FPGA BASED SELF-HEALING STRATEGY
FOR SYNCHRONOUS SEQUENTIAL CIRCUITS

G. NITHYA¹,*, M. RAMASWAMY²

¹Department of Electronics and Communication Engineering
²Department of Electrical Engineering
¹,²Annamalai University, Annamalai Nagar, Tamil Nadu, 608002, India
*Corresponding Author: nithyag658@gmail.com

Abstract

The paper develops an efficient mechanism with a view to healing bridging faults in synchronous sequential circuits. The scheme inserts faults randomly into the system at the signal levels, encompasses ways to intrigue the state of the signals and carries it with steps to rig out the true values at the primary output lines. The attempts espouse the ability of the methodology to explore the occurrence of a variety of single and multiple bridging faults and arrive at the true output. The approach enables to detect the occurrence of wired-OR and wired AND bridging faults in the combinational part of the serial binary adder as the CUT and heal both the inter and intra-gate faults through the use of the proposed methodology. It allows claiming a lower area overhead and computationally a sharp increase in the fault coverage area over the existing Triple Modular Redundancy (TMR) technique. The Field Programmable Gate Arrays (FPGA) based Spartan architecture operates through Very High-Speed Integrated Circuit Hardware Description Language (VHDL) to synthesize the ModelSim code for validating the simulation exercises. The claim incites to increase the reliability of the synchronous sequential circuits and espouse a place for the use of the strategy in the digital world.

Keywords: Area overhead, Bridging faults, Fault coverage, Self-repair, Synchronous sequential circuits, TMR.
1. Introduction

The digital circuits enjoy an equivocal place in specific niche markets that include space industry, avionics, and nuclear applications. The efforts continue to reduce their area occupancy and lower supply voltage requirements as part of the process of miniaturization. However, the advances bring in their adverse impact on their reliability in terms of operational challenges. Owing to the fact that the feature sizes experience aggressive scaling and find a need to handle powerful electromagnetic interference, wide temperature difference and deep space environments [1-3], it augurs measures for the current hardware systems to be more prone to faults.

The reliability of the digital circuits appears to be the focus of industry and researchers in the sensed fault-tolerant requirements gather critical importance. Besides the circuits remains influenced by external effects in the form of ionising radiation, which can cause errors leading to temporary or even permanent malfunctions.

The threshold model of the circuit enables to distinguish between the acceptable and non-acceptable faults in order that it serves to diagnose the faults [4]. Though there exist methods [5] for the detection of open and short circuit faults, still it requires a sensing element to be connected in parallel with the circuit nodes for fault detection.

Among the many faults that influence the operation of digital circuits, the bridging defects invite considerable attention to ensure a fault tolerant operation for the digital circuits [6-9]. The studies hover over a variety of bridging faults in the form of wired-AND and wired-OR that occur either at the inter-gate or intra-gate level.

Zhong et al. [10] and Zhong et al. [11] presented the effects of static and dynamic bridging faults and showed that a static bridging fault affects the static logic value of a shorted node and a dynamic bridging fault that emerges because of the resistive bridging effect distorts the timing of a shorted node. Hassan et al. [12] commented that the switch box architecture mitigates bridging and short faults caused by cosmic particles in SRAM-based FPGAs has been brought out. The proposed architecture has been evaluated on several MCNC benchmarks using VPR tool and the experimental results reveal it decreases the susceptibility of switch boxes to single event upset by about 18% on an average compared to the traditional methods.

According to Choi and Saluja [13], a test approach that detects inter-port faults modelled by four-way bridging in a dual port memory has been envisaged. Alampally et al. [14] presented a test data volume reduction technique to detect dynamic bridging faults and transition delay faults. Higami et al. [15] commented that the scan-based simulation methods have been applied to diagnosis the bridging faults between a clock line and a gate signal line and the effectiveness evaluated by the experimental results for benchmark circuits. Based on a study by Wu and Lee [16], a procedure has been discussed to distinguish the occurrence of stuck-at faults and static BFs. Roca and Rubio stated [17] that 30% to 50% of the total defects in a circuit relate to its occurrence from unintentional bridges.

The frequent occurrence of BFs demands their detection and diagnosis in order that if real-time field faults repairs remain unattended, the digital systems may lose parts of the functions and construe to provide erroneous outputs [2, 18, 19]. Although the theory relates to being simple, redundant backup technology becomes unsuitable for all components and/or chips [20, 21]. Therefore, its scope of application turns out
to be limited [22, 23], it creates a need to evolve a self-healing strategy with improved reliability and self-repair capacity of digital systems.

