UNIVERSIDADE FEDERAL DO AMAZONAS

PRÓ-REITORIA DE PESQUISA E PÓS-GRADUAÇÃO

DEPARTAMENTO DE APOIO Á PESQUISA

PROGRAMA INSTITUCIONAL DE INICIAÇÃO CIENTÍFICA

VERIFICAÇÃO DE CONTROLADORES DIGITAIS DE PONTO FIXO USANDO REALIZAÇÕES DE FORMAS DIRETAS E DELTAS

Bolsista: Vithória dos Santos Barbosa, CNPq

MANAUS

2016

UNIVERSIDADE FEDERAL DO AMAZONAS

PRÓ-REITORIA DE PESQUISA E PÓS-GRADUAÇÃO

DEPARTAMENTO DE APOIO Á PESQUISA

PROGRAMA INSTITUCIONAL DE INICIAÇÃO CIENTÍFICA

RELATÓRIO FINAL

PIB-E/0002/2015

VERIFICAÇÃO DE CONTROLADORES DIGITAIS DE PONTO FIXO USANDO REALIZAÇÕES DE FORMAS DIRETAS E DELTAS

Bolsista: Vithória dos Santos Barbosa, CNPq

Orientador: Prof. Dr. Lucas Carvalho Cordeiro

MANAUS

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.

RESUMO

Controles digitais são amplamente utilizados pela comunidade de engenharia de controle por causa das várias vantagens sobre os controladores analógicos, tais como a melhoria da confiabilidade, sensibilidade, flexibilidade e custo. Entretanto, existem algumas desvantagens em usar controladores digitais, como por exemplo: 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; e em particular, problemas relacionados com os efeitos da palavra finita. Este projeto de pesquisa tem como base o uso de ferramentas de verificações automáticas. Em especial, a verificação de modelo limitada, baseada nas teorias do módulo da satisfatibilidade usada para verificar cinco tipos de propriedades, que incluem: estouro aritmético, ciclo limite, restrições temporais, estabilidade, e fase mínima. O problema de determinar o comprimento de palavra finita, possui efeitos que podem ser subsequentemente resolvidos usando métodos numéricos apropriados . O objetivo geral desta pesquisa é demonstrar que o método *BMC*, baseado nas *SMT* pode ser uma poderosa ferramenta no projeto de verificação de controladores digitais, auxiliando o engenheiro de controle com um mecanismo eficiente de verificação, que é mais confiável e menos trabalhoso do que os instrumentos de simulação, considerando que estes são pouco precisos e exigem intervenção manual de projetistas. Além de estudar e desenvolver o Procedimento de Otimização, a fim de observar se houve desempenho e, se possível, integrá-lo no verificador de software ESBMC (*Efficient SMT-Based Bounded Model Checking*), para a verificação de aplicações reais.

Palavras-chave: Controladores Digitais. Comprimento de palavra finita. Métodos Formais. Otimização.

ABSTRACT

Digital controls are widely used by the control engineering community because of several advantages over analog controllers, such as reliability improvement, sensitivity, flexibility and cost. However, there are some disadvantages in using digital controllers, like errors introduced during the quantization, for example. In this context, there are some initiatives to solve the problems that appear in the discrete time domain; and particularly, problems relating to the effects of finite word. This research project is based on the automatic verification tools usage. In particular, the limited model verification, based on satisfiability modulo theories used to verify five properties types, which include: arithmetic overflow, limit cycle, timing constraints, stability, and minimum phase. The problem of determining the finite word length has effects that may, subsequently, be resolved using appropriate numerical methods. The research main purpose is to demonstrate that BMC method, based on SMT can be a powerful tool in digital controllers verification project, helping the control engineer with an efficient verification mechanism, which is more reliable and less laborious than the simulation tools, whereas these are inaccurate and require designers' manual intervention. In addition to study and develop the Optimization Procedure, to see if there were performance and, if possible, integrate it into the software checker ESBMC (Efficient SMT-based Bounded Model Checking), for real applications verification.

Keywords: Digital controllers. Finite word length. Formal Methods. Optimization.

