UNIVERSIDADE FEDERAL DO AMAZONAS
PRÓ-REITORIA DE PESQUISA E PÓS-GRADUAÇÃO
DEPARAMENTO DE APOIO A PESQUISA
PROGRAMA INSTITUCIONAL DE INICIAÇÃO CIENTÍFICA

VERIFICAÇÃO DE CONTROLADORES DIGITAIS DE PONTO FIXO REPRESENTADOS POR ESPAÇO DE ESTADOS

Bolsista: Felipe Rodrigues Monteiro Sousa, CNPq

MANAUS – AM
2016

RELATÓRIO FINAL PIBIC
PIB-E/0046/2015
VERIFICAÇÃO DE CONTROLADORES DIGITAIS DE PONTO FIXO REPRESENTADOS POR ESPAÇO DE ESTADOS

UNIVERSIDADE FEDERAL DO AMAZONAS
PRÓ-REITORIA DE PESQUISA E PÓS-GRADUAÇÃO
DEPARAMENTO DE APOIO A PESQUISA
PROGRAMA INSTITUCIONAL DE INICIAÇÃO CIENTÍFICA

RELATÓRIO FINAL PIBIC
PIB-E/0046/2015

VERIFICAÇÃO DE CONTROLADORES DIGITAIS DE PONTO FIXO REPRESENTADOS POR ESPAÇO DE ESTADOS

Bolsista: Felipe Rodrigues Monteiro Sousa, CNPq
Orientador: Prof. Dr. Lucas Carvalho Cordeiro

MANAUS – AM
2016

Esta pesquisa, financiada pelo Conselho Nacional de Pesquisa – CNPq, através do Programa Institucional de Bolsas de Iniciação Científica da Universidade Federal do Amazonas, foi desenvolvida pelo Núcleo de Estudo e Pesquisa em Ciência da Informação e se caracteriza como subprojeto do projeto de pesquisa Bibliotecas Digitais.

"Os cientistas estudam o mundo como ele é,
os engenheiros criam um mundo como ele nunca havia sido".

Theodore von Karman.

RESUMO

Em sistemas digitais de controle, mais conhecidos como controladores digitais, um computador é responsável pelas funções dos componentes eletrônicos, a computação de erros, bem como a execução de algoritmos de controle. Tais sistemas são amplamente utilizados pela comunidade de engenharia de controle e automação, devido às várias vantagens sobre os controladores analógicos, tais como a melhoria da confiabilidade, sensibilidade, flexibilidade e custo. No entanto, existem algumas desvantagens no uso de controladores digitais, como por exemplo, os erros que são introduzidos durante o processo de quantização. Neste contexto, existem algumas iniciativas para resolver os problemas que surgem no domínio de tempo discreto e, em particular, os problemas relacionados com o comprimento finito da palavra. Logo, é importante uma melhor compreensão dos problemas típicos relacionados aos controladores digitais, de modo que possamos propor alternativas mais eficientes para que a quantização e os efeitos do comprimento da palavra sejam, potencialmente, reduzidos durante o projeto de tais sistemas. Tendo isso em mente, este trabalho visa descrever uma metodologia de verificação, apoiada pela ferramenta denominada *Digital System Verifier* (DSVerifier), a qual é capaz de verificar propriedades relacionadas ao *overflow* aritmético, ciclo-limite, restrições temporais, estabilidade e fase mínima na implementação de controladores digitais representados por funções de transferência. Particularmente, este projeto visa ampliar a automação do processo de verificação com o DSVerifier através do suporte à verificação de controladores digitais representados em espaço de estado. Deste modo, engenheiros de controle poderão verificar através da ferramenta que sistemas digitais de controle apresentam o desempenho desejado, quando ele é incorporado em um determinado hardware com certas limitações de recursos.

