Verification of Delta Form Realization in Fixed-Point Digital Controllers Using Bounded Model Checking

Iury Bessa, Hussama Ibrahim, Lucas Cordeiro, and João Edgar Chaves Filho
Electronic and Information Research Center, Federal University of Amazonas, Brazil
E-mails: {iurybessa,hussamaibrahim,lucascordeiro,jo_edgar}@ufam.edu.br

***Abstract*—The extensive use of fixed-point digital controllers demands a growing effort to prevent design's errors that appear in the discrete-time domain. This paper presents a novel verification methodology that employs Bounded Model Checking (BMC) based on the Satisfiability Modulo Theories to verify the occurrence of design's errors, due to the finite word-length format, in fixed-point digital controllers. Here, the performance of digital controllers realizations that use delta-operators are compared to those that use traditional direct forms. The experimental results show that the delta form realization reduces substantially the digital controllers' fragility. Additionally, the proposed methodology can be very effective and efficient to verify real-world digital controllers, where conclusive results are obtained in nearly 95% of the benchmarks.**

Keywords*—*fixed-point digital controllers, delta form, formal methods, bounded model checking.

I. Introduction

The digital controllers are now widely used by the control engineering community due to various advantages over the analog controllers such as improved reliability, sensitivity, flexibility, and cost. However, there are some disadvantages in the use of digital controllers, for example, errors that are introduced during the quantization process. In this context, several control systems researches aim to solve problems that appear in the discrete-time domain, in particular, problems related to the finite word-length realizations.

Digital controllers are generally implemented in microcomputers, microprocessors, digital signal processors [1], and field programmable gate array [2]. According to the hardware choice, the format and arithmetic used to represent and manipulate numbers may change (e.g., number of bits, fixed- or floating-point arithmetic); these representations directly influence the digital controller's precision.

Floating-point processor has a greater number of representable values and consequently, an enhanced precision; however, fixed-point processor is the fastest and cheapest solution, and it is largely used in practice. The scenario aforementioned thus requires better understanding and handling of typical problems in digital controllers realizations so that we can potentially reduce quantizations and finite word-length (FWL) effects during the digital controller's design. Several factors can influence, intensify, or attenuate these effects, e.g., via the use of realization structures such as direct and delta forms as well as the definition of the number of bits and sample rate. The possible influences of these effects bring important stability and eigenvalues sensitiveness issues to the digital controller's design, which were previously investigated by several researchers (e.g., [3]–[7]).

To avoid the performance degradation, control engineers usually invest a greater effort on the design phase, solving problems due to the FWL with more robust and arduous solutions. Previous work propose appropriate scaling using special metrics [4], [8]. Others developed different methodologies to estimate the optimal word-length to avoid FWL effects [9]–[12]. More specialized work in control systems proposes more complex controllers to maintain the performance inside an error bound (or uncertains), e.g., the robust and non-fragile controllers [9], [13]. Automated verification tools have also been applied to find design's errors in discrete-time systems (e.g., UPPAAL [14], Open-Kronos [15], CPN [16], and Maellan [17]). However, there is still a gap in formal verification of embedded systems; in particular in digital controllers, which are in a continuous interaction with the enviroment.

Differently from others, this work presents a novel methodology to formally verify the occurrence of design's errors in digital controllers realizations. In particular, bounded model checking (BMC) based on the satisfiability modulo theories (SMT) is used to verify five types of properties, which include overflow, limit cycle, time constrain, stability, and minimum-phase. Additionnally, six different realization structures are considered, which include three direct forms and three delta forms. The main objective of this research is thus to demonstrate that an SMT-based BMC method can be a powerful tool in the design and verification of digital controllers, aiding the control engineer with an efficient verification tool that is more reliable and less laborious than traditional simulation tools (e.g., Matlab [18] and LabVIEW [19]), since simulation tools are hardly precise and significantly require manual intervention from designers. In particular, simulations tools either generate false alarms or neglect some failures due to the low coverage achieved during simulations.

The proposed verification methodology aims to replace the current validation process used by control engineers. In this sense, this paper describes two important contributions: it marks the first work that applies an SMT-based BMC technique to verify the occurrence of various design's errors related to FWL and quantization effects in numerical fixed-point digital controllers. Additionally, we demonstrate that the use of the delta form realization presents a higher maintenance capability of important properties (e.g., stability and minimum-phase) than the direct form realization using an appropriate FWL format, which is proved with the aid of an SMT-based BMC approach.

