Low-Cost and Highly Reliable Detector for Transient and Crosstalk Faults Affecting FPGA Interconnects*

M. Omaña **J.M. Cazeaux** **D. Rossi** **C. Metra**

DEIS, University of Bologna, Viale Risorgimento 2, 40136 Bologna, Italy

Abstract

In this paper we present a novel circuit for the on-line detection of transient and crosstalk faults affecting the interconnects of systems implemented using Field Programmable Gate-Arrays (FPGAs). The proposed detector features self-checking ability with respect to faults possibly affecting itself, thus being suitable for systems with high reliability requirements, like those for space applications. Compared to alternate solutions, the proposed circuit requires a significantly lower area overhead, while implying a comparable, or lower, impact on system performance. We have verified our circuit operation and self-checking ability by means of post-layout simulations.

1. Introduction

Field Programmable Gate Arrays (FPGAs) have been widely employed for prototyping [1, 2, 3] and low-volume production, even for space applications [4]. This mainly because of their low cost, easiness of implementation and reconfiguration.

Interconnects constitute the largest fraction of the FPGA area (about the 80%) [5]. With the scaling of technology, they are becoming very likely to be affected by Transient Faults (TFs) (generally due to α-particles or cosmic rays)[6], which may temporarily alter the voltage value they transfer, thus giving rise to glitches. In a synchronous system, if the generated glitches reach the inputs of a functional block (composed by one or more Configurable Logic Blocks, CLBs), when such inputs should be stable due to sampling constraints, they may result in the generation of a soft error, possibly compromising the global system correct operation.

Beside TFs, the continuous scaling of technology is making global interconnects very likely to be affected by Crosstalk Faults (CFs), because of the continuous increase in the interwire parasitic capacitance and the way interconnects are scaled down [7, 8]. In fact, interconnections become ever higher and thinner, and closer to each other, thus making their coupling capacitance increase with respect to the substrate capacitance. This might cause an anomalous increase in signal propagation delay, making the connected flip-flops sampling an incorrect voltage value. Therefore, similarly to TFs, also CFs may make a functional block generate an incorrect data at the output, possibly compromising the global system correct operation.

Of course, making FPGA implemented systems able to test themselves on-line with respect to TFs and CFs, and to employ proper recovery techniques is a possible approach to avoid the, otherwise inevitable, consequent decrease of system's reliability.

The problem of testing FPGAs has been largely addressed in the literature (e.g., [9, 10, 11, 12]). However, so far, the general problem of testing on-line FPGA-implemented systems has been addressed only in [14, 13, 15]. In [13], a synthesis algorithm for the generation of self-checking combinational circuits implemented by means of FPGAs has been proposed. The derived self-checking implementations allow to detect also faults affecting the FPGA interconnects. In [15], an approach for the on-line diagnosis and location of faults affecting FPGA interconnects has been presented. However, these works consider only possible Stuck-At faults (SAs) of the interconnects.

Conversely, a self-checking detector for TFs and CFs affecting the FPGA interconnects of implemented synchronous systems has been proposed in [14]. However, this scheme may imply a non negligible area overhead that, especially for space applications, may constitute a problem.

In this regard, it should be noted that, in the past, electronic circuits operating in systems for space applications have been implemented using rad-hardened components, in order to reduce the likelihood of TFs. However, since such components are very expensive and present a lower performance compared to standard commercial ones, in the last years their application has been reduced in favor of non rad-hardened ones [4], for instance implemented by standard FPGAs.

As for CFs, techniques to reduce their likelihood have been presented also in [16, 7]. However, they do not avoid completely the occurrence of CFs and do not protect the system against possible TFs.

Based on these considerations, in this paper we present a novel circuit for the on-line detection of TFs and CFs affecting the FPGA global interconnects. Compared to the approach in [14], our detecting scheme requires significantly lower area overhead, while implying a similar or lower impact on system's performance and a comparable self-checking ability with respect to possible internal faults.

The rest of this paper is organized as follows. In Section 2, we introduce the proposed self-checking CFs and TFs detector. In Section 3, we show some of the results of the post-layout simulations that we have

* Work partially supported by the Italian Space Agency (ASI).

performed to verify its operation. In Section 4, we analyze its self-checking ability. In Section 5, we evaluate its costs and compare them to those of the previous scheme presented in [14]. Finally, some conclusive remarks are given in Section 6.

