a great deal of attention in order to improve speed, area and power. These include cascode voltage switch logic (CVSL) [24], [25], differential split-level logic [26], domino-like dynamic gates [27] and pass-transistor based logics [28]–[30]. Among these, pass-transistor logic is one of the most appealing design styles, as it results in area effective, fast and robust logic circuits.

In this paper, we propose an efficient self-checking adder schemes based on an optimized XOR/XNOR gate implemented in CMOS pass transistor logic.

### III. PROPOSED DIFFERENTIAL XOR GATE

The exclusive-OR (XOR) and exclusive-NOR (XNOR) are fundamental components in full adders [31], [32], and in
larger circuits such as parity checkers [17]. The performance of these larger circuits is affected by the individual performance of the included XOR/XNOR gates.

A. Design

Pass transistor logic is attractive as fewer transistors are needed to implement important logic functions, smaller transistors and smaller capacitances are required, and it is faster than conventional CMOS. However, the pass transistor gate generates degraded signals, which slow down signal propagation. This situation will be more critical when the output signals should be propagated to next stage as is the case for the carry gate in ripple carry adder.

A novel differential XOR designed in CMOS pass transistor logic is presented in Fig. 4. This gate has dual inputs and generates dual outputs. XOR and XNOR functions are performed with only four transistors.

![Fig. 4. Differential XOR Gate](image)

The scheme has the advantage of the pass transistor CMOS gates, which are very fast and low power gates. The speed can be evaluated using the critical path (in term of transistors number). Accordingly, the critical path of the differential XOR gate is only one transistor. The drawback of this scheme (as pass transistor gates) is that it generates “weak” logical levels at the outputs, and input signals are not fully transmitted. This occurs when an NMOS transmits logic ‘1’ or a PMOS transmits logic ‘0’. To cope with this situation, the signals can be restored by adding static CMOS inverters at the outputs.

B. Analysis of the differential XOR (TSC property)

The correct operation of TSC circuits rests on following two assumptions:

1. Faults occur one at a time.
2. Sufficient time elapses between any two faults so that all the required code inputs can be applied to the circuit.

With the fault secure property it is guaranteed that, a first fault always generates detectable errors. Then, assuming that between the occurrence of two faults a sufficient time elapses so that the functional block receives all inputs required to test its faults (i.e. sufficiently long MTBF), the self-testing property guaranties that the first fault is detected before a second fault occurs in the self-checking system. This way the TSC goal is achieved for a TSC functional block [18].

In the following, we analyze the behaviour of the differential XOR gate shown in Fig. 4 in terms of fault secure and self-testing properties with respect to the set of faults including logical stuck-at faults, transistor stuck-on and transistor stuck-open faults.

For inputs, we consider the logical stuck-at fault model (gate stuck-at 0 and gate stuck-at 1 faults). For this set of faults, the scheme of Fig. 4 is fault secure for multiple faults. Table I gives the response of the gate for all inputs combinations. Any single or multiple fault on primary input will result in a non-valid code word and produce no complementary output (will be detected) as shown in Table I. (a and a~, b and b~ are normally complementary data).

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
<th>Conclusion</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0 0</td>
<td>0 0</td>
<td>Multiple fault (detected)</td>
</tr>
<tr>
<td>0 0 1 0</td>
<td>0 0</td>
<td>Single fault (detected)</td>
</tr>
<tr>
<td>0 0 1 1</td>
<td>0 0</td>
<td>Multiple fault (detected)</td>
</tr>
<tr>
<td>0 1 0 0</td>
<td>1 1</td>
<td>Single fault (detected)</td>
</tr>
<tr>
<td>0 1 0 1</td>
<td>0 1</td>
<td>OK (valid input word)</td>
</tr>
<tr>
<td>0 1 1 0</td>
<td>1 0</td>
<td>OK (valid input word)</td>
</tr>
<tr>
<td>0 1 1 1</td>
<td>0 0</td>
<td>Single fault (detected)</td>
</tr>
<tr>
<td>1 0 0 0</td>
<td>0 0</td>
<td>Single fault (detected)</td>
</tr>
<tr>
<td>1 0 0 1</td>
<td>1 0</td>
<td>OK (valid input word)</td>
</tr>
<tr>
<td>1 0 1 0</td>
<td>0 1</td>
<td>OK (valid input word)</td>
</tr>
<tr>
<td>1 0 1 1</td>
<td>1 1</td>
<td>Single fault (detected)</td>
</tr>
<tr>
<td>1 1 0 0</td>
<td>1 1</td>
<td>Multiple fault (detected)</td>
</tr>
<tr>
<td>1 1 0 1</td>
<td>1 1</td>
<td>Single fault (detected)</td>
</tr>
<tr>
<td>1 1 1 0</td>
<td>1 1</td>
<td>Multiple fault (detected)</td>
</tr>
<tr>
<td>1 1 1 1</td>
<td>1 1</td>
<td>Multiple fault (detected)</td>
</tr>
</tbody>
</table>

