Increasing the Dependability of VLSI Systems Through Early Detection of Fugacious Faults

Jaime Espinosa, David de Andrés, and Pedro Gil
Fault-Tolerant Systems Group (GSTF), Instituto de Aplicaciones de las TIC Avanzadas (ITACA)
Universitat Politècnica de València (UPV), Campus de Vera s/n, 46022, Valencia, Spain
Phone: +34 96 3877007 Ext {75774, 75752, 79707}, Fax: +34 96 3877579
Email: {jaiesgar, ddandres, pgil}@disca.upv.es

***Abstract*—Technology advances provide a myriad of advantages for VLSI systems, but also increase the sensitivity of the combinational logic to different fault profiles. Shorter and shorter faults which up to date had been filtered, named as fugacious faults, require new attention as they are considered a feasible sign of warning prior to potential failures. Despite their increasing impact on modern VLSI systems, such faults are not largely considered today by the safety industry. Their early detection is however critical to enable an early evaluation of potential risks for the system and the subsequent deployment of suitable failure avoidance mechanisms. For instance, the early detection of fugacious faults will provide the necessary means to extend the mission time of a system thanks to the temporal avoidance of aging effects. Because classical detection mechanisms are not suited to cope with such fugacious faults, this paper proposes a method specifically designed to detect and diagnose them. Reported experiments will show the feasibility and interest of the proposal.**

Keywords: Fugacious faults; Fault detection; Fault diagnosis

I. Introduction

In recent years, manufacturing capabilities have been evolving at a fast pace, bringing a new breadth of improvements to embedded systems in terms of logic density, processing speed and power consumption. However, those same benefits also become threats to the dependability of systems by causing higher temperatures, shorter timing budgets and lower noise margins, decreasing the probability of manufacturing defect-free devices, and increasing the likelihood of failures originated by wear-out.

Examples of such threats include, among others, i) the growing susceptibility to α-particles and neutrons arriving from outer space or radioactive materials, yielding a non-negligible degree of so called *soft errors* [1], ii) the noise affecting power supply lines which creates unexpected delays in critical paths [2], and iii) the Electromagnetic Interference (EMI), which can be inserted in the system over the air or through wires [3]. With increasing miniaturisation scales, a lesser amount of energy is required for an upset to reach the voltage threshold of the technology and generate a transient fault and, for the same amount of energy, those faults are shorter in duration [4]. New fabrication techniques such as silicon-on-insulator (SOI) follow this trend, with pulse widths decreasing from $250ps$ for $250nm$ feature sizes to $110ps$ for $100nm$ [5]. In the same way, the steady shrinking of voltage thresholds and the dramatic increase of speed have allowed for the propagation of shorter and shorter transient faults. For instance, the minimum pulse width required for propagating a transient fault in SOI has been reduced from $105ps$ for $350nm$ to a $40ps$ for $100nm$ sizes [5]. Finally, in high frequency systems the occurrence of transient upsets increases linearly with the frequency [6]. Altogether, these facts raise concern on the growing numbers and variety of faults next generation systems must face effectively.

Research in the field has focused mainly on fault tolerance [7], i.e. preventing the system from failing via error detection (identifying the presence of an error) and system recovery (transforming the system state into one without errors and without faults that can be activated again). In order to be effective, existing techniques usually require errors to manifest in the state of the system, commonly represented by the contents of its sequential logic. However, deep integration scales are causing an increasing number of transient faults with shorter and shorter activation times (with regards to the clock period), known as *fugacious* faults [8], to be missed as they are not affecting the state of the system (they are not captured by the sequential logic). Although this is the expected behaviour of common fault tolerance mechanisms, the occurrence of fugacious faults is an excellent indicator of harsh environments, wear-out conditions, ageing, extreme temperatures, etc., which may finally lead to a system failure. If such faults are properly detected and diagnosed it could be possible for the system to face new operating conditions, effectively adapting its hardware and software towards new intrinsic or extrinsic demands, even enabling a forecast on future fault impact rates.

Some illustrative examples of the usefulness of an early detection and diagnosis of such fugacious faults include i) a satellite suddenly crossing a high radiation level area, which may trigger a reconfiguration process to increase the system redundancy before the radiation level is high enough to cause a failure, ii) an intentional EMI attack to break secrecy in an encryption core, allowing to change data codification or deploy any other countermeasures, or iii) an ageing problem in a braking control system of a train, which could well raise service alarms before total loss of control. Accordingly, the proper detection and diagnosis of fugacious faults may provide valuable information when taking decisions for the reconfiguration/evolution of the system to keep or improve its safety. If faults are missed, or even detected too late, they could well cause safety hazards, financial losses, or profit drop.