2. The Proposed Detector

We consider, as a reference, the case of Xilinx FPGAs. For this kind of FPGA, each CLB is composed of three Look-Up Tables (LUTs). The input LUTs (hereafter referred to as LUT F and LUT G) can implement any Boolean function of four inputs. The cascaded LUT (denoted as LUT H) receives as inputs the output of LUT F and LUT G, plus another independent input, and it can implement any Boolean function of the three input signals [17]. Moreover, each CLB includes several MUXs and two D flip-flops/latches with enable, set and reset signals. However, our detection scheme can be easily employed for any other FPGA device by means of straightforward modifications. Moreover, it can be noticed that our scheme can be applied also to general designs.

In order to detect on-line CFs or TFs affecting the global interconnects of an FPGA implemented synchronous system, we have developed a CFs and TFs Detector (CTD) that is continuously monitoring such interconnects.

It consists of an internal *Detection Cell Array (DCA)* and a *Checker* (*TRC$_n$*), as schematically shown in Fig. 1.

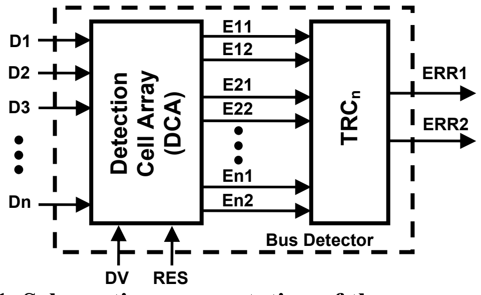

Fig. 1. Schematic representation of the proposed CFs and TFs Detector (CTD).

D1-Dn denote the monitored FPGA interconnects. We suppose that they are given to the inputs of a synchronous block through CLB internal flip-flops. Consequently, they should present a stable logic value at the flip-flops' sampling instants. *DV* (Data Valid) is a signal that we assume equal to 1 in the time intervals during which *D1-Dn* should be stable in the fault-free case. We assume that such a signal is available within the considered system. Otherwise, it could be properly generated on the basis of system timing analysis.

RES is an external signal to be set to the high logic value in order to possibly start again the CTD operation after the detection of a TF or CF. It could also be set by our detector automatically, with a given delay (depending on the possibly adopted recovery technique) with respect to detection accomplishment.

The *2n* outputs of the *DCA, Eij (i=1, ..., n; j=1, 2),* are encoded by means of a two-rail code. In case of a fault affecting the interconnect line *Di*, *Ei1* and *Ei2* will not belong to the two-rail code. This error indication is maintained until reset (i.e., RES=1).

The outputs of the *DCA* are then checked by means of an n-variable two-rail code checker (TRC$_n$ in Fig. 1). As an example, it can be implemented as a tree of 2-variable TRCs, as introduced in [18]. It gives to its output the signal *ERR1* and *ERR2*. In the fault-free case, *ERR1* and *ERR2* are two-rail encoded, while they are not in the case of CFs or TFs affecting the monitored interconnects.

As discussed later on, we designed the *DCA* and *TRC$_n$* in order to guarantee that a non codeword is produced at the output of the CTD, also in case of faults affecting the *DCA* or the *TRC$_n$*.

More in details, the *DCA* consists of *n* basic cells, each connected to a monitored interconnect. Each cell is able to detect on-line undesired transitions (due to TFs or CFs) on the monitored interconnect, occurring in the time intervals during which it should be stable in the fault-free case (i.e., when *DV = 1*). This way our detector can detect CFs and TFs affecting up to all the monitored interconnects (*D1-Dn*).

The internal structure of each basic cell is shown in Fig. 2. *FFDi* denotes a D flip-flop triggered on the rising edge of *DV*. *LDi* denotes a D latch that is transparent when *DV=1*, and that latches the input datum *Di* on the falling edge of *DV*.

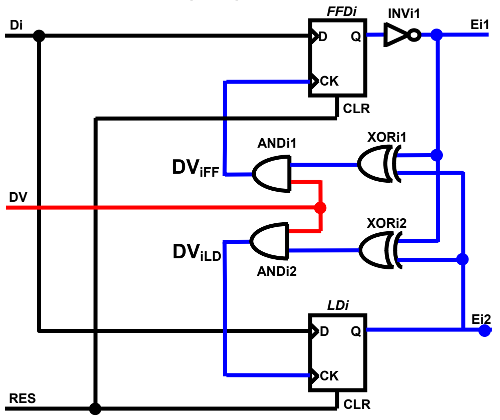

Fig. 2. Internal structure of a basic cell of the Detection Cell Array of our CTD.