The emphasis orients to detect the occurrence of a wired-OR and wired AND bridging fault in the combinational part of the CUT and ascribe a self-healing procedure for arriving at the correct primary output. It develops a comprehensive methodology to address both single and multiple faults in synchronous sequential circuits. The mechanism owes to examine the healing process both at the inter and intra-gate levels for illustrating its unique attributes. The exercise involves evaluating its performance through Modelsim based simulation and engaging the Spartan architecture for synthesizing the codes.

2. Circuit Under Test

A serial binary adder forms the CUT that can perform serial addition, dealing with a pair of bits in one clock cycle. The process starts by adding the zeroth bit and in the next clock cycle, it adds the first bit, including a possible carry from the bit-position 0 and continues in this fashion. Figure 1 shows a block diagram where it includes three shift registers, used to hold $x_1$ and $x_2$ and the sum $z_1$ as the computation proceeds. Since the input shift register be-hive parallel-load capability, the additional task begins by loading the values of $x_1$ and $x_2$ into the registers. Thus, in each clock cycle, it adds a pair of bits and at the end of the cycle, the resulting sum bit shifts into the sum $z_1$ register.

![Fig. 1. Block diagram of CUT.](image)

The design of CUT based on the Mealy model includes two states denoted by $G$ and $H$ where the carry in values be 0 and 1 respectively. Figure 2 shows the state diagram in which the output value $z_1$ depends on both the state and the present value of the inputs $x_1$ and $x_2$. Each transition follows the labelling using the notation $x_1 x_2 / z_1$, which indicates the value of $z_1$ for a given valuation $x_1 x_2$. In state $G$, the input valuation 00 produces $z_1 = 0$, and the CUT remains in the same state. Similarly, for the input combination 01 and 10, the output becomes $z_1 = 1$, and the CUT remains in $G$. However, for 11, the output $z_1 = 0$ and the CUT moves to the state $H$. In the state $H$, the combination 01 and 10 causes $z_1 = 0$, while 11 causes $z_1 = 1$. However, when the combination 00 occurs, the output of 1 occurs and a change into state $G$ takes place. Equations (1) and (2) are the next state and output equations of the serial binary adder and the corresponding state stable reflects in Table 1.

\[
C_{\text{out}} = (x_1 C_{\text{in}}) + (x_1 x_2) + (x_2 C_{\text{in}})
\]

(1)
\[ Z_1(\text{sum}) = x_1 \oplus x_2 \oplus C_{in} \] (2)

Table 1. State table for the serial binary adder.

<table>
<thead>
<tr>
<th>Present state ( x_1x_2 = 00 )</th>
<th>Next state ( z_I )</th>
<th>Output ( z_I )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( G )</td>
<td>( 1 )</td>
<td>( 1 )</td>
</tr>
<tr>
<td>( H )</td>
<td>( 0 )</td>
<td>( 0 )</td>
</tr>
</tbody>
</table>

Fig. 2. State diagram for the serial binary adder.

3. Fault Modelling

The procedure requires a model for investigating the presence of BFs in order to arrive at measures for correcting the output of the synchronous sequential circuit. As explained by Favalli and Dalpasso [24], Pomeranz [25], Benabboud et al. [26], Favalli and Dalpasso [27], Pomeranz and Reddy [28], Yang et al. [29] and Nakura et al. [30], though there exist few combinational test generators to explicitly address BFs, still the study prefers to use the BF model and examine the BFs between two or three lines only in accordance the assumption that the occurrence of more than three lines bridged together may be fairly unlikely, it follows that zero-dominant logic often gives rise to BFs behaving as a wired-AND, and one-dominant logic wired-OR.

Figures 3 and 4 show the wired-AND BF and wired-OR BF in logic gates used in the CUT and the corresponding equivalent faulty circuit respectively.

![Figure 3](image)

Fig. 3. Wired-AND BF in logic gates and equivalent faulty circuit.

