INTEGRAÇÃO DAS DISCIPLINAS DE CONTROLE E SISTEMAS DIGITAIS NO DESENVOLVIMENTO DE UM CPBM EM FPGA USANDO O HDL CODER

Letícia R L Neves – leticiaruy@gmail.com
Universidade Federal do Espírito Santo
Av. Fernando Ferrari, 514, Goiabeiras
CEP 29075-910 - Vitória – ES

José L F Salles – jleandro@ele.ufes.com
Universidade Federal do Espírito Santo
Av. Fernando Ferrari, 514, Goiabeiras
CEP 29075-910 - Vitória – ES

Wagner T da Costa – wagnercosta@ifes.edu.br
Instituto Federal do Espírito Santo
Rod, ES-010 km 6.5, Manguinhos
CEP 29173-087 - Serra - ES

Resumo: *O Controlador Preditivo Baseado em Modelo (CPBM) originou-se no meio industrial, avançando consideravelmente na área química e do petróleo, tendo em vista a capacidade do CPBM em garantir a operação otimizada de processos multivariáveis sujeitos a restrições e com dinâmica não linear. O avanço tecnológico dos processadores digitais e o desenvolvimento de algoritmos de otimização mais eficientes, impulsionou uma série de aplicações do CPBM em controladores de sistemas embarcados. Este trabalho tem por objetivo a implementação de um algoritmo de CPBM em linguagem de descrição de Hardware (HDL) para implementação em Field Programmable Gate Array (FPGA). Para obter resultados em um tempo menor, o código em HDL pode ser gerado a partir do aplicativo HDL Coder, do Matlab.*

Palavras-chave: *Controle preditivo, Sistemas embarcados, FPGA, HDL Coder, Matlab*.

1. INTRODUÇÃO

O avanço tecnológico dos processadores digitais e o desenvolvimento de algoritmos de Controle mais eficientes, impulsionou uma série de pesquisas sobre a aplicação do Controle Preditivo Baseado em Modelo (CPBM) em sistemas com dinâmica rápida encontrados em robótica móvel (Akiba e. al, 2010 e Klancar, 2007) e veículos autônomos (Falcone et al, 2007). Assim, a implantação deste controlador em sistemas embarcados tem motivado alguns pesquisadores a investigar a aplicabilidade do CPBM

em linguagens de HDL para implementação em FPGA (Ling et al., 2006 e Ling et al. 2008).

A solução explicita do CPBM deve ser descrita, em linguagem de *hardware*, de forma que possa ser sintetizada e gravada em uma placa de FPGA. Esta solução normalmente exige a utilização de valores reais, tornando complexa a programação em linguagem de *hardware*. Contudo, é possível utilizar a ferramenta HDL Coder, desenvolvida pela MathWorks. Este aplicativo é capaz de gerar um código em HDL a partir de funções do Matlab ou modelos do Simulink, tornando rápida a implementação de novos algoritmos.

O objetivo deste artigo é mostrar o desenvolvimento do algoritmo CPBM, para realizar o controle dinâmico de uma cadeira de rodas robotizada, embarcado em uma placa de FPGA, usando o HDL Coder para na geração do código. Além disso é mostrada a possibilidade de verificação dos resultados através da técnica de *hardware-in-the-loop.*

2. MOTIVAÇÃO PARA O USO DE FPGA NA IMPLEMENTAÇÃO DO CPBM

A maioria dos circuitos digitais atuais é fisicamente desenvolvida usando dispositivos de circuitos integrados (CIs). Os projetistas podem implementar circuitos usando uma variedade de tipos de CIs disponíveis, sendo que cada tipo possui características específicas que influenciam no tempo e no custo de cada implementação.

Alguns dos tipos de CIs disponíveis para a implementação de circuitos digitais são *full-custom, standar cell, gate array*, FPGA e PLDs (Vahid, 2012, Chu 2006). Dentre estes, o FPGA - *Field Programmable Gate Array*, que foi introduzido por *Xilinx* em meados dos anos 80, tem se destacado por se tratar de um dispositivo lógico que contém dois arranjos bidimensionais de células lógicas genéricas e switches programáveis. As células lógicas (*logic cell*) e as chaves programáveis (*switches*), observadas na Figura 1, podem ser respectivamente, programadas de forma a realizar uma simples função e utilizadas para fornecer interconexões entre células lógicas. Assim projetos complexos podem ser elaborados através da especificação da função de cada célula lógica e definição da conexão de cada *switch* programável.