On the other hand, switch level fault models (stuck-open, stuck-on, bridging) and transistor level fault models (shorts, opens) are used for more accurate representations of defects [33]–[35]. We consider then, the stuck-on and stuck-open CMOS transistor model. To show that the differential XOR is TSC for this class of faults, we will examine all possible single faults (transistor stuck-on and transistor stuck-open) within the circuit of the Fig. 4.

We consider first, transistor stuck-on faults. The differential XOR gate of Fig. 4 is fault secure and self-testing for all single stuck-on transistor faults (see Table II).

<table>
<thead>
<tr>
<th>Transistor stuck-on</th>
<th>Input vector (a,b) detecting the fault</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>(0,1)</td>
</tr>
<tr>
<td>P2</td>
<td>(0,0)</td>
</tr>
<tr>
<td>N1</td>
<td>(0,0)</td>
</tr>
<tr>
<td>N2</td>
<td>(0,1)</td>
</tr>
</tbody>
</table>

We consider at present, transistor stuck-open faults. These faults may cause output floating and the circuit will have a sequential behaviour. The differential XOR is TSC for all transistors stuck-open faults, if it is fault secure and self testing for these faults.

The Fault secure property stipulate that For any valid input code word, any single transistor open fault within the gate produces an invalid code word on the output, or does not produce an error on the output (fail safe). Let’s examine the behaviour of the differential XOR gate under any single transistor open fault to make the proof that it is fault secure for this class of faults. Given that the differential XOR is contains four Transistors, there are four possible transistor open faults.

We note a, b, X and Xb the previous states of a, b, X and Xb respectively.

Z: represents the output-floating state (in fact output
remains at its previous state).

(1) Transistor P1 stuck-open

<table>
<thead>
<tr>
<th>case</th>
<th>Inputs a b</th>
<th>Outputs without fault X Xb</th>
<th>Outputs with fault X Xb</th>
<th>Conclusion</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0 0</td>
<td>0 1</td>
<td>0 Z</td>
<td>X (OK), Xb (?)</td>
</tr>
<tr>
<td>B</td>
<td>0 1</td>
<td>1 0</td>
<td>1 0</td>
<td>OK</td>
</tr>
<tr>
<td>C</td>
<td>1 0</td>
<td>1 0</td>
<td>1 Z</td>
<td>X (OK), Xb (?)</td>
</tr>
<tr>
<td>D</td>
<td>1 1</td>
<td>0 1</td>
<td>0 Z</td>
<td>OK</td>
</tr>
</tbody>
</table>

In cases B and D, the gate is fault secure because the fault does not produce error on the output. For the cases A and D, the fault secure property will not be lost and here is the proof.

**Case A (a b = 00):** The output Xb depends of the previous states of inputs, three cases are considered:

- a'b = 01 ⇒ X Xb = 10 (when a b pass from 01 to 00) ⇒ X Xb = 0Z = 00: invalid code word (will be detected)
- a'b = 10 ⇒ X Xb = 1Z = 10: valid code word (the same as the fault free gate).
- a'b = 11 ⇒ X Xb = 01 ⇒ X Xb = 1Z = 11: invalid code word (will be detected).

(2) Transistor P2 stuck-open

<table>
<thead>
<tr>
<th>case</th>
<th>Inputs a b</th>
<th>Outputs without fault X Xb</th>
<th>Outputs with fault X Xb</th>
<th>Conclusion</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0 0</td>
<td>0 1</td>
<td>0 1</td>
<td>OK</td>
</tr>
<tr>
<td>B</td>
<td>0 1</td>
<td>1 0</td>
<td>Z 0</td>
<td>?</td>
</tr>
<tr>
<td>C</td>
<td>1 0</td>
<td>1 0</td>
<td>1 0</td>
<td>OK</td>
</tr>
<tr>
<td>D</td>
<td>1 1</td>
<td>0 1</td>
<td>Z 1</td>
<td>?</td>
</tr>
</tbody>
</table>

In cases A and C, the gate is fault secure. For the cases B and D, the fault secure property is not lost.

