electronics

MDPI

Article

Fault Tolerant Digital Data-Path Design via Control Feedback Loops

Oana Boncalo [1,*], Alexandru Amaricai [1] and Zsófia Lendek [2]

1 Department of Computers and Information Technology, Universitatea Politehnica Timisoara, Piata Victoriei 2, 300006 Timisoara, Romania; alexandru.amaricai@cs.upt.ro

2 Department of Automation, Technical University of Cluj-Napoca, Memorandumului 28, 400114 Cluj-Napoca, Romania; zsofia.lendek@aut.utcluj.ro

* Correspondence: oana.boncalo@cs.upt.ro; Tel.: +40-256-403264

Received: 31 August 2020; Accepted: 13 October 2020; Published: 19 October 2020

Abstract: In this paper, we propose a novel fault tolerant methodology for digital pipelined data-paths called Control Feedback Loop Error Decimation (CFLED), that reduces the error magnitude at the outputs. The data-path is regarded from a control perspective as a process affected by perturbations or faults. Based on the corresponding dynamic model, we design feedback control loops with the goal of attenuating the effect of the faults on the output. The correction loops apply correction factors to selected data-path registers from blocks that have their execution rewinded. We apply the proposed methodology on the data-path of a controller designed for a 2-degree of freedom robot arm, and compare the cost and reliability to the generic triple modular redundancy. For Field Programmable Gate Array (FPGA) technology, the solution we propose uses 30% less slices with respect to Triple Modular Redundancy (TMR), while having a third less digital signal processing blocks. Simulation results show that our approach improves the reliability and error detection.

Keywords: fault tolerance; reliability; arithmetic data-path; FPGA; control engineering; feedback controller

1. Introduction

Reliability represents a key factor in electronic devices that operate in radiation prone environments, frequent in applications in the aerospace domain. It is achieved by adding redundancy to the digital circuits, with cost increase acting as the main drawback. Fault tolerant design is based on adding some form of replication, or using specially designed error detection and correction codes, so that the cost overhead with respect to the nominal solution is acceptable, and the target reliability is achieved.

The most common way of increasing the reliability of a circuit is to employ Triple Modular Redundancy (TMR), i.e., using three copies of the digital circuit and a voter structure [1]. The main drawbacks of TMR are its high cost and the sensitivity of errors affecting the voters. Regarding the former, schemes to reduce the cost of the TMR, such as approximate TMR [2,3] or inexact TMR [4], have been proposed. Furthermore, the use of TMR in the context of approximate computing is discussed in [5]. For arithmetic circuits, the concept of reduced precision replicas has been applied, with the most significant part of the circuit being triplicated [6,7]. Reduced precision replicas can also be used in a dual manner, with the lower cost data-path working in a higher supply voltage domain and providing a highly reliable reference [8]. Other works improved reliability by increasing the resilience of the voting circuits within the TMR, such as the self checking voting circuits [9], adaptable bit-width voter [10], or employing dedicated voters for approximate TMR [11].

Modular redundancy approaches use multiple instances or truncated versions of the same nominal circuit. Other solutions use different types of redundancies. For example, approaches using error correction codes—Hamming or Reed–Muller codes—in combinational circuits have been proposed in [12]. Redundant residue number systems have been employed for signal processing data-paths in [13,14]; in this case, modular redundancy is employed, with each replica having its own residue representation. Another approach uses a design inspired from the Markov Random Field (MRF) theory. In [15], a complementary dual modular redundancy with an MRF voter is proposed, while [16] uses coding-based partial MRF, where the circuit is employed using MRF-based logic components. The common drawback of MRF-based implementations is the high implementation cost. Therefore, for non time-critical applications, several approaches using testing methods and redundancy have been proposed [17], with fault detection being performed during a circuit testing phase.

Different from all the mentioned approaches, this work proposes a control engineering approach—Control Feedback Loop Error Decimation (CFLED)—for improving the reliability of digital data-path processing pipelines, by employing feedback control loops to reduce the error magnitude. CFLED targets applications that rely heavily on arithmetic operations, such as signal and image processing or control and artificial intelligence applications. In arithmetic data-paths, error magnitude is an important factor, as low-magnitude errors may by tolerated by the application. The goal of reducing the error magnitude, and not to completely remove the fault, is also employed in fault tolerant techniques that rely on approximate redundancy [2,5], inexact redundancy [4], or reduced precision replicas [7]. The proposed technique may be applied for arithmetic data-paths implemented on both Application-Specific Integrated Circuit (ASIC) and Field Programmable Gate Array (FPGA) technology.

To the best of our knowledge, CFLED represents the first approach that uses control theory in order to improve the reliability of processing data-paths. Fault-tolerant control for finite state machines has been considered in e.g., [18,19]. The design for these use cases is aided by the finite state machine characteristics: finite and rather low (at most tens) number of states, low number of inputs (typically 1-bit inputs), and finite number of state transitions. In contrast, the proposed approach considers a dynamic model that captures high order input-state-output dynamics, where inputs, states, and outputs have continuous values within their variation domains.

We analyzed the proposed approach on the hardware architecture for the two-degree of freedom robotic arm controller (2-DOF) presented in [20]. The considered use case consists of a processing pipeline that relies on function evaluations—performed using Taylor series—scalar-matrix multiplications, and vector-matrix multiplications. Such operations are widely used in applications such as graphic processing and signal and image processing. Furthermore, this use case represents a novel application in control engineering: design of fault tolerant controller when the faults affect the controller implementation. Other approaches for fault-tolerant control, such as the ones used for electronic power converters [21], induction motors [22], filters [23], robot control, and localization [24,25], assume faults in components of the controlled system, usually actuators or sensors.

This paper is organized as follows: Section 2 discusses the methodology—the discrete time dynamical model of the circuit used to derive the feedback controller; Section 3 discusses a case study for the proposed methodology and presents FPGA implementation costs and reliability estimates; concluding remarks are presented in Section 4.