Previous studies devoted specifically to detect and diagnose

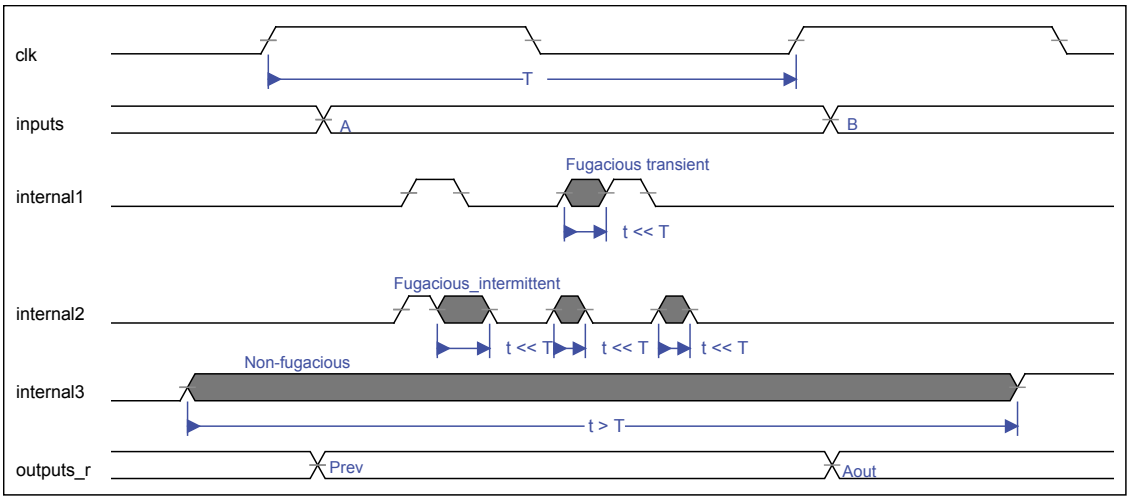


Fig. 1. Characterisation of fugacious faults

fugacious faults are scarce or non-existent. Some works have mainly focused on characterising transient faults caused by radiation along different technologies [9]. However, they do not deal with their detection and diagnosis in working circuits. To this end, certain known detection techniques could be applied to fugacious faults with limited success [10], since only a reduced period of time (with respect to the clock period) is monitored. Hence, new detection capabilities are key to tackle the ever-changing profile of faults as technology advances.

A first step towards this goal was presented at [11], but this paper greatly extends that work by presenting a novel architecture and methodology for the detection and diagnosis of fugacious faults, which may be later used to trigger the system reconfiguration, to keep or improve its dependability, based on fault forecasting.

The rest of the paper is structured as follows. Section II defines the proposed fault models for characterising fugacious faults. Section III introduces the novel architecture and methodology to provide early detection and diagnosis of fugacious faults. The selected case study to show the feasibility of the proposed approach is detailed in Section V, whereas the obtained results are commented in Section VI. Finally, Section VII concludes the paper and presents some ideas for future research.

II. Fugacious fault models

Faults are commonly classified according to their persistence [7] divided then into *permanent* faults, whose presence is continuous in time, and *transient* faults, whose presence is bounded in time. With new integration scales, the classification of transient faults was refined to characterise the particular behaviour of a new kind of faults, the so called *intermittent* faults. They account for transient faults that are repeated in the same area in a short period of time and due to the same cause [12]. Whether the final nature of a fault is transient or intermittent will depend on several factors, namely wear-out condition, ageing, extreme temperatures, etc.

This classical categorisation of faults required a further specialisation to account for the particular behaviour of more frequent and shorter faults induced by increasing integration scales. Accordingly, we propose a refinement of that classification for the characterisation of fugacious faults. It must be noted that the prime characteristic of a fugacious fault is its short duration, that may prevent the fault from being captured by sequential elements. That is why, the system clock cycle is taken as a reference for defining the models of fugacious faults. As such, faults with the same duration could be considered as fugacious or not depending on the system operation frequency. Figure 1 illustrates the proposed fugacious fault models.

Transient fugacious faults are those with a duration shorter than a system clock cycle and that occur just once during that clock cycle. Likewise, *intermittent fugacious* faults also had a duration shorter than a system clock cycle but, on the other hand, they appear more than once within the same clock cycle. As there could exist different activation patterns (number of pulses in a burst, duration of each pulse, and interval between pairs of pulses), this definition enables the characterisation of all the different possibilities under a common term.

Finally, those faults with a duration longer than a clock cycle are considered as *non-fugacious* from the perspective of fugacious faults. Being the scope of fugacious faults limited to the current clock cycle, it is not possible to determine the exact nature of the fault within that clock cycle, but it is necessary to observe the behaviour of the signal along consecutive clock cycles. That is why, from the perspective of fugacious faults, a very long pulse will be considered as a non-fugacious fault, which could be classified as a transient, intermittent or permanent fault from a higher level perspective.

Such differentiated activation patterns require tailored fault detection and diagnosis strategies for fugacious faults, as commonly used mechanisms rely on faults being captured by sequential logic to be detected. Due to the quickly 'evanescent' nature of fugacious faults, the latency of proposed mechanisms should be really low and, what is more, they should be able to detect two or more faults per clock cycle in order to discriminate intermittent from transient activations.

