Verification of Magnitude and Phase Responses in Fixed-Point Digital Filters

Daniel P. M. de Mello[1], Mauro L. de Freitas[2], Lucas C. Cordeiro[1],
Waldir S. S. Júnior[1], Iury V. de Bessa[1], Eddie B. L. Filho[1,3] and Laurent Clavier[2]
[1]Universidade Federal do Amazonas (UFAM), Manaus-AM, Brasil
[2]Univ. Lille, CNRS, ISEN, Univ. Valenciennes, UMR 8520, IEMN, 59000 Lille, France
[3]TPV Technology, Manaus-AM, Brasil
Email: dani-dmello@hotmail.com, m.lopesdefreitas@ed.univ-lille1.fr,
lucascordeiro@ufam.edu.br, waldirjr@ufam.edu.br, iurybessa@ufam.edu.br,
eddie.filho@tpv-tech.com, laurent.clavier@iemn.univ-lille1.fr

***Abstract*—In the digital signal processing (DSP) area, one of the most important tasks is digital filter design. Currently, this procedure is performed with the aid of computational tools, which generally assume filter coefficients represented with floating-point arithmetic. Nonetheless, during the implementation phase, which is often done in digital signal processors or field programmable gate arrays, the representation of the obtained coefficients can be carried out through integer or fixed-point arithmetic, which often results in unexpected behavior or even unstable filters. The present work addresses this issue and proposes a verification methodology based on the digital-system verifier (DSVerifier), with the goal of checking fixed-point digital filters w.r.t. implementation aspects. In particular, DSVerifier checks whether the number of bits used in coefficient representation will result in a filter with the same features specified during the design phase. Experimental results show that errors regarding frequency response and overflow are likely to be identified with the proposed methodology, which thus improves overall system's reliability.**

I. Introduction

Digital Filters with finite impulse response (FIR) or infinite impulse response (IIR) are used in different areas, such as digital signal processing (DSP), control systems, telecommunications, medical instrumentation, and consumer electronics. In general, such applications vary from simple frequency selection and adaptive filters to equalizers and filter banks, whose goal is to modify the characteristics of a certain signal, in accordance with pre-established requisites.

Digital filter design follows an abundant mathematical theory, both in frequency and time domains, and is usually realized with tools such as MATLAB [1], which normally assume fixed- or floating-point precision. Nonetheless, there can be a great disparity between a filter design and its practical implementation. For instance, many projects are implemented in digital signal processors or field programmable gate arrays (FPGAs), which may employ fixed-point arithmetic (with lower cost and complexity), whereas associated designs usually assume floating-point precision [15].

This difference has the potential of generating undesirable effects regarding a filter's frequency response, both in phase and magnitude, in addition to problems as overflow and instability [5]. Such behavior is due to quantization errors caused by a more constrained precision, which result in coefficients that are different from the ones originally designed. As result, one may argue about the effectiveness of digital filters and the number of bits needed for their representation, in such a way that design parameters are satisfied [16]. This paper presents a verification methodology for digital filters with fixed-point implementation, based on the Efficient SMT-Based Context-Bounded Model Checker (ESBMC), which employs Bounded Model Checking (BMC) techniques and Satisfiability Modulo Theories (SMT) [3], [4]. Such an approach indicates, according to previously defined parameters,if the chosen number of bits is sufficient and does not lead to unexpected errors or behaviors. The main advantage of this approach, over other filter analysis techniques [5], [8], is that model checking tools can provide precise information on how to reproduce errors (for instance, system input values) through counterexamples.

In order to apply the proposed methodology to digital filter verification, Digital System Verifier (DSVerifier) [17], a front-end tool for the verification of different types of digital systems, was used. DSVerifier is based on state-of-the-art bounded model checkers that support the C language and employ solvers for boolean satisfiability and satisfiability modulo theories. The present scheme was developed and integrated into DSVerifier [16], [17], because the latter was unable to support the verification of filter-specific properties, such as magnitude and phase, prior to this work. Many practical digital filters were used for verification, with the goal to validate them against real designs. As result, such a verifier, together with traditional design tools, can provide a complete digital filter synthesizing scheme, according to application conditions. Indeed, the present approach is effective in verifying magnitude and phase responses, which provides an analysis deeply based on DSP theoryThe performed experiments are based on a set of publicly available benchmarks[2].