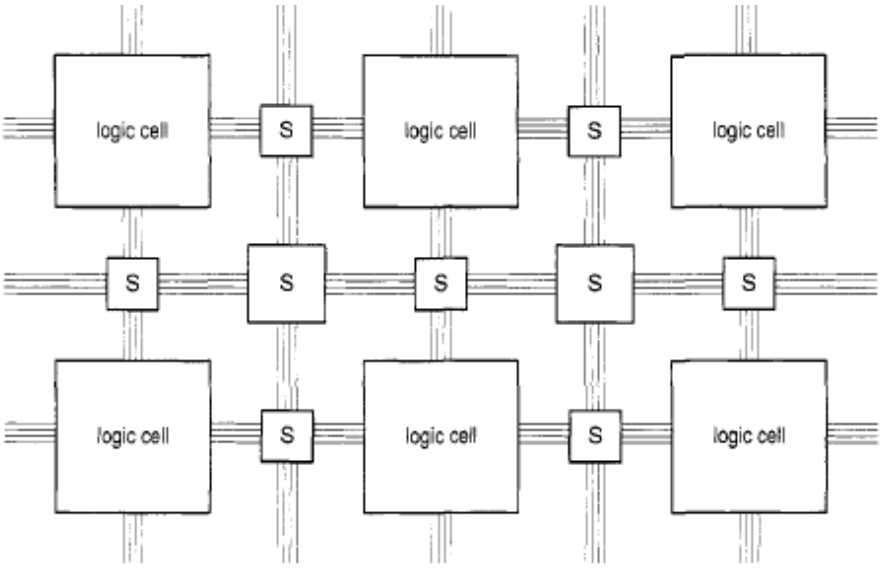

Figura 1 – Estrutura interna de um FPGA

Uma vez que o problema é descrito por uma HDL, um simples cabo adaptador pode ser utilizado para fazer o download das configurações das células lógicas e chaves para o dispositivo FPGA, obtendo um circuito adaptado.

A principal vantagem dos FPGAs é a possibilidade da computação do controle digital de forma paralela, resultando em uma velocidade muito rápida de operação. Além disso, sistemas embarcados como este, permitem reduzir espaço físico, recursos computacionais e custo do produto.

2.1. Linguagens de Descrição de Hardware

Um sistema digital pode ser descrito em diferentes níveis de abstração e a partir de diferentes pontos de vista. Uma HDL pode descrever e modelar um circuito com precisão, até mesmo na fase de desenvolvimento, para a visualização estrutural ou comportamental do mesmo, em vários níveis diferentes de abstração desde o nível de transistores até o nível de processadores.

É importante salientar que programar um FPGA significa simplesmente escrever uma série de bits nas memórias do chip. Esta atividade não deve ser confundida com a programação de *software*s em alto nível.

Como o próprio nome indica, a linguagem descreve como as unidades lógicas devem se comportar e o sintetizador projeta o arranjo físico das unidades lógicas.

As principais linguagens de descrição de *hardware* usadas em FPGAs são VHDL e Verilog. No desenvolvimento deste trabalho foi utilizada a linguagem VHDL, que significa *Very High Speed Hardware Description Language*, e o Xilinks como *software* para síntese do mesmo.

2.2. Controle Preditivo

Os algoritmos baseados em CPBM requerem rotinas de otimização e cálculos matriciais que consomem elevado tempo computacional e memória, de maneira que pode limitar suas aplicações em sistemas embarcados com dinâmica rápida, como os encontrados em robótica, aeronáutica, etc.