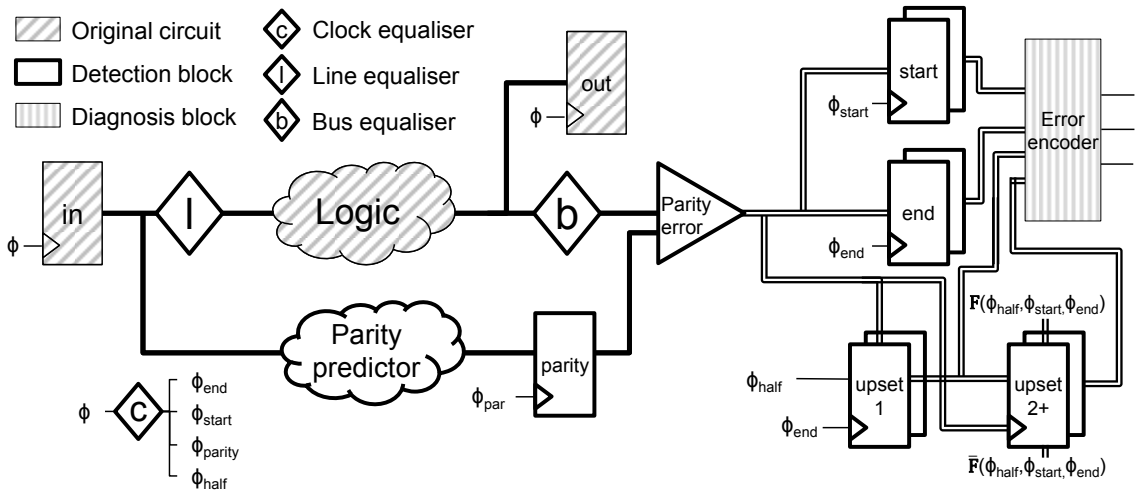


Fig. 2. Low level schematic implementation of the proposed detection and diagnosis architecture

III. Novel architecture for detecting and diagnosing fugacious faults

Many commercial systems make use of hardware replication and comparison, at the expense of greatly increasing the area taken by the circuit and reducing its clock frequency, to mitigate the effect of faults in the system [13]. Nevertheless, for detection and diagnosis, lighter and cheaper techniques would enable the deployment of detection schemes to a larger number of partitions spread around the system, while relying on failure suspectors to trigger reconfiguration strategies to prevent further faults from affecting the circuit. Next sections will show a high- and low-level view of the proposed architecture.

A high-level perspective of the architecture could be roughly described as an error detection code system, which runs in a timing controlled framework to provide inputs to tailored diagnosis infrastructure. Careful selection of such code can provide a number of advantages and literature on the field has been analysed. Systematic codes, like Berger codes and parity groups, are very interesting as the original bits are not altered, which alleviates the decoding of outputs in the original datapath. Equally important, conditions for fault security in parity predictors (outputs are either correct or form an invalid code word) were derived in [14], and a generic optimisation technique for parity prediction functions to achieve fast and small circuits was presented in [15].

Accordingly, a detailed low level view of the proposed architecture is depicted in Figure 2. The original circuit, comprising input and output registers and combinational logic, is highlighted in a diagonal patterned background. The *detection block*, comprising the encoder or parity predictor, the decoder or parity error, the detection auxiliary registers and the timing processing elements (for bus, line and clock), is shown in plain white background. Finally, the *diagnosis block* which consists of a fault tolerant encoder is depicted in vertical lines background. Interconnection double lines represent dual rail encoding to detect the occurrence of single faults in the detection block, as valid data should be either (‘0’, ‘1’) or (‘1’, ‘0’).

The *detection block* receives its inputs directly from partition input registers and from computed outputs just prior to registering. In first place, a parity predictor is in charge of generating the expected output parity code from registered inputs. This parity code is then stored in the *parity* auxiliary register on the ϕ_{parity} clock edge. Particularly tailored codifications could reduce block area and optimise speed, on a case per case basis.

In order to detect both transient (one pulse within one clock cycle) and intermittent (more than one pulse within one clock cycle) fugacious faults, it is necessary to continuously monitor computed outputs prior registering. However, it is not possible to monitor those outputs during the whole clock cycle, since they intermittently switch as the inputs are propagated through logic elements. The monitoring process can take place once computed outputs are stable. Accordingly, just a small fraction of the whole clock cycle could be monitored for fault detection by using this approach.

To overcome those shortcomings, our proposal relies on including a novel element into this architecture, a *Timing Control Unit* (TCU), represented by the clock equaliser depicted in Figure 2. Its function is simple: adjusting the timings of detection elements (ϕ_{parity}, ϕ_{start}, ϕ_{end}, ϕ_{half}) to stretch as much as possible the stability period of the signals (*observation window*), and thus maximise the probability of detecting any fault occurring during the clock cycle. Likewise, line and bus equalisers are also used to selectively delay the required signals with the same purpose.

The parity error decoder receives both the expected parity code and equalised circuit outputs to generate a dual-rail code which is then fed to the *diagnosis block*.

This code is then captured by different dual registers at different specific times. *Start* registers capture at the beginning of the stability period (within the observation window) using clock ϕ_{start}. *End* registers capture at the end of the stability period (within the observation window) using clock ϕ_{end}.