**Case B (a b = 01):** We have three possibilities:

- a'b = 00 ⇒ X Xb = 01, (when a b pass from 00 to 01) ⇒ X Xb = Z0 = 00: invalid code word (will be detected)
- a'b = 10 ⇒ X Xb = 10 ⇒ X Xb = Z0 = 10: no error produced the on the output (fail safe).
- a'b = 11 ⇒ X Xb = Z1 → the case D.

**Case D (a b = 11):** Three possibilities are to be treated:

- a'b = 00 ⇒ X Xb = 01 ⇒ X Xb = Z1 = 01: the gate is fail safe: does not produce the error on the output.
- a'b = 01 ⇒ X Xb = Z0 → the case B.
- a'b = 10 ⇒ X Xb = 10 ⇒ X Xb = Z1 = 11: invalid code word (will be detected).

(3) Transistor N1 stuck-open

<table>
<thead>
<tr>
<th>case</th>
<th>Inputs a b</th>
<th>Outputs without fault X Xb</th>
<th>Outputs with fault X Xb</th>
<th>Conclusion</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0 0</td>
<td>0 1</td>
<td>0 1</td>
<td>OK</td>
</tr>
<tr>
<td>B</td>
<td>0 1</td>
<td>1 0</td>
<td>1 Z</td>
<td>?</td>
</tr>
<tr>
<td>C</td>
<td>1 0</td>
<td>1 0</td>
<td>1 0</td>
<td>OK</td>
</tr>
<tr>
<td>D</td>
<td>1 1</td>
<td>0 1</td>
<td>0 Z</td>
<td>?</td>
</tr>
</tbody>
</table>

In cases A and C, the gate is fault secure. For the cases B and D, the fault secure property will not be lost.

**Case B (a b = 01):** We have three possibilities:

- a'b = 00 ⇒ X Xb = 01, (when ab pass from 00 to 01) ⇒ X Xb = 1Z = 11: invalid code word (will be detected)
- a'b = 10 ⇒ X Xb = 10 ⇒ X Xb = 1Z = 10: same outputs as fault free gate (fail safe)
- a'b = 11 ⇒ X Xb = 0Z → case D.

**Case D (a b = 11):** Three possibilities:

- a'b = 00 ⇒ X Xb = 01 ⇒ X Xb = 0Z = 01: same outputs as fault free gate (fail safe).
- a'b = 01 ⇒ X Xb = Z1 = 1Z → case B.
- a'b = 10 ⇒ X Xb = 10 ⇒ X Xb = 0Z = 00: invalid code word (will be detected).

(4) Transistor N2 stuck-open

<table>
<thead>
<tr>
<th>case</th>
<th>Inputs a b</th>
<th>Outputs without fault X Xb</th>
<th>Outputs With fault X Xb</th>
<th>Conclusion</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0 0</td>
<td>0 1</td>
<td>Z 1</td>
<td>?</td>
</tr>
<tr>
<td>B</td>
<td>0 1</td>
<td>1 0</td>
<td>1 0</td>
<td>OK</td>
</tr>
<tr>
<td>C</td>
<td>1 0</td>
<td>1 0</td>
<td>Z 0</td>
<td>?</td>
</tr>
<tr>
<td>D</td>
<td>1 1</td>
<td>0 1</td>
<td>0 1</td>
<td>OK</td>
</tr>
</tbody>
</table>

In cases B and D, the gate is fault secure. For the cases A and C, the fault secure property will not be lost.

**Case A (ab = 00):** We have three possibilities:

- a'b = 01 ⇒ X Xb = 10, (when ab pass from 01 to 00) ⇒ X Xb = Z1 = 11: invalid code word (will be detected).
- a'b = 10 ⇒ X Xb = 0Z → the case C.
- a'b = 11 ⇒ X Xb = 01 ⇒ X Xb = Z0 = 10: same outputs as fault free gate (failure secure).

**Case C (a b = 10):** We have also three possibilities:

- a'b = 00 ⇒ X Xb = Z1 → the case C.
- a'b = 01 ⇒ X Xb = 10 ⇒ X Xb = Z0 = 10: invalid code word (will be detected).