2. Control Feedback Loop for Fault Mitigation

The proposed methodology targets arithmetic data-path circuits consisting of several processing stages built out of arithmetic operations: additions, accumulations, multiplications, and multiply-add fused. The operands can be either registers or constants. Thus, we consider data-paths commonly used in a wide variety of applications fields that rely on matrix-vector/matrix-matrix multiplications, convolutions, or weighted sums, and are therefore dominant in fields like signal and image processing, computer graphics, or artificial intelligence. Each processing stage i consists of a sequence of registers $\{R_0^i, R_1^i, \ldots, R_{n_i}^i\}$, and it takes n_i clock cycles to compute a set of n_s^i data output elements out of a total of n_{el}^i data values stored to memory or inside registers. Thus, in order to produce a set of output results for stage i, a total of $n_i \times \lceil n_{el}^i / n_s^i \rceil$ clock cycles (cc) are required.

The design process for enhancing the reliability of a given nominal circuit data-path involves: (i) Modeling the nominal circuit, (ii) CFLED design and simulation, (iii) CFLED operation, and (iv) hardware implementation of the enhanced CFLED circuit.

2.1. Dynamic Model of Digital Data-Path

Given a digital data-path, our first goal is to develop a mathematical model of the operations performed and the propagation of values through the data-path. This model forms the basis for designing a control law that will be used to correct the output of the circuit.

A common way to model a system from a control perspective is by using a state space representation. Since digital circuits are synchronous, we employ a discrete time-domain state-space representation, where the sample k corresponds to the current clock cycle, and has the general form:

$$\boldsymbol{x}(k+1) = \boldsymbol{f}(\boldsymbol{x}(k), \boldsymbol{u}(k), \boldsymbol{\eta}) \tag{1}$$

where $\boldsymbol{x}$ denotes the states describing the circuit, $\boldsymbol{u}$ the inputs, $\boldsymbol{\eta}$ some parameters of the dynamic system, and $\boldsymbol{f}$ is a vector function—to be determined—that describes the evolution of the states in time. For a given circuit, each (partial) result (in case of iterative loops) is mapped to a state variable. Hence, each meaningful register value at different clock cycles has a corresponding state variable. Furthermore, a data-path (process) model is built by the composition of several processing stages i.

In this work we consider arithmetic data-path circuits, with the operations: addition, accumulation, multiplication, and multiply-add. Since we map each (partial) result to a state, the value of the state may be obtained as either: (1) addition of a constant to a state; (2) addition of two states; (3) multiplication of a state by a constant; or (4) multiplication of two states. The dynamic model that describes the circuit aggregates all the operations throughout the clock cycles and therefore it may contain only these four mathematical operations. Consequently, the model chosen is of the form:

$$\boldsymbol{x}(k+1) = Q\kappa(\boldsymbol{x}(k)) + A\boldsymbol{x}(k) + a + \delta \tag{2}$$

where $\kappa(\boldsymbol{x}(k))$ denotes the Kronecker product of the vector $\boldsymbol{x}(k)$ with $\boldsymbol{x}(k)$, Q, and A are matrices of appropriate dimensions, a is the vector of biases or affine terms, and δ denotes the faults, possibly with a known distribution.

To determine the exact values of the parameters—matrices Q and A and affine term a—and the correspondence between the states and registers, we adopt the following convention:

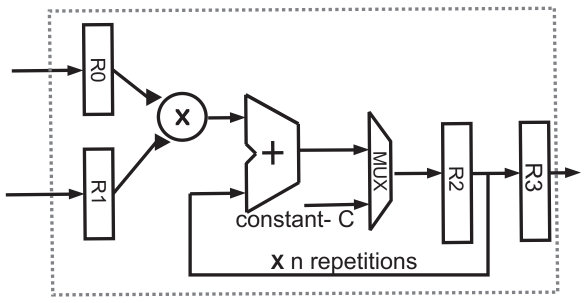
1. The affine term a contains all inputs to the circuit.
2. Since scalar multiplications are commutative, the Q matrix is not unique. Nonzero values should appear in the first element corresponding to a multiplication.

The model and its parameters are obtained as follows:

1. Write the chronological operations for each clock cycle;
2. assign a state variable to every register containing a partial result at different clock cycles;

3. define the dynamic of each state variable as the operations performed to obtain the value in the corresponding register at the corresponding clock cycle;
4. separate the bilinear, linear, and affine terms in the resulting equations;
5. collect the coefficients of the bilinear terms in the matrix Q, the coefficients of the linear terms in the matrix A, while the affine terms, including the inputs to the data-path, will form the vector a.

Note that a model developed for a given circuit is not unique. We have formulated a generic dynamic structure that can be used for any circuit implementing arithmetic operations. In what follows, we illustrate the modeling steps on the circuit in Figure 1.



(**a**) Circuit

$$\begin{aligned} R0(0) &= In_1 \\ R1(0) &= In_2 \\ R2(1) &= R0(0) \times R1(0) \\ R2(2) &= R2(1) + R0(0) \times R1(0) \\ &\vdots \\ R2(n) &= R2(n-1) + R0(0) \times R1(0) \\ R2(n+1) &= R2(n) + C \\ R3(n+2) &= R2(n+1) \end{aligned}$$

(**b**) Chronological operations for each clock cycle

Figure 1. Modeling example: $Ri(m)$ denotes the value assigned to register Ri at the clock cycle m, and In_1 and In_2 are the multiply-accumulate (MAC) design unit inputs.