De uma forma geral, o controlador preditivo é um algoritmo que calcula valores futuros de entrada e saída baseados em um Horizonte de Previsão e sinais de controle baseados em um Horizonte de Controle. Os sinais de controle são resultados da otimização de um critério, que é uma função do erro entre as saídas previstas e as referências previstas, chamada função custo ou função objetivo. Essa otimização é um algoritmo que leva em consideração a função custo e todas as restrições impostas ao sistema. A partir do conjunto de sinais de controle obtidos, descartam-se os valores previstos e aplica-se ao sistema o valor referente ao sinal do instante atual (estratégia de controle com horizonte retrocedente). A Figura 2 ilustra a estratégia de um controlador preditivo (CAMACHO & BORDONS, 2004).

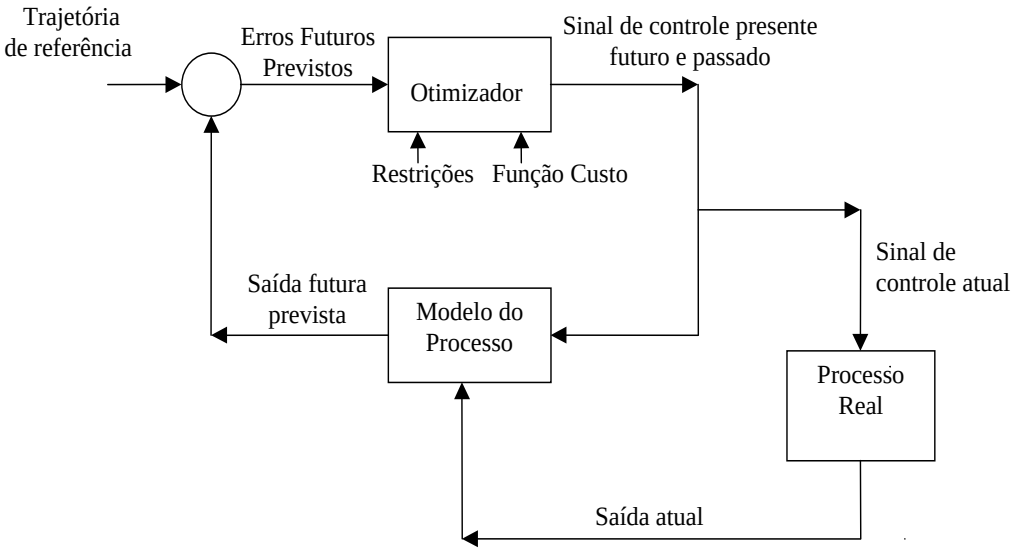

Figura 2 - Estratégia de um controlador preditivo. Adaptado de (CAMACHO & BORDONS, 2004)

Os métodos de otimização usuais realizam todos os seus cálculos a cada iteração, o que, dependendo da quantidade de entradas, saídas e restrições, inviabiliza seu uso para sistemas com dinâmicas rápidas. O método de otimização por Programação Multiparamétrica (PM) (BEMPORAD e MORARI, 1999), realiza todos os cálculos necessários para a obtenção das ações de controle explicitamente ou *off-line*, ou seja, antes do sistema estar em operação. Isso quer dizer que, antes do sistema entrar em operação, já serão de conhecidos todos os valores factíveis de ações de controle, trazendo agilidade para o controle preditivo e um menor dispêndio de poder computacional (KVASNICA, 2009).

A solução explícita do CPBM obtida através da PM é descrita por uma função afim por partes definida sobre uma região poliedral, ou seja, um sistema de equações lineares por partes. Estas funções lineares podem ser implementadas em controladores FPGA, de maneira que o mesmo atue sobre um sistema com dinâmica rápida.

2.3. O controlador do Robô

O robô é uma cadeira móvel com motores e um computador de bordo para realização do controle cinemático da trajetória. Seu equacionamento leva em consideração o modelamento dos motores, a dinâmica entre os motores e as rodas e o modelamento dos controladores de baixo nível (PID).

