UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA MECÂNICA
COMISSÃO DE PÓS-GRADUAÇÃO EM ENGENHARIA MECÂNICA

Utilização de Ferramentas de Prototipagem Rápida direcionada à Concepção de Sistemas Embarcados baseados em Computação Reconfigurável

Autor: Wlamir de Almeida Passos
Orientador: Prof. Dr. João Maurício Rosário

66/2008

UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA MECÂNICA
COMISSÃO DE PÓS-GRADUAÇÃO EM ENGENHARIA MECÂNICA
DEPARTAMENTO DE PROJETO MECÂNICO

Utilização de Ferramentas de Prototipagem Rápida direcionada à Concepção de Sistemas Embarcados baseados em Computação Reconfigurável

Autor: Wlamir de Almeida Passos
Orientador: Prof. Dr. João Maurício Rosário

Curso: Engenharia Mecânica
Área de Concentração: Mecânica dos Sólidos e Projeto Mecânico

Dissertação de mestrado acadêmico apresentada à comissão de Pós Graduação da Faculdade de Engenharia Mecânica, como requisito para a obtenção do título de Mestre em Engenharia Mecânica.

Campinas, 2008.
SP – Brasil

FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DA ÁREA DE ENGENHARIA E ARQUITETURA - BAE - UNICAMP

Passos, Wlamir de Almeida
P268u Utilização de ferramentas de prototipagem rápida direcionada à concepção de sistemas embarcados baseados em computação reconfigurável / Wlamir de Almeida Passos. --Campinas, SP: [s.n.], 2008.

Orientador: João Maurício Rosário
Dissertação (mestrado) - Universidade Estadual de Campinas, Faculdade de Engenharia Mecânica.

1. Sistemas embutidos de computador. 2. Controle preditivo. 3. Robótica. 4. Robôs moveis. 5. Simulação (Computadores). 6. Robôs – Sistema de controle. I. Rosário, João Maurício. II. Universidade Estadual de Campinas. Faculdade de Engenharia Mecânica. III. Título.

Título em Inglês: Utilization of rapid prototyping tools for conception of embedded systems based on reconfigurable computation
Palavras-chave em Inglês: Reconfigurable computing, Control, Robotics, Modeling
Área de concentração: Mecânica dos Sólidos e Projeto Mecânico
Titulação: Mestre em Engenharia Mecânica
Banca examinadora: Humberto Ferasoli Filho, Niederauer Mastelari
Data da defesa: 27/06/2008
Programa de Pós-Graduação: Engenharia Mecânica

UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA MECÂNICA
COMISSÃO DE PÓS-GRADUAÇÃO EM ENGENHARIA MECÂNICA
DEPARTAMENTO DE PROJETO MECÂNICO

DISSERTAÇÃO DE MESTRADO

Utilização de Ferramentas de Prototipagem Rápida direcionada à Concepção de Sistemas Embarcados baseados em Computação Reconfigurável

Autor: Wlamir de Almeida Passos
Orientador: Prof. Dr. João Maurício Rosário

A Banca Examinadora composta pelos membros abaixo aprovou esta Dissertação:

Prof. Dr. João Maurício Rosário, Presidente
UNICAMP

Prof. Dr. Humberto Ferasoli Filho
UNESP/Bauru

Prof. Dr. Niederauer Mastelari
UNICAMP

Campinas, 27 de junho de 2008

Dedicatória:

Dedico este trabalho à minha esposa e filha.

Agradecimentos

Presto a minha mais sincera homenagem a todos aqueles que de forma direta ou indireta ajudaram a construir este trabalho.

A minha família, pelo carinho, contínuo apoio e motivação dados durante estes anos.

Ao pessoal do Laboratório de Automação Integrada e Robótica da Faculdade de Engenharia Mecânica da Unicamp, em especial ao meu amigo e orientador Prof. Dr. João Maurício Rosário pelos valiosos conselhos e orientação, bem como pela confiança em mim depositada.

A todos os professores e funcionários do Departamento de Projeto Mecânico da Faculdade de Engenharia Mecânica da UNICAMP que ajudaram de forma direta e indireta na conclusão deste trabalho.

A empresa Altera e seus representantes, em especial ao Eng. Neimar Marques Duarte, pelo suporte oferecido.

Ao amigo e companheiro de trabalho Prof. Wellington Roberto Branco (in memoriam).

"Há muitas maravilhas neste mundo, mas a maior de todas é o homem.
Sófocles

Resumo

PASSOS, Wlamir de Almeida, Utilização de Ferramentas de Prototipagem Rápida direcionada à Concepção de Sistemas Embarcados baseados em Computação Reconfigurável, Campinas: Faculdade de Engenharia Mecânica, Universidade Estadual de Campinas, 2008. Dissertação (Mestrado).

Durante os últimos anos tem ocorrido uma grande evolução tecnológica na área de sistemas embarcados, abrangendo inovações tanto em hardware como em software. Tais inovações permitem o desenvolvimento de novas metodologias de projeto que levem em conta a facilidade de futuras modificações, modernizações e expansões do sistema projetado. Este trabalho apresenta um estudo de novas ferramentas de projeto para sistemas baseados em lógica reconfigurável. A principal motivação deste trabalho é a apresentação de ferramentas para prototipagem rápida baseadas em computação reconfigurável para implementação de protótipos de sistemas Embarcados. Será abordada a evolução deste tipo de tecnologia bem como os *softwares* de desenvolvimento disponíveis. Será também abordado um estudo de caso na área de Acionamento e Controle de Sistemas Mecatrônicos bem como a sua implementação e teste utilizando-se a técnica HIL - *Hardware In the Loop*.

Palavras Chave

Lógica Reconfigurável, Sistemas Embarcados, Robótica Móvel, Modelagem, Controle.

Abstract

PASSOS, Wlamir de Almeida, *Utilization of Rapid Prototyping Tools for conception of Embedded Systems based on Reconfigurable Computation*, Campinas: Faculdade de Engenharia Mecânica, Universidade Estadual de Campinas, 2008. Dissertação (Mestrado)

In the last years, there has been a great technologic evolution in the embedded systems area, covering innovations both in hardware and software. Those innovations allow the development of new project methodologies that considers the facility of future modifications, upgrades and expansions for the system. This work shows a study about new design tools made for systems based on reconfigurable computation. The main motivation of this work is the presentation of tools for fast prototyping based on prototyping seeking the implementation of prototypes of embedded systems. It will be discussed the evolution of this type of technology as well as the available development software. It will be also discussed a case study on mechatronic control system as well as it's implementation and tests using Hardware In the Loop (HIL) techniques.

Keywords:

Reconfigurable Computing, Embedded Systems, Mobile Robotics.

Índice

Lista de Figuras

Lista de Tabelas

Nomenclatura

Abreviações

AHDL - Altera Hardware Description Language

ALM - Adaptive Logic Modules

API – Application Program Interface

ASIC - Application Specific Integrated Circuits

CAD - Computer Aid Design

CCM - Custom Computing Machines

CPLD - Complex Programmable Logic Devices

DBF - Block Design File

DFGA - Dinamically Field Programmable Gate Array

DSP – Digital Signal Processor

EPLD – Erasable Programmable Logic Device

FCCM – FPGA based Custom Computing Machines

FFT – Fast Fourier Transform

FIR – Finite Impulse Response

FPGA - Field Programmable Gate Array

GAL - Generic array logic

HIL - Hardware In-the-Loop

IEEE - Institute of Electrical and Electronics Engineers, Inc.

IIR – Infinite Impulse Response

IP-CORE – Intellectual Property Core

JTAG - Joint Test Action Group

PAL - Programmable array logic

PID – Proporcional Integral Derivativo

PLDs - Programmable Logic Devices

PLL – Phase-Locked Loops

PWM - Pulse-Width Modulation

SOPC - System On Programmable Chip

VHDL - VHSIC Hardware Description Language

Capítulo 1

Introdução

O desenvolvimento de protótipos de sistemas embarcados tem sido alvo de inúmeros trabalhos na área de Controle e Automação. Dentre as inúmeras aplicações de sistemas embarcados ou que utilizam sistemas embarcados, podemos destacar diferentes dispositivos que integram sensores e atuadores para a execução de alguma tarefa e que exigem maior ou menor capacidade de controle como: impressoras e periféricos de computadores, automóveis, robôs industriais, robôs móveis, aparelhos de telefonia, cadeiras de rodas, próteses antropomórficas, etc. Nesses sistemas são geralmente utilizados microcontroladores com algoritmos desenvolvidos em várias linguagens de software para realizar o controle destes dispositivos. Mais recentemente, alguns destes algoritmos foram também desenvolvidos em trabalhos associados com computação reconfigurável, que apresentam vantagens em termos de desempenho.

Entre todos os campos associados com projeto de sistemas embarcados, as tecnologias de *software* e *hardware* são as que têm experimentado a mais rápida evolução. É muito grande a quantidade de novos microprocessadores, interfaces de comunicação, interfaces de potência, sensores, compiladores, sistemas operacionais e ambientes de desenvolvimento disponibilizados no mercado. Em função desta acelerada evolução tecnológica, a idéia de se utilizar estruturas abertas e reconfiguráveis, que possam adaptar-se a novas demandas, torna-se muito atraente, sendo mesmo um pré-requisito na consideração de um projeto de um sistema embarcado.

O conceito de sistemas abertos tem sido estudado nos últimos anos por várias instituições no campo de máquinas ferramentas e engenharia de produção, focalizando os aspectos de modularidade, os efeitos da arquitetura de controle e da rede de comunicação no desempenho do sistema.

O objetivo deste conceito aplicado à arquitetura reconfigurável é permitir uma fácil e rápida adaptação de dispositivos embarcados a novas evoluções tecnológicas, para uma melhor portabilidade e capacidade de intercâmbio para o sistema final. A divisão da estrutura em pequenos blocos funcionais, com interfaces bem especificadas, permite uma melhor definição das tarefas de uma equipe de projeto multidisciplinar, bem como a rápida adaptação de um determinado bloco a uma nova evolução tecnológica. Este tipo de solução não pode ser realizado em muitos dos sistemas comerciais disponíveis, pois estes não fornecem informações ou não apresentam recursos para permitir este tipo de manipulação.

Este trabalho tem como objetivo a utilização de ferramentas para implementação de novas arquiteturas de controle para sistemas mecatrônicos a fim de torná-los mais precisos e eficientes na execução de tarefas, aliando conceitos de prototipagem rápida para a implementação de controladores, a partir da apresentação de proposta de ambiente baseado em computação reconfigurável para implementação em protótipos de sistemas embarcados. As novas tecnologias envolvendo a concepção de microprocessadores, interfaces de potência e comunicação, aliados ao conceito de prototipagem rápida possibilitam pesquisas científicas, onde a implementação de controladores requer menores custos.

Dentro do contexto proposto para desenvolvimento dessa Dissertação de Mestrado são considerados aspectos de hardware e software, e a principal motivação deste trabalho é propiciar um ambiente que facilite o desenvolvimento de protótipos de sistemas embarcados, enfatizando o desenvolvimento de ferramentas simples que permitam o controle e a monitoração dos diferentes sensores e atuadores nesta categoria de projeto. Como conseqüência, é proposto o

desenvolvimento de pequenos módulos independentes com interfaces de comunicação bem definidas como parte de uma estrutura orientada a uma arquitetura aberta.

Neste capitulo apresentaremos um panorama geral do trabalho desenvolvido, sendo apresentado um estado da arte do conceito de lógica Reconfigurável, justificando o desenvolvimento desse projeto de pesquisa, seus objetivos gerais e específicos, a necessidade de capacitação nesta importante área e a forma como foi delimitada esta questão, além da estrutura geral dos capítulos da tese.

1.1 Apresentação do Problema

A Mecatrônica integra conhecimentos nas diferentes áreas de engenharia, matemática, computação, permitindo o desenvolvimento com maior rapidez de sistemas compostos de uma estrutura mecânica complexa, geralmente composta por atuadores, sensores de posicionamento e um sistema de acionamento e controle, inserindo dentro desse contexto a Robótica.

Robôs são encontrados nos diversos meios, desenvolvendo as mais diversas funções, em atividades insalubres, educacional, ou simplesmente em trabalhos que requerem força e agilidade. Cada grau de liberdade é geralmente composto por atuadores, sensores de posicionamento e pelo sistema de acionamento e controle.

Os elementos básicos do manipulador como estrutura, eixos, mancais, transmissão, devem ser dimensionados do ponto de vista de resistência e rigidez, com o objetivo de atingir um volume de trabalho e precisão mecânica exigidos numa operação automatizada. Os sensores de posição não são capazes de medir possíveis deformações estruturais, como também atritos e folgas nas juntas.

A robótica industrial possibilita que produtos sejam realizados de maneira mais rápida e eficiente do que no passado. Para isso, o desenvolvimento da mecânica de precisão quanto da eletrônica foi indispensável para que isso ocorresse. Mas isso pode ser melhorado, principalmente

no que se refere ao tempo de execução de diferentes tarefas, para tal, a melhoria do controle de robô com as grandes possibilidades de interações que um ambiente automatizado pode oferecer.

Dentre os sistemas embarcados, os robôs móveis têm-se apresentado como plataformas para consolidação de conhecimento em diversas áreas de pesquisa, como modelagem, controle, automação, sistemas de potência, sensores, transmissão de dados, eletrônica embarcada e engenharia de *software*, sendo cada vez mais usadas em instituições de ensino e pesquisa.

O estudo de novas arquiteturas de controle torna-se muito importante para que haja uma melhoraria no desempenho de robôs industriais face às mudanças bruscas de parâmetros associados à posição, velocidade e aceleração, durante a realização de uma determinada trajetória. A Automação Flexível possibilita que robôs sejam cada vez mais rápidos podendo interferir em um grande volume de trabalho, dessa forma, novas técnicas de controle são estudadas com a finalidade de torná-los mais eficientes.

Mais especificamente na área de controle, se faz necessária uma ferramenta que possibilite a simulação e a rápida implementação da metodologia aplicada, validando assim o projeto na prática. Esse tipo de ferramenta pode ser extremamente útil na área de desenvolvimento industrial para a validação de estudos e comprovação de sua eficiência.

As validações dos conceitos e ferramentas de prototipagem rápida direcionada a Conceitos de sistemas embarcados baseadas em lógica reconfigurável desenvolvidos nos capítulos desta dissertação de mestrado foram realizadas no Laboratório de Automação Integrada e Robótica da UNICAMP através da prototipagem de um controlador de junta robótica.

1.2 Motivação

Considerando-se o atual cenário tecnológico e o alto desenvolvimento de soluções cada vez mais integradas para concepção e validação de dispositivos mecatrônicos, a utilização de ferramentas de prototipagem rápida para concepção do sistema de controle torna-se uma indispensável opção para agilizar e facilitar a implementação de diferentes arquiteturas de controle, possibilitando assim a redução do tempo de finalização de um projeto e ou estudo cientifico, atendendo uma eficiência desejada.

A principal a motivação para o desenvolvimento desta dissertação de mestrado deve-se ao fato da necessidade de uma metodologia para a concepção e implementação de projetos na área de mecatrônica, visando permitir de forma mais rápida e com significativa redução de custos, a utilização mais adequada das tecnologias que compõem a área.

1.3 Objetivos do Trabalho

Considerando-se que a implementação de um sistema de controle, envolvendo o gerenciamento de entradas e de saídas, envolve muito tempo de concepção e alto custo de implementação. Neste trabalho de pesquisa são apresentadas ferramentas para concepção de controladores de juntas robóticas utilizando-se o conceito de prototipagem rápida. Conseqüentemente, o principal objetivo desta dissertação de mestrado é a apresentação de metodologias de Prototipagem Rápida de Controladores de juntas robóticas baseadas em lógica reconfigurável, através de ambiente voltado à capacitação e desenvolvimento de projetos na área de controle e automação industrial, onde os principais conceitos possam ser verificados e posteriormente implementados e validados na prática, fornecendo subsídios para a análise e estratégias para estas aplicações.

A utilização destas ferramentas possibilita a implementação, validação e testes de diferentes técnicas de controle baseadas no conhecimento, permitindo verificar rapidamente sua eficiência e desempenho com diferentes variações de parâmetros de controle. Esses procedimentos de

prototipagem rápida para a concepção de controladores de juntas robóticas permitem a realização de grande parte do projeto dentro de um ambiente de simulação, diminuindo tempo de projeto e custos envolvidos durante a sua fase de concepção.

1.4 Delineamento do trabalho

A busca do conhecimento tecnológico necessário para o profissional do mundo atual, deve ser levada em consideração ao elaborar uma proposta de metodologia educacional. Para conseguir estes objetivos tem-se que, necessariamente, considerar novos conceitos e diferentes técnicas de abordagens que possam ser aplicadas, tanto para a área educacional como para a área tecnológica, além, evidentemente, do aproveitamento eficiente e efetivo dos recursos tecnológicos disponíveis.

A prototipagem rápida, que pode ser feita em vários níveis de abstração, três fases se apresentam como representantes deste ciclo integrado:

a) Modelagem do sistema físico,
b) Visualização do modelo funcional
c) Descrição do software/hardware a ser implementado em dispositivos lógicos ou microcontroladores.

Estas três fases constituem o que se chama prototipagem rápida e estão associadas a um ambiente gráfico interativo e à utilização de um conjunto de bibliotecas e blocos personalizados.

A figura 1.1 apresenta um cronograma das diferentes fases do ciclo de concepção de um produto enfatizando a diminuição das fases de seu desenvolvimento através da utilização de prototipagem rápida.

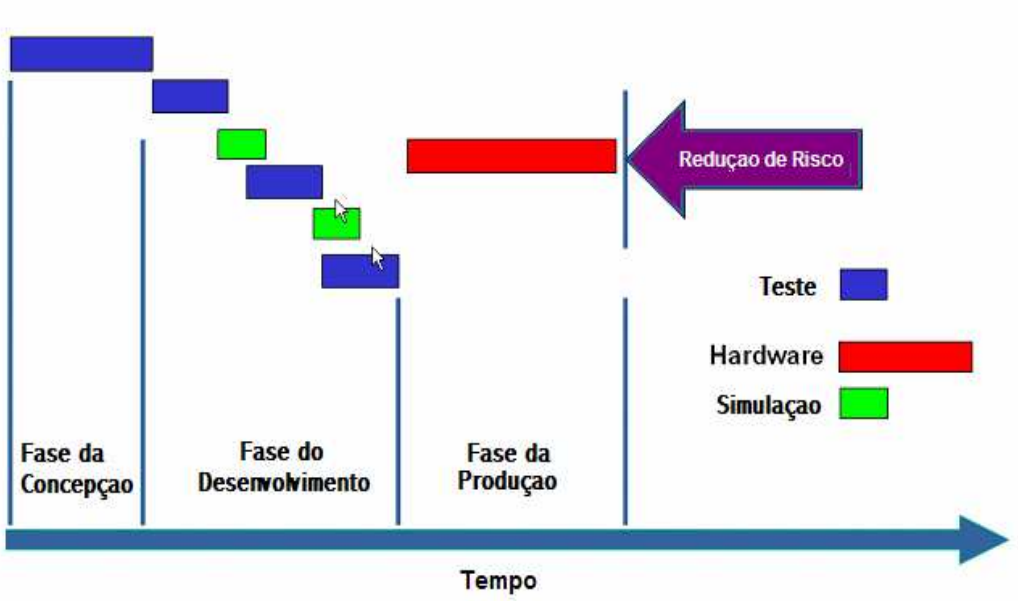


Figura 1.1: Ciclo de concepção de um produto através da prototipagem rápida.

Dentre as principais vantagens da utilização de Sistemas de Prototipagem Rápida em Mecatrônica podemos citar as seguintes:

- Detecção mais rápida de possíveis erros decorrentes da fase de implementação de um projeto, acarretando um menor custo de correção e/ou modificação do projeto,
- Concepção dentro de um ambiente de simulação e prototipagem comum, acarretando de tal modo na economia no desenvolvimento do projeto atual e de futuros projetos,
- Ambiente apropriado para Engenharia Colaborativa, com forte integração e conceito de equipe de trabalho.

Atualmente, os principais softwares tradicionais de modelagem e simulação de sistemas mecânicos e/ou eletrônicos direcionaram á área de prototipagem rápida em mecatrônica, com módulos e bibliotecas voltadas a aplicações no campo automobilístico, indústria aeronáutica e aeroespacial, etc. Esses ambientes de prototipagem rápida permitem a simulação, implementação e testes num ambiente cooperativo integrado. Dentre os principais aplicativos utilizados na

indústria, podemos destacar dentre os mais utilizados os seguintes: Labview da NI - National Instruments, DSpace, AMESim, Mathworks (Matlab-Simulink®), Altera e Xilinx.

O desenvolvimento deste trabalho de pesquisa envolve atividades de cooperação cientifica desenvolvida no Laboratório de Automação Integrada e Robótica (LAR) do DPM/FEM/UNICAMP e Departamento de Engenharia de Controle e Automação da SUPELEC, França, onde foram realizados ao longo dos últimos anos inúmeros projetos de parceria na área de controle de juntas robóticas de dispositivos mecatrônicos utilizando conceitos de prototipagem rápida associadas a uma metodologia para a organização de projetos que pode ser resumida em três etapas de desenvolvimento:

1) Elaboração e desenvolvimento da solução.
2) Simulação e Prototipagem.
3) Implementação e validação.

A partir da realização deste projeto de Pesquisa torna-se possível o desenvolvimento de projetos na área de Mecatrônica, utilizando conceitos de Prototipagem Rápida, permitindo assim a simulação de implementação de plantas industriais mecatrônicas, próximas a realidade do mercado.

1.5 Organização do trabalho

Esta dissertação de mestrado tem como principal proposta a apresentação e validação de ferramentas que permitam a rápida implementação de dispositivos de controle de juntas robóticas. O trabalho foi dividido nas seguintes etapas:

No Capítulo 2 desta dissertação é realizado um trabalho de revisão bibliográfica aprofundada abrangendo o estado da arte na área de prototipagem rápida de sistemas mecatrônicos e lógica reconfigurável utilizada para implementação de controladores de juntas robóticas. Assim, é mostrado o conceito de prototipagem rápida em mecatrônica bem como as

principais etapas que envolvem este importante conceito para o desenvolvimento de arquiteturas de controle mais eficientes, apresentando custo e tempo de implementação reduzido. Ainda neste capitulo são apresentados algumas justificativas que levaram a escolha dos ambientes Matlab/Simulink® e ALTERA como ferramenta de trabalho.

No Capitulo 3 deste trabalho são apresentados ambientes para prototipagem rápida e concepção de sistemas embarcados, onde são apresentados objetivos e propriedades destes sistemas, com ênfase nos aspectos de implementação de *hardware* e *software,* sendo também proposto neste capítulo uma estrutura mínima necessária para o funcionamento desse ambiente. Ainda neste capítulo, são descritas ferramentas para implementação, validação e testes desses controladores, além de outros tipos de interação possíveis com o sistema em estudo utilizando o ambiente Matlab Simulink®, e ALTERA como ferramenta para prototipagem de dispositivos mecatrônicos.

No Capítulo 4 é direcionada a modelagem de um sistema de acionamento e controle de uma junta robótica a partir da implementação de controladores PID expressos na forma genérica RST para posterior implementação em lógica reconfigurável, através da utilização de linguagem gráfica e linguagem VHDL.

No capitulo 5 concerne à etapa de validação e implementação experimental dos conceitos apresentados nos capítulos anteriores deste trabalho, através de um exemplo experimental implementado no Laboratório de Automação Integrada e Robótica da UNICAMP, para prototipagem de um controlador de posição de uma junta robótica baseado no conceito de *Hardware In-the-Loop* (HIL), demonstrando pelos resultados obtidos que o ambiente escolhido é eficaz e eficiente para a solução proposta.

Finalizando, no último capítulo deste trabalho, são apresentadas as conclusões finais e perspectivas futuras pertinentes ao desenvolvimento deste trabalho, bem como o uso e a escolha dessas novas ferramentas para soluções com prototipagem rápida.

Nos anexos são apresentados detalhadamente telas gráficas, programas implementados e resultados que não se encontram no corpo da tese, mas que permitem uma melhor compreensão por parte do leitor.

Capítulo 2

Prototipagem Rápida de Sistemas Mecatrônicos e Computação Reconfigurável - Revisão de Literatura

Dentre os novos modelos da mecatrônica podemos destacar que além da integração entre os diversos ramos das Engenharias Eletrônica, Mecânica e Computação podemos notar cada vez mais que a ênfase da mecatrônica é direcionada a integração de subsistemas mecânicos e eletrônicos com controle baseado em software, sendo que um dos principais objetivos dessa linha de pesquisa é estudar a integração no processo de *design* de ferramentas computacionais e métodos que permitam explorar cooperativamente as técnicas da mecatrônica em uma abordagem de Integração de Sistemas.

O conjunto de aplicações típicas para este estudo compreende os projetos de grande porte sejam acadêmicos ou desenvolvimentos industriais, onde se tem a necessidade de conceber e validar um conjunto grande de componentes, cujas características amparam decisões de projeto e de um comportamento coletivo.

2.1 Introdução

Nos dias atuais, a evolução tecnológica está direcionando uma grande gama de produtos na área de engenharia, acarretando em alterações drásticas na concepção de um projeto de um sistema mecatrônico no que diz respeito às evoluções do projeto e concepção mecânica, sistema de acionamento e de controle. Estes novos produtos mecatrônicos emergem da combinação

apropriada dos sistemas mecânicos, da eletrônica e do processamento em tempo real dos sistemas de controle e tratamento de informações através da implementação de diversificadas funções de controle embarcadas e integradas ao sistema. Como exemplos, podemos citar a intensidade crescente de componentes eletromecânicos inteligentes, de máquinas automatizadas, de veículos com grande número de sensores e eletrônica embarcada e de dispositivos mecânicos de precisão.

Um fator de grande importância industrial consiste na redução de tempo de desenvolvimento de um produto com a diminuição do número de etapas, gerando a produção de novos e diferenciados produtos. Isso é indispensável no mundo globalizado e particularmente necessário nas indústrias de alta tecnologia, como por exemplo, no que diz respeito à implementação de controladores cada vez mais rápidos e eficientes.

Muitos processos e produtos técnicos nas diferentes áreas da engenharia, principalmente na engenharia mecânica e elétrica, apresentam uma integração crescente dos sistemas mecânicos com processamento da eletrônica digital e de informação. O seu desenvolvimento envolve a busca de uma solução otimizada entre a estrutura mecânica básica, o sistema de sensoriamento e o elemento de atuação e controle, através do processamento automatizado de informações e controle global do sistema.

Tudo isso acarretará no desenvolvimento e utilização de ferramentas para o projeto simultâneo dos sistemas mecânicos e hardware de acionamento e controle, através da implementação do software e funções de controle embarcadas dentro de um ambiente de simulação, tendo como resultado e principal objetivo a eliminação da totalidade ou parte de protótipos intermediários e a geração de um componente ou um sistema integrado de controle. Para tal propósito descrito existe um conceito atual denominado prototipagem rápida de sistemas mecatrônicos.

A prototipagem rápida é uma ferramenta que permite a construção de protótipos de uma maneira econômica e segura. Esse conceito era muito utilizado anteriormente para a construção de protótipos de peças mecânicas a partir de um projeto desenvolvido em CAD (*Computer Aided Design*) ou da implementação de componentes eletrônicos em FPGA's a partir de um CAD de eletrônica. Mais recentemente, esse conceito é utilizado de forma mais ampla, envolvendo a concepção de todo um projeto de um sistema mecatrônico desde as fases de modelagem, simulação e arquitetura de controlador até a sua implementação final em hardware dedicado.

Conseqüentemente, a prototipagem rápida tem como objetivo a geração automática do código equivalente ao controlador para testar os sistemas reais, resultando em diminuição nos custos de implementação de um controlador, principalmente se esse for desenvolvido para um projeto específico, ou seja, a prototipagem rápida é uma ferramenta que possibilita a construção de protótipos de uma maneira econômica e segura, onde hardware pode ser implementado num sistema embarcado (*embedded system*) a partir de componentes virtuais.

Durante a implementação deste projeto de pesquisa, abordaremos conceitos de prototipagem rápida num contexto mais aberto, que inclui a implementação em ambiente virtual do modelo de um sistema mecatrônico (modelagem cinemática e dinâmica), simulação e hardware de supervisão e controle.

2.2 Estado da Arte

A Mecatrônica pode ser definida como um conjunto das técnicas que agregam inteligência a um sistema ou a uma máquina, através da integração entre a eletrônica e a tecnologia da informática com os sistemas mecânicos. Os principais domínios de aplicação são: os sistemas elétricos, eletromecânicos, os autômatos finitos programáveis, a pneumática, a hidráulica, os sensores e a domótica. Um sistema mecatrônico deve integrar a mecânica, circuitos analógicos, digitais, microprocessadores e computadores, sensores e atuadores além de algoritmos de comando e tecnologias da informação (figura 2.1).

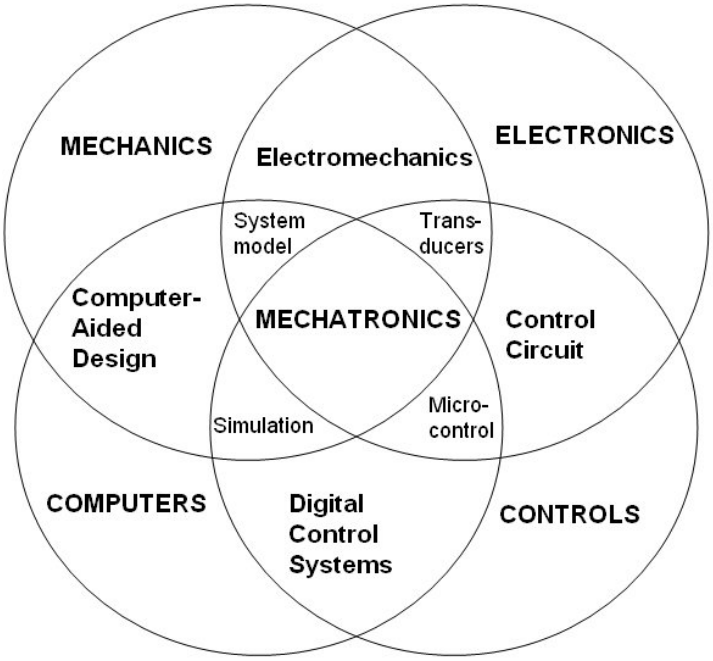


Figura 2.1 : Definição do domínio da Mecatrônica (IFAC 2005).

Dado que o ciclo de vida do processo de *design*, arregimentando várias disciplinas é o verdadeiro núcleo da Mecatrônica, o controle sobre o processo de *design* é de fundamental importância. A validação de requisitos dentro deste processo é muito discutida em várias comunidades acadêmicas (Ciência da Computação e Engenharias Civil, Elétrica e Eletrônica, Controle e Automação, Térmica, e outras) e representa um desafio especial, dado que a viabilidade da automação de um determinado processo é parte deste ciclo. Por exemplo, para a automação de processos cirúrgicos, ferramentas dedicadas devem ser validadas antes mesmo de se pensar no *design* destas ferramentas ou mesmo da composição de um robô para este fim. Entretanto, a fase inicial do ciclo de vida, que compreende o ciclo informal de iniciação de requisitos e formulação das especificações, não tem dados suficientes para se entrar em cálculo de cinemática e dinâmica ou design mecânico de um dispositivo robótico.

Genericamente, os conceitos oriundos da eletrônica, do controle, do fluxo de informação ou da interação direta com usuário, utilizados para o *design* de qualquer sistema mecatrônico, aplicado a qualquer área, precisam ser investigados a fundo e ferramentas de apoio ao *design* destes processos colocados a disposição do setor de engenharia responsável por eles.

O aspecto colaborativo é essencial dado que qualquer abordagem Taylorista redundaria em entrave ao processo de projeto. O desenvolvimento dos prédios inteligentes e os obstáculos para a integração do projeto civil com o projeto de automação são responsáveis por redundâncias de instalação, custos muito superiores ao esperado e decepção com os resultados, para citar outro exemplo.

O ciclo de vida de um produto mecatrônico apresenta a integração entre a modelagem, controle e a realimentação, pensando em sistemas de malha fechada. A figura 2.2 mostra detalhadamente a base desta integração que pode ser fundamentada em ferramentas específicas (ou em um conjunto delas).

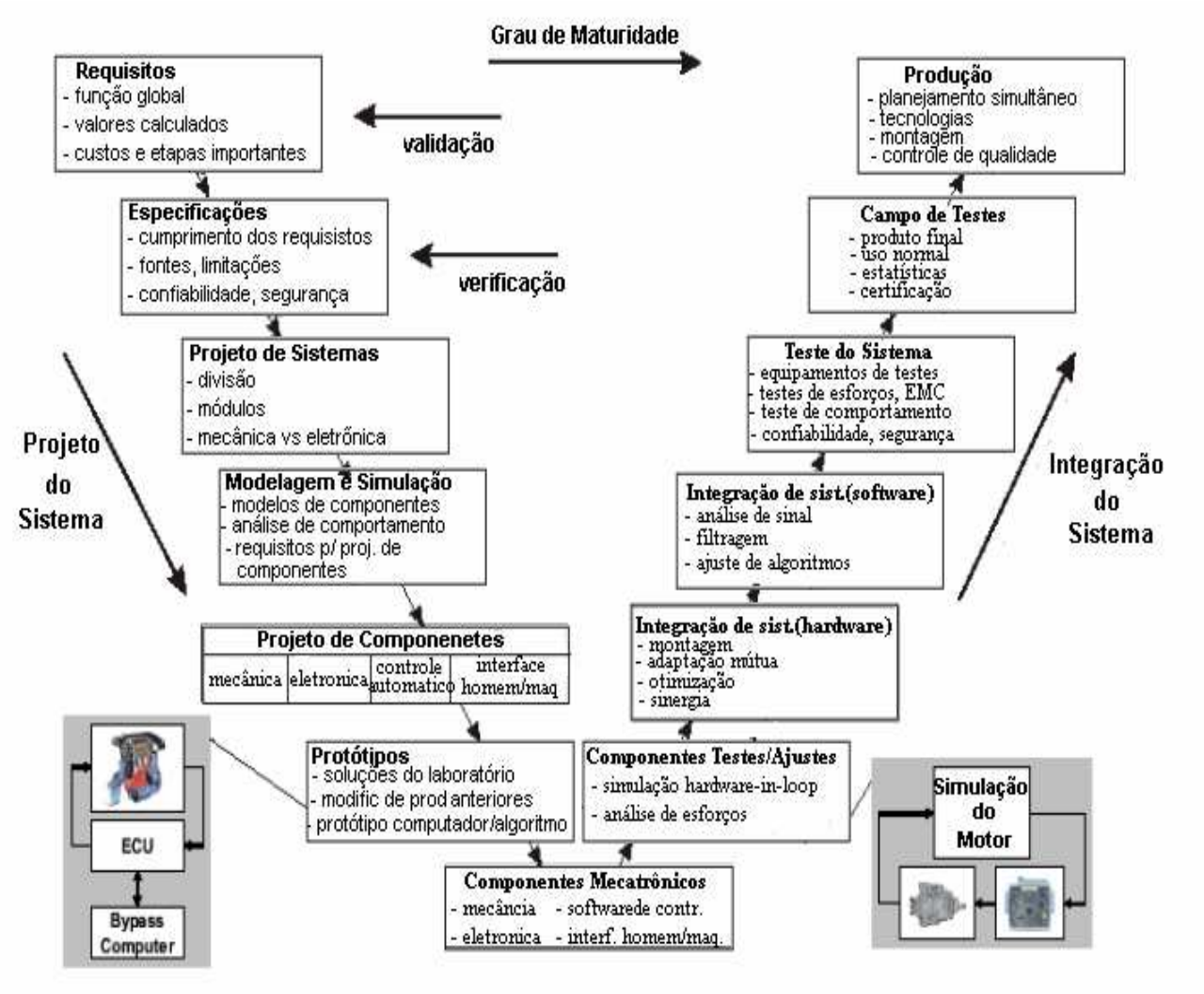


Figura 2.2: Ciclo em V – Metodologia para Concepção e Desenvolvimento para sistemas Mecatrônicos (Isermann, 2005).

O ciclo em V (Isermann, 2005), sintetiza as diferentes etapas associadas à concepção de um Produto, onde pode ser visto uma fase inicial de Projeto e Concepção (abrangendo fase de requisitos, especificações, modelo, componentes e prototipagem) e uma segunda fase de implementação final associada à Integração de Sistemas (abrangendo a integração final de hardware e de software, testes e certificações, produção e supervisão).

Os produtos mecatrônicos que queremos projetar têm dois aspectos fundamentais: possuem um grau de autonomia que pode variar desde um dispositivo comandado interativamente até um robô autônomo; tem características funcionais de um sistema dinâmico discreto ou híbrido, seja de maneira direta ou aproximada. Portanto, para ter um ciclo coerente para estes sistemas (que resultará em melhores produtos e em um *design* de menor custo) devemos tratar estes dois aspectos desde a fase inicial do ciclo, isto é, requisitos funcionais coerentes e válidos, chegando ao conceito de prototipagem rápida de sistemas mecatrônicos. A figura 2.3 apresenta o conceito de produto inteligente.

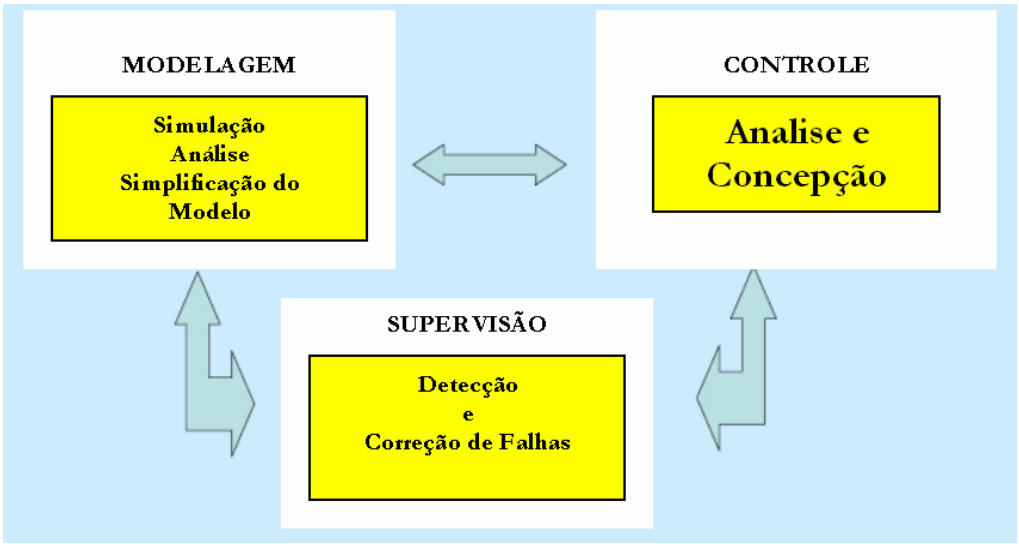


Figura 2.3: Concepção Integrada de um "Produto Inteligente"

2.3 O conceito de Prototipagem Rápida na Concepção de Sistemas Mecatrônicos

A prototipagem rápida, pode ser realizada em vários níveis de abstração, sendo a espinha dorsal do processo de concepção, onde os elementos de integração e de autonomia devem ser verificados a cada passo através de três etapas associadas a um ambiente gráfico interativo e à utilização de um conjunto de bibliotecas e blocos personalizados: modelagem do sistema físico,

visualização do modelo funcional e descrição do *software* a ser implementado em dispositivos eletrônicos ou microcontroladores.

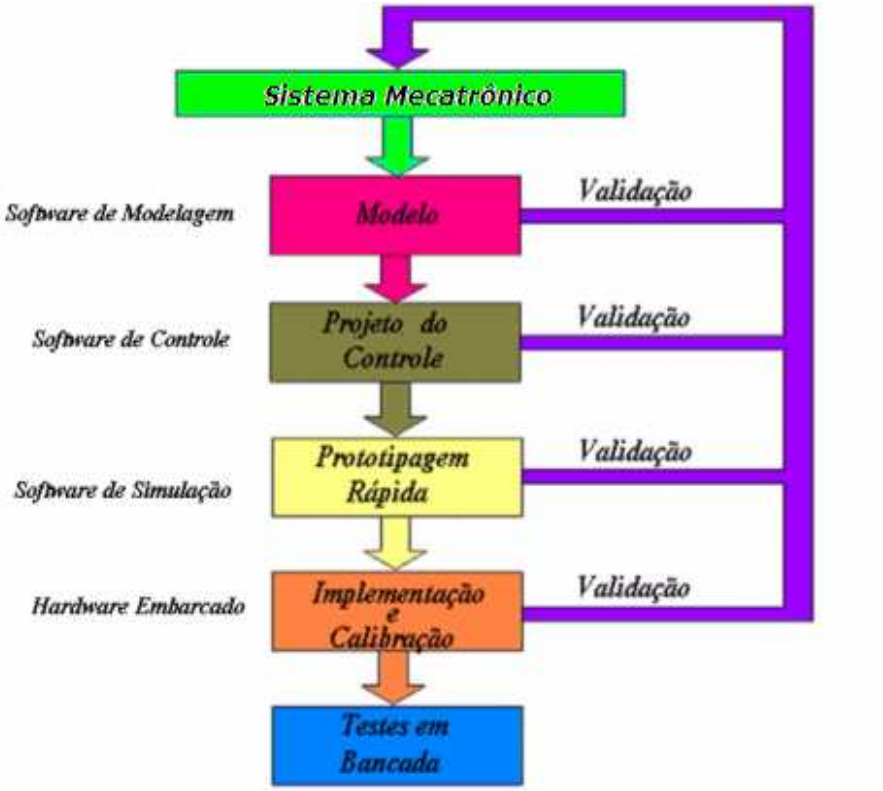


Figura 2.4: Fluxograma das diferentes fases de implementação de um sistema mecatrônico.

A figura 2.4 apresenta, sob forma de fluxograma, as diferentes fases de implementação de um dispositivo mecatrônico a partir de um modelo. Podemos observar a existência de várias etapas para a concepção de um sistema de controle utilizando a técnica de prototipagem rápida. Para a diminuição de custos, a maior parte da concepção resulta de simulações antes da implementação real. Deste modo, o desenvolvimento de um ambiente virtual que permita analisar o melhor desempenho do algoritmo de controle a ser implementado se faz necessário.

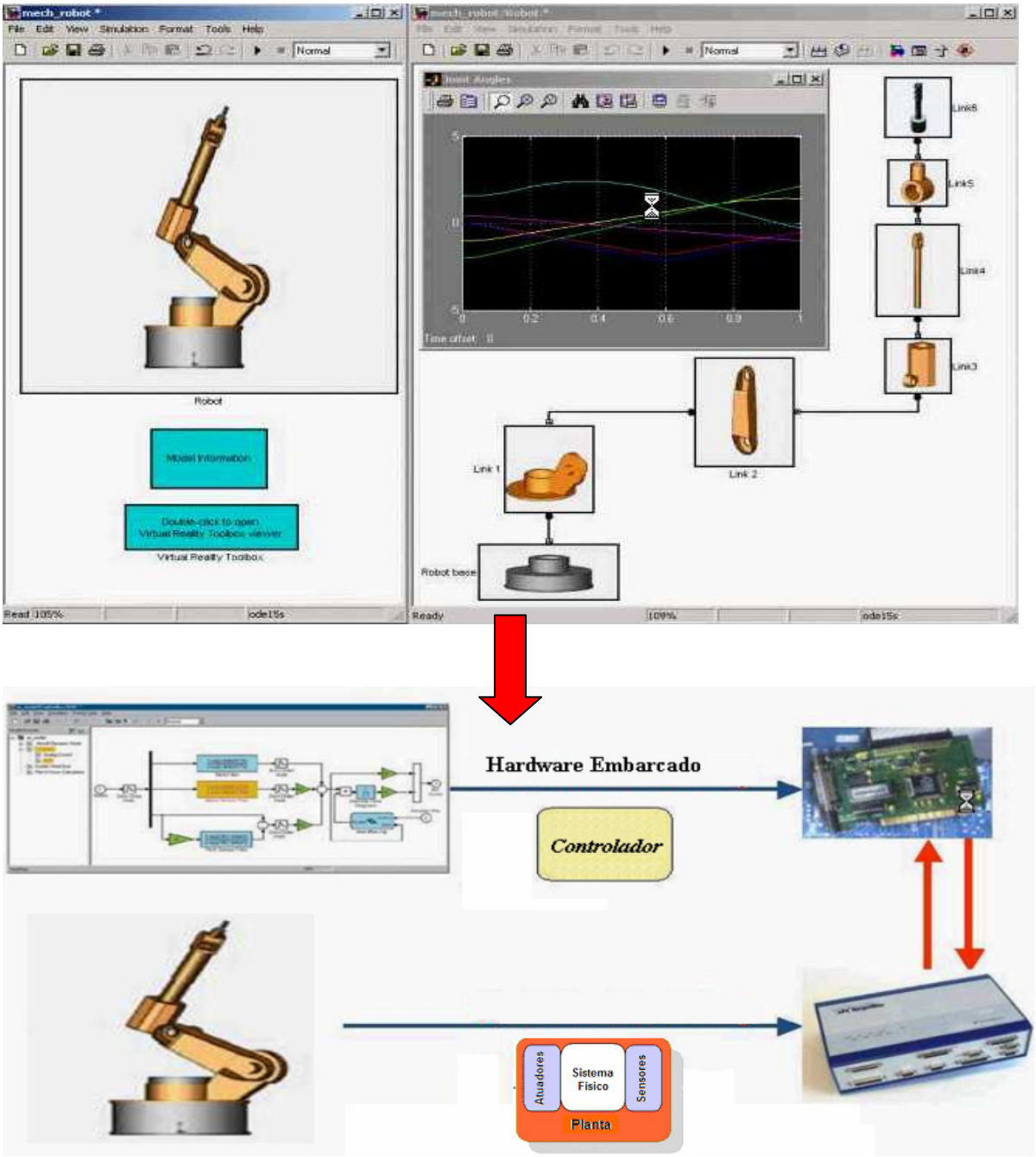


Figura 2.5: Princípio da Prototipagem Rápida de Sistemas Mecatrônicos.

A figura 2.5 exemplifica as diferentes fases da concepção de um sistema mecatrônico através do exemplo de prototipagem rápida de um controlador de juntas de um robô industrial com seis graus de liberdade. Neste exemplo, pode-se observar uma fase inicial referente à simulação do movimento cinemático dos diferentes graus de liberdade do robô e o projeto do sistema de controle referente a cada junta robótica, sendo considerados os acoplamentos dinâmicos entre as juntas implementados em simulador, para posterior prototipagem do sistema de controle através de hardware embarcado.

Em Ollero et al, 2005, é mostrada uma análise no desenvolvimento e concepção de sistemas mecatrônicos com o progresso tecnológico dos últimos anos, tais como a inovação na área de componentes, bem como a melhoria de softwares para aplicação na solução de sistemas embarcados, com a sinergia de várias de áreas de conhecimentos.