The self-testing property signify that for each single transistor open fault within the gate there is at least one input vector, occurring during the circuit normal operation that detects it. To make the proof of the self-testing property of the proposed design for single transistor open faults, we propose to use fault equivalence concept.

In the simplified MOS transistor model for digital applications, a transistor acts as a switch controlled by the gate voltage. When passing, a transistor provides a resistive path between source and drain. When opened, the source to drain path has high impedance, and is effectively an open circuit. Therefore, a PMOS transistor open is equivalent to a PMOS whose gate is stuck-at 1, and a NMOS transistor open is equivalent to a NMOS whose gate is stuck-at 0.

Let us examine the four possible single transistor stuck-open faults within the XOR of the Fig. 4. (a a~ b b~) are the inputs of the differential XOR gate (a and a~, b and b~ are normally complementary data).

(1) Transistor P1 stuck-open:

“P1 stuck-open” is equivalent to “b stuck-at 1”. As P1 gate receives the signal b, then this fault is detectable by the input vectors (a a~ b b~) = (0101) and (a a~ b b~) = (1010) (see Table I).

(2) Transistor N1 stuck-open:

“N1 stuck-open” is equivalent to “b stuck-at 0”. Given that N1 gate receives the signal b, the input vector (a a~ b b~) =
(0110) or (10 10) detects this fault.

(3) Transistor P2 stuck-open:
“P2 stuck-open” is equivalent to “b~ stuck-at 1”. Since P2 gate receives the signal b~, the input vector (a ≡ b b~) =
(0110) or (10 10) detects this fault.

(4) Transistor N2 stuck-open:
“N2 stuck-open” is equivalent to “b~ stuck-at 0”. As N2 gate receives the signal b~, the input vector (a ≡ b b~) =
(0101) or (10 01) detects this fault.

All these vectors belong to the set of valid input codes. Consequently, the differential XOR is self-testing for all
single stuck transistor stuck-open faults.

In this section, we made the proof that the scheme of the
Fig. 4 is fault secure for the logic stuck-at fault model and
transistor stuck-on/stuck-open fault model. That is to say, for
any valid input code word and any single fault from the
considered class of faults the proposed differential XOR gate
either produces an invalid code word on the output (fault
detected), or does not produce the error on the output (fail
safe). Moreover, the differential XOR gate is self-testing i.e.
any single fault is detectable by some valid input code word.
In consequence, the scheme proposed in this paper is totally self-checking for all stuck-at, stuck-on and stuck-open single faults.

IV. SELF-CHECKING FULL ADDER

In this section a self-checking full adder based on the
proposed differential XOR is presented and analyzed.

A. Design

The proposed self-checking full adder includes two sub
circuits: The differential carry gate and the differential sum
gate as shown in the target design presented in Fig. 3.

In static CMOS technology, the carry gate costs 12
transistors. As the majority voting function is auto-dual, the
carry gate is simply duplicated to generate double-rail carry.
This solution requires the maximum hardware overhead with
24 transistors. To cope with this problem, the carry gate can be
performed in pass transistor CMOS technology as proposed in [31]. A significant gain in hardware overhead
could be obtained with this solution. However, the pass
transistor gates generate degraded signals, which slow down
signal propagation. This situation will be more critical when
the output signals should be propagated to next stage (e.g.
ripple carry adder, grouped carry look-ahead adder).

In [14] the differential carry gate is designed in static
CMOS with only 16 transistors (see Fig. 5). This scheme is
fault secure for stuck-at fault model. Sure enough, any single
fault on primary input will result in a non-valid code word
and produce no complementary output. In addition, this gate
is self-testing. Indeed, any single internal fault is detectable
by some valid input code word, since it will affect only one
output and produce no complementary data. Therefore, this
scheme is TSC for stuck-at fault model.

This scheme performs also, the dual generation
signals \(G_1, G_i\). These signals are useful for implementation
of fast adders such as group and full carry look-ahead ones.

The sum function \(S_i = a_i \oplus b_i \oplus \overline{C_i}\) is implemented with two
differential XOR. The first differential XOR performs signals
\(P_i = a_i \oplus b_i\) and \(\overline{P_i} = a_i \oplus b_i\). These signals and the dual carry are
the inputs of the second differential XOR that generates the
dual sum function as shown in Fig. 6.