The remainder of this work is organized as follows. Section II presents verification schemes available in literature, highlighting its main characteristics, while in Section III, the BMC technique is presented. Then, in Section IV, the proposed method is described, and Section V presents the simulations results. Finally, the conclusions are set out in Section VI.

[2]benchmarks available on http://www.esbmc.org/benchmarks/sbrt2017.zip

II. Related Work

The application of tools that implement the BMC technique, regarding software verification, is becoming quite popular, mainly due to the advent of sophisticated SMT solvers, which are constructed based on efficient satisfiability solvers (SAT) [9]. Previously published studies related to SMT-based BMC, for software, handle the problem of verifying ANSI-C programs that use bit operations, fixed-point and floating-point arithmetic, comparisons and pointers arithmetic [3]; however, there are few evidences of studies that address the verification of properties related to digital filters implementation, in ANSI-C, especially when assuming arbitrary word-length. One of such studies was previously conducted by Freitas *et al.* [2], where digital filter properties, such as overflow, magnitude and stability, were verified by employing ESBMC. Those results served as main inspiration for the present work, whose proposal is to further extend and reproduce its verified properties, while supporting them on DSVerifier. Currently, this new implementation allows passband-filter verification.

Akbarpour and Tahar [10], [11] presented an approach for error detection in digital-filter design, which is based on a high-order logic (HOL) theorem solver. The authors describe valuation functions that find the real values of a digital filter's output, through fixed- and floating-point representations, aiming to define an error. The latter represents the difference between the found values, through this valuation function, and the output corresponding to design specifications.

Recently, Cox, Sankaranarayanan and Chang [12] introduced a new approach that uses bit-precise analysis for verifying of digital filter implementations, in fixed-point. This approach is based on the BMC technique and employs SMT solvers for checking verification conditions, which are generated in the digital-filter design phase. The authors show that such an approach is more efficient and produces fewer false alarms, if compared with those that use real arithmetic solvers

Nonetheless, the mentioned studies do not address intrinsic filters characteristics, such as errors or modifications related to poles, zeros, and frequency response. In that sense, Abreu *et al.* proposed a new methodology for the verification of digital filters, employing DSVerifier [16]. The authors verified digital systems properties, such as limit-cycle and overflow, and checked output errors and time constraints, based on discrete-time models implemented in C.

The present article extends the approach proposed by Cox *et al.* [12] and Abreu *et al.* [16], by including new digital filter properties, such as magnitude and phase responses, which is closely related to DSP theory. In addition, it applies DSVerifier to the verification of a more diverse set of benchmarks, including different classes of filters (*e.g.*, passband filters).

One may also find a fixed-point implementation of a digital filter by directly designing it and optimizing the number of bits used in the fixed-point format [6], [7]; however, traditional design methodologies do not address fragility issues, *i.e.*, the filter dynamics sensitivity w.r.t. the implementation issues, *e.g.*, FWL effects and round-offs. Even a correct filter-design might lose performance and stability due to those. The method proposed here addresses this problem, since it is able to represent all dynamic and rounding variations as non-deterministic variables, which will produce a (transition) system that models all the possible states a system could reach.

III. The BMC technique

With ESBMC, a program under analysis is modeled by a state transition system, which is generated from a program control-flow graph (CFG) [13] that is automatically created during verification. A node in a CFG represents an assignment (deterministic or nondeterministic) or a conditional expression, while an edge represents a change in a program's flow.