In this example, In_1 and In_2 denote the inputs of the circuit. We define the state variables as the register values (states) in reverse order, i.e., $x_1(k+1) = R3(n+2)$, $x_2(k+1) = R2(n+1)$, etc., and express each of them as a function of the state variables at the previous sample. This leads to the state equations:

$$\begin{aligned} x_{n+3}(k+1) &= In_1 \\ x_{n+2}(k+1) &= In_2 \\ x_{n+1}(k+1) &= x_{n+3}(k) \times x_{n+2}(k) \\ x_n(k+1) &= x_{n+1}(k) + x_{n+3}(k) \times x_{n+2}(k) \\ &\vdots \\ x_3(k+1) &= x_4(k) + x_{n+3}(k) \times x_{n+2}(k) \\ x_2(k+1) &= x_3(k) + C \\ x_1(k+1) &= x_2(k) \end{aligned} \tag{3}$$

which can be written in a vector form as

$$\boldsymbol{x}(k+1) = Q\kappa(\boldsymbol{x}(k)) + A\boldsymbol{x} + a$$

where

$$Q \;=\; 0_{(n-1)\times((n+2)(n+3)-1)} \quad \mathbb{1}_{(n-1)} \quad \begin{matrix} 0_{2\times(n+3)^2} \\ 0_{n\times(n+3)} \\ 0_{2\times(n+3)^2} \end{matrix}\;, \; A \;=\; \begin{pmatrix} 0_{n\times 1} & I_n & 0_{n\times 2} \\ 0_{n\times 1} & 0_n & 0_{n\times 2} \end{pmatrix}, \; a \;=\; \begin{pmatrix} 0 \\ C \\ 0_{(n-1)\times 1} \\ In_2 \\ In1 \end{pmatrix},$$

with $0_{n\times m}$ we denote the zero matrix of dimensions $n \times m$, I_n the identity matrix, and $\mathbb{1}_n$ a column vector of 1 s.

Models of multiple pipelined sub-components can be combined as follows: (1) The outputs of pipeline stage block i become states of pipeline stage $i+1$; (2) modify the matrices A and Q and vector a appropriately. Thus, larger models of more complex circuits can be built from individual sub-models of its parts.

2.2. Controller Design

The first objective of the controller is to reduce the fault effect at the digital circuit outputs, such that it is as close as possible to a reference value (i.e., gold output result). The second objective is to converge as fast as possible to the correct value. A way to include control inputs into model (2) is to include a linear input, modifying the system model as

$$\begin{aligned} \boldsymbol{x}(k+1) &= Q\kappa(\boldsymbol{x}(k)) + A\boldsymbol{x}(k) + a + \delta + B\boldsymbol{u}(k) \\ \boldsymbol{y}(k) &= C\boldsymbol{x}(k) \end{aligned} \tag{4}$$

where $\boldsymbol{u}$ denotes the vector of the control inputs, k is the current clock cycle, B is the input matrix that determines which state is affected by which input and in what measure, $\boldsymbol{y}$ is the vector of outputs available for correction, and C is the output selection matrix. Next, we define some specific design constraints.

From the circuit side, we need to ensure that even if multiple states are mapped to the same register, the control input is the same. Note that in (4), the matrix B is an n_x-by-n_u matrix, where n_x is the number of states and n_u is the number of inputs. In this matrix, each element B_{ij} indicates the measure in which the input corresponding to the column j is applied to the state corresponding to the row i. If the same input is applied to several states, then the values corresponding to these states can all be set to the same value. Thus, we enforce that the same control (correction) input is applied to all the states corresponding to the same register by the appropriate choice of the values in the matrix B.

The control input $\boldsymbol{u}(k)$ is usually computed based on the outputs of the circuit and a given reference. Cost-wise, the preferred solution is the simplest possible, mainly, a linear control law, i.e., when the control input $\boldsymbol{u}(k)$ is computed as $\boldsymbol{u}(k) = -K\boldsymbol{y}(k)$, with K a constant matrix. The controller gain K is determined such that the closed-loop system—meaning that the control input is applied to the system described by (4)—is asymptotically stable. Thus, we have experimented with the control input computed as $\boldsymbol{u}(k) = -K(\boldsymbol{y}(k) - \boldsymbol{y}^d)$, where $\boldsymbol{y}^d$ are the desired values. However, since model (4) is in most cases nonlinear, such a controller may not suffice, in which case nonlinear controllers can be designed.

The proposed methodology consists of the steps depicted in Figure 2. The inputs are: a register transfer level nominal digital design, a set of application specific constraints (e.g., the acceptable output error), and the aforementioned design specific constraints. The output is a CFLED model.

A final remark concerns the operands data representation, since it is an important implementation cost factor. Although in control engineering floating point is the preferred representation format, from the circuit design point of view, having fixed point operands is a major source of cost saving. Many designs are fine tuned to the application specific needs and use custom fixed point representation. Thus, if a fixed-point representation is chosen, the CFLED gains, as well as resulting states, cannot exceed the representation used for the initial implementation, and use fixed point representation as well.

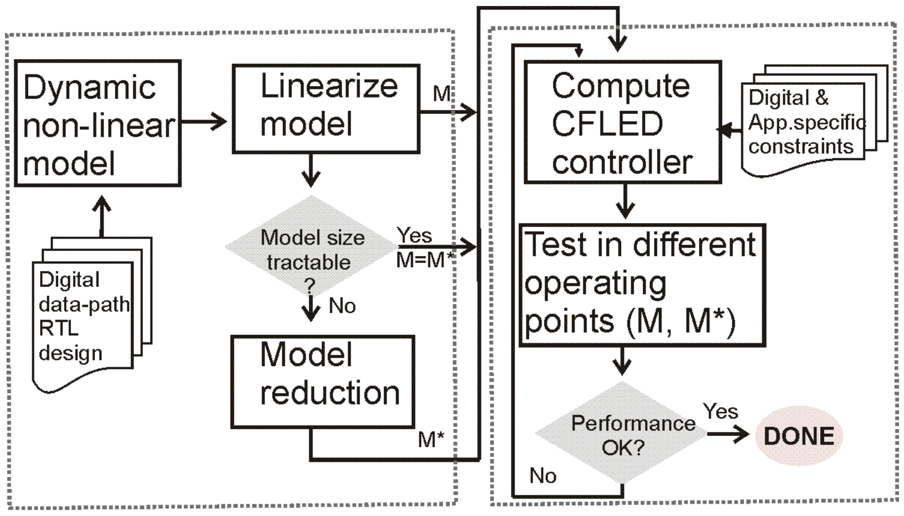


Figure 2. CFLED design steps.