2.4 Modelagem de Sistemas Físicos

A fase de modelagem e controle de sistemas físicos exige a utilização de um conjunto de ferramentas para modelagem intuitiva e de fácil utilização, a partir de um ambiente virtual de simulação de sistemas físicos. Estas ferramentas direcionadas a engenheiros de concepção e análise de sistemas físicos deverão possibilitar a concepção de um modelo físico simplificado, próximo ao modelo real, permitindo a sua implementação em tempo real e atendendo os níveis de precisão exigidos para o desenvolvimento do modelo.

A modelagem de sistemas físicos necessita de ferramentas que facilitem a implementação do modelo de sistemas associado a sua arquitetura de controle de maneira rápida e precisa, acarretando a diminuição do tempo relativo às fases de implementação e testes, como também uma maior precisão na formulação e implementação de modelos de planta mais realísticos.

Estas ferramentas deverão dar suporte, através de bibliotecas de blocos hierarquizados, a funções dedicadas à mecânica, sensores, atuadores e outros elementos, que permitirão a modelagem de componentes mecatrônicos de corpos rígidos e seus acoplamentos dinâmicos,

sistemas de coordenadas, análise de parâmetros físicos: posição, força, análise cinemática e dinâmica dos sistemas mecânicos (figura. 2.6).

A partir da modelagem completa do sistema mecânico que se deseja controlar, faz-se o projeto do controlador utilizando a técnica de controle que melhor se adapte à estrutura estudada ou qualquer método que se pretenda utilizar. Dessa forma, o modelo acentua a concepção do sistema, tornando possível a validação da lei de controle em ambiente de simulação próximo ao real, suportando a concepção e integração do controlador para sistema físico, com a associação de um componente a outro.

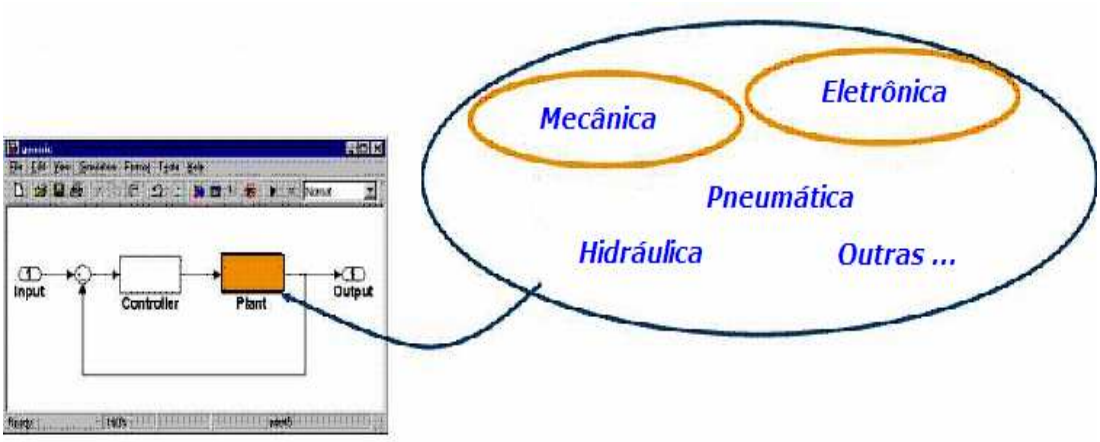


Figura 2.6: Conceitos a serem implementados na planta baseados na modelagem de sistemas físicos.

Dentre os aplicativos comerciais, podemos citar a MathWorks® que elaborou ferramentas com esta capacidade, de modo a permitir o desenvolvimento completo de um sistema mecatrônico dentro do SIMULINK®. Estas ferramentas permitem a modelagem dos componentes mecânicos, simulação dos movimentos e análise direta dos resultados sem ter que derivar as equações do sistema e ainda realiza a modelagem das montagens mecânicas e seus controladores dentro de um mesmo ambiente SIMULINK®, ou seja, eliminam as interfaces "pesadas", diminuem o ciclo de concepção e proporcionam criação de algoritmos de controle mais convenientes. As principais funcionalidades são as seguintes:

- Implementação de diagrama de blocos equivalentes às equações diferenciais que descrevem os sistemas físicos ou blocos dedicados.
- Modelagem de bibliotecas utilizando outras ferramentas de concepção já existentes no mercado (ADAMS®, PRO-ENGINEER®, AMESIM®, etc.).

2.5 Implementação do Sistema de Controle

Com o rápido desenvolvimento de novas tecnologias, sobretudo no que concerne à informática e à eletrônica, pode-se aprimorar a pesquisa científica e tecnológica na área da engenharia mecatrônica. Entretanto, a implementação de robôs para aplicações específicas requer normalmente um elevado custo investimento financeiro até chegar ao desempenho apresentado pelos fabricantes, sendo que os equipamentos acabam tornando obsoletos e com uma grande depreciação. O desempenho de um controlador exige que sua concepção esteja fortemente ligado ao modelo físico da planta associada, exigindo constantes atualizações de seu controlador (*hardware/software*) e aquisição de dispositivos externos a serem integrados no controlador do robô, nem sempre possíveis de serem realizadas pelo usuário.

Com isso, torna-se imprescindível o estudo e aprimoramento de técnicas de controle de sistemas mecatrônicos, com ênfase na sua implementação em tempo real a partir de microprocessadores e microcontroladores industriais. Assim, dentro do estudo da prototipagem rápida, os seguintes aspectos deverão ser implementados:

- Ambiente de prototipagem que represente um modelo muito próximo ao modelo real, ocasionando um menor tempo de execução e com custos relativamente mais baixos.
- Desenvolvimento de algoritmos de controle que admitam algoritmos complexos e inteligentes para serem processados num tempo mínimo.
- Ferramentas inteligentes para auxiliar o modelamento e que sejam de fácil utilização.
- Integração e concepção do sistema, implementação e validação de algoritmos de controle/comando,

- Arquitetura modular, através de blocos de funções de transferência, de modo a associar um componente aos outros e que sejam facilmente reconfigurável.

Portanto, a utilização de um ambiente único para cálculo científico, análise de dados e visualização, modelagem de sistemas e simulação; implementação de lógica embarcada em tempo real através da implementação de um sistema completo da fase de concepção até a eletrônica embarcada, pode ser realizada pelo Matlab/Simulink®, que facilita o trabalho em grupo dos projetistas de controle.

A implementação de um controlador de um sistema mecatrônico através de prototipagem rápida apresenta inúmeras vantagens, dentre elas o fato dos circuitos eletrônicos serem programados de maneira rápida, segura e otimizada, e ainda, de maneira personalizada: Simulação *Hardware In the Loop* (HIL).

Uma das principais vantagens dos sistemas de prototipagem rápida é que após a simulação do modelo virtual do sistema, o sistema total ou partes desse modelo poderão ser facilmente trocados pelo *hardware* de controle real para validação e testes do modelo, de modo que o protótipo virtual se transformará num produto muito próximo do real, simplificando etapas de concepção, validação e testes. Conseqüentemente, após a fase de simulação do sistema, que deverá incluir o modelo virtual do *hardware*, esta parte do modelo poderá ser trocada pelo *hardware* real, similar ao modelo virtual, fazendo que o *hardware* passa ser integrado no modelo de simulação virtual e possibilitando o estudo do comportamento do novo componente com o resto do sistema, em particular o estudo do comportamento global do sistema.

Essa fase de testes é chamada de *Hardware In the Loop* (HIL), que deverá permitir a integração do hardware físico dentro do ambiente de simulação. As principais ferramentas necessárias para realização desse procedimento são as seguintes:

- Ferramentas de programação e hardware aberto para implementação do protótipo;

- Unidade de cálculo que permita assegurar a comunicação com o hardware e a área de trabalho, que deverá comportar: portas de entradas e saídas, conversores analógico/digital e digital/analógico.
- Implementação de programa que permita a importação de modelos representando o sistema e que seja capaz de gerar as entradas e programável por blocos.

As aplicações do HIL são utilizadas para avaliar e validar os elementos desenvolvidos por um novo sistema. Elas consistem em testar esses elementos, antes de concretizar o sistema real, a partir unicamente da simulação do resto do sistema. Os componentes do hardware testados respondem aos sinais enviados pelo computador e simulam o resto do sistema, como se encontrassem dentro de um sistema real.

Atualmente existem várias empresas que oferecem soluções que realizam o *hardware in loop* (HIL), como por exemplo: dSPACE®, National Instruments® (NI), Altera® e Opal-RT® que apresentam produtos dedicados a prototipagem rápida.

A figura 2.7 apresenta esquematicamente um exemplo de implementação através do ambiente MathWorks® utilizando HIL. Neste exemplo, observa-se que a partir do ambiente de simulação utilizado para validação e testes do modelo físico implementado, o hardware de controle poderá ser gerado e incorporado dentro do simulador, para validação e testes dentro desse mesmo ambiente de prototipagem.

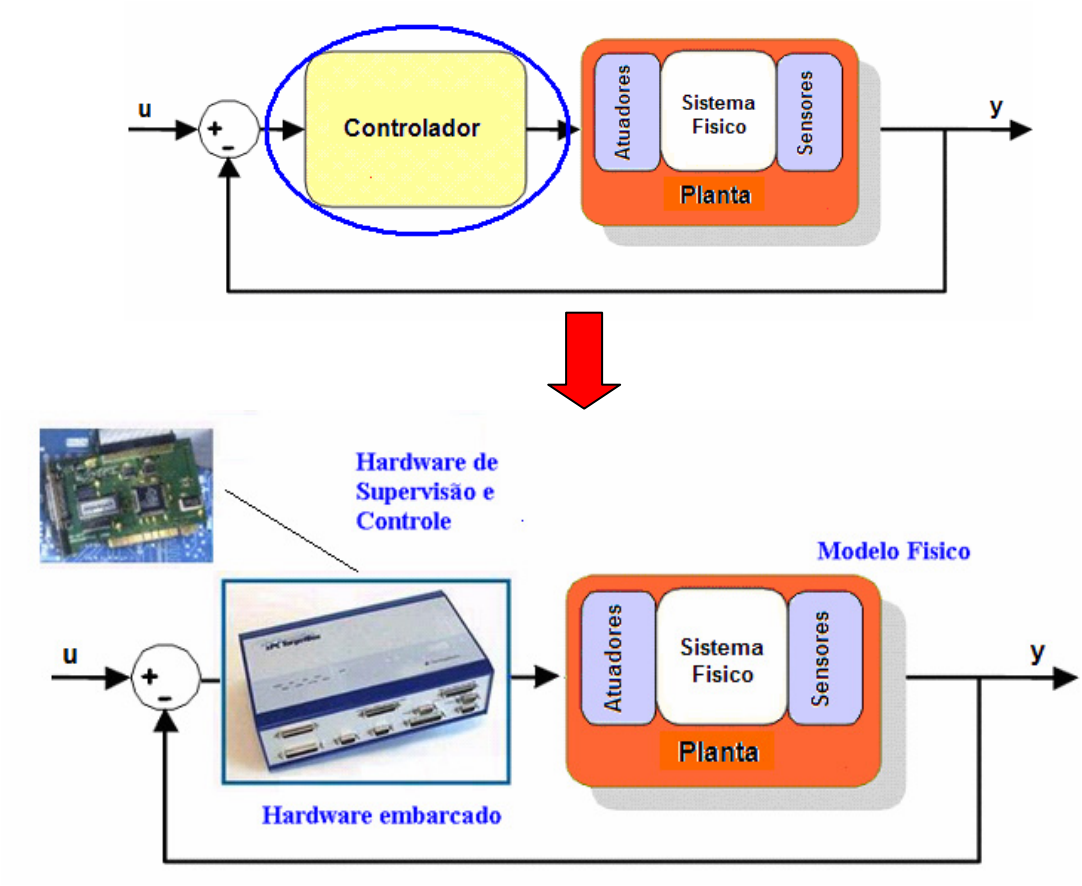


Figura 2.7: Hardware In the Loop através do Mathworks®.

As principais etapas a serem implementas no HIL são as seguintes:

- Desenvolvimento de um programa de simulação usando ambiente tal como o Matlab/Simulink®;
- Configuração das entradas e saídas necessárias para o funcionamento do sistema simulado com o elemento hardware;
- Geração automática do código (com possibilidade de troca de parâmetros em tempo real e iniciação imediata da simulação em HIL);
- Visualização dos resultados em tempo real.

Dentre as principais dificuldades na utilização do HIL podemos destacar:

- Nível de integração de plataformas e interfaces utilizadas;
- Simulação diferenciada: sincronização e tempo de simulação;
- Atualização constante de diferentes softwares e compatibilidade;
- Utilização freqüente;
- Criação de interfaces com ferramentas do mercado existentes nem sempre é evidente.

A figura 2.8 apresenta um exemplo completo de implementação *Hardware in Loop* (HIL) para um sistema mecatrônico, onde partes do modelo físico são geradas no ambiente Matlab-Simulink®.

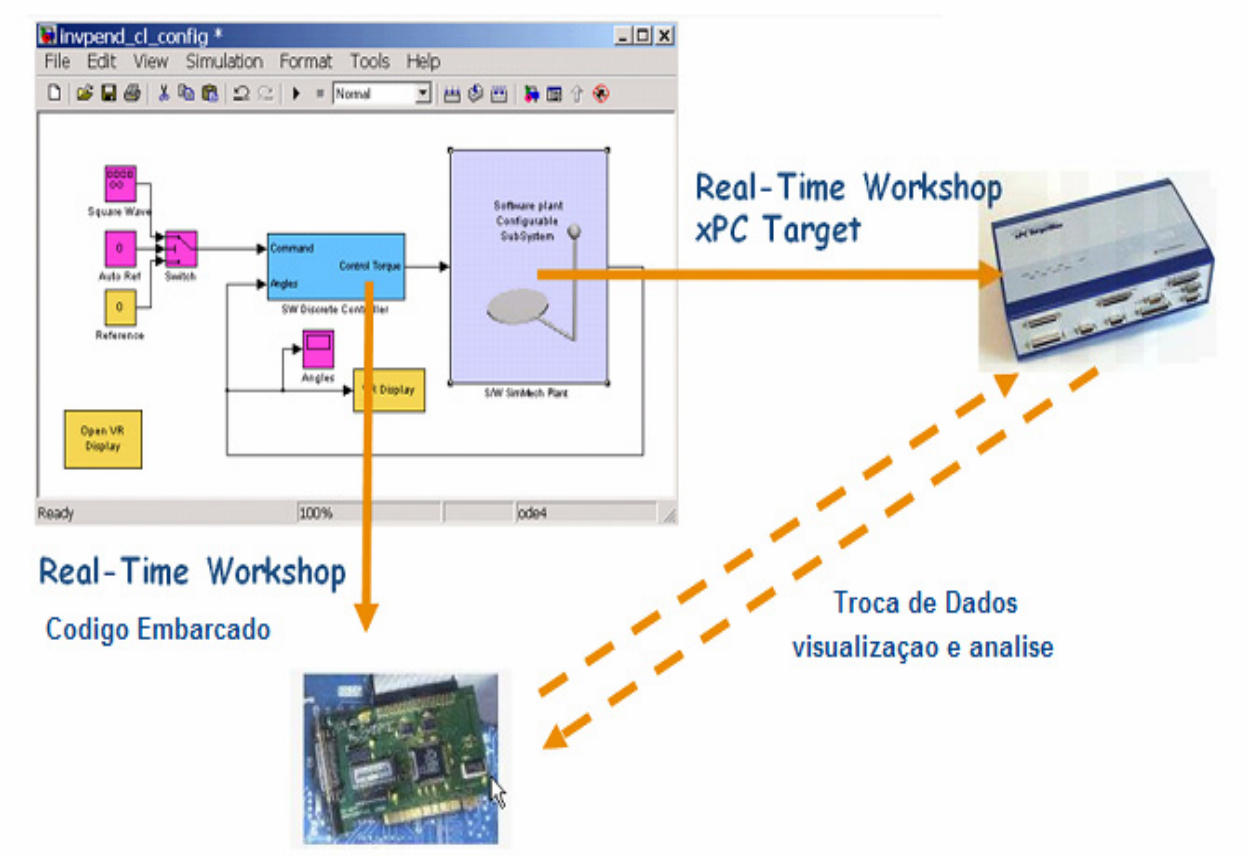


Figura 2.8: Esquema de prototipagem do controlador.

2.6 Simulador Virtual

Como exemplo de desenvolvimento de um Simulador Virtual para Prototipagem de um sistema de controle de dispositivos mecatrônicos pode citar o trabalho de Melo, 2007, utilizou os conceitos de prototipagem rápida para implementação de um ambiente virtual para simulação de um sistema de supervisão e controle para robôs móveis capazes de operar e de se adaptar a diferentes ambientes e condições de trabalho.

A implementação desse simulador considerou aspectos relacionados à modelagem cinemática e dinâmica e controle de sistemas robóticos móveis, que permitiu a escolha de um modelo, considerando suas capacidades e limitações, possibilitando ainda uma proposta de arquitetura aberta de supervisão e controle que integre as melhores características das técnicas que existem atualmente.

A figura 2.9 apresenta as diferentes fases de implementação de um projeto a partir da concepção de um modelo físico utilizando o conceito de prototipagem rápida. Através de um simulador virtual é executada a simulação de hardware e software do sistema mecatrônico em estudo. Este procedimento deverá acarretar uma diminuição de tempo de desenvolvimento do produto e de seu custo, resultando em testes relevantes durante a fase de validação e conseqüente diminuição de possíveis erros que ocorrem durante o processo de implementação e testes.

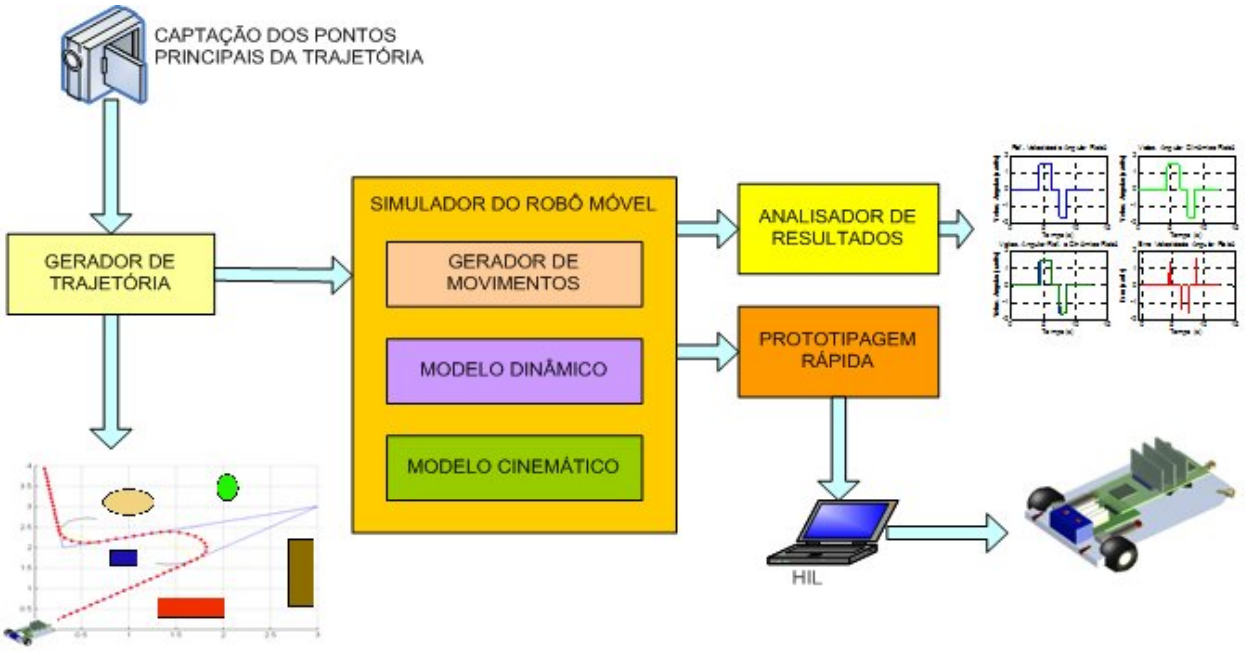


Figura 2.9: Simulador Virtual dentro do processo de Prototipagem Rápida.

O sistema de controle, que consiste em malha de controle em cascata para cada grau de liberdade do sistema mecatrônico, será ser implementado em blocos do Simulink®. O conjunto de malhas de controle de posição, velocidade e torque podem fazer parte do modelo do sistema de acionamento e controle de uma junta robótica. O controle de posição de cada grau de liberdade do dispositivo será implementado através de realimentação para cada junta isolada, requerendo o modelo dinâmico de cada junta.

Durante o desenvolvimento desse projeto de pesquisa será implementada a malha de controle de posição acoplada ao modelo completo de uma junta robótica, conforme mostra a figura. 2.10, utilizando arquitetura aberta, de modo à facilmente serem implementados diferentes estratégias de controle, para posterior simulação, análise e comparação de desempenhos.

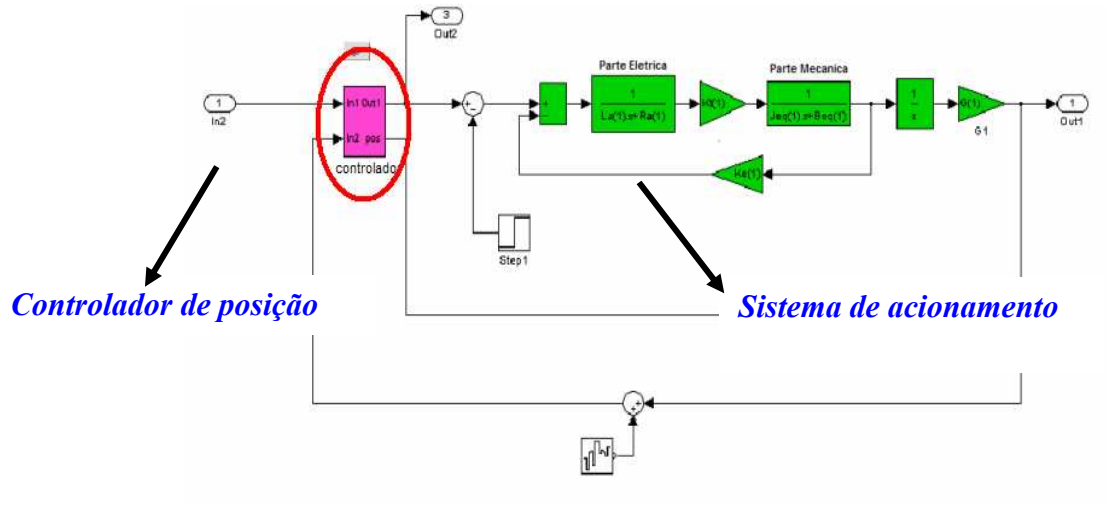
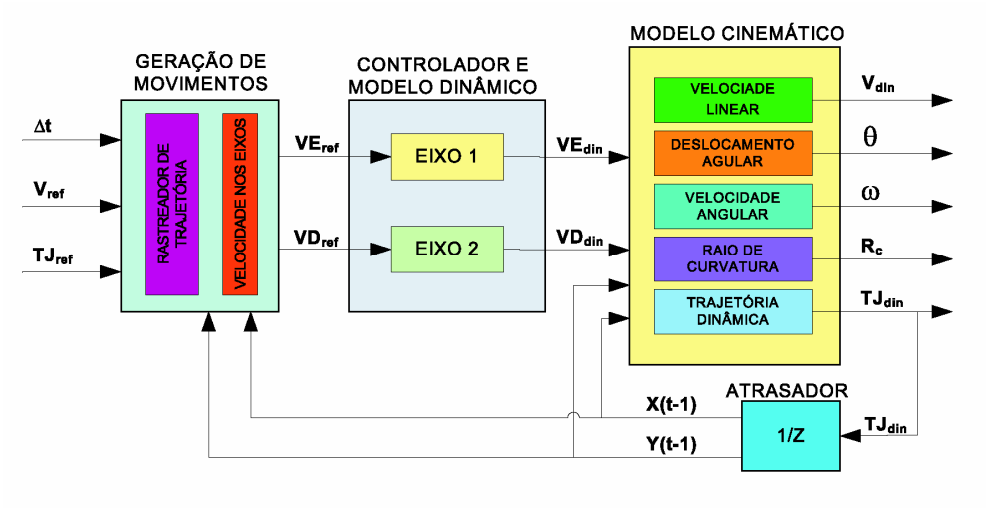


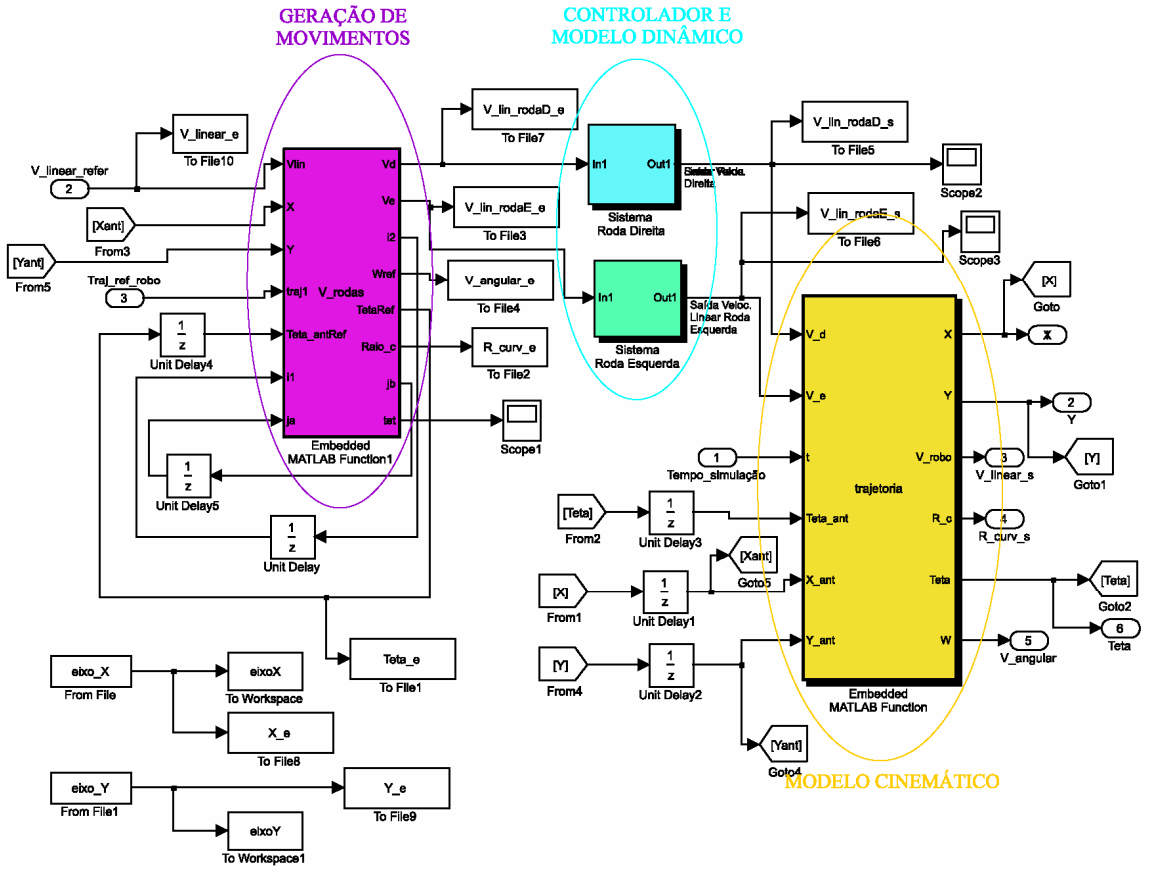
Figura 2.10: Malha de acionamento e controle de posição de uma junta.

Outros elementos do sistema mecatrônico (incluindo possíveis cargas externas) serão representados por modelos não-lineares, um para cada acionador. A figura 2.11 apresenta a proposta de simulador implementado no trabalho Melo, 2007. Este simulador possui os seguintes módulos:

- Geração de trajetórias: geração de sinais de referência das juntas para cada controlador.
- Sistema de Acionamento: modelo eletro-mecânico do dispositivo e projeto do controlador de posição.
- Cinemático: modelo do sistema mecatrônico em estudo foi implementado utilizando funções "S-function", dentro do ambiente Matlab® integrados nos blocos do Simulink®.
- Interface Gráfica: visualização temporal das saídas e entradas do sistema em estudo. Para melhor compreender e analisar o comportamento espacial do sistema torna-se imprescindível a implementação de um simulador gráfico de movimentos espaciais, mostrando resultados dos movimentos obtidos através de trajetórias de referência.



a) Diagrama de Blocos Funcionais



b) Implementação em Ambiente Matlab-Simulink®.

Figura 2.11: Simulador de um Robô Móvel.

2.7 Sistema de Supervisão e Controle

O ambiente gráfico LabVIEW™ é muito utilizado industrialmente em projetos de dispositivos relacionados as áreas de medição e controle, possibilitando assim a criação de aplicações personalizadas que rodam em plataformas NI (National Instruments) com E/S reconfiguráveis baseadas em FPGAs. Juntos, o LabVIEW™ FPGA e o hardware NI, permitem a criação de uma plataforma flexível para o desenvolvimento de sistemas sofisticados que antes somente eram possíveis com hardware projetado de forma dedicada.

O software LabVIEW™ pode ser usado para implementação do sistema de supervisão e controle de movimentos, residente num computador e responsável pelo gerenciamento e controle das informações provenientes de sensores e atuadores do sistema.

A figura 2.12, apresenta uma proposta de interface de visualização para sistemas robóticos móveis, implementada em ambiente gráfico, possuindo telas de supervisão que coletam as informações dos sensores, dos programas de tratamento matemático (modelagem cinemática) e as informações sobre o sistema (inicialização, velocidade, parâmetros do regulador, números de pontos da trajetória, etc).

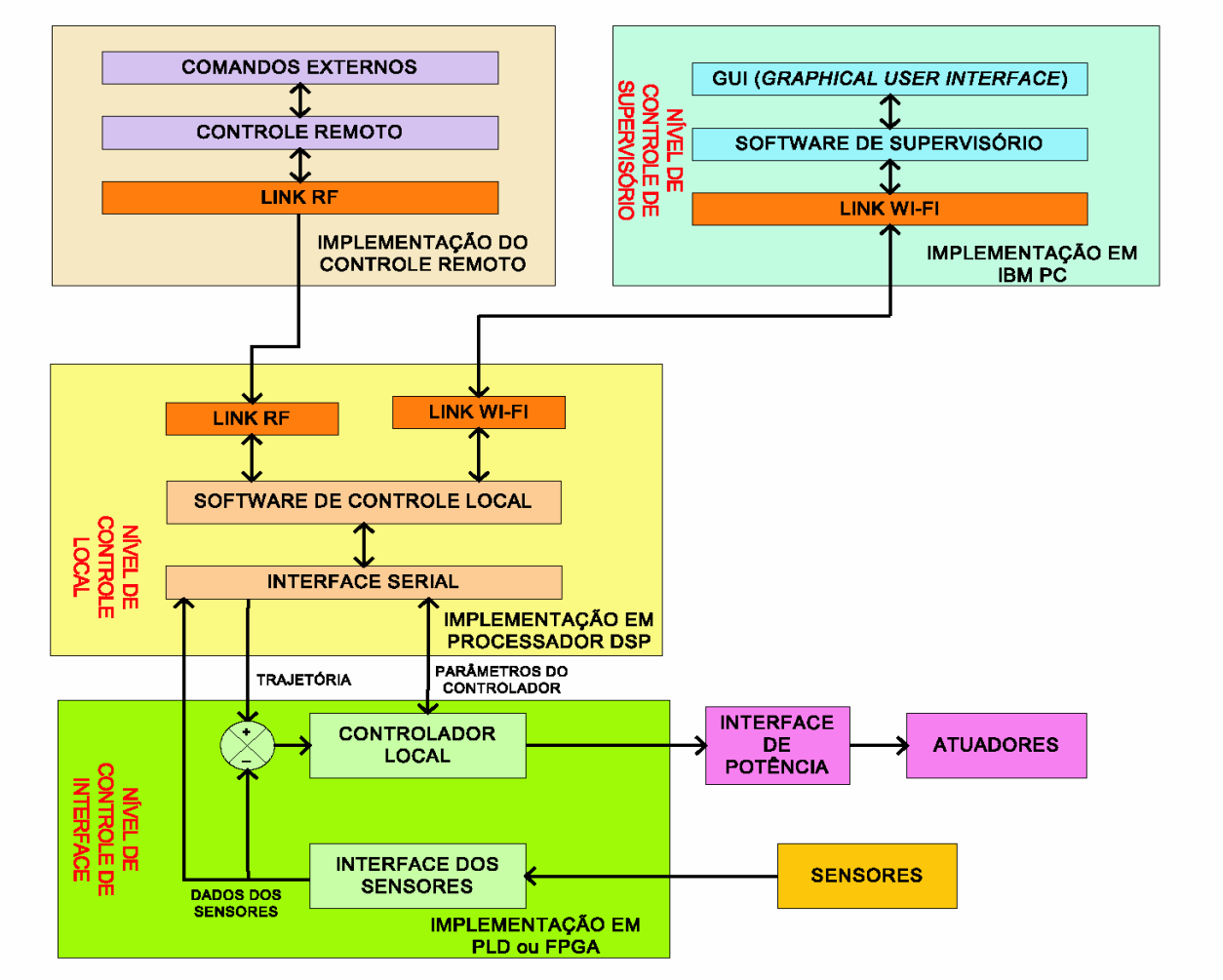


Figura 2.12: Proposta de interface de visualização para sistemas robóticos móveis .

2.8 Sistemas Reconfiguráveis

Sistemas reconfiguráveis são sistemas que têm como principal característica a capacidade de adaptar-se a tarefas específicas através da substituição de parte de seu software ou hardware, possibilitando assim obtermos alto desempenho com baixo custo de produção, sendo uma alternativa às máquinas de Von Neumman implementadas nos sistemas com microprocessadores. Sistemas reconfiguráveis por *software* são bastante comuns, sendo exemplos os sistemas embarcados em automóveis, eletrodomésticos, vídeo games dentre outros. Nestes sistemas, a

mudança de uma ROM ou de um CD-ROM leva à reconfiguração de funções responsáveis pela operação dos mesmos [Miyazaki 1998].

Sistemas com hardware reconfigurável são mais recentes e estão associados ao aparecimento das FPGA (*Field Programmable Gate Array*) na década de 90. No trabalho de Page, [Page, 1996] é proposta a união em um único circuito integrado de um microprocessador e uma FPGA para atender dinamicamente novas aplicações.

Comparados aos sistemas de software reconfiguráveis os sistemas de hardware reconfiguráveis apresentam um maior potencial em termos de desempenho e adaptabilidade. Outras siglas estão associadas aos sistemas de *hardware* reconfigurável: CCM (*Custom Computing Machines*) ou FCCM (*FPGA-based Custom Computing Machines*). As expressões computação reconfigurável e lógica reconfigurável também estão associadas a sistemas de hardware reconfiguráveis.

Tradicionalmente, a execução de um algoritmo na computação convencional pode seguir dois métodos de utilização:

a) Tecnologia de hardware, a exemplo de ASICs (*Application Specific Integrated Circuits*) ou de placas de circuito impresso; apresenta como vantagem altas velocidades de execução, mas pouca flexibilidade a modificações;
b) Microprocessadores programáveis por *software*: apresenta alta flexibilidade para modificações, mas não é executado com a mesma velocidade dos algoritmos programados em hardware [Compton, 2002]

O fato dos microprocessadores executarem de forma seqüencial suas tarefas (máquina de Von Neumann) faz com que os algoritmos programados por *software* tenham tempos de processamento que não são aceitáveis para muitas aplicações [Pimentel e Le-Huy, 2000, Dido et al., 2002].

A computação reconfigurável tem por objetivo suprir a lacuna entre a solução por *software* e a solução por *hardware*, atingindo desempenhos muito superiores aos desempenhos obtidos por *software*, mas obtendo uma flexibilidade muito maior que a flexibilidade obtida com uma solução por *hardware* [Chen et al., 2000, Compton, 2002 e Coric et al., 2002].

Os sistemas embarcados estão sujeitos a grandes variações tecnológicas num curto espaço de tempo, ocasionada pela demanda de novas tarefas e desempenhos, e disponibilidade de novas tecnologias de sensores e atuadores, sendo comum para estas aplicações a utilização de blocos funcionais de software testados e validados, reduzindo assim, o tempo de projeto de um sistema (API – *Application Program Interface*).

No que concerne a implementação através de hardware existem blocos funcionais que podem assumir a mesma função. Estes blocos podem ser combinados com outros circuitos para programar um determinado algoritmo. Este aspecto modular permite facilitar a manutenção e a atualização de projetos [Compton, 2002, Renner et al., 2002 e Coric et al., 2002]. A utilização de IP-CORE (*Intellectual Property Cores*), permite a integração de soluções já desenvolvidas por diversos fornecedores para minimizar o tempo de projeto. Como exemplos de IP-CORE disponíveis [Ito e Carro, 2000, Kean,2000 e Diniz e Park, 2002] podem destacar os barramentos PCI, interfaces de comunicação, e funções de processamento de sinal, como FFT (*Fast Fourier Transform*), codificação e decodificação de imagens digitais e mesmo Microprocessadores e DSPs.

As PLDs (*Programmable Logic Devices*), também chamadas de EPLD (*Erasable Programmable Logic Devices*), ou ainda CPLD (*Complex Programmable Logic Devices*), e as FPGA (*Field Programmable Gate Array*) ou DFGA (*Dinamically Field Programmable Gate Array*) são dispositivos que permitem a execução de algoritmos diretamente em hardware. Com estes dispositivos programáveis é possível executar os algoritmos explorando o paralelismo inerente da solução por hardware, executando-os muito mais rápido do que se os mesmos algoritmos fossem executados de forma seqüencial por microcontroladores ou por DSPs, sujeitos ao modelo de Von Neumann. As PLDs e FPGAs apresentam diferenças quanto à aplicação e

quanto à estrutura interna. As FPGA são geralmente aplicadas em sistema com menor sensibilidade ao custo e de maior complexidade. [Miyazaki 1998, Pimentel e Le-Huy, 2000]. Miyazaki apresentada uma notação para classificação dos tipos de dispositivos lógicos:

- Lógica configurável: dispositivos lógicos que podem ser "customizados" uma única vez.
- Lógica reconfigurável: dispositivos que podem ser customizados mais de uma vez, que adotam tecnologias EPROM, EEPROM ou FLASH e podem ser reprogramados após serem montados em uma placa de circuito impresso.
- Lógica dinamicamente reconfigurável: dispositivos que permitem sua reprogramação durante a operação, mesmo após a montagem em uma placa de circuito impresso. Esta propriedade é chamada de reconfiguração *in-circuit*.
- Interconexão dinamicamente reconfigurável: expressão para dispositivos interconectados que podem ser programados por conexões pino a pino após a montagem em placa de circuito impresso.
- Lógica virtual: dispositivos dinamicamente reconfiguráveis que apresentam uma capacidade parcial de reconfiguração, onde apenas parte do dispositivo pode ser reprogramado enquanto a outra parte do dispositivo executa alguma função definida, ou seja, diferentes circuitos lógicos podem compartilhar ao longo do tempo uma mesma parte do dispositivo. São também chamados de dispositivos de lógica adaptativa ou dispositivos de lógica compartilhada.

Dentre as principais vantagens dos dispositivos lógicos estão o paralelismo, velocidade [Aporntewan e Chongstitvatana, 2001], capacidade de atualização, padronização de plataformas, amortização do custo de desenvolvimento e alto desempenho. Dentre as poucas desvantagens estão em ambientes de desenvolvimento pobres se comparados com os ambientes de desenvolvimento de processadores.

Paralelamente ao desenvolvimento de dispositivos que permitiram a implementação da computação reconfigurável, ambientes de projeto, depuração, simulação e testes foram desenvolvidos pelos diversos fabricantes. Estes ambientes permitem a criação de módulos desenvolvidos com linguagens de alto nível de abstração, chamadas linguagens de descrição de hardware, a exemplo de VHDL (*VHSIC Hardware Description Language*), Verilog e AHDL (*Altera Hardware Description Language*).

A linguagem VHDL é padrão IEEE (std 1076) para descrição de hardware, a qual propicia um ambiente integrado que possibilita o projeto, a simulação, o teste e a documentação de circuitos digitais, ou seja através da mesma podemos implementar blocos com representações de mais baixo nível de abstração, a exemplo de esquemáticos (linguagem gráfica).

A integração de blocos criados com diferentes linguagens permite a criação de projetos de forma bastante flexível, facilitando a interação entre os membros de uma equipe de trabalho. A síntese dos circuitos lógicos em FPGA é feita através de sistemas CAD (*Computer Aided Design*), permitindo a utilização de diferentes interfaces de projeto. Dentre as principais vantagens da metodologia de projeto baseado em FPGA e VHDL podemos destacar:

- Maior facilidade na interação de uma equipe de trabalho.
- Diminuição do tempo de projeto, com possibilidades de eventuais correções e de integração de novas versões.
- Opera de forma paralela, que pode ser considerada como a característica mais vantajosa em relação aos microprocessadores. Não obedece ao modelo de Von Neumann.
- Permite o desenvolvimento modular e hierárquico de um projeto.
- Existência diversas interfaces de desenvolvimento, baseadas em linguagens gráficas (esquemáticos) ou em linguagens de descrição de hardware (VHDL, Verilog, AHDL).
- Permite as abordagens de projeto *top-down* e *bottom-up*.

- Permite a utilização de funções pré-testadas (IP-CORE), diminuindo o tempo de projeto.

Diversos fabricantes de sistemas de desenvolvimento e de dispositivos lógicos reconfiguráveis disputam o mercado mundial. Entre os fabricantes de *Hardware* destacam-se Altera, Xilinx, Atmel, Triscend e Actel, e desenvolvedores de *software* apresentam produtos CAD de suporte ao desenvolvimento de projetos: Mentor Graphics, Synopsys e Accolade.

2.9 Exemplos de Aplicações de Sistemas Reconfiguráveis

A seguir apresentaremos diversas aplicações utilizando computação reconfigurável nas mais diversas áreas da Engenharia, dentre as quais: Sistemas de Comunicação, Controle, Manipulação Matemática, Tolerância a falhas, Criptografia:

a) Projeto do sistema de controle do veículo *Pathfinder*, enviado ao planeta Marte pela NASA (Woerner e Lehman, 1995);
a) Controle de inversores de potência utilizando FPGA (De Pablo et al., 1997).
b) Implementação do sistema de controle para um motor de indução usando linguagem VHDL (Empringham et al., 2000, Cirstea et al., 2000, 2001);
c) Controle adaptativo fuzzy de motor utilizando linguagem VHDL (Changuel et al., 1996);
d) Projeto de um sistema de controle digital utilizando FPGA para uma cadeira de rodas (Chen et al., 2000, Lima, 2003);
e) Implementação de algoritmos em *hardware* para acelerar o processamento de imagens médicas para tomografia computadorizada (Coric et al., 2002);

f) Implementação de algoritmos de controle de aceleração e frenagem para motores de CNC e robôs industriais usando VHDL (Endemano, 2002, Jeon et al., 2002);
g) Controladores de motor AC usando FPGA (Kharrat et al., 1998).

h) Sistema de análise de tolerância às falhas baseado em ferramentas implementadas em VHDL (Baraza et al., 2002);
i) Algoritmos de processamento de sinal de vídeo de alta resolução (HDTV) (Dido et al., 2002);
j) Projeto de sistemas de navegação por satélite através de FPGA (Thor e Akos, 2002).
k) Implementação de algoritmos em FPGA para reconhecimento de voz (Varga et al., 2001).
l) Sistema de monitoração de ambiente radioativo através de FPGA (Hasuko et al., 2001).
m) Metodologia de projeto de controladores baseados em *hardware* e *software* para aplicações mecatrônicas (Renner et al., 2002).
n) Arquiteturas mais utilizadas para computação reconfigurável, metodologias de projetos com sistemas reconfiguráveis (Page, 1996, Pimentel e Le-Huy, 2002, Yasunaga et al., 2000, Harkin et al., 2001).
o) Descrição de um compilador C para sistemas com microcontroladores que tenham unidades funcionais reconfiguráveis (Ye et al., 2000);
p) Paralelismo em dispositivos reconfiguráveis e técnicas para implementação de controladores em Paralelo usando Redes de Petri (Snider et al., 2001, Kozlowski et al., 1995);
q) Esquema de criptografia para utilização em funções IP-CORE (Kean, 2000).

Do mesmo modo, a literatura mostra a utilização de FPGA para a implementação de funções matemáticas, controladores digitais, filtros e controladores avançados:

a) Implementação de uma PLL digital (Phase-locked loop) (Kobayashi e Haratsu, 1995);
b) Projeto de algoritmos matemáticos e equações diferenças usando VHDL (Kollig et al., 1996 e Kollig et al., 1997);
c) Implementação de funções aritméticas em FPGAs (Klotchkov e Pedersen, 1996);

d) Implementação de controladores PID digitais (Samet et al., 1998), algoritmos fuzzy para a sintonia de PID (Hu e Li, 1996) e modelagem, simulação e implementação de controladores em lógica reconfigurável (Wegrzyn et al., 1998);

e) Implementação através de FPGA e VHDL de controladores por espaço de estados (Garbergs e Sohlberg, 1996, 1998);

f) Desenvolvimento de Filtro preditor de erro em lógica reconfigurável (Hwang e Han, 1999);

g) Implementação de algoritmo FFT em lógica reconfigurável (Fanucci, 2002);

h) *Hardware* dedicado para filtros de processamento de imagem (Khriji et al., 1999).

i) Abordagem de *software* e *hardware* para algoritmos genéticos (Harkin et al., 2001);

j) Algoritmo de código corretor de erro em sistemas de comunicação digital usando hardware reconfigurável (Swaminathan et al., 2002);

k) Implementação em FPGA de sistemas de acionamento e controle: controlador PI (proporcional e integral), um gerador PWM (modulação por largura de pulso) e um conversor digital-analógico (DA), (Pimentel e Le-Huy, 2000);

l) Projeto de interfaces em sistemas de controle baseados em microprocessadores (Vargas et al., 2001);

m) Conceito de computação digital-serial com implementação de filtros FIR (Finite Impulse Response) nas formas inversa e canônica usando FPGA (Valls et al., 1998, Peiró et al., 1999) e Metodologia de síntese de arquiteturas para implementação de filtros IIR (Infinite Impulse Response) em FPGAs (Rathna et al., 1994) e implementação de filtro adaptativo não-linear usando FPGA (Franco et al., 2000);

n) Implementação de código CRC (Cyclic Redundancy Checking) para uso em sistemas de telecomunicação com o uso de FPGA (Monteiro et al., 2001);

o) Implementação em hardware reconfigurável de máquinas de estado finitas (Köster e Teich, 2002);

p) Aplicação de redes neurais usando VHDL (Acosta e Tosini, 2001);

q) Apresentação de técnica baseada em computação reconfigurável para processamento criptográfico (Yamaguchi et al., 2000).