Palavras-chaves: Controladores Digitais; Verificação Formal; Espaço de Estados; *Model Checking*;

ABSTRACT

In digital control systems, which are well known as digital controllers, a computer is responsible for the functionality of the electronic components and error calculations, as well as the execution of control algorithms. Such systems are widely used by the control and automation engineering community, due to its many advantages over analog control systems, such as reliability, sensibility, flexibility, and lower costs. However, there are some disadvantages in digital controllers as well, for instance, the arithmetic errors produced by the quantization processes. In this context, there are a few initiatives to solve the problems that emerge in the discrete-time domain and, particularly, the problems related to the finite word-length (FWL) effects. Thus, it is important to have a better understanding of the typical problems related to digital controllers, in order to propose more efficient alternatives to substantially reduce the impact of quantization processes and FWL effects in such systems. Keeping this in mind, this work proposes a verification methodology, supported by the Digital System Verifier (DSVerifier) tool, which is able to verify properties related to arithmetic overflow, limit cycle, temporal restrictions, stability, and minimum phase in the implementation of digital controllers represented by transfer functions. In particular, this project aims to extend the verification power of DSVerifier through the support of digital systems represented by state-space equations. Thus, control engineers will be able to verify through DSVerifier whether a certain digital control system follows its specifications, when it is embedded into certain hardware with limited resources.

Keywords: Digital Controllers; Formal Verification; State-Space; Model checking;

LISTA DE SIGLAS

ESBMC – *Efficient SMT- Based Context-Bounded Model Checker*.

SV-COMP – Competição Internacional de Verificação.

DSVerifier – *Digital System Verifier.*

FPGA – *Field-ProgrammableGgate Array.*

BMC – *Bounded Model Checking*.

SMT – Teorias do Módulo da Satisfação.

LTL – Lógica Linear Temporal.

CTL – *Computation Tree Logic*.

CBMC – *C Bounded Model Checker*.

SISO – *Single-Input Single-Output*.

MIMO – *Multi-Input Multi-Output*.

LISTA DE FIGURAS

SUMÁRIO

INTRODUÇÃO

Em sistemas controlados por computador, o computador faz as funções dos componentes eletrônicos nos clássicos sistemas de controle analógico, a computação de erros, bem como a execução de algoritmos de controle [1, 2]. Os sinais analógicos de saída são tipicamente convertidos para a forma digital pelo conversor analógico-digital (A/D) em uma amostra pré-definida. O computador digital executa a rotina de controle e entrega um sinal de controle discreto à planta, que é então convertido em sinais analógicos, por meio de um conversor digital-analógico (D/A) e do (*zero-order hold* - ZOH).

Parte fundamental de tais sistemas é o controlador digital, um sistema dinâmico de tempo discreto, linear, causal e invariante no tempo. Um controlador digital lida com sinais numéricos discretos; a sua implementação é um programa executado por um microprocessador. Existem várias representações matemáticas para um controlador (e.g., a função de transferência, as equações de estado e as equações diferenciais). Essas representações são estudadas em vários livros de sinais (e.g., [3]). É importante ressaltar também a representação de espaço de estado, a qual é abordada neste projeto e é amplamente utilizada nas aplicações reais.

Além disso, controladores digitais são amplamente utilizados pela comunidade de engenharia de controle devido as várias vantagens sobre os controladores analógicos, tais como a melhoria da confiabilidade, sensibilidade, flexibilidade e custo. No entanto, existem algumas desvantagens no uso de controladores digitais, como por exemplo, os erros que são introduzidos durante o processo de quantização. Neste contexto, existem algumas iniciativas para resolver os problemas que aparecem no domínio de tempo discreto, em particular, os problemas relacionados com o comprimento finito da palavra [4, 5].

REVISÃO BIBLIOGRÁFICA

Controladores digitais são geralmente aplicados em microcomputadores, microprocessadores, processadores de sinais digitais [6], e *field-programmable gate arrays* (FPGAs) [7]. De acordo com a escolha do hardware, o formato e aritmética usados para representar e manipular números podem mudar (por exemplo, número de bits fixo ou aritmética de ponto flutuante); essas representações influenciam diretamente na precisão e desempenho do controlador digital.

No caso do processador de ponto flutuante, existe um número maior de valores representáveis e, consequentemente, uma precisão reforçada; no entanto, o processador de ponto fixo é a solução mais rápida e mais barata que, por sua vez, é amplamente utilizado na prática. Portanto, o cenário mencionado requer uma melhor compreensão e um melhor tratamento dos problemas típicos relacionados aos controladores digitais para que a quantização e os efeitos do comprimento da palavra sejam, potencialmente, reduzidos durante o projeto do controlador digital. Vários fatores podem influenciar, intensificar ou atenuar esses efeitos (por exemplo, através da utilização de estruturas de realização, tais como formas diretas e delta, bem como a definição do número de bits e da taxa de amostragem). As possíveis influências de tais efeitos trazem uma importante estabilidade e sensibilidade aos autovalores para o controlador digital, que são previamente investigados por vários autores [8-11].