LISTA DE ABREVIAÇÕES

BMC - *Bounded Model Checking*

CTL - *Computer Tree Logic*

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

FWL - *Finite Word Length*

IIR - *Infinite Impulse Response*

LTL - Logic Temporal Linear

SMT - *Satisfatibility Module Theories*

LISTA DE ILUSTRAÇÕES

SUMÁRIO

1. INTRODUÇÃO

Um sistema embarcado é um sistema micro processado no qual o computador é completamente encapsulado ou dedicado ao dispositivo ou sistema que ele controla, caracterizado por ser um conjunto de tarefas geralmente com requisitos pré-definidos. Nos dias atuais, a quantidade de software tem aumentado de forma significativa em produtos embarcados de tal maneira que a verificação de sistemas de hardware e software desempenha um papel importantíssimo para assegurar a qualidade do produto. As empresas instaladas no Polo Industrial de Manaus (PIM) têm focado grande parte dos seus investimentos no desenvolvimento desse tipo de sistema. Estes conjuntos são desenvolvidos para uma tarefa específica, por questões de segurança e usabilidade, e alguns inclusive, possuem restrições para computação em tempo real. Equipamentos de redes portáteis de medição, impressoras e telefones celulares são alguns exemplos de sistemas embarcados. Dentro deste cenário, o desenvolvimento de hardware e software para sistemas embarcados tem desempenhado um papel fundamental na economia da região amazônica. Em grande parte dos sistemas embarcados, pode-se encontrar o uso extensivo de controladores digitais de ponto fixo, já que estes exigem uma crescente demanda de esforços para evitar erros de concepção que aparecem no domínio de tempo discreto. Deste modo, este projeto de pesquisa tem como objetivo propor uma nova metodologia de verificação, assim como o procedimento de Otimização, que emprega a verificação de modelos limitados, tendo como base nas teorias do módulo da satisfatibilidade (do inglês, *Satisfiability Modulo Theories* - SMT) para verificar a ocorrência dos erros nos projetos, que fazem menção ao comprimento de palavra finita nos controladores digitais de ponto fixo. Esse procedimento ganha destaque nesta pesquisa pelo fato de reduzir, consideravelmente, possíveis erros dentro de um sistema.

2. REVISÃO BIBLIOGRÁFICA

Com o avanço da tecnologia, é cada vez mais comum a nossa interação com sistemas computadorizados. Para garantir a ausência de erros não previstos em tais sistemas, o uso de técnicas convencionais de simulação e testes pode ser inviável. Portanto, torna-se útil o uso da Verificação Formal, das propriedades do sistema, necessárias para o seu correto funcionamento. O uso desse método de verificação exige que o sistema e suas propriedades sejam modelados de uma forma matemática e abstrata, para que os algoritmos de verificação possam ser plenamente aplicados.

O termo Verificação Formal corresponde a uma porção de técnicas para análise automática de sistemas reativos. Mas são principalmente duas as técnicas bem estabelecidas: Verificação de Modelos (geralmente um matemático ou um lógico) e Prova de Teoremas (adequado para situações onde a manipulação de dados tem maior importância). Para tratar a Verificação de Modelos de forma algorítmica, é representado matematicamente o sistema e suas especificações utilizando lógica, como a lógica proposicional e a lógica temporal linear, de tal forma que se possa verificar se uma dada fórmula é satisfeita dada uma determinada estrutura. De forma a realizar este processo automaticamente, são utilizados softwares, conhecidos como verificadores de modelo, tendo como principal objetivo verificar de forma confiável, as propriedades de um determinado sistema.

Atualmente, algumas técnicas alternativas tem sido propostas para a verificação das implementações de ponto fixo de IIR de filtros digitais, que se baseiam na verificação de modelo limitado (BMC) e sugere o uso de solucionadores da teoria do modulo da satisfatibilidade (SMT) [8]. A principal idéia por trás do SMT-base BMC é considerar contra-exemplos de um determinado comprimento K e gerar uma fórmula lógica de primeira ordem que pode ser satisfeita se, e somente se, tal contra-exemplo existe[9].

A Técnica BMC (*Bounded Model Checking*) baseada em solucionadores SMT (*Satisfatibility Module Theories*), verifica (a negação de) uma determinada propriedade em uma determinada profundidade. A ferramenta BMC utilizada nesse trabalho é chamada DSVerifier(um módulo de software para o ESBMC) [11] e tem sido recentemente utilizada para verificar sistemas digitais. Em Abreu et al. [7], propriedades como estouro aritmético, ciclo limite, restrições temporais, estabilidade e resposta em frequência para filtros digitais são verificadas usando o ESBMC(*Efficient SMT-Based*

Context - Bounded Model Checker). Este é um verificador de modelos eficiente e eficaz baseado em solucionadores SMT (*Satisfatibility Module Theories*) para programas C/C++. Ele permite procurar por violações em propriedades relacionadas à: segurança de ponteiros, limite de vetores, atomicidade, estouro aritmético, bloqueio fatal, corrida de dados e vazamento de memória para programas sequenciais e paralelos (com memória compartilhada e mutex). Além de verificar programas que envolvem níveis de bits, ponteiros, estruturas, uniões e aritmética de ponto fixo.