In order to guarantee that each cell gives a two-rail codeword on *Ei1* and *Ei2* at the beginning of operation, it must be initialized by applying a logic 1 on the *RES* signal. This produces a logic 0 at the outputs of both *FFDi* and *LDi*, so that *Ei1* = 1 and *Ei2* = 0 for all *i*, independently of the logic value of *DV*. As a consequence, the outputs of both *XORi1* and *XORi2* present a logic 1. This way, the signal *DV* acts as clock signal for both *FFDi* (DV$_{iFF}$ in Fig. 2) and *LDi* (DV$_{iLD}$ in Fig. 2). An analogous procedure has to be carried on when we want to start again the operation of the CTD after the generation of an error message.

During normal operation, the monitored interconnect *Di* is sampled by *FFDi* and its complementary logic value is transferred through *INVi1* to *Ei1* (Fig. 2) at every rising edge of *DV*. Instead, *LDi* (Fig. 2) is transparent when *DV=1*, and transfers to its output *Ei2* the value of the monitored interconnect *Di* at each falling edge of *DV*. This way, if *Di* remains constant when *DV=1*, the outputs *Ei1* and *Ei2* are two-rail encoded and no error indication is generated.

Conversely, if a delayed transition occurs, or a glitch affects the monitored interconnect *Di* when *DV=1*, the latch (which is still transparent) changes the value of *Ei2* to the logic value opposite to that sampled by *FFDi* (that is *Ei1* in Fig. 2). Therefore, in this case, an error indication is generated (i.e., *Ei1* and *Ei2* are no longer two-rail encoded). Moreover, since *Ei1=Ei2*, both *XORi1* and *XORi2* produce a logic 0 at their outputs, thus giving a logic 0 also at the outputs of *ANDi1* and *ANDi2*. Consequently, DV_{iFF} and DV_{iLD} remain at a low logic value independently of the DV signal, thus maintaining the error indication until reset.

As an example, Fig. 3 shows the FPGA implementation of our detector for the case of 2 monitored interconnects. The combinational logic of each basic cell (Fig. 2) is implemented by means of the LUT G and F of a single CLB (LUTs G_{DCA1} and F_{DCA1} for DCA_1, and LUTs G_{DCA2} and F_{DCA2} for DCA_2 in Fig. 3). As for the two memory elements within each CLB, the considered FPGA allows to drive them only with one clock signal. Thus, since the clock signals for *FFDi* and *LDi* of a same basic cell are different (i.e., DV_{iFF} and DV_{iLD} in Fig. 2), one of the memory elements of the basic cell must be placed into an additional CLB. However, this does not imply an increase in the total area required by our detector. In fact, observing the implementation of the TRC_2 in Fig. 3, we can see that it uses only the LUT F and G of a CLB (LUTs G_{TRC2} and F_{TRC2} in Fig. 3). This allows us to map the second memory element of one basic cell into the CLB used by the TRC_2 (Fig. 3).

Similarly for the general case of *n* monitored interconnects. In fact, the considered TRC_n uses *(n-1)* 2-variable TRCs (TRC_2s), structured as a binary tree with $log_2(n)$ levels [18]. Each TRC_2 is mapped on the LUT F and G of the same CLB, leaving unused 2 memory elements. We can employ these unused memory elements to map the second memory element of each *DCA* cell. In particular, we employ *(n-1)* memory elements from the *(n-1)* TRC_2 CLBs and an additional CLB to implement the secondary memory elements of the *n* basic cells of the *DCA*. It is worth noticing that if we had employed Altera FPGAs, since they allow to use different clocks within a single basic logic block, a better optimization in terms of area overhead could have been achieved [19].

3. Verification of our Detector Behavior

We have verified the behavior of our detector by means of post-layout simulations. As an example, we have implemented our scheme by means of the Xilinx ISE tool. In particular, we have considered the case of a Xilinx Spartan XCS30XL FPGA.

Fig. 4(a) and 4(b) show some of the results of the post-layout simulations of a basic cell. We have considered that the rising edge of *DV* is anticipated with respect to the rising edge of the system clock (*CK*) by a time chosen accordingly to the sampling constraints of the blocks connected to the interconnects.