Meanwhile, two sets of dual registers, *upset1* and *upset2+*, are devoted to capture upsets occurred during the whole stability period. To do so, ϕ_{half} which is half the clock

TABLE I. DIAGNOSIS OF FUGACIOUS FAULTS*

Fault	Start	End	Upset1	Upset2+
Non-fugacious	'1'	'1'	'1'	'1'
Transient fugacious	Any '0'	'0' '1'	'1' '1'	'0' '0'
Intermittent fugacious	'1' '0'	'0' Any	'1' '1'	'1' '1'
None	'0'	'0'	'0'	'0'
Diagnosis error	Any other combination			

* For simplicity, '1' here means "00" or "11", and '0' means "10" or "01".

frequency keeps those registers switching each period. In the case of the *upset1* registers, each register of the pair is fed by ϕ_{half} and $not(\phi_{half})$, respectively, using the ϕ_{end} clock edge. Accordingly, their state alternates between ('0', '1') and ('1', '0') each clock cycle, ensuring they store a valid data code. The output of the parity error decoder is connected to the set input of both *upset1* registers, thus causing an invalid ('1', '1') data code whenever an upset is detected. The initialisation of the *upset2+* registers is performed through a combinational function that takes into account ϕ_{half}, ϕ_{start}, and ϕ_{end}. This function and its inverse are respectively connected to the set and reset inputs of one of the registers, and to the reset and set inputs of the other register. This function will only activate those signals outside the observation window. It will also ensure that the registers will store a valid data word alternating between ('0', '1') and ('1', '0') each clock cycle. Upon the occurrence of an upset, the output of the parity error decoder will cause these registers to capture the current state of the *upset1* registers. In case of being the first upset during the clock cycle, they will receive a valid data code. When further upsets are detected, they will store ('1', '1'), an invalid data code signalling that two or more upsets have been detected.

All these data (the parity error at the beginning and end of the observation window, and the state of the *upset1* and *upset2+* registers) will be passed to the encoder in charge of diagnosing the kind of fault, if any, that have been detected. Table I lists the different results of the diagnosis process according to the received inputs. The encoder output is a 3-bit signal that guarantees fault security, as only 4 different codes are required out of the 8 available. The result of the diagnosis process will be passed to a failure suspector that may trigger any corrective measures if required (like reconfiguring the affected component to increase its fault tolerance capabilities or relocating it to another fault free area of the device).

IV. PROPOSED IMPLEMENTATION FLOW

The typical semi-custom design flow for VLSI products should be adapted in order to automate the process of deploying the proposed detection and diagnosis infrastructure. The left hand side of Figure 3 depicts the common implementation flow, where *Technology files* can represent a silicon foundry design kit or an FPGA manufacturer primitives library. Likewise, *Physical* element can be a layout file or a programming bitstream for an FPGA. The right side of the figure details the required addition to that flow to support this novel detection and diagnosis strategy.

The first intervention consists in introducing all the required infrastructure, as previously presented in section III, to support the desired detection and diagnosis. This is performed just before *Synthesis* and after *Design Entry*. Entry files are

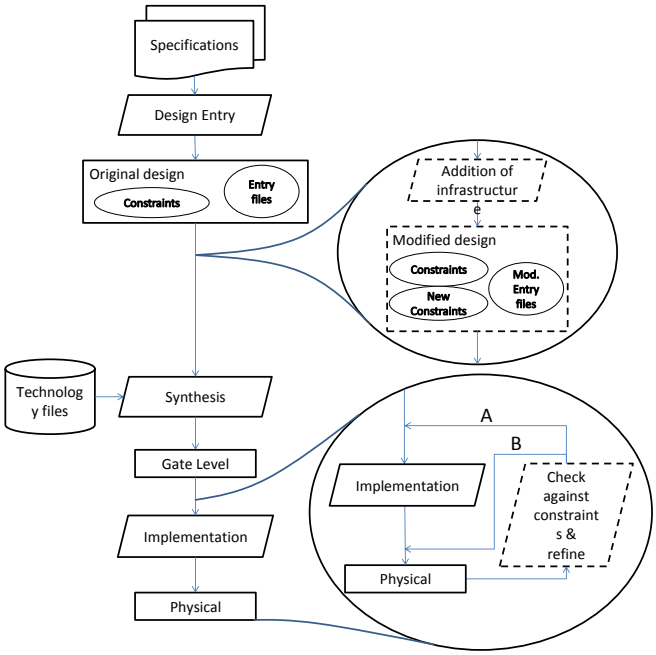


Fig. 3. Proposed implementation flow

modified as required and new timing constraints are generated for the *Timing Control Unit* (TCU).

A second and more complex intervention takes place in a loop between *Gate-level* and *Physical* stages of the design. Its purpose is to stretch the observation window by checking timings against new constraints, mainly affecting the TCU, and successively refining the implementation by tweaking in one of the 2 possible re-entry points. Path *B* leads to a faster, more precise process, but requires a deep knowledge of the underlying technology and reduces the portability of the approach. A more general solution will be obtained using path *A* at the cost of increasing the implementation time. This second intervention follows the flow shown in Figure 4.

The initial step tries to adjust the datapath to stretch the observation window by selectively delaying the output and input lines. First, a bus delay equaliser is used to reduce the dispersion of output paths delays reaching the detection infrastructure (see Figure 5). This is an iterative process in which the slowest output will be preserved as reference, and the fastest output will be delayed by inserting delay elements in its path. This process ends whenever the fastest output is also the slowest one. Once bus outputs have been equalised, a further compaction can be possibly achieved by applying a similar process to the inputs. This time a line equaliser delay is applied to those inputs which appear quicker at the first logic level (see Figure 5), provided they do not violate the same conditions of the bus equaliser. This guarantees that the maximum clock frequency attainable is not affected.

The next step dephases ϕ_{end} so that the clock edge is in close vicinity to end of the stability period (the beginning of the switching period due to the quickest line of the datapath in the fast process corner). By forcing this condition iteratively the end of the observation window is pushed as late in time as possible.