O controlador embarcado recebe as referências geradas no controlador cinemático para executar o controle da velocidade angular e linear dos motores, ou seja, o controle dinâmico. O modelo matemático usado pelo controlador preditivo para realizar o controle dinâmico do robô (Celeste, 2009 & Martins et al, 2008) é mostrado abaixo:

$$\begin{cases} v_{ref} = \theta_1 \dot{v} + \theta_3(-\omega^2) + \theta_4 v + \theta_7(-\dot{\omega}) - \delta_v \\ \omega_{ref} = \theta_2 \dot{\omega} + \theta_5 v\omega + \theta_6 \omega + \theta_8(-\dot{v}) - \delta_\omega \end{cases} \quad (1)$$

Onde θi, i = 1,2, ...,8, são parâmetros que dependem unicamente de características físicas do sistema, δv e δω são perturbações inerentes a atuação de forças e torques externos e do deslizamento lateral das rodas de tração e vref e ωref são referências dos controladores PIDs dos motores da cadeira de rodas. O cálculo de vref e ωref é feito pelo controlador dinâmico, que neste projeto é o controlador preditivo (Camacho e Bordons, 2009), formulado através do seguinte problema de otimização:

$$
\begin{aligned}
J^*_{h_p}(x_t, U) = \quad & \min_U \left\{ x'_{t+h_p|t} P x_{t+h_p|t} + \sum_{k=0}^{h_p-1} \left\| Q_x x_{t+k|t} \right\|_2 + \left\| Q_u u_{t+k|t} \right\|_2 \right\} \\
& \sum_{k=0}^{h_p-1} \left\| Q_x x_{t+k|t} \right\|_2 + \left\| Q_u u_{t+k|t} \right\|_2 \Big\} \\
& subj.to \\
& u_{\min} \le u_{t+k|t} \le u_{\max}, \forall k = 0, \dots, h_c \\
& y_{\min} \le y_{t+k|t} \le y_{\max}, \forall k = 0, \dots, h_p \\
& x_{t+k+1|t} = A x_{t+k|t} + B u_{t+k|t} \\
& y_{t+k|t} = C x_{t+k|t} \\
& u_{t+k|t} = \mathrm{K} x_{t+k|t}, h_c \le k \le h_p
\end{aligned}
\tag{2}
$$

Onde h_p, h_c, h_r representam o horizonte de controle e previsão, respectivamente e satisfaz $h_p \ge h_c$ e $h_p - 1 \ge h_r$; $U = (u_{t|t}, \dots, u_{t+h_c|t})^T$ é a sequência de controle ótimo a ser determinado, P, Q_x, Q_u são matrizes positivas definidas, e K representa a realimentação de estado para estabilização do peso. O problema mostrado na equação (2) é resolvido a cada instante de tempo t, começando a partir do Vetor de Estado Preditivo $x_{t+k}, k = 1 \cdots h_p$. A Lei de Controle Ótimo correspondente é obtida pela equação $U^* = (u^*_{t|t}, \dots, u^*_{t+h_c|t})^T$. O sinal utilizado no processo é a Primeira Lei de Controle $u_t = u^*_{t|t}$ e x é um vetor que representa as velocidades angular e linear e U é a referência das velocidades linear e angular calculadas pelo controlador preditivo.

A solução do problema é obtida explicitamente de acordo com (Kvasnica, 2009), sendo dada por uma função do tipo:

$$
U^*(x) = L_r x + C_r, \qquad se \ x \in \mathrm{P}_r \tag{3}
$$

ou seja,

$$
\begin{bmatrix} vref \\ wref \end{bmatrix} = \begin{cases}
\begin{bmatrix} -1.895 & -0.045 \\ -0.580 & -1.090 \end{bmatrix} X + \begin{bmatrix} 1.164 \\ 1.190 \end{bmatrix} se \begin{bmatrix} -1.000 & -0.024 \\ -0.470 & -0.883 \end{bmatrix} X \le \begin{bmatrix} -0.403 \\ -0.559 \end{bmatrix} \\
Região\ 1 \\
\begin{bmatrix} 0.000 & 0.000 \\ -0.215 & -1.082 \end{bmatrix} X + \begin{bmatrix} 0.400 \\ 1.043 \end{bmatrix} se \begin{bmatrix} -0.997 & -0.082 \\ -0.195 & -0.981 \end{bmatrix} X \le \begin{bmatrix} -0.214 \\ -0.492 \end{bmatrix} \\
Região\ 2 \\
\dots \\
\begin{bmatrix} 0.000 & 0.000 \\ 0.000 & 0.000 \end{bmatrix} X + \begin{bmatrix} 0.400 \\ 0.500 \end{bmatrix} se \begin{bmatrix} 0.963 & 0.269 \\ -1.000 & 0.026 \end{bmatrix} X \le \begin{bmatrix} 0.158 \\ -0.219 \end{bmatrix} \\
Região\ 14
\end{cases}
\tag{4}
$$