A state transition system $M = (S, T, S_0)$ is an abstract machine, which consists in a state set S, where $S_0 \subseteq S$ represents a initial state set and $T \subseteq S \times S$ is the transition relation. A state $s \in S$ consists of values of a program counter pc and all system variables. An initial state s_0 assigns a program's initial location in a CFG to the pc. We identify each transition $\gamma(s_i, s_{i+1}) \in T$ between two states s_i and s_{i+1}, with a logical formula $\gamma(s_i, s_{i+1})$, which captures the constraints on corresponding values of a program counter and system variables. Given a transition system M, a property ϕ, and a bound k, ESBMC [4] unfolds a system k times and transforms the associated result into a verification condition ψ, in such a way that ψ is satisfiable if and only if ϕ contains a counterexample with length smaller or equal to k [3]. Thus, the BMC technique problem is formulated as follows

$$\psi_k = I(s_0) \wedge \bigvee_{i=o}^{k} \bigwedge_{j=0}^{i-1} (\gamma(s_j, s_{j+1}) \wedge \phi(s_i)), \quad (1)$$

where ϕ is a property, I is a set of the initial states in M, and $\gamma(s_j, s_{j+1})$ is the state transition function of M, between steps j and $j+1$. Thus, $I(s_0) \wedge \bigwedge_{j=0}^{i-1} \gamma(s_j, s_{j+1})$ represents the execution of M for i times. Eq. (1) will be satisfied if, and only if, for each $i \leq k$, there is a reachable state where ϕ is violated. If Eq. (1) is satisfiable, then ESBMC shows a counterexample, defining which variable values are needed to lead to the related error. The counterexample for a property ϕ is a state sequence $s_0, s_1, ..., s_k$ with $s_0 \in S_0$ and $\gamma(s_i, s_{i+1})$, for $0 \leq i < k$. If Eq. (1) is unsatisfiable, one can conclude that no error state is reachable within k steps or less.

IV. The New Verification Methodology

In general, fixed-point implementations use standard registers to store the inputs and outputs along adders, multipliers, and delays; however, results of those elements might exceed the limits of the allocated variables or generate values different than what was expected, due to coefficients accuracy or associated number of bits. In conclusion, it is possible that results differ from those specified or even that a filter becomes unstable. Finally, the proposed verification methodology is split into three main parts: magnitude and phase verification, poles and zeros stability, and overflow verification.

A. Magnitude and phase verification

Changes in coefficients, due to fixed-point quantization, modify responses in magnitude and phase [11] as shown in Fig. 1.

Here, the input of the proposed verification system is composed by filter coefficients, in floating-point, which must be analyzed according to the adopted conditions, such as passband, cut-off frequency, and rejection band, as well as the gains in each region and the amount of bits used for its representation in fixed-point.

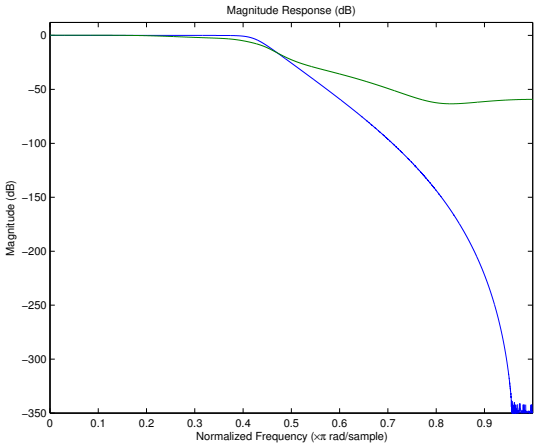

Fig. 1. Magnitude response of an IIR Chebyshev filter with order 12: the blue curve represents the projected answer and the green one the response with fixed-point, containing one sign bit and 7 and 6 bits for the integer and fractional parts, respectively.