2.10 Ferramenta de Projeto Quartus II

Os blocos desenvolvidos para serem programados nas FPGAs usadas no ambiente proposto foram desenvolvidos, testados e simulados com a ferramenta de projeto Quartus II [Altera, 2008], através de um programa de incentivo educacional (*Altera University Program*) da empresa Altera utilizado na Faculdade de Engenharia Mecânica da UNICAMP, nos cursos de pós-graduação e graduação, permitindo a utilização destas ferramentas de projeto com custo relativamente baixo.

Esta ferramenta apresenta interfaces de projeto em linguagem gráfica, linguagem VHDL,Verilog e linguagem AHDL, com filosofia de projeto hierárquico, onde sub-blocos podem ser detalhados a partir de blocos hierarquicamente superiores. Estes blocos e sub-blocos podem ser projetados utilizando-se de diversas linguagens descritivas de *hardware*.

É importante destacar a existência de outras ferramentas de projeto disponíveis no mercado, onde os blocos em lógica reconfigurável desenvolvidos nesse trabalho de pesquisa poderão ser facilmente reproduzidos em ferramentas de outros fabricantes.

A figura 2.13 apresenta exemplo de tela gráfica na qual a visão geral de um projeto é apresentada em linguagem gráfica (esquemático). Neste projeto denominado "PWM_top", podem ser observados dois sub-blocos: "pwm" e "dsp7seg" descritos em VHDL e interligados em linguagem gráfica.

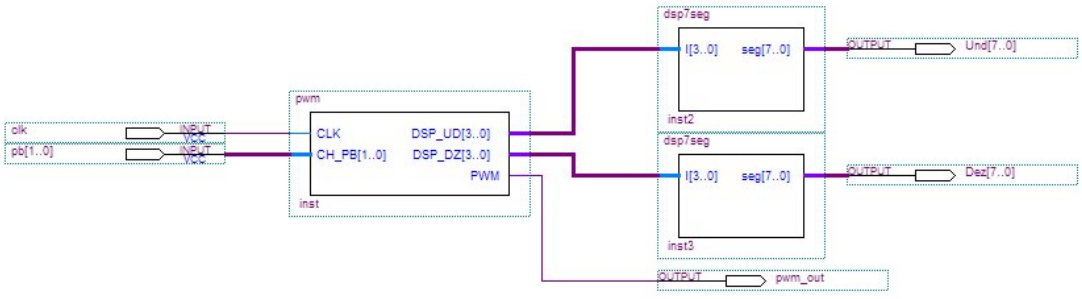


Figura 2.13: Projeto PWM_top em linguagem gráfica (esquemático).

O sub-bloco “pwm”, visto parcialmente na Figura 2.14, é projetado em linguagem VHDL. É comum nos diversos projetos desenvolvidos neste trabalho, a utilização de linguagem gráfica em alguns blocos e de linguagem VHDL em outros blocos. Isto se deve à adequação de uma ou outra linguagem a uma determinada função que se deseja implementar. Geralmente, por motivo de clareza na documentação, considerando uma abordagem *top-down* do projeto, é utilizada a linguagem gráfica para os blocos de mais alto nível.

A ferramenta Quartus II oferece a possibilidade de realizar a simulação do projeto a ser implementado. Isto permite a rápida depuração do projeto, abreviando o tempo do mesmo. A figura 2.15 apresenta a simulação do projeto “PWM_top”.

```
ARCHITECTURE a OF pwm IS
     SIGNAL pulsopwm : STD_LOGIC_VECTOR(7 DOWNTO 0);
BEGIN

PROCESS (CLK)
     VARIABLE vcntpwm : STD_LOGIC_VECTOR(7 DOWNTO 0);
 BEGIN
IF CLK'event AND CLK='1' THEN
     vcntpwm := vcntpwm+1;
     IF vcntpwm > pulsopwm THEN pwm <= '0'; ELSE pwm <= '1'; END IF;
 END IF;
 END PROCESS ;

PROCESS (clk)
     VARIABLE vsetpoint : STD_LOGIC_VECTOR(7 DOWNTO 0);
 BEGIN
IF clk'EVENT AND clk='1' THEN
            if CH_PB(0)='0' AND vsetpoint < x"FF" then vsetpoint := vsetpoint + 1; end if;
            if CH_PB(1)='0' AND vsetpoint > x"00" then vsetpoint := vsetpoint - 1;  end if;
 END IF;
 pulsopwm <= vsetpoint;

 END PROCESS ;

 DSP_UD <= pulsopwm(3 downto 0);
 DSP_DZ <= pulsopwm(7 downto 4);
```

Figura 2.14: Representação de um sub bloco de projeto em linguagem VHDL

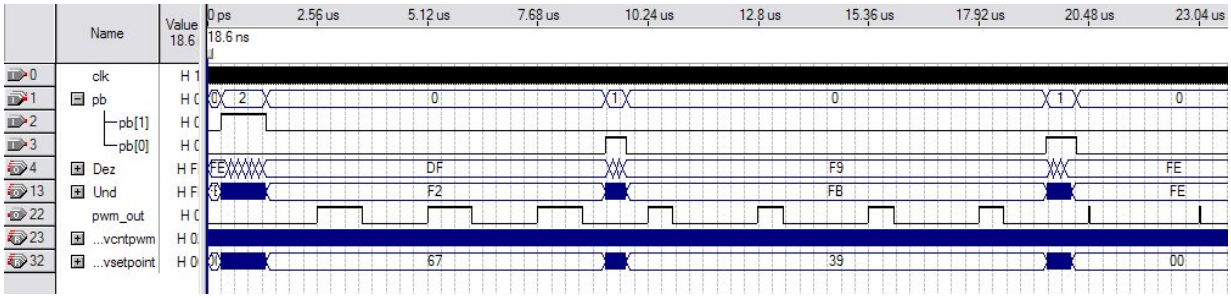


Figura 2.15: Simulação de um projeto em forma de onda

No próximo capítulo deste trabalho serão apresentados ambientes para prototipagem rápida e concepção de sistemas embarcados, onde são apresentados objetivos e propriedades destes sistemas, com ênfase nos aspectos de implementação de *hardware* e *software*, sendo também proposto neste capítulo uma estrutura mínima necessária para o funcionamento desse ambiente.

2.11 Conclusões

Neste capitulo foram apresentados aspectos sobre ferramentas de prototipagem rápida de sistemas mecatrônicos. Dentre as principais vantagens da utilização dessas ferramentas, pode-se destacar:

- Possibilidade de programação através de blocos estruturados, com uma modelagem hierarquizada; a utilização de softwares que permitem a modelagem de componentes mecânicos a partir da simulação do movimento; a análise direta dos movimentos sem necessidade de derivar as equações.
- Diminuição do ciclo de concepção e interfaces convenientes com o usuário.
- Possibilidade de implementação de algoritmos de controle adequados ao problema em estudo.
- Melhor produtividade, acarretando menores atrasos, custos de desenvolvimento reduzidos e uma melhor qualidade.
- Busca de ambiente de integração único, possibilitando o cálculo cientifico, modelagem de sistemas e simulação, análise de dados e visualização, e implementação de *software* embutido em tempo real.

Capítulo 3

Ambientes de Prototipagem Rápida para concepção de Sistemas Embarcados

3.1 Conceitos Básicos de Lógica Reprogramável

Um circuito digital é formado basicamente de uma equação lógica com operadores do tipo AND , OR e NOT para circuitos combinacionais, como também blocos de registradores (flip-flops) no caso de lógicas seqüenciais. Inicialmente, estes circuitos integrados eram projetados utilizando-se famílias de circuitos comerciais com estas funções e com blocos, altamente utilizados em projetos, como contadores e decodificadores, sendo então interligados via circuito impresso para formar a lógica desejada, como exemplo podemos citar a família 74xx.

Em 1978, a MMI (Monolithic Memories, Inc) lançou no mercado um componente denominado PAL (*Programmable array logic*), que possui internamente uma matriz de interconexões entre portas lógicas programável, de modo que uma certa quantidade de equações poderiam ser programadas em um único componente. Esta tecnologia proporcionou o desenvolvimento de circuitos menores e com freqüências de operação maiores devido à redução dos atrasos de propagação dos sinais pelo circuito. A figura 3.1 ilustra parte de um circuito de uma PAL, onde as linhas verticais da matriz representam as entradas do componente, e as horizontais às entradas das portas lógicas AND da equação a ser programada.

Dentre as limitações de utilização de uma PAL, podemos destacar: a não possibilidade de implementar um grande número de equações em um mesmo componente (8 a 10 no máximo),

pouca flexibilidade no uso de lógica com registros ou combinações, devido a quantidade de saídas com registradores ser fixa. Ao mesmo tempo, a matriz de interconexão ser programada com tecnologia que não permitia a regravação do componente em caso de alteração da lógica.

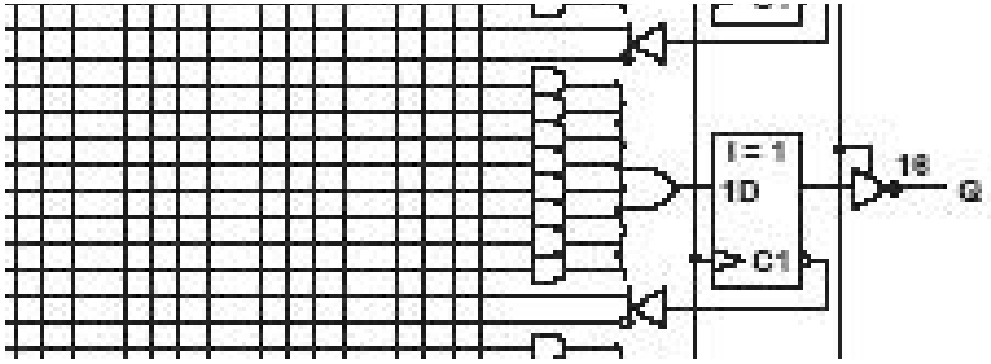


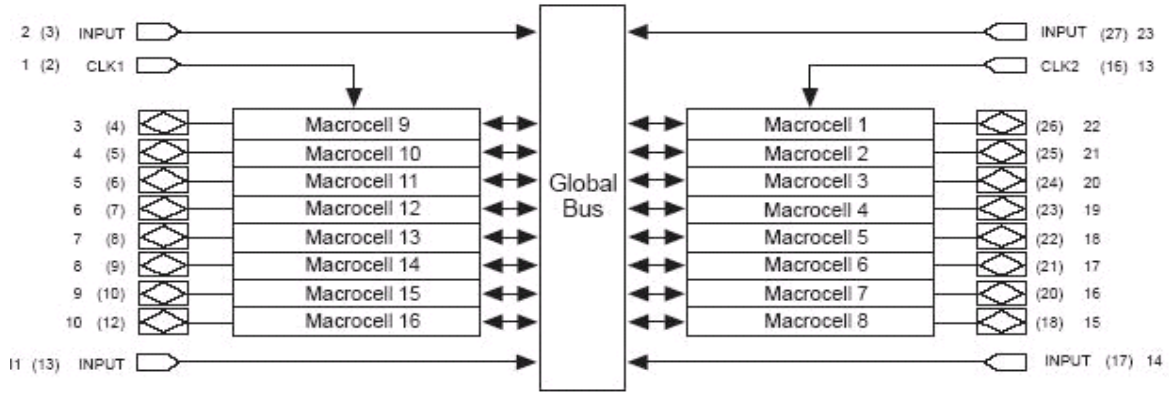
Figura 3.1: Diagrama interno de uma PAL.[Texas, 1992]

Anos depois, a Lattice Semiconductor lançou no mercado as GAL (*Generic array logic*) que possuem a mesma arquitetura encontrada nas PAL, com a vantagem de serem reprogramáveis e com tipo de saída programável, permitindo ainda a utilização de registradores de saída.

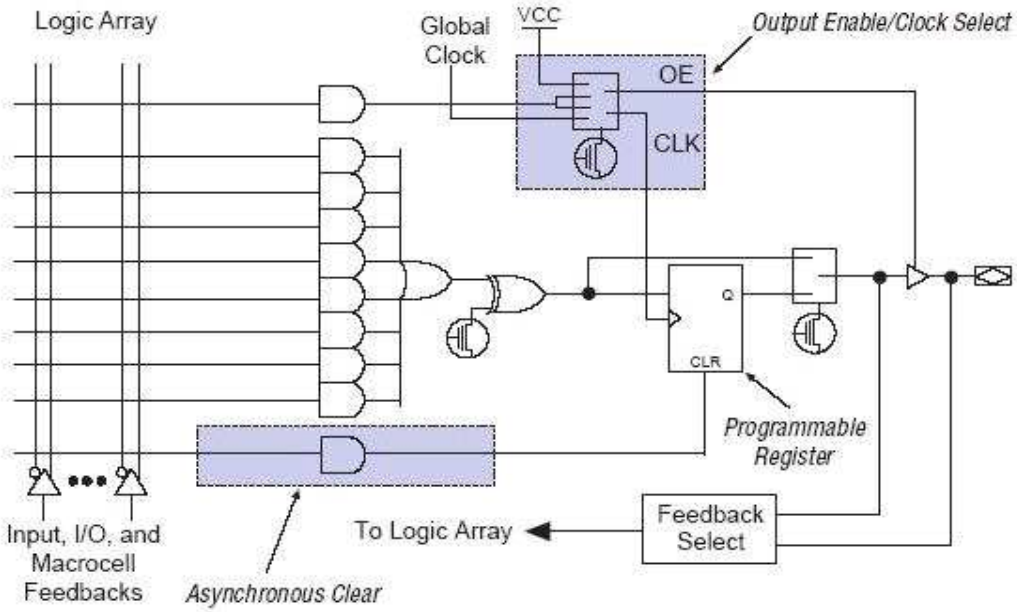
Em 1984 a ALTERA^{TM} colocou no mercado a família de componentes programáveis denominados EPLD (*Erasable Programable Logic Device*,) que possibilitava a utilização de uma maior quantidade de equações e registradores num único componente em relação às PAL e GAL convencionais.

As EPLDs são formadas por vários blocos denominados de Macro-células (figura 3.2a), onde cada macro-célula possui uma matriz de interconexão com sinais e entradas de portas AND, consecutivamente interligada a uma porta OR e uma porta XOR, onde a saída da equação pode ou não utilizar um registrador disponível na macro-célula (figura 3.2b).

Posteriormente a ALTERA lançou uma nova família com o nome de CPLD (*Complex Programable Logic Device*), bem como *software* para o projeto e programação destes componentes.



a) Diagrama interno de uma EPLD



b) Diagrama interno de uma Macro-célula

Figura 3.2: EPLD - Erasable Programable Logic Device. [ALTERA, 1999]

A CLPD é basicamente várias EPLDS em um mesmo CHIP, porém com uma arquitetura mais complexa e flexível. Um bloco formado de 16 macro-células é agrupado formando um LAB (*Logic Array Block*). Estes sinais de entrada e saídas dos LABs podem ser interligados através de uma matriz de interconexão chamada PIA (*Programable Interconect Array*). Os pinos físicos do componente se interligam através de blocos de controle de *I/O* (Figura 3.3). Existem diversos tamanhos de CPLD disponíveis em cada família, partindo de componentes com 32 até 512 macro-células.

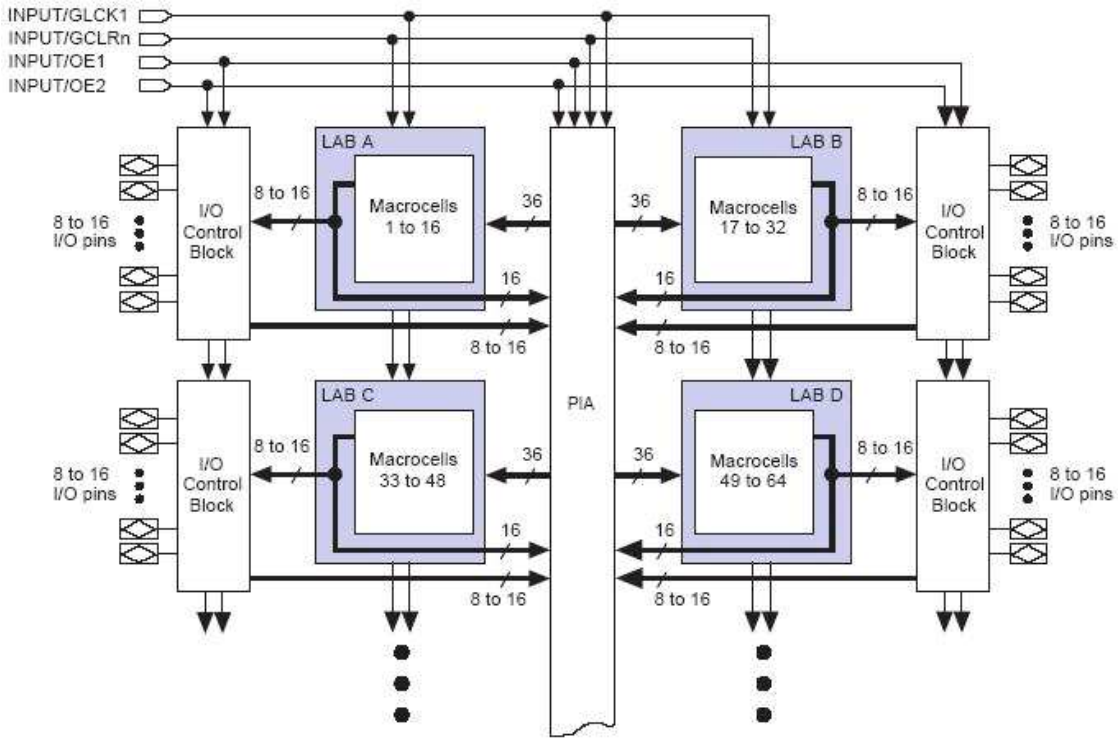


Figura 3.3: Diagrama interno de um CPLD.[ALTERA, 2005]

A macro-célula de uma CPLD (figura 3.4) é mais complexa das utilizadas numa EPLD, permitindo assim uma maior integração de lógica e um melhor aproveitamento dos recursos colocados internamente ao silício.

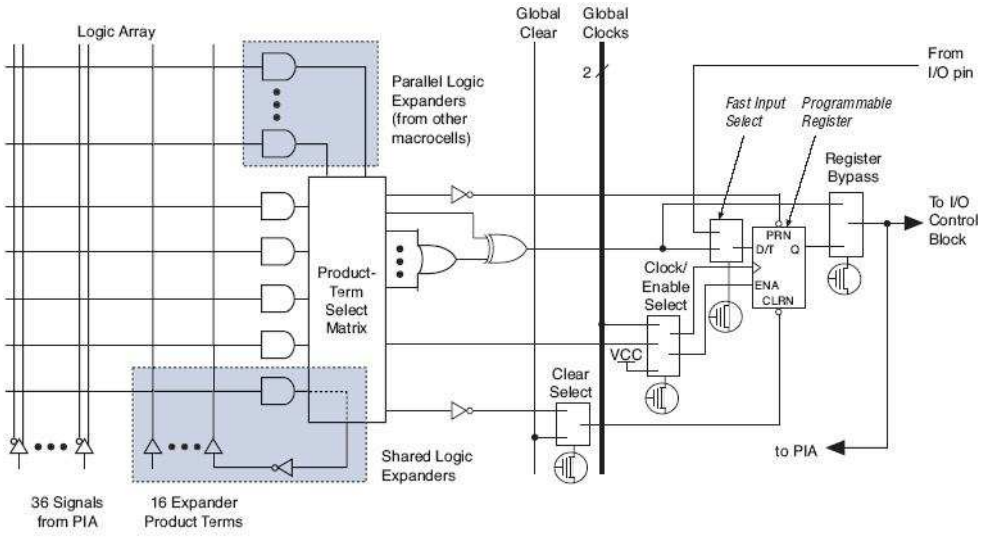


Figura 3.4: Diagrama interno de uma macro-célula.[ALTERA,2005]

Em 1985, XILINX revolucionou o mercado com o lançamento de uma nova arquitetura de lógicas programáveis, as FPGA (*Field Programmable Gate Array*). Nesta nova tecnologia a equação lógica não é mais formada por interligação de portas lógicas, mas sim por interligações de memórias RAM de 4 entradas e 1 saída que armazena a "tabela verdade" da equação desejada.

Uma das principais vantagens de PALs, EPLDs e CPLDs é que estes dispositivos são programados, isto é utilizam conexões programáveis do tipo não volátil. Por outro lado, as interconexões de um FPGA são voláteis, e são desfeitas quando não há alimentação no componente, necessitando assim, para a programação destas interligações, de uma EPROM serial externa ao componente que armazena o conteúdo do mapeamento das interconexões, e estas informações são transferidas automaticamente após o *Power-On* do circuito. [XilinX,1998]

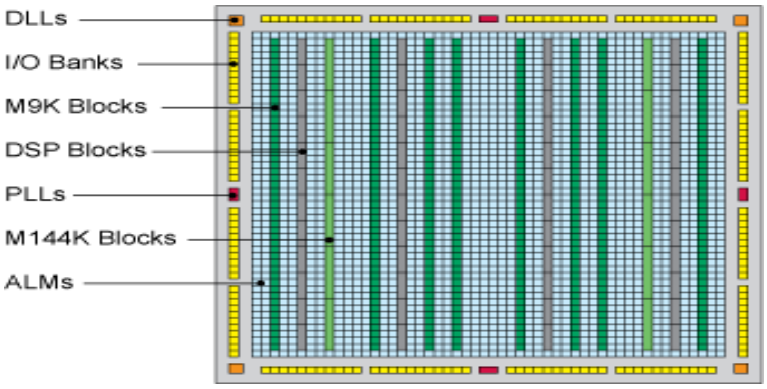
Em 1992, a ALTERA coloca no mercado sua primeira família de FPGA, muito similar com a XILINX, porém com melhorias na disposição dos elementos lógicos, possibilitando assim, um

melhor roteamento das interligações. Hoje temos diversos fabricantes de FPGA, mas a ALTERA e XILINX se destacam como lideres de mercado.

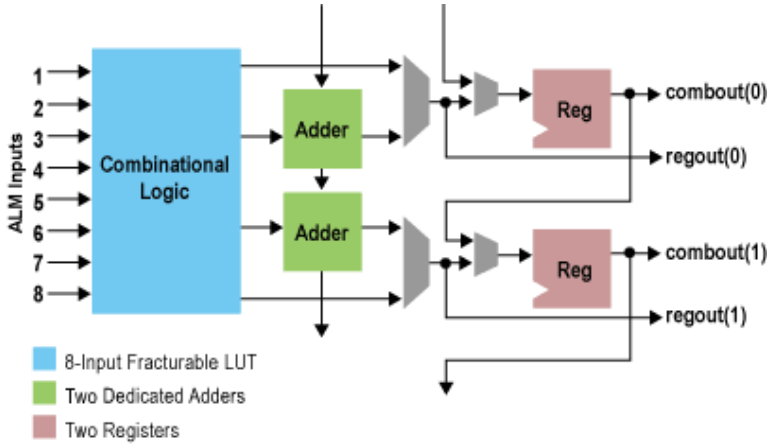
Com a evolução da micro-eletrônica estes fabricantes chegaram a uma densidade de lógica disponível em uma FPGA e uma quantidade de recursos adicionais como memória, blocos de DSP e outras, que já é possível se colocarem um sistema inteiro em um único CHIP, contendo processadores, memórias e periféricos.

A família Stratix III da ALTERA (figura 3.5a), por exemplo, possui componentes com capacidade de mais de 330 mil equações lógicas e de 270 mil registradores. O elemento lógico da Stratix (ALM - *Adaptive Logic Modules*, figura 3.5b) pode implementar até 2 equações e possui ainda 2 blocos somadores e 2 registradores.

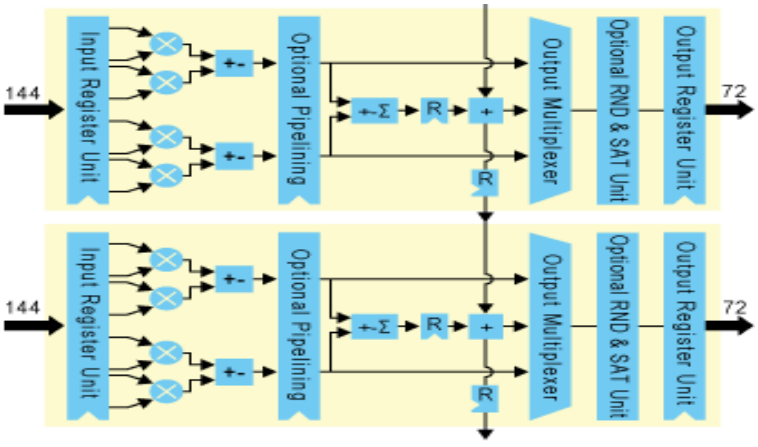
Para a implementação de funções de processamento digital a família possui blocos de uma DSP do Stratix (figura 3.5c) que possuem multiplicadores, somadores-acumuladores e registradores, podendo ser configurados conforme a necessidade de uma aplicação.



a) Diagrama de um silício.



b) Diagrama de um elemento lógico.



c) Diagrama de blocos DSP.

Figura 3.5: FPGA família Stratix III. [ALTERA, 2007].

3.2 Descrição de Ferramentas para desenvolvimento de FPGA

3.2.1 Ferramenta de projeto Quartus II

O *software* Quartus II é uma das ferramentas disponíveis para o desenvolvimento de FPGA. Essa ferramenta é disponibilizada gratuitamente para o projetista pela ALTERA, porém com limitação de algumas funcionalidades e comercializadas na versão completa do software.

O Quartus II disponibiliza todas as ferramentas necessárias para a elaboração, simulação, roteamento e testes de um FPGA, com possibilidade de desenvolvimento de circuitos de pequena integração e baixa complexidade, como contadores e decodificadores, até circuitos de alta complexidade e alta integração, como processadores, filtros digitais, interfaces de comunicação e outros. A figura 3.6 apresenta uma tela típica deste ambiente de desenvolvimento, com o editor gráfico, relatório de atrasos, simulação e definição de pinos do componente.

O ciclo de desenvolvimento de um projeto em um FPGA segue os seguintes passos:

- ***Design Specification***: Definições de características e interfaces
- ***Design Entry*****:** Entrada do projeto em formato de Esquema elétrico, Diagrama de Blocos ou VHDL/Verilog.
- ***Synthesis*****:** Transformação o projeto em primitivas do FPGA e otimização de área e atrasos entre sinais.
- ***RTL Simulation*****:** Simulação funcional do circuito e validação da lógica do projeto.
- ***Place & Route*****:** Posicionamento das primitivas dentro da tecnologia escolhida e execução do roteamento das interligações.
- ***Timing Analysis*****:** Verificação e análise das especificações de tempos de propagação do projeto.
- ***Gate Level Simulation*****:** Simulação do circuito com as características de atrasos físicos do componente conforme a informação do roteamento elaborado.

- ***PC Board Simulation & Test*:** Programação do componente na placa de circuito impresso e verificação dos sinais.

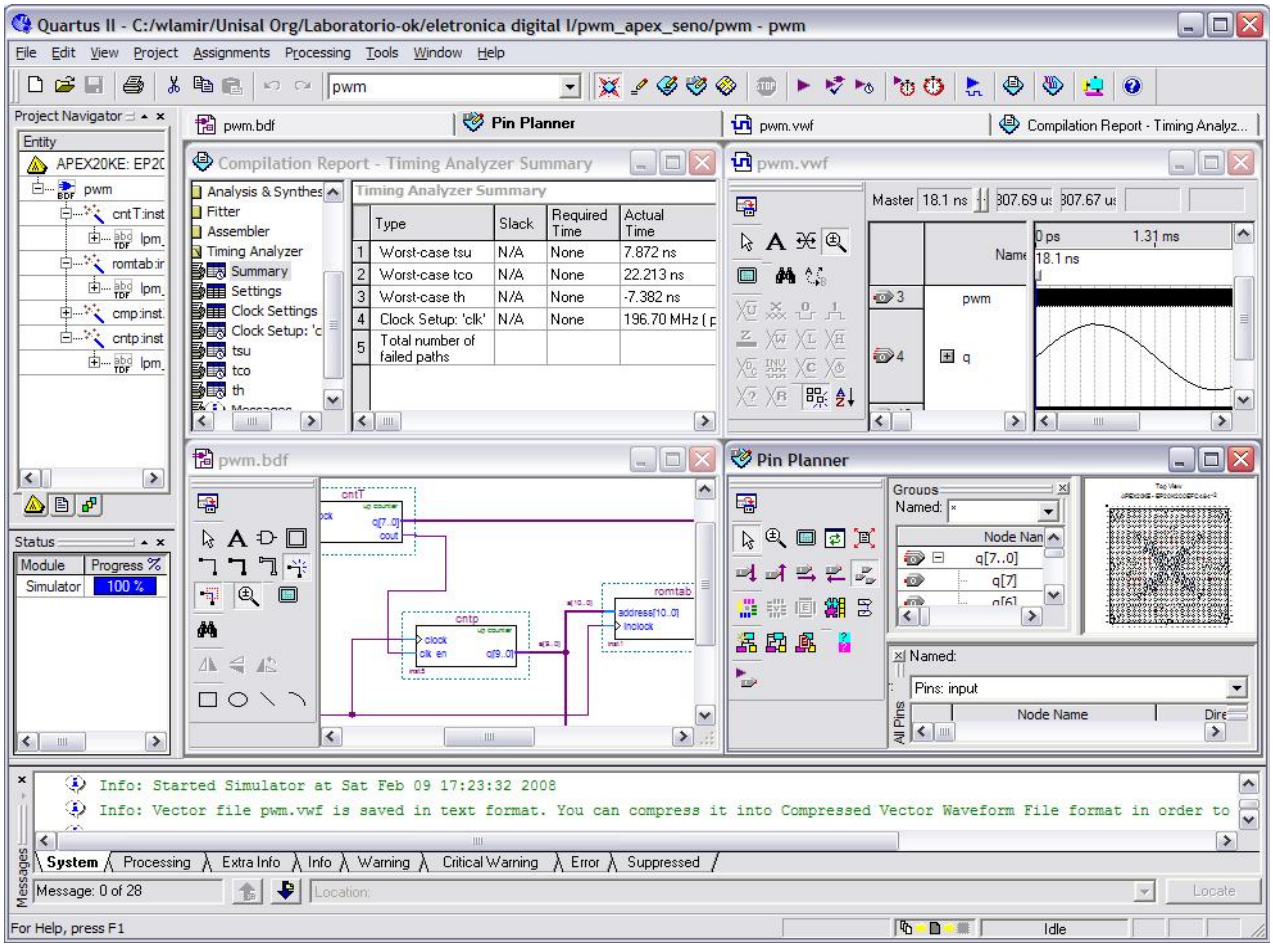


Figura 3.6: Projeto em linguagem gráfica utilizando a ferramenta Quartus II.

3.2.2 Implementação de um projeto

Um projeto de uma FPGA consiste de um conjunto de vários arquivos estruturados contendo as especificações lógicas do circuito bem como as especificações elétricas e físicas do componente. Estes arquivos são interligados através de uma construção hierárquica (figura 3.7),

sendo que um projeto deverá ter um arquivo principal de topo de hierarquia, este arquivo irá apontar para os demais arquivos do projeto que poderão por sua vez apontar para outros arquivos.

Na entrada do projeto, há a optação em elaboração via editor gráfico (esquema) ou texto (VHDL/Verilog). Um projeto pode conter vários tipos de arquivos diferentes, permitindo flexibilidade na elaboração de um projeto. [ALTERA, 2007]

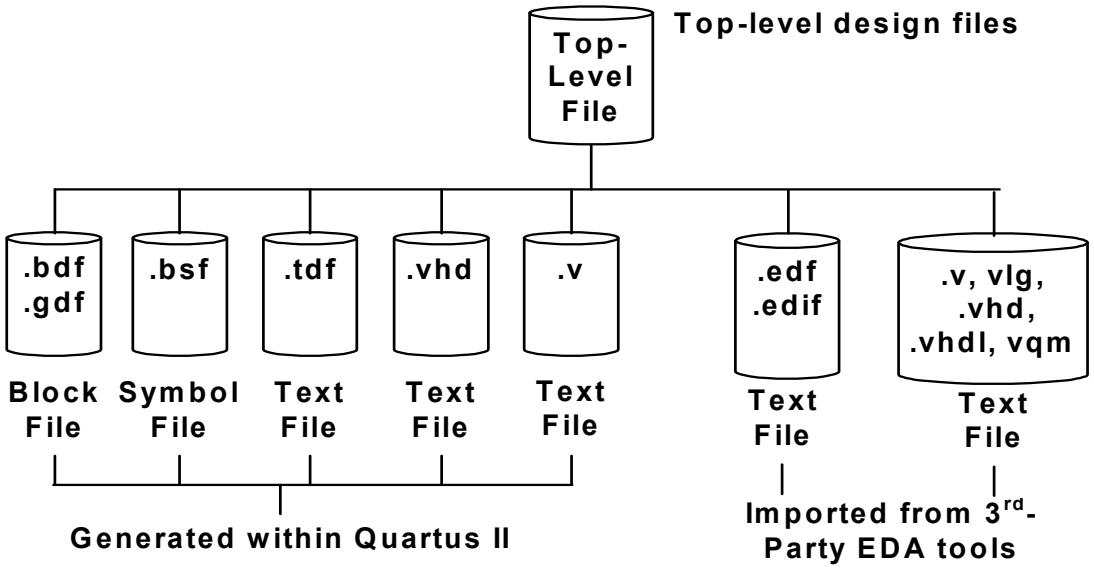


Figura 3.7: Hierarquia de Projeto – Tipos de arquivos.

3.2.3 Entrada de dados por esquema

O editor gráfico será utilizado para a entrada de esquema de um arquivo tipo DBF (*Block Design File*). Uma interface de simples operação permite a implementação de forma rápida do circuito desejado.

Para a entrada de circuitos, este editor possui várias bibliotecas contendo blocos funcionais básicos de sistemas digitais bem como funções já pré-definidas e complexas. Nas funções básicas

podemos citar a as funções lógicas combinacionais (*and,or, not, nor, xor, ...*) , Registradores (*flip-flop, latch*) e pinos de entrada e saída do circuito (figura 3.8).

Uma biblioteca de grande utilidade é a dos componentes da família 74xx (figura 3.9), que possui praticamente todas as funções destes tradicionais componentes utilizadas em projetos digitais. Tal biblioteca permite uma migração rápida de projetos anteriores à tecnologia FPGA, pois basta redesenhar o circuito neste ambiente sem a necessidade de um reprojeto. Permite também que o conhecimento pré-adquirido desta família, (com tempo de experiência e outros fatores) possa ser aproveitado para novos projetos. Porém a utilização desta biblioteca torna-se inviável em projetos de grande complexidade devido à quantidade de desenhos e interligações a serem elaborados, principalmente com circuitos com barramentos de processadores.

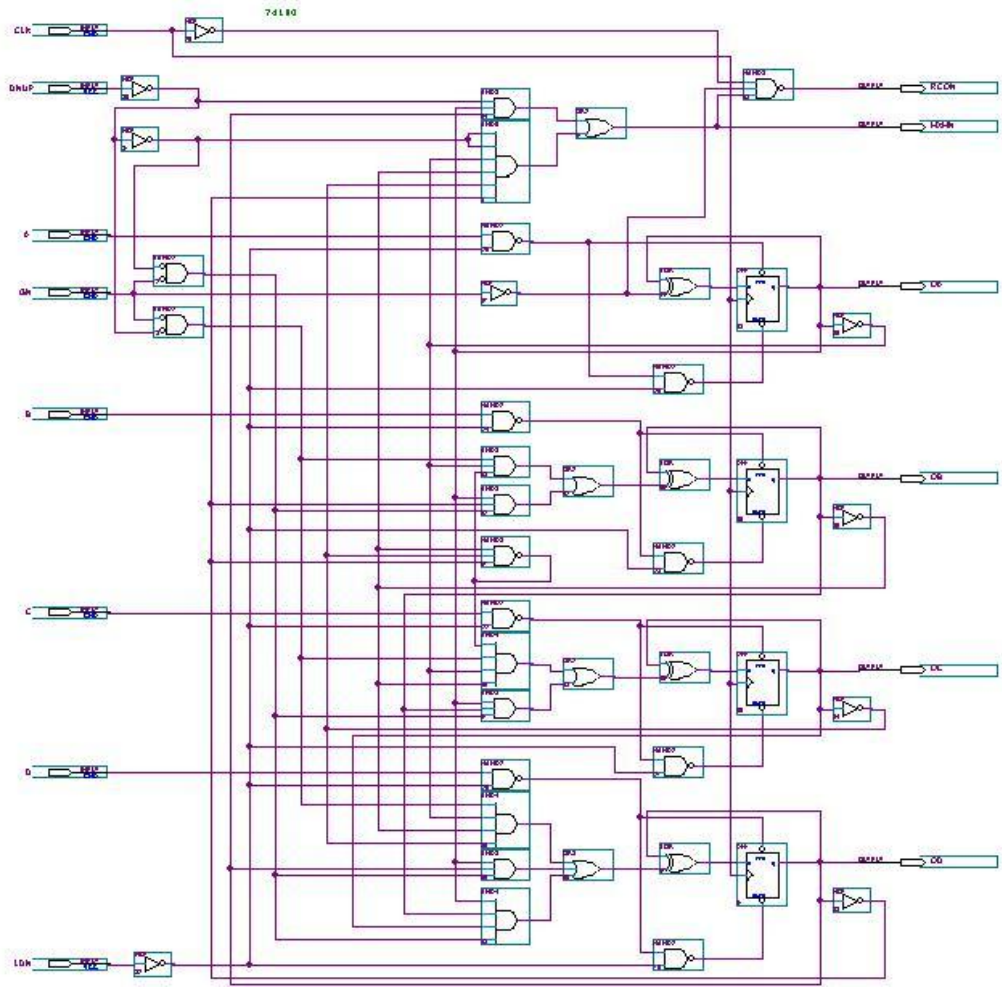


Figura 3.8: Projeto de um contador com portas lógicas e flip-flop.

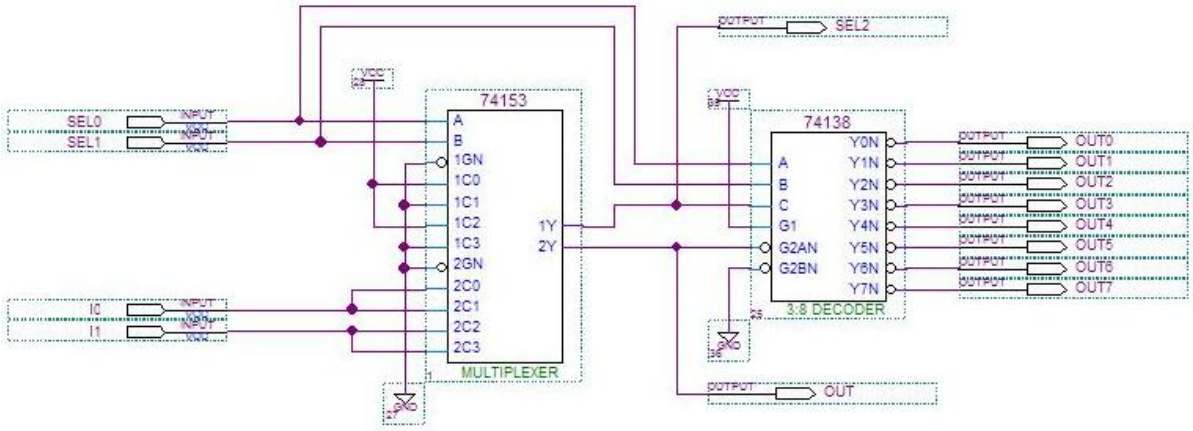


Figura 3.9: Projeto com biblioteca 74xx.

Para solucionar este problema a ALTERA disponibiliza uma biblioteca de mega-funções, ou seja, blocos configuráveis com as principais funções ou interfaces necessárias para projetos em sistemas digitais. Tais blocos permitem um grande ganho no processo de desenvolvimento de um projeto pois o projetista se concentra mais no modo de como implementar a solução e não mais em detalhes de como interligar pequenos blocos como os da família 74xx (figura 3.9). A versatilidade desta ferramenta permite que além destas bibliotecas disponíveis pela ALTERA, outras bibliotecas especificas poderão ser compradas ou desenvolvidas pelo próprio projetista.

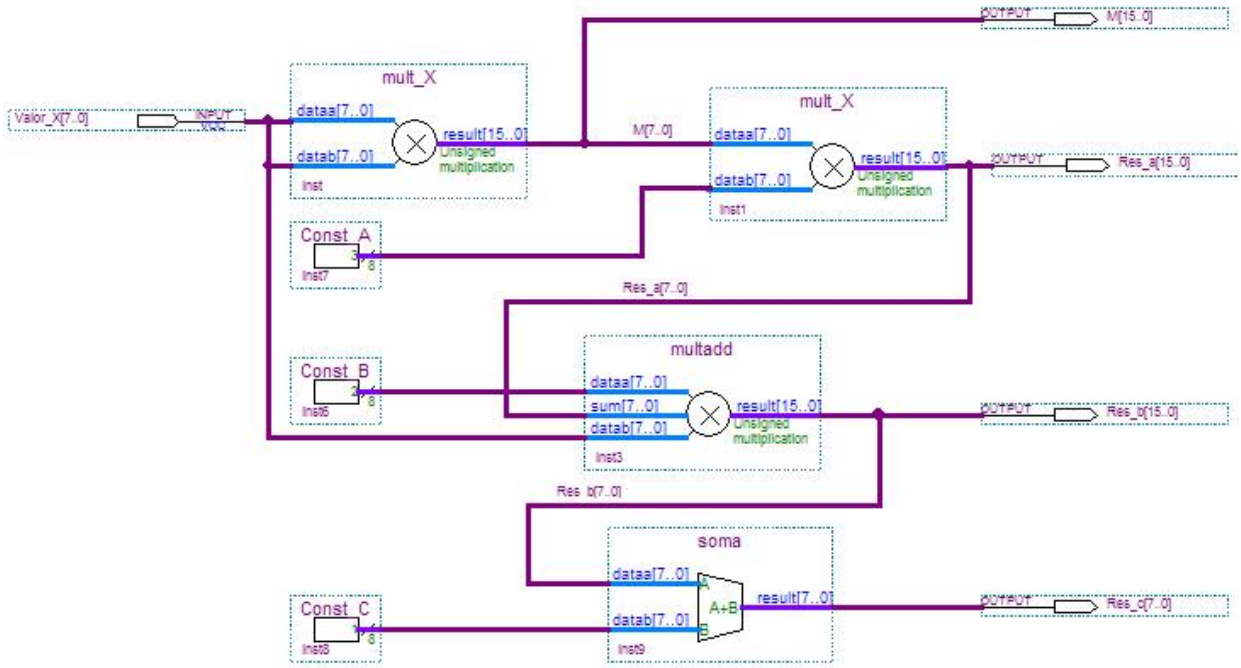


Figura 3.9: Projeto com blocos configuráveis.

3.2.4 Entrada em arquivos formato Texto

A entrada de projetos em formato texto é utilizada quando o projeto é elaborado com linguagens de descrição de hardware, tais como VHDL e Verilog. Estas linguagens permitem que o projeto seja descrito em forma de modelos comportamentais. Estes modelos são convertidos pelo software em um hardware, conforme a tecnologia utilizada.

Na década de 80 o Departamento de Defesa Americano (DoD) criou a linguagem VHDL (*Very High Speed Integrated Circuits (VHSIC) Hardware Description Language*) de modo a permitir uma padronização entre as empresas prestadoras de serviços bem como uma documentação clara para projetos de complexidade crescente.

Em 1987 o VHDL foi normalizado pelo IEEE (*Institute of Electrical and Electronics Engineers, Inc*.) tornando-se um padrão mundial para projetos de circuitos integrados digitais. Nos dias atuais, essa linguagem é adotada como padrão no desenvolvimento de projetos industriais e todas as ferramentas comerciais de desenvolvimento de projeto, aceitam esta linguagem como modo de entrada de dados.[Perry 2002]

3.2.5 Descrição das etapas para concepção e síntese de um FPGA

3.2.5.1 Síntese RTL

A síntese RTL (*Register Transfer Level*) consiste na transformação dos arquivos de projetos em seus diversos formatos (Esquemático, VHDL e Verilog) em um circuito lógico formado por primitivas como funções lógicas (AND, OR, NOT) e lógicas seqüenciais (Flip-Flop). Após esta transformação, um processo de otimização lógica é executado para minimizar o circuito, objetivando um melhor aproveitamento do silício e um melhor desempenho de operação. A figura 3.10 sintetiza as diferentes etapas para síntese lógica.

3.2.5.2 Simulação RTL

Após a síntese do circuito uma simulação tipo funcional pode ser executada. Este tipo de simulação é utilizado para se verificar o funcionamento lógico do projeto, validação dos modelos e validação da síntese. Nesta etapa, não é verificado características de atrasos, pois ainda não temos definições de roteamento do componente.

3.2.5.3 Place & Route

Com a lógica já definida, a próxima etapa é o posicionamento e roteamento do circuito elaborado dentro das interligações pré-definidas no silício do FPGA. Há diversas opções e especificações que podem alterar a forma com que o software execute este posicionamento, de modo a se atingir as especificações de *timing* do projeto.

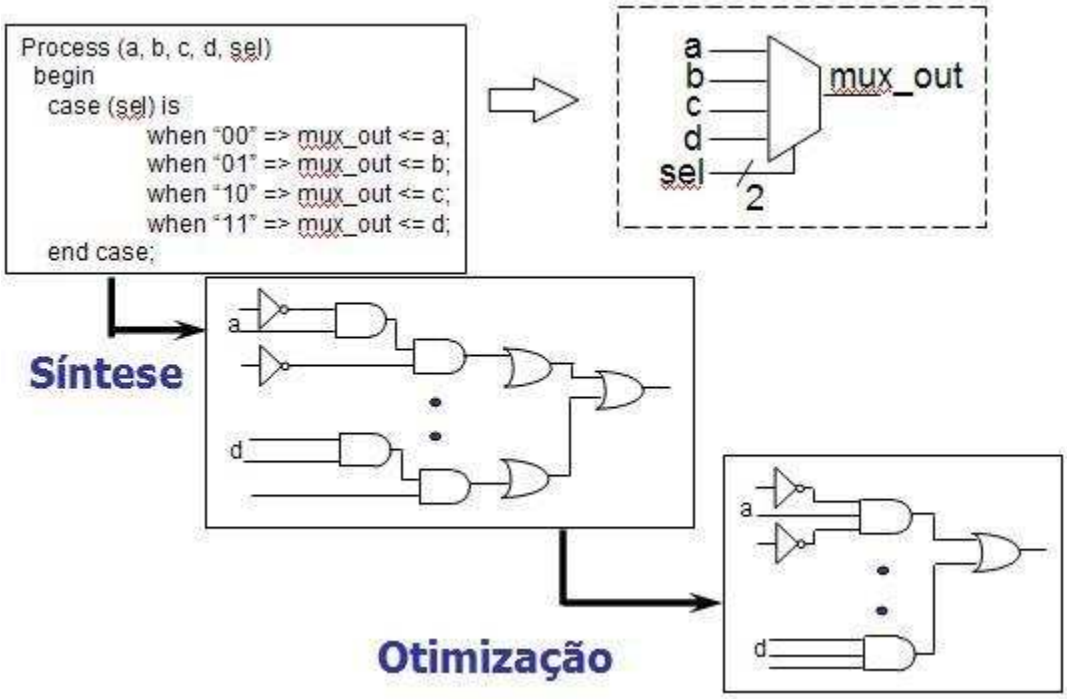


Figura 3.10: Síntese Lógica.