onde $X = [v \ \omega]^T$

3. A FERRAMENTA HDL CODER

O Matlab é um ambiente de programação amplamente utilizado em diversas áreas de engenharia e ciências exatas. Suas principais aplicações são para análise de dados, solução de problemas e projeto de algoritmos.

Nos últimos anos, a MathWorks, desenvolvedora do Matlab, tem buscado produzir ferramentas para linguagem de desenvolvimento de sistemas. Uma destas é o HDL Coder, que tem por finalidade gerar um código sintetizável em VHDL ou Verilog diretamente de um código do Matlab.

Para muitos programadores, inclusive os mais acostumados com paradigmas de programação, desenvolver algoritmos para FPGA pode ser considerado um desafio, uma vez que o desenvolvimento de *hardware* exige o entendimento e a visualização do processamento paralelo. Já a linguagem de programação do Matlab, que recebe este mesmo nome, é uma das preferidas entre os programadores devido a sua simplicidade. Trata-se de uma linguagem intuitiva, de sintaxe concisa e que permite a utilização de matrizes complexas e criação de gráficos com facilidade.

3.1. Fluxo para a tradução de um código

As etapas iniciais para a tradução de um código consistem na programação e simulação do projeto no Matlab/Simulink para verificar se o mesmo não apresenta erros de execução e se funciona conforme esperado. Isto é feito através da criação de dois arquivos: um contendo uma *Matlab function* e o outro contendo um *Matlab TestBench*. Após a verificação, se cria um novo projeto no HDL Coder, adiciona estes dois arquivos e clica no botão "*Work Flow Adivisor*", que abrirá uma nova janela, conforme mostra a figura 3.