Given that N is the number of points of the Discrete-Time Fourier Transform (DTFT) [14], $h[n]$ is the filter impulse response, and H_k is the k-th component of it's sampled equivalent in frequency domain, we have that

$$H_k = \sum_{n=0}^{N-1} h(n) e^{-j(2\pi/N)kn}. \tag{2}$$

In addition, suppose that ω_p, ω_r, and ω_c are the digital frequencies of passband, stopband and cutoff, respectively, and A_p, A_r, and A_c are the gains that will be checked. We assumed the following assertions to verify magnitude and phase properties for lowpass and highpass filters:

$$\begin{aligned} I_{lp\_mag} \Leftrightarrow & \left[\Big(|H_k| > A_p\Big) \wedge \Big(0 \leq \tfrac{2\pi k}{N} \leq \omega_p\Big)\right] \vee \\ & \left[\Big(|H_k| < A_c\Big) \wedge \Big(\tfrac{2\pi k}{N} = \omega_c\Big)\right] \vee \\ & \left[\Big(|H_k| < A_r\Big) \wedge \Big(\omega_r \leq \tfrac{2\pi k}{N} \leq \pi\Big)\right], \end{aligned} \tag{3}$$

$$\begin{aligned} I_{hp\_mag} \Leftrightarrow & \left[\Big(|H_k| < A_r\Big) \wedge \Big(0 \leq \tfrac{2\pi k}{N} \leq \omega_r\Big)\right] \vee \\ & \left[\Big(|H_k| > A_c\Big) \wedge \Big(\tfrac{2\pi k}{N} = \omega_c\Big)\right] \vee \\ & \left[\Big(|H_k| > A_p\Big) \wedge \Big(\omega_p \leq \tfrac{2\pi k}{N} \leq \pi\Big)\right] \end{aligned} \tag{4}$$

and

$$\begin{matrix} I_{lp\_phase}, \\ I_{hp\_phase} \end{matrix} \Leftrightarrow |\sphericalangle H(k) - \sphericalangle H_{fixed}(k)| > threshold. \tag{5}$$

In case of an assertion violation, an error is generated to indicate that the chosen amount of bits is insufficient, taking into account the initial design restrictions.

B. Poles and zeros verification

the Jury's algorithm is used to check stability in the z-domain, for a given characteristic polynomial of the form

$$S(z) = a_0 z^N + a_1 z^{N-1} + ...a_{N-1} z + a_N = 0, a_0 \neq 0. \tag{6}$$

In particular, the Jury's stability test is already explained in the control system literature [19]. This study, however, limits itself to explain the SMT encoding of the Jury's criteria. For the stability test procedure, the following Jury's matrix $M = [m_{ij}]_{(2N-2)\times N}$ is built from $S(z)$ coefficients:

$$M = \begin{bmatrix} V^{(0)} \\ V^{(1)} \\ \vdots \\ V^{(N-2)} \end{bmatrix}, \tag{7}$$

where $V^{(k)} = [v_{ij}^{(k)}]_{2\times N}$, such that

$$v_{ij}^{(0)} = \begin{cases} a_{j-1}, & \text{if } i = 1 \\ v_{(1)(N-j+1)}^{(0)}, & \text{if } i = 2 \end{cases} \text{ and} \tag{8}$$

$$v_{ij}^{(k)} = \begin{cases} 0, & \text{if } j > n-k \\ v_{1j}^{(k-1)} - v_{2j}^{(k-1)} \cdot \frac{v_{11}^{(k-1)}}{v_{21}^{(k-1)}}, & \text{if } j \leq n-k \text{ and } i = 1 \\ v_{(1)(N-j+1)}^{(k)}, & \text{if } j \leq n-k \text{ and } i = 2 \end{cases}, \tag{9}$$

where $k \in \mathbb{Z}$, such that $0 < k < N-2$. $S(z)$ is the characteristic polynomial of a stable system, if and only if the following four propositions hold:

- R_1: $S(1) > 0$;
- R_2: $(-1)^N S(-1) > 0$;
- R_3: $|a_0| < a_N$;
- R_4: $m_{11} > 0 \iff m_{31} \wedge m_{51} \wedge \cdots \wedge m_{(2N-3)(1)}$.

The stability property is then encoded by creating a constraint using the fixed size bit-vector theory, which is typically supported by state-of-the-art SMT solvers [18]:

$$\phi_{stability} \iff (R_1 \wedge R_2 \wedge R_3 \wedge R_4), \tag{10}$$

where the literal $\phi_{stability}$ represents the validity of the stability condition. In particular, an SMT-solver checks whether the Jury's criteria hold for characteristic polynomial coefficients.