3.2.5.4 Timing Analysis

O processo de *Timing Analysis* é utilizado para verificar se as especificações definidas no projeto foram atingidas e para o levantamento de todos os atrasos dos sinais devido ao roteamento elaborado.

3.2.5.5 Gate Level Simulation

Através dos dados de roteamento e atrasos dos sinais, pode-se então executar simulações levando em conta estas características físicas do componente. Neste tipo de simulação será verificado o funcionamento do projeto focando-se nas freqüências de operação do circuito, problemas de pulsos indesejáveis devido a atrasos (*glitch*) e validação do roteamento executado.

3.2.5.6 PC Board Simulation & Test

A ultima etapa deste processo é a gravação do roteamento no componente e verificação final de funcionamento. O Quartus II possui diversas ferramentas para a verificação de funcionamento em tempo real, como analisador lógico e editores de memória que utilizam a própria interface de gravação JTAG (*Joint Test Action Group* ou IEEE 1149.1 standard).

3.3 Ambiente de simulação e prototipagem MATLAB Simulink®

A Mathworks desenvolveu em 1984 uma linguagem dedicada para desenvolvimento e simulação de algoritmos matemáticos – o MATLAB®. Desde o inicio de seu desenvolvimento, esta linguagem incorporou as mais diversas aplicações, sendo líder de mercado, possuindo um grande numero de usuários tanto no mundo acadêmico como industrial, e é considerada uma ferramenta indispensável para o ensino e desenvolvimento industrial (figura 3.11).

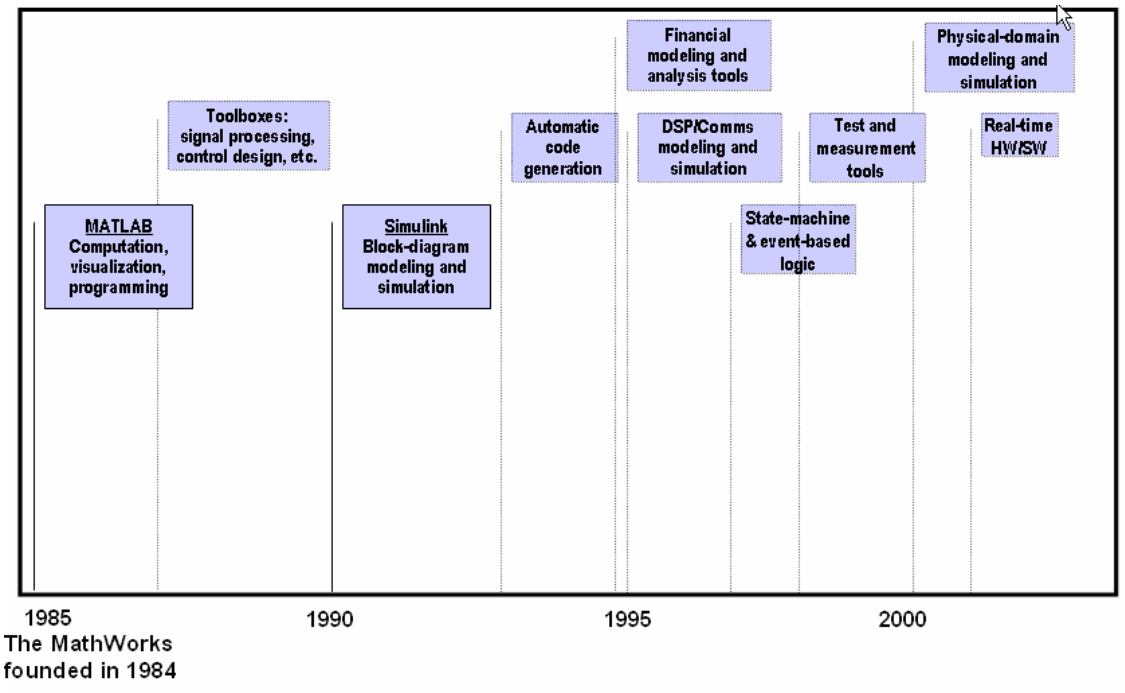


Figura 3.11: A evolução da Mathworks desde a sua criação.

3.3.1 Aplicações Multi-domínios

Nos dias atuais o MATLAB®, pode ser considerado uma ferramenta para simulação e desenvolvimento em multi-domínios de aplicação (figura 3.13), possuindo uma grande variedade de ferramentas matemáticas dedicadas, destinadas a diversas aplicações, tais como processamento de imagens, comunicações, controle de sistemas, modelagem matemática e outros.

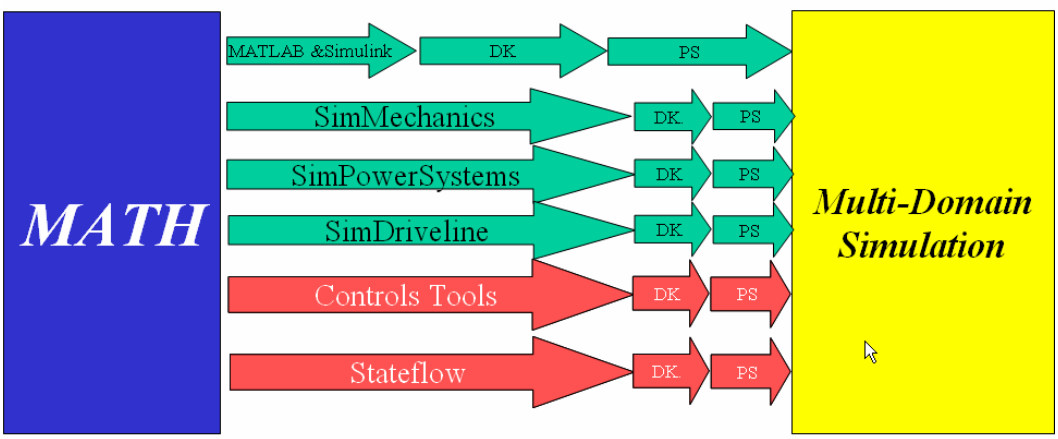
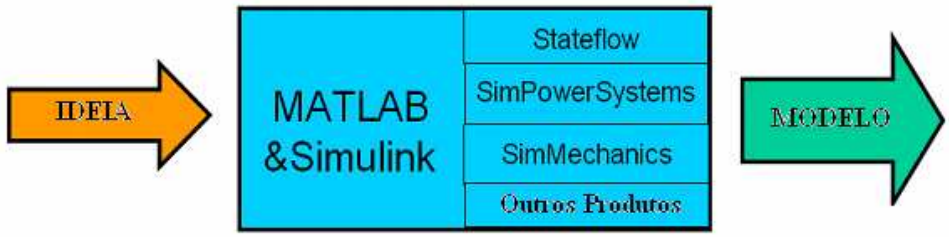
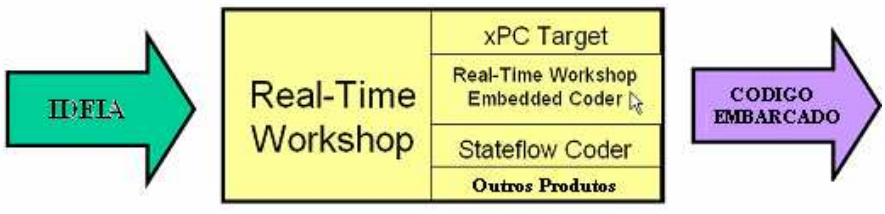


Figura 3.13: Multi-domínios de aplicação do MATLAB®.

A sua utilização pode ser feita através de comandos executados diretamente em interface amigável, permitindo rápida execução de cálculos e manipulação de dados. Algoritmos podem ser implementados em arquivos de texto utilizando-se a linguagem de programação MATLAB®. Para prototipagem de dispositivos mecatrônicos, esta linguagem dispõe de ferramentas para a Modelagem de Sistemas Físicos e Geração Automática de Códigos (figura 3.14).



a) Modelagem de sistemas físicos



b) Geração automática de códigos

Figura 3.14: Ferramentas para prototipagem de sistemas Mecatrônicos com MATLAB®.

3.3.3 Implementação do Sistema de Controle

A performance de um controlador exige que sua concepção esteja fortemente associada ao modelo físico da planta associada (figura 3.15), permitindo constantes atualizações de seu controlador (*hardware*/*software*) e aquisição de dispositivos externos a serem integrados no controlador do robô, nem sempre possíveis de serem realizadas pelo usuário.

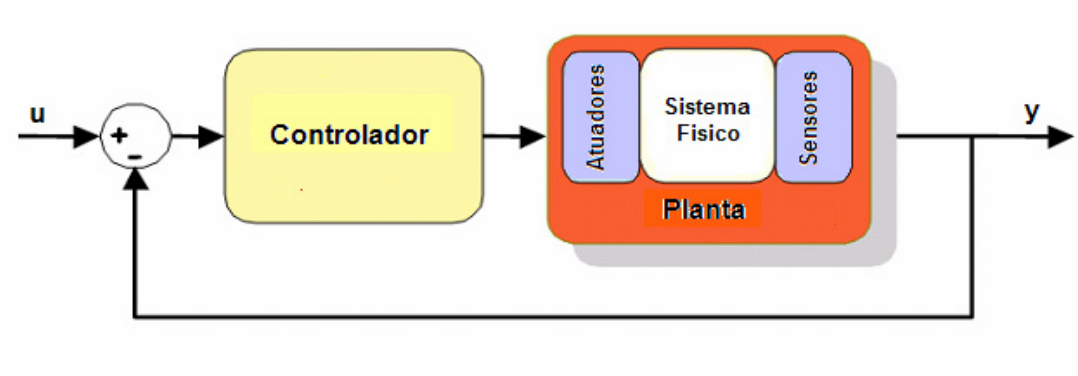


Figura 3.15: Sistema de controle baseado na modelagem de sistemas físicos.

A utilização de um ambiente único para cálculo científico, análise de dados e visualização, modelagem de sistemas e simulação, lógica embarcada em tempo real através da implementação de um sistema completo da fase de concepção até a eletrônica embarcada, torna-se imprescindível para o estudo e aprimoramento de técnicas de controle de sistemas mecatrônicos. Assim, dentro do conceito da prototipagem rápida, os seguintes aspectos deverão ser contemplados:

- Ambiente de prototipagem que represente um modelo muito próximo ao real, ocasionando um menor tempo de execução do mesmo com custos relativamente mais baixos.
- Desenvolvimento de algoritmos de controle que admitam complexidade e possam ser processados num tempo mínimo. Todas as tarefas requeridas precisam ser desempenhadas dentro de uma prototipagem padrão e requerem ferramentas inteligentes para auxiliar, modelem intuitivamente o problema e sejam de fácil utilização.

- Integração e concepção do sistema, implementação e validação de algoritmos de controle/comando,
- Arquitetura modular, através de blocos de funções de transferência, de modo a associar um componente aos outros e que sejam facilmente reconfigurados.

3.3.4 A utilização do Simulink® como ferramenta gráfica para a concepção

Simulink® é uma ferramenta gráfica integrada ao MATLAB®, que é um ambiente para simulação e criação de modelos com forte direcionamento a área de Controle e Automação. Este aplicativo permite a utilização de uma grande variedade de funções e modelos disponíveis sob a forma de diferentes bibliotecas. (figura 3.16)

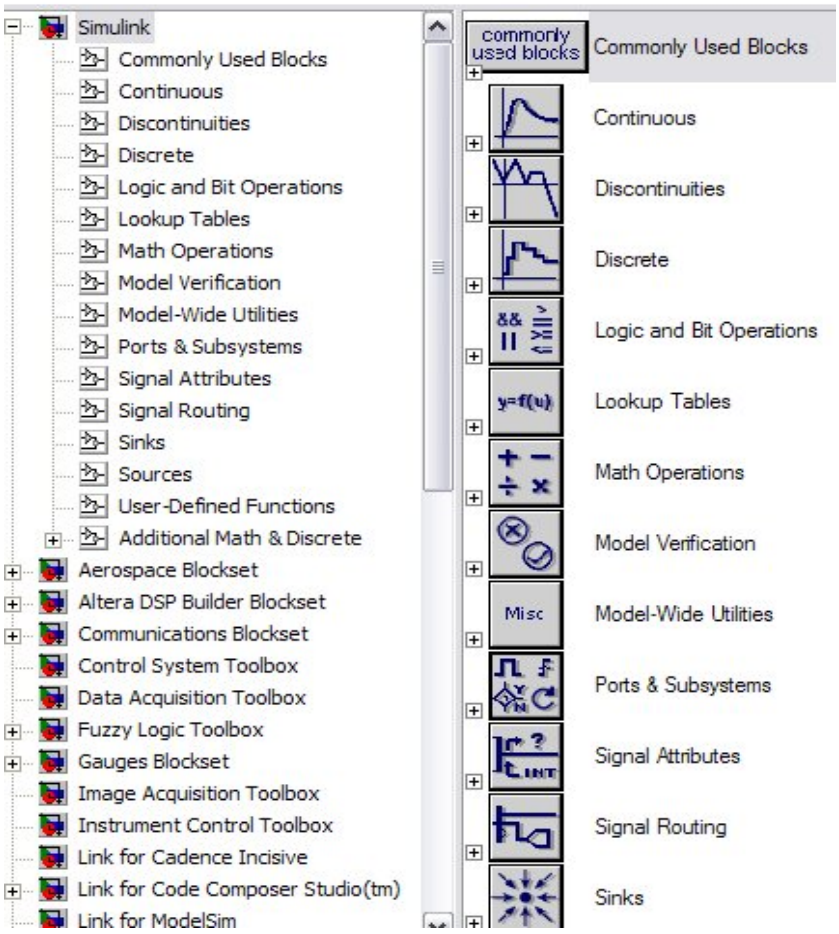


Figura 3.16: Bibliotecas Matlab/Simulink®.

3.4 DSP Builder

O DSP Builder é uma ferramenta de desenvolvimento de projetos que utiliza processamento digital de sinais, que permite a prototipagem através de interface entre o Matlab/Simulink® e o Quartus II.

Instalado como uma biblioteca do Matlab/Simulink®, o DSP Builder permite a integração de blocos do Matlab/Simulink® com blocos de implementação de lógica em FPGA, permitindo assim a elaboração e simulação de sistemas no ambiente MATLAB.

Estes modelos funcionais do Matlab/Simulink® (.mdl) são então convertidos em VHDL de modo a serem transformados em lógicas que elaborem suas funções e que possam ser implementadas em um circuito digital FPGA.

Outra opção interessante é a possibilidade de utilização destes blocos já implementados em uma FPGA para acelerar um processo de simulação do Matlab/Simulink®, ou de verificar o funcionamento de uma lógica utilizando o Matlab/Simulink® como ferramenta de geração e verificação de sinais. Este processo é chamado de HIL (*Hardware In the Loop*). Através de blocos em HIL o Matlab/Simulink® fornece pacotes de dados que são enviados ao componente via interface JTAG, sendo estes processados pelo hardware, os resultados gerados são lidos pelo MatLab também através da interface JTAG.

A figura 3.17 mostra um computador conectado a uma placa de desenvolvimento da ALTERA, esta conexão é feita via cabo USB-Blaster que interliga as interfaces USB do PC a interface JTAG da FPGA. Na tela temos a simulação HIL com Matlab/Simulink® e Quartus II.

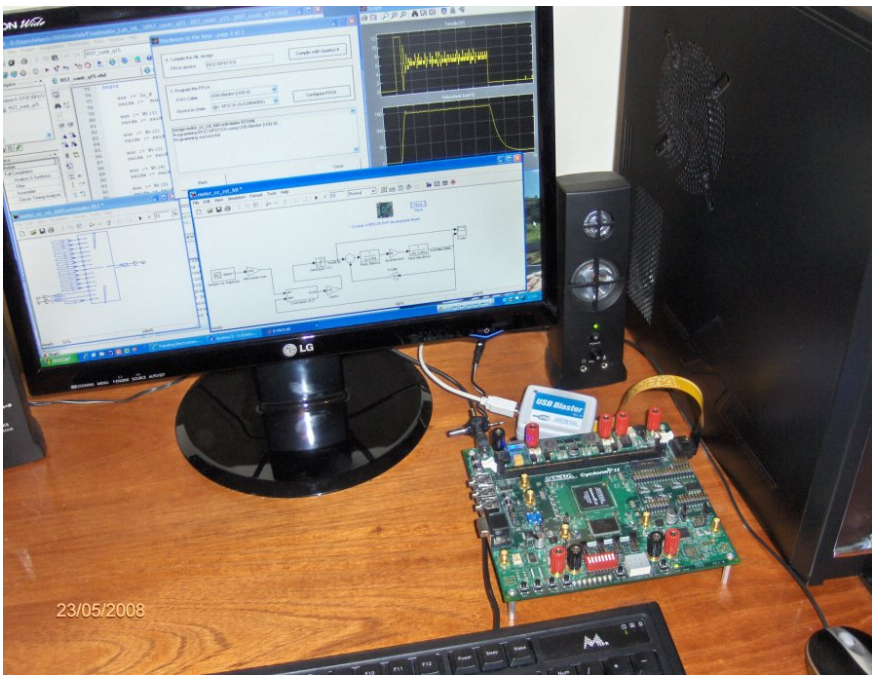


Figura 3.17: Simulação HIL do Matlab/Simulink®.

3.5 SOPC Builder

Com o aumento de capacidade dos componentes programáveis que além de possuírem uma grande quantidade de elementos lógicos também possuem grandes blocos de memórias, tais características possibilitam a implementação de processadores com memória e interfaces em um único componente.

Várias opções de implementação de processadores comerciais estão disponíveis no mercado ou disponibilizados por universidades ou por comunidades de desenvolvedores. Estes blocos chamados de IP-Cores são normalmente descritos e disponíveis em VHDL ou Verilog. Outra opção é fornecida pelos próprios fabricantes de FPGA, que fornecem uma opção de processador e de ferramentas de desenvolvimento incorporadas em seus softwares.

ALTERA possui um core de um processador RISC de 32 bits denominado Nios II, como também diversas interfaces normalmente necessárias para a implementação de um sistema microprocessado, como interfaces seriais, memórias, timers, portas de I/O (figura 3.18).

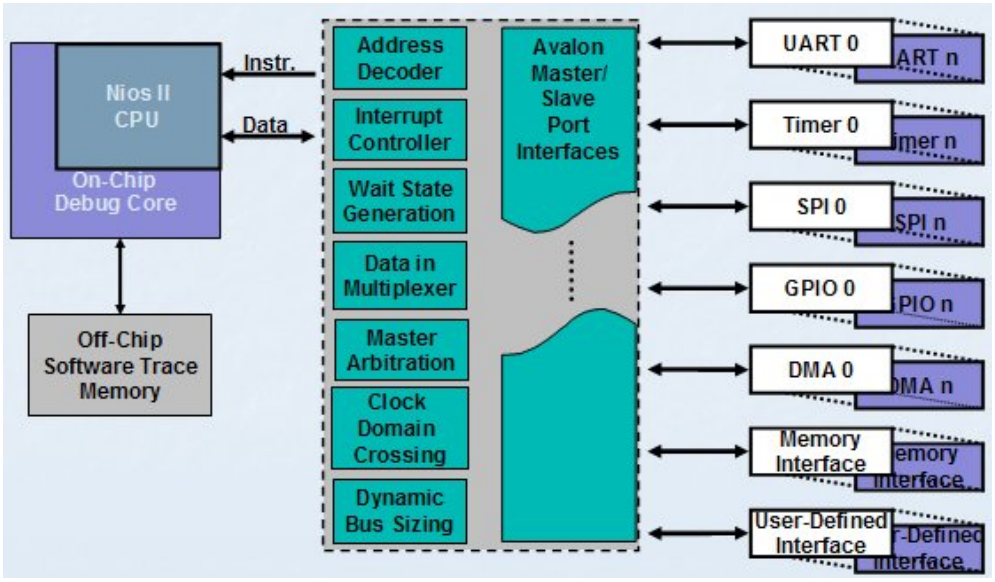


Figura 3.18: Arquitetura das interfaces do processador Nios II ALTERA.

A figura 3.18, ilustra o processador com as suas interfaces. Por ser um sistema implementado em um componente programável, a quantidade e os tipos de interfaces a serem implementadas estão limitadas apenas pelo tamanho do componente a ser utilizado. Tal característica permite uma grande flexibilidade implementação, pois a quantidade de interfaces não está fixa, como acontece quando se utiliza microcontroladores comerciais.

Para uma rápida implementação do sistema, uma ferramenta chamada SOPC Builder (*System On Programmable Chip*) permite uma rápida configuração do processador desejado, possibilitando a escolha e configuração das interfaces desejadas. É possível também a adição de novas interfaces projetadas pelo usuário, esta interface pode ser descrita em linguagem VHDL e incorporada na biblioteca do SOPC (figura 3.19).

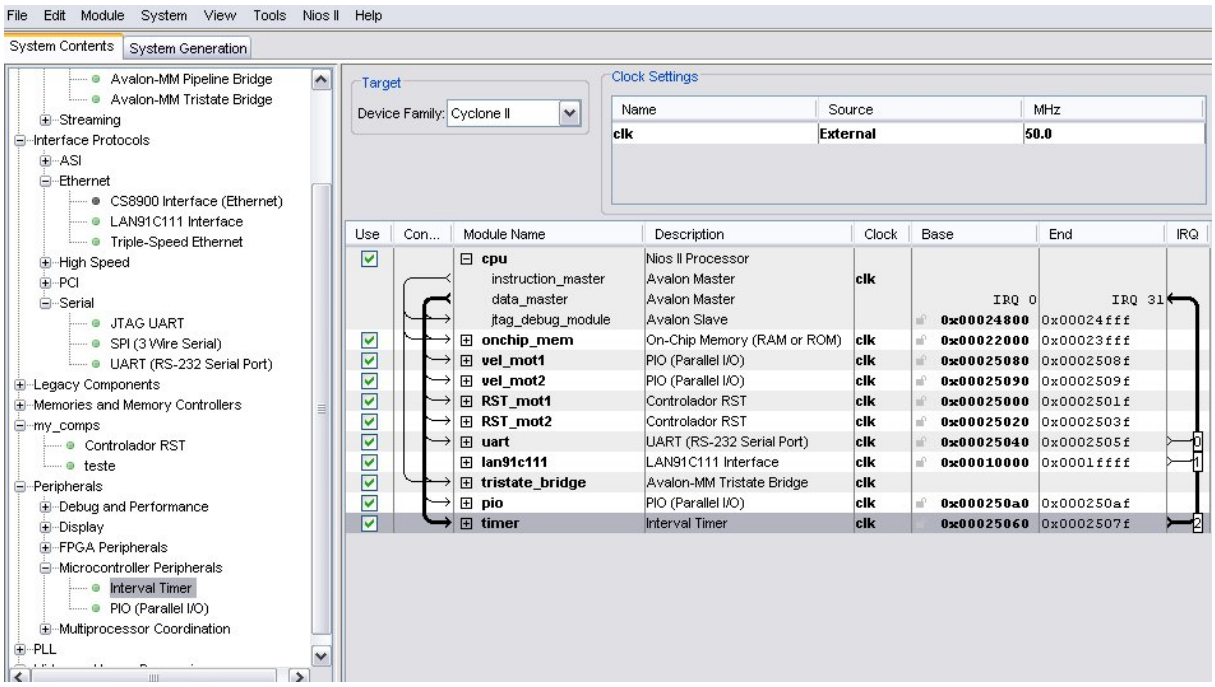


Figura 3.19: Tela de configuração do SOPC Builder.

Após a configuração do processador e suas interfaces, o SOPC executa a geração de arquivos de descrição do hardware do sistema em VHDL bem como arquivos de definições das interfaces para a programação em linguagem "C/C++" utilizando-se o Nios II IDE (Eclipse). Estes arquivos são então adicionados ao projeto da FPGA pelo Quartus II implementando todo o sistema na FPGA, conforme é apresentado na figura 3.20.

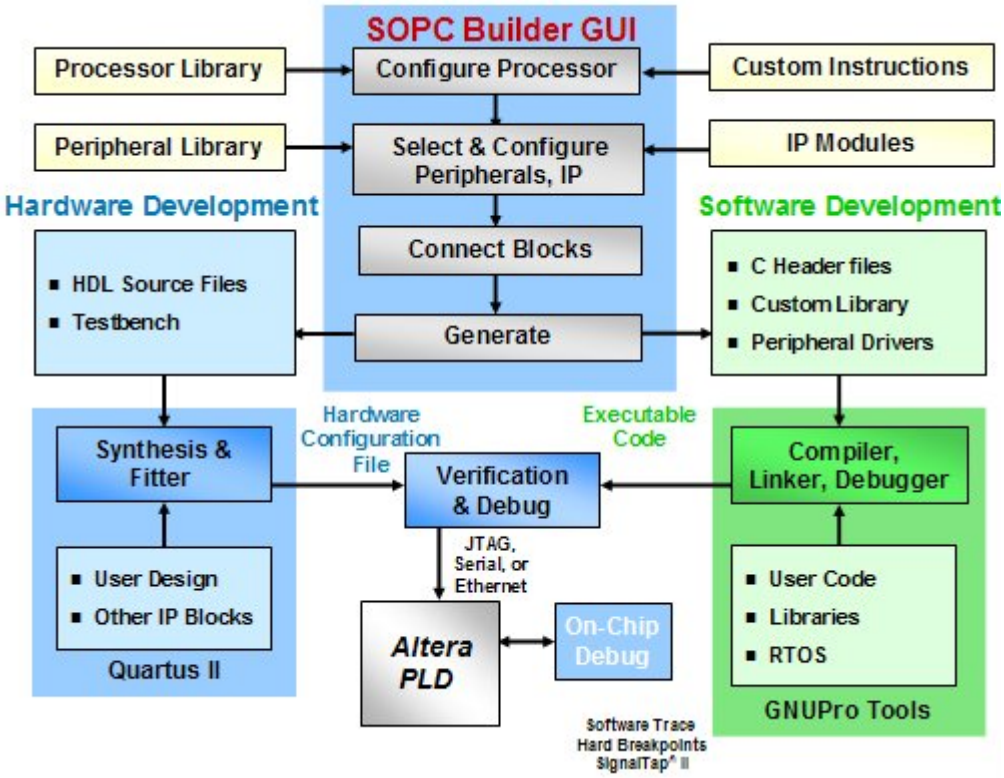


Figura 3.20: Fluxo de projeto do SOPC Builder.

3.6 Conclusões

Neste capitulo foram apresentados ambientes para prototipagem rápida e concepção de sistemas embarcados baseados nos aplicativos Matlab, Simulink® e ALTERA, onde foram mostrados propriedades destes sistemas, com ênfase nos aspectos de implementação de *hardware* e *software*. Com isto é permissível a prototipagem rápida de dispositivos mecatrônicos através das etapas de implementação, validação e testes de controladores em lógica embarcada. Desta forma, constituindo de referência para o próximo capítulo deste trabalho onde direcionado a modelagem de um sistema de acionamento e controle de uma junta robótica com controladores PID expressos na forma genérica RST para posterior implementação em lógica reconfigurável, através da utilização de linguagem gráfica e linguagem VHDL.

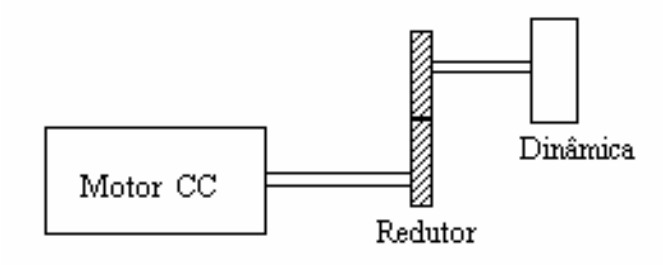
Capítulo 4

Modelagem de um sistema de acionamento e controle para juntas Robóticas

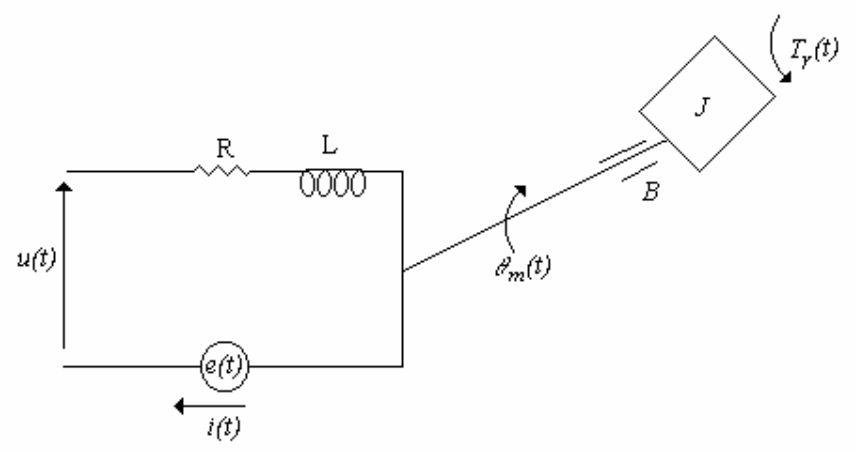
Neste capítulo será realizado um estudo detalhado à modelagem e controle de motores elétricos de corrente contínua, largamente utilizado em aplicações de robótica, permitindo assim a apresentação, no próximo capitulo desta dissertação de um exemplo prático de prototipagem rápida para o sistema de acionamento e controle de uma junta robótica disponível no Laboratório de Automação Integrada e Robótica da Faculdade de Engenharia Mecânica da UNICAMP.

4.1 Modelagem de Sistemas Mecatrônicos

Consideremos um motor de corrente continua (CC), uma transmissão mecânica (redutor) e a dinâmica do sistema. O redutor é utilizado para ampliar o torque, obtendo assim um melhor desempenho do motor. A dinâmica do sistema consiste no estudo das forças que condicionam o movimento. Para acelerar um manipulador do seu estado inercial até uma velocidade constante e promover uma desaceleração, devem ser aplicados um conjunto de equações dinâmicas nas juntas dos atuadores [Craig, 1986]. No estudo do motor CC serão desenvolvidas as equações referentes ao seu movimento. A representação de um motor CC controlador pela corrente da armadura é mostrada detalhadamente na figura 4.1, onde seus parâmetros são:



a) Modelo dos principais componentes de um sistema de acionamento.



b) Representação esquemática

Figura 4.1: Motor CC controlado pela corrente de armadura.

i(t) - corrente (A);

R - resistência induzida (Ω)

L - indutância (H);

e(t) – força contra-eletromotriz (V);

θ_m – deslocamento angular (rad);

u - tensão aplicada no circuito da armadura (V);

J_m - momento de inércia do motor (kg m^2);

K_e - constante da força contra-eletromotriz (V/rad s^{-1});

K_t- constante de torque ((Nm/A);

T_r - torque resistente (Nm)

T_m - torque mecânico (Nm)

B – fricção viscosa do motor (Kg m/rad/s)

4.2 Modelagem de um motor CC (Ogata, 1998)

4.2.1 Equação Elétrica

Em um motor CC controlado por armadura a velocidade é controlada pela tensão de armadura *u(t)* que é suprida por uma fonte de corrente contínua. A equação diferencial para o circuito de armadura é:

$$u(t) = Ri(t) + L\frac{di(t)}{dt} + e(t), \qquad (4.1)$$

onde $e(t)=K_e\Omega(t)$

Para um motor de corrente contínua (c.c.), $K_e \approx K_t$

4.2.2 Equação de Acoplamento Eletro-mecânico

No motor CC controlado por armadura a corrente de campo é mantida constante. Para uma corrente de campo constante, que resulta um fluxo magnético constante, o torque torna-se diretamente proporcional à corrente de armadura, de modo que:

$$T_m(t) = K_t i(t) \qquad (4.2)$$

onde K_t é a constante de torque do motor.

4.2.3 Equação Mecânica

A corrente de armadura produz o torque que é aplicado à inércia e à fricção, portanto,

$$T_m(t) = J_m\frac{d\Omega(t)}{dt} + B\Omega(t) + T_r(t), \text{ onde } \Omega(t) = \dot{\theta} \qquad (4.3)$$

4.3 Diagrama de Blocos Associados

A figura 4.2 apresenta os diagramas de blocos associados às equações apresentadas anteriormente, e a figura 4.3 o digrama de blocos relativo a equação completa (parte elétrica, mecânica e acoplamento).

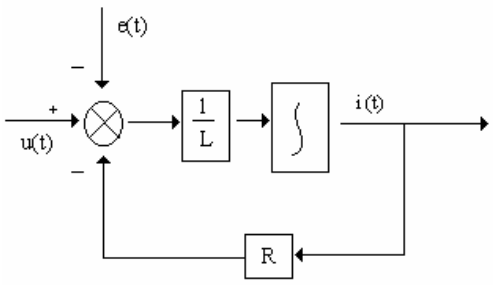
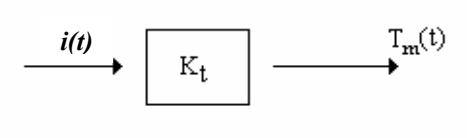
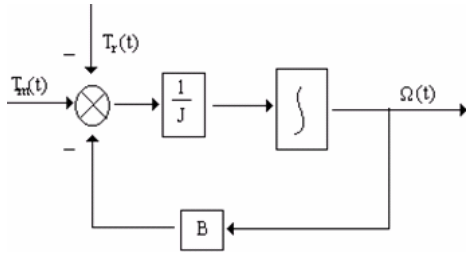
Equação Dinâmica	Diagrama de Blocos Associado
$\frac{di(t)}{dt} = \frac{1}{L}\left(u(t) - R.i(t) - e(t)\right)$	
$T_m(t) = K_t\, i(t)$	
$\frac{d\Omega(t)}{dt} = \frac{1}{J_m}.\left(T_m(t) - B_m\Omega(t) - T_r(t)\right)$	

Figura 4.2: Motor CC - Diagrama de blocos associados às equações do modelo.

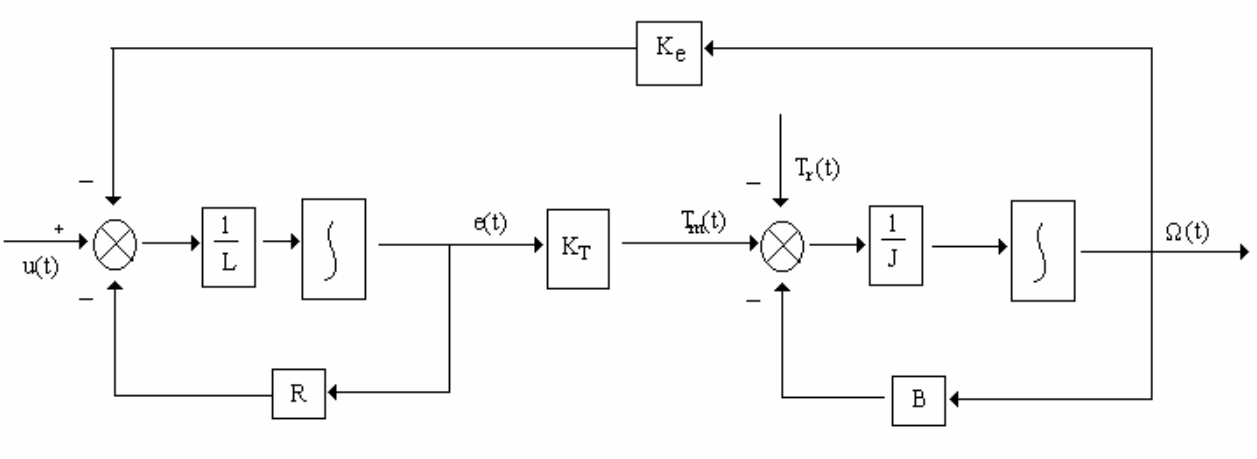


Figura 4.3: Diagrama de blocos para a equação completa.

4.4 Transformada de Laplace

Utilizando as transformadas de Laplace chegamos as Funções de Transferência associadas às equações de um motor CC e diagrama de blocos associado ao modelo completo do sistema (figura 4.4) , expressas por:

4.4.1 Parte Elétrica

$$U(t) = RI(s) + LsI(s) + E(s) \qquad (4.4)$$

$$U(t) = (R + Ls)I(s) + E(s) \qquad (4.5)$$

$$I(s) = \frac{1}{R + Ls}(U(t) - E(s)) = I(s) = H_1(s)(U(s) - E(s)) \qquad (4.6)$$

onde

$H_1(s) = \dfrac{1}{R + Ls}$;

$P_e = \dfrac{R}{L}$ (pólo elétrico);

$\tau_e = \dfrac{L}{R}$ (constante de tempo elétrica)

4.4.2 Parte Mecânica

$$T_m(s) = J_m s\Omega(s) + B\Omega(s) + T_r(s) \tag{4.7}$$

$$T_m(s) = \Omega(s)(J_m s + B) + T_r(s) \tag{4.8}$$

$$\Omega(s) = \frac{1}{J_m s + B}\left(T_m(s) - T_r(s)\right) = \Omega(s) = H_2(s)\left(T_m(s) - T_r(s)\right) \tag{4.9}$$

onde:

$$H_2(s) = \frac{1}{J_m s + B}$$

$P_m = \frac{B}{J_m}$ pólo mecânico

$\tau_m = \frac{J_m}{B}$ constante de tempo mecânica

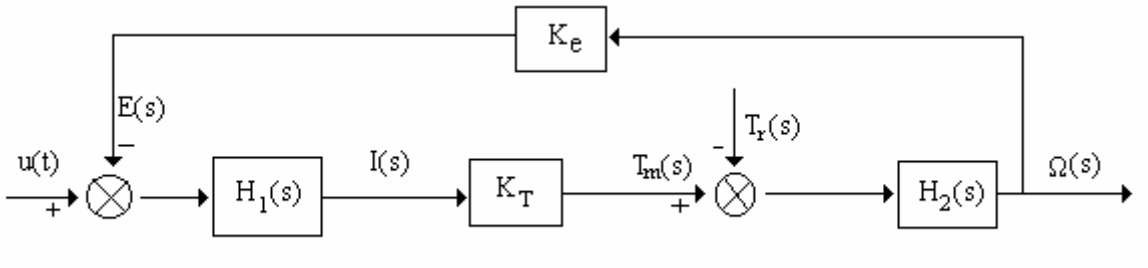
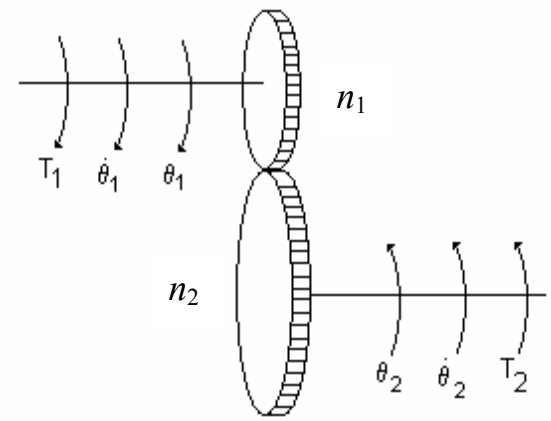


Figura 4.4: Diagrama de blocos para o modelo completo das funções de transferência

4.5 Transmissão Mecânica

Considerando que uma junta robótica é constituída por motores, redutores e acoplamento, cabe aqui ressaltar os redutores de velocidades. Os sistemas de engrenagens transmitem potência de um eixo para outro. As relações físicas que relaciona estes sistemas com outras grandezas, estão relacionadas por n que é a relação de engrenagens. A figura 4.5 apresenta as relações entre

as grandezas, velocidade, deslocamento e torque relacionados a um sistema de transmissão mecânica (engrenagens).



Relação	Equação Associada
Transmissão Mecânica (engrenagens)	$\frac{n_2}{n_1} = n$
Deslocamento	$\frac{\theta_2}{\theta_1} = \frac{n_1}{n_2} = \frac{1}{n}$
Velocidade	$\frac{\dot{\theta}_2}{\dot{\theta}_1} = \frac{n_1}{n_2} = \frac{1}{n}$
Torque	$\frac{T_2}{T_1} = \frac{n_2}{n_1} = n$

Figura 4.5: Sistema de engrenagens.

A figura 4.6 apresenta sistemas mecânicos equivalentes e suas respectivas equações relacionando as grandezas, inércia, rigidez e atrito viscoso acoplado por meio de sistemas de transmissão por engrenagens. A figura 4.7 apresenta o diagrama referente a uma carga acoplada no eixo de um redutor.

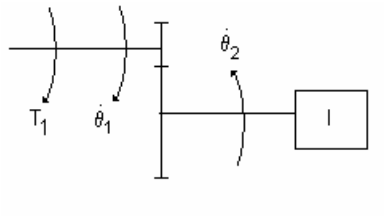
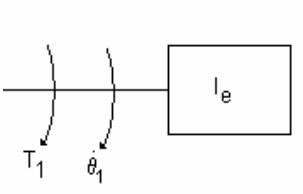
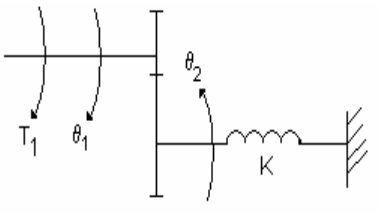
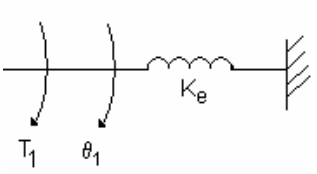
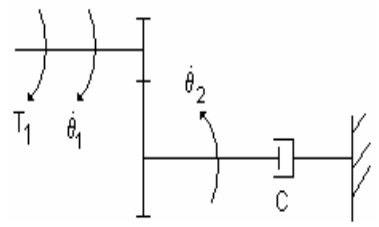
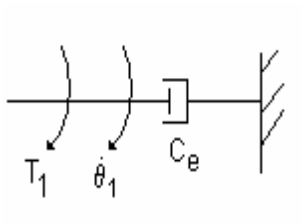
Sistema Original	Sistema Equivalente	Equação Matemática
Inércia+ sistema de engrenagens 		$I_e = I\left(\frac{\dot{\theta}_2}{\dot{\theta}_1}\right)^2 = \frac{1}{n^2} I$
Rigidez + sistema de engrenagens 		$K_e = K\left(\frac{\theta_2}{\theta_1}\right)^2 = \frac{1}{n^2} K$
Atrito viscoso + sistema de engrenagens. 		$B_e = B\left(\frac{\dot{\theta}_2}{\dot{\theta}_1}\right)^2 = \frac{1}{n^2} B$

Figura 4.6: Sistemas mecânicos equivalentes.

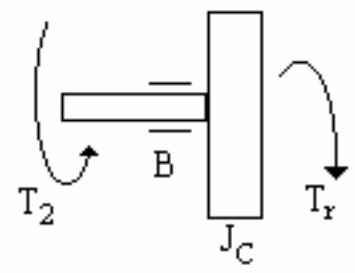


Figura 4.7: Representação do acoplamento de uma carga no eixo do redutor.

4.5.1 Redutor acoplado a uma carga

A equação que representará essa carga no redutor é dada por:

$$T_2 = J_C \ddot{\theta} + B_C \dot{\theta} + T_r \tag{4.10}$$

Aplicando a transformada de Laplace a função de transferência associada será representada pela equação 4.11 (figura 4.8):

$$\theta(s) = \frac{T_2(s) - T_r(s)}{J_C s^2 + B_C s} \tag{4.11}$$

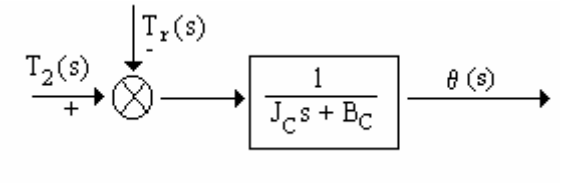


Figura 4.8: Diagrama de blocos associado.

4.5.2 Dinâmica do sistema

Considerando agora o efeito de carga, as equações dinâmicas do sistema são apresentadas na tabela da figura 4.9.

Relação	Modelo Matemático
Função Transferência Motor Elétrico	$(T_m(s) - T_r(s))H_2(s) = \Omega_{motor}(s)$
Função Transferência Carga	$(T_{carga}(s) - T_{pert.}(s))H_3(s) = \Omega_{carga}(s)$
Razão de Transmissão (Velocidade)	$\Omega_{carga}(s) = \frac{1}{\eta}\Omega_{motor}(s)$
Razão de Transmissão (Torque)	$T_{carga}(s) = \eta T_{motor}(s)$

η: redutor

Figura 4.9: Relações Matemáticas – Motor acoplado a uma carga.

Através da manipulação dessas relações matemáticas (figura 4.9), para melhor compreensão, consideraremos $T_r(s), T_{pert}(s) \approx 0$ e assim, obtemos as equações 4.12 e 4.13, representadas na figura 4.10:

$$T_m(s)H_2(s) = \Omega_{motor}(s) \quad (4.12)$$

$$\eta^2 T_m(s)H_3(s) = \Omega_{motor}(s) \quad (4.13)$$

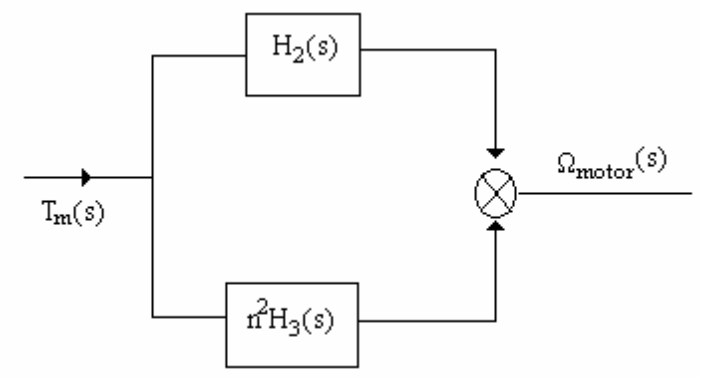


Figura 4.10: Diagrama de blocos para a equação do sistema.

onde:

$$H_2(s) = \frac{1}{J_m(s) + B}$$

$$H_3(s) = \frac{1}{J_C(s) + B_C}$$

Através destas equações, obtemos:

$$\Omega_{motor}(s) = (H_2(s) + \eta^2 H_3(s))T_m(s) \tag{4.13}$$

ou ainda

$$\Omega_{motor}(s) = (J_m + \eta^2 J_C) + (B_m + \eta^2 B_C)T_m(s) \tag{4.14}$$

4.6 Controlador PID baseado no modelo dinâmico

O controlador é o elemento fundamental de um sistema completo de controle, cabendo ao mesmo a tarefa de controle global num estado/posição pré-determinado. O tipo de controlador mais utilizado em dispositivos robóticos é o PID (Proporcional/Integral/Derivativo). Este tipo de controlador possui um bom desempenho desde que o sistema a controlar seja conhecido, bem comportado (linear e invariante no tempo) e os parâmetros do controlador bem ajustados. A sua principal limitação, derivada do procedimento de ajuste, é ser sensível às diferentes condições de funcionamento. Nesta situação, a utilização de um controlador com ganhos variáveis é vantajoso.

