UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA MECÂNICA

UM PROJETO DE CONTROLE DE MOVIMENTAÇÃO VEICULAR PROJETADO EM UM PROCESSADOR EMBARCADO EM FPGA COM AMBIENTE DE SIMULAÇÃO USANDO INSTRUMENTAÇÃO VIRTUAL

ANDERSON PEREIRA CORREIA

ORIENTADOR: CARLOS HUMBERTO LLANOS QUINTERO

CO-ORIENTADOR: SADEK CRISOSTOMO ABSI ALFARO

DISSERTAÇÃO DE MESTRADO EM SISTEMAS MECATRÔNICOS

PUBLICAÇÃO:
BRASÍLIA: JUNHO - 2007

UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA MECÂNICA

UM PROJETO DE CONTROLE DE MOVIMENTAÇÃO VEICULAR PROJETADO EM UM PROCESSADOR EMBARCADO EM FPGA COM AMBIENTE DE SIMULAÇÃO USANDO INSTRUMENTAÇÃO VIRTUAL

ANDERSON PEREIRA CORREIA

DISSERTAÇÃO SUBMETIDA AO DEPARTAMENTO DE ENGENHARIA MECÂNICA DA FACULDADE DE TECNOLOGIA DA UNIVERSIDADE DE BRASÍLIA COMO PARTE DOS REQUISÍTOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM SISTEMAS MECATRÔNICOS.

APROVADA POR:

_____
Prof. Dr. Carlos Humberto Llanos Quintero (ENM-UnB)
(Orientador)

_____
Prof. Dr. Ricardo Pezzuol Jacobi (CIC-UnB)
(Examinador Externo)

_____
Prof. Dr. Liu Hsu (COPPE-UFRJ)
(Examinador Externo)

BRASÍLIA/DF,

FICHA CATALOGRÁFICA

CORREIA, ANDERSON PEREIRA

UM PROJETO DE CONTROLE DE MOVIMENTAÇÃO VEICULAR PROJETADO EM UM PROCESSADOR EMBARCADO EM FPGA COM AMBIENTE DE SIMULAÇÃO USANDO INSTRUMENTAÇÃO VIRTUAL [Distrito Federal] 2007.
xvii, 178p., 210 x 297 mm (ENM/FT/UnB, Mestre, Sistemas Mecatrônicos, 2007).

Dissertação de Mestrado – Universidade de Brasília. Faculdade de Tecnologia.

Departamento de Engenharia Mecânica.

1.Automação Veícular 2.Computação Reconfigurável

3.Processador Embarcados 4. Instrumentação Virtual

I. ENM/FT/UnB II. Título (série)

REFERÊNCIA BIBLIOGRÁFICA

CORREIA., A. P. (2007). Processador Embarcado em Lógica Reconfigurável para o Controle de Movimentação de Veículo de Passeio. Dissertação de Mestrado em Sistemas Mecatrônicos **ENM.DM-14A/07**, Departamento de Engenharia Mecânica, Universidade de Brasília, Brasília, DF, 178p.

CESSÃO DE DIREITOS

AUTOR: Anderson Pereira Correia.

TÍTULO: Processador Embarcado em Lógica Reconfigurável para o Controle de Movimentação de Veículo de Passeio.

GRAU: Mestre ANO: 2007

_____

Anderson Pereira Correia

Campus Universitário Darcy Ribeiro.
Asa Norte, Gleba A, Colina Bloco K, apt 205.
CEP: 70910-900 – Brasília – DF – Brasil.

AGRADECIMENTOS

A Deus que em todos os momentos sempre esteve a me conduzir em caminhos de paz, mesmo nos momentos mais difíceis, nunca me abandonou cuidando dos menores detalhes em minha existência.

Aos meus pais Pedro e Lúcia por me educar e me ensinar o caminho da verdade.

Aos meus irmãos Peterson, Andresa e Ismael por serem pessoas com as quais aprendi muitas coisas nos mais importantes instantes de minha vida. Ao meu mais novo irmão Luiz Toledo por tudo quanto me ensinou e por sua amizade.

Ao professor Dr. Carlos Llanos por sua ajuda e orientação e sobre tudo sua paciência. Também a sua esposa Carmem por seu carinho, cuidado e os deliciosos quitutes.