![Figure 4](image)

Fig. 4. Wired-OR BF in logic gates and equivalent faulty circuit.
4. Fault Injection Technique

The philosophy revolves around the need for a fault injection procedure in the sense it becomes necessary to observe the performance of a system in the presence of faults. Among the many methods, the simulation-based fault injection enjoys being the preferred choice and facilitates the introduction of faults in the simulated model of the system based on predetermined distribution. The focus relies on the VHDL code modification, wherein the faults can be injected by the addition of dedicated fault injection components called saboteurs. The saboteur usually remains inactive under normal system operation and augurs to change the value or the timing characteristics of one or more signals when active. The saboteur in Fig. 5 allows injecting the faults by forcing changes to the output signal of the component at the schematic editor level directly into the VHDL source code before it reaches the rest of the circuit. The benefits of the fault injection techniques include inherent hierarchical abstraction capability and ability to describe both the structure and the behaviour of a system through the uniquely constructed syntactical format.

![Fig. 5. Series saboteurs fault injection method.](image)

5. Design Methodology

The detection and healing of faults concurrently with system operation envisage being the most desirable forms of fault tolerance in light of the fact that it does not degrade in the system performance. However, the concurrent mechanisms usually imply a large overhead in terms of fault detecting and masking operations.

Das et al. [31] reported that the issues relating to the BF detection in cluster-based FPGA using Muller C element owes to detect the stuck-at and BF by a single test configuration and Tsai et al. [32] describes the structural reduction techniques for logic-chain BF diagnosis. Biswal and Biswas [33] highlighted that an online testing technique of digital circuits for AND – OR BFs that most of the works on OLT of BFs cover only non-feedback faults because the feedback faults cause oscillations and become difficult to be detected on-line. According to Krstic et al. [34], two enhanced architectures described fault detection and correction in combinational and sequential circuits, called EDPEC and FEDC. The proposed method works on fault detection and self-healing logic in the combinational circuit part combined with memory elements implemented using a fault detection unit and inverter as a healer.

5.1. Hardware redundancy-based methods

Hardware redundancy also referred to as masking redundancy uses more physical copies of a hardware component to mask fault. Triple modular redundancy and N-modular redundancy are the two major fault masking techniques to prevent generation of erroneous results.
5.1.1. Triple modular redundancy

The most general hardware masking technique is TMR [35]. The concept was originally suggested by Von Neumann. The concept is illustrated in Fig. 6, where the boxes labelled ‘M’ are identical modules that feed a voting element (called a majority organ by Von Neumann). The voting element accepts the outputs from the three sources and delivers the majority vote as output.

![Fig. 6. Triple modular redundancy.](image)

5.1.2. N-modular redundancy

The concept of TMR could be expanded to include any number of redundant modules to produce an NMR (N-modular redundancy) [36] system and it is illustrated in Fig. 7. An NMR system can be tolerated up to ‘n’ module failure, where $n = (N-1)/2$. In general, in an NMR system ‘N’ is considered to be an odd number, however, it can also be even.

![Fig. 7. N-modular redundancy.](image)

5.2. Proposed self-healing methodology

The architecture primarily consists of the CUT, a fault detection unit, a healer and the fault-free circuit, which acts as the source of reference inputs and the fault injection module introduces fault patterns at the interconnect levels of the system. The theory assumes that the fault-free circuit does not provide faulty outputs owing to information redundancy and the redundant code enables the fault-free circuit to choose the correct output for a given combination of inputs. Figure 8 illustrates the proposed self-healing architecture wherein the central theme of the scheme assures to negate the effect of single and multiple BFs of digital circuits and bring out the expected behaviour.

In this figure $x_1, x_2, \ldots, x_n$ are the “$n$” primary inputs of the CUT and the output predictor (fault-free circuit) and $z_1, z_2, \ldots, z_m$ and $z'_1, z'_2, \ldots, z'_m$ are the outputs of the CUT and the fault free circuit respectively. The main idea underscores to retain the primary outputs at the desired logic state even in the presence of error. It involves a fault detection unit with as many EX-OR gates as the number of primary
outputs of the circuit, which localizes the origin of the fault through the logic states of the EX-OR gates and attempts to correct the fault by toggling the logic state of the faulty interconnect line using NOT gates.

![Proposed self-healing architecture](image)

**Fig. 8. Proposed self-healing architecture.**

**Algorithm**

The algorithm enumerates the steps involved in the process of building the self-healing design being able to come up with fault-free output in the occurrence of BFs. It owes to derive two expressions, one of which, indicates the normal fault-free behaviour of the circuit and the other represents the logical behaviour under an assumed OR-bridging or AND-bridging fault condition. The two expressions read as exclusive ORed; if the result falls as 1, it indicates the presence of a fault and vice versa. The scheme incites to tolerate the faults through a healing unit, which senses the logic state of the primary output and toggles it arrive at the correct output.