Note that degraded outputs of the first gate (diff XOR1)
are not used to drive the transistor gates of the second gate
(diff XOR2) and thus, signals are not degraded once more. In
fact, the output voltage does not depend on the number of
switch transistors that the signal travels through. It only
depends on the gate voltage of those switches. This gate
performs also, the dual propagation signals \(\{P_i, \overline{P_i}\}\).

Fig. 7 gives the scheme of the proposed self-checking full
adder. It is implemented with the differential SUM gate of the
Fig. 6 and the static CMOS dual carry gate of the Fig. 5. This
design combines the pass transistor CMOS technology with
static CMOS technology. Inverters are added to restore
degraded signals generated by the differential SUM gate.
This fully differential implementation requires only 28
transistors.

On the other hand, a conventional (not self-checking)
one-bit full adder has two operands, A and B, and input carry
\(C_{in}\). It generates the sum \(S = A \oplus B \oplus C_{in}\) and the output carry
\(C_{out} = AB + BC + AC\). A standard static CMOS
implementation of the 1-bit full adder costs 24 transistors
[38]. The area overhead for the proposed adder is only 17%
of a traditional adder.
B. Analysis (TSC property)

We made the proof in the previous sections that, the sub-circuits of the full adder (differential XOR gate and differential carry gate) are fault-secure. Thus, the proposed differential full adder is fault-secure for the considered set of faults. We now show that the proposed full adder is self-testing by proving that every fault can be detected by at least one input codeword.

The differential XOR1 is self-testing since it receives primary inputs. The differential carry gate is also self-testing \cite{14}. However, the differential XOR2 gate receives dual-rail carry signals that are not primary inputs since they are generated by the previous stage.

As mentioned previously, transistor stuck-on/stuck-open faults are equivalent to transistor gate stuck-at 0/1. Therefore, we just have to demonstrate that any internal carry signal \( C_{i+1} \) (and \( \overline{C}_{i+1} \)) can be set to “1” or “0” from primary inputs \( a_i \) and \( b_i \).

The carry out is given by the expression:

\[
C_{i+1} = a_i b_i + P C = a_i b_i + (a_i \oplus b_i)C_i
\]

The input codeword \( a_i b_i b_i = 1010 \) sets the carry out \( C_{i+1} \) to high logical level whatever the state of the carry in \( C_i \) signal \( (a_i b_i b_i = 1010 \Rightarrow C_{i+1} \overline{C}_i + 1 = 10) \).

The input codeword \( a_i b_i b_i = 0101 \) sets the carry out \( C_{i+1} \) to low logical level whatever the state of the carry in \( C_i \) signal \( (a_i b_i b_i = 0101 \Rightarrow C_{i+1} \overline{C}_i + 1 = 01) \). Therefore, the differential full adder of the Fig. 7 is self-testing with respect to the entire set of faults.

From the above arguments, we conclude that the proposed design of SC full adder is TSC for all stuck-at, stuck-on and stuck-open single faults.

Moreover, the proposed differential full adder delivers propagation signals \( \{P_i, \overline{P}_i\} \) by the SUM gate, and generation \( \{G_i, \overline{G}_i\} \) signals by the differential carry gate. These signals are useful for the implementation of fast adders (e.g. carry look-ahead, carry skip, etc.).

V. SIMULATION RESULTS

A. The differential XOR gate

The differential XOR is implemented in full-custom 32nm CMOS technology \cite{36}. SPICE simulations of the circuit extracted from the layout, including parasitic, are used to demonstrate that this gate has an acceptable electrical behaviour.

The layout of the differential XOR gate with the two restoring inverters is as shown in Fig. 9(a). It occupies an area of 0.840\( \times \)0.675 \( \mu m^2 \). Fig. 9(b) gives SPICE simulation of the differential XOR of the Fig. 4 with two inverters to restore outputs. We can see on this simulation that inverters correctly restore outputs. X is the output signal without inverter and XR is the restored signal.