O DSVerifier pode executar uma análise similar para o sistemas zeros, a fim de verificar a fase mínima para controladores digitais. Controles digitais são amplamente utilizados em microcomputadores, microprocessadores, processadores de sinais digitais [3] e portas de campo de matrizes programáveis [4]. De acordo com a escolha do *hardware*, o formato e a aritmética usados para representar e manipular os números pode mudar (por exemplo, número de bits, aritmética de ponto fixo ou flutuante). Um controlador digital pode ser definido como um sistema causal, linear, discreto e invariante no tempo, podendo ser representado matematicamente de várias maneiras (e.g., funções de transferência, equações de espaço de estado e equações de diferença). No entanto, a estrutura de realização escolhida pode influenciar bastante no seu desempenho. Devido a algumas desvantagens, por exemplo, erros que são introdução durante o processo de quantização, existem algumas iniciativas para resolver os problemas que aparecem no domínio de tempo discreto; e em particular, problemas relacionados com os efeitos do comprimento de palavra finita (do inglês, *Finite Word Length* – FWL) [1,2].

3. METODOLOGIA

Esta pesquisa tem como principal objetivo desenvolver um modelo operacional, com a finalidade reduzir os erros causados pelo comprimento da palavra finita e integrar o respectivo modelo no verificador ESBMC. De forma a alcançar tal objetivo, foi adotada uma metodologia que pode ser dividia em três etapas principais: estudo das tecnologias utilizadas por meio de uma revisão bibliográfica, a realização de testes no DSVerifier para que seja feita as comparações necessárias e a criação de funções específicas no MATLAB do novo Procedimento de Otimização.

Primeiramente, foi realizada uma revisão da literatura a respeito da teoria de verificação de modelos. Nesta etapa, a maioria dos conceitos importantes acerca de Lógica Proposicional, Lógica Temporal Linear (LTL), Lógica de Árvore de Computação (CTL), foram estudadas com o intuito de entender o funcionamento do verificador ESBMC[11]. Com base neste, com intuito de analisar o funcionamento da ferramenta em questão, foram feitos testes , no sistema operacional Linux, com funções no domínio Z e em Espaço de Estados, para verificar se houve alguma violação das propriedades.

Devido ao tipo de respresentação em Espaço de Estado ter apresentado resultados satisfatórios, com o intuito de observar tal comportamento e comprová-los, foram feitos testes, tirados da literatura. Cada teste foi implementado na suíte de teste criada na pasta chamada “State Sapace”, onde é possível, em sistemas Linux, fazer a verificação diretamente pelo terminal. Com o intuito de reduzir as degradações numéricas devido ao arredondamento e quantização, houve um estudo aprofundado no novo Procedimento de Otimização. Suas funções foram desenvolvidas e validadas no MATLAB. Este programa é definido um software sistema interativo e uma linguagem de programação para computação técnica e científica em geral, integrando a capacidade de fazer cálculos, visualização gráfica e programação[15]. Ele também dispõe de diversas extensões(chamadas toolboxes ou block sets). Além dos módulos adicionais, o MATLAB conta com o Simulink, um ambiente de simulação baseado em diagrama de blocos e plataforma para *Model-Based Design*.

4. RESULTADOS E DISCUSSÕES

O modelo de Espaço de Estados de um sistema dinâmico de tempo contínuo pode ser derivado a partir do modelo do sistema dado no tempo domínio por uma equação diferencial ou a partir de sua representação de função de transferência. O modelo de sistema completo para um sistema linear invariante no tempo consiste de um conjunto de estado n equações definidas, em termos de as matrizes A e B, e um conjunto de equações de saída que relaciona todas as variáveis de interesse de saída para as variáveis de estado e insumos, e é expresso em termos das matrizes C e D. Esse sistema pode ser descrito como:

$$\mathbf{x = Ax + Bu}$$

$$\mathbf{y = Cx + Du}$$

As matrizes A e B são as propriedades do sistema e são determinadas pela estrutura e pelos elementos do sistema. A equação de matrizes de saída C e D são determinadas pela escolha particular de variáveis de saída. Esse procedimento de modelagem global se baseia nas seguintes etapas: determinação da ordem do sistema n e seleção de um conjunto de variáveis de estados a parti da representação do sistema gráfico linear, geração de um conjunto de equações de estado e sistema de matrizes A e B, determinação de um conjunto de equações de saída adequado e derivação das matrizes C e D apropriados .