C. Overflow verification

The third part tackles overflow verification after coefficient quantization, which would be considered infeasible without computational tools. Additions, subtractions, multiplications, and divisions can be approximately implemented with fixed-point representations, so that values are constrained according to the available number of bits; if this condition is violated, then an overflow has occurred. In order to better understand that, our work describes overflow by saturation and overflow by wrap-around.

Definition 1. ***(Saturation)*** *Saturation occurs when values outside a bit range are represented by minimum or maximum values.*

Despite the easy in finding those limits, it would be difficult to know which input will lead to saturation.

Definition 2. ***(Wrap-around)*** *Wrapping around consists in attributing minimum values when the maximum limit is reached and vice-versa [12].*

Overflow verification generates an array of nondeterministic fixed-point numbers $\tilde{x}[n]$, which represents a discrete input signal and applies it to a filter's difference equation $h[n]$, which was converted to fixed-point. The result is an array describing the output $y[n]$ at each step n of an input signal. The length N of an input array, as well as the maximum and minimum values of each input element are defined by users. The internal realization of $h[n]$ might also be selected by users. All overflow check iterations are given by

$$I_{overflow} \Leftrightarrow \Big(y[n] \geqslant V_{min}\Big) \wedge \Big(y[n] \leqslant V_{max}\Big), \tag{11}$$

where n varies from 0 to N and V_{min} and V_{max} are the minimum and maximum values representable by a given fixed-point bit format. The procedure used by DSVerifier to calculate this interval is described in section V-A.

In oder to detect an error, a counterexample is generated, which consists of the violated states, providing access to inputs that generated the associated error, in a specific order, as well as output values. This approach provides knowledge about error conditions regarding overflow, allowing a designer to look for an alternative implementation. Until the present moment, DSVerifier does not support underflow verification.

V. Experiments

This section consists in two parts. The adopted system configuration is described in Subsection V-A, while Subsection V-B summarizes our goals and Subsection V-C describes the results obtained with DSVerifier[1] [3], [4], [16], [17], regarding filter magnitude and phase responses, as well as preexisting functions for verifying stability and overflow.

A. System configuration and preparation for experimentation

Magnitude and phase verifications were split into two main groups: one consisting of IIR and another of FIR filters. Each set is divided into 3 categories: lowpass, highpass, and bandpass, with three filters with low orders (2nd or 4th order), and three with high orders (12th or 30th order), in each set, with different cut-off frequencies. Three types of IIR filters were used: Butterworth, Chebyshev, and Elliptic. For FIR filters, types Equiripple, Hann Window, and Maximally Flat.

Altogether, 54 stable filters were created during the design stage, with 18 FIR and 36 IIR, with a sample frequency of $48kHz$ (commonly used in audio applications). All filter transfer-functions were obtained with the Filter Design and Analysis Tool application available in MATLAB [1] and then encoded in a ".c" file.

Aiming to explore different theories employed by SMT [3] solvers, non-integer numbers were encoded into two different ways: in binary (when bit vector arithmetics is used) and also in real representations (when using rational arithmetic). Fixed-point representation was performed by dividing the number to be represented between its integer part I, with m bits, and its fractional one F, with n bits [15]. such an approach is represented with tuple $\langle I, F\rangle$, which can be encoded both in bit vector and rational arithmetic and is interpreted as $I + F/2^n$. Thus, all associated represented values must be between the maximum and minimum expected ones:

$$V_{max} = 2^m - 1/2^n \tag{12}$$

and

$$V_{min} = -2^m, \tag{13}$$

with

$$V_{min} \leq v_{fixed} \leq V_{max}. \tag{14}$$

All experiments were conducted with an Intel Core i7-2600 PC, with $3.40GHz$ of clock speed and $16GB$ RAM, running 64-bits Ubuntu as operational system. The execution duration of each verification was obtained with the Unix's *time* command.