This paper is an extension and improvement of Bessa et al. [20]. The major difference is the verification of delta forms implementations (Bessa et al. verify digital controllers in direct forms only). Furthermore, the present article adds the minimum phase and the stability verification. We have also significantly expanded the experimental basis to demonstrate the feasibility of the proposed verification methodology.

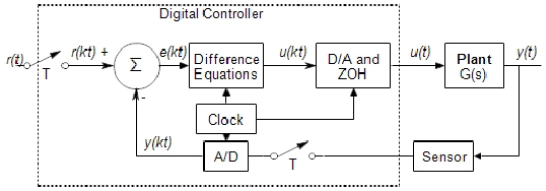

Fig. 1: Digital Control System

II. Fixed-point Digital Controllers

This section introduces the main concepts about digital controllers, including their representations and realizations; in particular, the fixed-point representation and the related problems. Finally, the delta form realization is presented and its advantage is compared to the direct form.

A. Computer-Controlled Systems

In computer-controlled systems, computer does the functions of electronics components in the classics of analog control systems, the computation of errors, and the control algorithms execution [21], [22]. The output analog signals are typically converted into digital form by the analog-to-digital converter (A/D) at pre-defined samples times. The digital computer runs the control routine and delivers to the plant a discrete control signal, which is then converted into analog signals via the digital-to-analog converter (D/A) and the zero-order hold (ZOH). Figure 1 shows a typical digital control system, where the dynamic system is mathematically represented via difference equations.

B. Fixed-Point Digital Controllers Representations and Implementations

A digital controller is a linear time-invariant causal discrete-time dynamic system. A digital controller deals with discrete numerical signals; its implementation is a program executed by a microprocessor. There are various mathematical controller representations (e.g., transfer function, states equations, and difference equations). These representations are studied in several signals books (e.g. [23]). An important representation is the difference equation. In particular, a digital controller difference equation can be described as follows:

$$y(n) = -\sum_{k=1}^{N} a_k y(n-k) + \sum_{k=0}^{M} b_k x(n-k) \qquad (1)$$

where $y(n)$ and $x(n)$ are the the output and input, respectively, in instant n [23]. Using the z-transform in Eq. (1), the digital controller can be represented via the following transfer function:

$$H(z) = \frac{b_0 + b_1 z^{-1} + ... + b_M z^{-M}}{1 + a_1 z^{-1} + ... + a_N z^{-N}} \qquad (2)$$

where z is called forward-shift operator and z^{-1} is called backward-shift operator. There are many ways to implement a digital controller in software. The realization structure of controllers influences their performance. Different realizations of digital controllers are studied in [9], [21]. In this work, however, the delta form is considered and its performance is compared to the direct form, from the formal verification perspective. Others types of realizations to implement a digital controller are explained in [21].

The direct realization uses directly the coefficients in Eq. (1) in its implementation. The advantage of this implementation is that states variables are derivations of delayed inputs and outputs. However, direct form implementations make the controller extremely sensitive to numerical errors, which are strongly evident in fixed-point implementations; as a result, it may specially harm the stability of the system. The delta form proposed by Middleton and Goodwin [4] represents a viable alternative to prevent the violation of the system's properties from the numerical errors perspective, which are typically caused by the quantizations and FWL effects.

C. Implementation of Digital Controllers Using the Delta Operator

Goodwin proposed an alternative to minimize FWL effects in fixed-point digital controller, the delta operator (δ), which is defined by [4]:

$$\delta = \frac{q-1}{\Delta} \qquad (3)$$

where q is a shift operator and δ is the Euler approximation to a derivative. A system in delta form has exactly the same behavior than a continuous sampled system with ZOH. However, this realization presents an improved round-off performance, a sensitive-less controller, more accurate coefficients representation, and a greater region of convergence so that the discrete stability domain gets close to the continuous domain.

The delta operator is equivalent to the shift operator and all the analysis done for the shift operator can be translated into the delta form. However, the delta operator is the forward-difference approximation of the differential continuous operator; therefore, the delta form is closer to the continuous behavior than the ordinary direct form, which uses the shift operator. Furthermore, the z-domain region of convergence (i.e., the region where the stability is guaranteed) is only the unitary radius circle and the δ-domain region of convergence is a circle inversely proportional to the sample period; in other words, the set of coefficients values for which a polynomial is stable may be much greater with the delta operator. In this paper, controllers' properties are compared using the delta and direct forms, from the formal verification perspective.