Para que seja realizada a verificação deste processo, é necessário inserir as características das matrizes em um código na linguagem C. Esse tipo de sistema pode ser representado da seguinte maneira:

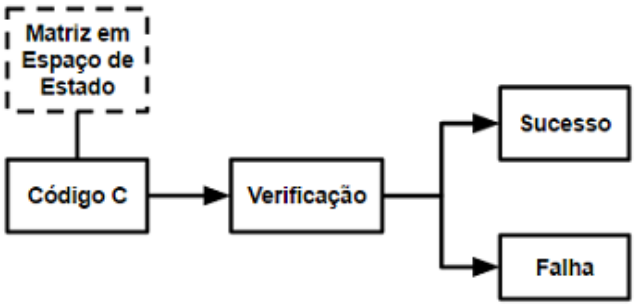

Figura 1: Diagrama ilustrando o processo de verificação de funções em Espaço de Estado.

O diagrama da Figura 1 ilustra como decorre o processo de verificação utilizando o modelo operacional. Como se trata de um processo automático deve-se passar para o verificador o código que será verificado, juntamente com o modelo operacional e, a partir disso, o verificador realizará todo o processo de verificação, retornará como resultado a existência ou a inexistência de algum *bug* e, além disso, no caso da detecção do respectivo *bug*, também retornará um contraexemplo indicando o caminho que deve ser percorrido, durante a execução do programa, para expor o determinado erro. A parti disto foi criada uma suíte de teste para verificarmos o comportamento de alguns casos em Espaço de Estados.

Quando implementados em dispositivos de computação digital, os controladores são submetidos a degradações numéricas devido ao arredondamento e quantização. Existem dois efeitos principais da palavra finita (FWL): o ruído de arredondamento, que é a adição de ruído no sistema resultante dos arredondamentos das várias outras, antes e depois, após cada operação aritmética e erros paramétricos que são a quantização dos coeficientes do controlador/parâmetros. Eles degradam o desempenho e/ou a estabilidade do controlador.

O problema de realização do controlador ótimo se coloca a procura de uma realização de ponto flutuante que maximiza a proposta FWL de circuito fechado medindo estabilidade, e uma técnica numérica de otimização é adotada para resolver o problema de otimização resultante[12]. Como diferentes realizações a variável, X, têm diferentes valores do FWL – medida de estabilidade em malha fechada $\rho_1(X)$, que é de importância prática para encontrar uma realização "ótima" Xopt que maximiza $\rho_1(X)$. O controlador implementado com esta realização óptima Xopt necessita de um comprimento mínimo de bits e tem uma tolerância máxima para o erro FWL. Este problema de realização controlador ótimo é formalmente definida como:

$$\upsilon \stackrel{\triangle}{=} \max_{\mathbf{X} \in \mathcal{S}_C} \rho_1(\mathbf{X})$$

Figura 2: Definição de Controlador Ótimo.

Visando a implementação deste tipo de controlador, foi necessário a criação de cada função específica no MATLAB. O modelo de planta é assumido como sendo estritamente correta como uma descrição de Espaço de Estado(A_p, B_p, C_p), onde $A_p \in R^{mxm}$, $B_p \in R^{mxl}$ e $C_p \in R^{qxm}$. Considerando(A_c, B_c, C_c, D_c) uma descrião de Espaço de

Estados do controlador C*(z)*, com $A_c \epsilon R^{nxn}$, $B_c \epsilon R^{nxq}$ e $C_c \epsilon R^{lxn}$ e $D_c \epsilon R^{lxq}$. Um sistema linear com uma determinada matriz de uma função de transferância tem um número infinito de espaço de estados. Se (A^0c, B^0c, C^0c, D^0c) é uma descrição de espaço de estado de C*(z)*, onde a matriz de transformção T ϵ R^{nxn} é uma matriz não singular arbitrária, podemos representa-la por:

$$\mathbf{X} = [x_{j,k}] \triangleq \begin{bmatrix} \mathbf{D}_C & \mathbf{C}_C \\ \mathbf{B}_C & \mathbf{A}_C \end{bmatrix}$$

Figura 3: Representação da matriz não singular.