[1] http://www.dsverifier.org

B. Experimental Goals

While creating our benchmark, we aimed at a variety of filter parameters, such as order, frequency, and type, in such a way that we were able to demonstrate the usefulness of our method, in all sorts of situations. Some filters were defined with an extremely short passband-interval, so that DSVerifier was taken to the limit when trying to represent unrealistically challenging situations.

C. Results

Table I summarizes verification results regarding phase and magnitude, with the first letter in the identification name indicating which filters are FIR and which ones are IIR. Filters that contain "hp" are high-pass, while "lp" stands for low-pass. The filter order is displayed as the number in each filter identification name. Columns "CF", "PF" and "SF" indicate, respectively, the cut-frequency, the pass-frequency, and the stop-frequency, in kHz, which were used when synthesizing filters. Some of those frequencies were not used (indicated by "NE") in design specifications of certain filter types. "VTM" stands for magnitude verification time, while "SM" stands for magnitude verification status, which can be either Successful (S), Passband fail (FP), Stopband fail (FS), or Cutoff-frequency fail (FC). SP is the status for phase verification. The phase status can be either Successful (S) or Fail (F).

FP represents the number of bits used for fixed-point representation. The minimum gain, for passbands, is fixed in $-1dB$ for Eliptic filters, while the maximum gain for stopbands is fixed in $-80dB$, for Eliptical and Chebychev filters. Those specifications apply for both low-pass and high-pass filters. Near identical criteria were used for band-pass filters, except that now a pair of each region frequencies is needed for full specification. All filters in Table V-C were verified considering a fixed-representation tuple of $\langle 4, 10\rangle$, except for second order filters, where $\langle 1, 5\rangle$ was used.

One may notice that verification procedures of low- and high-pass filters with the same order occurred with largely different elapsed times. That occurs because failures on the frequency response might occur sooner for one filter and later for another, regardless of order. Thus, DSVerifier (together with an SMT solver) takes different amounts of time to find a violation and provide the respective counterexample that leads to that.

Results for magnitude verification regarding band-pass filters are included in Table II. It is worth noticing that instead of a single frequency as specification, now a frequency pair is used. The column "FP tuple" indicates the considered FWL constriction. Poles and zeros verification procedures occurred for IIR filters and the result is shown in Table III. "VT" stands for verification time, in seconds, and "SPZ" represents the status of a verification. Besides, results for filters with a cutoff-frequency of $100Hz$ are interesting, since magnitude failures were found for all IIR ones. Table IV presents results for overflow verification, which indicates the efficacy regarding detection of this type of error. All tested filters were IIR, which were also used in Table I.

VI. Conclusion

The present work proposed a methodology for verifying digital filter design parameters through the BMC technique, which

TABLE I
MAGNITUDE AND PHASE VERIFICATION OF IIR AND FIR FILTERS.

IIR filters	CF	PF	SF	TVM (s)	SM	SP
ilp2	9.6	Na	Na	5.25	S	S
ihp2	9.6	Na	Na	5.39	S	S
ilp2EST	0.1	Na	Na	207.34	FP	F
ilp12	9.6	Na	Na	17.07	S	F
ihp12	9.6	Na	Na	17.15	S	F
ilp12EST	0.1	Na	Na	423.56	FP	F
ilp4C	Na	9.6	Na	173.44	FP	F
ihp4C	Na	9.6	Na	162.24	FS	S
ilp4ESTC	Na	0.1	Na	163.70	FS	F
ilp12C	Na	9.6	Na	430.85	FS	F
ihp12C	Na	9.6	Na	432.66	FS	S
ilp12ESTC	Na	0.1	Na	523.98	FP	F
ilp4E	Na	Na	9.6	161.44	FP	S
ihp4E	Na	Na	9.6	164.64	FP	F
ilp4ESTE	Na	Na	0.1	164.68	FS	F
ilp12E	Na	Na	9.6	460.18	FP	F
ihp12E	Na	Na	9.6	429.14	FP	F
ilp12ESTE	Na	Na	0.1	455.35	FP	F
fhp10	7.2	Na	9.6	14.63	S	S
flp10	9.6	Na	9.6	1902.8	FC	S
fhp30	9.6	Na	9.6	1164.9	S	S
flp30	Na	Na	9.6	41.02	S	S
flp10EST	0.1	Na	0.1	14.95	S	F
flp30EST	Na	Na	0.1	41.91	S	S
fhp10Equi	9.6	Na	9.6	360.9	FS	S
flp10Equi	9.6	Na	9.6	70.8	FP	S
fhp30Equi	9.6	Na	9.6	1043.2	FP	S
flp30Equi	Na	Na	9.6	1061.3	FS	F
flp10ESTEqui	0.1	Na	0.1	363.59	FP	F
flp30ESTEqui	Na	Na	0.1	1052.6	FP	S
fhp10Hann	9.6	Na	9.6	361.84	FC	F
flp10Hann	Na	Na	9.6	15.42	S	S
fhp30Hann	9.6	Na	9.6	41.91	S	S
flp30Hann	9.6	Na	9.6	40.25	S	F
flp10ESTHann	Na	Na	0.1	15.36	S	F
flp30ESTHann	Na	Na	0.1	41.22	S	S