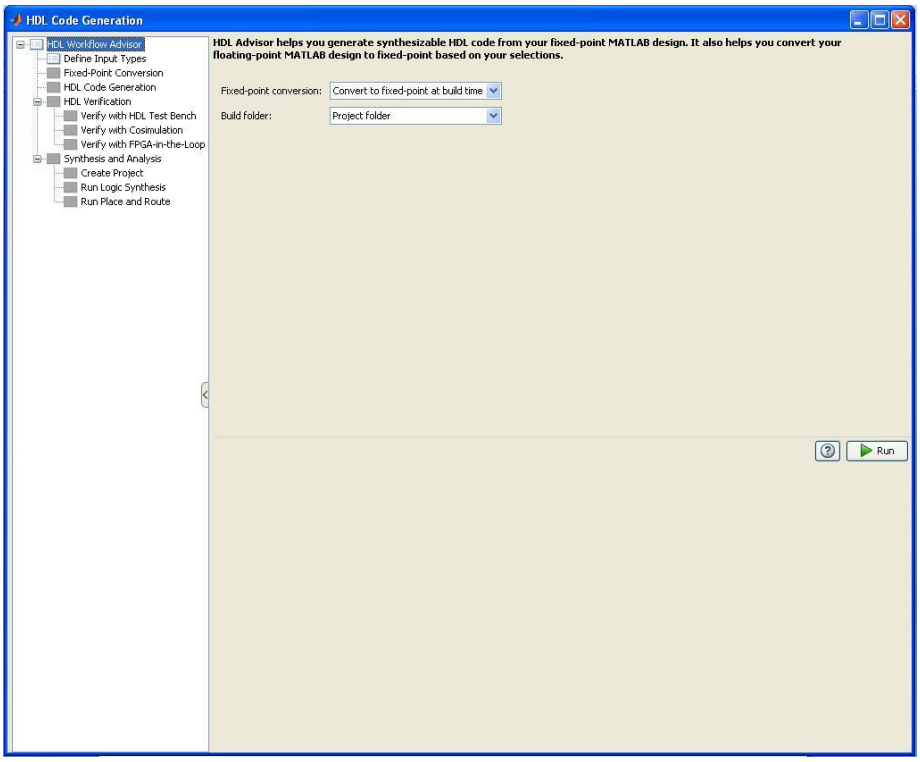

Figura 3 – Tela inicial do HDL Coder

Geração do código HDL em ponto fixo

Grande parte das aplicações de engenharia consiste em algoritmos especificados usando operações em ponto flutuante, porém por questões de consumo de energia, custo e desempenho, os circuitos devem ser preferencialmente implementados em ponto-fixo. Por isso o aplicativo transforma os dados em ponto flutuante para ponto fixo. Uma vez que a opção "*Convert to fixed-point at build time*" já vem selecionada, basta clicar no botão "**Run**", apresentado na figura 3.

Feito isto e se não houver nenhum erro, o programa define automaticamente os tipos de dados das entradas de acordo com os valores apresentados no arquivo contendo o *Matlab TestBench*, faz a conversão para ponto fixo e gera o código HDL no formato VHDL ou Verilog, conforme escolhido durante a execução.

A medida que o aplicativo vai sendo executado, a janela do *Work Flow Advisor* mostra o status de cada operação. Caso não ocorra nenhum erro, ao final da geração do código teremos uma janela conforme mostrado na figura 4.

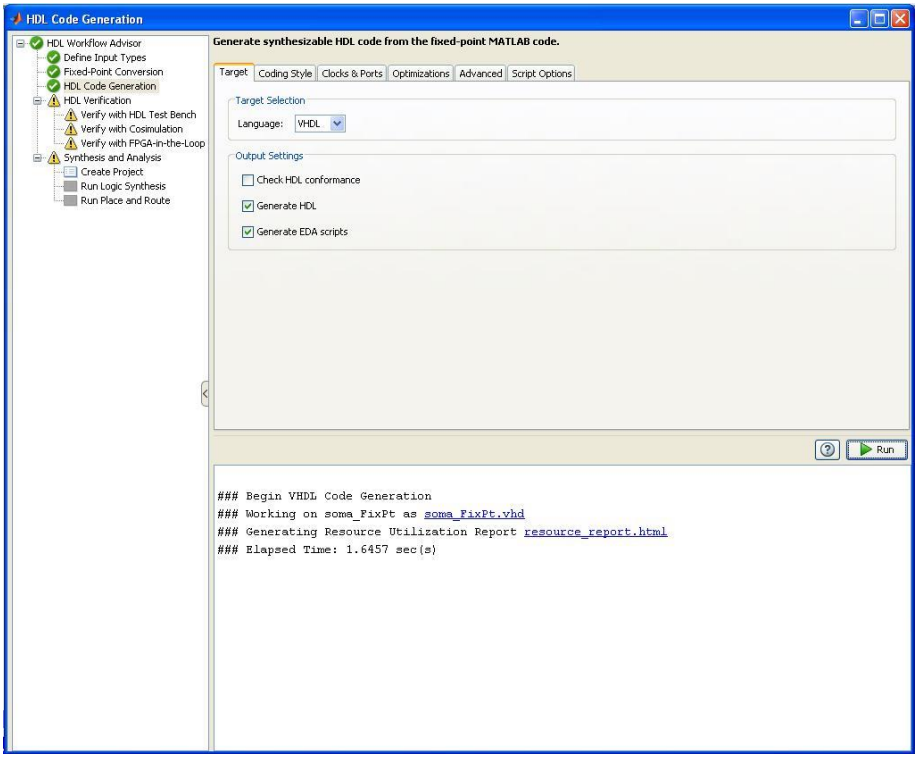

Figura 4 – Tela do HDL Coder ao finalizar a Geração do Código

Neste momento, o código HDL já estará criado e salvo na pasta atual do Matlab, podendo ser acessado diretamente seguindo os links da janela de registros, conforme mostrado na figura 5.