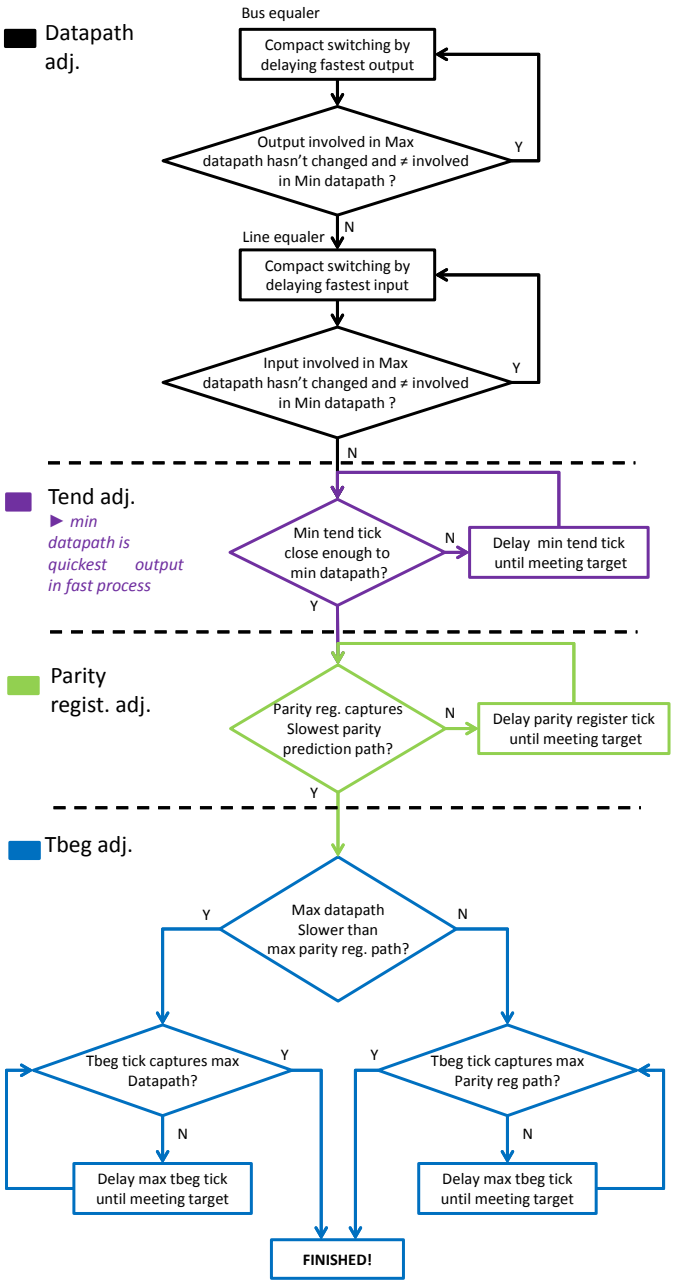


Fig. 4. Control flow for stretching the observation window

The registers devoted to capture the predicted parity are tweaked in the third step. The ϕ_{parity} clock is delayed as necessary to capture the slowest parity prediction signal path. This is done iteratively to adjust the capture tick to the earliest possible moment.

Finally the fourth step focuses on dephasing ϕ_{start} so that the clock edge is located at the beginning of the stability period (observation window). First, it is necessary to determine whether the maximum propagation time of data feeding ϕ_{start} is longer than that of parity registers outputs feeding ϕ_{start}. This clock tick will be iteratively delayed to ensure that both data and parity registers outputs are correctly captured.

In this way the switching period has been shrunk as much as possible, thus enlarging the stability period, and the capture clock edge of all registers has been carefully dephased to observe this whole period.

V. First Prototype and Case Study

In order to test the functionality of the proposed architecture, a standard 4-bit adder has been selected as a suitable candidate. This is a state of the art combinational circuit, used in multitude of designs, which can suffer from undetected fugacious faults. Furthermore its simplicity eases the task of performing controlled experiments (only a small amount of eligible fault injection points for the placed and routed design) within a reasonable time frame.

As this circuit is modelled in Verilog, the required infrastructure to deploy the proposed architecture is also described in this HDL language. This infrastructure and the associated new timing constraints are manually inserted into the design before synthesis, although the automation of this process is currently under development. For its implementation, a Virtex-6 FPGA platform has been chosen due to the ease and speed of deployment, the availability of information related to its internal structure (XDL plain text files), and its partial dynamic reconfiguration capabilities that could be exploited after detecting and diagnosing the occurrence of faults. For the first prototype of a tool supporting this methodology, a publicly available toolset named TORC [16] has been used to interface with the manufacturer tools, which present certain limitations in the routing side hindering the required equalisation and dephasing. As the goal of this first implementation is just showing the feasibility of the proposed approach, it does not tune inserted delays in the finest possible way, so the observation window is not stretched to its physical limits.

The procedure for inserting delays in a target path is based on selecting an intermediate point between origin and destination, for the path to pass through. The location of that intermediate point is determined according to the algorithm shown in Figure 6. This algorithm, based on geometrical principles takes into account different possible cases. Initially, if the delay to be introduced is very small, a position next to the origin or destination points is selected. For small and medium delays, the position is selected somewhere midway between origin and destination, modifying the total path distance according to the required delay. For bigger delays and also when origin and destination points are too close, the intermediate point is placed around a circumference, which includes both origin and destination, whose radius is modulated by the required delay.