On the other hand, in CMOS circuits there are two sources for power dissipation: static and dynamic. To estimate the power consumption of a circuit equation below is used \cite{37}.

\[
P_D = \sum \left[ C_i V^2_{\text{swing}} \cdot \alpha_i \right] f_{\text{clk}} + i_{\text{leak}} \cdot V_{\text{DD}} + \sum \left[ i_{\text{leak}} \cdot V_{\text{DD}} \right]
\]

In this equation, \( C_i \) is the load capacitance, \( V_{\text{swing}} \) is the voltage swing, \( \alpha_i \) is the probability of a switch, \( f_{\text{clk}} \) is the clock frequency, \( i_{\text{leak}} \) is the short-circuit current, \( i_{\text{leak}} \) is the leakage current and \( V_{\text{DD}} \) is the supply voltage. The main components of the power dissipation are the isc and \( V_{\text{swing}} \) components. The \( i_{\text{leak}} \) component of the equation is very low and is sometimes omitted in literature. The voltage swing of a circuit is the change in voltage that occurs during a transition. It is equal to the voltage difference between logic ‘1’ and logic ‘0’. When the signal transmission is perfect, the logic 1 is equal to \( V_{\text{DD}} \) and the logic 0 is equal to \( V_{\text{SS}} \). The voltage swing is therefore equal to the supply voltage and so, a reduction in supply voltage results in lower power dissipation. The voltage swing is also reduced when the signals are not fully transmitted. A circuit that has a less driving capability often dissipates less power. However, to do useful circuits in this style, we need at least inverters or buffers.

Short-circuit current \( i_{\text{leak}} \) is established by a direct path between VDD and VSS. The repeated presence of such connection causes higher power consumption. This part of the power is less when direct paths from VDD to VSS are limited or become non-existent when either (or both) VDD or VSS is not present, as in the case of the differential XOR proposed in this paper.

On the other hand, the transistors P1 and N1 (respectively P2 and N2) of the differential XOR gate do not conduct
simultaneously, since they have the same signal on theirs gates (respectively \( b \) and \( \overline{b} \)).

**B. The differential full adder**

In order to evaluate the usefulness of the differential XOR and to demonstrate that this is an acceptable design style, we analyze in the following, the performance of the differential full adder shown in Fig. 7 in terms of power consumption and propagation delay. The differential full adder of the Fig. 7 is implemented in full-custom 32nm CMOS technology at 0.8V power supply. All transistors lengths are at the minimum size (\( LN=LP=2λ=30nm \)). The NMOS transistors are at the minimum width size (\( WN=60nm \)). For the PMOS transistors we take \( WP=2.5WN=150nm \). We apply these sizes for the two XOR gates of the differential full adder of the Fig. 7. However, the output inverters are upsized in order to restore efficiently the degraded output signals (\( LN=LP=30nm \) and \( WN=120nm \), \( WP=240nm \)). Fig. 10(a) gives the layout of the self-checking full adder. It occupies an area of 2,730×1,770 µm².

The electrical circuit of the adder was extracted from the layout, and simulated with SPICE. Fig. 10(b) gives simulation result of this circuit. Simulations are performed at varying frequencies to take into account the fact that different applications work at different frequencies. The same applies to capacitive loading conditions.

For completeness, we implement a conventional (not self-checking) CMOS 1-bit full adder in 32nm CMOS technology and simulate it in same conditions.

Fig. 11(a) gives the average power dissipation of the proposed scheme and the standard full adder for different capacitive loading at the output node \( S_i \). The measurements of this figure are carried out on the layout of the differential full adder of the Fig. 7 and the standard full adder.

The average power dissipation is obtained by simulating circuits to compare (SC and conventional adder) for the same number of cycles and the same inputs setting. The power consumption we get by this way corresponds to a specific working frequency. Power consumption of CMOS circuit is directly proportional to the working frequency. Both designs are simulated for the same time simulation (from 0 to 80ns, which corresponds practically, to a simulation for 10 cycles) and with the same inputs setting (the input signals frequencies are kept the same for both circuits).

As expected, the average power dissipation is lower for the proposed self-checking adder than the conventional CMOS adder since it was partially implemented in CMOS pass transistor logic.

We analyze also, the response time of the circuit in term of rise and fall delays at the output node \( S_i \). The Fig. 11(b) gives rise and fall delays vs. load capacitance.

The formal proof of TSC propriety is done in the section 3. However, in order to verify the self-checking proprieties for realistic circuit defects, we simulate the differential full adder in the presence of faults. Some faults are voluntary injected on the physical layout of the circuit. As it proved in section 4, any single fault produces an invalid code word on the output or/and on the carry out, and will be therefore detected.