D. Problems Related to Fixed-point Implementation

Implementations of digital controllers are subjected to FWL effects; these effects are amplified in a fixed-point processor. The FWL effects are related to small imprecision or functional problems such as instability. The commonly error sources are round-offs and quantizations [23]. Quantization occurs during the A/D conversion and consists of approximating analog signal values from quantized (discrete) values. This process generates a rounding error, whose maximum value is 2^{-b-1}, where b is the number of bits in the fractional part.

A realistic model of a FWL system must include the quantization of every numerical value, including each arithmetic results (e.g., sums and products), input signals, and system coefficients; these are narrowly related to the system's dynamics. These accumulated errors might affect the position of the digital controller poles and zeros (mainly in direct forms) and make the controller lose the stability or the minimum phase characteristics; in the control literature, this is called controller's fragility. In the design phase, control engineers try to avoid poles and zeros positions that might be fatally affected by the FWL effects, i.e., they seek positions slightly

away from the unitary circle; sometimes it cannot be done easily. An example is the resonant controllers, whose fixed-point arithmetic effect influence is studied in [8].

Furthermore, a chosen fixed-point representation $< k, l >$, where k is the number of bits of the integer part and l is the number of bits of the fractional part, can only represent values into the range from $2^{k-1} - 2^l$ to -2^{k-1}. An overflow occurs when some operation (e.g., addition or product) returns a result outside the range of representable values. A microprocessor generally handles an overflow via wrap-around (i.e., allow the numerical representation wrapping it) or saturation (i.e., hold the maximum representation). These round-off errors or overflows might cause the appearing of periodic oscillations called limit cycles. There are some books that explain completely the fixed-point theory and operations, as in Granas and Dugundji [24]. Additionally, problems related to FWL effects on fixed-point format can also be found in [9], [23]. Understanding that the presence of these phenomenons might degrade the controller's performance, this work proposes a novel verification method that improves the design process to guarantee that a designed controller is immune to FWL effects.

III. Verification of Fixed-point Digital Controllers

This section describes concepts of formal verification; in particular, the bounded model checking technique that is used to verify digital controllers realizations. We make use of the overflow, limit cycle, and time constraints verifications as well as the direct form realizations implemented in previous work [20]. The verification of these properties are performed automatically by our verification engine, which verifies the digital controller implementation with its specification, using an exhaustive checking via non-deterministic inputs [25], [20]. We then focus our effort on the verification methods to check for stability and minimum phase properties for delta form realizations.

A. Bounded Model Checking (BMC)

The basic idea of bounded model checking (BMC) is to check for the negation of a given property at a given depth. Supposing a transition system M, a property ϕ and a bound k, BMC unrolls the system k times and translate it into a verification condition (VC) ψ, in such a way that ψ is satisfiable if and only if ϕ has a counterexample of depth less than equal to k. SMT solvers such as Z3 [26] and Boolector [27] can be used to check whether ψ is satisfiable. In BMC of digital controllers, the bound *k* limits the number of loop iterations and recursive calls in the controller's realization. BMC thus generates VCs that reflect the exact path in which a statement is executed, the context in which a given function is called, and the bit-accurate representation of expressions [28].

In this work, we use the ESBMC (Efficient SMT-Based Context-Bounded Model Checker) tool as the verification engine, since it represents one of the most efficient BMC tools that participated in the last software verification competitions [29], [30]. ESBMC is an SMT-based bounded model checker for C/C++ programs. ESBMC finds properties violations such as pointer safety, array bounds, atomicity, overflows, deadlocks, data race, and memory leaks in single- and multi-threaded software (with shared variables and locks). It also verifies programs that make use of bit-level, pointers, structs, unions, fixed-point arithmetics; it was already used in previous work to verify properties of digital filters [25] and digital controllers [20]. Inside ESBMC, the associated problem is formulated by constructing the following logical formula:

$$\Psi_k = I(S_0) \wedge \bigvee_{i=0}^{k} \bigwedge_{j=0}^{i-1} \gamma(s_j, s_{j+1}) \wedge \overline{\phi(s_1)} \qquad (4)$$