**Case I:** \[ (z_1, z_2, \ldots, z_m) \neq (z'_1, z'_2, \ldots, z'_m) = 0; \text{ No fault} \]

then, \((z_1, z_2, \ldots, z_m) = \text{Fault free output}\)

**Case II:** \[ (z_1, z_2, \ldots, z_m) \neq (z'_1, z'_2, \ldots, z'_m) = 1; \text{ Fault detected} \]

then, \((z_1, z_2, \ldots, z_m) = \text{Fault free output}\)

**6. Results and Discussion**

**6.1. Simulation**

The single fault in CUT produces either a single error or a sequence of architecture-dependent burst errors that can solely change the output of a block or a sub-block in each part of the CUT. The injection of faults in any signal line can be either OR-bridging or AND-bridging. The process involves the injection of single BF or multiple BFs simultaneously in random locations during one cycle, with each simulation iteration being run for five cycles on the assumption that it starts from all 0 states.
6.2. Simulation results of triple modular redundancy

The results in Figs. 9 and 10 explain the ability of the CUT to remain in the fault tolerant state on the occurrence of wired-AND and wired-OR BFs respectively at the 0th ns in any one module, through the use of TMR technique. However, the timing diagram in Fig. 11 reveals that it results in the faulty output in the event of the occurrence of the faults at two inputs of the voter.

![Fig. 9. Fault-free output of the TMR based CUT when wired-AND BF at one module.](image9)

![Fig. 10. Fault-free output of the TMR based CUT when wired-OR BF at one module.](image10)

![Fig. 11. Faulty output of the TMR based CUT when wired-AND and wired-OR BFs at two modules.](image11)

6.3. Simulation results of proposed self-healing architecture

A full adder with five logic gates and a D-latch constitute the CUT as seen from Fig. 12. The first four lines of all the timing diagrams relating to the CLK signal
along with the RST signal and inputs of the serial binary adder. It takes five clock cycles to obtain the final fault-free output as observed from the green lines.

The exercise attempts to study both inter and intra-gate level faults to result in single and multiple wired-AND or OR wired BF{s} under four cases. With the introduction of the BF{s}, the states of the CUT become incorrect and the faulty state equations are seen in Eqs. (3) to (18) for the four cases respectively and the corresponding state tables in Tables 2 to 9.

6.3.1. Single wired-AND or wired-OR BF at inter-gate level

It considers a single wired-AND or wired-OR BF at the inter-gate level between points ‘1’ and ‘3’ as seen in Fig. 12, brings in influence with the circuit elements $G_i$, $G_2$, $G_3$, $G_4$, $G_5$, and D flip-flop and results in the occurrence of a fault at the primary output. Figures 13 and 14 show the timing diagrams for the faulted state and the recovery of the fault after the fifth clock cycle for the two cases respectively.

**Fig. 12.** Serial binary adder with nodes for introduction of faults.

**Fig. 13.** Self-healed fault free output when single wired-AND BF at points 1 and 3.

**Fig. 14.** Self-healed fault-free output when single wired-OR BF at points 1 and 3.
Single wired-AND BF

The next state and output equation of CUT suffered from single wired-AND BF at inter-gate level is seen in Eqs. (3) and (4) respectively.

\[
C_{out} = (x_1c_{in}) \cdot (x_1c_{in} \oplus x_2) + (x_1c_{in}) x_2
\]  

(3)

\[
Z_1(sum) = x_1c_{in} \oplus (x_1c_{in} \oplus x_2)
\]  

(4)

Table 2. State table of CUT when single wired-AND BF at inter-gate level.

<table>
<thead>
<tr>
<th>Present state</th>
<th>Next state</th>
<th>Output z1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>x_1, x_2 = 00 01 10 11</td>
<td>00 01 10 11</td>
</tr>
<tr>
<td>G</td>
<td>G</td>
<td>G</td>
</tr>
<tr>
<td>H</td>
<td>G</td>
<td>G</td>
</tr>
</tbody>
</table>

Single wired-OR BF

The occurrence of single wired-OR BF at the inter-gate level of CUT leads to faulty next state and output equations in accordance with the Eqs. (5) and (6).

\[
C_{out} = (x_1 + c_{in}) \cdot ((x_1 + c_{in}) \oplus x_2) + (x_1 + c_{in}) x_2
\]  

(5)

\[
Z_1(sum) = (x_1 + c_{in}) \oplus ((x_1 + c_{in}) \oplus x_2)
\]  

(6)

Table 3. State table of CUT when single wired-OR BF at inter-gate level.