2.3. Fault Tolerant CFLED Operation

Before describing the CFLED operation, we revise the circuit design constraints with respect to timing (i.e., number of clock cycles required computing a result). In digital circuit operation, a pipeline stage result is obtained at well defined moments in time (i.e., after a number of n_i clock cycles for an arbitrary processing stage i). In addition to this, we add the latency constraint that the CFLED enhanced circuit execution can not be larger than twice the execution of the nominal circuit. The latter implies that re-execution of an input data set is not applicable, otherwise we would be unable to justify the additional logic overhead with respect to time redundancy. However, given the principle of the correction loop, an extended execution, such that the correction offsets are added is required. This is handled in parallel for all the pipeline stages having non-zero coefficients in the K matrix, and we refer to it as pipeline stages execution rewind. Thus, latency increase is limited, and all computation sub-blocks are synchronized to the worst case number of clock-cycles it takes to compute a pipeline stage output result. This operating principle is depicted in Figure 3.

To summarize, the operation of the CFLED augmented digital circuit requires the following steps:

1. compute the difference between the output and the reference $(\boldsymbol{y}(k) - \boldsymbol{y}^d))$,
2. determine the correction factors by means of $-K \times (\boldsymbol{y}(k) - \boldsymbol{y}^d)$,
3. rewind the computation by adding the obtained correction factors to the corresponding registers.

It should be emphasized that for a circuit composed of several sub-blocks, the rewinding process is performed only in the sub-modules for which the coefficients in K are non-zero. Furthermore, the re-executions is performed in parallel for the sub-blocks where the correction factors are applied—see Figure 3. Therefore, a correction phase duration is equal to the highest latency of the rewinded sub-modules. The correction process may take several phases/iterations, until the result is below the tolerated threshold.

A key issue is represented by the need of the $\boldsymbol{y}^d$ reference. We solved this problem by employing two CFLED augmented processing pipelines, each providing the reference to the other circuit. In this case, an application specific filter may be employed, in order to exclude results that are out-of-bounds for the specific application.

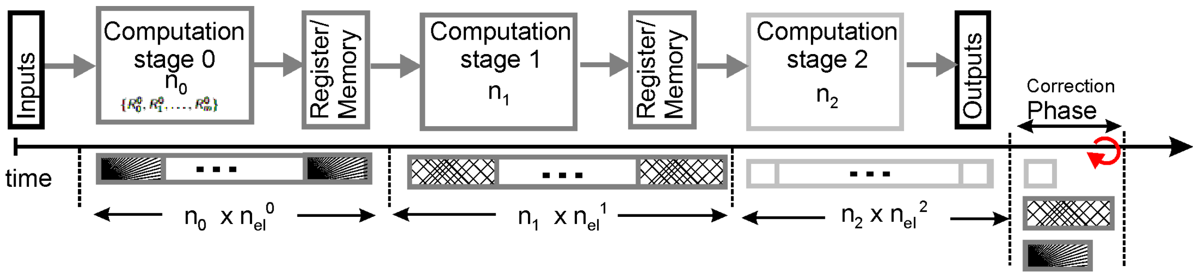


Figure 3. CFLED execution example. Three stages are considered in this example, each computing n_{el}^{i} results. Each result takes n_i cc for computation. First, the normal operation is performed. If, after finishing the computation, a difference is detected at the output, the correction operation is started. All three stages are executed—rewinded —in parallel with correction inputs added to designated registers.

3. Case Study—A Robot Arm Controller

3.1. Nominal Circuit Design

To illustrate the proposed approach, we consider the hardware implementation of a fuzzy controller for a two degree of freedom (2-DOF) robot arm. The parameters of the physical robot are described in [26], and the hardware implementation of the controller in [20]. The output of the nominal (controller) circuit implementation is the vector u (2 elements), that represents the inputs for controlling the robot arm. With x^{Ir} (2 element vector) we denote the controller state, while x^r represents the current state of the robot (4 element vector), and y^r is the reference that the robot arm should follow.

The hardware architecture is a fixed point implementation, using 24 bits operands, 8 bits for the integer part, and 16 bits for the fractional part. The equations and the numerical approximations used are described in [20]. For the purpose of this study, we focus on the register transfer level operations of the five pipelined processing blocks of the data-path depicted in Figure 4:

1. Trigonometric function approximator, that computes the $\sin(x)$, $\sin(2x)$, and $\cos(x)$—these three functions are evaluated based on the 2nd and 3rd order Taylor series, and are computed in parallel;
2. computation of weighting functions—the computation of the three weighting functions h_1, h_2, and v is serialized, with one element computed at a time;
3. multiplication between the weighting functions—this block performs the 8 multiplications between h_1 or $(1 - h_1)$, h_2 or $1 - h_2$, and v or $(1 - v)$ sequentially;
4. final gain matrix computation—this block performs the accumulation of 8 scalar-matrix products; the size of the matrices is 2×6; it employs a single multiply-add fused unit, and therefore, the 12 elements of the final gain elements are computed sequentially;
5. output computation—the output control vector is obtained by multiplication of the final gain matrix with the vector composed of the process state x^r and internal state x^{Ir}; the two elements of the output vector are computed sequentially;

The 2-DOF robot arm controller contains 34 data registers and the processing of a set of input samples requires 175 clock cycles.