De forma a validar o processo de desenvolvimento de um controlador digital, usamos a ferramenta denominada ESBMC (*Efficient SMT- Based Context-Bounded Model Checker*) como mecanismo de verificação, uma vez que a mesma representa uma das ferramentas de *bounded model checking* (BMC) com notória participação nas últimas competições de verificação de software [12, 13]. De modo particular, o ESBMC é a ferramenta mais eficiente para analisar

programas que fazem uso de aritmética de vetores de *bits* de acordo com a edição de 2015 da Competição Internacional de Verificação de Software (SV-COMP). O ESBMC é um verificador de modelos de contexto limitado para códigos embarcados ANSI-C/C++. Ele é capaz de encontrar violações de propriedades, tais como segurança ponteiro, limites de *arrays*, atomicidade, overflows, *deadlocks*, corrida de dados e vazamento de memória em software simples e *multi-threaded*. Vale ressaltar que ele já foi utilizado em trabalhos anteriores para verificar propriedades de filtros digitais [14] e controladores digitais [15].

Em particular, a metodologia de verificação apresentada neste projeto é suportada pelo software *Digital System Verifier* (DSVerifier) [16], o que é uma extensão do ESBMC. Tal ferramenta é capaz de verificar propriedades relacionadas ao overflow, ciclo-limite, restrições temporais, estabilidade e fase mínima na implementação de controladores digitais. Através do DSVerifier, os engenheiros de controle podem verificar que o controlador digital concebido apresenta o desempenho desejado, quando ele é incorporado em um determinado hardware com certas limitações de recursos. Além disso, este projeto visa ampliar a automação do processo de verificação com o DSVerifier através do suporte a verificação de controladores digitais representados em espaço de estado.

METODOLOGIA

Este trabalho de pesquisa tem como objetivo ampliar a automação do processo de verificação de controladores digitais de ponto-fixo, combinando o suporte à representação de espaço de estado à técnica de verificação formal utilizando um método BMC baseado em teorias do módulo de satisfação (SMT). Nesta pesquisa, verifica-se de forma automática controladores digitais de ponto-fixo reais utilizando uma ferramenta denominada de DSVerifier, a qual foi desenvolvida utilizando a linguagem de programação ANSI-C. Particularmente, foi desenvolvido um novo módulo para a respectiva ferramenta, de tal forma que fosse possível realizar este tipo de verificação.

A metodologia empregada nesta pesquisa pode ser dividia em três principais etapas. Primeiramente, uma revisão da literatura a respeito da teoria de Verificação de Modelos foi realizada. Nesta etapa, a maioria dos conceitos importantes acerca de Lógica Proposicional, Lógica Temporal Linear (LTL) e Lógica de Árvore de Computação (CTL) foram estudados com o intuito de entender o funcionamento dos verificadores ESBMC [19], CBMC [28] e DSVerifier. Esta parte comtempla também um estudo aprofundado sobre os controladores digitais e sua utilização em sistemas digitais, onde códigos que utilizam controladores digitais foram analisados sistematicamente.

A etapa seguinte consiste no desenvolvimento de uma estrutura capaz de representar sistemas em espaço de estado. A estrutura em questão está integrada ao software DSVerifier possibilitando assim o suporte a verificação de controladores digitais de ponto-fixo na representação de espaço de estado. Essa metodologia aumenta a automação no processo de verificação de tais controladores e foi utilizada para verificar cinco tipos diferentes de propriedades (estouro aritmético, erro de quantização, estabilidade, observabilidade e

controlabilidade). É importante ressaltar que tais propriedades podem ser verificadas em malha fechada. Foi desenvolvida também uma função para converter controladores representados com funções de transferências para espaço de estados,

Finalmente, a metodologia proposta foi aplicada na verificação de sistemas digitais de forma que possa ser checada as propriedades citadas acima a partir da nova representação, validando assim o seu uso através de controladores digitais.