No MATLAB, esta matriz foi desenvolvida como mostra o código abaixo. É necessário ressaltar que as matrizes de entrada podem ter diversos tamanhos e que o caso estudado está comentado em verde.

```
function [X0] = matrix(Ac, Bc, Cc, Dc)

%Ac = [2.3985e+0, -1.8017e+0, 4.0317e-1; 1, 0, 0; 0, 1, 0];
%Bc = [1; 0; 0];
%Cc = [-7.3591e+1, 1.4661e+2, -7.3018e+1];
%Dc = [1.2450e+2];

X0 = [Dc, Cc; Bc, Ac];

end
```

Figura 4: Sequência lógica da matriz em função de X0.

A estabilidade do sistema de controle depende dos valores próprios da transição de circuito fechado da matriz. Todas diferentes realizações *X* possuem exatamente o mesmo conjunto de polós de malha fechada se eles forem implementados com precisão infinita. Desta forma a função *Matrix(X)* é por definição:

```
function [ matrix ] = AmatrixX(X, M0, M1, M2)

Ap = [1.9980e+0, -9.9800e-1; 1, 0];
Bp = [1; 0];
Cp = [3.9880e-3, -4.0040e-3];
Dp = [0];

Ac = [2.3985+0, -1.8017e+0, 4.0317e-1; 1, 0, 0; 0, 1, 0];
Bc = [1; 0; 0];
Cc = [-7.3591e+1, 1.4661e+2, -7.3018e+1];
Dc = [1.2450e+2];

X = [Dc, Cc; Bc, Ac];

m = 2;
n = 3;
l = 1;
q = 1;

matrix = M0+M1*X*M2

end
```

Figura 5: Sequência lógica da matriz em função de X.

A matriz *X* pode assumir uma matriz qualquer, desde que suas dimensões correspondem com as matrizes a qual será feita a sua multiplicação. Considerando p_{0i} o autovetor da direita da Matriz de *X* correspondente ao autovalor λ_i e ao autovetor da esquerda y_{0i}, relacionado a $p_{0i.}$ Todas descrições em espaço de estado (*Sc*) podem ser definidas por:

```
function [ functionT ] = XmatrixT(X0, X1, X2)

%Ap = [1.9980e+0, -9.9800e-1; 1, 0];
%Bp = [1; 0];
%Cp = [3.9880e-3, -4.0040e-3];
%Dp = [0];

Ac = [2.3985+0, -1.8017e+0, 4.0317e-1; 1, 0, 0; 0, 1, 0];
Bc = [1; 0; 0];
Cc = [-7.3591e+1, 1.4661e+2, -7.3018e+1];
Dc = [1.2450e+2];

T = rand(3);
To = inv(T);
X0 = [Dc, Cc; Bc, Ac];
X1 = [eye(1), zeros(1,3); zeros(3,1), To];
X2 = [eye(1), zeros(1,3); zeros(3,1), T];

% Ip = eye(2,1)
% In = eye(1,2)
% Im = eye (2)

functionT = X1*X0*X2
functionTm = abs(functionT)

end
```

Figura 6: Sequência lógica da matriz Function(T).

Depois de definidas as funções primordiais, temos por definição uma função ϕ_i. Nela os valores encontrados de autovetor, tanto da esquerda como da direita, e do autovalor serão utilizados para cada valor de ϕ_i, com "i" variando de 1 até m+n.

```
%
%functionFi = (M1'[lambdai*Y0i*P0i]M2')
%
function [Fi] = functionFi(M1, M2, i)
    lambda = d(i, i);
    Y0 = w(i, :)
    P0 = v(i, :)
    P0t = P0'
    Fi(i) = (M1') * real(lambda*Y0(1, i)*(P0t(1, i))) * (M2')
end
```

Figura 7: Sequência lógica da função ϕ_i.