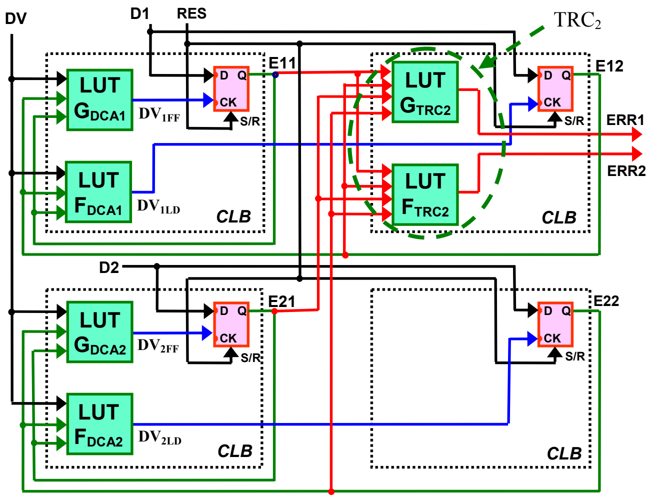

Fig. 3. Schematic of the internal structure of the FPGA implemented detector for the case of 2 monitored interconnects.

Initially, the circuit is reseted by making RES=1. Then at time *t0* RES=0, and the circuit starts its normal operation.

Fig. 4(a) shows the results of a post-layout simulation of a basic cell implemented by means of our proposed circuit (Fig. 2) in the case of a delay (due to a CF) affecting the interconnect *Di* at time *t1* (Fig. 4(a)), when *DV=1*. As can be seen, an error indication is generated on the outputs *Ei1* and *Ei2,* that remains latched until reset at time *t2*. Similarly, in Fig. 4(b), an error indication is produced on *Ei1* and *Ei2* because of a TF affecting the same interconnect *Di* at time *t1*. The error indication remains latched until reset at *t2*.

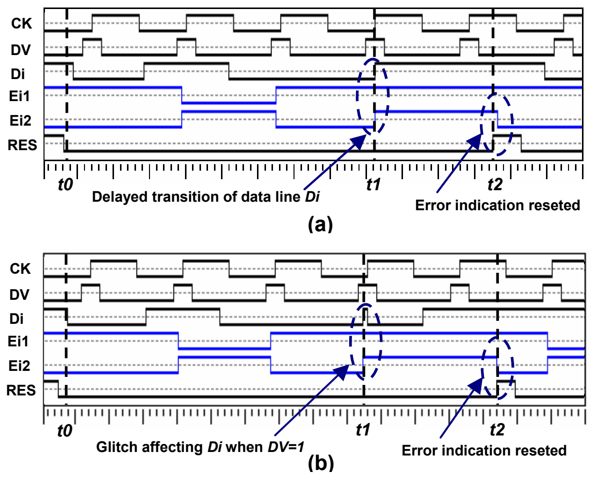

Fig. 4. Post-layout simulation waveforms obtained considering a basic cell in the case of: (a) a delayed transition (due to a CF) of the interconnect *Di* occurring at the instant *t1*; and (b) a glitch affecting the interconnect *Di* at the instant *t1*.

In both figures, we can see that, before the delayed transition or the glitch affecting the monitored interconnect Di, Ei1 and Ei2 present complementary logic values (as stated in Section 3).

4. Self-Checking Ability

Let us discuss the self-checking ability of our CTD with respect to its possible internal faults. As usual for self-checking circuits, we assume that faults occur one at a time and that the time elapsing between two following faults is long enough to allow the application of all possible input code words [21].

We have considered the following realistic set of possible internal faults for FPGA implemented devices [22]: 1) all possible stuck-ats (SAs) affecting the input lines of the *DCA*; 2) all possible SAs affecting the internal lines of the *DCA*; 3) all possible SAs affecting the output lines of the *DCA*; 4) transient faults (TFs) and crosstalk faults (CFs) affecting the input lines of the *DCA*; 5) TFs affecting the internal lines of the DCA; 6) TFs affecting the output lines of the DCA; 7) all possible SAs affecting the input lines of the *TRCn*; 8) all possible SAs affecting the internal lines of the *TRCn*; 9) all possible SAs affecting the output lines of the *TRCn*; 10) TFs affecting the input lines of the *TRCn*; 11) TFs affecting the internal lines of the *TRCn*; 12) TFs affecting the output lines of the *TRCn.*

It should be noted that CFs affecting the CTD internal lines have been not considered, since CFs are not likely for local interconnects. As for the CF between the CTD two outputs, it should be avoided by proper routing of the ERR1 and ERR2 lines.