where ϕ is a property (e.g., overflow and limit cycle) and S_0 is a set of initial states of M, and $\gamma(s_j, s_{j+1})$ is the transition relation of M between time steps j and $j+1$. Hence, $I(S_0) \wedge \bigwedge_{j=0}^{i-1}$ represents the executions of a transition system M of length i. The above VC Ψ can be satisfied if and only if, for some $i \leq k$ there exists a reachable state at time step i in which ϕ is violated. If the logical formula (4) is satisfiable (i.e. returns $true$), then the SMT solver provides a satisfying assignment, from which the values of the digital controller's variables can be extracted, in order to construct a counterexample. If it is unsatisfiable (i.e., returns $false$), then we can concluded that there is no error state in k steps or less.

B. Stability Verification

The stability is a basic requirement, but very important during the digital controller's design. In particular, digital controllers are updated every sampling period and we have to ensure that the system will be stable during its execution. A discrete system is stable if all its poles are in the interior region of the unitary circle of z-plane (i.e., the poles must have the module less than one) [21].

In previous work [20], [25], stability verification using Schur Decomposition is used and implemented inside ESBMC using the Eigen Library [31]. This method, however, involves many matrix operations that makes it computationally expensive. The advantage of the Jury's algorithm can easily be observed via its complexity, which is $O(n^2)$, while the complexity of the previous stability verification, based on the Schur decomposition, is $O(n^3)$ [31].

In this paper, we choose another method to check for stability. We use the Jury's Stability Test [21], since it is computationally less expensive than the Schur Decomposition and does not require the use of an external library. Jury can be used for a given polynomial of the form:

$$F(z) = a_n z^n + a_{n-1} z^{n-1} + ...a_1 z + a_0 = 0, a_n > 0, \quad (5)$$

where a_n until a_0 represent the digital controller denominator coefficients. In particular, these coefficients are distributed in a Jury's table using the following format:

row	z^n	z^{n-1}	...	z^{n-k}	...	z^1	z^0
1	a_n	a_{n-1}	...	a_{n-k}	...	a_1	a_0
2	a_0	a_1	...	a_k	...	a_{n-1}	a_n
3	b_0	b_1	...	b_{n-k}	...	b_{n-1}	0
4	b_{n-1}	b_{n-2}	...	b_k	...	b_0	0
5	c_0	c_1	...	...	c_{n-2}	0	0
6	c_{n-2}	c_{n-3}	...	...	c_0	0	0
...	...	...	...	...	...	...	...
$2n-1$	r_0	0	0	0	0	0	0

Considering the Jury's table as a matrix m with dimensions $[2n-1][n]$ where n is the number of coefficients. We have some considerations for the algorithm:

1) The first line of matrix m has the digital controller denominator coefficients.

2) The even number lines have the inverse order of their previous lines (i.e., despising final zeros).
3) The b_0 is in line 3 of column 1 and its value is $b_0 = m[3-2][1] - (m[3-2][p]/m[3-2][1]) * m[3-1][1]$. Generalizing b_j and c_j is equal to $m[i][j] = m[i-2][j] - (m[i-2][p]/m[i-2][1]) * m[i-1][j]$, where p is the last nonzero column number for line $i-2$.
4) The line $2n-1$ has only one element nonzero in the first column.

with the Jury's table properly filled in, it is necessary to check the stability using the following definitions:

Definition 1. *If the element $m[1][1]$ is positive, then F(z) will be stable iff all the first elements in odd lines (i.e., $m[3][1]$, $m[5][1]$, ..., $m[2n-1][1]$) are positive too.*

Definition 2. *If the element $m[1][1]$ is negative, then F(z) will be stable iff the first elements in odd lines (i.e., $m[3][1]$, $m[5][1]$, ..., $m[2n-1][1]$) are negative too.*

As a running example, we check for the stability of the following digital controller extracted from our benchmarks:

$$H(z) = \frac{2.813z^2 - 0.0163z^1 - 1.872}{z^2 + 1.068z^1 + 0.1239} \quad (6)$$

it has the following Jury's table:

row	z^2	z^1	z^0
1	1.0	1.068	0.1239
2	0.1239	1.068	1.0
3	0.984649	0.935675	0
4	0.935675	0.984649	0
5	0.095512	0	0

Analyzing the Jury table above and considering the Definition 1, we concluded that the digital controller represented by Eq. (6) is stable, once $m[1][1]$, $m[3][1]$, and $m[5][1]$ are positive numbers.