Todas as diferentes realizações *X* em *Sc* possuem exatamente o mesmo conjunto de polós de malha fechada se eles são implementados com precisão infinita. Uma vez que o circuito de malha fechada foi concebido para ser estável, todos os

valores próprios, $\lambda i(\bar{A}(X))$, $1 \leq i \geq m + n$, estão dentro da unidade de disco. Para minimizar os cálculos, antes de usarmos os valores encontrados na última função, foi necessário criar duas funções denominadas: Minvalue e Maxvalue. A função Minvalue foi desenvolvida com a finalidade de calcular os valores máximos e os valores mínimos de qualquer matriz, podendo estar em módulo ou não. Ela foi desenvolvida com o objetivo de representar, por definição :

$$\|X\| \max \triangleq \max_{j,k} |x_{j,k}|$$

```
function minValue = mixMatrix(Matrix, pMod, pMax)
if pMod
    Matrix = abs(Matrix);
end
if pMax
    aux = max(Matrix);
    minValue = max(aux);
else
    aux = min(Matrix);
    minValue = min(aux);
end
end
```

Figura 8: Sequência lógica da função Minvalue.

São parâmetros dessa função “pMod” e “pMin”, diante disto se pMod for igual a zero a operação será realizada na variável *Matrix* original, se for igual a um a operação será realizada com o módulo da mesma. Para o parâmetro pMax, se a mesma for igual a zero será calculado o valor mínimo de determinada matriz e se for igual a um será calculado o valor máximo.

A função *Maxvalue* foi desenvolvida com a finalidade de calcular a soma dos maiores elementos de uma matriz, em módulo ou não. Assim como a função anterior, esta função também possui os seus parâmetros, pMod e pMax, que seguem o mesmo comando.

```
function maxValue = maxMatrix(Matrix, pMod, pMax)
if pMod
    Matrix = abs(Matrix);
end
if  pMax
    aux = max(Matrix);
    maxValue = sum(aux);
else
    aux = min(Matrix);
    maxValue = sum(aux);
end
end
```

Figura 9: Sequência lógica da função Maxvalue.

A função *Function(F)* possui cinco parâmetros de entrada. Ela é formada por uma serie de operações, algumas calculadas em funções específicas anteriormente e outras com operações já definidas na ferramenta de verificação utilizada.

```
function [F] = functionF(F1, Fi(i), F2, functionT, lambda)
T = rand(3)
Tt = T'
Tti = inv(T')
F1 = [eye(1), zeros(1, 3); zeros(1, 3), Tt]
F2 = [eye(1), zeros(1, 3); zeros(1, 3), Tti]
Fi(i) = (M1') * real(lambda*Y0(1, i)*(P0t(1, i))) * (M2')
lambda = d(i, i)

F = min(inv(sum[abs((F1*Fi*F2)*(dot(functionT))]/abs(lambda)*
    (1 - abs(lambda))]*log2(4*(maxMatrix(functionT, 1,1)))/
    mixMatrix(functionT,0,0)])
end
```

Figura 10: Sequência lógica da função F.

Este problema de otimização sugere que, ao contrário de otimizar a medida da mantissa sozinha, a medida do expoente em critério, ajuda a vincular o conjunto solução e a função de custo a comportar-se melhor. Os resultados das simulações mostram o desempenho quanto a estabilidade de loop fechado e que os procedimentos propostos são computacionalmente eficientes com melhor FWL.

5. CONCLUSÃO

Nesta pesquisa foram abordados e estudados todos os conceitos fundamentais acerca da funcionalidade do ESBMC, como por exemplo, a Verificação Formal, a Verificação de Modelos, SMT, BMC, Lógica Proposicional, Lógica Temporal Linear e dentre outros. Para um melhor entendimento, foi essencial a compilação de Função de transferências na transformada Z e em Espaço de Estados. Para isso é utilizado o ESBMC, um verificador limitado de modelos para software escritos na linguagem ANSI-C, baseado nas teorias dos módulos da satisfatibilidade e no mecanismo MATLAB para fazer as devidas alterações de base e validar os resultados.

As validações de testes feitos em Funções de Transferências e de Espaço de Estados foram de fundamental importância para o avanço do projeto, visto que, os resultados gerados neste e em outros tipos de procedimentos saõ comparados para verificar o seu desempenho. O uso do MATLAB foi essencial para projetar o Procedimento de Otimização, pois em seu ambiente foram estruturadas todas as funções além de validá-las. O método proposto permite que o projetista verifique formalmente uma determinada implementação em uma estrutura específica, pois a biblioteca será acoplada a ferramenta de verificação que busca contrapor os modelos inseridos a ela.

Embora sejam utilizadas várias técnicas para minimizar os efeitos FWL, dispositivos mais potentes e representações especiais com um maior número de bits, a sensibilidade dos polos e zeros ao efeito da palavra finita ainda é um desafio. Os trabalhados futuros a serem desenvolvidos nessa pesquisa, visam aumentar a variedade de bibliotecas representadas no modelo operacional e de testes na suíte de teste de cada função, possibilitando assim uma maior exatidão dos cálculos computacionais.