DISCUSSÃO DOS RESULTADOS

O primeiro passo deste trabalho é fazer com que o DSVerifier possa aceitar a representação em espaço de estados. Para atingir esse objetivo, foi preciso modificar o *front-end* da ferramenta de modo que ela conseguisse identificar sistemas na respectiva representação. Como já mencionado anteriormente, os dois verificadores de modelos utilizados, ESBMC e CBMC, respectivamente, são capazes de verificar programas em ANSI-C/C++. Portanto, foi criado no DSVerifier uma estrutura denominada de *state_space_system*. Tal estrutura, combinada com um arquivo escrito em ANSI-C com todas as especificações do sistema, permite que o DSVerifier possa identificar sistema representados em espaço de estado. Contudo, o arquivo de entrada em ANSI-C necessariamente traria muita informação, ou seja, o usuário poderia perder muito tempo apenas na criação de tais arquivos quando fosse testar vários sistemas. Tendo isso em mente, um novo sistema de entrada foi criado para representar sistemas em espaço de estado. De maneira a entender melhor como tal sistema funciona, observe o sistema abaixo:

$$\begin{cases} \dot{x} = \begin{bmatrix} 2{,}0 & 1{,}0 \\ 1{,}0 & 5{,}0 \end{bmatrix} x + \begin{bmatrix} 0 \\ 1{,}5 \end{bmatrix} u \\ y = [1{,}0 \quad 0{,}6] x \end{cases}$$

De modo a representar o respectivo sistema, o usuário deveria descrever as especificações do respectivo sistema utilizando o código ANSI-C mostrado na Fig. 1. Como pode ser observado nesta figura, muitas informações são necessárias para representar de fato o sistema, no entanto, softwares, como por exemplo o MATLAB, utilizam notações mais simples para representar sistemas em espaço de estado. Deste modo, aproveitamos a mesma representação já utilizada por

sistemas de simulação conhecidos para representar controladores em espaço de estado no DSVerifier.

```
#include "../../../bmc/dsverifier.h"

digital_system_state_space _controller;

implementation impl = {
    .int_bits = 15,
    .frac_bits = 16
};

int rowA = 2;
int columnA = 2;
int rowB = 2;
int columnB = 1;
int rowC = 1;
int columnC = 2;
int rowD = 1;
int columnD = 1;
int rowInputs = 1;
int columnInputs = 1;
int rowStates = 2;
int columnStates = 1;
int rowOutputs = 1;
int columnOutputs = 1;
double error_limit = 0.001018;

void initials(){

    _controller.A[0][0] = 2.0;
    _controller.A[0][1] = -1.0;
    _controller.A[1][0] = 1.0;
    _controller.A[1][1] = 5.0;

    _controller.B[0][0] = 0.0;
    _controller.B[1][0] = 1.5;

    _controller.C[0][0] = 1.0;
    _controller.C[0][1] = 0.6;

    _controller.D[0][0] = 0.0;

    _controller.inputs[0][0] = 1.0;

}
```

Figura 1. Código ANSI-C representando um sistema em espaço de estado.

De modo a facilitar esse tipo de representação, um sistema de *parser* foi criado. Tal sistema recebe as especificações de um determinado sistema digital (representado em espaço de estado) de uma forma mais simplificada (tomamos como base a representação utilizada pela ferramenta MATLAB) e então tal informação é pré-processada criando um arquivo em ANSI-C com todas as informações necessárias para a verificação do respectivo sistema. Foi padronizado então, que a especificação para o DSVerifier utilizada seria descrita num arquivo com extensão *.ss* contendo o tipo de implementação do sistema, a quantidade de variáveis de estado, entradas e saídas, as matrizes que representam o sistema, a entrada e qualquer informação adicional que seja utilizada para testar uma determinada propriedade. Fig. 2 mostra um exemplo de arquivo *.ss*.

```
implementation <15,16>
states = 2;
inputs = 1;
outputs = 1;
A = [2.0,-1.0;1.0,5.0]
B = [0.0;1.5]
C = [1.0,0.6]
D = [0.0]
inputs = [1.0]
double error_limit = 0.001018;
```

Figura 2. Arquivo de entrada para o DSVerifier com as especificações do sistema.