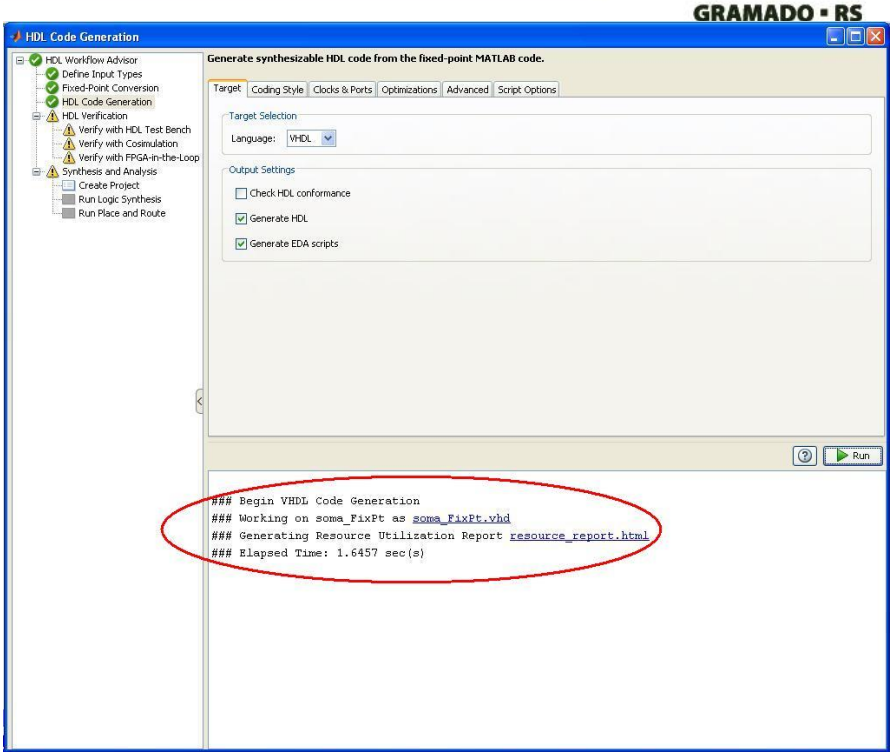

Figura 5 – Link para acesso ao código HDL automaticamente gerado

Tendo o código HDL gerado, o desenvolvedor possui duas opções a serem seguidas. A primeira é continuar usando o HDL Coder para a Verificação, Síntese e Análise do código, bastando seguir os próximos passos do *Work Flow Advisor*, e a segunda opção, utilizada neste trabalho, é utilizar o arquivo gerado em um ambiente de desenvolvimento próprio para HDL, como o Xilinks por exemplo. Com o código neste ambiente, basta fazer o mapeamento dos pinos de entrada e saída da placa e salvar o código na mesma. Assim o controlador está pronto para ser aplicado no robô.

4. USO DA FERRAMENTA EM SALA DE AULA

O curso de Engenharia Elétrica da Instituição é voltado para a área de Controle e Automação. Os alunos tinham uma grande dificuldade de implementar algoritmos de controle em código VHDL devido a carga horária da disciplina de Sistemas Digitais para o desenvolvimento deste algoritmos, pois é nesta disciplina que se aprende a linguagem VHDL. Quando os alunos se matriculam na disciplina de Sistemas Digitais, eles já cursaram pelo menos duas disciplina de controle com isto já sabem utilizar o *software* Matlab® para desenvolver e simular projetos de controle. Com estes conhecimentos prévios dos alunos, observou-se uma vantagem de integrar a ferramenta do HDL Coder para construções de aplicações de controle em VHDL. Esta integração melhorou o rendimento das aulas, houve um maior interesse dos alunos, o número de aplicações com FPGA aumentou significativamente, além do desenvolvimento de vários projetos de controle com algoritmos em VHDL.

5. CONCLUSÃO

A união das duas diferentes áreas de estudo, a de sistemas de controle e a de sistemas digitais, torna possíveis aplicações industriais como a obtida neste projeto e a

ferramenta HDL Coder torna muito mais fácil e rápido o processo de desenvolvimento do mesmo. Estes são motivos de grande motivação para os pesquisadores.