6. REFERÊNCIA BIBLIOGRAFICA

[1] STEPANIAN R, WHIDBORNE J (2001) Digital Controller Implementation and Fragility: A Modern Perspective. Advances in Industrial Control, Springer, London.

[2] YANG G, GUO X, CHE W, GUAN W (2013) Linear Systems: Non-Fragile Control and Filtering. Taylor & Francis.

[3] MASTEN M, PANAHI I (1997) Digital signal processors for modern control systems. Control Engineering Practice 5(4):449 {458, DOI doi: 10.1016/S0967-0661(97)00024-5.

[4] MONMASSON E, CIRSTEA M (2007) FPGA Design Methodology for Industrial Control Systems. IEEE TIE 54(4):1824{1842, DOI 10.1109/TIE.2007.898281.

[5] OPPENHEIM, A. V.; WILLSKY, A. S.; NAWAB, S. H. Signals &Amp; Systems (2Nd Ed.). Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1996. ISBN 0-13-814757-4.

[6] PROAKIS, J. G.; MANOLAKIS, D. G. Digital Signal Processing (3rd Ed.): Principles, Algorithms, and Applications. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1996. ISBN 0-13-373762-4.

[7] RB CORDEIRO LC, F. E. A. Verifying fixed-point digital filters using smt-based bounded model checking. SBrT, 2013.

[8] A. COX, S. SANKARANARAYANAN, and BOR-YUH E. CHANG, "A bit too precise? Bounded verification of quantized digital filters", In TACAS,LNCS 7214, pp. 33-47, 2012.

[9] L. CORDEIRO, B. FISCHER, and J. MARQUES-SILVA, "SMT-based bounded model checking for embedded ANSI-C software", In IEEE Transactions on Software Engineering, vol. 38, no. 4, pp. 957-974, 2012.

[10] R, W. J. I. Digital Controller Implementation and Fragility: A Modern Perspective. [S.l.]: Advances in Industrial Control, Springer, London, 2001.

[11] MORSE, J., CORDEIRO, L., NICOLE, D. and FISCHER, B. Context-Bounded Model Checking of LTL Properties for ANSI-C Software. In: 9TH INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING AND FORMAL METHODS, 2011, LNCS 7041, p. 302-317.

[12] WU, JUN; CHEN, SHENG; WHIDBORNE, JAMES F., CHU, JIAN; Optimal Realizations of Floating-Point Implemented Digital Controllers with Finite Word Length Considerations, International Journal of Control, Volume 77, Issue 5, Mar 2004.

[13] ROWEL, DEREK. Analysis and Design of Feedback Control Systems State-Space Representation of LTI Systems, October 2002.

[14] CLARKE, E.; KROENING, D.; LERDA, F. A Tool for Checking ANSI-C Programs. In: Tools and Algorithms for the Construction and Analysis of Systems. : Springer, 2004. v. 2988, p. 168–176. ISBN 978-3-540-21299-7.

[15] ADRIANA M. TONINI E BRÁULIO R.G.M. Couto, "Ensinando Geometria Analítica com uso do MATLAB," Departamento de Ciências Exatas e Tecnologia do Centro Universitário de Belo Horizonte / DECET - UniBH.