<table>
<thead>
<tr>
<th>Present state</th>
<th>Next state</th>
<th>Output z1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>x_1, x_2 = 00 01 10 11</td>
<td>00 01 10 11</td>
</tr>
<tr>
<td>G</td>
<td>G</td>
<td>G</td>
</tr>
<tr>
<td>H</td>
<td>H</td>
<td>H</td>
</tr>
</tbody>
</table>

6.3.2. Multiple wired-AND or wired-OR BF at inter-gate level

The introduction of the multiple wired-AND or wired-OR BF at the inter-gate level between points 1, 2 and 3 as seen in Fig. 12, depicts the differences in the timing diagrams as noticed from Figs. 15 and 16. The faults at the inputs of G_1, G_2, G_3 and G_4 affects their outputs and end up with a fault at the primary output. In any case, the healing touch offers the recovery of the fault-free output in the same timing diagrams after the fifth clock cycle.

Fig. 15. Self-healed fault free output when multiple wired-AND BF at points 1 to 3.
Fig. 16. Self-healed fault-free output when multiple wired-OR BF at points 1, 2 and 3.

Multiple wired-AND BF

The next state and output equation of the CUT in Eqs. (7) and (8) reflect the introduction of multiple wired-AND BFs at the signal level.

\[ C_{\text{out}} = (x_1x_2C_{\text{in}} \oplus x_1x_2C_{\text{in}}) \cdot (x_1x_2C_{\text{in}}) + (x_1x_2C_{\text{in}}) \cdot (x_1x_2C_{\text{in}}) \quad (7) \]

\[ Z_1(\text{sum}) = (x_1x_2C_{\text{in}} \oplus x_1x_2C_{\text{in}}) \oplus x_1x_2C_{\text{in}} \quad (8) \]

Table 4. State table of CUT when multiple wired-AND BFs at inter-gate level.

<table>
<thead>
<tr>
<th>Present state</th>
<th>Next state</th>
<th>Output ( z_I )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x_1x_2 = 00 )</td>
<td>( 00 ) ( 01 ) ( 10 ) ( 11 )</td>
<td>( 0 ) ( 0 ) ( 0 ) ( 0 )</td>
</tr>
<tr>
<td>( H )</td>
<td>( G ) ( G ) ( G ) ( G )</td>
<td>( 0 ) ( 0 ) ( 0 ) ( 1 )</td>
</tr>
</tbody>
</table>

Multiple wired-OR BF

Equations (9) and (10) correspond to the next stage and output expressions for the multiple wired-OR BFs in the CUT from Fig. 12.

\[ C_{\text{out}} = \left( (x_1 + x_2 + C_{\text{in}}) \oplus (x_1 + x_2 + C_{\text{in}}) \right) \cdot (x_1 + x_2 + C_{\text{in}}) + (x_1 + x_2 + C_{\text{in}}) \cdot (x_1 + x_2 + C_{\text{in}}) \quad (9) \]

\[ Z_1(\text{sum}) = \left( (x_1 + x_2 + C_{\text{in}}) \oplus (x_1 + x_2 + C_{\text{in}}) \right) \oplus (x_1 + x_2 + C_{\text{in}}) \quad (10) \]

Table 5. State table of CUT when multiple wired-OR BFs at inter-gate level.

<table>
<thead>
<tr>
<th>Present state</th>
<th>Next state</th>
<th>Output ( z_I )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x_1x_2 = 00 )</td>
<td>( 00 ) ( 01 ) ( 10 ) ( 11 )</td>
<td>( 0 ) ( 1 ) ( 1 ) ( 1 )</td>
</tr>
<tr>
<td>( H )</td>
<td>( H ) ( H ) ( H ) ( H )</td>
<td>( 1 ) ( 1 ) ( 1 ) ( 1 )</td>
</tr>
</tbody>
</table>