As for faults possibly affecting signals DV and RES, we have assumed that they are properly checked using specific detectors, for instance of the kind in [23].

We have analyzed the effects of all considered faults by means of post-layout simulations. We have verified that the *Detection Cell Array* of our detector is Totally Self-Checking (TSC) [18] or Strongly Fault Secure (SFS) [21] with respect to all the considered faults, while the *checker* is TSC or Strongly Code Disjoint (SCD) [24].

Faults of kind 1) correspond to SA0 or SA1 affecting the monitored interconnect Di. These faults produce a non code word on Ei1 and Ei2 by design (Section 2), thus an error indication at the output of our CTD. This error indication remains latched until our detector is reseted (RES=1). Therefore, the *DCA* is TSC with respect to faults of kind 1).

Faults of kind 2) correspond to SA0 or SA1 affecting lines DV_{iFF} and DV_{iLD} (Fig. 2).

Consider now SA0 or SA1 affecting DV_{iFF}, that is the clock of FFDi. They prevent FFDi from sampling the signal on the monitored line Di. This way, Ei1 will be always set at a logic 1, due to the initial reset of FFDi (Fig. 2). Therefore, when the latch LDi samples (when DV=1) a logic 1 on Di, the non code word Ei1= Ei2 = 1 is generated and then an error indication is produced by our CTD. This error message remains latched until our detector is reseted. Therefore, the *DCA* is TSC with respect to SA0 or SA1 affecting DV_{iFF}.

As for SA0 affecting DV_{iLD}, it produces a constant 0 on the clock signal of latch LDi, independently of the DV signal. Therefore, LDi, which is transparent when its clock is set at 1, will never sample the signal on its monitored line Di, and Ei2 will be always at 0. Thus, when (at the rising edge of DV) FFDi samples a 1 on the Di, a non code word is generated on Ei1 and Ei2, and then an error indication is provided by our CTD. As in the previous case, this error indication remains latched until our detector is reseted. Therefore, the *DCA* is TSC with respect to SA0 affecting DV_{iLD}.

SA1 faults affecting DV_{iLD} produce a constant 1 on the clock signal of LDi, independently of the DV signal. As a consequence, LDi will be always transparent giving as output the signal on the monitored interconnect Di. Thus, when DV=0 and Di changes, a non codeword will be produced on Ei1 and Ei2. This error indication gives a logic 0 on DV_{iFF}, preventing FFDi from sampling on the following rising edge of DV after the error detection. This way, the error indication is maintained for at least one system's clock period, making the *DCA* fulfill the TSC property with respect to SA1 affecting DV_{iLD}. In order to maintain latched this error indication until reset (if required), an error indicator, for instance of the kind in [25], can be connected at the output of our CTD.

We have verified that the *DCA* of our detector is TSC with respect to faults of kind 2).

Faults of kind 3) correspond to SA0 or SA1 affecting the lines Ei1 and Ei2 of the basic cells (Fig. 2). Since these signals assume complementary logic values in the fault-free case, these faults, when activated, produce a non code word on Ei1 and Ei2, and thus an error indication at the output of our CTD, which remains latched until our detector is reseted. Therefore, the *DCA* is TSC with respect to faults of kind 3).

Faults of kind 4) are TFs and CFs affecting the monitored interconnect Di. They produce an error indication on ERR1 and ERR2 by design (Section 2). Thus, the *DCA* is TSC with respect to faults of kind 4).

Faults of kind 5) correspond to TFs and CFs affecting the lines DV_{iFF} and DV_{iLD}. These faults, depending on their duration, may propagate through the LUT F and G implementing the combinational logic of the basic cells composing the *DCA*. In this case, similarly to the case of faults of kind 3), they produce an error indication at the output of the CTD. If these faults do not propagate through the LUTs of the basic cells, they do not produce an error indication at the output of our detector. However, we have verified that, when this is the case, our scheme behaves as if it were fault-free. In addition, if following faults belonging to the considered set occur, the *DCA* either gives an output error message, or behaves as if it were fault-free. This holds true for all possible sequences of faults in the considered set. Therefore, our *DCA* is SFS with respect to faults of kind 5). Of course, the detector also maintains its ability to detect the occurrence of CFs or

TFs possibly affecting its monitored line *Di*.