This pass-through element can be a simple buffer in ASIC, or a Look-Up Table (LUT) implementing the identity function in an FPGA platform. In the latter case, when the chosen LUT is busy a spiral shaped search for a free close-by LUT is initiated until one is found or a new intermediate position is recalculated. Future work will focus on improving this implementation.

The controlled injection of fugacious hardware faults on an FPGA is also quite difficult. That is why, to test the detection and diagnosis capabilities of the proposed architecture, fugacious faults will be injected in the post-place and route model of the system following a model-based fault injection

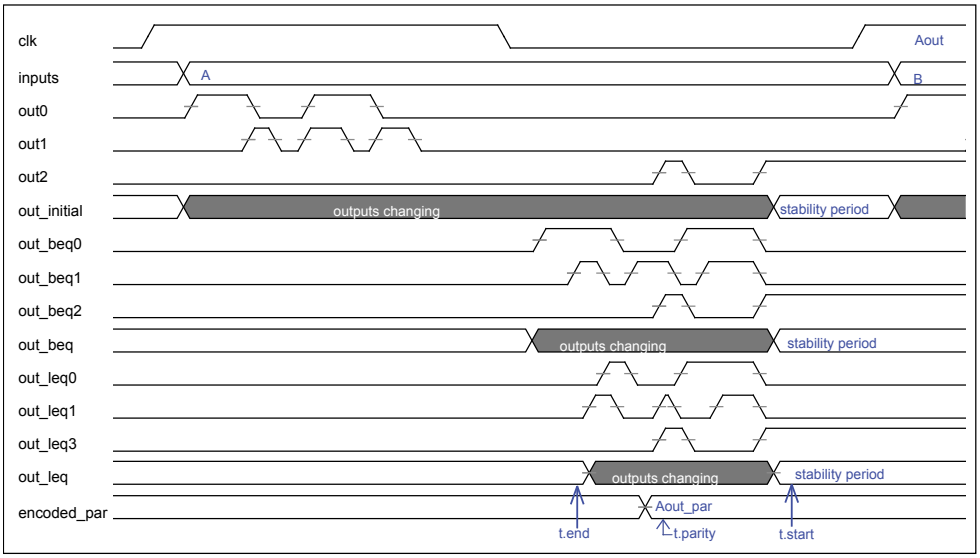
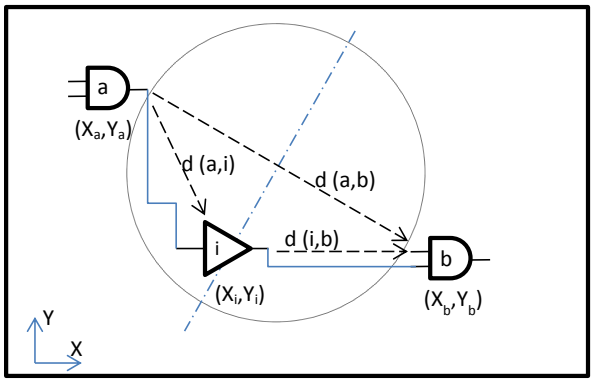


Fig. 5. Stretching the observation window step by step




```
If (Rdelay < 0.5) then
      (Xi,Yi) = (Xa, Ya-1)
else if (Rdelay < 1) then
      Xi = (Xa + dx(a,b)/2)
      Yi = (Ya + dy(a,b)/2)
else if (Rdelay < 2) and (d(a,b) > dmin) then
      α = Rdelay*π/2
      k = atan (dy(a,b)/dx(a,b))
      Xi = Xa + d(a,b)/2 *cos(α + k)
      Yi = Ya + d(a,b)/2 *sin(α + k)
else
      α = Rdelay*π/10
      k = atan (dy(a,b)/dx(a,b))
      Xi = Xa + (d(a,b)/2 +Rdelay*10) *cos(α + k)
      Yi = Ya + (d(a,b)/2 +Rdelay*10) *sin(α + k)
```

Fig. 6. Strategy for locating delay pass-through elements (X_i, Y_i), showing physical distances inside device. Delays are expressed in relative units.

technique. A number of Tcl scripts have been developed to ease the injection of different fugacious and non-fugacious faults using the Modelsim simulator [17]. Next section reports on the results obtained from experimentation.

TABLE II. MINIMUM WIDTH OF FUGACIOUS TRANSIENT FAULTS FOR CORRECT DETECTION

Pulse width T=10ns	Detected in Fast Process Corner	Detected in Slow Process Corner
0.5·T	✓	✓
0.1·T	✓	✓
0.05·T	✓	✓
0.01·T	✗(a)	✓
0.005·T	✗(a)	✓
0.001·T	✗(a)	✓

VI. RESULTS AND DISCUSSION

The first set of experiments is devoted to test the minimum pulse width that could be correctly detected as a transient fugacious fault. Table II lists whether detection is achieved for shorter and shorter pulses in both fast and slow corners of the technology.

The narrowest pulses present a *pulse swallowing* effect, annotated as (a) in the table. In this cases, the physical limitation of the technology applies, as pulses are not wide enough to be propagated through the logic elements. This prevents those faults from being detected. Curiously enough, simulations do not feature this behaviour in the slow process corner, but common sense dictates it will happen at a certain moment. The proper characterisation of this behaviour requires further research.