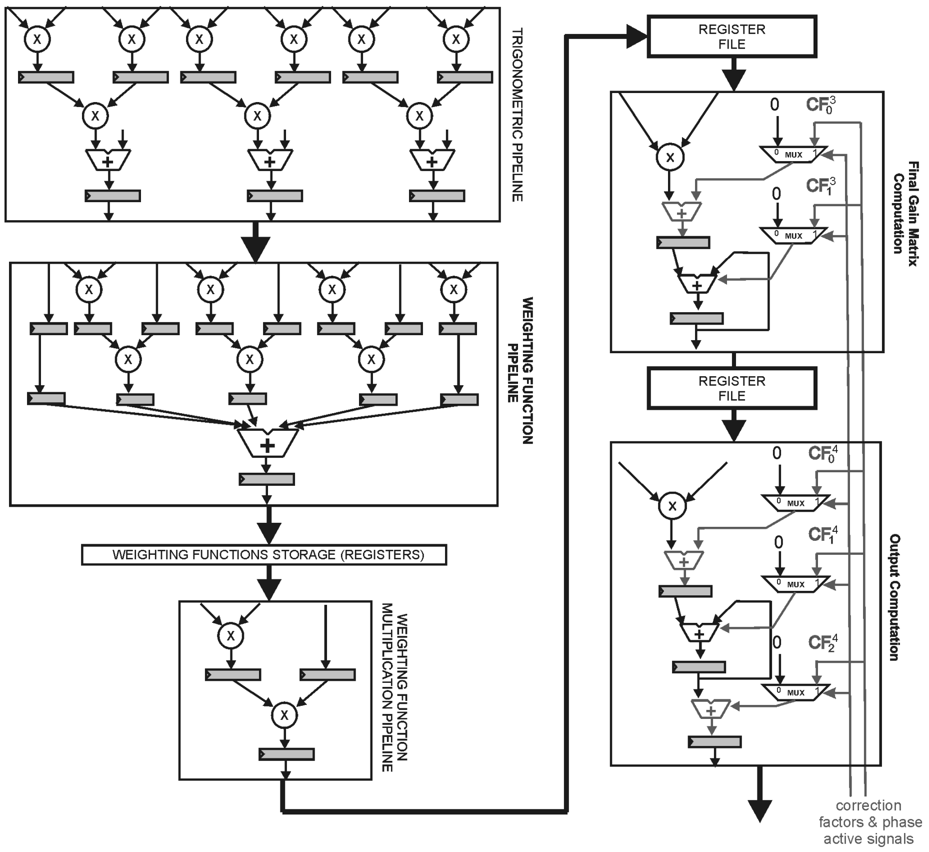

TRIGONOMETRIC PIPELINE
WEIGHTING FUNCTION PIPELINE
WEIGHTING FUNCTIONS STORAGE (REGISTERS)
WEIGHTING FUNCTION MULTIPLICATION PIPELINE
REGISTER FILE
Final Gain Matrix Computation
REGISTER FILE
Output Computation
CF$_0^3$
CF$_1^3$
CF$_0^4$
CF$_1^4$
CF$_2^4$
MUX
correction factors & phase active signals

Figure 4. 2-degree of freedom (DOF) robot arm controller hardware architecture.

3.2. Dynamic Model and CFLED Design

For the implemented architecture, a dynamic model has been developed, following the steps described in Section 2. The output of the system has been defined as the output of the circuit. The model is non-linear (consequence of the multiplication operation), of the form (4), and has 310 states. In addition to the discussion from Section 2, in order to avoid obtaining a switching system or the need to include clock-dependence in the model, the blocks that compute results sequentially (e.g., weighting functions, weighting function multiplication, and output computation) are modeled as parallel in the dynamic model.

For the actual design of the controller, we have opted to linearize the model in the zero equilibrium point, reduce it using a balanced realization, and compute the gain matrix by placing the closed-loop system poles as close as possible to the origin [27]. As a result that K is constant, it is computed offline. Finally, the resulting controller has been tested for the original nonlinear model. The correction gains are constant during the entire correction process and are independent of the fault rate affecting the circuit. The computed gains are multiplied by the difference between the actual and desired output vector $(\boldsymbol{u}[k] - \boldsymbol{u}^d[k])$, and their sum added at each clock cycle to the designated register during the correction operation phase. Furthermore, the vector $\boldsymbol{u}^d$ is updated at the end of each correction phase execution.

Regarding the values in the gain matrix K, we have obtained values that are non-negligible for two out of five blocks: the final gain matrix computation and output computation blocks (5 registers out of 34). Thus, we employ CFLED for these two computational stages. The values for the gain factors corresponding to the two blocks are given in Table 1, where R_i^3 denotes registers within the final gain matrix computation block, while R_i^4 denotes registers within the output computation. During the correction phase, processing is performed only on these two blocks and the duration of the correction is given by the maximum of the latency of the two modules augmented with CFLED. Since the correction gain values are constant (computed offline), optimized implementation of multiplication with constants can be employed, using a simple shift-and-add approach.

First, the Matlab model of the CFLED enhanced 2-DOF controller with golden reference has been built. In the next step, the golden reference has been replaced by a second CFLED controller that is also subject to probabilistic errors. A simple control unit determines if the difference between the outputs is smaller than the application specific error tolerance, chosen as 10^{-3}. If the computed output register difference is larger than this bound, the error correction phase is started. Thus, the fault tolerant solution relies on two CFLED enhanced models connected in reaction, that exchange data at the beginning of each correction phase.

We simulated the Matlab model to asses the performance of the overall system. The behavior of the proposed solution can be observed in Figure 5, which depicts outputs u_1, and u_2 of the two controllers in the presence of errors. Note that the error at the output of one circuit influences the other. The simulation shows that up to cycle 27, the output of the two CFLED circuits are identical; thus, the black and red lines overlap. Then, the first controller has errors manifesting at the output u_1; thus, triggering the correction phase during the next simulation cycle. Due to the correction factors the two outputs influence each other, leading to a small perturbation in output u_2, that is eliminated successfully during further correction phases. This is also the case for the other two errors that are larger in amplitude. The only difference is that the recovery time is larger. Given this observation, it makes sense to include a check in the control unit, that verifies if the output values are within the allowed range. For the considered use-case, the range is $[-3.5, 3.5]$. If both CFLED enhanced circuits yield out of bounds results, then system failure occurs, and computation is restarted. Otherwise, they may serve as reference for each other during an error correction phase, if needed. If only one of the CFLED enhanced circuit outputs exceeds the range, then the other output is considered correct.