Faults of kind 6) correspond to TFs affecting lines Ei1 and Ei2 of the *DCA*. Considerations similar to those for faults of kind 5) hold true. Therefore, our *DCA* is SFS also with respect to faults of kind 6).

Therefore, we conclude that the *DCA* of our CTD is TSC or SFS with respect to the set of considered faults.

Faults of kind 7) are the same as faults of kind 3). Therefore, our detector is TSC with respect to faults of kind 7).

As for faults of kind 8) and 9), similar considerations to those for faults of kind 7) hold true. Therefore, the *TRCn* is TSC with respect to faults of kind 8) and 9).

Faults of kind 9) correspond to SA0 or SA1 affecting the signals ERR1 and ERR2. Therefore, when activated, these faults produce an error indication at the output of our detector. Thus, the *TRCn* is TSC with respect to this kind of faults.

Faults of kind 10) correspond to TFs affecting the inputs of the *TRCn*. Depending on the fault duration, the checker may, or may not, provide an output error message. As for the faults which do not result in an error message, we have verified that, if they are followed by other faults of the considered set, the checker either gives an output error message, or behaves as if it were fault-free. This holds true for all possible sequence of internal faults in the considered set. Therefore, the checker is SCD with respect to faults of kind 10). It should be noted that, of course, our CTD maintains its ability to detect the occurrence of CFs or TFs affecting the monitored lines.

As for faults of kind 11) and 12), similar considerations to those for faults of kind 10) hold true.

Therefore, we can conclude that the *TRCn* of our detector is TSC or SCD with respect to the set of considered faults.

5. Costs

We have evaluated the costs of our proposed detector in terms of area overhead and impact on system performance (input-output delay). We have compared our solution to the self-checking one presented in [14].

As for the area overhead, we have roughly estimated it in terms of the total number of required CLBs. As for the *DCA*, as previously stated, for each basic cell, our scheme requires 1 CLB to map its combinational logic and one of its two memory elements (i.e., either *FFDi* or *LDi* in Fig. 2). As for the second memory element, it can be mapped into the unused memory elements of the CLBs implementing the checker (Fig. 3). Thus, for *n* monitored interconnects, our detector requires: i) *n* CLBs to map the combinational logic and 1 of the 2 memory elements of the *n DCA*; ii) *(n-1)* CLBs to map the output TRC and *(n-1)* of the remaining *n* memory elements of the *n DCA*; and iii) 1 CLB to map the remaining memory element of the DCA. Therefore, the total area occupation of our *CTD* is: $A_{OUR}=2n$ CLBs.

As for the cost in terms of impact on system performance, the input-output delay of our detector (*d*) is given by the sum of the delays of its *DCA* and connected TRCn.

As for the delay of the Detection Cell Array (d_{DCA}), it is given by the flip-flop/latch propagation delay from the input D to the output Q, plus the propagation delay of a 4-input LUT: $d_{DCA} = \tau_{dq}+T_{LUT} \cong 1.5ns$ [20] for the considered FPGA. As for the propagation delay of the TRCn (d_{TRC}), we should consider that, for a *n*-variable TRC, the number of levels of the binary tree is $log_2(n)$. Since the delay of each TRC level corresponds to the delay of an FPGA LUT, the total delay for a *n*-variable TRC is given by: $d_{TRC}=log_2(n)\cdot T_{LUT}$, where $T_{LUT} \cong 1ns$ [20] is the propagation delay of a 4-input LUT of the considered FPGA. Consequently, the total delay of our *CTD* is: $d_{OUR} = d_{DCA}+ d_{TRC} = \tau_{dq} + T_{LUT} \cdot(1+log_2(n))$.

We have performed a comparison in terms of area occupation and impact on system's performance between our proposed detector and that in [14]. The obtained results are reported in Tab. 1 for n = 16, 32, 64 and 128. The table shows the reductions in terms of area (calculated as $\Delta A(\%)= (A_{[14]} - A_{OUR}) / A_{[14]}$) and delay (calculated as $\Delta d(\%)= (d_{[14]} - d_{OUR}) / d_{[14]}$) allowed by the solution proposed here.

Tab. 1. Area occupation, propagation delay and their relative reductions achievable by our detector over that in [14].