A partir dessa nova representação, o próximo passo é a implementação da verificação de novas propriedades. Até o momento da elaboração deste relatório, foram implementadas a verificação de cinco propriedades: erro de quantização para sistemas *single input single output* (SISO), estouro aritmético, estabilidade, controlabilidade e observabilidade para sistemas SISO e *multi input multi output* (MIMO). Como já mencionado, tais propriedades podem ser verificadas em malha fechada.

1. *Erro de quantização*

Os processo envolvidos na conversão de um determinado valor para aritmética de ponto fixo ou convertendo o valor em ponto fixo para seu valor real contém aproximações e arredondamentos, o que acaba alterando o valor real que foi convertido. Dependendo da implementação que está sendo aplicada no sistema, esses erros por arredondamentos podem ser atenuados ou suavizados. Portanto, para cada implementação e para cada sistema, nós temos uma margem de erro (%) aceitável para manter o funcionamento correto do respectivo sistema. Baseado nas especificações do sistema e na margem de erro determinada pelo projetista, o DSVerifier faz uma simulação do comportamento do sistema e verifica se o mesmo gera valores de saída e estados que ultrapassem a margem de erro permitida pelo projetista levando em consideração detalhes de implementação, como por exemplo a precisam utilizada.

2. *Estouro Aritmético*

Os estouros aritméticos podem ocorrer quando valores ultrapassam a quantidade disponível pela palavra finita (número de *bits* na parte inteira). De modo a identificar esse tipo de problema, o DSVerifier simula o comportamento do sistema para um dada entrada e verifica ao final de todas as operações aritméticas se os valores gerados respeitam os limites estipulados para o tamanho da palavra.

3. *Estabilidade*

Um das propriedades mais importantes dos sistemas é denominada de Estabilidade. Neste caso, um sistema é dito estável se dado uma entrada limitada, o sistema fornece uma saída que também é limitada. É importante ressaltar que, dependendo de como o sistemas é implementado, essa propriedade pode não ser mantida, o que pode ocasionar o sistema ter um comportamento totalmente diferente do qual ele foi projetado. De maneira a verificar esta propriedade, o DSVerifier calcula todos os polos do sistema e verifica se os mesmos contém a parte real negativa, caso contrário, o sistema é dito instável.

4. *Observabilidade*

Observabilidade é medida que informa ao projetista o quanto é possível identificar os estados a partir das saídas de um determinado sistema. Neste caso, um sistema é dito observável se podemos inferir qualquer dos seus estados a partir apenas das suas saídas. O DSVerifier verifica se um dado sistema é ou não observável. A ferramenta identifica a matriz de observabilidade de um sistema e verifica se o seu determinante é diferente de zero. Caso ele seja diferente de zero, o sistema é observável.

5. Controlabilidade

Controlabilidade e observabilidade são propriedades matemáticas duais. Essa propriedade determina se um dado sistema pode ser controlável ou, pelo menos, parcialmente controlável. O DSVerifier extrai a matriz de controlabilidade de um dado sistema e verifica seu determinante. Caso o mesmo seja diferente de zero, o sistema é controlável.

De maneira a confirmar o poder de verificação do DSVerifier, foi desenvolvida um suite de testes com 25 sistemas estáveis, controláveis e observáveis, os quais foram retirados da literatura. Cada sistema foi verificado com relação as propriedades acima citadas e um resumo dos resultados pode ser visto na Fig. 3. Além disso, cada sistema foi testado utilizando-se um micro-controlador de 32-bits com 3 tipos de precisão (8, 16 e 32-bits).