Table 1. CFLED K matrix values for registers in Figure 4.

Input No.	Register	Gains u	
1	R_1^3	−0.0011	−0.0038
2	R_0^3	0.0007	0.0024
3	R_1^4	−0.0017	−0.0022
4	R_0^4	−0.0103	−0.0092

3.3. Results and Discussion

In this section, we discuss the results obtained by the proposed approach and compare them with those obtained by a TMR.

3.3.1. Reliability Analysis

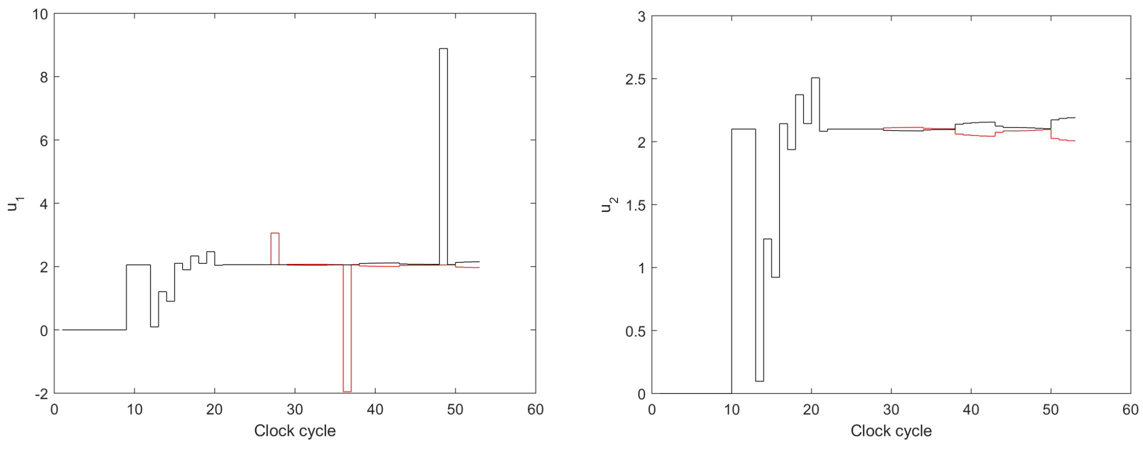
The reliability analysis of the proposed approach has been performed using simulated fault injection, in a Matlab–Modelsim co-simulation environment. Two main components are needed: the Matlab model for the process, and the HDL CFLED RTL design simulated using Modelsim. The Matlab simulation is responsible for generating and sending input data to the robot arm controller. It also generates the correct output results, and handles the statistics recording and result post-processing. The communication between the two CAD programs is through TCP/IP connection.

The simulated fault injection is performed in three phases:

I Setup: initializes all data structures from Matlab from the configuration files, loads the HDL design inside the Modelsim environment with the parameters sent from Matlab, and generates a TCL data structure in the HDL simulation environment that mirrors the Matlab simulated fault injection configuration data (i.e., bit error probability, fault location).

II Simulation runs: One simulation step is run for both Matlab and Modelsim. In the Matlab environment, a set of input data and output gold values are computed. A TCL script with appropriate command parameters is invoked through the TCP/IP connection. Next, the register values for outputs, and the debug registers are read in Matlab, and data are logged in *.csv files.

III Results processing: Matlab scripts process the logged data and compute statistics.



(**a**) u_1 (**b**) u_2

Figure 5. Matlab simulation example. Clock cycle accurate simulations of two CFLED enhanced 2-DOF units connected in reaction, and denoted with colors red and black, respectively.

We have performed the simulated fault injection both on the CFLED augmented architecture, and on a fault tolerant controller that uses TMR. We have considered bit-flips that affect each flip-flop within all data registers, with different fault rates per clock cycle: 10^{-4} , 10^{-5}, and 10^{-6}. During a sample processing, for a bit-flip rate of 10^{-4}, the number of faults injected in the baseline architecture—without any fault tolerant mechanism—is 14. For the fault tolerant designs—CFLED and TMR—faults have been applied for the entire design and during the entire execution.

We present the results in terms of:

1. Output failure rate—in this case, we have counted the number of outputs that have a difference greater than 10^{-3} with respect to the correct output.
2. Gaussian distribution of the output error magnitude in terms of mean (μ) and covariance (σ)—this type of analysis has been performed because in arithmetic data-paths the error magnitude is in many cases more important than the number of errors (such as bit errors) affecting the result. Smaller values of the mean and of the covariance mean an improved distribution of error magnitude, and therefore, increased fault tolerance.

Results are depicted in Table 2 for the first component of the output vector (u_1), and Table 3 for the second component of the output vector (u_2). Although the TMR presents better output failure rate—a smaller number of erroneous frames—the distribution of error magnitude is improved in the case of CFLED. This indicates that although the number of erroneous results is higher in the case of CFLED, the error magnitudes achieved with the proposed method are lower compared to TMR. Therefore, it can be noted that CFLED achieves its primary goal, to reduce or mitigate the magnitude of errors in arithmetic data-paths.

Table 2. Parameters of the Gaussian distributions and output failure rates for $\boldsymbol{u}_1^r$.

Bit Error Probability	Method	μ Mean	σ Covariance	Output Failure Rate
10^{-4}	CFLED	-0.1036	79.54	0.3202
	TMR	-0.5192	236.39	0.1516
10^{-5}	CFLED	-0.0043	0.4196	0.0186
	TMR	0.0277	1.3228	0.0066
10^{-6}	CFLED	-5.99×10^{-4}	1.3×10^{-3}	0.002
	TMR	-2.09×10^{-2}	2.826	0.0007

Table 3. Parameters of the Gaussian distributions and output failure rates for $\boldsymbol{u}_2^r$.