N	Area (# of CLB)			Propagation delay (ns)		
	$A_{[14]}$	A_{OUR}	ΔA	$d_{[14]}$	d_{OUR}	Δd
16	47	32	32.1%	5.7	5.5	3.5%
32	95	64	32.6%	6.7	6.5	3%
64	191	128	32.9%	7.7	7.5	2.6%
128	383	256	33%	8.7	8.5	2%

From Tab. 1 we can see that our detector allows a reduction in area greater than the 32% for all considered cases, a feature particularly important for several applications, like the space ones. In particular, this reduction approaches the 33.3% for $n\rightarrow\infty$. Of course, the overhead of the proposed technique and that in [14] with respect to the system being implemented, depends on the complexity of this latter. In particular, it increases proportionally with the number of monitored interconnects.

Instead, the reduction in the propagation delay allowed by our detector decreases as *n* increases, and the propagation delay of both schemes become equal for $n\rightarrow\infty$.

As previously discussed, in order not to risk to lose any error indication due to faults affecting the Detection Cell Array or Checker itself, both the solution proposed here and that in [14] may require the connection of an output error indicator. As an example, for both solutions, we have considered an error indication of the kind presented in [25].

In this case, our scheme as well as the solution in [14], require an additional CLB at the output of the TRC in order to implement the error indicator [25]. However, the total area occupation of our *CTD* is the same as in the previous case (i.e., without the output error indicator). This because of in our detector, the output TRC implementing the error indicator is mapped in the extra CLB required to map the remaining *DCA* memory element. Therefore, the area occupation of our CTD for *n* monitored interconnects is: $A_{OUR\_EI} = 2n$ CLBs.

Similarly, the input-output delay of our detector, as well as that of [14], increase by the delay of one 4-input LUT. Consequently, the total delay of our proposed detector becomes: $d_{OUR\,EI}= \tau_{dq} + T_{LUT} \cdot (2+ \log_2(n))$.

The costs of our detector and that in [14], including the error indicator at their outputs, are reported in Tab. 2. The table also shows the reductions in terms of area and delay allowed by our proposed detector.

Tab. 2. Area occupation, propagation delay and their relative reductions achievable by our detector over that in [14], assuming an error indication of the kind in [25] connected to their outputs.

N	Area (# of CLB)			Propagation delay (ns)		
	$A_{[14]\_EI}$	A_{OUR_EI}	ΔA	$d_{[14]\_EI}$	d_{OUR_EI}	Δd
16	48	32	33.3%	6.7	6.5	3%
32	96	64	33.3%	7.7	7.5	2.6%
64	192	128	33.3%	8.7	8.5	2.3%
128	384	256	33.3%	9.7	9.5	2%

It is worth noticing that the advantages in terms of area occupation and delay of our detector over that in [14] change negligibly with respect to those reported in Tab. 1. In particular, from Tab. 2 we can observe that our detector allows a constant reduction of area of the 33.3%, for any number of monitored interconnects. As for the reduction in the propagation delay, it decreases as *n* increases, and again the propagation delay of both compared schemes become equal for $n \rightarrow \infty$.

6. Conclusions

We have presented a novel circuit for the on-line detection of CFs and TFs affecting the interconnects of synchronous systems implemented using FPGAs.

We have shown that our proposed CF and TF Detector features self-checking ability with respect to faults possibly affecting itself, thus being suitable for systems with high reliability requirements, like those for space applications.

We have verified the correct operation of our circuit, as well as its self-checking ability, by means of post-layout simulations.

Compared to previous, alternate solutions, the detector proposed here allows a significant reduction of area overhead, while implying a comparable or lower impact on system performance.

References

[1] A.Vasilliou1, K. Gounaris, K. Adaos, D. Mitsainas, G.Alexiou1, D. Nikolos, “Development of Reusable E1 Transeiver Suitable for Rapid Prototyping”, in *Proc. of IEEE Int. Work. on Rapid System Prototyping*, pp. 21 – 26, 1999.

[2] L. Antoni, R. Leveugle, B. Fehér, “Using Run-Time Reconfiguration for Fault Injection Applications”, *IEEE Trans. on Inst. & Measurement*, Vol. 52, No. 5, October 2003.

[3] K. Wu, R. Karri, G. Kuznetsov, M. Goessel, “Low Cost Concurrent Error Detection for the Advanced Encryption Standard”, in *Proc. of IEEE Int. Test Conf.*, pp. 1242-1248, 2004.

[4] G. C. Cardarilli, A. Leandri, P. Marinucci, M. Ottavi, S. Pontarelli, M. Re, A. Salsano, “Design of a Fault Tolerant Solid State Mass Memory”, *IEEE Trans. on Reliability*, vol. 52, No. 4, Dec. 2003.