Para podermos comparar o controlador proposto neste trabalho (não linear utilizando lógica reprogramável), foi realizado durante esta etapa um estudo inicial de um controlador PID com ganhos variáveis (é realizada uma tabela de ganhos em função de suas principais configurações inerciais). Isto se deve ao fato de que num manipulador robótico, durante a execução de uma

determinada trajetória, apresentará uma grande variação de seus parâmetros dinâmicos (termos inerciais, coeficientes de atrito, efeitos gravitacionais), acarretando na mudança dos parâmetros do controlador projetado em função de sua configuração inercial.

4.6.1 Projeto de um Controlador de posição PID

Neste tópico apresentaremos aspectos concernentes a modelagem de sistemas dinâmicos multi-variáveis acoplados, típicos da área de robótica, onde, através de simulações, via SIMULINK®, poderemos analisar o comportamento do sistema em questão.

O procedimento para cálculo dos ganhos do regulador PID, será realizado em cada junta, para diferentes configurações inerciais, considerando-se o efeito da inércia equivalente no eixo de rotação do motor (cálculo do raio de giração e massa equivalente). As constantes serão descritas como se segue:

$J = J_m + mk^2$

B (4.15)

$T_r = mgk\ \text{sen}\theta$

onde k é o raio de giração

Para o cálculo dos ganhos do regulador PID, o sistema será linearizado para as diferentes configurações inerciais para pequenos ângulos ($\text{sen}\theta \approx \theta$). A partir da transformada de Laplace:

$$\tau(s) = (Js^2 + Bs + T_t)\,\theta(s) \tag{4.16}$$

A função de transferência associada será:

$$T(s) = \frac{\theta(s)}{\tau(s)} = \frac{1}{Js^2 + Bs + T_t} \tag{4.17}$$

Através do critério de estabilidade de Routh-Hurwitz (Ogata,1995), podemos verificar a estabilidade do sistema e, aplicando o método de Ziegler-Nichols (Ogata,1995), vamos encontrar os valores utilizados no controlador PID. Estas simulações nos darão as bases necessárias para que possamos comparar os resultados obtidos com o do controlador proposto.

4.6.2 Determinação dos ganhos de um controlador PID

Após verificar a estabilidade do sistema, vamos calcular os ganhos do regulador PID. Neste trabalho, os ganhos do regulador PID para diferentes configurações inerciais serão calculados e validados a partir de simulação computacional, utilizando métodos baseados na otimização do comportamento da resposta do sistema, tradicionalmente utilizados em aplicações industriais (figura 4.11).

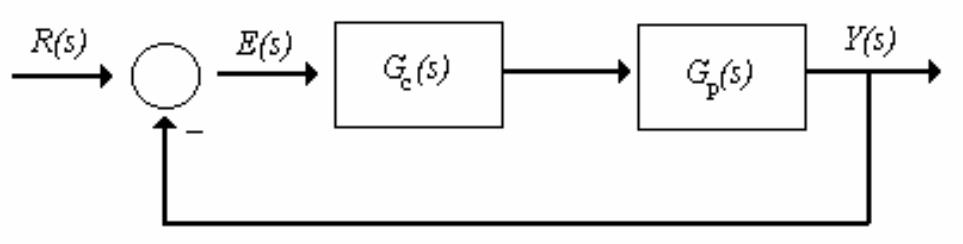


Figura 4.11: Esquema de um sistema para o cálculo do ganho do PID.

4.6.3 Método de Ziegler-Nichols

A figura 4.11 mostra o diagrama de blocos de um sistema do qual se deseja calcular os ganhos do regulador PID, onde $G_c(s)$ é equação do regulador PID e $G_p(s)$ é a equação do sistema.

A equação de um controlador PID é dada por:

$$G_c(s) = K_p + \frac{K_i}{s} + K_d s \qquad (4.18)$$

onde:

K_p é o ganho proporcional

K_i é o ganho integral

K_d é o ganho derivativo

O método de Ziegler-Nichols para o cálculo dos ganhos do controlador PID é baseado no cálculo de energia mínima da resposta do sistema, consistindo na execução dos seguintes procedimentos:

1. Um compensador proporcional é aplicado:

$$G_c(s) = K_p \tag{4.19}$$

O ganho deverá ser ajustado até o sistema tornar-se marginalmente estável. Este valor de ganho é denominado K_{po}, e o período de oscilação T_o.

2. Assim, o ganho do compensador será definido por:

$$G_c(s) = K_p\left(1 + \frac{1}{T_1 s} + T_d s\right) \tag{4.20}$$

onde:

Ganho do termo integrador: $K_i = \frac{K_p}{T_i}$

Ganho do termo Derivativo: $K_d = K_p T_d$

Os valores utilizados para o cálculo dos ganhos do controlador PID utilizando o método de Ziegler-Nichols são dados apresentados na figura 4.12.

	Ganhos do Controlador		
Tipo de Controlador	**K_p**	**T_i**	**T_d**
P	$K_p = 0{,}5K_{po}$	∞	**0**
PI	$K_p = 0{,}45K_{po}$	$T_i = 0{,}83T_o$	**0**
PID	$K_p = 0{,}6K_{po}$	$T_i = 0{,}5T_o$	$T_d = 0{,}125T_o$

Figura 4.12: Ganhos de um controlador PID.

Neste trabalho de pesquisa utilizaremos o modelo dinâmico representativo de uma junta robótica (motor acoplado a uma carga). A metodologia de cálculo dos parâmetros do regulador será realizada utilizando a mesma metodologia apresentada anteriormente, mas com o ajuste do ganho K_p realizado através do critério de Routh-Hurwitz (a linha do polinômio s^2 é forçada a zero, conseqüentemente o sistema é marginalmente estável).

4.6.4 Método de Chien-Hrones-Reswick (CHR)

Este método é bastante utilizado na indústria, no caso de não conhecimento do modelo dinâmico de um sistema. Esse método consiste no estudo da resposta do sistema em malha aberta submetido a uma entrada degrau (na prática um off-set) e a obtenção da respectiva curva de resposta temporal, como é mostrado na figura 4.13.

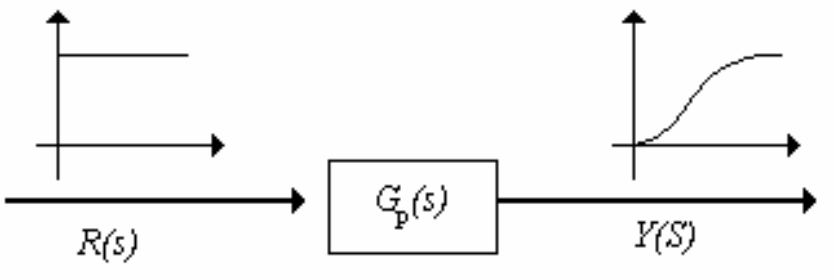


Figura 4.13: Método de CHR - obtenção da resposta temporal a uma entrada degrau.

Conseqüentemente, os ganhos do controlador PID poderão ser facilmente obtidos a partir da curva resposta temporal do sistema submetido a uma entrada degrau (figura 4.14).

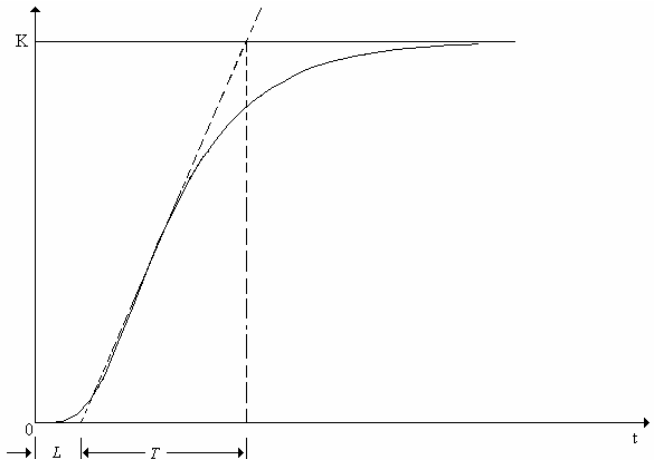


Figura 4.14: Curva resposta temporal utilizada no método CHR

Os seguintes procedimentos deverão ser utilizados:

1. Com o auxilio da curva resposta teremos os valores de T e L, obtidos a partir do cruzamento da reta tangente com a interseção no eixo do tempo, conforme mostra a figura 4.14.
2. Os valores para a escolha dos ganhos do controlador PID resultam do cálculo da razão R, obtida através da relação 4.21, onde a escolha do melhor controlador é obtida através da tabela mostrada na figura 4.15.

$$R = \frac{T}{L} \tag{4.21}$$

Tipo de controlador	R
P	R>10
PI	7,5<R<10
PID	3<R<7,5

Figura 4.15: Escolha do tipo de Controlador utilizando o método de CHR.

3. A partir da escolha do controlador, a tabela apresentada na figura 4.16 deverá ser utilizada para o cálculo dos ganhos dos diferentes tipos de controlador, onde $K_p=G_p(0)$.

	Ganhos do Controlador		
Tipo de Controlador	**K_p**	**T_i**	**T_d**
P	$\frac{T}{L}$	∞	0
PI	$0{,}9\frac{T}{L}$	$\frac{L}{0{,}3}$	0
PID	$1{,}2\frac{T}{L}$	2L	0,5L

Figura 4.16: Cálculo dos Ganhos do Controlador utilizando o método de CHR.

4. 7 Conclusões

Nesse capítulo foram apresentados os elementos que compõem uma junta robótica, tais como: motor de corrente contínua, inércia, redutores e acoplamento, sendo apresentado também procedimento para projeto de um controlador PID com ganhos variáveis devido as suas diferentes configurações inerciais.

Este estudo nos possibilitou obter a modelagem dinâmica de uma junta robótica, necessário para o estudo do modelo dinâmico de um manipulador robótico, onde a implementação do controlador de posição utilizando prototipagem rápida será realizada no próximo capitulo desta dissertação, utilizando ferramentas de prototipagem apresentadas nos capítulos 2 e 3 desse trabalho.

Capítulo 5

Validação e implementação experimental utilizando o conceito de *Hardware In-the-Loop* (HIL)

Este capítulo apresenta a aplicação experimental dos conceitos de prototipagem rápida apresentadas nos capítulos anteriores desse trabalho utilizando uma maquete experimental representando uma junta robótica construída no Laboratório de Automação Integrada e Robótica da Unicamp.

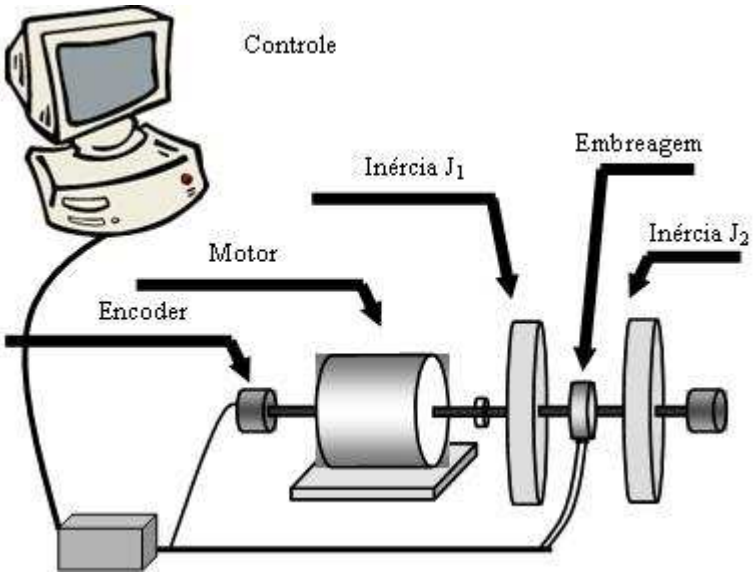
Esta bancada para validação e testes de juntas robóticas foi implementada com o propósito de simular um grau de liberdade de um robô, com a possibilidade de introduzir efeitos de mudanças inerciais ao longo de uma trajetória. Desta forma, é avaliado o desempenho de uma dada técnica de controle, sujeita a perturbações (acoplamento de outras juntas).

5.1 Descrição da Bancada Experimental

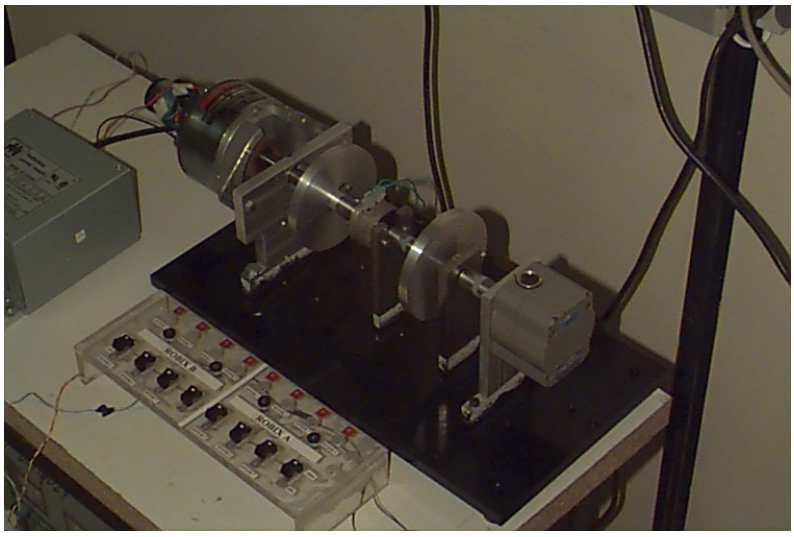
Uma bancada experimental foi implementada utilizando componentes cujos parâmetros fossem conhecidos. E com o propósito de simular um grau de liberdade de um robô, com a possibilidade de introduzir efeitos de mudanças inerciais ao longo de uma trajetória. Desta forma, avalia-se o desempenho de uma dada técnica de controle, sujeita a perturbações.

A figura 5.1 apresenta um esquema representativo desta bancada que é composta de um motor de corrente contínua, dois discos de inércia e sensores incrementais de posição. Para o

acoplamento da segunda inércia é utilizada uma embreagem eletromagnética, possibilitando assim a entrada e saída desta segunda carga, simulando a variação de inércia dentro de uma trajetória realizada por um robô.



a) Desenho representativo.



b) Implementação final.

Figura 5.1: Bancada experimental.

No momento em que a embreagem é acionada, surge um torque de ligação que depende da segunda carga. A velocidade inicial desta carga induz uma perturbação sobre o sistema. Após certo tempo a embreagem não desliza mais e a velocidade da segunda inércia é, então, igual a do motor. Podemos considerar este torque como sendo uma perturbação relativa à mudança de parâmetros do sistema.

Considerando nula a elasticidade do sistema, fisicamente o acoplamento impõe a conservação da quantidade de movimento de todo o sistema, com perda de energia cinética, ligada ao deslizamento inicial da embreagem. Entretanto, no momento em que a embreagem é acionada, não há modificação da inércia do motor, não produzindo assim perturbações sobre o sistema e nem mudança de velocidade ou perda de energia.

O torque fornecido pela embreagem deve ser suficientemente forte para evitar o deslizamento durante o acoplamento. Mesmo assim, esta ação introduz uma não linearidade sobre o sistema. Para evitar este deslizamento é necessário verificar:

$$T_{emb} \geq T_{mot} \frac{J_2}{J_{sistema}} \tag{5.1}$$

onde:

T_{emb} torque aplicado pela embreagem,

T_{mot} torque do motor,

J_2 inércia da segunda carga,

$J_{sistema}$ inércia do sistema (motor + carga 1).

Em resumo, a bancada proposta, aproxima-se muito a um grau de liberdade de um robô, com uma pequena diferença dada pelo fato da segunda carga entrar no sistema quase de forma instantânea. No caso de um robô essa variação é feita de forma continua no tempo. A embreagem é acionada através de comando elétrico e uma vez acionada, acopla a segunda carga ao sistema.

Com essa variação da carga temos uma perturbação ao sistema, o que nos permitirá verificar o comportamento do controlador devido a essa perturbação. Sob estes aspectos, esta bancada se apresenta, de uma forma geral, como uma estrutura interessante para validação, não somente de controladores preditivos como de outros algoritmos de controle.

5.2 Elementos Constituintes da Bancada Experimental

A bancada experimental implementada é composta dos elementos descritos abaixo, cujos parâmetros utilizados durante o processo de prototipagem rápida são descritos na tabela 5.1: [Eletrocraft, 1977]

Sistema de Acionamento: Motor elétrico de corrente contínua: M1040 da Eletrocraft^{TM} .

Sistema de Transmissão com inércia variável: Com o objetivo de se aproximar o modelo utilizado na simulação com o modelo de uma junta robótica, a bancada experimental simula um eixo do motor acoplado a uma inércia que poderá variar o seu valor através do acoplamento/desacoplamento de uma embreagem eletromagnética.

Transdutores de Posição: Os transdutores utilizados para medida de posição e velocidade são do tipo *encoders* incrementais da Hohner com 1024 pulsos por volta, onde um deles se encontra acoplado diretamente no eixo do motor e outro no final do eixo das inércias . Como os sinais oriundos dos transdutores são pulsos digitais, foi necessário desenvolver uma interface (*hardware*) e um programa computacional (*software*) para tratamento dessas informações (pulsos digitais relativos a um deslocamento angular realizado).

Tabela 5.1: Parâmetros do sistema de acionamento e controle da plataforma experimental.

Sigla	Descrição	Valor	Unidade
Jm	Inércia Motor	4,802 e-6	Kg m^2
Bm	Atrito viscoso	6,74 e-5	Nm/(rad/s)
τm	Tempo mecânico	2,3 e-3	s
Kt	Constante de acoplamento	4,090 e-2	Nm/A
Ke	Força contra eletromotriz	4,11 e-2	V.(rad/s)
Ra	Resistência da Armadura	0,85	Ω
La	Indutância	< 0,1	mH
J1	Inércia da carga 1	1,09 e-3	Kg m^2

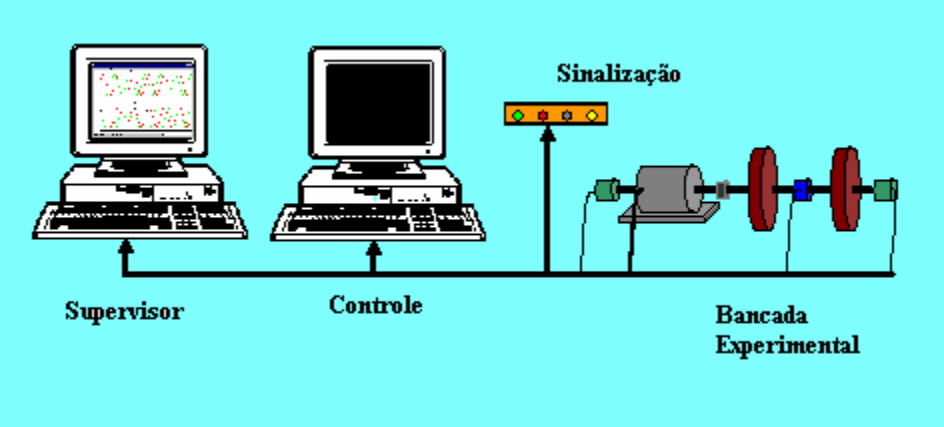


Figura 5.2 – Sistema de supervisão e controle.

5.3 Sistema de Monitoramento e Controle

Para aquisição de sinais e controle do sistema foi implementado a plataforma experimental apresentada na figura 5.2, onde para tratamento de informações dos *encoders* e *hardware* de aquisição foram utilizados dispositivos lógicos reprogramáveis da ALTERA™ e para tratamento das informações através do MATLAB-SIMULINK®.

A infra-estrutura aberta implementada possibilitou o desenvolvimento de algoritmos de controle usando bibliotecas próprias e até mesmo o desenvolvimento de programas e inclusão na

malha de controle do sistema. Neste aspecto existe a necessidade de tornar todos os sinais compatíveis.

O sistema é constituído de dois microcomputadores (figura 5.3), um responsável pela aquisição das informações provenientes dos transdutores de posição da bancada experimental e, o outro, responsável pela geração de sinais de referência (ou trajetórias) para o controlador. Isto permitirá uma análise do erro de comportamento da trajetória de referência depois de comparada com a posição final da carga medida em relação ao eixo do motor e também em relação à carga. O projeto foi implementado através do sistema de desenvolvimento Altera Quartus II, através de uma interface gráfica e de linguagem VHDL.

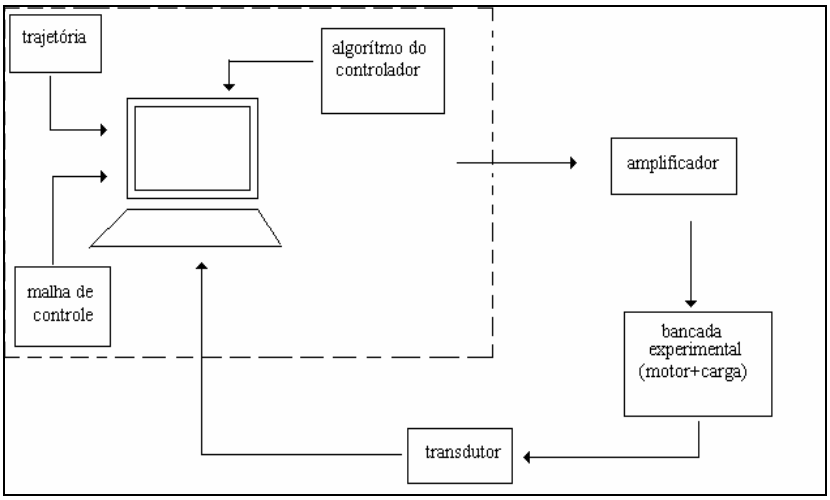


Figura 5.3: Esquema representativo do Sistema de aquisição e controle.

A geração de uma trajetória será realizada através de um programa computacional para o envio de um arquivo de pontos para o controlador de posição de uma determinada junta em *hardware* dedicado, desenvolvido em lógica reprogramável, utilizando o sistema ALTERA^{TM}.

5.4 Implementação experimental do leitor de posição e velocidade

Umas das maneiras de se medir a velocidade de um motor é utilizando-se um dispositivo que produz uma determinada quantidade de pulsos a cada rotação do motor. Na bancada de testes o dispositivo utilizado foi um *encoder* de posição da marca Hohner, que possui um conector com cinco sinais, correspondente a duas entradas para alimentação e três saídas de pulsos (A,B Z).

Os sinais A e B fornecem uma onda quadrada com uma diferença de fase de 90º conforme o sentido de rotação do eixo e com 1024 pulsos por rotação. O sinal Z fornece um pulso por rotação. O período do sinal de saída é diretamente proporcional à velocidade do motor.

O bloco de medição implementado com o sistema ALTERA, consiste de um contador de pulsos que é habilitado durante um determinado período de tempo. Durante este período os pulsos gerados pelo *encoder* são contados. No final deste tempo, o valor total da contagem é armazenado em um registrador e uma nova contagem é iniciada. Na figura 5.4 temos o bloco implementado na FPGA, com o gerador de janela de tempo de 32 bits, um contador de pulsos de 12 bits e o registro de saída de 12 bits.

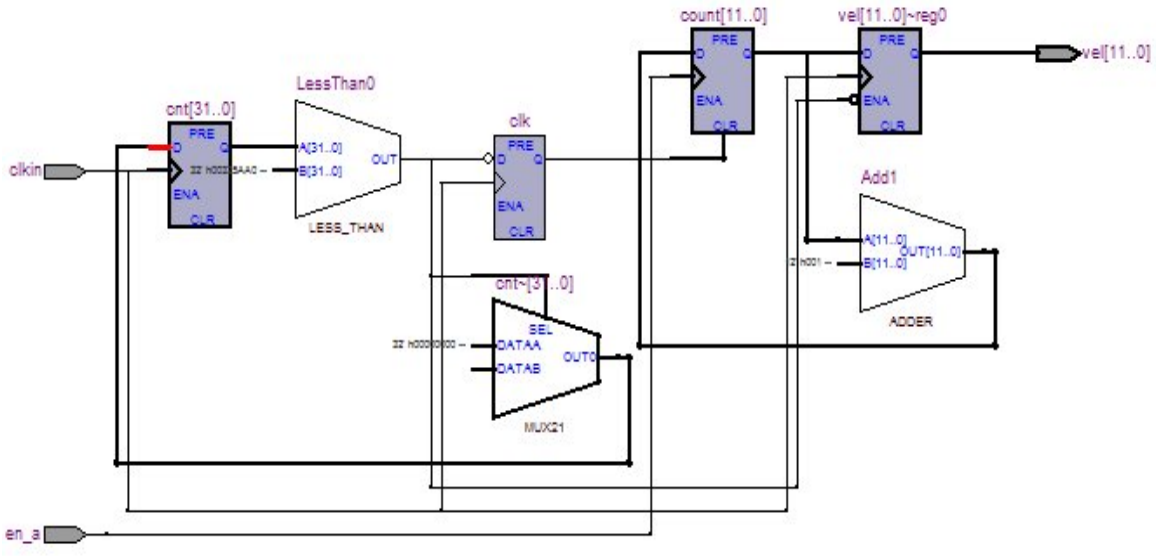


Figura 5.4: Sistema de medição de posição e velocidade.

Para compatibilizar os níveis de tensão dos sinais de saída do *encoder* de posição com os níveis permitidos com a FPGA, pode-se utilizar um foto acoplador conforme figura 5.5. Outro circuito integrado com característica de histerese na entrada (*schmidt-trigger*) é usado para garantir a retirada de ruídos.

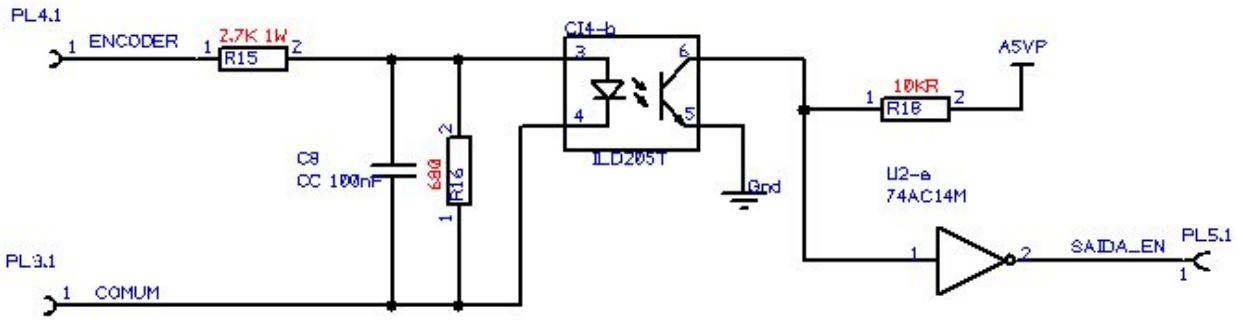


Figura 5.5: Circuito interface com encoder.

5.5 Interface com o motor

Para o acionamento do motor, foi utilizado um bloco gerador de sinal com modulação por largura de pulso ou PWM (*Pulse-Width Modulation*), este tipo de modulação consiste em variar o *duty-cycle* de um sinal sem alterar a sua freqüência. Este sinal digital, é aplicado a uma etapa de potência com transistores tipo MOS-FET, que executam o chaveamento do sinal de alimentação do Motor.

O bloco gerador de PWM foi implementado a partir da utilização de um contador e um comparador. O contador determina o período total do sinal e o comparador determina o *duty-cycle* do sinal comparando a saída do contador com um valor determinado, as formas de onda do circuito podem ser verificadas na figura 5.6 e o circuito de saída de potência na figura 5.7.

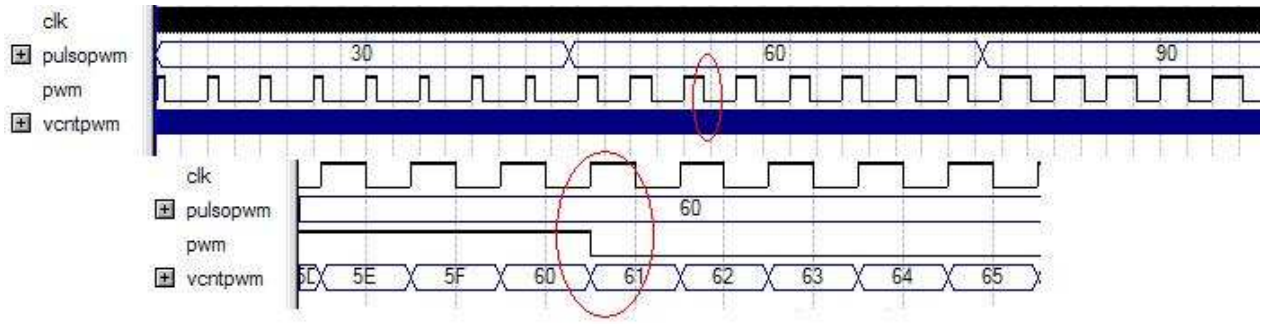


Figura 5.6: Geração da saída PWM.

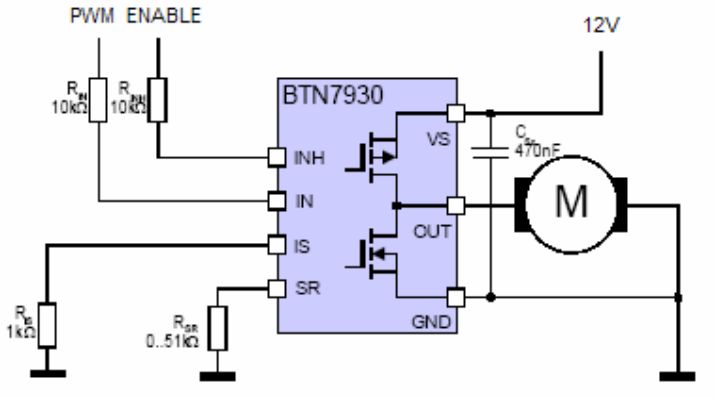


Figura 5.7: Circuito de potência para acionamento do Motor CC.

5.6 Resultados obtidos na implementação experimental do modelo

5.6.1 Análise da resposta do sistema submetido a uma entrada degrau

A figura 5.8, corresponde à simulação em SIMULINK® do modelo de uma junta robótica implementado na bancada experimental descrita anteriormente. Neste modelo temos um sinal de referência para a junta (Gerador de trajetória), um bloco correspondente a interface de potência (fonte de alimentação do motor) e os blocos correspondentes ao modelo do motor de CC acoplado a uma carga (parte elétrica e mecânica).

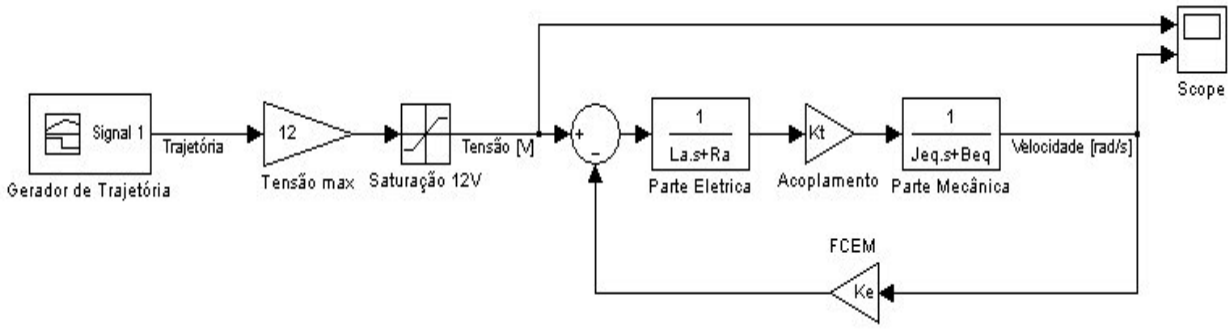


Figura 5.8: Modelo motor CC.

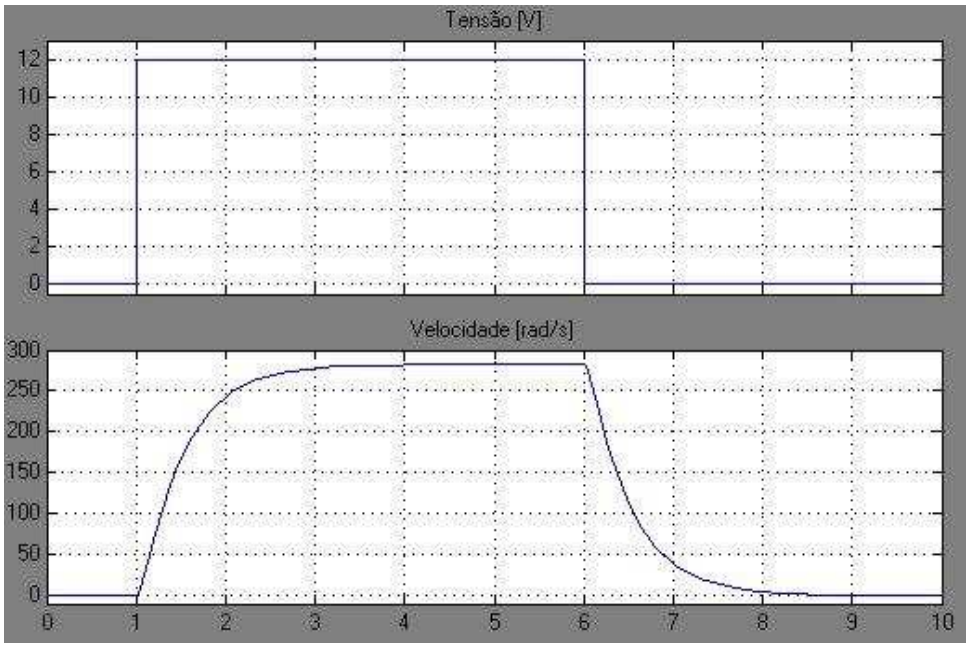
Para efeitos de validação do modelo serão comparados os sinais correspondentes à saída obtida através de simulação com a medida realizada através de osciloscópio digital na bancada experimental, como foi mostrado anteriormente na figura 5.8. Para o teste foi aplicado um sinal degrau de 12V de referência ao motor e verificada a curva de velocidade. A figura 5.9 (a e b) apresenta os resultados obtidos a partir da simulação em SIMULINK® do sistema em estudo e na bancada experimental implementada no LAR-UNICAMP, podendo-se observar o tempo de aproximadamente de 2s para estabilização da velocidade tanto na partida como na parada.

5.6.2 Controlador PID

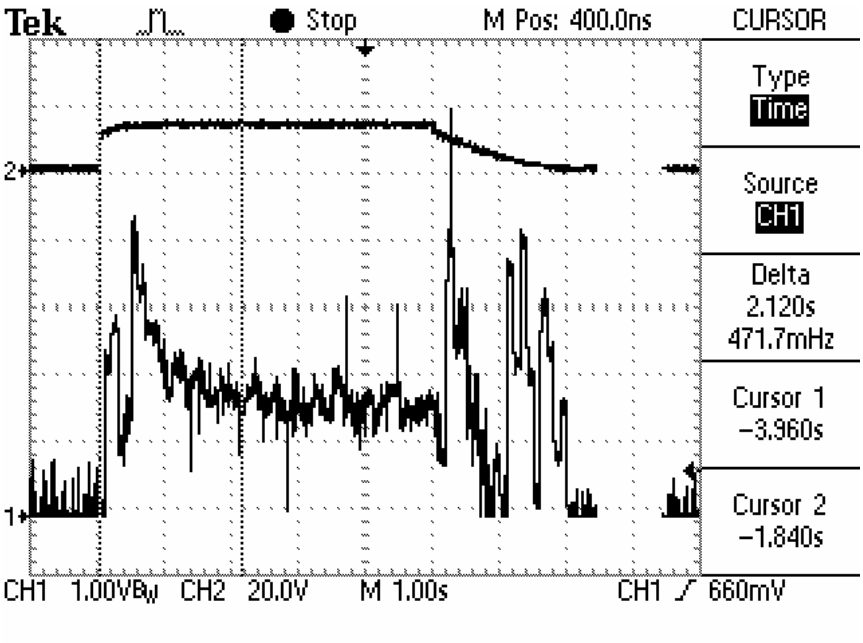
Na figura 5.10 são apresentados o diagrama de blocos do sistema motor com um bloco de controle PID e resultados obtidos na simulação (resposta de saída de velocidade).

5.6.3 Controlador PID na forma RST

Para a implementação de controladores em tempo real, torna-se necessário a utilização de controladores discretos no tempo. No projeto de um controlador através de alocação de pólos, utilizado no projeto de um controlador discreto, utiliza-se uma lei de controle linear, designada forma RST. No anexo A desta dissertação é apresentado um desenvolvimento teórico correspondente a um controlador PID na forma RST.

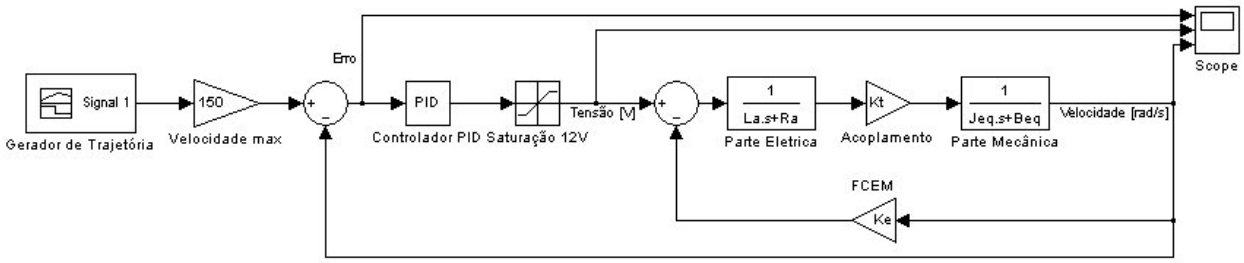


a) Resposta Simulada motor CC com carga.

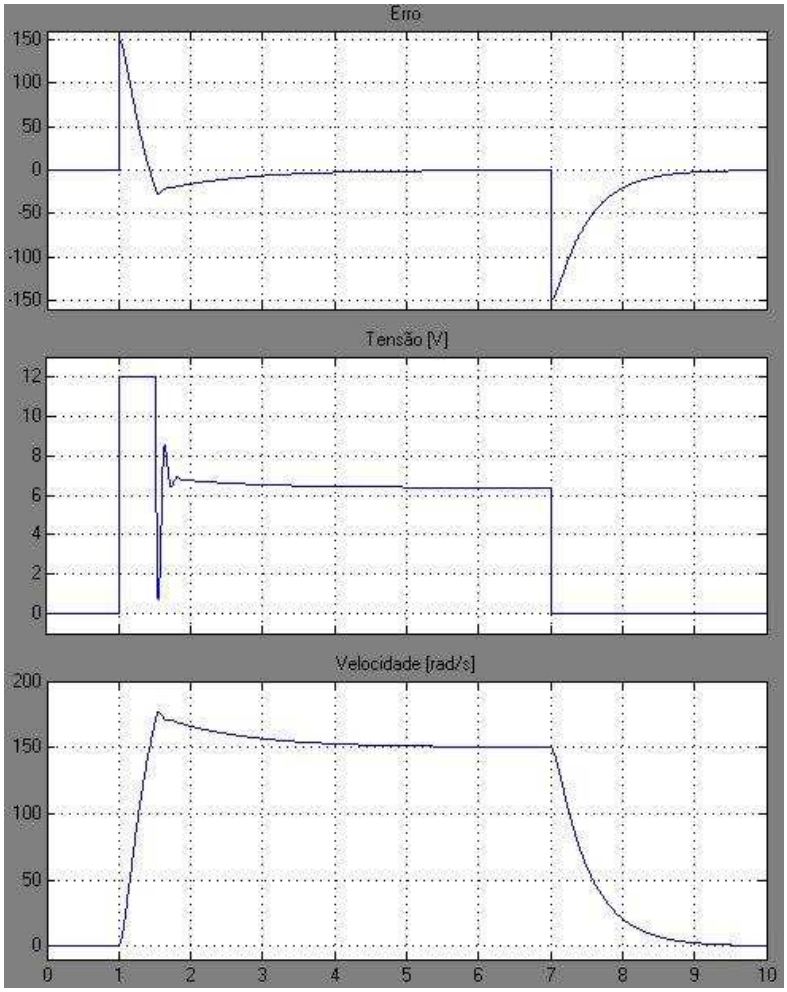


b) Resposta medida motor CC com carga

Figura 5.9: Resposta motor CC com carga.



a) Controle com PID em Simulink®.



b) Saída do controlador PID

Figura 5.10: Implementação Controlador PID na forma RST.

5.6.3.1 Estrutura de controle na forma RST

Essa forma polinomial (figura 5.11) é introduzida na malha, que representa uma relação entre a saída $y(t)$, a variável de controle $u(t)$ e a referência $w(t)$. A estrutura RST permite a elaboração de leis de controle a partir de uma simples equação diferencial:

$$S(q^{-1})\Delta(q^{-1})u(t) = -R(q^{-1})y(t) + T(q^{-1})w(t) \tag{5.2}$$

com q^{-1} sendo o operador de atraso

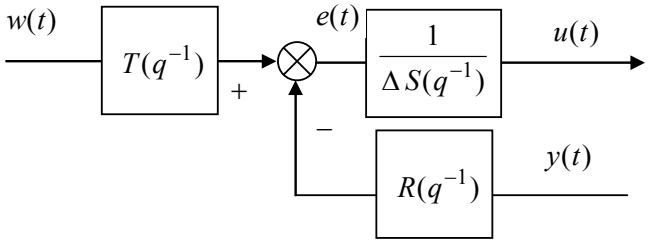


Figura 5.11: Estrutura de controlador na forma RST.

Muitos aspectos utilizados no projeto de um controlador através de alocação de pólos são utilizados no projeto de um controlador preditivo. A forma RST utilizada neste trabalho, mesmo no caso da implementação de um controlador PID discreto permitirá a comparação e validação dos diferentes algoritmos de controle, como também, uma fácil implementação da lei de controle em tempo real. Desse modo, todos os controladores implementados no simulador desenvolvido neste trabalho são apresentados nesse formato.

Outro aspecto a ressaltar é que um controlador implementado na forma RST permite visualizar facilmente a estrutura interna do controlador. Dessa forma, pode-se facilmente constatar um dos principais inconvenientes de um controlador do tipo PID em relação a controladores preditivos, ou seja, um controlador PID na forma RST apresenta apenas um grau de

liberdade, pois a dinâmica de rejeição da perturbação se iguala à dinâmica trajetória de referência, o que não acontece com os controladores preditivos, que possuem dois graus de liberdade. Assim sendo, aqui será enfatizada a utilização de controladores do tipo GPC.

5.6.3.2 Controlador PID na forma RST

Com o objetivo de utilizar a mesma estrutura de programação para o controlador, o controlador PID será implementado na forma RST, que poderá ser facilmente realizada, a partir do conhecimento da função de transferência discreta do controlador PID ($C(q^{-1})$).

Ao mesmo tempo, a equação discreta do controlador (T_a período de amostragem) poderá ser obtida através da transformação de Euler de sua função de transferência contínua:

$$C(s) = K_p\left[1 + \frac{1}{T_i s} + T_d\, s\right] \quad \underset{Euler}{\Rightarrow} \quad C(q^{-1}) = K_p\left[1 + \frac{T_a/T_i}{1-q^{-1}} + \frac{T_d}{T_a}(1-q^{-1})\right] \tag{5.3}$$

A estrutura RST é uma representação do sistema de controle possuindo como parâmetros os sinais de entrada, saída e do controlador, como mostrado na figura 5.12.

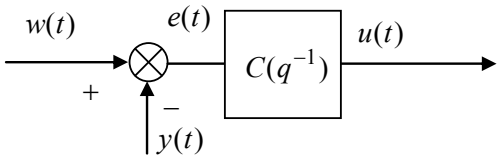


Figura 5.12: Estrutura controlador RST.

$$e(t) = w(t) - y(t) \tag{5.4}$$

$$u(t) = C(q^{-1})\, e(t) \text{ com } q^{-1} \text{ operador atraso} \tag{5.5}$$

Da equação 5.43 em 5.5, obtém-se:

$$u(t) = C(q^{-1})\,(w(t) - y(t)) \tag{5.6}$$

A partir do conhecimento da função de transferência discreta do controlador PID $C(q^{-1})$ é possível representar o controlador na forma RST:

$$\begin{aligned} C(q^{-1}) &= \frac{u(t)}{e(t)} = \frac{K_p}{1-q^{-1}}\left[1 - q^{-1} + \frac{T_a}{T_i} + \frac{T_d}{T_a}(1-q^{-1})^2\right] \\ \Rightarrow\ (1-q^{-1})\,u(t) &= K_p\left[(1 + \frac{T_a}{T_i} + \frac{T_d}{T_a}) - (1 + 2\frac{T_d}{T_a})\,q^{-1} + \frac{T_d}{T_a}q^{-2}\right]\left[w(t) - y(t)\right] \end{aligned} \tag{5.7}$$

Comparando a equação 5.7 com a equação 5.2, pode-se escrever o controlador representado pelo conjunto de equações:

$$\begin{aligned} R(q^{-1}) = T(q^{-1}) &= K_p\left[(1 + \frac{T_a}{T_i} + \frac{T_d}{T_a}) - (1 + 2\frac{T_d}{T_a})\,q^{-1} + \frac{T_d}{T_a}q^{-2}\right] \\ S(q^{-1})\Delta(q^{-1}) &= 1 - q^{-1} \end{aligned} \tag{5.8}$$

O fato de um controlador PID possuir apenas um grau de liberdade pode ser verificado através da igualdade dos polinômios R e T na relação matemática descrita anteriormente.