Bit Error Probability	Method	μ Mean	σ Covariance	Output Failure Rate
10^{-4}	CFLED	-0.0886	66.8871	0.3176
	TMR	0.10132	131.82	0.1436
10^{-5}	CFLED	-0.0334	3.6056	0.0229
	TMR	-0.0452	5.501	0.0039
10^{-6}	CFLED	0.0018	0.016	0.002
	TMR	0.0056	0.7344	0.0007

Figure 6 depicts the average number of correction phase executions, depending on the bit error probability. It can be noted that for a bit error probability of 10^{-6}, less than a quarter of phase executions are required on average. Given that a phase execution requires 20 clock cycles—out of 175 clock cycles required for processing one sample—less than 5 clock cycles are added on average to each sample processing. This means that the execution time is increased on average with less than 0.3%.

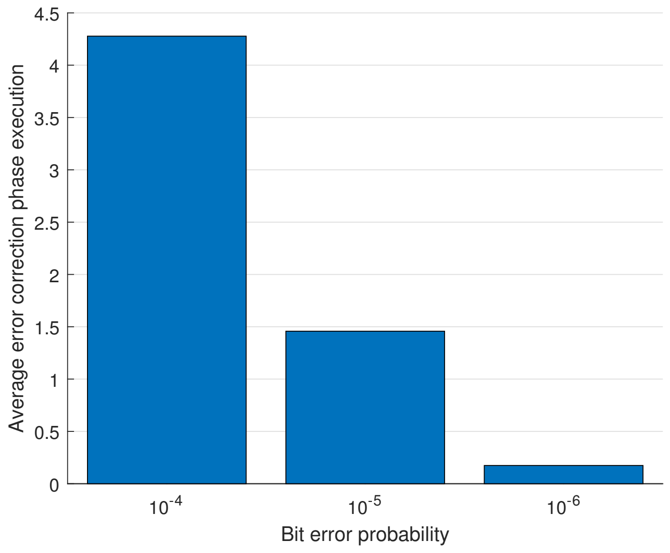


Figure 6. CFLED average phase execution increase for a maximum of 24 correction phase executions.

3.3.2. FPGA Implementation Results

The implementation results for the proposed CFLED enhanced 2-DOF controller design for Xilinx Virtex-7 VX485T-2 FPGA device, using the Xilinx Vivado 2017.1 tool are depicted in Table 4 for the baseline—non-fault tolerant—circuit, and the controlled, fault tolerant version, with two CFLED enhanced circuits providing the reference to the other.

Table 4. Implementation results.

Design	Slices	DSP Blocks	Frequency [MHz]
Baseline	1948	34	142
TMR	5844	102	142
CFLED	4052	68	142

FPGA implementation results indicate that the two circuit CFLED solution has an overhead of 2.1$\times$ in terms of slices, with respect to the baseline circuit. In terms of DSP blocks, the CFLED solution contains the same number of blocks as two baseline circuits. Therefore, the CFLED solution uses 30% less slices with respect to the TMR, while having a third less DSP blocks. The results in Table 4 also indicate the low cost of the correction feedback with respect to the baseline circuit. As a result that this block consists of 4 multipliers with constants—using a simple shift-and-add approach—no dedicated DSP blocks are required, while the overhead in logic slices is rather negligible.

4. Conclusions

To the best of our knowledge, this paper represents the first attempt to improve the reliability of digital processing pipelines by employing control engineering techniques. CFLED requires the development of a dynamic model associated to the processing pipeline, and the design of a feedback controller that computes correction factors. The correction to be applied is computed as the multiplication between offline computed constants with the difference between the circuit's output and the reference, and they are added to a sub-set of registers within the processing pipeline. A key issue is represented by obtaining the reference. In this paper, we propose to use two CFLED augmented data-paths, each providing the reference to the other.

Reliability estimates obtained by means of simulated fault injection for a hardware architecture of a two degree of freedom robotic arm controller indicates that the error magnitude of the CFLED augmented circuit is reduced with respect to a classic TMR-based solution. Regarding the cost overhead for an FPGA-based implementation, CFLED shows a 30% reduction in slice-based resources compared to the TMR.

Thus, future work will consist of reducing the cost of this solution, by employing reduced precision replicas or approximate methods for part of the data-paths.

Author Contributions: Conceptualization, O.B., A.A., and Z.L.; methodology, O.B. and Z.L.; software Matlab model, Z.L.; HDL architecture O.B. and A.A.; validation, O.B. and A.A.; investigation, O.B., A.A., and Z.L.; writing—original draft preparation, O.B. and A.A.; writing—review and editing, Z.L.; visualization, O.B.; supervision, O.B.; project administration, O.B.; funding acquisition,O.B., A.A., and Z.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by European Space Agency ITI REDOUBT activity.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

FPGA	Field Programmable Gate Array
TMR	Triple Modular Redundancy
ASIC	Application-Specific Integrated Circuit
MRF	Markov Random Field
CFLED	Control Feedback Loop Error Decimation
DSP	Digital Signal Processing
HDL	Hardware Description Language
RTL	Register Transfer Level
2-DOF	Two Degree of Freedom
MAC	Multiply-Accumulate