The second set of experiments aims at determining the minimum separation (inactivity time) between consecutive pulses within a burst to properly detect it as an intermittent fugacious fault. In this case, two pulses with fixed width of $0.05 \cdot T$ are injected with decreasing inactivity time. Table III lists whether faults are correctly detected, for fast and slow process corners, with decreasing inactivity time.

As focusing on detecting faults with very short duration, in case that the total length of the burst is longer than one clock cycle, then several transient faults could be reported instead (not shown in the table). On the other hand, technology limitations (*pulse swallowing*) appear again in those cases

TABLE III. MINIMUM INACTIVE TIME OF INTERMITTENT FUGACIOUS FAULTS FOR CORRECT DETECTION

Inactivity time width T=10ns	Detected in Fast Process Corner	Detected in Slow Process Corner
0.5·T	✓	✓
0.2·T	✓	✓
0.15·T	✓	✓
0.1·T	✓	✗(a)
0.05·T	✓	✗(a)
0.01·T	✗(a)	✗(a)
0.005·T	✗(a)	✗(a)

TABLE IV. CHECK ALL DIAGNOSIS CASES IN ALL ELIGIBLE FAULT INJECTION POINTS

Fault type	Injected value	Number of injections	Correctly detected	Incorrectly detected	Error in diagnosis	Not detected
Transient fugacious	Low pulse	9	6	0	0	3 (b)
	High pulse	9	2	0	0	7 (b)
Intermittent fugacious	Low pulse	9	6	0	0	3 (b)
	High pulse	9	2	0	0	7 (b)
Non-fugacious	Low pulse	9	2	5 (b)	0	2 (b)
	High pulse	9	0	6 (b)	0	3 (b)

marked with (a). It is remarkable that for a slow process corner longer inactivity periods will be required as compared to fast process corner, where correct detection can be achieved for closer pulses within a burst.

Once the technological limitations imposed by the selected Virtex-6 platform are known, a new set of experiments is performed to check the different entries of the diagnosis table (see Table I) on all the 9 eligible fault injection points of the circuit selected as case study. Table IV details the results of this experimentation, reporting the number of faults correctly and incorrectly detected, or not detected at all. It must be noted that, as pulses width has been carefully selected according to the previous experiments, results are exactly the same for both fast and slow process corners, so they are just reported once on the table.

The first unexpected, but foreseeable result, is that non-fugacious faults are first detected as transient fugacious faults. This makes sense, as those faults last more than one clock cycle and there occurrence will be reported as transient faults during this first cycle. However, on the next clock cycle, they will be correctly diagnosed as non-fugacious faults. Accordingly, failure suspectors should take into account this phenomenon to wait for the next clock cycle before taking any decision and action.

Likewise, it can be noted that there exists a big number of not detected faults, marked as (b) on the table. After carefully analysing the simulations, we can conclude that this is the result of *logic filtering*. For instance, injecting a logic '0' in a node holding this same value, or injecting a logic '1' to the input of an AND logic gate when some other input holds a '0', prevents the fault from being propagated and thus detected. When this *logic filtering* is applied to non-fugacious faults, they will be probably detected as transient fugacious faults when the affected line should be switching along consecutive clock cycles. Obviously, the erroneous value should appear during two consecutive clock cycles at least to be correctly diagnosed. As in this case non-fugacious faults have been injected for $1.3 \cdot T$, they are mostly erroneously detected or not detected at all.

Finally, the overhead induced by the required detection and diagnosis infrastructure is shown in Table V.

TABLE V. OVERHEAD INDUCED IN TERMS OF AREA AND CLOCK PERIOD

Design	Area utilisation	Clock period
Original with I/O registers	6 Slices	1.83 ns
Protected with infrastructure	36 Slices	3.49 ns

Although the extra area for the required infrastructure could seem a bit too high in comparison with that required by the original circuit (about six times more), it must be noted that this is really a very small circuit. Furthermore, the Virtex-6 slice is huge in size (4 LUTs, 8 FFs, plus infrastructure) and those 30 additional slices are mostly using a small fraction of the available resources ($< 30\%$), so there is plenty of room for further circuit packing.

Finally, the attained clock period is not as close to that of the original circuit as desired. This is due to the tools used for the bus, line, and clock equalisation processes, which did not allow a fine control of the extra delay to be inserted. For the future it is intended to switch to a different more controlled method to add finely measured delays, taking into account manufacturing dispersions.

VII. CONCLUSIONS

This paper presents a novel methodology to detect and diagnose a specific type of previously neglected faults, the so called fugacious faults. They are described as faults whose active period is smaller than the clock period of the system and they are not usually captured by the sequential logic of the system. Accordingly, they have been usually neglected, but their proper detection and diagnosis could be of great use to foresee the the proximity of harsh environments or the occurrence of wear-out mechanisms. The proposed approach appears as an effective method to quickly detect and diagnose such faults.

From the set of experiments performed to shown the feasibility of this approach, a number of lessons has been learned. First and foremost is that technology limits set a barrier against the quickest detectable upset in the circuit, which applies to each single pulse or the separation between pulses in the same burst. Another important fact is that logic filtering limits the observability of faults, so it makes sense to pay attention exclusively to output lines. Additionally, although its implementation in a modern FPGA may seem costly in terms of area penalty, if applied to large blocks or critical circuits cost versus benefit ratio is greatly improved.