Fig. 12 simulates the circuit with \( N_2 \) of the XOR1 opened. Fig. 13 shows simulation with \( P_1 \) of the XOR1 opened. In both cases, the fault either does not produce an error on the outputs, or generates non code word and therefore detected by the double-rail checker. Similar results are observed when the XOR2 is faulty with \( N_4 \) or \( P_4 \) opened. Fig. 14 and Fig. 15 simulate the differential full adder with one shorted transistor. These faults are detected by producing errors on outputs.
or/and carries signals.

Fig. 12. Simulation with the fault: N2 (XOR1) stuck-open

Fig. 13. Simulation with the fault: P1 (XOR1) stuck-open

Fig. 14. Simulation with the fault: P1 (XOR1) stuck-on

Fig. 15. Simulation with the fault: P2 (XOR2) stuck-on

The above simulations show that under single fault, the produced erroneous outputs do not belong to the output code. 

($S_i = S_i' :$ error detected by the double-rail checker) or/and on the carry out ($C_{i+1} = C_{i+1}' :$ error propagated to the next stage).

We note also, that transistors stuck-on faults involve a large difference between normal operating current and current under faulted condition. These faults can be therefore, detected by an additional $I_{DDQ}$ testing. Fig. 16 simulates differential full adder and gives $I_{DD}$ current in the cases of fault free circuit and under transistor short fault.

These simulations are performed with the input “a” set to logical “0” ($V_{SS}$) and “$\overline{a}$” set to logical “1” ($V_{DD}$) in the differential full adder of the Fig. 10(a). Under these conditions, and for some specific inputs combinations (b and $C_{in}$), there will exist a direct conducting or partially conducting path in the circuit between $V_{DD}$ and $V_{SS}$. Thus, an abnormal $I_{DD}$ increasing can be observed in SPICE simulations.

Fig. 16 illustrates $I_{DD}$ current for transistor N2 (XOR1) stuck-on fault and for transistor P4 (XOR2) stuck-on fault.

We obtain similar results for all transistors short faults in the two differential XOR gates of the differential full adder layout of the Fig. 10(a).

VI. CONCLUSION

In this paper, we proposed new 4-transistor differential XOR gate. This gate is fault secure and self-testing for all stuck-at, stuck-on and stuck-open faults. The proposed differential XOR gate had been designed and analyzed using 32nm CMOS technology.

In order to evaluate the usefulness of the proposed scheme, a self-checking full adder was implemented, simulated and analyzed. This scheme combine two CMOS styles: pass transistor CMOS technology to perform the sum function and static CMOS technology for the carry gate to avoid propagation problems. After that, we made the proof that the proposed design is TSC for the entire set of faults. In addition, this scheme costs only 28 transistors and for the known self-checking schemes, it requires the lowest hardware overhead. The proposed differential full adder involves only 17% area overhead of a traditional adder without error detection.

Then, the layout of the proposed full adder had been implemented and simulated. The proposed full adder can operate at low voltages, yet giving quite a good speed.

Finally, the differential full adder was simulated with voluntary injected faults to verify the self-checking proprieties, and show the effectiveness of the proposed design for realistic circuit defects. Simulation results were
widely in accordance with theoretical study.

As a future work, we can plan the employment of the analyses of the section 3 in a CAD tool for automatically analyzing TSC property for a CMOS circuit according to a class of fault and a self-checking technique. Moreover, the optimized differential XOR gate and full adder can be used to implement SC adder data path such as ripple carry adder. It can be, as well, adopted to implement faster SC adders (e.g. carry look-ahead, carry select, carry skip, etc.). The optimized differential XOR gate and full adder can also, be used to implement others SC arithmetic operators such as multipliers and dividers.

REFERENCES


Belgacem Hamdi is with the Electronic & microelectronics’s LAB, Monastir, Tunisia. Ph.D. in Microelectronics from INP Grenoble (France). Assistant Professor at ISSAT Sousse, Tunisia. His main areas of interest are: IC design, Test, Built In self Test, DFT tools, self-checking and Fault tolerant systems. E-mail: belgacem.hamdi@issatgb.rnu.tn

Chiraz Kediri is with the Electronic & microelectronics’s LAB, Monastir. Pursuing Ph.D. in Electronic & microelectronics design at Tunis University, Tunisia.

Tourki Rached is the Director of the Electronic & microelectronic’s LAB, Monastir. Professor at FS Monastir university (Tunisia) E-mail: rached.tourki@fsm.rnu.tn