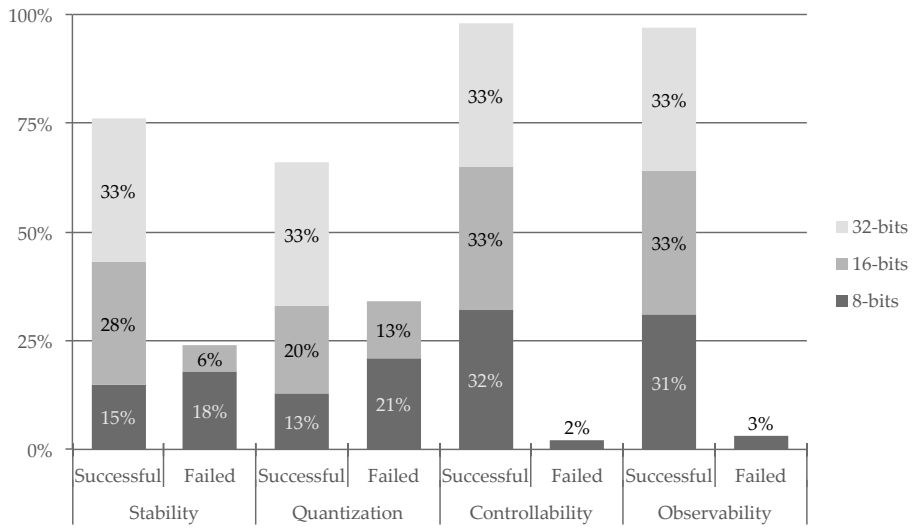

Figura 3. Resultados experimentais.

A partir da análise dos resultados, 3 observações importantes foram feitas: **(i)** que de fato as propriedades dos sistemas podem variar devido a mudança na precisão, **(ii)** quanto menor a precisão, maior a sensibilidade das propriedades do sistema e **(iii)** a pesar da mudança de precisão, controlabilidade e observabilidade não foram afetadas.

CONCLUSÃO

Esta pesquisa visa expandir o verificador de sistemas digitais DSVerifier para suportar a verificação de sistemas representados por espaço de estados. Além de modificar o *front-end* da ferramenta para aceitar esse tipo de representação, este trabalhando também visa expandir as propriedades que são verificadas e, até o presente momento, consegue verificar propriedades como erro de quantização de sistemas SISO e estabilidade, estouros aritméticos, observabilidade e controlabilidade para sistemas SISO e MIMO representados em espaço de estados. Além disso, tais propriedades podem ser verificadas em malha fechada, considerando não somente um sistema isolado, mas o impacto de um controlador numa determinada planta.

Neste relatório foram descritas as implementações da estrutura base para representar sistemas em espaço de estado. Também foi abordada a implementação de um sistema de *parser* para transformar as especificações de sistemas em uma entrada aceitável pelos verificadores de modelo utilizados pela ferramenta (ESBMC e CBMC). As propriedades que são atualmente verificadas e seus respectivos processos de verificação foram também apresentados.

Por fim, os trabalhados futuros a serem desenvolvidos nessa pesquisa visam aumentar a variedade de propriedades suportadas pela ferramenta, tais como, ciclo limite, fase mínima. Por fim, devemos levar em consideração no processo de verificação, as variações que podem ocorrer na especificação de uma determinada planta.

REFERÊNCIAS BIBLIOGRÁFICAS

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

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

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

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

[5] Yang G, Guo X, Che W, Guan W. Linear Systems: Non-Fragile Control and Filtering. Taylor & Francis, 2013.

[6] Masten M, Panahi I. Digital signal processors for modern control systems. Control Engineering Practice 5(4):449 – 458, DOI doi:10.1016/S0967-0661(97)00024-5 Monmasson E, Cirstea M (2007) FPGA Design Methodology for Industrial Control Systems 2014;A Review. IEEE TIE 54(4):1824–1842, DOI 10.1109/TIE.2007. 898281, 1997.

[7] Li G. On pole and zero sensitivity of linear systems. IEEE TCS 44(7):583–590, DOI 10.1109/81.596939, 1997.

[8] Li G. On the structure of digital controllers with finite word length consideration. IEEE TAC 43(5):689– 693, DOI 10.1109/9.668838, 1998.

[9] Mantey P. Eigenvalue sensitivity and state-variable selection. IEEE TAC 13(3):263–269, DOI 10. 1109/TAC.1968.1098902, 1968.

[10] Middleton R, Goodwin G. Improved finite word length characteristics in digital control using delta operators. IEE TAC 31(11):1015–1021, DOI 10.1109/TAC. 1986.1104162, 1986.

[11] Wu J, Chen S, Chu J. Computing a FWL stability measure for second order digital systems. In: ICARCV, pp 1593–1598, DOI 10.1109/ICARCV.2004.1469297, 2004.

[12] Morse J, Cordeiro L, Nicole D, Fischer B. Handling Unbounded Loops with ESBMC 1.20 - (Competition Contribution). In: TACAS, LNCS 7795, pp 619–622, 2013.