A ferramenta HDL Coder é fundamental na implementação da função do controlador preditivo, dada na equação 4, pois o desenvolvimento na forma tradicional de escrita de código demanda um esforço muito grande para a representação das matrizes em ponto fixo e um tempo elevado é gasto na programação.

Tendo o código HDL, recomenda-se ainda a verificação do código através de outra ferramenta, o *HDL Verifier*, que foge ao escopo deste trabalho, mas faz parte dos objetivos futuros da pesquisa. Esta ferramenta é capaz de automatizar a verificação do projeto, simulando o *testbench* do FPGA em loop com o computador. Com isto, após salvar o código HDL em uma placa de FPGA, é possível simular o código do CPBM como um bloco do Simulink e assim verificar, antes da aplicação direta no robô, a eficácia do controlador atuando em uma planta, cujo código de simulação deve estar previamente salvo no Simulink.

6. REFERÊNCIAS BIBLIOGRÁFICAS

Livros:

VAHID, Frank. Digital Design with RTL Design, VHDL and Verilog. 2. ed. Wiley. 2012

CHU, P.P. FPGA Prototyping by VHDL examples. John Wiley & sons, New Jersey. 2008

CAMACHO, E.F. and BORDONS,C. Model Predictive Control. 2. ed. Springer Verlag.

KVASNICA, M. Real-Time Model Predictive Control via Multi-Parametric Programming. 1ª. ed. Saarbrücken, Alemanha: VDM. 2009.

Artigos de periódicos:

TOZZI, M.; OTA, J. Vertedouro em degraus. Revista da Vinci, Curitiba, v.1, n.1, p. 9-28, 2004.

KLANKAR, G. and SKRJANC, I., Tracking-error model based predictive control for mobile robots in real time, Robotics and Automation System, 55, pp 460-469, 2007

FALCONE, P., BORRELIi, F, ASGARI, J., Tseng, H. E. and HROVAT, D. Predictive Active Steering Control for Autonomous Vehicle System, IEEE Transactions Systems Technology, 15, 566-580. 2007

BEMPORAD, A.; MORARI, M. Control of Systems Integrating Logic, Dynamics, and Constraints. Automatica, v. 35(3), p. 407-427, Março 1999.

Monografias, dissertações e teses:

CELESTE, W. C. Um sistema autônomo para navegação de cadeiras de rodas robóticas orientadas a pessoas com deficiência motora severa. PPGE. UFES. [S.l.]. 2009. Tese (Doutorado)

Trabalhos em eventos
AKIBA, Z., Nanma, T. and Ishida, M. Model predictive control based optimal crusing control of two-wheeled mobile robot,2010 IEEE Conference on Robotics, Automation and Mechatronics. 2010

LING, K.V. , Wu, B.F. and Maciejowisk, J.M.(2006) A FPGA Implementation of Model Predictive Control. American Control Conference, USA

LING, K.V. , Wu, B.F. and Maciejowisk, J.M.(2008) Embedded Model Predictive Control (MPC) Using FPGA. 17 th IFAC World Congress, Coréia do Sul.

Internet:
SHURE, L. **Matlab to FPGA using HDL Coder**. Disponível em:< http://blogs.mathworks.com/loren/2013/04/11/matlab-to-fpga-using-hdl-codertm/> Acesso em: 15 jun. 2013

INTEGRATION OF SUBJECTS CONTROL SYSTEMS AND DIGITAL SYSTEMS IN THE DEVELOPMENT OF MBPC IN FPGA USING HDL CODER

Abstract: *The Model Based Predictive Control (MBPC), originated in the industrial environment, advanced considerably the chemical and petroleum areas, in view of the ability of MBPC to ensure optimal operation of multivariable processes subject to restrictions and nonlinear dynamics. Technological advances in digital processors and development of algorithms with optimization more efficient, boosted a number of applications of MBPC controllers in embedded systems. This paper aims at implementing an algorithm MBPC in Hardware Description Language (HDL) for implementation in Field Programmable Gate Array (FPGA). For results in a shorter time, the HDL code can be generated from the application HDL Coder, from Matlab.*

Key-words: *Predictive controller, Embedded systems, FPGA, HDL Coder, Matlab.*