TABLE II
MAGNITUDE VERIFICATION OF IIR PASSBAND FILTERS.

IIR filters	CF Pairs	PF Pairs	SF Pairs	TVM (s)	SM	FP tuple
ipb2	7.2, 16.8	Na	Na	106.46	FP	<1,5>
ipb12	7.2, 16.8	Na	Na	17.20	S	<4,10>
ipb12EST	7.2, 7.32	Na	Na	349.95	FP	<4,10>
ipb4E	Na	7.2, 16.8	Na	160.24	FP	<4,10>
ipb12E	Na	7.2, 16.8	Na	447.75	FP	<4,10>
ipb12ESTE	Na	7.2, 7.32	Na	438.95	FS	<4,10>
ipb4C	Na	Na	7.2, 16.8	140.27	FP	<4,10>
ipb12C	Na	Na	7.2, 16.8	564.95	FS	<10,16>
ipb12ESTC	Na	Na	7.2, 7.92	445.73	FS	<10,16>

TABLE III
STABILITY VERIFICATION OF IIR FILTERS.

IIR filters	O	FC (Hz)	VT (s)	SPZ	FP tuple
hp12	12	9600	1.35	S	<4,10>
hp12C	12	9600	1.35	S	<4,10>
hp12E	12	9600	79.99	F	<4,10>
hp2	2	9600	1.20	S	<1,5>
hp4E	4	9600	1.52	S	<4,10>
lp12	12	9600	1.35	S	<4,10>
lp12C	12	9600	1.30	S	<4,10>
lp12E	12	9600	79.83	F	<4,10>
lp12ESTC	12	100	1.88	F	<4,10>
lp12ESTE	12	100	79.94	F	<4,10>
lp2	2	9600	1.16	S	<1,5>
lp4E	4	9600	1.22	S	<4,10>

TABLE IV
OVERFLOW VERIFICATION OF IIR FILTERS.

Filter	Order	VT (s)	FP tuple	Status
lp2	2	77.184	<1,5>	S
hp2	2	90.034	<1,5>	F
lp4E	4	185.945	<1,5>	F
hp4E	4	184.254	<1,5>	F

indicates if the number of bits used for coefficient and sample representation changes previously specified characteristics.

During the present simulations, both IIR and FIR filters were addressed, in order to ensure the outcome of real projects, in different realizations and applications. Results show that it is possible to detect low and medium order filters, with moderate verification times. The main contributions of this work are the incorporation of parameters related to filter design theories, such as magnitude and phase responses and the location of poles and zeros, which complement other tests, such as overflow. Besides support for intrinsic filter properties verification was added to DSVerifier tool. For future work, it would be of interest to incorporate other filter parameters and perform an automatic computation of the minimum number of bits for a successful validation, for fixed-point.

REFERENCES

[1] Mathworks, Inc., "Matlab Getting Started Guide R2011b", USA, 2011.