[13] Morse J, Ramalho M, Cordeiro L, Nicole D, Fischer B. ESBMC 1.22 - (Competition Contribution). In: TACAS, LNCS 8413, pp 405–407, 2014.

[14] Abreu RB, Cordeiro LC, Filho EBL. Verifying Fixed-Point Digital Filters using SMT-Based Bounded Model Checking. In: SBrT, DOI http://dx.doi.org/10. 14209/sbrt.2013.57, 2013.

[15] Bessa I, Abreu R, Cordeiro L, Filho JE. SMT- Based Bounded Model Checking of Fixed-Point Digital Controllers. In: IECON, pp 295–301, 2014.

[16] Bessa I, Ibrahim H, Cordeiro L, Filho JE. Verification of Delta Form Realization in Fixed-Point Digital Controllers Using Bounded Model Checking. To appear in SBESC, 2014.

[17] Harnefors L. Implementation of Resonant Controllers and Filters in Fixed-Point Arithmetic. IEEE TIE 56(4):1273–1281, DOI 10.1109/TIE.2008.2008341, 2009.

[18] Carletta J, Veillette R, Krach F , Fang Z. Determining appropriate precisions for signals in fixed-point IIR filters. In: DAC, pp 656–661, DOI 10.1109/DAC.2003. 1219100, 2003.

[19] Istepanian R, Whidborne J. Multi-objective design of finite word-length controller structures. In: CEC, pp 68, DOI 10.1109/CEC.1999.781908, 1999.

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

[21] Sung W, Kum KI. Simulation-based word-length optimization method for fixed-point digital signal processing systems. IEEE TSP 43(12):3087–3090, DOI 10.1109/78.476465, 1995.

[22] Istepanian RH, Chen S. Stability issues of finite-precision controller structures for sampled-data systems. International Journal of Control 72(15):1331–1342, DOI 10.1080/00207179922046, 1999.

[23] Wo S, Zou Y, Chen Q, Xu S. Non-fragile controller design for discrete descriptor systems. Journal of the Franklin Institute 346(9):914 – 922 DOI 10.1016/j. jfranklin.2009.07.008, 2009.

[24] Behrmann G, David A, Larsen KG. A tutorial on uppaal. In: SFM-RT, LNCS 3185, pp 200–236, 2004.

[25] Tripakis S, Yovine S, Bouajjani A. Checking timed buechi automata emptiness efficiently. In: Formal Methods in System Design, pp 267–292, 2005.

[26] Jensen K, Kristensen LM. Coloured Petri Nets - Modelling and Validation of Concurrent Systems. Springer, DOI 10.1007/b95112, 2009.

[27] Magellan. Hybrid RTL formal verification. < http://www.synopsys.com/tools/verification/functionalverification/pages/magellan.aspx >, Acessado dia 12 de setembro de 2014.

[28] KROENING, D.; TAUTSCHNIG, M. CBMC – C Bounded Model Checker. In: Tools and Algorithms for the Construction and Analysis of Systems. : Springer Berlin Heidelberg, 2014. v. 8413, p. 389–391. ISBN 978-3-642-54861-1.

CRONOGRAMA DE ATIVIDADES

Nº	Descrição	Ago 2015	Set	Out	Nov	Dez	Jan 2016	Fev	Mar	Abr	Mai	Jun	Jul
1	Estudar a técnica de Verificação Formal de Modelos.	X	X										
2	Estudar sobre controladores digitais e suas aplicações em sistemas de controle.		X	X	X								
3	Estudar a conversão de representação em espaço de estados para função de transferência.			X	X	X							
4	Implementar uma estrutura no *front-end* da ferramenta para aceitar a representação em espaço de estado					X	X	X					
5	Integrar a estrutura desenvolvida ao processo de verificação com o DSVerifier.								X				
6	Implementar a verificação de propriedades relacionadas a esse tipo de representação								X	X	X		
7	Aplicar a metodologia proposta à verificação de *benchmarks* e, através dos resultados obtidos, concluir sobre a sua eficácia.									X	X		
8	- Elaboração do Resumo e Relatório Final (atividade obrigatória) - Preparação da Apresentação Final para o Congresso (atividade obrigatória)												X