6.3.3. Single wired-AND or wired-OR BF at intra-gate level

The single wired-AND or wired-OR BF at the intra-gate level in the point 4 and 5 seen in Fig. 12 and explain the arrival of the incorrect SUM output travel through the gate \( G_2 \), \( G_3 \), \( G_4 \) and D-flip flop respectively. However, the scheme ensures the recovery of the fault-free output as seen in the timing diagrams (Figs. 17 and 18) after the fifth clock cycle.
Single wired-AND BF

The next state and the output expression in Eqs. (11) and (12) relate to the effect of single wired-AND BF at the intra-gate level of the CUT in Fig. 12.

\[ C_{\text{out}} = (x_1 \oplus x_2) C_{\text{in}} \cdot (x_1 \oplus x_2) C_{\text{in}} + x_1 x_2 \]  
\[ Z_1(\text{sum}) = ((x_1 \oplus x_2) C_{\text{in}}) \oplus ((x_1 \oplus x_2) C_{\text{in}}) \quad (12) \]

<table>
<thead>
<tr>
<th>Present state</th>
<th>( x_1 x_2 = 00 )</th>
<th>( 01 )</th>
<th>( 10 )</th>
<th>( 11 )</th>
<th>( 00 )</th>
<th>( 01 )</th>
<th>( 10 )</th>
<th>( 11 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( G )</td>
<td>( G )</td>
<td>( G )</td>
<td>( G )</td>
<td>( H )</td>
<td>( 0 )</td>
<td>( 0 )</td>
<td>( 0 )</td>
<td>( 0 )</td>
</tr>
<tr>
<td>( H )</td>
<td>( G )</td>
<td>( H )</td>
<td>( H )</td>
<td>( H )</td>
<td>( 0 )</td>
<td>( 0 )</td>
<td>( 0 )</td>
<td>( 0 )</td>
</tr>
</tbody>
</table>

Table 6. State table of CUT when single wired-AND BF at intra-gate level.

Single wired-OR BF

The resulting state equations of CUT (next state and output equations) due to the introduction of single wired-OR BF at the points 4 and 5 follows from Eqs. (13) and (14).

\[ C_{\text{out}} = \left( (x_1 \oplus x_2) + C_{\text{in}} \right) + \left( (x_1 \oplus x_2) + C_{\text{in}} \right) + x_1 x_2 \]  
\[ Z_1(\text{sum}) = \left( (x_1 \oplus x_2) + C_{\text{in}} \right) \oplus \left( (x_1 \oplus x_2) + C_{\text{in}} \right) \quad (14) \]
Table 7. State table of CUT when single wired-OR BF at intra-gate level.

<table>
<thead>
<tr>
<th>Present state</th>
<th>Next state</th>
<th>Output $z_t$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1x_2 = 00$</td>
<td>$G$</td>
<td>$G$</td>
</tr>
<tr>
<td>$G$</td>
<td>$G$</td>
<td>$H$</td>
</tr>
<tr>
<td>$H$</td>
<td>$H$</td>
<td>$H$</td>
</tr>
</tbody>
</table>

6.3.4. Multiple wired-AND or wired-OR BFs at intra-gate level

The presence of multiple wired-AND or wired-OR BFs between points 4 and 5 and points 6 and 7 at the intra-gate level can be seen from the differences in the timing diagram in Figs. 19 and 20. The methodology offers to heal and correct them after the fifth clock cycle as observed from the same figures.

Fig. 19. Self-healed fault-free output when multiple wired-AND BFs at points 4 and 5 & 6 and 7.

Fig. 20. Self-healed fault-free output when multiple wired-OR BFs at points 4 to 7.

Multiple wired-AND BFs

Equations (15) and (16) correspond to the next state and the output expressions of the CUT for the multiple wired-AND.

$$ C_{out} = (x_1 \oplus x_2) \cdot (x_1 \oplus x_2) \cdot x_1x_2 $$

$$ + ((x_1 \oplus x_2) \cdot (x_1 \oplus x_2) \cdot x_1x_2 $$

$$ Z_1(sum) = (x_1 \oplus x_2) \oplus (x_1 \oplus x_2) \cdot C_{in} $$

(15)

(16)

Table 8. State table of CUT when multiple wired-AND BFs at intra-gate level.