References

1. von Neumann, J. Probabilistic Logics and the Synthesis of Reliable Organisms from Unreliable Components. *Autom. Stud.* **1956**, *34*, 43–98.
2. Sánchez, A.; Entrena, L.; Kastensmidt, F. Approximate TMR for selective error mitigation in FPGAs based on testability analysis. In Proceedings of the 2018 NASA/ESA Conference on Adaptive Hardware and Systems (AHS), Edinburgh, UK, 6–8 August 2018; pp. 112–119.
3. Sánchez-Clemente, A.J.; Entrena, L.; García-Valderas, M. Partial TMR in FPGAs Using Approximate Logic Circuits. *IEEE Trans. Nucl. Sci.* **2016**, *63*, 2233–2240. [CrossRef]
4. Chen, K.; Han, J.; Lombardi, F. Two Approximate Voting Schemes for Reliable Computing. *IEEE Trans. Comput.* **2017**, *66*, 1227–1239. [CrossRef]
5. Rodrigues, G.; Lima Kastensmidt, F.; Bosio, A. Survey on Approximate Computing and Its Intrinsic Fault Tolerance. *Electronics* **2020**, *9*, 557. [CrossRef]
6. Huang, Y. High-Efficiency Soft-Error-Tolerant Digital Signal Processing Using Fine-Grain Subword-Detection Processing. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2010**, *18*, 291–304. [CrossRef]
7. Wali, I.; Casseau, E.; Tisserand, A. An efficient framework for design and assessment of arithmetic operators with Reduced-Precision Redundancy. In Proceedings of the 2017 Conference on Design and Architectures for Signal and Image Processing (DASIP), Dresden, Germany, 27–29 September 2017; pp. 1–6.
8. Wey, I.; Peng, C.; Liao, F. Reliable Low-Power Multiplier Design Using Fixed-Width Replica Redundancy Block. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2015**, *23*, 78–87. [CrossRef]
9. Afzaal, U.; Lee, J. A Self-Checking TMR Voter for Increased Reliability Consensus Voting in FPGAs. *IEEE Trans. Nucl. Sci.* **2018**, *65*, 1133–1139. [CrossRef]
10. Ló, T.B.; Kastensmidt, F.L.; Beck, A.C.S. Towards an adaptable bit-width NMR voter for multiple error masking. In Proceedings of the 2014 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT), Amsterdam, The Netherlands, 1–3 October 2014; pp. 258–263.
11. Arifeen, T.; Hassan, A.; Lee, J.A. A Fault Tolerant Voter for Approximate Triple Modular Redundancy. *Electronics* **2019**, *8*, 332. [CrossRef]
12. Laurenciu, N.C.; Gupta, T.; Savin, V.; Cotofana, S.D. Error Correction Code protected Data Processing Units. In Proceedings of the 2016 IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH), Beijing, China, 18–20 July 2016; pp. 37–42.
13. Luan, Z.; Chen, X.; Ge, N.; Wang, Z. Simplified fault-tolerant FIR filter architecture based on redundant residue number system. *Electron. Lett.* **2014**, *50*, 1768–1770. [CrossRef]
14. Chang, C.; Molahosseini, A.S.; Zarandi, A.A.E.; Tay, T.F. Residue Number Systems: A New Paradigm to Datapath Optimization for Low-Power and High-Performance Digital Signal Processing Applications. *IEEE Circuits Syst. Mag.* **2015**, *15*, 26–44. [CrossRef]

15. Li, Y.; Li, Y.; Jie, H.; Hu, J.; Yang, F.; Zeng, X.; Cockburn, B.; Chen, J. Feedback-Based Low-Power Soft-Error-Tolerant Design for Dual-Modular Redundancy. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2018**, *26*, 1585–1589. [CrossRef]
16. Li, Y.; Li, Y.; Wey, I.; Hu, J.; Yang, F.; Zeng, X.; Jiang, X.; Chen, J. Low-Power Noise-Immune Nanoscale Circuit Design Using Coding-Based Partial MRF Method. *IEEE J. Solid-State Circuits* **2018**, *53*, 2389–2398. [CrossRef]
17. Krištofík, Š.; Baláž, M.; Malík, P. Hardware redundancy architecture based on reconfigurable logic blocks with persistent high reliability improvement. *Microelectron. Reliab.* **2018**, *86*, 38–53. [CrossRef]
18. Yang, J.; Kwak, S.W. Corrective control of parallel interconnected asynchronous sequential machines with output feedback. *IET Control Theory Appl.* **2019**, *13*, 693–701. [CrossRef]
19. Yang, J.; Kwak, S.W. Model matching and fault-tolerant control of switched asynchronous sequential machines with transient faults. *IET Control Theory Appl.* **2019**, *13*, 1882–1890. [CrossRef]
20. Boncalo, O.; Amaricăi, A.; Lendek, Z. Configurable Hardware Accelerator Architecture for Takagi-Sugeno Fuzzy Controller. In Proceedings of the Euromicro Conference on Digital System Design, Kallithea, Greece, 28–30 August 2019; pp. 1–6.
21. Jamshidpour, E.; Poure, P.; Saadate, S. Photovoltaic Systems Reliability Improvement by Real-Time FPGA-Based Switch Failure Diagnosis and Fault-Tolerant DC-DC Converter. *IEEE Trans. Ind. Electron.* **2015**, *62*, 7247–7255. [CrossRef]
22. Cabal-Yepez, E.; Valtierra-Rodriguez, M.; Romero-Troncoso, R.J.; Garcia-Perez, A.; Osornio-Rios, R.A.; Miranda-Vidales, H.; Alvarez-Salas, R. FPGA-based entropy neural processor for online detection of multiple combined faults on induction motors. *Mech. Syst. Signal Process.* **2012**, *30*, 123–130. [CrossRef]
23. Karimi, S.; Poure, P.; Saadate, S. FPGA-based fully digital fast power switch fault detection and compensation for three-phase shunt active filters. *Electr. Power Syst. Res.* **2008**, *78*, 1933–1940. [CrossRef]
24. Srebro, A. Fault-tolerant algorithm for a mobile robot solving a maze. *Challenges Mod. Technol.* **2013**, *4*, 21–29.
25. Zhao, Z.; Wang, J.; Cao, J.; Gao, W.; Ren, Q. A Fault-tolerant Architecture for Mobile Robot Localization. In Proceedings of the 2019 IEEE 15th International Conference on Control and Automation (ICCA), Edinburgh, UK, 16–19 July 2019; pp. 584–589.
26. Lendek, Z.; Nagy, Z.; Lauber, J. Local stabilization of discrete-time TS descriptor systems. *Eng. Appl. Artif. Intell.* **2018**, *67*, 409–418. [CrossRef]
27. Levine, W.S. *The Control Systems Handbook, Second Edition: Control System Advanced Methods, Second Edition*, 2nd ed.; CRC Press, Inc.: Boca Raton, FL, USA, 2009.