C. Minimum Phase Verification

A minimum phase system is defined by a stable system with all the zeros stable. Conceptually, a minimum-phase system has all poles and zeros inside the unitary circle [21]. Minimum-phase is a desirable property in digital controllers, because in a closed-loop system, a feedback-controlled system shows the controllers zeros as the general systems poles, i.e., a digital control system with a non-minimum-phase controller is potentially unstable. The verification engine for this particular property is similar to the stability verification and also uses the Jury's stability test, but instead of using digital controller denominator coefficients, we use digital controller numerator coefficients to check for the minimal phase.

As a running example, we check whether the digital controller represented by Eq. (6) has minimum phase. In particular, it has the following Jury's table using numerator coefficients:

row	z^2	z^1	z^0
1	2.813	-0.0163	-1.872
2	-1.872	-0.0163	2.813
3	1.567218	-0.027147	0
4	-0.027147	1.567218	0
5	1.566748	0	0

Analyzing the Jury's table above and considering the Definition 1, we conclude that the digital controller represented by Eq. (6) has minimum-phase, once $m[1][1]$, $m[3][1]$, and $m[5][1]$ are positive numbers.

IV. Experimental Evaluation

The experimental evaluation of our work consists of three parts. Section IV-A describes the mathematical models of the plants that are used and summarizes the digital controllers characteristics for the respective plants. Section IV-B describes the experiments configuration, while Section IV-C analyzes and summarizes the experimental results.

A. Digital Controllers' Design

We designed 35 digital controllers for the verification of three different plants, where 18 of these controllers are designed for a commercial ball and beam plant, which has the following mathematical model:

$$G_1(z) = \frac{1.0067 \times 10^{-8}(z + 9.256)(z + 0.9324)}{(z-1)^3(z - 0.7041)} \quad (7)$$

where the sample time is $0.01s$. Others 8 controllers are designed for an A/C motor plant and described by [22]:

$$G_2(s) = \frac{1}{s(s+1)} \quad (8)$$

Finally, 9 controllers are designed for a synthetic plant [22]:

$$G_3(s) = \frac{1}{s(s+0.4)} \quad (9)$$

The controllers' FWL format is chosen via the methodology presented by Carletta et al. [10], which is based on the impulse response sum ($\sum h(k)$).

B. Experimental Setup

To statically verify the digital controller (i.e., analyze the digital controller without running it on the target platform), we consider a 16-bit microcontroller with a clock rate of 16 MHz as the hardware model. This work employs ESBMC v1.23 [28] with the SMT solver Z3 v4.0 [26]. To prevent overflows, scaling factors are considering during the implementation. All controllers are checked against five types of properties, as described in Section III. In this work, the performance of the delta form realizations are compared to the direct form realizations. As a result, all test cases are verified in six different realizations, which include: Direct Form I (DFI), Direct Form II (DFII), Transposed Direct Form II (TDFII), Delta Direct Form I (DDFI), Delta Direct Form II (DDFII), and Transposed Delta Direct Form II (TDDFII). The verification engine is invoked as follows:

```
esbmc <name> \-DTESTCASE= <i> -DFILTERTYPE= <j> \ --no-bounds-check --no-pointer-check --no-div-by-zero-check --timeout <t> --z3-bv
```

where `<name>` represents the verification type (e.g., stability and overflow), `<i>` represents the test case index, `<j>` represents the controller's realization index, and `<t>` is the maximum verification time allowed; the test cases are not determined successful or failed if a timeout occurs. The experiments run on a computer with the following hardware configurations: Intel Core i7 $-$ 2600 3.40 GHz processor, 24 GB of RAM, Ubuntu 11.10 Maverick Meerkat 64-bits OS.