<table>
<thead>
<tr>
<th>Present state</th>
<th>Next state</th>
<th>Output $z_t$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1x_2 = 00$</td>
<td>$G$</td>
<td>$G$</td>
</tr>
<tr>
<td>$G$</td>
<td>$G$</td>
<td>$G$</td>
</tr>
<tr>
<td>$G$</td>
<td>$G$</td>
<td>$G$</td>
</tr>
<tr>
<td>$G$</td>
<td>$G$</td>
<td>$G$</td>
</tr>
<tr>
<td>$H$</td>
<td>$G$</td>
<td>$G$</td>
</tr>
<tr>
<td>$H$</td>
<td>$G$</td>
<td>$G$</td>
</tr>
<tr>
<td>$H$</td>
<td>$H$</td>
<td>$H$</td>
</tr>
<tr>
<td>$H$</td>
<td>$H$</td>
<td>$H$</td>
</tr>
</tbody>
</table>

Multiple wired-OR BFs

The next state and output expressions in Eqs. (17) and (18) relate to the multiple wired-OR BFs introduced at points 4 to 7 in Fig. 12.

\[ C_{\text{out}} = \left( (x_1 \oplus x_2) + C_{\text{in}} \right) \cdot \left( (x_1 \oplus x_2) + C_{\text{in}} \right) + x_1 x_2 \]

\[ + \left( (x_1 \oplus x_2) + C_{\text{in}} \right) \cdot \left( (x_1 \oplus x_2) + C_{\text{in}} \right) + x_1 x_2 \] (17)

\[ Z_1(\text{sum}) = \left( (x_1 \oplus x_2) + C_{\text{in}} \right) \oplus \left( (x_1 \oplus x_2) + C_{\text{in}} \right) \] (18)

Table 9. State table of CUT when multiple wired-OR BFs at intra-gate level.

<table>
<thead>
<tr>
<th>Present state</th>
<th>Next state</th>
<th>Output z1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>x_1 \times x_2 = 00 01 10 11</td>
<td>00 01 10 11</td>
</tr>
<tr>
<td>(G)</td>
<td>(G) (H) (H) (H)</td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>(H)</td>
<td>(H) (H) (H) (H)</td>
<td>0 0 0 0</td>
</tr>
</tbody>
</table>

The entries in Table 10 show the value of faulty and fault-free outputs for single and multiple wired-AND/OR inter and intra gate faults introduced at the chosen points.

Table 10. Fault tolerant characteristics of serial binary adder.

<table>
<thead>
<tr>
<th>Fault location</th>
<th>Gate level</th>
<th>Input (x_1)</th>
<th>Input (x_2)</th>
<th>Fault pattern</th>
<th>Faulty output</th>
<th>Fault free output</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 and 3</td>
<td>Inter-gate</td>
<td>0101</td>
<td>0111</td>
<td>Single wired-AND</td>
<td>0111</td>
<td>1100</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0101</td>
<td>0111</td>
<td>Single wired-OR</td>
<td>0001</td>
<td>1100</td>
</tr>
<tr>
<td>1 to 3</td>
<td></td>
<td>0101</td>
<td>0111</td>
<td>Multiple wired-AND</td>
<td>0000</td>
<td>1100</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0101</td>
<td>0111</td>
<td>Multiple wired-OR</td>
<td>1111</td>
<td>1100</td>
</tr>
<tr>
<td>4 and 5</td>
<td>Intra-gate</td>
<td>0101</td>
<td>0111</td>
<td>Single wired-AND</td>
<td>0000</td>
<td>1100</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0101</td>
<td>0111</td>
<td>Single wired-OR</td>
<td>0000</td>
<td>1100</td>
</tr>
<tr>
<td>4 and 5 &amp; 6 and 7</td>
<td></td>
<td>0101</td>
<td>0111</td>
<td>Multiple wired-AND</td>
<td>0000</td>
<td>1100</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0101</td>
<td>0111</td>
<td>Multiple wired-OR</td>
<td>0000</td>
<td>1100</td>
</tr>
</tbody>
</table>

7. Experimental Results

The efforts relate to evaluating the self-healing methodology through the indices that include the fault coverage (FC) and area overhead (AO) obtained based on the analytical procedure outlined using expressions (19) and (20). Figure 21 brings out the merits of the self-healing strategy over the TMR scheme in terms of the fault coverage and the extent of resources utilized for real-time implementation, which in turn, measures the area overhead.