5.6.3.3 Controlador PID na forma RST no Simulink®

Utilizando o bloco *Discrete Filter* do Simulink® (figura 5.13), um controlador PID na forma RST é descrito através das equações descritas a seguir:

$$
\begin{aligned}
&T = t_0 + t_1 z^{-1} + t_2 z^{-2} \\
&R = r_0 + r_1 z^{-1} + r_2 z^{-2} \\
&S = 1 + s_1 z^{-1} \\
&U(t) = -u(t^{-1}) \times s_1 - r_0 y(t) + r_1 y(t^{-1}) + r_2 y(t^{-2}) + t_0 w(t) + w_1 w(t^{-1}) + w_2 w(t^{-2})
\end{aligned}
\tag{5.9}
$$

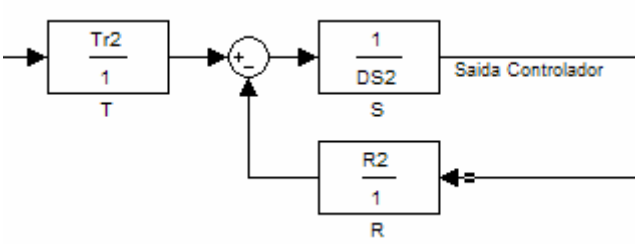
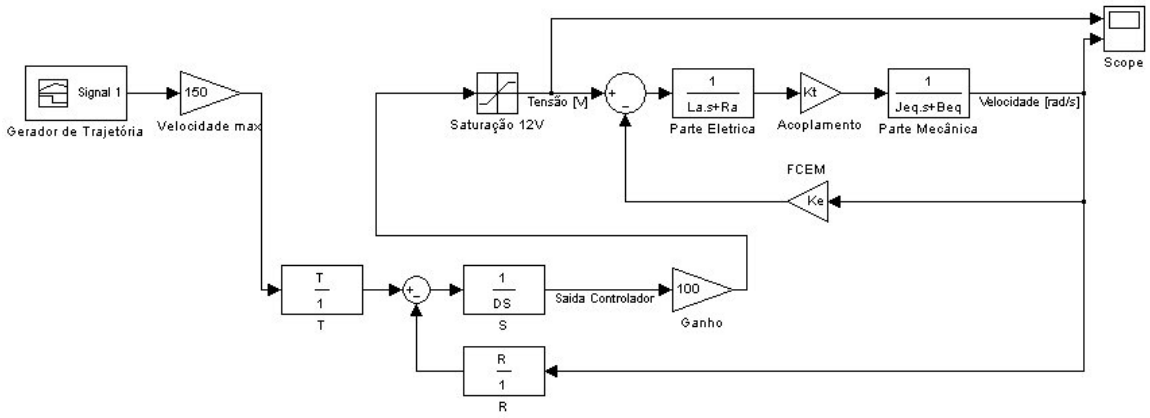
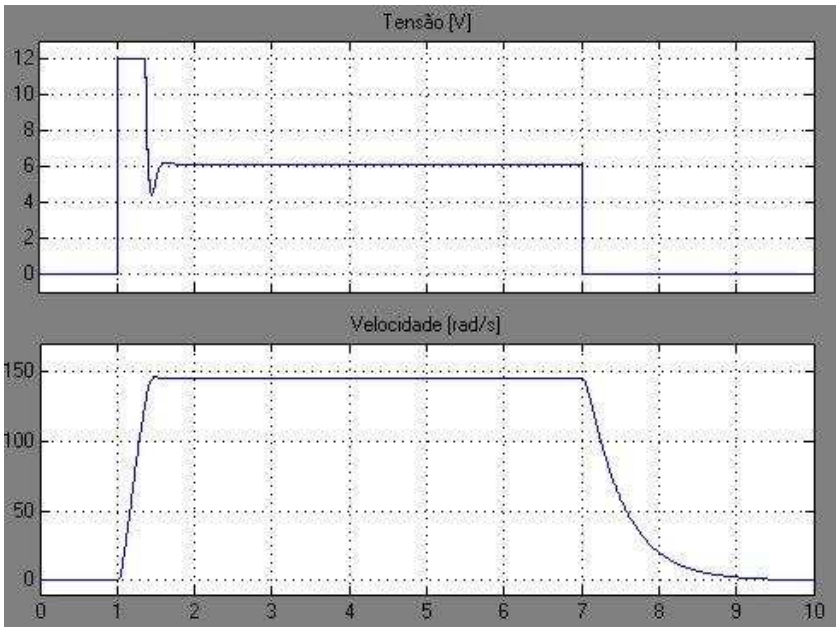


Figura 5.13: Controlador RST em Simulink®.

A implementação completa de uma junta robótica em Simulink® é apresentada na figura 5.14.



a) Diagrama de blocos Simulink®.



b) Resultados - saída controlador

Figura 5.14: Controlador RST: Implementação em Simulink® e Resultados obtidos.

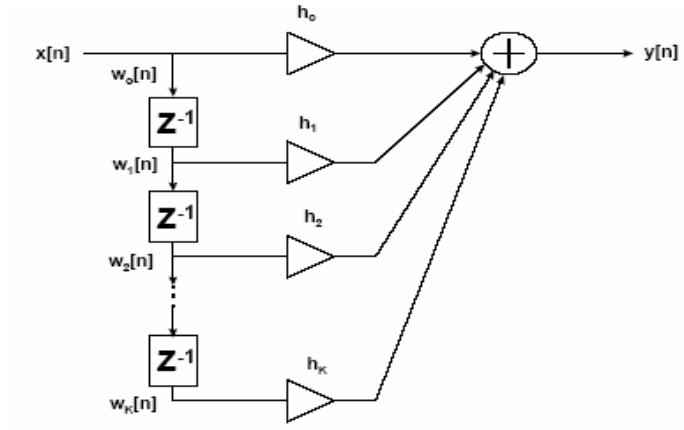
5.6.4 Controlador PID na forma RST no DSP Builder

A função de transferência correspondente a um modelo direto de filtro do Simulink® é dada por:

$$H(s) = \frac{y(s)}{u(s)} = \frac{num(s)}{den(s)} = \frac{num(1)s^{nn-1} + num(2)s^{nn-2} + \ldots + num(nn)}{den(1)s^{nd-1} + den(2)s^{nd-2} + \ldots + den(nd)} \quad (5.10)$$

5.6.4.1 Filtros FIR e IIR

Um filtro do tipo FIR– *Finite Impulse Response* correspondente a uma resposta de um impulso que tem comprimento finito. Assim, os valores de saídas (y[n]) são calculados a partir de valores anteriores de x[n] conhecidos, como podemos verificar na equação diferença representada na figura 5.15.

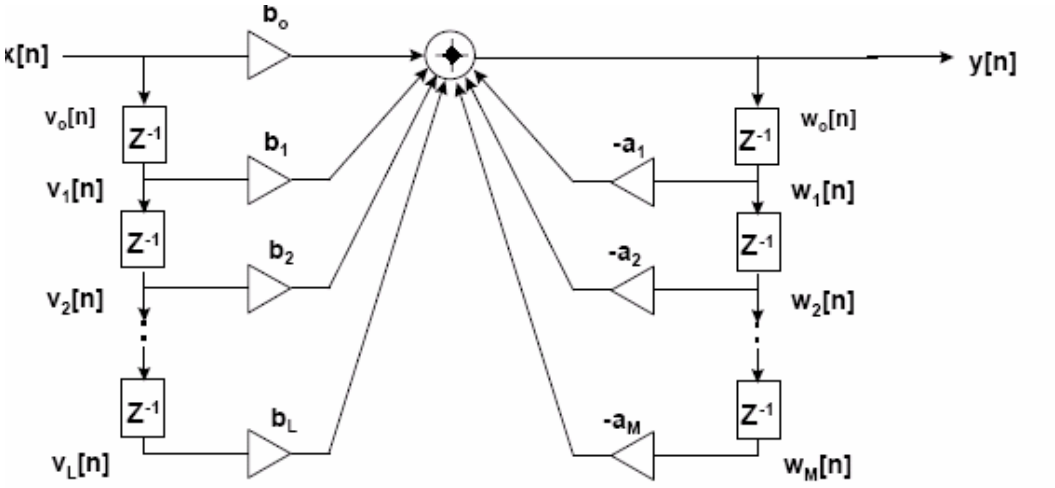


$$y[n] = \sum_{k=0}^{\infty} h[k]x[n-k]$$

Figura 5.15: Filtro FIR.

Por outro lado, um filtro do tipo IIR– *Infinite Impulse Response* correspondente a uma resposta de um impulso que tem comprimento infinito. Assim, os valores de saídas (y[n]) são calculados a partir dos valores anteriores de x[n] conhecidos e de y[n], como podemos verificar

na equação diferença representada na figura 5.16. Por esta razão os filtros IIR são freqüentemente chamados filtros recursivos.



$$y[n] = \sum_{i=1}^{M} \left(-a_i y[n-i]\right) + \sum_{j=0}^{L} \left(b_i x[n-j]\right)$$

Figura 5.16: Filtro IIR.

5.6.4.2 Implementação do controlador RST

Os blocos de filtros podem ser implementados através de filtros digitais constituídos de blocos de atraso, multiplicadores e acumuladores, conforme mostra a figura 5.17.

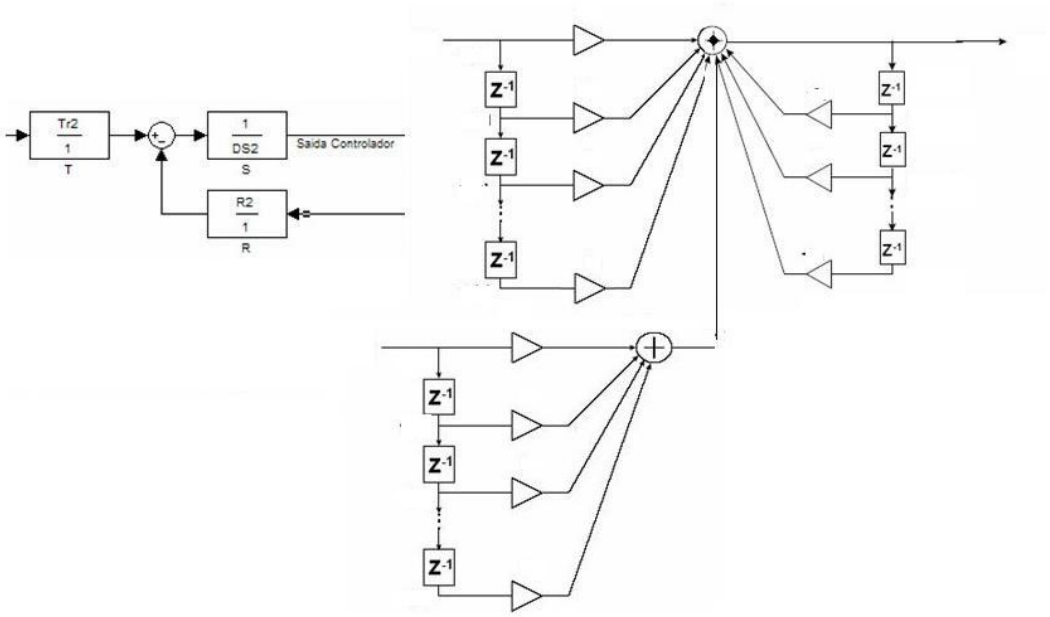


Figura 5.17: Implementação do controlador

Inicialmente implementamos o controlador utilizando blocos de multiplicadores, somadores e atrasos, da biblioteca do DSP Builder. Deste modo, pode-se comparar o resultado das simulações considerando, agora, os erros gerados devido à quantização dos valores em um sistema amostrado digitalmente.

No modelo inclui-se também o bloco do *Signal Compiler*, *Board Definitions* e *Clock Definition*, Estes blocos permitirão a conversão gráfica do modelo efetuado em um projeto em VHDL para ser implementado na FPGA. Tais blocos devem estar no topo de hierarquia do arquivo de modelo, conforme mostrado na figura 5.18, a implementação do filtro pode ser verificada na figura 5.19.

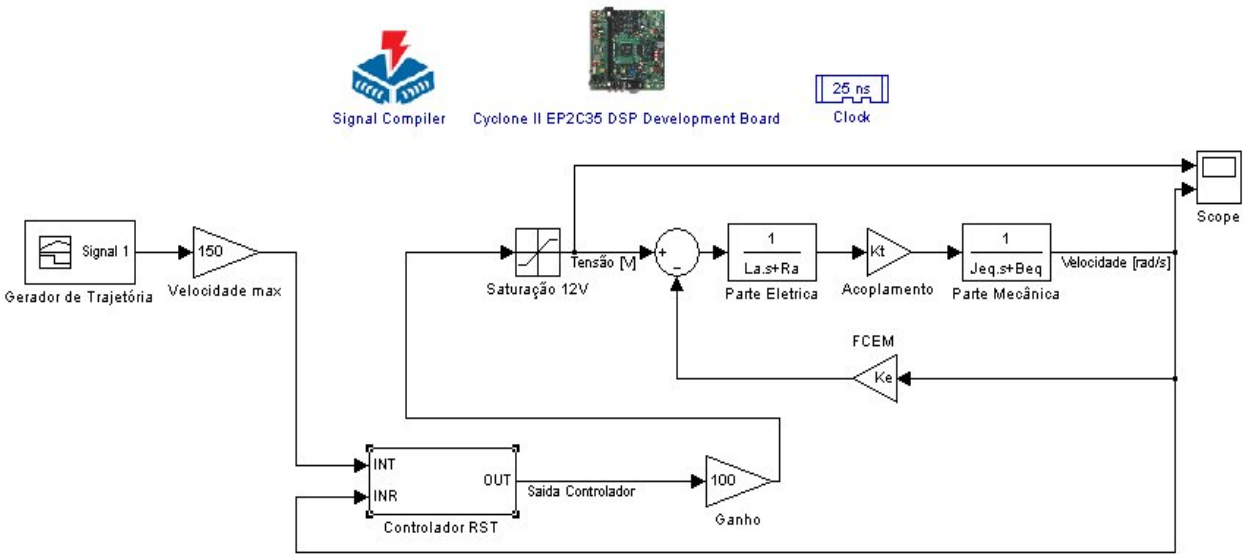
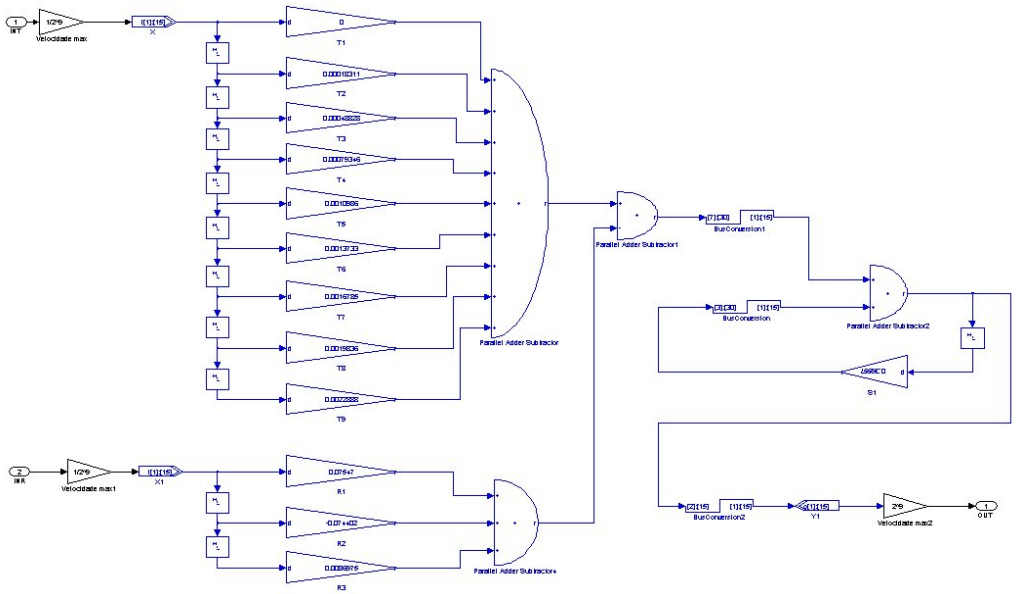
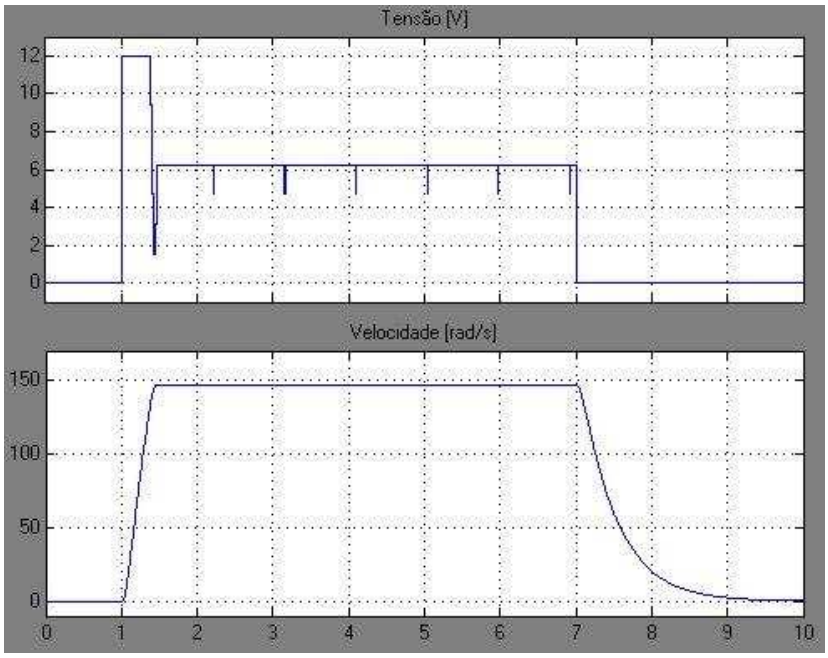


Figura 5.18: Controle com RST – Simulink®.

O controlador RST implementado na figura 5.19 possui uma resolução de 16 bits, sendo que os dados estão no formato de Q15, ou seja 1 bit para sinal e 15 bits como casa decimais (formato 1.15). Após a etapa de multiplicação têm-se os dados no formato 2.30 e após a soma 7.30. Estes dados são então convertidos novamente para o formato 1.15. Os blocos de ganho nas entradas e saída foram utilizados para conversão de tipo de dados entre os blocos do Simulink® e DSP Builder.



a) Bloco controlador RST



b) Saída controle com RST – Simulink®

Figura 5.19: Implementação do Controlador RST – DSP Builder.

Após a verificação do resultado no simulador é possível a geração do projeto em VHDL, bem como a síntese e gravação da lógica na placa de teste. Para isto, basta executar o bloco *Signal compiler*. A figura 5.20 mostra a tela de compilação, após o passo 3 (*program*) o controlador já estará gravado e operando na placa.

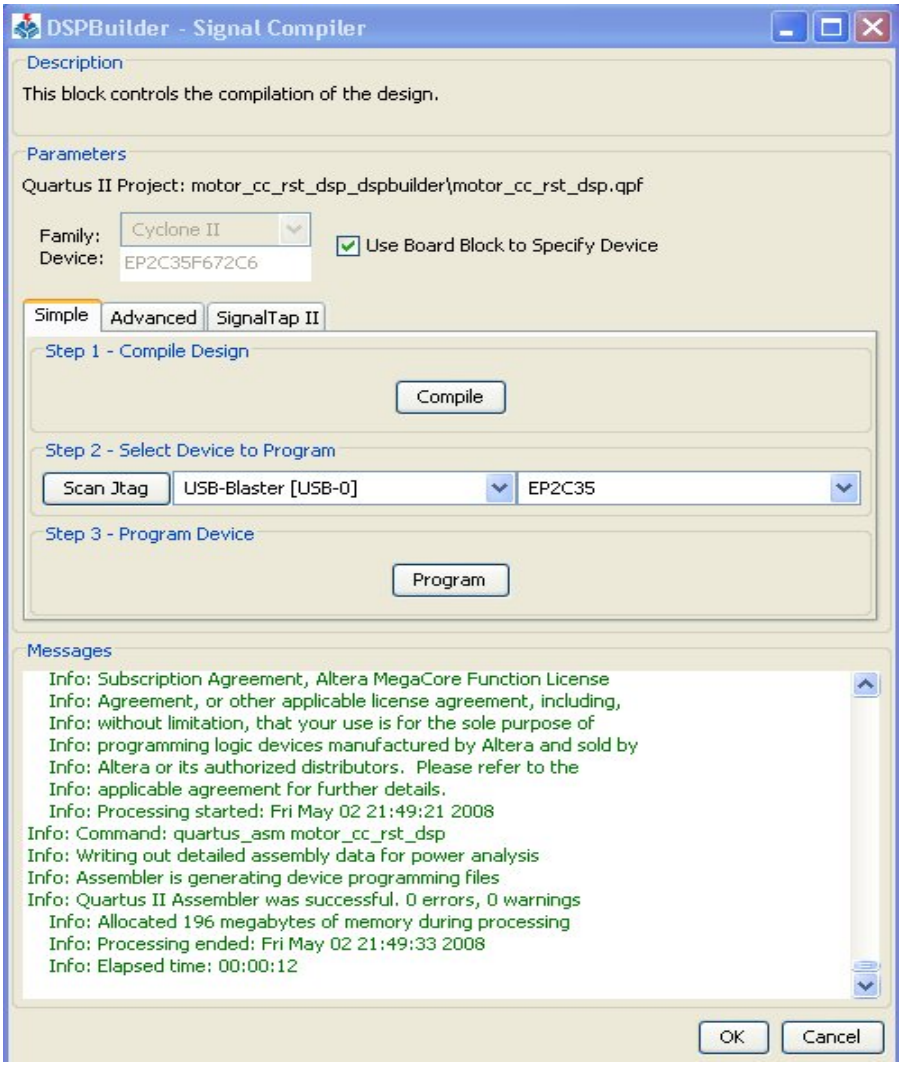


Figura 5.20: Tela de geração do DSP Builder.

No relatório de compilação no ambiente Quartus II (figura 5.21), pode-se verificar a quantidade de elementos lógicos utilizados bem como a quantidade de multiplicadores dedicados utilizados para a implementação do controlador. Neste modelo não é possível alterar os valores dos coeficientes sem que o projeto seja novamente compilado, pois os valores estão colocados como constantes.

Flow Status	Successful - Fri May 02 21:49:33 2008
Quartus II Version	7.2 Build 151 09/26/2007 SJ Full Version
Revision Name	motor_cc_rst_dsp
Top-level Entity Name	motor_cc_rst_dsp
Family	Cyclone II
Device	EP2C35F672C6
Timing Models	Final
Met timing requirements	N/A
Total logic elements	440 / 33,216 (1 %)
Total combinational functions	353 / 33,216 (1 %)
Dedicated logic registers	160 / 33,216 (< 1 %)
Total registers	160
Total pins	50 / 475 (11 %)
Total virtual pins	0
Total memory bits	0 / 483,840 (0 %)
Embedded Multiplier 9-bit elements	16 / 70 (23 %)
Total PLLs	0 / 4 (0 %)

Figura 5.21: Relatório de compilação Quartus

5.6.5 Controlador RST – VHDL Import

O código gerado pelo DSP Builder utiliza bibliotecas do fabricante, no caso a ALTERA, permitindo assim a rápida implementação, entretanto este código não poderá ser sintetizado em componentes de outro fabricante. No entanto, é possível gerar blocos próprios ou implementação própria de codificações, para isto, deve-se implementar o modelo desejado em VHDL e utilizar o bloco HDL *Import*. Esta ferramenta criará um modelo para simulação no ambiente Matlab/Simulink® com a descrição do modelo desejado. Considerando o controlador utilizado

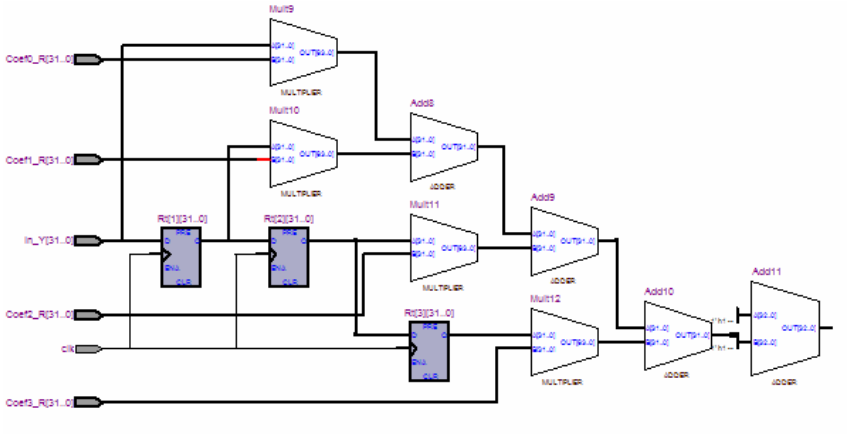
para o sistema, foi elaborado o código de modelagem em VHDL. No anexo B dessa dissertação é apresentado o código do controlador implementado em linguagem VHDL.

Neste modelo, os coeficientes do controlador estão disponíveis como sinais de entrada. Deste modo, estes valores podem ser alterados durante a simulação. Outra opção é que estes valores sejam programados via portas de saídas de um processador também colocado internamente à FPGA. Porém, há um consumo maior de elementos lógicos para a implementação do controlador em relação ao elaborado pelo DSP Builder, conforme mostra o relatório de compilação do Quartus II (figura 5.22).

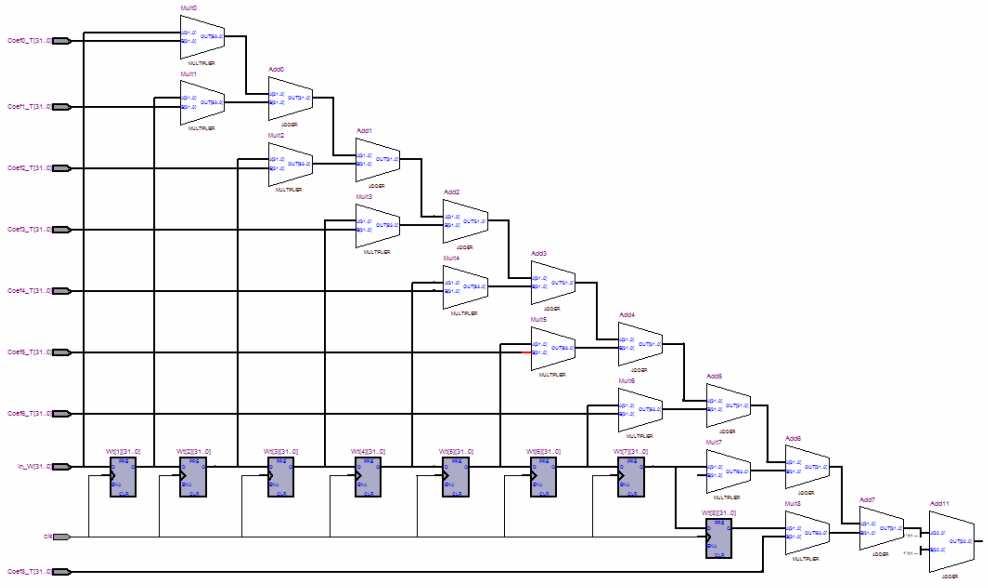
Flow Status	Successful - Fri May 02 21:57:52 2008
Quartus II Version	7.2 Build 151 09/26/2007 SJ Full Version
Revision Name	RST_contr_q15
Top-level Entity Name	RST_contr_q15
Family	Cyclone II
Device	EP2C35F672C6
Timing Models	Final
Met timing requirements	Yes
Total logic elements	320 / 33,216 (< 1 %)
Total combinational functions	192 / 33,216 (< 1 %)
Dedicated logic registers	128 / 33,216 (< 1 %)
Total registers	128
Total pins	257 / 475 (54 %)
Total virtual pins	0
Total memory bits	0 / 483,840 (0 %)
Embedded Multiplier 9-bit elements	26 / 70 (37 %)
Total PLLs	0 / 4 (0 %)

Figura 5.22: Relatório de compilação Quartus II.

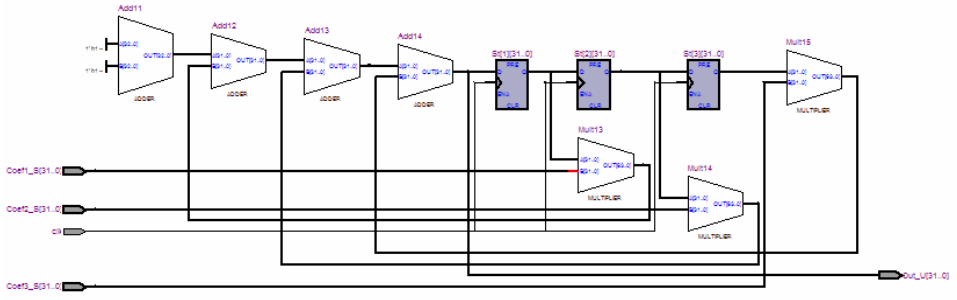
Após a compilação podemos verificar a síntese do projeto do controlador. Na figura 5.23 pode-se observar as etapas de soma, multiplicação e os registros de atrasos das amostras.



a) Bloco R



b) Bloco T



c) Bloco S

Figura 5.23 – Síntese do controlador

Para a geração do modelo com bloco de HDL *Import*, basta ter os arquivos descritos em VHDL ou um projeto do Quartus II com estes arquivos. A tela do HDL *Import* é mostrada na figura 5.24. Após a compilação do projeto, o bloco mostrado na figura 5.25b é criado pelo DSP Builder.

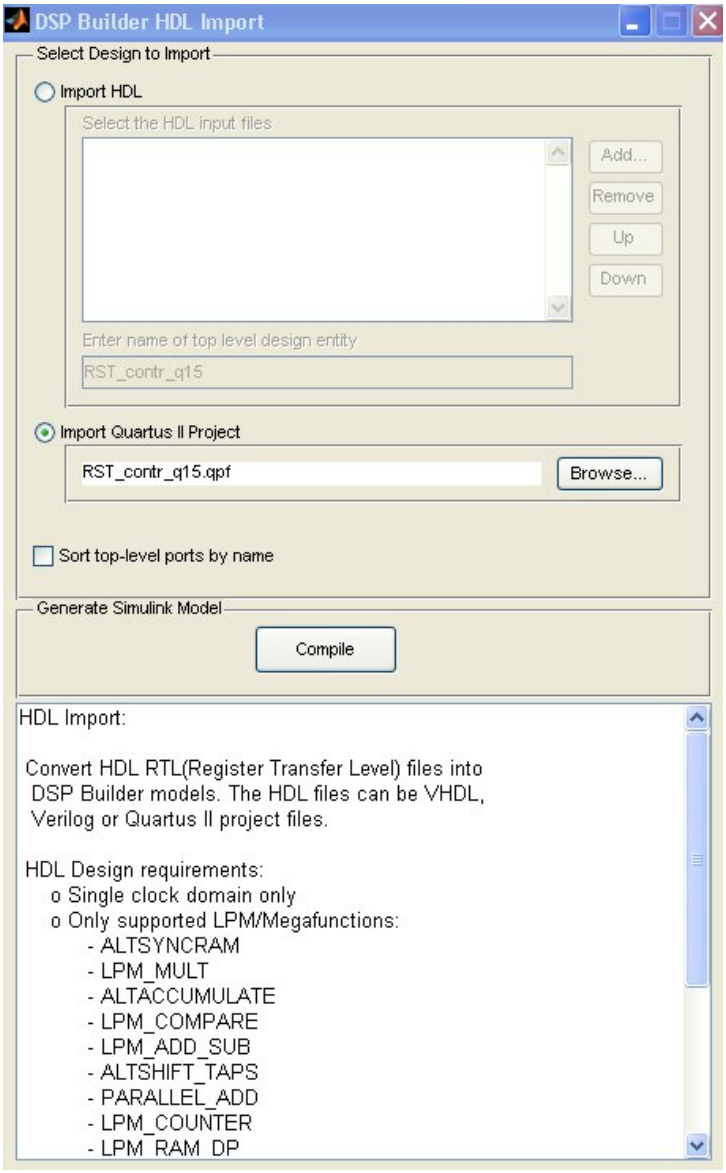
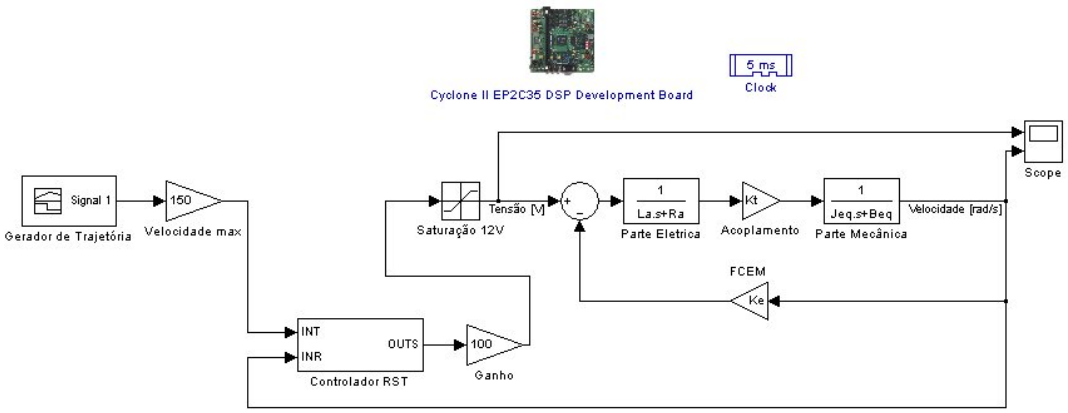
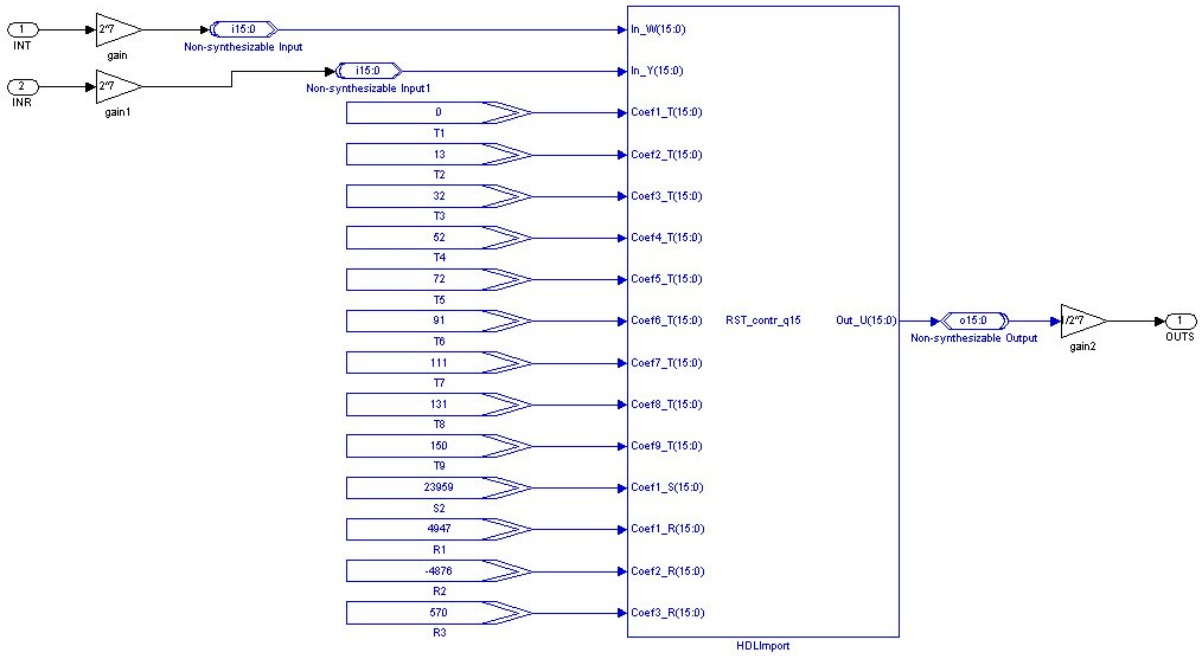


Figura 5.24: Tela de geração do DSP Builder.

Na figura 5.25 temos a implementação dos modelos em VDHL utilizando-se o *HDL Import Block*, os resultados da simulação (figura 5.26).



a) Controle com RST – VHDL Import.



b) Controle com RST – Bloco VHDL Import

Figura 5.25: Implementação do Controlador RST – VHDL import.

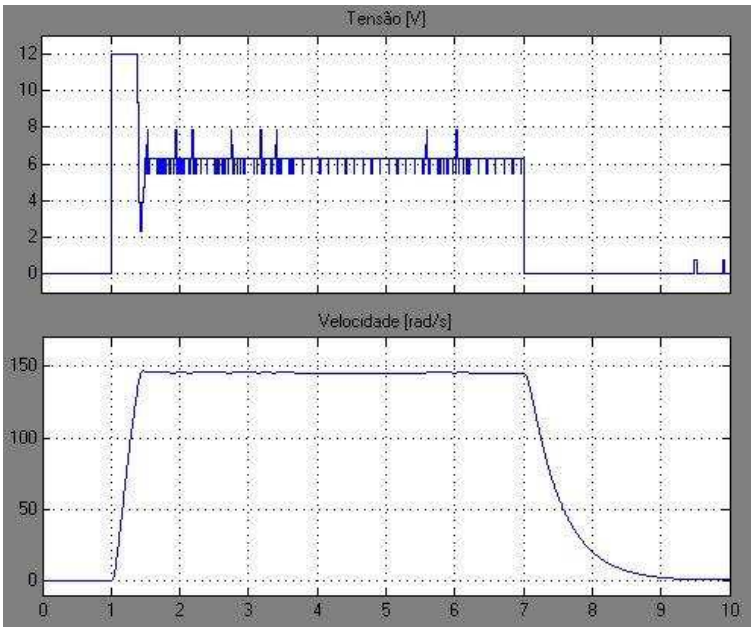


Figura 5.26: Saída com controle RST – VHDL import.

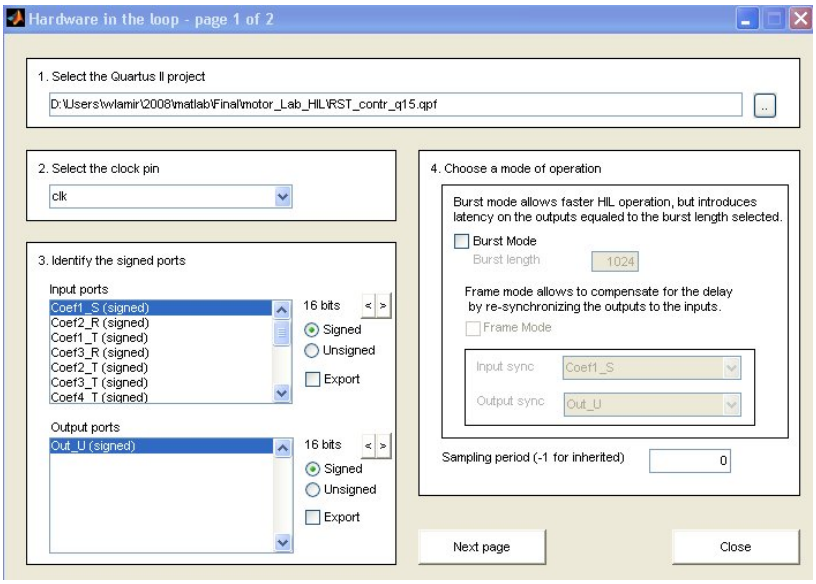
5.7 Controlador RST – HIL (Hardware In the Loop)

Para a verificação da implementação final no FPGA, o DSP Builder e o MatLab possuem a opção de *Hardware In de Loop* (HIL). Após o projeto já sintetizado e roteado pelo ambiente Quartus II, executa-se a configuração da FPGA e a criação de um bloco para a simulação no Matlab.

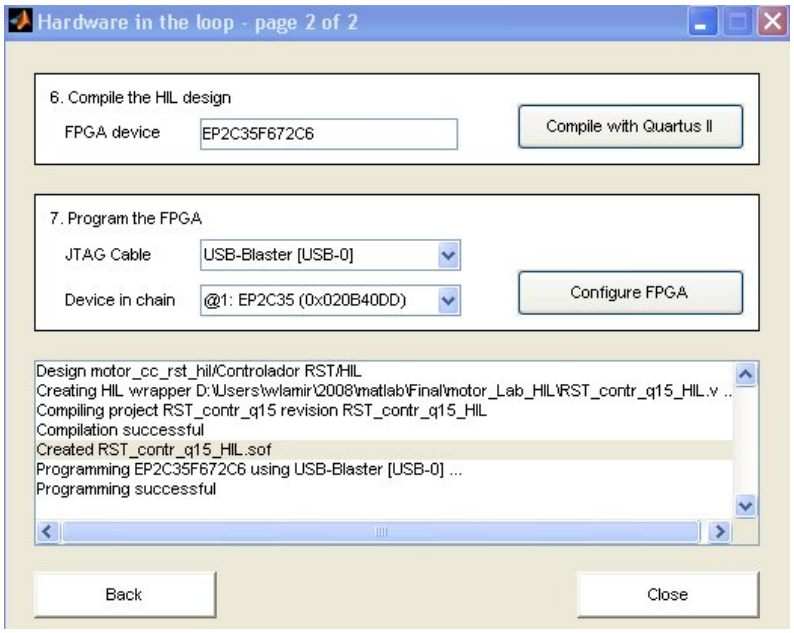
Para a geração do bloco HIL é necessário que um projeto do Quartus II já esteja criado com o conteúdo do bloco a ser simulado. Após a leitura deste projeto pelo DSP Builder é necessário a definição dos sinais de interface e o modo de transferência de dados entre o MatLab e componente.

Após isto, uma nova compilação do projeto é executada para se adicionar o hardware necessário para a transferência dos dados internamente ao FPGA. Ao término da compilação executa-se a configuração ou gravação do FPGA.

Este processo é executado na tela do *Hardware In the Loop*, conforme mostrado na figura 5.27.



a) Tela de geração do HIL – pág 1



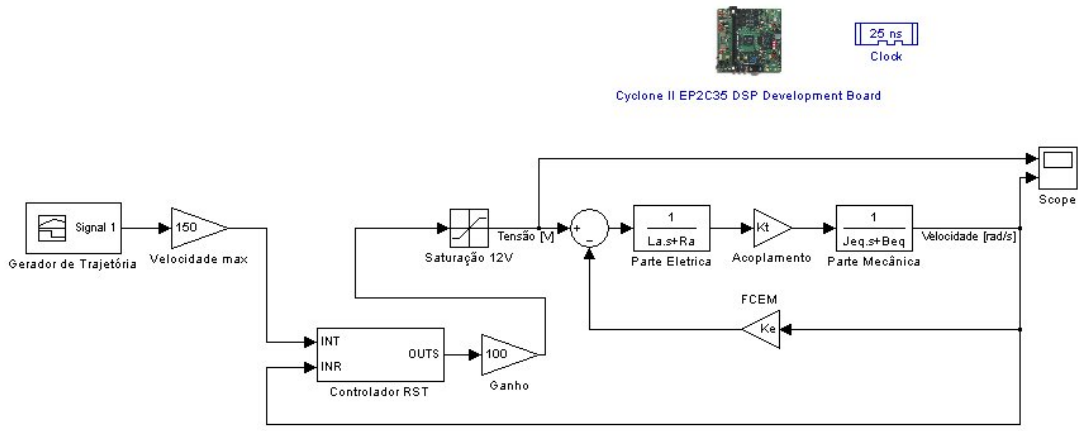
b) Tela de geração do HIL – pág 2

Figura 5.27 – Tela de geração do HIL

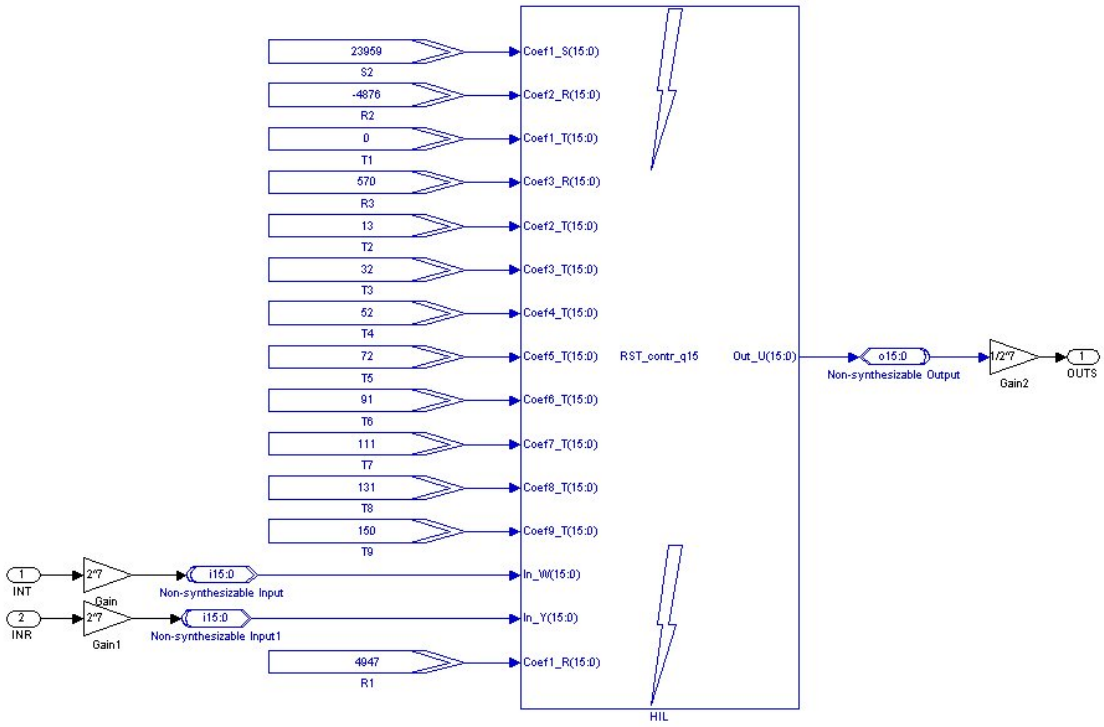
Ao se iniciar a simulação, o MatLab irá gerar blocos de dados que são então enviados para a FPGA via interface JTAG. Estes dados de entrada são processados pelo *hardware* implementado internamente ao FPGA, o resultado da saída é por sua vez recuperado pelo MatLab.

O MatLab utiliza estes dados lidos do componente para continuar a simulação e exibir os resultados. Deste modo é possível validar o circuito após o roteamento final do componente.