[2] M. de Freitas, M. Gadelha, W. da Silva Jr., E. de Lima Filho, "Verificação de Propriedades de Filtros Digitais Implementados com Aritmética de Ponto Fixo", *XXXI Brazilian Telecommunications Symposium*, pp. 1-4, 2013.

[3] L. Cordeiro, B. Fischer e J. M. Silva, "SMT-Based Bounded Model Checking for Embedded ANSI-C Software", *IEEE Transactions on Software Engineering (TSE)*, v. 38, n.4, pp. 957–974, 2012.

[4] L. Cordeiro e B. Fischer, "Verifying Multi-threaded Software using SMT-based Context-Bounded Model Checking". *in Proc. Int. Conf. on Software Engineering (ICSE)*, pp. 331–340, IEEE/ACM, 2011.

[5] A. Oppenheim, R. Schafer e J. Buck, "Discrete-time signal processing", *Prentice-Hall, Inc.*, Ed. 2, 1999.

[6] Sung W, Kum KI (1995) Simulation-based word-length optimization method for fixed- point digital signal processing systems. IEEE T SIGNAL PROCES 43(12):30873090. DOI 10.1109/78.476465

[7] Carletta J, Veillette R, Krach F, Fang Z (2003) Determining appropriate precisions for signals in fixed-point IIR filters. In: Proceedings of Design Automation Conference, pp. 656661. DOI 10.1109/DAC.2003.1219100

[8] W. T. Padgett and D. V. Anderson, "Fixed-point signal processing", Synthesis Lectures on Signal Processing, vol. 4, no. 1, pp. 1–133, 2009.

[9] L. de Moura e N. Bjrner, "Z3: An efficient SMT solver", *in Proc. Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, LNCS 4963, pp. 337–340, 2008.

[10] B. Akbarpour e S. Tahar: "Error Analysis of Digital Filters Using Theorem Proving". *in Proc. Int. Conf. on Theorem Proving in Higher Order Logics (TPHOLs)*, pp. 1–17, 2004.

[11] B. Akbarpour e S. Tahar. "Error analysis of digital filters using HOL theorem proving". *Journal Applied Logic*, v.5, n. 4, pp. 651–666, 2007.

[12] A. Cox, S. Sankaranarayanan, e Bor-Yuh E. Chang, "A bit too precise? Bounded verification of quantized digital filters", *in Proc. Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, pp. 33–47, 2012.

[13] S. S. Muchnick, "Advanced compiler design and implementation.", *Morgan Kaufmann Publishers Inc.*, 1997.

[14] P. Diniz, E. Silva e S. Netto, "Processamento Digital de Sinais - Projeto e Anlise de Sistemas", *Bookman Editora*, 2002.

[15] I. Bessa, H. Ismail, L. Cordeiro, J. Chaves Filho, "Verification of fixed-point digital controllers using direct and delta forms realizations". *Design Autom. for Emb. Sys.* 20(2): 95-126 (2016).

[16] R. Abreu, M. Gadelha, L. Cordeiro, E. de Lima Filho, W. da Silva Jr.: Bounded model checking for fixed-point digital filters. Journal of the *Brazilian Computer Society*. v. 22, n. 1, pp. 1–20, May 2016.

[17] H. Ismail, I. Bessa, L. Cordeiro, E. de Lima Filho, J. Chaves Filho: DSVerifier: A Bounded Model Checking Tool for Digital Systems. *in Proc. Int. Conf. on Symposium on Model Checking of Software (SPIN)*: pp. 126-131, 2015.

[18] L. Keel and S. Bhattacharyya, Stability margins and digital implementation of controllers, in *Proc. Amer. Control Conf.*, vol. 5, 1998, pp. 28522856.

[19] I. Bessa, H. Ismail, R. Palhares, L. Cordeiro, and J.Chaves Filho "Non-Fragile Stability Verification of Digital Control Systems with Uncertainty", in *IEEE TRANSACTIONS ON COMPUTERS*, VOL. 66, NO. 3, 2017.