7. ANEXOS

Tabela 1: Testes em Espaço de Estados

	Matrix A	Matrix B	Matrix C	Matrix D	Verification
1	[2.0,-1.0;1.0,5.0]	[0.0;1.5]	[1.0,0.6]	[0.0]	SUCCESSFUL
2	[-2.0,-1.0;1.0,0.0]	[1.0;0.0]	[1.0,2.0]	[1.0]	SUCCESSFUL
3	[-6.0,-11.0,-6.0; 1.0,0. 0,0.0; 0.0,1. 0,0.0]	[1.0; 0.0; 0.0]	[1.0, 9.0, 20.0]	[0.0]	SUCCESSFUL
4	[2.6743, -5.7446, 2.5101,-0.9178; 0.2877, -0.0274,-0.6944,-0.0089; -0.3377,0. 9870,-0.3292,-0.0042; -[0.0830, -0.0032, 0.9191, 0.0010]	[1095900; 638270;3026 20; 74392]	[0.1818,- 0.2831,0.0500, 0.0617]	[0.0]	FAILED
5	[2.3985, -1.8017, 0.40317; 1.0, 0.0,0. 0; 0.0,1. 0,0.0]	[1.0;0.0; 0.0]	[-73.591; 146.6; - 73.018]	[124.50]	SUCCESSFUL
6	[0.0,1.0, 0.0, 0.0, 0.0, 0.0; 0.0, 0.0, 1.0, 0.0, 0.0, 0.0;0.0, 0.0, 0.0, 1.0, 0.0, 0.0;0.0, 0.0, 0.0, 0.0, 1.0, 0.0;0.0, 0.0, 0.0, 0.0, 0.0, 1.0;0.0, -8.11, - 12131.0, -97.8, -463.0, -0.996]	[0.0; 0.0; 0.0; 0.0; 0.0; 1.0]	[19080.0, 90.6, - 576.0, -0.331, 1.65, 0.0]	[0.0]	FAILED
7	[-1.0, 0.0, 0.0; 0.0, -2.0, 0.0; 0.0, 0.0, -3.0]	[1.0; 1.0; 1.0]	[6.0, -6.0, 1.0]	[0.0]	SUCCESSFUL
8	[-5.0, 0.0, 0.0, 0.0; 0.0, -10.0, 0.0, 0.0; 0.0, 0.0, 0.0, 1.0; 0.0, 0.0, -2.0, -2.0]	[1.0; 1.0; 0.0; 1.0]	[2.0; 03.0; 8.0; 8.0]	[0.0]	FAILED
9	[-1.0, 1.0, 0.0; 0.0, -1.0, 0.0; 0.0, 0.0, -3.0]	[0.0; 1.0; 1.0]	[1.5, 1.25, -0.25]	[0.0]	SUCCESSFUL
10	[0.0, 1.0; -2.0, -3.0]	[0.0; 1.0]	[1.0, 0.0]	[0.0]	FAILED

Tabela 2: Testes em Função de Transferência

	Numerador	Denominador	Intervalo	Bits
1	{ 0.0, 0.1, 0.0}	{1.0,4. 0,5. 0}	[-1.0, 1.0]	< 11,12 >
2	{0.0,1. 0, 2.0, 3.0}	{1.0,3. 0,3. 0,1.0}	[-1.0, 1.0]	< 10, 12 >
3	{0.0, 0.0, 4.0, 16.0, -12.0}	{1.0,12. 0,44. 0,0. 0,48. 12}	[-3.0, 3.0]	< 15, 12 >
4	{1.2670, -0.8493}	{1.0, -0.2543}	[-10.0, 10.0]	< 4, 12 >
5	{1.5840, -1.553}	{1.0, -0.96}	[-3.0, 3.0]	< 4, 12 >
6	{0.1, -0.28, 0.26, -0.08}	{1.0, -2.57, 2.18, -0.60}	[-10.0, 10.0]	< 5, 12 >
7	{1.0, 8.0, 23.0, 35.0, 28.0, 3.0}	{0.0, 0.0, 0.1, 0.6, 0.8, 0.0}	[-1.0, 1.0]	< 10, 12 >
8	{2.0, 3.0, 1.0}	{1.0, 4.0, 1.0}	[-1.0, 1.0]	< 4, 12 >
9	{10.0}	{1.0, 8.5, 20.5, 15}	[-10.0, 10.0]	< 15, 12 >
10	{-2.813, 5.719, -2.905}	{1.0, 0.18330, -0.8107}	[-6.0, 6.0]	< 15, 12 >

CRONOGRAMA

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	X									
2	Estudar conceitos de Controladores e Sistemas Digitais.		X	X	X								
3	Estudar as estruturas de realização e a implementação em C.			X	X	X							
4	Análise de controladores digitais e construção de uma suíte de teste.				X	X	X						
5	Análise de *benchmarks* e definição das propriedades para verificar controladores digitais.					X	X	X					
6	Propor uma metodologia de projeto e verificação para controladores digitais.						X	X	X				
7	Avaliar se o uso da realização da forma delta apresentará uma maior capacidade de corretude do que a forma direta							X	X	X			
8	Avaliar se a metodologia de verificação proposta será efetiva e eficiente para verificar controladores digitais do mundo real.								X	X	X		
9	Integrar as implementações no ESBMC.									X	X	X	
	- Elaboração do Resumo e Relatório Final (atividade obrigatória) - Preparação da Apresentação Final para o Congresso (atividade obrigatória)												X