Nas figuras 5.28 e 5.29 temos respectivamente o Bloco H*ardware In the Loop* no MatLab e o resultado da simulação.



a) Controle com RST – HIL



b) Controle com RST – Bloco HIL

Figura 5.28 – Controle com RST - HIL

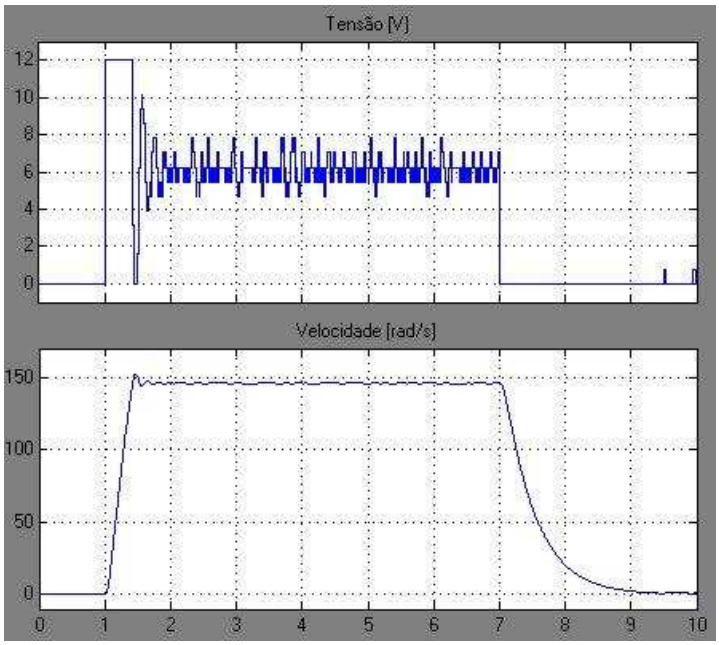
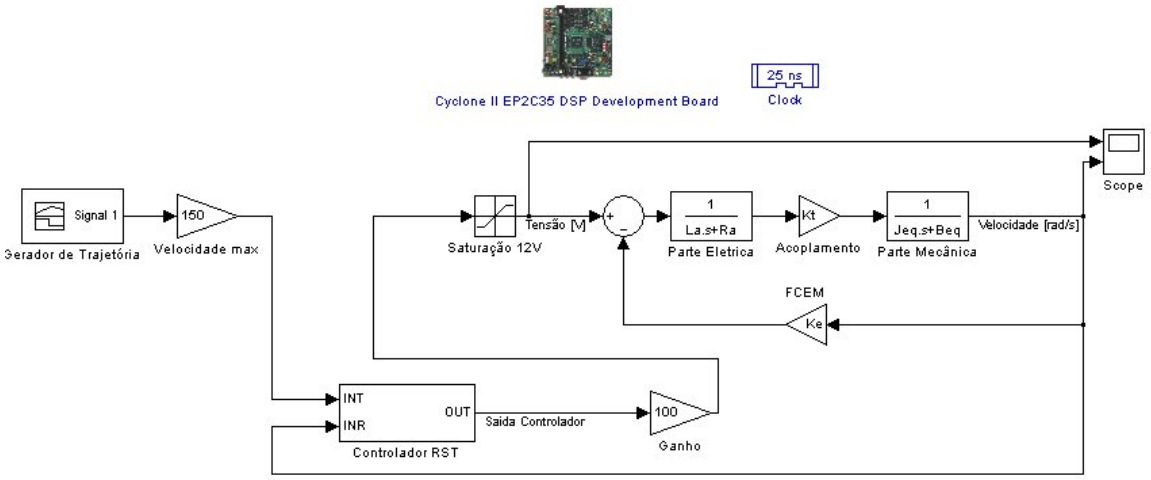


Figura 5.29 – Saída controle com RST – HIL

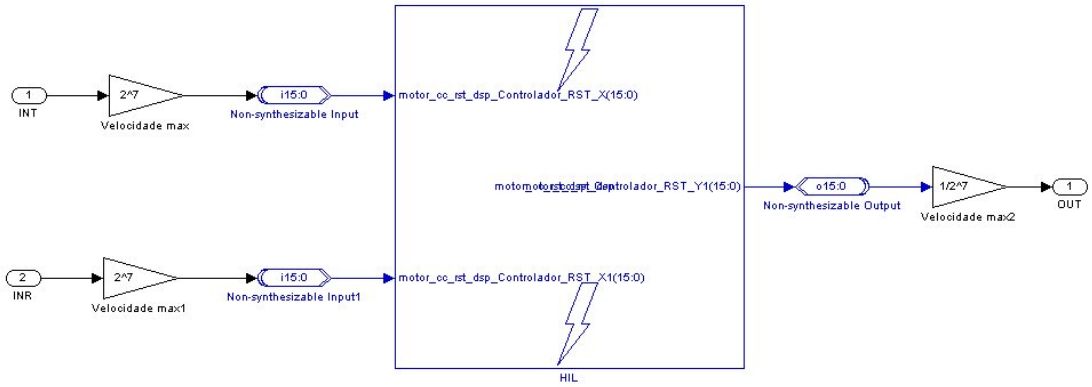
5.8 Controlador RST via DSP Builder - HIL

No item 5.7 foi mostrado a utilização do HIL com o projeto do controlador elaborado inicialmente em VHDL. Neste item temos o controlador projetado utilizando-se o DSP Builder sendo utilizado para a execução do bloco HIL.

Os processos de criação do bloco e a simulação são o mesmo descrito anteriormente, porém o projeto utilizado foi criado anteriormente pelo DSP Builder. (Item 5.6.4). As figuras 5.30 e 5.31 ilustram os modelos no MatLab e o resultado da simulação. Podemos comparar estes resultados com os da simulação na figura 5.19.



a) Controle com RST DSP Builder - HIL



b) Controle com RST DSP Builder – Bloco HIL

Figura 5.30 – Controle com RST DSP Builder - HIL

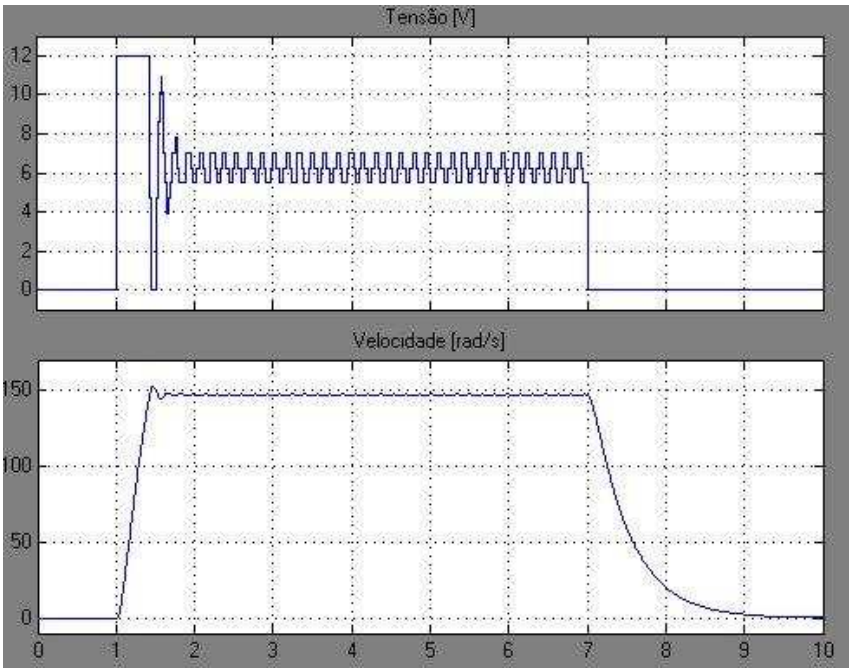


Figura 5.31 – Saída controle com RST DSP Builder - HIL

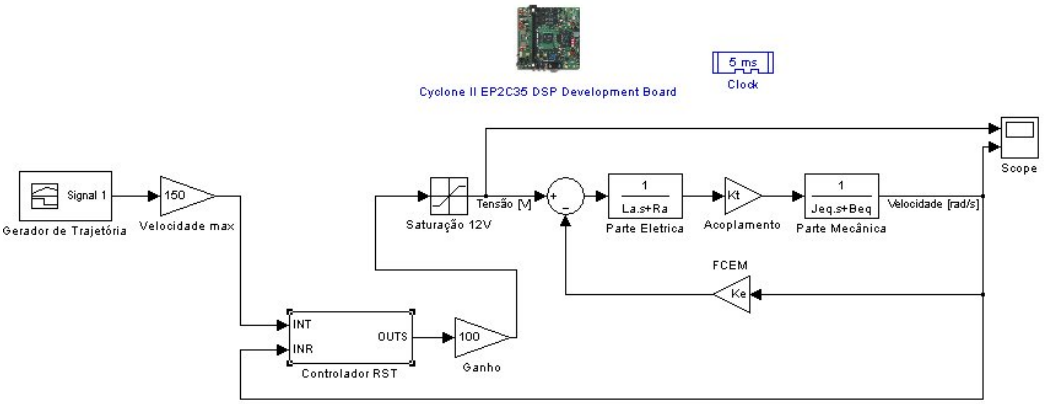
5.9 Controlador RST – VHDL parâmetros constantes no VHDL

A partir da definição dos coeficientes do controlador, pode-se implementar um código em VHDL com estes valores utilizados como constantes, visando assim a redução de elementos lógicos utilizados, bem como de multiplicadores, conforme relatório de compilação apresentado na figura 5.32.

Assim, o código VHDL seria praticamente o mesmo, a única alteração é que os coeficientes não serão mais sinais de entrada do bloco, mas sinais internos com valores pré-definidos, como podem ser verificados através do código apresentado no Anexo C. O bloco gerado não possui entrada para os coeficientes (figura 5.33).

Flow Status	Successful - Sat May 03 20:40:33 2008
Quartus II Version	7.2 Build 151 09/26/2007 SJ Full Version
Revision Name	RST_contr_q15
Top-level Entity Name	RST_contr_q15
Family	Cyclone II
Device	EP2C35F672C6
Timing Models	Final
Met timing requirements	Yes
Total logic elements	263 / 33,216 (< 1 %)
Total combinational functions	140 / 33,216 (< 1 %)
Dedicated logic registers	128 / 33,216 (< 1 %)
Total registers	128
Total pins	49 / 475 (10 %)
Total virtual pins	0
Total memory bits	0 / 483,840 (0 %)
Embedded Multiplier 9-bit elements	22 / 70 (31 %)
Total PLLs	0 / 4 (0 %)

Figura 5.32 – Relatório de compilação Quartus II.



a) Controle com RST – VHDL constantes

b) Controle com RST – VHDL constantes – Bloco HIL

Figura 5.33 – Controle com RST – VHDL constantes

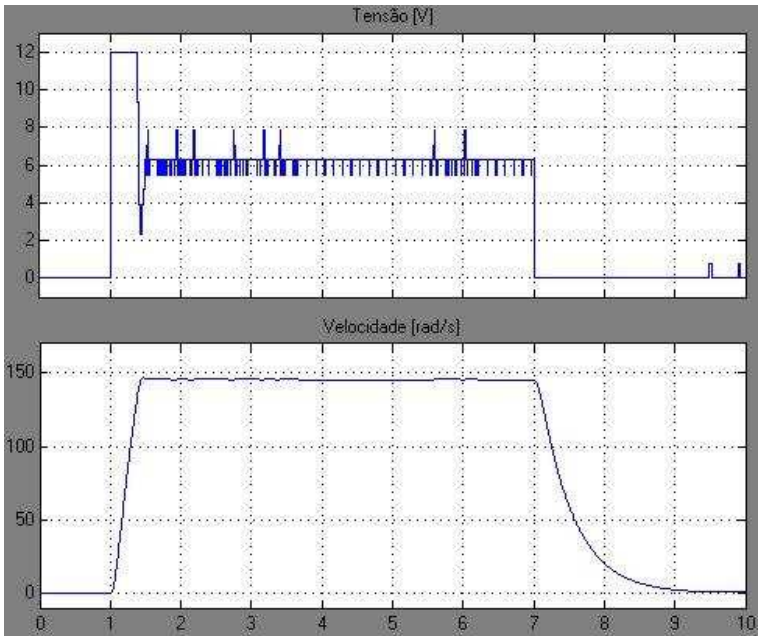


Figura 5.34 – Saída controle com RST - VHDL constantes

5.10 Implementação Final

Como proposta de validação dos blocos simulados, podemos considerar a interface de controle de juntas de um robô móvel autônomo acionado por dois motores CC com respectivos encoders de posição (figura 5.35).

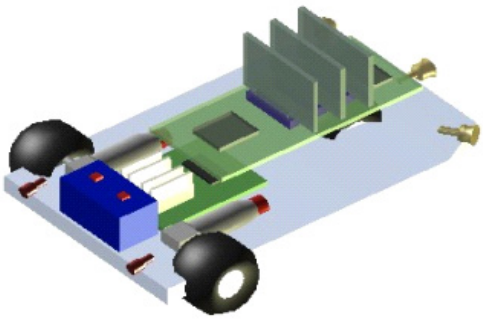


Figura 5.35 – Robô Móvel Autônomo

5.10.1 Arquitetura proposta para implementação

A figura 5.36 ilustra a arquitetura proposta para validação da interface de acionamento e controle sendo constituída de um bloco de processamento, bloco de contador de posição para tratamento de informações de *encoders* para obtenção da velocidade, bloco controlador para cada roda, bloco interface PWM e interface de potência para acionamento dos motores de cada roda. Através de uma interface de comunicação o processador poderá receber comandos e tabelas de trajetórias, bem como a reprogramação dos parâmetros do controlador.

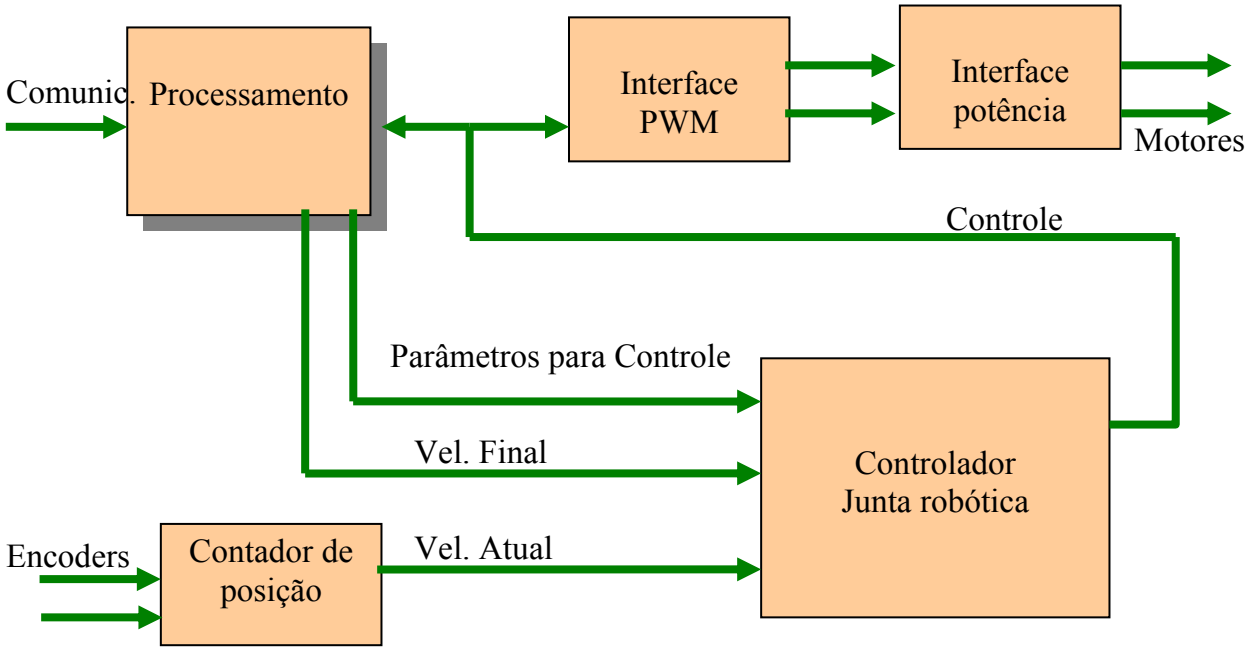


Figura 5.36 –Controle Robô Móvel Autônomo

5.10.2 Implementação com processador Nios II

Através da utilização do SOPC (*System On Programable Chip)* foi implementado a partir de um processador Nios com interface para controladores RST, duas portas de saída para definir a velocidade dos motores e uma interface serial para a comunicação com o sistema supervisório. Os blocos de leitura de velocidade e PWM foram adicionados e interligados em esquema elétrico do Quartus II, a folha principal do projeto pode ser verificada na figura 5.37.

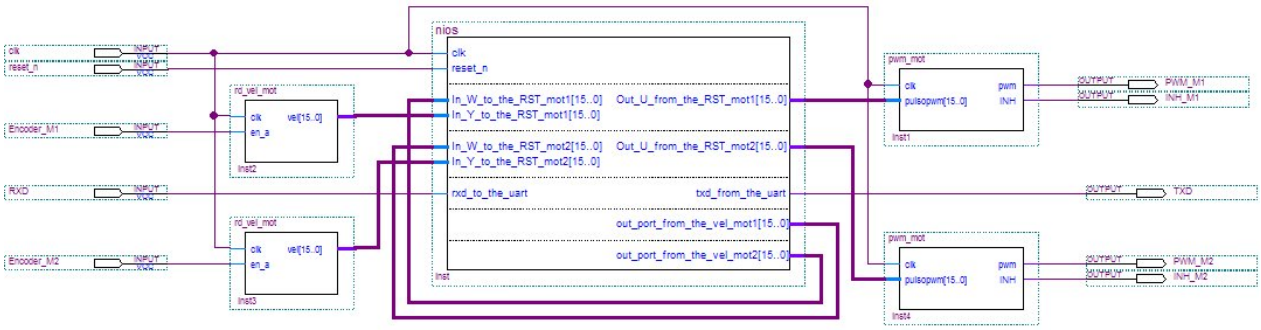


Figura 5.37 – Sistema de controle com processador Nios II

Para validação e testes do sistema foram utilizados kits de desenvolvimento da ALTERA disponíveis no LAR-UNICAMP em bancada de testes descrita anteriormente. As figuras 5.38, 5.39 e 5.40 ilustram o procedimento de teste utilizado.

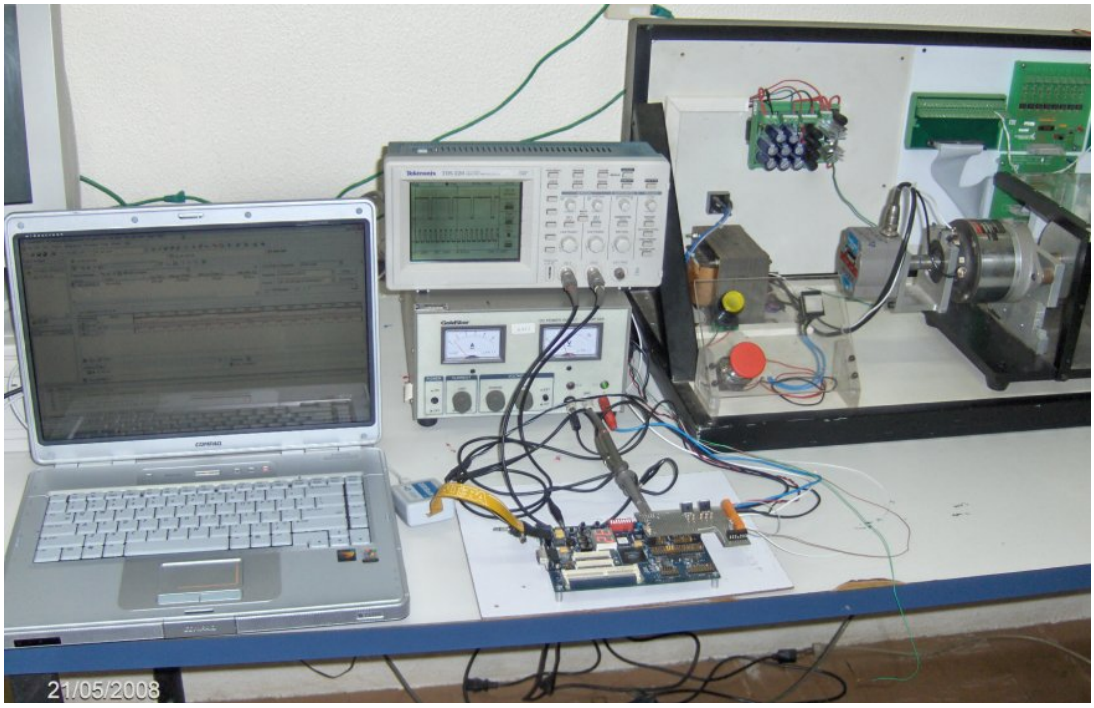


Figura 5.38 – Teste do sistema controle com processador Nios II

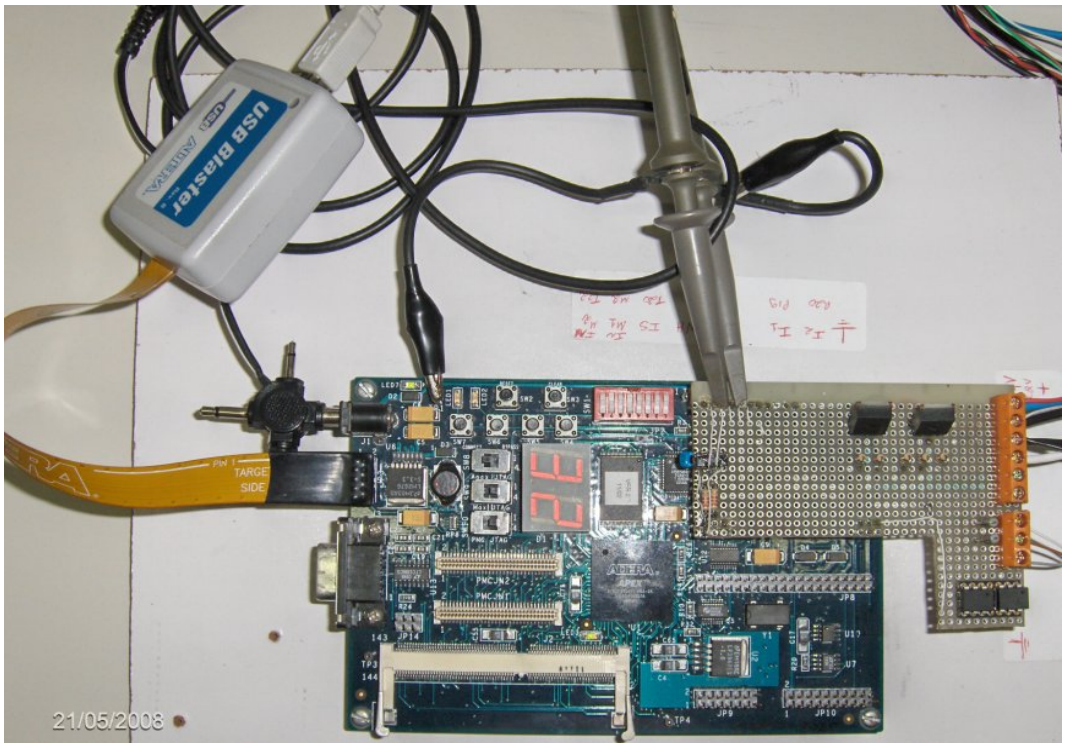


Figura 5.39 – Kit Altera com interface Encoder e Motor

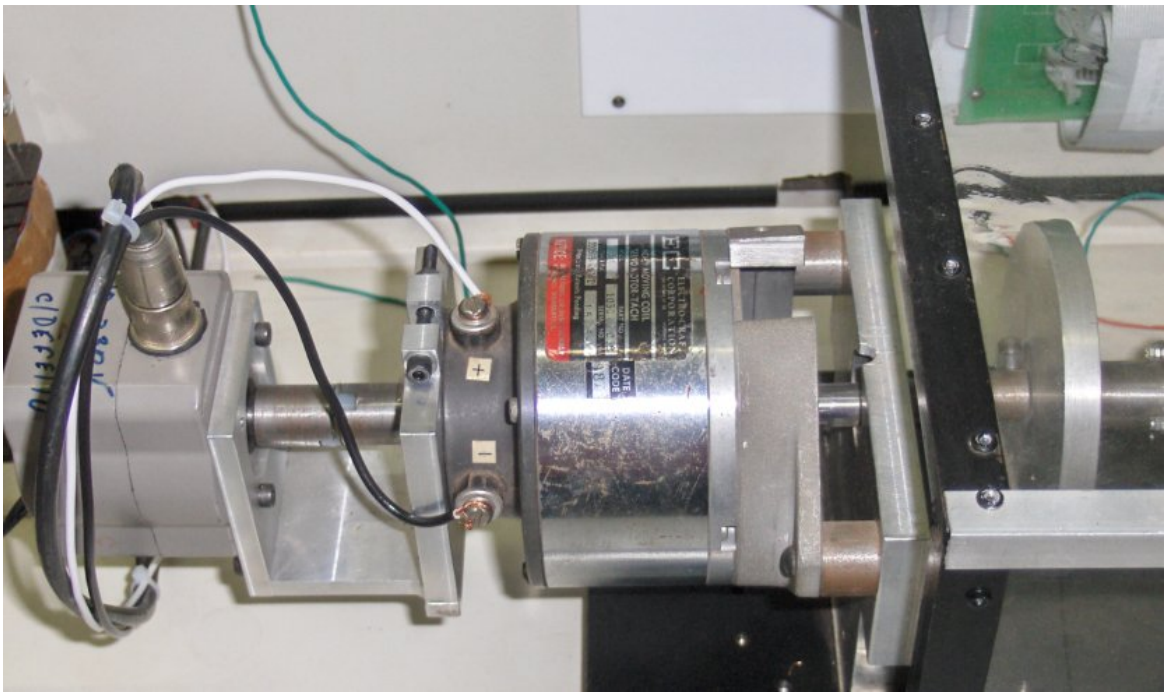


Figura 5.40 – Bancada Implementada com Encoder, Motor e Inércia

A figura 5.41 mostra a tela do osciloscópio com a entrada do *encoder* (canal 1) e a saída do PWM (canal 2).

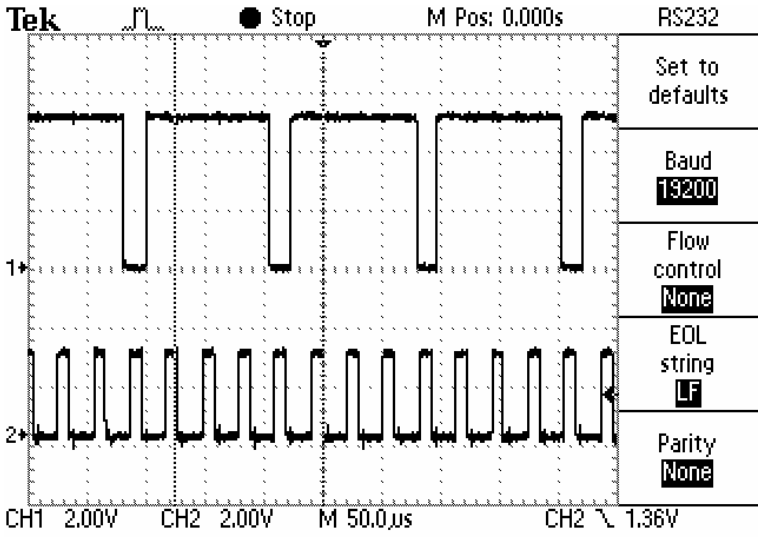


Figura 5.41 – Formas de Onda do Encoder e PWM.

Para verificação e validação dos resultados, o ambiente Quartus II permite também a inclusão de um analisador lógico no projeto. Esta ferramenta é chamada de Signal Tap II e permite a visualização em tempo real dos sinais internos da FPGA implementada. Os valores destes sinais são adquiridos e armazenados em blocos de memória RAM interna e transferidos ao PC através de interface JTAG. Os dados podem ser vistos no Quartus II ou no MatLab. A figura 5.42 apresenta as mesmas formas de onda do osciloscópio e a saída do medidor de velocidade, pelo Quartus II.

Figura 5.42 – Formas de Onda no Signal Tap II

5.11 Conclusão

Nesse capítulo foram descritos o procedimento experimental, proposta para validação dos conceitos de prototipagem rápida abordados nos capítulos iniciais deste trabalho, através da implementação de bancada experimental contendo os principais elementos constituintes de uma junta robótica (motor de corrente contínua, inércia, redutores e acoplamento). Este estudo possibilitou a validação experimental dos conceitos apresentados através da prototipagem rápida do controlador de posição de uma junta robótica, utilizando inicialmente Matlab-Simulink® e posteriormente ferramentas de prototipagem utilizando linguagem VHDL.

Capítulo 6

Conclusões Finais e perspectivas futuras

6.1 Conclusões e resultados obtidos

Considerando-se o atual cenário tecnológico e o alto desenvolvimento de soluções cada vez mais integradas para concepção e validação de dispositivos mecatrônicos, a utilização de ferramentas de prototipagem rápida para modelagem de controladores industriais torna-se uma indispensável opção para agilizar e facilitar a implementação de diferentes arquiteturas de controle, possibilitando assim a redução do tempo de finalização de um projeto ou estudo cientifico, atendendo uma eficiência desejada.

A eletrônica embarcada apresenta restrições quanto ao consumo de energia e quanto ao volume físico ocupado. Considerando ainda o grande número de tarefas e operações que dispositivos embarcados móveis podem realizar, bem como a variedade de combinações de sensores e atuadores usados na realização destas tarefas, é necessário prever em sua eletrônica, aspectos de flexibilidade e facilidade de alterações no projeto.

Sistemas de desenvolvimento baseados em computação reconfigurável apresentam características adequadas para execução desta classe de projeto. Eles permitem, por exemplo, que o controle dos atuadores seja feito de modo a aproveitar as características de baixo consumo, alta velocidade de operação, capacidade de integração, flexibilidade e facilidade de programação dos dispositivos lógicos reprogramáveis.

Ferramentas que venham a permitir a rápida implementação prática de um projeto de hardware não são apenas uma comodidade técnica, mas são também essenciais para a viabilidade econômica deste projeto. Isto é válido para a empresa, para a universidade e centros de pesquisa, cada vez mais sujeitos as competições do mercado. Espera-se com este trabalho contribuir para o esclarecimento dos potenciais destes sistemas, bem como, através de exemplos práticos, demonstrarem sua aplicação na área de controle de sistemas embarcados. A utilização de interfaces gráficas e VHDL no projeto do controlador permite descrever um projeto típico em lógica reconfigurável, onde diferentes blocos são projetados separadamente para posterior integração. O ambiente de simulação permite a visualização de possíveis erros de projeto, antes mesmo da execução física do mesmo. O ambiente de desenvolvimento contribui ainda para o baixo custo final do projeto, sua facilidade de implementação e de correção de erros. Outra vantagem do ambiente de desenvolvimento é a característica de hierarquização, a qual facilita o trabalho em equipe.

A utilização de lógica reconfigurável contribui para verificar os limites da aplicação desta alternativa de implementação de algoritmos em problemas práticos. Outra contribuição são as interfaces desenvolvidas em hardware, disponíveis para reprodução da comunidade acadêmica. O ambiente proposto é ainda uma ferramenta de apóio didático em laboratórios de graduação e pós-graduação de engenharia, já sendo utilizado com sucesso no Laboratório de Automação Integrada e Robótica da Faculdade de Engenharia Mecânica da Unicamp.

Conseqüentemente, através do desenvolvimento desta dissertação de mestrado foi possível a apresentação e validação de ferramentas que permitem a rápida implementação de dispositivos de controle de juntas robóticas, sendo descritos através dos seguintes capítulos:

No Capítulo 2 desta dissertação foi apresentado um trabalho de revisão bibliográfica aprofundada abrangendo o estado da arte na área de prototipagem rápida de sistemas mecatrônicos e computação reconfigurável utilizada para implementação de controladores de juntas robóticas.

No Capitulo 3 foram apresentados ambientes para prototipagem rápida e concepção de sistemas embarcados, com ênfase nos aspectos de implementação de *hardware* e *software*, com ênfase na especificação de estrutura necessária para funcionamento correto desse conceito.

No Capítulo 4 foi realizado um estudo completo enfatizando a modelagem de um sistema de acionamento e controle de uma junta robótica a partir da implementação de controladores PID expressos na forma genérica RST para posterior implementação em lógica reconfigurável, através da utilização de linguagem gráfica e da linguagem VHDL, permitindo assim, a validação e implementação experimental.

No capitulo 5, através da prototipagem de um controlador de posição de uma junta robótica baseado no conceito de Hardware In-the-Loop (HIL).

6.2 Próximas etapas e perspectivas futuras

Os resultados obtidos nesse trabalho permitem concluir que a utilização de prototipagem rápida, embora necessite de uma equipe de trabalho com muita experiência em diversas áreas de atuação dentro da Mecatrônica, se mostra muito eficiente, com redução significativa de tempo de projeto e custos inerentes a confecção de protótipos.

Assim, a principal contribuição deste trabalho de pesquisa foi apresentação de ambientes para auxílio no desenvolvimento de protótipos de sistemas embarcados, seguido de um estudo completo de concepção, desenvolvimento, e validação experimental de controladores de juntas robóticas, visando à redução significativa de custos e tempo de implementação, através da utilização mais adequada das tecnologias que compõem a área.

São aspectos promissores do ambiente proposto que sugerem uma possível continuidade desse trabalho de pesquisa enfatizando aspectos relacionados a:

- Flexibilidade: estudo de outras possíveis configurações para implementação de soluções mais aberta para diversos problemas associados com desenvolvimento de protótipos de sistemas móveis embarcados.

- Adaptação a tarefas: apenas os recursos necessários a execução de uma tarefa são utilizados. Sistemas mais simples são tratados com versões mais simples do ambiente proposto, evitando-se perda de recursos.
- Ambiente aberto: permitir que seus blocos componentes sejam totalmente acessados e eventualmente modificados para agregar novas soluções.
- Facilidade de expansão: novos recursos podem ser facilmente adicionados a um projeto, permitindo a adição de novas funcionalidades.
- Implementação de interfaces de controle supervisório de vários protótipos simultaneamente utilizando LABVIEW.
- Estudo e implementação de técnicas de fusão de sensores utilizando lógica reconfigurável direcionado a robôs móveis.
- Desenvolver interfaces para operação de sistemas embarcados cooperativos, como o caso de vários robôs móveis operando em conjunto para um determinado objetivo.

Referências Bibliográficas

Acosta, G.; Tosini, M. (2001): "*A firmware digital neural network for climate prediction applications*", Proceedings of the 2001 IEEE International Symposium on Intelligent Control, ISIC '01, pp. 127 – 131, 2001.

Aihara, C.K. (2005): *"Implementação de Ferramentas para Capacitação e Pesquisa em Automação Industrial utilizando conceitos de e-learning"*, tese de doutoramento, Unicamp, 2005.

Alford, G. D. (1984): "*A low cost CAD/CAM system with simulation"*, In: Schrefler, B. A., Lewis, R. W. (Eds.). Engineering software for microcomputers. Swansea: Pineridge, 1984, pp.201-214.

Altera Corporation (1999): *"Classic EPLD Family May 1999, ver. 5 Data Sheet"*, A-DS-CLASSIC-05, 1999.

Altera Corporation (1998): *"In-System Programmability"*, in MAX Devices Application Note, 1998.

Altera Corporation (2005): *"MAX 7000 Programmable Logic Device Family Data Sheet"*, DS-MAX7000-6.7, 2005.

Altera Corporation (2007): *"Quartus II Development Software Handbook v7.2"* , Disponível na internet via www., URL: http://www.altera.com ,2007.

Altera Corporation (2007): *"Stratix III Device Family Overview"*, Disponível na internet via www., URL: http://www.altera.com , SIII51001-1.3 , 2007.

Altera Corporation (1998): *"MAX 7000 Programmable Logic Family Data Sheet"*, Disponível na internet via www., URL: http://www.altera.com, 1998.

Altera Corporation (2008): *"Introduction to the Quartus II Software"*, Disponível na internet via www., URL: http://www.altera.com, 2008.

Aporntewan, C.: Chongstitvatana, P. (2001): *"A Hardware Implementation of the Compact Genetic Algorithm"*. Proceedings of the IEEE Congress on Evolutionary Computation, pp. 624 – 629, 2001.

Associação Brasileira de Normas Técnicas (1989) : *"NBR-6023 Referências bibliográficas"*. Rio de Janeiro, 1989, 19p.

Aström, K.J. and B. Wittenmark (1997): *"Computer Controlled Systems"*. Third Edition, Prentice Hall, New Jersey, 1997.

Baraza, J. C.; Gracia, j.; Gil, D.; Gil, P. J. (2002): *"A Prototype of a VHDL-Based Fault Injection Tool: Description and Application"*. Journal of System Archicteture 47, pp. 847-867, published by Elsevier Science Ltd., 2002.

Bemporad, A., Morari, M.(1999): "*Robust model predictive control: a survey. In Robustness in Identification and Control"*. A. Garulli, A. Tesi, A. Vicino (Eds.), Lecture Notes in Control and Information Sciences, Vol. 245, Springer-Verlag, pp.207-226, 1999.

Bolton, W. (1996): "*Mechatronics-Eletronic control systems in mechanical engineering",* 2ª edição, Ed. Longman, 1996, 380p.

Borenstein, J., (1995): *"Control and Kinematic Design of Multi-Degree-of-Freedom Mobile Robots with Compliant Linkage"*. IEEE Transactions on Robotics and Automation, Vol. 11, No 1, pp 21-35, February 1995.

Borenstein, J., Everett H. R. and Feng L. (1996): *"Sensors and Methods for Mobile Robot Positioning"*. University of Michigan, 1996.

Boucher P., Boutillon A., Carmant C., Dumur D., Foix Th., Jeandel A.,Raguenaud P. (1997): "*Automation in the Gallienne Distillery"*. 6th IEEE Conference on Control, Applications, Hartford, Octobre 1997.

Boucher P.; Dumur, D. (1997): *"Survey on Predictive Control"*. Orsay: École Supérieure D'Électricité, 1997. 37 p.

Boucher, P; Codron, P. (1993): *"Multivariable generalized prediticve control with multiple reference model: a flexible arm application"*. Sidney: 12th IFAC World Congress, 1993. 4p.

Carvajal Rojas, J.H. (2004): "*Modelagem, Simulação e Programação off-line de Robôs e Mecanismos Mecatrônicos, integrados a células de manufatura flexível"*. Unicamp, 2004.

Cassemiro, E.R., Rosário, J. M., Dumur D. (2005): "*Robot Axis Dynamics Control using a Virtual Robotics Environment"*. Preprints of IEEE International Conference on Emerging Technologies and Factory Automation- ETFA'05, pp. 305-311, Catania, Italia, 2005.

Cassemiro, E.R. (2006): "*Projeto de controlador preditivo robusto para juntas robóticas a partir de ambiente virtual de simulação e prototipagem"*. Tese de doutoramento, Unicamp, 2006.

Cassemiro, Edna Rodrigues.(2002): *"Metodologia para Desenvolvimento de Dispositivos Biomecânicos para Aplicação em Próteses Antropomórficas"*. Tese de Mestrado. Faculdade de Engenharia Mecânica, Unicamp, 103 pp., fevereiro de 2002.

Changuel, A.; Rolland, R.; Jerraya, A.A.(1996): *"Design of an Adaptive Motors Controller Based on Fuzzy Logic Using Behavioural Synthesis"*. IEEE Design Automation Conference, with EURO-VHDL '96 and European Exhibition Proceedings, EURO-DAC '96, pp. 48 – 52, 1996.

Chen, Ruei-Xi; Chen, Liang-Gee; Chen, Lilin.(2000): *"System Design Consideration for Digital Wheelchair Controller"*. IEEE Transactions on Industrial Electronics, pp. 898 – 907, Volume 47, Issue 4, 2000.

Cirstea, M.; Aounis, A.; McCormick, M.; Urwin, P.; Haydock, L.(2000): *"Induction Motor Drive System Modelled in VHDL"*. VHDL International Users Forum Fall Workshop Proceedings, pp. 113 – 117, 2000.

Cirstea, M.; Dinu, A.(2001): *"A New Neural Networks Approach To Induction Motor Speed Control"*. IEEE 32nd Annual Power Electronics Specialists Conference, PESC. 2001 IEEE, pp. 784 – 787, vol.2, 2001.

Clarke, D. W.; Mohtadi, C. (1989): *"Properties of Generalized Predictive Control"*. Automatica, Vol. 25, No. 6, p. 859-875, 1989.

Clarke, D. W.; Mohtadi, C. (1987): *"Tuffs P. S. Generalized Predictive Control - Part I. The Basic Algoritm"*. Automatica, Vol. 23, No. 2, p. 137-148, 1987.

Codron, P.; Boucher, P. (1993): *"Multivariable generalized prediticve control with multiple reference model: a new choise for the reference model"*. Groningen European Control Conference, 1993. 4p.

Codron, Pascal.(1993) : *"Commande Prédictive Multivariable"*. Orsay: École Supérieure d'Électricité, 1993. 168p. Tese (Docteur en Science).

Compton, Katherine.(2002): *"Reconfigurable Computing: A Survey of Systems and Software, Acm Computing Surveys"*. pp.171-210, vol 34, n. 2, 2002.

Coric, S.; Leeser, M.; Miller, E. (2002): *"Parallel-Bean Backprojection: An FPGA Implementation Optimized for Medical Imaging"*. FPGA´02, pp 217-226, Monterey, California, USA, fevereiro 2002.

Craig, J. J.,(1986): *"Introduction to Robotics: Mechanical & Control"*. Addison-Wesley, Publishing Company, 1986.

Daly, Alan and Marnane, William.(2002): *"Efficient Architectures for Implementing Montgomery Modular Multiplication and RSA Modular Exponentiation on Reconfigurable Logic"*. FPGA´02, pp 40-49, Monterey, California, USA, fevereiro 2002.

David, S. A.,(1996): *"Modelagem, Simulação e Controle de Robôs Flexíveis"*. Dissertação de Mestrado, Unicamp, 1996.

De Pablo, S.; Dominguez, J.A.; Lorenzo, S. Vaquero, L.J.(1997): *"A Flexible Inverter Controller for Prototypes. Proceedings of the IEEE International Symposium on Industrial Electronics"*. ISIE '97, pp. 254 – 257, vol.2 , 1997.

Dechechi, Eduardo Cesar. (1998): *"Controle avançado preditivo adaptativo "DMC" multivariável"*. Campinas: Faculdade de Engenharia Elétrica, Universidade Estadual de Campinas , 1998. 168 p. Tese (Doutorado).

Dido, J.; Geraudie, N.; Loiseau, L.; Payeur, O.; Savaria, Y. (2002): *"A Flexible Floating-Point Format for Optimizing Data-Paths and Operators in FPGA-Based DSPs"*. FPGA´02, pp 50-55, Monterey, California, USA, fevereiro 2002.

Diniz, Pedro C; Park, Joonseok.(2002): *"Data Reorganization Engines for the Next Generation of System-On-a-Chip FPGAs"*. FPGA´s 2002 Proceedings, pp. 237-244, 2002.

Dumur D., Boucher P., Murphy M., Déqué F.,(1997): *"Comfort Control in Residential Housing using Predictive Controllers"*. 6th IEEE Conference on Control Applications, Hartford, Octobre 1997.

Dumur, D.; Daumüller, S.; Boucher, P.(1992): *"Popcorn: CAD for Predictive Polynomial Control. Application to Motor Drives"*. The First IEEE Conference on Control Applications, 1992. 5p.

Dumur, Didier.(1993) : *"Commande Prédictive et Machine-Outil"*. Orsay: École Supérieure d'Électricité, 1993. 156p. Tese (Docteur en Science).

Elecro-Craft Corporation, (1977): *"DC Motors Speed Controls Servo Systems"*. Pergamon Press, 1977,

Empringham, L.; Wheeler, P.; Clare, J. (2000): *"A Matrix Converter Induction Motor Drive Using Intelligent Gate Drive Level Current Commutation Techniques"*. Conference Record of the IEEE Industry Applications Conference, pp.1936 – 1941, vol.3, 2000.

Endemano, Aitor.(2002): *"System Level Simulation of a Double Stator Wobble Electrostatic Motor. Sensors and Actuator A 99"*. pp. 312-320, published by Elselvier Science Ltd.,2002.

Fanucci, L. (2002): *"FAST: FFT ASIC Automated Synthesis"*, Integration, the VLSI journal, pp. 1-15, published by Elselvier Science Ltd., 2002.

Franklin, G. F., Powel, J. D., Workman, M. L.(1992): *"Digital control of dynamic systems"*. 2.ed. Reading: Addison-Wesley, 1992, Cap. 11: "Nonlinear Control", pp.516-640.

Garbergs, B.; Sohlberg, B.(1998): *"Implementation of a State Space Controller in a FPGA"*. 9th Mediterranean Electrotechnical Conference, MELECON 98., pp. 566 – 569, vol.1, 1998.

Garbergs, B.; Sohlberg, B.(1996): *"Specialised Hardware for State Space Control of a Dynamic Process"*. IEEE Digital Signal Processing Applications Proceedings, TENCON '96, pp. 895 – 899, vol.2, 1996.

Garcia, Carlos E.; Prett, David M.; Morari, Manfred.(1989): *"Model Predictive Control: Theory and Practice - a Survey"*. Automatica, Vol. 25, No. 3 , p. 335-348, 1989.

Harkin, J.; McGinnity, T. M.; Maguire, L. P.(2001): *"Genetic Algorithm Driven Hardware-Software Partioning for Dynamically Reconfigurable Systems"*. Microprocessors and Microsystems 25, pp. 263-274, published by Elselvier Science Ltd., 2001.

Hasuko, K.; Fukunaga, C.; Ichimiya, R.; Ikeno, M.; Ishida, Y.; Kano, H.; Kurashige, H.; Mizouchi, K.; Nakamura, Y.; Sakamoto, H.; Sasaki, O.; Tanaka, K.;(2001): *"A Remote Control System for FPGA-Embedded Modules in Radiation Environments"*. IEEE Transactions on Nuclear Science, pp. 501 – 506, 2001.

Hervella, C., (1995): *"Projeto e Desenvolvimento de um Controlador Lógico Programável Flexível para Controle de Manipuladores e Robôs Industriais"*. Dissertação de Mestrado, Unicamp, 1995.

Histand, M.B., Alciatore, D.G.,(1999): *"Introduction to Mechatronics and Measurement Systems"*. McGraw Hill, 1999.

Hu, Bao-Sheng; Li, Jing.(1996): *"The Fuzzy PID Gain Conditioner: Algorithm, Architecture and FPGA Implementation"*. Proceedings of The IEEE International Conference on Industrial Technology, ICIT '96, pp. 621 – 624, 1996.

Hwang, Yin-Tsung; Han, Jih-Cheng.(1999): *"A Novel FPGA Design of a High Throughput Rate Adaptive Prediction Error Filter"*. The First IEEE Asia Pacific Conference on ASICs, AP-ASIC '99, pp. 202 – 205, 1999.

IEEE, (2000): *"IEEE Standard VHDL Language Reference Manual*" , IEEE Std 1076, 2000 Edition, ISBN 0-7381-1949-0 SS94817, 2000.

Isermann, R. (2005): *"Mechatronic Systems – Innovative Products with Embedded Control"*. In: 3rd IFAC Symposium on Mechatronic Systems, Darmstadt, Germany, 2005, pp. 595–607.

Ito, S.A.; Carro, L.(2000): *"A comparison of microcontrollers targeted to FPGA-based embedded applications"*. IEEE 13th Symposium on Integrated Circuits and Systems Design Proceedings, pp. 397 – 402, 2000.

Jeon, Jae Wook; Kin, Yun-Ki.(2002): *"FPGA Based Acceleration and Deceleration Circuit for Industrial Robots and CNC Machine Tools"*. Mechatronics 12, pp. 635-642, published by Elselvier Science Ltd., 2002.

Kalra, Punit. (2001): *"Reconfigurable FPGAs Help Build Upgradable Systems"*. Embede Developers Journal, pp. 16-21, September 2001.

Kamm, L.J. (1996): *"Understanding Electro-Mechanical Engineering: An Introduction to Mechatronics"*. IEEE Press, 1996.

Kean, Tom. (2000): *"Cryptographic Rights Management of FPGA Intellectual Property Cores"*. FPGA´s 2000 Proceedings, pp. 113-118, 2000.