[5] F. Hanchek, S. Dutt, “Methodologies for Tolerating Cell and Interconnect Faults in FPGAs”, *IEEE Trans. on Comp.*, vol. 47, pp. 15-33, January 1998.

[6] R. Katz, K. LaBel, J. J. Wang, B. Cronquist, R. Koga, S. Penzin, G. Swift, “Radiation Effects on Current Field Programmable Technologies”, *IEEE Trans. on Nuclear Science*, Vol. 44, Issue 6, P. 1, pp. 1945-1956, Dec. 1997.

[7] A. Mukherjee, “Reducing Crosstalk Noise in High Speed FPGAs”, in *Proc. of IEEE Int. SOC Conf.*, pp. 163-164, 2004.

[8] D. Rossi, C. Metra, A. K. Nieuwland, A. Katoch, “Exploiting ECC Redundancy to Minimize Crosstalk Impact”, *IEEE Design & Test of Computers*, Vol. 22, Issue 1, pp. 59 – 70, 2005.

[9] M. Abramovici, C. Stroud, “BIST-Based Detection and Diagnosis of Multiple Faults in FPGAs”, in *Proc. of IEEE Int. Test Conf.*, pp. 785-794, 2000.

[10] M. Renovell, “A Structural Test Methodology for SRAM-Based FPGAs”, in *Proc. of Integrated Circuits and Systems Design*, pp. 9-14, 2002.

[11] C. Bolchini, F. Salice, D. Sciuto, R. Zagaglia, “An Integrated Design Approach for Self-Checking FPGAs”, in *Proc. of IEEE Int. Symp. on Defect and Fault Tolerance in VLSI Syst.*, pp. 443-450, 2003.

[12] S. McCracken, Z. Zilic, “Design for Testability of FPGA Block”, in Proc. of Quality Electronic Design, pp. 86-91, 2004.

[13] P. K. Lala, A. L. Burress, “Self-Checking Logic Design for FPGA Implementation”, *IEEE Trans. on Inst. & Measurement*, vol. 52, no. 5, October 2003.

[14] C. Metra, A. Pagano, B. Riccò, “On-Line Testing of Transient and Crosstalk Faults Affecting Interconnections of FPGA-Implemented Systems”, in *Proc. of IEEE Int. Test Conf.*, pp. 939-947, 2001.

[15] K. Elshafey, “On-Line Diagnosis of Interconnect Faults in FPGA-Based Systems”, in *Proc. of Int. Conf. on Micoelectronics*, pp. 396-399, 2004.

[16] Y. Ran, M. Marek-Sadowska, “Crosstalk Noise in FPGAs”, in *Proc. of Design Automation Conf.*, pp. 944-949, June 2003.

[17] S. Brown, J. Rose, “Architecture of FPGAs and CPLDs: A tutorial”, *IEEE Design & Test of Computers*, vol. 13, Issue 2, pp. 42-57, 1996.

[18] D. A. Anderson, “Design of Self-Checking Digital Network Using Coding Techniques”, *Technical Report R-527*, CSL, Univ. of Illinois, 1971.

[19] http://www.altera.com/literature/hb/stx2/stx2_sii51002.pdf.

[20] Spartan and Spartan-XL Families FPGAs, *Specifications*, http://direct.xilinx.com/bvdocs/publications/ds060.pdf

[21] J. E. Smith and G. Metze, “Strongly Fault-Secure Logic Networks,” *IEEE Trans. Comput.*, vol. C-27, pp. 491 – 499, June 1978.

[22] L. Schiano, C. Metra, D. Marino, “Design and Implementation of a Self-Checking Scheme for Railway Trackside Systems”, in *IEEE Int. On-Line Testing Workshop*, pp. 243-247, 2002.

[23] C. Metra, M. Favalli, B. Riccò, “On-Line Testing Scheme for Clocks’ Faults”, in *Proc. of IEEE Int. Test Conference,* pp. 587-596, 1997.

[24] M. Nicolaidis, “Fault Secure Property Versus Strongly Code Disjoint Checkers”, *IEEE Trans. on CAD,* vol. 13, No. 5, pp. 651-658, May 1994.

[25] N. Gaitanis, D. Gizopoulos, A. Paschalis, P. Kostarakis, “An Asynchronous Totally Self-Checking Two-Rail Code Error Indicator”, in *Proc. of IEEE VLSI Test Symp.*, pp. 151-156, 1996.