The TMR requires an extremely high area overhead, which rises to as high as 360% since it needs two exact replicas of the CUT together with the voter circuit whereas the proposed scheme needs relatively very few redundant components and as a result reduces the area overhead to as low as 180%. The method under study offers to heal single as well as multiple BFs under any circumstances owing to its built-in self-healing property.
\[ FC = \frac{\text{Number of detected fault}}{\text{Total number of fault}} \times 100 \]  
\[ AO = \frac{\text{Number of redundant gates}}{\text{Number of actual gates to implement the logic}} \times 100 \]

**Fig. 21. Comparative chart.**

**Hardware implementation**

The RTL schematic in Figs. 22 and 23 explain the framework of the self-healing architecture of serial binary adder obtained for both the existing TMR and the proposed healing approach and gathers the use of the commercial XC3S500E FPGA evaluation board for introducing the BFs through HDL codes.

Table 11 compares the logic utilization between the proposed scheme and the existing TMR technique in the FPGA architecture and the same table registers the lower number of slices and LUTs utilized for the proposed scheme over the TMR-approach.

The Field Programmable Gate Array (FPGA) belongs to a class of reconfigurable computing systems and enable a platform to realize the functions of digital devices [37]. The photograph in Fig. 24 relates to the experimental setup and functions using VHDL and the methodology allow synthesizing the codes to validate the simulated performance.

It echoes the use of a Core 2 Duo CPU at 3.0 GHz and 4GB RAM for enabling the FPGA to operate using VHDL and synthesize the codes that realize the fault healing mechanism. The procedure depends on the glow of LEDs to epitomize the correctness of the output of the serial binary adder through the healing scheme.

**Table 11. Summary of logic utilization.**

<table>
<thead>
<tr>
<th>Logic utilization</th>
<th>Available</th>
<th>Proposed scheme used</th>
<th>TMR used</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of slice FFs</td>
<td>9312</td>
<td>13</td>
<td>35</td>
</tr>
<tr>
<td>No. of 4 input LUTs</td>
<td>9312</td>
<td>18</td>
<td>45</td>
</tr>
<tr>
<td>No. of occupied slices</td>
<td>4656</td>
<td>10</td>
<td>28</td>
</tr>
</tbody>
</table>
Fig. 22. RTL schematic of TMR.

Fig. 23. RTL schematic of proposed scheme.

Fig. 24. Real time experimental set up.
8. Conclusion

A self-healing strategy has been developed to detect the occurrence of single and multiple bridging faults and retrieve the correct primary outputs in synchronous sequential circuits. It has been oriented to cater the needs of BFs that structurally modify the circuits and appear in wired-AND or wired-OR configurations. The model has been simulated on a Modelsim platform and evaluated under four cases to realize inter and intra-gate level faults. The framework of a Spartan architecture has been programmed using VHDL to synthesize the Modelsim codes and establish the practical viability of the methodology. The results have been compared with that obtained using the traditional TMR method and the benefits projected using performance indices. The fault detection and healing circuits have been laid to provide an inherent mechanism through which, the correctness of the data can be relied upon. The significant contribution has been envisaged to find a place for its use in communication networks to transmit error-free information. The other areas of use have been forayed in mission and safety-critical applications to ensure uninterrupted operation of the systems.

<table>
<thead>
<tr>
<th>Nomenclature</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_{in}$</td>
</tr>
<tr>
<td>$C_{out}$</td>
</tr>
<tr>
<td>$x_1, x_2$</td>
</tr>
<tr>
<td>$z_1$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Abbreviations</th>
</tr>
</thead>
<tbody>
<tr>
<td>AO</td>
</tr>
<tr>
<td>BF</td>
</tr>
<tr>
<td>CLK</td>
</tr>
<tr>
<td>CUT</td>
</tr>
<tr>
<td>EDPEC</td>
</tr>
<tr>
<td>FC</td>
</tr>
<tr>
<td>FEDC</td>
</tr>
<tr>
<td>FPGA</td>
</tr>
<tr>
<td>HDL</td>
</tr>
<tr>
<td>LUTs</td>
</tr>
<tr>
<td>MCNC</td>
</tr>
<tr>
<td>OLT</td>
</tr>
<tr>
<td>RST</td>
</tr>
<tr>
<td>RTL</td>
</tr>
<tr>
<td>SRAM</td>
</tr>
<tr>
<td>TMR</td>
</tr>
<tr>
<td>VHDL</td>
</tr>
<tr>
<td>VHSIC</td>
</tr>
<tr>
<td>VPR</td>
</tr>
</tbody>
</table>
References


Symposium on Design and Diagnostics of Electronic Circuits & Systems. Liberec, Czech Republic, 264-269.