Future research will focus on improving the equalisation tools to fine tune the insertion of controlled delays into the system, thus increasing the observation window and reducing the clock period penalty to the minimum possible. Likewise, the inclusion of a failure suspector in conjunction with the dynamic partial reconfiguration capabilities of FPGA will greatly increase the safety of the target system through its adaptation to face the unexpected events that may arise.

ACKNOWLEDGEMENTS

This work has been funded by the Spanish Ministry of Economy ARENES project (TIN2012-38308-C02-01).

REFERENCES

[1] JEDEC, "Measurement and reporting of alpha particle and terrestrial cosmic ray-induced soft errors in semiconductor devices," in *JEDEC Standard JESD89A*. JEDEC, 2006.

[2] J. Freijedo, L. Costas, S. J., J. J. Rodríguez-Andina, M. J. Moure, F. Vargas, I. C. Teixeira, and J. P. Teixeira, "Impact of power supply voltage variations on fpga-based digital systems performance," *Journal of Low Power Electronics*, vol. 6, no. 2, pp. 339–349, 2010.

[3] Y. Hayashi, S. Gomisawa, Y. Li, N. Homma, K. Sakiyama, T. Aoki, and K. Ohta, "Intentional electromagnetic interference for fault analysis on aes block cipher ic," in *Electromagnetic Compatibility of Integrated Circuits (EMC Compo), 2011 8th Workshop on*, Nov 2011, pp. 235–240.

[4] V. Ferlet-Cavrois, P. Paillet, D. McMorrow, A. Torres, M. Gaillardin, J. S. Melinger, A. R. Knudson, A. Campbell, J. Schwank, G. Vizkelethy, M. Shaneyfelt, K. Hirose, O. Faynot, C. Jahan, and L. Tosti, "Direct measurement of transient pulses induced by laser and heavy ion irradiation in deca-nanometer devices," *Nuclear Science, IEEE Transactions on*, vol. 52, no. 6, pp. 2104–2113, Dec 2005.

[5] P. E. Dodd, M. R. Shaneyfelt, J. A. Felix, and J. R. Schwank, "Production and propagation of single-event transients in high-speed digital logic ics," *IEEE Transactions on Nuclear Science*, vol. 51, pp. 3278–3284, dec 2004.

[6] S. Buchner, M. Baze, D. Brown, D. McMorrow, and J. Melinger, "Comparison of error rates in combinational and sequential logic," *Nuclear Science, IEEE Transactions on*, vol. 44, no. 6, pp. 2209–2216, Dec 1997.

[7] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," *IEEE Trans. Dependable Secur. Comput.*, vol. 1, pp. 11–33, January 2004.

[8] J. Espinosa, D. Andrés, J.-C. Ruiz, and P. Gil, "The challenge of detection and diagnosis of fugacious hardware faults in vlsi designs," in *Dependable Computing*, ser. Lecture Notes in Computer Science, M. Vieira and J. Cunha, Eds. Springer Berlin Heidelberg, 2013, vol. 7869, pp. 76–87. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-38789-0_7

[9] V. Ferlet-Cavrois, L. Massengill, and P. Gouker, "Single event transients in digital cmos -a review," *Nuclear Science, IEEE Transactions on*, vol. 60, no. 3, pp. 1767–1790, June 2013.

[10] S. D'Angelo, G. R. Sechi, and C. Metra, "Transient and permanent fault diagnosis for fpga-based tmr systems," in *Proceedings of the 14th International Symposium on Defect and Fault-Tolerance in VLSI Systems*, ser. DFT '99. Washington, DC, USA: IEEE Computer Society, 1999, pp. 330–338.

[11] J. Espinosa, D. de Andrs, J. C. Ruiz, and P. Gil, "Ideas towards early detection of fugacious faults for increased safety of vlsi systems," in *ITACA WIICT - Workshop on Innovation on Information and Communication Technologies*, 2014, pp. 25–34.

[12] C. Constantinescu, "Intermittent faults and effects on reliability of integrated circuits," in *Proceedings of the 2008 Annual Reliability and Maintainability Symposium*. Washington, DC, USA: IEEE Computer Society, 2008, pp. 370–374.

[13] F. de Lima Kastensmidt, L. Carro, and R. Reis, *Fault-Tolerance Techniques for SRAM-based FPGAs*, ser. Frontiers in Electronic Testing. Springer, 2006, vol. 32.

[14] M. Nicolaidis, S. Manich, and J. Figueras, "Achieving fault secureness in parity prediction arithmetic operators: General conditions and implementations," in *Proceedings of the 1996 European conference on Design and Test*, ser. EDTC '96. Washington, DC, USA: IEEE Computer Society, 1996, pp. 186–. [Online]. Available: http://dl.acm.org/citation.cfm?id=787259.787620

[15] S.-B. Ko and J.-C. Lo, "Efficient realization of parity prediction functions in fpgas," *J. Electron. Test.*, vol. 20, no. 5, pp. 489–499, oct 2004. [Online]. Available: http://dx.doi.org/10.1023/B:JETT.0000042513.15382.e7

[16] University of South California, "Tools for Open Reconfigurable Computing," 2014. [Online]. Available: http://torc-isi.sourceforge.net

[17] Mentor Graphics, "Modelsim," 2014. [Online]. Available: http://www.mentor.com/products/fpga/model