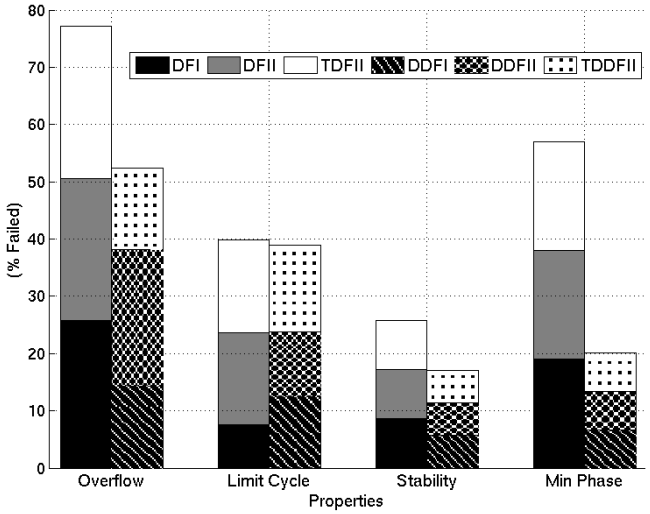

Fig. 2: Results of the digital controllers verification for the DFI, DFII, TDFII, DDFI, DDFII, and TDDFII realizations.

C. Experimental Results

Fig. 2 shows the verification results, where the horizontal axis represents the digital controllers properties that are verified; these properties are divided into 2 groups, the first one is the direct realization while the second one is the delta realization. The vertical axis represents the property violations percentage (%) found by ESBMC for each group. For each property, we executed 105 test cases running on the direct form and the delta form realizations. According to the experimental results, our verification methodology can detect several errors in direct form and in delta form realizations, but the transformation of coefficients to the delta form decreased up to 36.9% of minimum phase errors, 24.8% of overflows errors, 8.7% of stability errors, and 0.8% of limit cycle errors, where the reduced number of overflow violations is concentrated on the DFI and TDFII implementations.

In some benchmarks, our verification engine cannot find property violations given the time limits. Most timeouts occurred during the overflow or the limit cycle verifications, which are hard to be verified. A total of 17.1% of overflows verifications timed out for the direct realizations. In the limit cycle verification, the percentage goes down to 10.5% for direct form and 24.8% for delta form. Indeed, the SMT-based BMC approach represents an appropriate verification technique for digital controllers, considering that it returns results in 94.8% of the benchmarks. Note that the time constraints property is not shown in Fig. 2 since all tests returned success for direct form and delta form realizations. In particular, time constrains verifications do not present failures, because the controllers' order is relatively low and the sample time is reasonably high.

Additionally, the verification of the minimum phase, stability, and time constraints are very fast. All of these verifications do not take a time longer than two seconds since non-deterministic inputs are not used. All timeouts are concentrated on overflows and limit cycles verifications.

The implementation choice influences the verification time. In overflow verification, the direct forms are slower than the delta forms. In particular, all timeouts in the overflow verification occur in direct forms. However, the limit cycle verification of direct forms is faster than in delta forms. About 70% of timeouts in limit cycle verification occur in delta forms; in particular, limit cycles verifications present the highest verification time.

The experimental results also show that the controller order influences the verification time, which tends to be longer in high-order controllers; for this reason, the majority of timeouts are concentrated on controllers, whose order is higher than 3. About 92.7% of timeouts occur in controllers with order higher than 3. Additionally, other factors might influence the verification time, e.g., the number of bits of the fixed-point representation; the verification time tends to increase if the number of bits is higher and the size of the input vector is longer. More information about digital controller coefficients, verification results, and execution time can be found on-line. [3]

V. Related Work

The FWL effects on digital controllers are already well-known in the control systems literature. Most control systems researchers tried to prevent these problems with an additional effort in the design phase. These efforts involve the implementation of non-fragile controllers and the pursuit for the optimal FWL format. An overview about these techniques can be seen in Istepanian and Whidborne [9]. Indeed, a few related work develop tools to verify the occurrence of these problems; they have only studied how to prevent them. However, there are some particular examples of the application of formal methods to control systems monitoring and diagnostic.

In Dutertreet al. [32] is shown the advantages of formal methods over traditional debugging tools; the authors use SMT-based tools to diagnostic and monitor aircrafts systems. Simko et al. [33] demonstrate that digital controllers can be formally verified with a combination of SMT solving (to verify the control software) and Taylor models (to predict the continuous plant dynamic). Prabhu and Dasgupta [34] model check discrete controllers that only react to specific events and that can be represented via a finite-state machine. The authors use a combination of SMT solvers and existing industrial model checking tools [17].