Kharrat, M.W. Masmoudi; N. Ghariani, M. Kamoun, L. (1998): *"A Digital Control System Based on Codesign Technology"*. Proceedings of the Tenth International Conference on Microelectronics, ICM '98, pp. 257 – 261, 1998.

Khriji, Lazhar; Benacchia, Giuseppe; Gabbouj, Moncef; Sicuranza, Giovanni. (1999): *"A Dedicated Hardware System for a Class of NonLinear Order Statistics Rational Hybrid Filters with Applications to Image Processing"*. IEEE 1999.

Klotchkov, I. V.; Pedersen, S. (1996): *"A Codesign Case Study: Implementing Arithmetic Functions in FPGA´s"*. IEEE 1996.

Kobayashi, F. Haratsu; M..(1995): *"A Digital Pll with Finite Impulse Responses"*. IEEE International Symposium on Circuits and Systems, ISCAS '95, pp. 191 – 194, vol. 1, 1995.

Kollig, P.; Al-Hashimi, B.; Abbott, K. M.(1996): *"Design and Implementation of Digital Systems for Automatic Control Based on Behavioural Descriptions"*. IEEE 1996.

Kollig, P.; Al-Hashimi, B.M.; Abbott, K.M. (1997): *"Efficient Scheduling of Behavioural Descriptions in High-Level Synthesis"*. IEEE Proceedings- Computers and Digital Techniques, pp. 75 – 82, 1997.

Köster, Markus; Teich, Jürgen.(2002): *"(Self-)Reconfigurable Finite State Machines: Theory and Implementation"*. Proceedings of the 2002 Design, Automation and Test in Europe Conference and Exhibition (DATE´2002). IEEE 2002.

Kozlowski, T.; Dagless, E.L.; Saul, J.M.; Adamski, M.; Szajna, J. (1995): *"Parallel Controller Synthesis Using Petri Nets"*. IEEE Proceedings - Computers and Digital Techniques, IEE Proceedings, pp.263 – 271, 1995.

Lima, C.R.E.,(2003): *"Proposta de Ambiente Baseado em Computação Reconfigurável para aplicação em Protótipos de Sistemas"*. Tese de Doutorado, Campinas, Brasil, 2003.

Lima, C.R.E., Rosário, J.M.,(1999): *"Um Controlador por Comparação de Pulsos Implementado em PLD"*. Revista Robótica, Portugal, 1999.

Lima, Carlos R. E.; Rosário, João M.,(2001): *"Utilização de Circuitos Lógicos Programáveis para Controle de Atuadores em Robôs Móveis"*. Revista Robótica, Portugal, 2001.

Lima, R. C. E, Silva; N. C., Rosário, J. M., (2000): *"A Proposal of Flexible Architecture for Mobile Robotics"*. In Mechatronics 2000 – The 7th Mechatronics Forum International Conferênce and Mechatronics Education Workshop, 6 a 8 de setembro de 2000. Atlanta, 6p.

Limebeer, D.J.N., Kasenally, E.M, Perkins, J.D., (1993): *"On the design of robust two degree of freedom controllers"*. Automatica, 29(1), pp. 157-168, 1993.

Machado, J. A. T., (1988): *"Gestão dos Recursos Estruturais no Controlo de Robots Manipuladores"*. Tese de Doutorado, Universidade do Porto, 1988.

Maia, Maria de Lourdes Oliveira. (1994): *"Controle Preditivo de uma Coluna de Absorção"*. Campinas: Faculdade de Engenharia Química, Universidade Estadual de Campinas, 1994. 94 p. Tese (Mestrado).

Melo, Leonimer, F (2007): *"Proposta de Simulador Virtual para Sistema de Navegação de Robôs Móveis Utilizando Conceitos de Prototipagem Rápida"*. tese de doutoramento, UNICAMP, 2007.

Mentor Graphics Corporation. (2002): Disponível na internet via www, URL: http://www.mentor.com, 2002.

Miyazaki, T. (1998): *"Reconfigurable Systems: a Survey"*. IEEE Proceedings of the Design Automation Conference 1998, ASP-DAC '98, pp. 447 - 452 , 1998.

Monteiro, Fabrice; Dandache, Abbas; M´Sir, Amine; Lepley, Bernard. (2001): *"A Fast CRC Implementation on FPGA Using a Pipelined Architecture for the Polynomial Division"*. IEEE 2001.

Navabi, Z.; Day, S.(1991): *"Tutorial on Use of VHDL for Description of Digital Systems"*. Fourth Annual IEEE International ASIC Conference and Exhibit Proceedings, pp. T12 - 1/1-8, 1991.

Nomadic Technologies, (1999): *"XR4000 Mobile Robot"*. Disponível na Internet via www, URL: http://www.robots.com/products.htm, 1999.

Ogata, K., (1985): *"Engenharia de Controle Morderno"*. Editora Prentice/Hall do Brasil Ltda. Rio de Janeiro, 1985.

Oliveira, Gustavo Henrique da Costa. (1997): *"Controle Preditivo para Processos com Incertezas Estruturadas Baseado em Séries de Funções Ortonormais"*. Campinas: Faculdade de Engenharia Elétrica, Universidade Estadual de Campinas, 1997. 151 p. Tese (Doutorado).

Ollero, A., Boverie, S., Goodal, R. (2005): *"Mechatronics, Robotics and Components for Automation and Control"*. Annual Reviews in Control IFAC Journal, No 26, 2005, pp 203-228.

Page, Ian. (1996): *"Reconfigurable Processor Archictetures"*. Microprocessors and Microsystems 20", pp. 185-196, published by Elselvier Science Ltd., 1996.

Pape, David A.(1993): *"A modal analysis approach to flaw detection in ceramic insulators. In: International Modal Analysis Conference"*. 11, 1993, Kissimmee. Proceedings. Bethel: Society for Experimental Mechanics, 1993, v.1, pp.35-40.

Paul, R. P.,(1986): *"Robot Manipulators: Mathematics, Programing, and Control"*. The M.I.T. Press, 1986.

Peiró, Marcos M.; Valls, Javier; Sansaloni, T.; Pascual, A. P.; Boemo E. I.(1999): *"A Comparation Between Lattice, Cascade and Direct-Form FIR Filter Structures by Using a FPGA Bit-Serial Distributed Arithmetic Implementation"*. IEEE 1999.

Perry , Douglas L.(2002): *"VHDL: Programming by Example"*. McGraw-Hill, Fourth Edition 2002.

Pimentel, Júlio C. G.; Le-Huy, Hoang.(2000): *"A VHDL-Basead Methodology to Develop High Performance Servo Drivers"*. Industry Applications Conference. Conference Record of the 2000 IEEE, pp. 1505 – 1512, vol.3, 2000.

Radiometrix Ltd, (2002): *"Low Power UHF Data Tranceiver Module"*. Disponível na Internet via www, URL: http://www.radiometrix.co.uk, 2002.

Rathna, G. N.; Nandy, S. K.; Parthasarathy, K.(1994): *"A Methodology for Architecture Synthesis of Cascaded IIR Filters on TLU FPGAs"*. 7th International Conference on VLSI Design - IEEE, pp. 225-228, January, 1994.

Renner, F.M.; Hoffmann, K. J.; Markert, R.; Glesner, M. (2002): *"Desing Methodology of Application Specific Integrated Circuit for Mechatronic Systems"*. Microprocessors and Microsystems, pp. 95-102, published by Elselvier Science Ltd., 2002.

Richalet, Jacques (1993): *"Commande predictive"*. Orsay: École Supérieure D'Électricité, 1993. 140p.

Richalet, Jacques.(1997): "Initiation à la commande préditive". Orsay: École Supérieure D'Électricité, 1997. 42p.

Rosário, J.M., Oliveira, C., Sa, C.E.A.,(2002): *"Proposal Methodology for the Modeling and Control of Manipulators"*. International Journal of the Brazilian Society of Mechanical Engineering, Vol. XXIV – N 3, Julho 2002.

Rosário, J.M.,(2004): *"Princípios de Mecatrônica"*. Pearson Prentice Hall do Brasil, 360 pp, 2004, ISBN: 85-7605-010-2.

Rutenbar, R.A. (2001): *"(when) will FPGAs kill asics?"*. IEEE Design Automation Conference Proceedings, 2001. Proceedings, pp. 321 - 322 , 2001.

Samet, L. Masmoudi; N. Kharrat, M.W. Kamoun, L.(1998): *"A digital PID Controller for Real Time and Multi Loop Control: a Comparative Study"*. IEEE International Conference on Electronics, Circuits and Systems, pp. 291 – 296, vol.1, 1998.

Saramago, Marcos Antonio Porta.(2002): *"Elaboração de Dispositivos Inteligentes Utilizando Conceitos de Domótica Direcionados a Automação Hospitalar"*. Faculdade de Engenharia Mecânica, Unicamp, 224pp, agosto de 2002.

Silva, Jose Edson Lima e. (1997): *"Simulação e Controle Preditivo Linear (com modelo de convolução) e Não-linear (com modelo baseado em redes neurais artificiais) de Colunas Recheadas de Absorção com Reação Química"*. Campinas Faculdade de Engenharia Química, Universidade Estadual de Campinas, 1997. 169 p. Tese (Doutorado).

Silva, N. C., Rosário, J. M., Lima, C. E., (2000): *"Actuator Selection for Antropomorphic Robots Using Power and Heating Rate"*. In Mechatronics 2000 – The 7th Mechatronics Forum

International Conferênce and Mechatronics Education Workshop, 6 a 8 de setembro de 2000. Atlanta, 5 p.

Silva, N. C., Rosário, J. M., Silva; Lima, C. E., (2000): *"Computer Aided Choice for DC Motors of Antromorphics Robots"*. Mechatronics 2000 – 1st IFAC-Conferênce Mechatronic System, 18 a 20 de setembro de 2000. Darmstadt – Germany, 5 p. 2000. 519 a 522 pre prints.

Snider, Greg; Shackleford, Barry; Carter, Richard J.(2001): *"Attacking the Semantic Gap Between Application Programming Languages and Configurable Hardware"*. FPGA´s 2001 Proceedings, pp. 115-124, 2001.

Soeterboek, R.(1992): *"Predictive Control: A Unified Approach"*. Cambridge: Prentice Hall International, 1992. 352 p.

Souza, J. P. et al (2000): *"Simulation and Experimental Result on Predictive Control of Robotic Joints"*. Orsay : École Supérieure D'Électricité, 2000.

Souza, Jocarly Patrocínio de., (2001): *"Implementação de Algoritmos Preditivos para Controle de Juntas Robóticas"*. Tese de Doutorado. Faculdade de Engenharia Mecânica, Unicamp, 130 pp., março de 2001.

Spong, M.W., Vidyasagar, M.,(2002): *"Robot Dynamics and Control"*. John Wiley & Sons, New York, 1989.

Swaminathan, S.; Tessier, R.; Goeckel, D.; Burleson, W. (2002): *"A Dynamically Reconfigurable Adaptative Viterbi Decoder"*. FPGA´02, pp 227-236, Monterey, California, USA, fevereiro 2002.

Synopsys, Inc.(2002): Disponível na internet via www, URL: http://www.synopysis.com, 2002.

Texas Instruments,(1992): *"Standard High-speed PAL Circuits"* , Disponível na internet via www, URL: http://www.ti.com, 1992.

Thor, Jonas; Akos, Denis M.(2002): *"A Direct RF Sampling Multifrequency GPS Receiver"*. IEEE, 2002.

Triscend Corporation., (2002): Disponível na internet via www, URL: http://www.triscend.com/, 2002.

Valdes, M.D.; Nogueiras-Melendez, A.A.; Moure, M.J.; Mandado, E.(1998): *"An Alternative Solution for the Implementation of Configurable Interfaces Oriented to Microprocessor-Based Control Systems"*. IEEE International Symposium on Industrial Electronics Proceedings, ISIE '98, pp. 643 – 647, vol.2, 1998.

Valls, J.; Peiro, M.M. ;Sansaloni, T. ;Boemo, E.(1998): *"A Study About FPGA-Based Digital Filters"*. IEEE Workshop on Signal Processing Systems, SIPS 98, pp. 192 – 201, 1998.

Vargas, F.L.; Fagundes, R.D.R. Junior, D.B. (2001): *"A FPGA-Based Viterbi Algorithm Implementation for Speech Recognition Systems"*. Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, pp. 1217 – 1220, vol.2 2001.

Wegrzyn, Marek; Adamski, Marian A., Monteito, Joao. L.(1998): *"The Application of Reconfigurable Logic to Controller Design"*. Control Enginnering Practice 6, pp. 879-887, published by Elselvier Science Ltd., 1998.

Woerner, D.F.; Lehman, D.H.(1995): *"Faster, better, cheaper technologies used in the attitude and information management subsystem for the Mars Pathfinder mission"*. Aerospace Applications Conference, 1995. Proceedings., IEEE , pp. 155 - 167 vol.2 , 1995.

Xilinx Technology, Inc, (1998): *"XC3000 Series Field Programmable Gate Arrays"*. Data Sheet, November 9, 1998 (Version 3.1).

Xilinx Technology, Inc.(2002): Disponível na internet via www, URL: http://www.xilinx.com, 2002.

Yamaguchi, T.; Hashiyama, T.; Okuma, S.(2000): *"A Study on Reconfigurable Computing System for Cryptography"*. IEEE International Conference on Systems, Man, and Cybernetics, pp. 2965 – 2968, vol.4, 2000.

Yasunaga, M.; Kin, J. H.; Yoshihara, I. (2000): *"The Application of Genetic Algorithms to the Desing of Reconfigurable Reasoning VLSI Chips"*. FPGA´s 2000 Proceedings, pp. 118-125, 2000.

Ye, Zhi Alex; Sehnoy, Nagaraj; Banerjee, Prithviraj. (2000): *"A C Compiler for a Processor with a Reconfigurable Functional Unit"*. FPGA´s 2000 Proceedings, pp. 95-100, 2000.

Zelenovsky, Ricardo; Mendonça Alexandre.(1999): *"PC: Um Guia Prático de Hardware e Interfaceamento"*. 762 pp. Editora Mz Ltda., 2a Edição, 1999.

Anexo A – Controladores Preditivos

A.1 - Aspectos gerais

A grande flexibilidade dos sistemas digitais tem facilitado o desenvolvimento de novas estratégias de controle digital. Entre estas novas estratégias encontram-se os controladores preditivos. Os controladores preditivos foram introduzidos por [Richarlet, 1978], [Cutler e Ramaker, 1979] e [Garcia et al., 1989]. Os trabalhos subseqüentes sobre este tema são muitos, sendo que alguns apresentam resultados de aplicações práticas em modelos industriais ou de laboratório. Entre as áreas de aplicação destes controladores encontram-se o controle de processos químicos e o controle de elementos eletro-mecânicos, desde simples motores até manipuladores robóticos e robôs móveis.

Os controladores preditivos trabalham com o princípio de antecipação da resposta do sistema. Devido a sua natureza, os controladores preditivos podem ser utilizados em processos simples ou em processo mais complexos, com atrasos de transporte, de fase não mínima, não lineares e ainda com limitações de resposta. Em função da complexidade do processo, consta na literatura comparações entre o controlador preditivo e outros controladores como PID (controlador proporcional, integral e derivativo), como controladores por alocação de pólos e controle linear quadrático. Dependendo do processo os controladores preditivos são equivalentes aos controladores comparados, mas existem duas características únicas que os classificam como os mais adequados para o controle de determinados processos:

- Em contraste com controle linear quadrático e controle por alocação de pólos podem também ser derivados para processos não lineares.

- É a única metodologia que pode manipular restrições do processo durante o projeto do controlador. Uma vez que restrições do processo é uma prática comum, isto se torna um ponto importante. Esta é considerada uma das mais importantes características do controlador preditivo.

Outras características dos controladores preditivos são:

- O conceito de controle preditivo não esta restrito a processos SISO. Controladores preditivos podem ser usados em processos MIMO.
- Controle preditivo é uma metodologia aberta. Isto é, existem muitas maneiras de se projetar o controlador preditivo. Alguns controladores preditivos conhecidos são: GPC (*Generalized Predictive Controller*), DMC (*Dynamic Matrix Control*), EPSAC (*Extended Prediction Self-Adaptative Control*), PFC (*Predictive Functional Control*), EHAC (*Extended Horizon Adaptative Control*) e UPC (*Unified Predictive Control*) (Clark et al., 1989) (Soeterboek, 1992).
- Uma vez que controladores preditivos pertencem à classe de métodos de projeto de controladores baseados em modelos, um modelo do processo deve estar disponível.

Os controladores preditivos são classificados dentro dos controladores baseados em modelos. Um modelo do processo é usado para projetar o controlador. A **Figura** ilustra este processo.

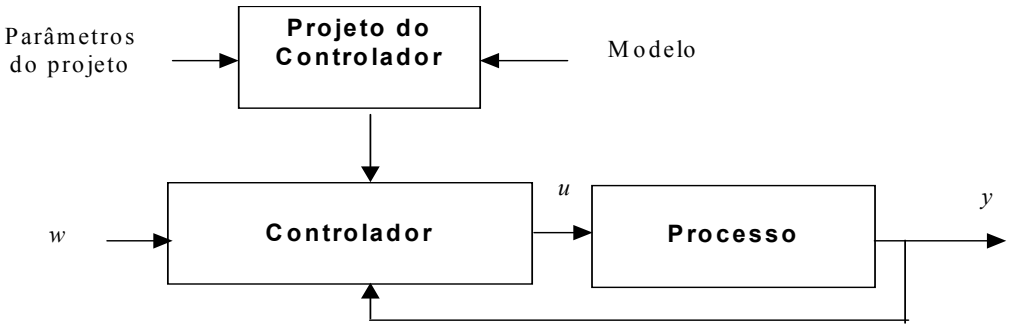


Figura A1 - Controle baseado em modelo de processo.

Usualmente, controladores preditivos são usados em tempo discreto. Contudo é possível projetar controladores preditivos para uso em tempo contínuo. O modo como controladores preditivos operam em sistemas SISO é ilustrado na **Figura** . As escalas de tempo nos gráficos são relativas ao tempo discreto k. As escalas de tempo mostradas na figura são escalas de tempo absolutas. Além disto, u(k), y(k) e w(k) denotam a saída do controlador, a saída do processo e a saída desejada do processo na amostra k, respectivamente.

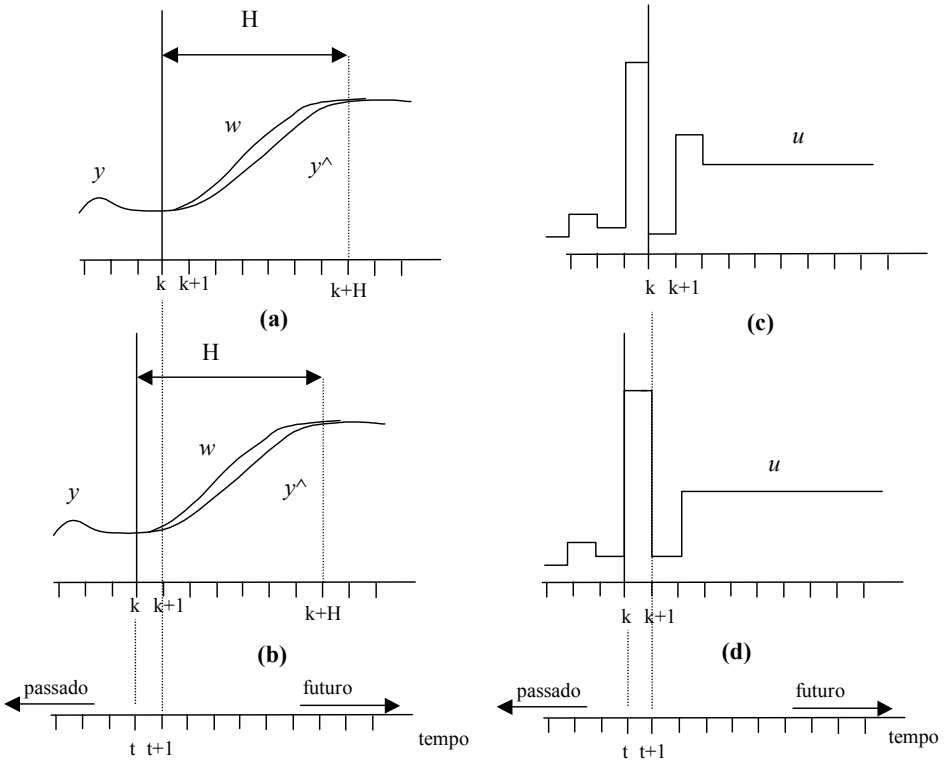


Figura A2 - Representação do princípio de operação dos controladores preditivos.

Agora é definido:

$$\underline{u} = [u(k), u(k+1)\ldots,u(k+H-1)] \quad (A.1)$$

$$\underline{\hat{y}} = [\hat{y}(k+1),\ldots,\hat{y}(k+H)] \quad (A.2)$$

$$\underline{w} = [w(k+1),\ldots,w(k+H)] \quad (A.3)$$

Onde H é o horizonte de predição e o símbolo $\hat{}$ denota estimação. Então o controlador

preditivo calcula uma seqüência u de saídas futuras do controlador (Figura A2d) tal que a saída preditiva do processo $\underline{\hat{y}}$ é próxima a saída desejada do processo w (Figura A2b). Esta saída desejada do processo é usualmente chamada de trajetória de referência e pode ser uma seqüência arbitrária de pontos.

Em lugar de usar a seqüência de saída do controlador determinada sobre o método de controlar o processo nas próximas H amostras, somente o primeiro elemento desta seqüência de saída do controlador u(k) é usado para controlar o processo. Na próxima amostra (em t+1), o procedimento é todo repetido, usando a última informação medida. Isto é chamado princípio do retrocesso do horizonte [Soeterboek, 1992] e é ilustrado na Figura A2 e Figura A2c, as quais mostram o que acontece no tempo t+1.

Assumindo que não há perturbações e que não existem erros de modelamento do processo a saída do processo preditivo $\hat{y}$ (k+1), predito no tempo t, é exatamente igual à saída do processo y(k), medida em t+1. Em geral, a seqüência de saída é diferente de uma seqüência obtida em uma amostra prévia, como ilustrado na Figura A2c. A razão para utilizar o princípio do retrocesso do horizonte é que isto permite compensar futuras perturbações ou erros de modelamento. Como resultado do princípio de retrocesso do horizonte, o horizonte sobre qual a saída do processo é predita é deslocado uma amostra no futuro a cada instante de amostragem.

A saída do processo é predita pelo uso do modelo do processo a ser controlado. Qualquer modelo que descreve um relacionamento entre entrada e saída do processo pode ser usado: modelos de função transferência, modelos de resposta ao degrau, modelos de espaço de estados e modelos não-lineares [Boucher et al., 1993] [Maia, 1994][Oliveira, 1997]. Se o processo é sujeito a perturbações, um modelo de perturbação ou ruído pode ser adicionado ao modelo do processo, permitindo que o efeito destas perturbações seja levado em conta.

De modo a definir quão bem a saída do processo preditivo segue a trajetória de referência, uma função de custo é usada. Por exemplo, uma função de custo simples é [Maia, 1994]:

$$J = \sum_{i=0}^{N} (\hat{y}(k+1) - w(k+1))^2 \qquad \text{(A.4)}$$

O controlador miniminiza esta função de custo em relação à saída u do controlador, obtendo valores ótimos para esta saída. O erro é minimizado dentro do horizonte de predição.

Calcular a seqüência de saída do controlador é um problema de otimização, ou mais especificamente, um problema de minimização. Usualmente, resolver um problema de minimização requer um procedimento interativo. Porém, uma solução analítica está disponível quando o critério é quadrático, o modelo é linear e invariante no tempo e não existem restrições. Idealmente, uma função de custo é baseada em especificações de projeto. Porém, pelo uso de uma função de custo qualquer, o problema de otimização pode ser difícil de resolver. Esta é a razão pela qual uma função de custo quadrática é usada em todos os controladores preditivos. Em função do controlador utilizado, adaptações são feitas nesta função de custo quadrática.

As especificações de projeto devem ser transladadas em parâmetros da função de custo quadrático, de modo que, quando esta função é minimizada, as especificações originais do projeto são atendidas (Soeterboek, 1992).

A.2 Relacionamento com Outros Métodos

Os controladores preditivos, controladores lineares quadráticos e controladores por alocação de pólos pertencem à classe de controladores baseados em modelo. Mais exatamente, controladores preditivos e lineares quadráticos são baseados na minimização de uma função de custo.

A.2. 1 Controle Linear Quadrático

Uma discussão com respeito ao controlador quadrático discreto pode ser encontrada em [Soeterboek, 1992]. Alguns dos métodos lineares quadráticos são baseados em realimentação de estados, enquanto outros são baseados em realimentação de saída.

Usando o enfoque de realimentação de estados, em controladores lineares quadráticos discretos o processo é dado por:

$$x(k+1) = \mathbf{A}x(k) + \mathbf{b}u(k) \qquad (A.5)$$

$$y(k) = \mathbf{C}^T x(k), \qquad (A.6)$$

onde x(k) denota o estado do processo e **A**, **b**, **C** são os parâmetros do processo.

Similar ao projeto de controladores preditivos, uma função de custo é minimizada em relação a um horizonte. Contudo, em contraste com controladores preditivos, o princípio do retrocesso do horizonte não é empregado. Esta é a diferença fundamental. A função de custo é minimizada apenas uma vez (em k = 0), resultando uma saída ótima para o controlador a partir do qual o controle de processo é tomado a cada amostra. Assim, futuras perturbações e erros de modelagem não são levados em conta.

Em conclusão pode ser dito que, para um horizonte finito de predição, a diferença fundamental entre controle preditivo e o controle linear quadrático é o princípio do retrocesso do horizonte, com um horizonte de tamanho fixo empregado pelo controle preditivo em contraste com o horizonte de predição de tamanho decrescente do controlador linear quadrático. Como resultado disto, existem as sequintes diferenças fundamentais entre os dois controladores:

- Devido ao fato da estrutura do controlador linear quadrático ser determinada previamente, eventuais restrições na saída do controlador não podem ser consideradas.
- Os projetos de controladores lineares quadráticos consideram apenas processos linearizados, enquanto que controladores preditivos permitem o uso de modelos não lineares para predição da saída.
- Uma desvantagem importante do controle linear quadrático é que, em geral, é bastante difícil traduzir as especificações de projeto em matrizes de peso na função de custo. Isto ocorre porque os estados de um sistema discreto são artificiais e não estão diretamente relacionados com os estados reais do processo. Regras práticas para a escolha dos parâmetros de projeto não estão prontamente disponíveis. Uma vantagem importante dos controladores lineares quadráticos de horizonte finito é que, sobre condições quase gerais, o sistema em malha fechada é sempre estável. Esta alegação não pode ser usualmente feita sobre os controladores preditivos de horizonte finito [Soeterboek, 1992].

A.2.2 Projeto por Alocação de Pólos

Outro projeto de controlador baseado em modelos que é freqüentemente usado para o projeto de controladores discretos é o método de projeto por alocação de pólos. Uma discussão com respeito ao projeto por alocação de pólos pode ser encontrada em [Soeterboek, 1992], [Oliveira, 1997] e [Souza, 2000]. Neste método a saída do controlador é generalizada pela lei de controle linear:

$$Ru(k) = - Sy(k) + TSp, \qquad (A.7)$$

onde R, S e T são polinômios do controlador no operador q^{-1} e Sp denota o *set-point*. Os controladores polinomiais R e S são selecionados pelos pólos em malha fechada desejados e T é selecionado em função do comportamento do sistema em malha fechada a variações no *set-point*.

A maior dificuldade deste método é decidir o posicionamento dos pólos em malha fechada. Ou seja, como transladar as especificações de projeto (geralmente fornecidas no domínio do tempo) em localização de pólos e zeros em malha fechada.

Somente em casos de sistemas de baixa ordem sem zeros é possível fazer isto facilmente. Além disto, unido ao fato de que as saídas do controlador são geradas por (A.7), restrições a estas mesmas saídas e uma trajetória de referência arbitrária não podem ser adicionadas ao sistema.

Em contraste com o controle preditivo, este método pode apenas ser aplicado em processos lineares (ou linearizados). Apesar disto, em muitas publicações o projeto por alocação de pólos também é discutido junto com o projeto de controladores preditivos [Clarke et al., 1987] [Soeterboek, 1992] [Oliveira, 1997].

O artigo de [Clarke et al., 1987] apresenta, pela primeira vez, uma descrição do projeto de controladores preditivos generalizados (GPC). Posteriormente é feita uma comparação (por simulação de um sistema SISO) com outros controladores, como o controlador por alocação por pólos e o controlador linear quadrático. Este trabalho é freqüentemente referenciado por trabalhos posteriores. No trabalho de [Clarke et al., 1989] discute-se detalhadamente as propriedades dos controladores preditivos generalizados. São apresentados exemplos práticos do uso do GPC, como em manipuladores robóticos, em sistemas de aquecimento e em secadores, entre outros.

[García et al., 1989] apresenta um resumo comparativo de diversos métodos de controle preditivo e outros métodos tradicionais, mostrando ainda um interessante resumo histórico da matéria, chegando a discutir controladores preditivos não lineares.

A proposta de uma teoria unificada no projeto de controladores preditivos é feita por [Soeterboek, 1992]. Devido ao fato do conceito de controle preditivo ser uma metodologia aberta, muitos controladores preditivos podem ser obtidos durante um projeto, cada um com características próprias. Embora, a primeira vista, a diferença entre estes controladores tenda a parecer pequena, a mesma pode provocar comportamentos muitos diferentes do sistema em malha fechada. Como resultado, pode ser um tanto difícil determinar qual dos controladores preditivos pode ou deve ser usado. Visando resolver este problema, uma abordagem unificada do projeto de controladores preditivos é apresentada, permitindo o tratamento de cada problema dentro de uma mesma estrutura, reduzindo significativamente os custos de projeto. O controlador preditivo unificado unifica conhecidos controladores preditivos, como GPC, DMC, EPSA e EHAC. Conseqüentemente, outra vantagem da abordagem unificada proposta no livro é que, uma vez que este controlador preditivo unificado é analisado, conclusões podem ser tiradas a respeito dos controladores preditivos acima mencionados.

O trabalho de [Richalet, 1993] explora detalhadamente o conceito do controlador preditivo funcional (PFC - Prédictive Functional Control). Em um trabalho posterior [Richalet, 1997], é apresentado um resumo, onde podem ser observados os princípios básicos do controle preditivo.

O trabalho de [Boucher et al., 1997] apresenta uma introdução e idéias gerais do controle preditivo. São considerados as linhas de trabalho de Clark, com o controle preditivo generalizado [Clark et al. , 1987] [Clark et al., 1989] e de Richalet [Richalet, 1993]. São apresentados resultados práticos do controle de uma planta de destilação de conhaque.

O trabalho de [Maia, 1994] é um exemplo de aplicação de controle preditivo a um processo químico. No caso foi apresentado um controlador DMC (Dynamic Matrix Control) controlando uma coluna de absorção. O trabalho utiliza a função de custo simplificada:

$$J = \sum_{i=0}^{N} (y\hat{}(k+1) - w(k+1))^2 \qquad (8)$$

O modelo do processo é gerado segundo a utilização do modelo de convolução. A identificação dos coeficientes do modelo gerado é feito *off-line.*

Ainda dentro da industria química, foi apresentado o trabalho de [Maia, 1994][Silva, 1997], no qual o modelo linear do processo também foi gerado por modelo de convolução. Adicionalmente, foi gerado um modelo não linear através da utilização de redes neurais.

O trabalho intitulado Controle Preditivo Generalizado Aplicado a Sistemas Flexíveis foi apresentado por (Oliveira, 1997), sendo um trabalho de simulação sobre uma junta isolada de um manipulador robótico. São apresentados os resultados de simulação de controladores por alocação por pólos e do controlador preditivo GPC. A função de custo utilizada para o GPC é:

$$J = \sum_{i=N1}^{NY} [P(q^{-1})\hat{y}(k+i)] - D(q^{-1})w(k+i)]^2 + \sum_{i=1}^{NU} \rho \Delta u(k+i-1)^2 \qquad (A.9)$$

Onde são considerados parâmetros adicionais em relação à (3.1)como:

N1 - horizonte inicial de predição.

NY - horizonte final de predição.

NU - horizonte de controle.

ρ - constante de ponderação do sinal de controle.

$P(q^{-1})$ e $Q(q^{-1})$ são polinômios de ponderação que filtram os sinais de saída e da referência.

É apresentado um estudo sobre a robustez do sistema utilizando um controlador preditivo generalizado. Uma contribuição sobre a análise dos parâmetros de projeto do GPC para o controle de um sistema flexível é feita. Parâmetros como horizonte de predição e período de amostragem são considerados. Os trabalhos de (Clarke et al., 1987) e de (Soeterboek, 1992) são importantes na compreensão do texto.

Em [Dechechi, 1998] é proposto um trabalho com o controlador DMC aplicado a sistemas multivariáveis. O modelo do processo sofre um enfoque adaptativo.

Outro trabalho abordando a utilização do controle preditivo para sistemas multivariáveis, mas com uma nova abordagem para o modelo de referência é apresentado em [Codron, 1993]. Este trabalho descreve um algoritmo do controlador GPC com modelos de múltipla referência, com posterior análise da robustez destes algoritmos. São apresentados estudos da atuação destes controladores em modelos dinâmicos de um helicóptero e de um braço flexível. Dois trabalhos diretamente associados ao tema de escolha de modelos de referência para controladores preditivos são apresentado em [Boucher et al., 1993] e [Codron et al., 1993].

O controlador preditivo generalizado aplicado em máquinas ferramentas por comando numérico é apresentado em [Dumur, 1993]. Neste trabalho é detalhada a síntese do GPC segundo um controlador equivalente do tipo R, S e T. Esta estrutura polinomial facilita o tratamento numérico do controlador e é também explorada nos trabalhos de [Soeterboek, 1992], [Oliveira, 1997], [Dumur et al., 1992] e [Souza, 2000].

O trabalho de [Souza, 2000] compara o desempenho do controlador GPC com métodos de controle clássico para sistema SISO, a exemplo de controladores PID. Usa para tanto um modelo de uma junta robótica, analisando a resposta do sistema para diferentes padrões de excitação.

Anexo B – Código implementado em linguagem VHDL

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_signed.all;
ENTITY RST_contr_q15 IS
	PORT
	(
		clk                          : IN    STD_LOGIC;
		In_W,In_Y                    : IN    STD_LOGIC_VECTOR(15 DOWNTO 0);
		Coef1_T                      : IN    STD_LOGIC_VECTOR(15 DOWNTO 0);
		Coef2_T                      : IN    STD_LOGIC_VECTOR(15 DOWNTO 0);
		Coef3_T                      : IN    STD_LOGIC_VECTOR(15 DOWNTO 0);
		Coef4_T                      : IN    STD_LOGIC_VECTOR(15 DOWNTO 0);
		Coef5_T                      : IN    STD_LOGIC_VECTOR(15 DOWNTO 0);
		Coef6_T                      : IN    STD_LOGIC_VECTOR(15 DOWNTO 0);
		Coef7_T                      : IN    STD_LOGIC_VECTOR(15 DOWNTO 0);
		Coef8_T                      : IN    STD_LOGIC_VECTOR(15 DOWNTO 0);
		Coef9_T                      : IN    STD_LOGIC_VECTOR(15 DOWNTO 0);
		Coef1_S                      : IN    STD_LOGIC_VECTOR(15 DOWNTO 0);
		Coef1_R                      : IN    STD_LOGIC_VECTOR(15 DOWNTO 0);
		Coef2_R                      : IN    STD_LOGIC_VECTOR(15 DOWNTO 0);
		Coef3_R                      : IN    STD_LOGIC_VECTOR(15 DOWNTO 0);
		Out_U                        : BUFFER       STD_LOGIC_VECTOR(15 DOWNTO 0)
	);
END RST_contr_q15;
ARCHITECTURE a OF RST_contr_q15 IS
	TYPE data IS ARRAY(0 TO 9) OF STD_LOGIC_VECTOR(15 DOWNTO 0);
	TYPE data32 IS ARRAY(0 TO 9) OF STD_LOGIC_VECTOR(31 DOWNTO 0);

	SIGNAL Wt,Rt,St :     data; -- (Delay Line z-n)

	SIGNAL Out_T,Out_R,In_S : STD_LOGIC_VECTOR(15 DOWNTO 0);

BEGIN


-- Implementação dos atrasos das amostras Bloco T (Z-n) (T delay line)
process(clk)
begin
if clk'event and clk='1' then
	Wt(1) <= In_W;         -- z-1
	Wt(2) <= Wt(1);                -- z-2
	Wt(3) <= Wt(2);        -- z-3
	Wt(4) <= Wt(3);                -- z-4
	Wt(5) <= Wt(4);        -- z-5
	Wt(6) <= Wt(5);                -- z-6
	Wt(7) <= Wt(6);        -- z-7
	Wt(8) <= Wt(7);                -- z-8
end if;
end process;
```

```
-- Implementação dos atrasos das amostras Bloco R (Z-n) (R delay line)
process(clk)
begin
if clk'event and clk='1' then
       Rt(1) <= In_Y;         -- z-1
       Rt(2) <= Rt(1);               -- z-2
end if;
end process;




-- Implementação dos atrasos das amostras Bloco S (Z-n) (S delay line)
process(clk)
begin
if clk'event and clk='1' then
       St(1) <= Out_U;        -- z-1
end if;
end process;

-- equação saída bloco T
process(In_W,Wt,Coef1_T,Coef2_T,Coef3_T,Coef4_T,Coef5_T,Coef6_T,Coef7_T,Coef8_T,Coef9_T)
 variable aux : STD_LOGIC_VECTOR(31 DOWNTO 0);
 variable saida : std_logic_vector(15 DOWNTO 0);
begin

       aux := In_W  * Coef1_T;
       saida :=  aux(31 downto 16);          -- t0 * W(t)

       aux := Wt(1) * Coef2_T;
       saida := saida + aux(31 downto 16);   -- t0 * W(t) + t1 * W(t-1)

       aux := Wt(2) * Coef3_T;
       saida := saida + aux(31 downto 16); -- t0 * W(t) + t1 * W(t-1) + t2 * W(t-2)

       aux := Wt(3) * Coef4_T;
       saida := saida + aux(31 downto 16); -- t0 * W(t) + t1 * W(t-1) + t2 * W(t-2) + t3 * W(t-3)

       aux := Wt(4) * Coef5_T;
       saida := saida + aux(31 downto 16);

       aux := Wt(5) * Coef6_T;
       saida := saida + aux(31 downto 16);

       aux := Wt(6) * Coef7_T;
       saida := saida + aux(31 downto 16);

       aux := Wt(7) * Coef8_T;
       saida := saida + aux(31 downto 16);

       aux := Wt(8) * Coef9_T;
       saida := saida + aux(31 downto 16);

       out_T <= saida;
end process;

-- equação saída bloco R
process(In_Y,Rt,Coef1_R,Coef2_R,Coef3_R)
 variable aux : STD_LOGIC_VECTOR(31 DOWNTO 0);
 variable saida : std_logic_vector(15 DOWNTO 0);
begin

       aux := In_Y  * Coef1_R;
       saida :=  aux(31 downto 16);          -- r0 * Y(t)
```

```
        aux := Rt(1) * Coef2_R;
        saida := saida + aux(31 downto 16);   -- r0 * Y(t) + r1 * Y(t-1)

        aux := Rt(2) * Coef3_R;
        saida := saida + aux(31 downto 16); -- r0 * Y(t) + r1 * Y(t-1) + r2 * Y(t-2)

        Out_R <= saida;

end process;




-- função Soma

In_S <=   Out_T - Out_R;

-- equação saída bloco U
process(In_S,St,Coef1_S)
 variable aux : STD_LOGIC_VECTOR(31 DOWNTO 0);
 variable saida : std_logic_vector(15 DOWNTO 0);
begin
        saida := In_S ;

        aux := St(1) * Coef1_S;
        saida := saida + aux(31 downto 16);          -- s0 * In_S(t) + s1 * U(t-1)

        Out_U <=  saida;

end process;

END a;
```

Anexo C – Código implementado em linguagem VHDL (parâmetros constantes)

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_signed.all;
USE ieee.std_logic_arith .all;
ENTITY RST_contr_q15 IS
	PORT
	(
		clk                          : IN    STD_LOGIC;
		In_W,In_Y             : IN    STD_LOGIC_VECTOR(15 DOWNTO 0);
		Out_U                 : BUFFER       STD_LOGIC_VECTOR(15 DOWNTO 0)
	);
END RST_contr_q15;
ARCHITECTURE a OF RST_contr_q15 IS
	TYPE data IS ARRAY(0 TO 9) OF STD_LOGIC_VECTOR(15 DOWNTO 0);
	TYPE data32 IS ARRAY(0 TO 9) OF STD_LOGIC_VECTOR(31 DOWNTO 0);

	SIGNAL Wt,Rt,St :	data; -- (Delay Line z-n)

	SIGNAL Out_T,Out_R,In_S : STD_LOGIC_VECTOR(15 DOWNTO 0);

	SIGNAL  Coef1_T               :	STD_LOGIC_VECTOR(15 DOWNTO 0);
	SIGNAL  Coef2_T               :	STD_LOGIC_VECTOR(15 DOWNTO 0);
	SIGNAL  Coef3_T               :	STD_LOGIC_VECTOR(15 DOWNTO 0);
	SIGNAL  Coef4_T               :	STD_LOGIC_VECTOR(15 DOWNTO 0);
	SIGNAL  Coef5_T               :	STD_LOGIC_VECTOR(15 DOWNTO 0);
	SIGNAL  Coef6_T               :	STD_LOGIC_VECTOR(15 DOWNTO 0);
	SIGNAL  Coef7_T               :	STD_LOGIC_VECTOR(15 DOWNTO 0);
	SIGNAL  Coef8_T               :	STD_LOGIC_VECTOR(15 DOWNTO 0);
	SIGNAL  Coef9_T               :	STD_LOGIC_VECTOR(15 DOWNTO 0);
	SIGNAL  Coef1_S               :	STD_LOGIC_VECTOR(15 DOWNTO 0);
	SIGNAL  Coef1_R               :	STD_LOGIC_VECTOR(15 DOWNTO 0);
	SIGNAL  Coef2_R               :	STD_LOGIC_VECTOR(15 DOWNTO 0);
	SIGNAL  Coef3_R               :	STD_LOGIC_VECTOR(15 DOWNTO 0);

BEGIN


Coef1_T <= CONV_STD_LOGIC_VECTOR(0,16);
Coef2_T <= CONV_STD_LOGIC_VECTOR(13,16);
Coef3_T <= CONV_STD_LOGIC_VECTOR(32,16);
Coef4_T <= CONV_STD_LOGIC_VECTOR(52,16);
Coef5_T <= CONV_STD_LOGIC_VECTOR(72,16);
Coef6_T <= CONV_STD_LOGIC_VECTOR(91,16);
Coef7_T <= CONV_STD_LOGIC_VECTOR(111,16);
Coef8_T <= CONV_STD_LOGIC_VECTOR(131,16);
Coef9_T <= CONV_STD_LOGIC_VECTOR(150,16);
Coef1_S <= CONV_STD_LOGIC_VECTOR(23959,16);
Coef1_R <= CONV_STD_LOGIC_VECTOR(4947,16);
```

```
Coef2_R <= CONV_STD_LOGIC_VECTOR(-4876,16);
Coef3_R <= CONV_STD_LOGIC_VECTOR(570,16);


-- Implementação dos atrasos das amostras Bloco T (Z-n) (T delay line)
process(clk)
begin
if clk'event and clk='1' then
        Wt(1) <= In_W;          -- z-1
        Wt(2) <= Wt(1);                 -- z-2
        Wt(3) <= Wt(2);         -- z-3
        Wt(4) <= Wt(3);                 -- z-4
        Wt(5) <= Wt(4);         -- z-5
        Wt(6) <= Wt(5);                 -- z-6
        Wt(7) <= Wt(6);         -- z-7
        Wt(8) <= Wt(7);                 -- z-8
end if;
end process;

-- Implementação dos atrasos das amostras Bloco R (Z-n) (R delay line)
process(clk)
begin
if clk'event and clk='1' then
        Rt(1) <= In_Y;          -- z-1
        Rt(2) <= Rt(1);                 -- z-2
end if;
end process;

-- Implementação dos atrasos das amostras Bloco S (Z-n) (S delay line)
process(clk)
begin
if clk'event and clk='1' then
        St(1) <= Out_U;         -- z-1
end if;
end process;

-- equação saída bloco T
process(In_W,Wt,Coef1_T,Coef2_T,Coef3_T,Coef4_T,Coef5_T,Coef6_T,Coef7_T,Coef8_T,Coef9_T)
 variable aux : STD_LOGIC_VECTOR(31 DOWNTO 0);
 variable saida : std_logic_vector(15 DOWNTO 0);
begin

        aux := In_W  * Coef1_T;
        saida :=  aux(31 downto 16);          -- t0 * W(t)

        aux := Wt(1) * Coef2_T;
        saida := saida + aux(31 downto 16);   -- t0 * W(t) + t1 * W(t-1)

        aux := Wt(2) * Coef3_T;
        saida := saida + aux(31 downto 16); -- t0 * W(t) + t1 * W(t-1) + t2 * W(t-2)

        aux := Wt(3) * Coef4_T;
        saida := saida + aux(31 downto 16); -- t0 * W(t) + t1 * W(t-1) + t2 * W(t-2) + t3 * W(t-3)

        aux := Wt(4) * Coef5_T;
        saida := saida + aux(31 downto 16);

        aux := Wt(5) * Coef6_T;
        saida := saida + aux(31 downto 16);

        aux := Wt(6) * Coef7_T;
        saida := saida + aux(31 downto 16);

        aux := Wt(7) * Coef8_T;
        saida := saida + aux(31 downto 16);

        aux := Wt(8) * Coef9_T;
```

```
        saida := saida + aux(31 downto 16);

        out_T <= saida;
end process;

-- equação saída bloco R
process(In_Y,Rt,Coef1_R,Coef2_R,Coef3_R)
 variable aux : STD_LOGIC_VECTOR(31 DOWNTO 0);
 variable saida : std_logic_vector(15 DOWNTO 0);
begin

        aux := In_Y  * Coef1_R;
        saida :=  aux(31 downto 16);          -- r0 * Y(t)

        aux := Rt(1) * Coef2_R;
        saida := saida + aux(31 downto 16);   -- r0 * Y(t) + r1 * Y(t-1)

        aux := Rt(2) * Coef3_R;
        saida := saida + aux(31 downto 16); -- r0 * Y(t) + r1 * Y(t-1) + r2 * Y(t-2)

        Out_R <= saida;

end process;

-- função Soma

In_S <=   Out_T - Out_R;

-- equação saída bloco U
process(In_S,St,Coef1_S)
 variable aux : STD_LOGIC_VECTOR(31 DOWNTO 0);
 variable saida : std_logic_vector(15 DOWNTO 0);
begin
        saida := In_S ;

        aux := St(1) * Coef1_S;
        saida := saida + aux(31 downto 16);          -- s0 * In_S(t) + s1 * U(t-1)

        Out_U <=  saida;

end process;

END a;
```