There is an extensive use of model checking tools to verify real-time systems. An example is the UPPAAL [14], which is a model checker based on the theory of timed automata and it is designed to verify systems that are modeled via a timed automata network. This tool has a large and successful application in communication protocols verification. Another similar tool is the Open-Kronos [15], which is able to check the reachability of timed automata and the emptiness of timed Büchi automata. The CPN tools [16] are also used to verify systems modeled via a colored (timed and untimed) Petri Net.

Some recent work use SMT-based BMC to verify properties of digital filters and controllers. Cox et al. [36], [37] show that simulations tools are useful, but insufficient; the authors propose the use of SMT-based bounded and unbounded tools to verify digital filters. In Abreu et al. [25], digital filters properties are verified using ESBMC, where overflows, limit cycles, time constrains, stability, and frequency response are checked. Most recently, Bessa et al. [20] apply formal verification to check for overflows, limit cycles, and time constrains in digital controllers. Here, we present an extension of Bessa et al. [20], which marks the first use of an SMT-based BMC approach to verify FWL effects on a wide variety

[3] http://www.esbmc.org/

of digital controllers properties and realizations. In particular, the present paper expands the experimental basis of Bessa et al. to demonstrate the feasibility of the methodology in a different implementation (i.e. delta form) and with other types of properties.

VI. Conclusions

This paper describes a comparative analysis between direct form and delta form realizations for digital controllers implementation using an SMT-based BMC approach to verify properties that are hard to be checked with simulation tools. In particular, this work describes two important contributions to the embedded systems community. Firstly, the experimental results show that the delta form realizations present a superior performance if compared to the direct form realizations, reducing the occurrence of FWL related errors and preserving the stability and minimum-phase properties, which are important indicators of systems non-fragility. Secondly, this paper offers an important alternative to control systems verification via an SMT-based BMC approach. The BMC tool, used in this work, is conclusive in 94.8% of the benchmarks, showing the applicability of formal verifications to control systems; additionally, we show that BMC tools represent an automated and most reliable alternative if compared to simulation tools.

The experimental results pointed out that direct form realizations presented 40% of errors in properties verification after quantization of controllers' coefficients; however, using delta coefficients transformations, this number goes down to 25.7%, solving errors of overflows, limit cycle, stability, and minimum phase. Moreover, our verification engine returns a solution for 94.8% of the benchmarks and has 5.2% of timeouts only. We can conclude that the use of delta form realizations in digital controllers, implemented in fixed-point devices, is an adequate approach to prevent design's errors. Note that this approach does not remove all design's errors, but it decreases them substantially. In future, we plan to apply formal verification to closed-loops systems and to design requirements.

Acknowledgements. The authors thank the anonymous reviewers for their comments, which helped them to improve the draft version of this paper. This research was supported by Samsung, CNPq, CAPES, and FAPEAM grants.

References

[1] M. Masten and I. Panahi, "Digital signal processors for modern control systems," *Control Engineering Practice*, vol. 5, no. 4, pp. 449 – 458, 1997.

[2] E. Monmasson and M. Cirstea, "FPGA Design Methodology for Industrial Control Systems 2014;A Review," *IEEE TIE*, vol. 54, no. 4, pp. 1824–1842, 2007.

[3] P. Mantey, "Eigenvalue sensitivity and state-variable selection," *IEEE TAC*, vol. 13, no. 3, pp. 263–269, 1968.

[4] R. Middleton and G. Goodwin, "Improved finite word length characteristics in digital control using delta operators," *IEE TAC*, vol. 31, no. 11, pp. 1015–1021, 1986.

[5] Wu et al., "Computing a FWL stability measure for second order digital systems," in *ICARCV*, vol. 3, 2004, pp. 1593–1598 Vol. 3.

[6] G. Li, "On pole and zero sensitivity of linear systems," *IEEE TCS*, vol. 44, no. 7, pp. 583–590, 1997.

[7] ——, "On the structure of digital controllers with finite word length consideration," *IEEE TAC*, vol. 43, no. 5, pp. 689–693, 1998.

[8] L. Harnefors, "Implementation of Resonant Controllers and Filters in Fixed-Point Arithmetic," *IEEE TIE*, vol. 56, no. 4, pp. 1273–1281, 2009.

[9] R. Istepanian and J. Whidborne, *Digital Controller Implementation and Fragility: A Modern Perspective*, ser. Advances in Industrial Control. Springer London, 2001.

[10] Carletta et al., "Determining appropriate precisions for signals in fixed-point IIR filters," in *DAC*, 2003, pp. 656–661.

[11] W. Sung and K.-I. Kum, "Simulation-based word-length optimization method for fixed-point digital signal processing systems," *IEEE TSP*, vol. 43, no. 12, pp. 3087–3090, Dec 1995.

[12] V. Mohta, *Finite Worldlength Effects in Fixed-point Implementations of Linear Systems*. Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science, 1998.

[13] Wo et al., "Non-fragile controller design for discrete descriptor systems," *Journal of the Franklin Institute*, vol. 346, no. 9, pp. 914 – 922, 2009.

[14] "UPPAAL," http://www.uppaal.org, accessed: 2014-09-12.

[15] "Open-kronos," http://www-verimag.imag.fr/~tripakis/openkronos.html, accessed: 2014-09-12.

[16] "CPN tools," http://www.synopsys.com/tools/verification/functionalverification/pages/magellan.aspx, accessed: 2014-09-12.

[17] "Magellan," http://www.synopsys.com/tools/verification/functionalverification/pages/magellan.aspx, accessed: 2014-09-12.

[18] T. A. Davis and K. Sigmon, *MATLAB primer (7. ed.)*. CRC Press, 2005.

[19] G. W. Johnson, *LabVIEW Graphical Programming: Practical Applications in Instrumentation and Control*, 2nd ed. McGraw-Hill School Education Group, 1997.

[20] Bessa et al., "SMT-Based Bounded Model Checking of Fixed-Point Digital Controllers)," in *IECON (to appear)*, 2014.

[21] K. Åström and B. Wittenmark, *Computer-controlled systems: theory and design*, ser. Prentice Hall information and system sciences series. Prentice Hall, 1997.

[22] K. Ogata, *Discrete-Time Control Systems*, ser. Prentice Hall International editions. Prentice-Hall International, 1995.

[23] J. Proakis and D. Manolakis, *Digital signal processing: principles, algorithms, and applications*, ser. Prentice-Hall International editions. Prentice Hall, 1996.

[24] A. Granas and J. Dugundji, *Fixed Point Theory*, ser. Monographs in Mathematics. Springer, 2003.

[25] Abreu et al., "Verifying Fixed-Point Digital Filters using SMT-Based Bounded Model Checking," in *SBrT*, 2013.

[26] L. De Moura and N. Bjørner, "Z3: An Efficient SMT Solver," in *TACAS*, ser. TACAS'08/ETAPS'08. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 337–340.

[27] R. Brummayer and A. Biere, "Boolector: An Efficient SMT Solver for Bit-Vectors and Arrays," in *TACAS*, 2009, pp. 174–177.

[28] Cordeiro et al., "SMT-Based Bounded Model Checking for Embedded ANSI-C Software," *IEEE TSE*, vol. 38, no. 4, pp. 957–974, 2012.

[29] Morse et al., "ESBMC 1.22 - (Competition Contribution)," in *TACAS*, 2014, pp. 405–407.

[30] Morse et al., "Handling Unbounded Loops with ESBMC 1.20 - (Competition Contribution)," in *TACAS*, 2013, pp. 619–622.

[31] G. Guennebaud, "Eigen: a C++ Linear Algebra Lirary," 2011.

[32] Dutertre et al., "Formal Verification and Automated Testing for Diagnostic and Monitoring Systems," in *AIAA Guidance, Navigation and Control Conference and Exhibit*. American Institute of Aeronautics and Astronautics, 2008.

[33] G. Simko and E. K. Jackson, "A Bounded Model Checking Tool for Periodic Sample-hold Systems," in *HSCC*, ser. HSCC '14. ACM, 2014, pp. 157–162.

[34] S. Prabhu and P. Dasgupta, "Model Checking Controllers with Predicate Inputs," in *VLSID*, 2013, pp. 332–337.

[35] Anta et al., "Automatic Verification of Control System Implementations," in *EMSOFT*, ser. EMSOFT '10, 2010, pp. 9–18.

[36] Cox et al., "A Bit Too Precise? Bounded Verification of Quantized Digital Filters," in *TACAS*, 2012, pp. 33–47.

[37] Cox et al., "A bit too precise? Verification of quantized digital filters," *STTT*, vol. 16, no. 2, pp. 175–190, 2014.