Sinceros ao professor José Maurício Motta que no momento crucial me apoiou a continuar e chegar a conclusão desta etapa.

A todo o corpo docente do Departamento de Engenharia Mecânica da Universidade de Brasília em especial Marrocos por tão grande atenção e dedicação, Professor Dr. Walter Britto por seus concelhos e ao professor Dr. Sadek por sua confiança. Também a professora Dr.Carla Maria Koike por sua dedicação, apoio e colaboração.

A todo o DPP da Universidade de Brasília em especial ao senhor Nonato por sua atenção, confiança e apoio.

Finalmente a todos os colegas pela paciência, atenção e grande ajuda.

Ao Centro Nacional de Pesquisa CNPq pela concessão da bolsa que foi fundamental para a realização deste trabalho.

DEDICATÓRIA

Dedicado

Primeiramente a Deus por minha existência e por me capacitar a chegar até aqui aos que me ajudaram a superar todas as dificuldades que surgiram durante toda essa trajetória.

Meus pais Pedro e Lúcia por apoio, motivação e por suas orações nos momentos mais difíceis.

Meus irmãos Peterson, Andresa e Ismael por torcerem por meu sucesso e por me darem sobrinhos e sobrinhas muito lindos.

A todos os amigos e colegas que contribuiram para a realização desse sonho.

RESUMO

Este trabalho descreve a implementação de uma plataforma baseada em arquitetura reconfigurável e conceitos de instrumentação virtual aplicados a implementação do controle de um carro, que inclui tarefas de controle relacionado à direção, câmbio, acelerador, freio e embreagem. A novidade desta abordagem é o uso de sistemas reconfigurável (para desenvolver o controlador do carro) e instrumentação virtual o que permite desenvolver uma abstração de alto-nível em um ambiente de teste e simulação. O sistema de controle do carro foi desenvolvido em um microprocessador com vários periféricos embarcados na FPGA (***F**ield **P**rogrammable **G**ate **A**rray*). A comunicação entre o sistema de controle baseado na FPGA para o carro é realizada por um módulo eletrônico formado por blocos separados, e placas de circuito de potência. A instrumentação virtual foi usada para implementar um ambiente com um alto-nível de abstração para a simulação com a ferramenta LabVIEW que permite representar o movimento do carro em tempo real. A comunicação entre o simulador e o controlador é realizada por uma interface serial na qual um protocolo RS-232 foi implementado. O usuário pode enviar comandos ao sistema de controle por um teclado com uma interface de PS2. Os comandos estão definidos com uma semântica e sintaxe específicas, permitindo ao usuário executar várias manobras com o carro. Esta abordagem abre uma grande variedade de possibilidades para validar e simular soluções para vários problemas em robotica e outras áreas da mecatrônica. Os testes e validações do sistema foram realizados no ambiente de simulação e no carro real. Então, esta abordagem tornou possível comparar os resultados de simulação com as variáveis de acionamento do carro real coletados em tempo real. Esta abordagem torna possível testar e validar o sistema de controle com baixo custo e maior segurança.

ABSTRACT

This work describes the implementation of a platform based on reconfigurable architecture and concepts of virtual instrumentation applied to the implementation of a car's control, which includes control tasks related to the steering wheel, gear, throttle, brake and clutch. The novelty of this approach is the use of both reconfigurable systems (for developing the car's controller) and virtual instrumentation issues for developing a high-level abstraction testing and simulation environment. The car control system was developed in a microcontroller with several peripherals embedded in a FPGA (**F**ield **P**rogrammable **G**ate **A**rray). The communication between the FPGA-based control system and the car is accomplished through an electronic module, which comprises several insulating and power circuit boards. The virtual instrumentation approach was used for implementing a high-level abstraction simulation environment in LabVIEW tool, which allows representing the movement of the car in real time. The communication between the simulator and the controller is accomplished through a serial interface in which a RS-232 based protocol was implemented. The user can send commands to the control system through a keyboard with a PS2 interface. The commands were defined with a specific syntax and semantics, allowing the user to execute several maneuvers with the car. This approach opens a great variety of possibilities to validate and simulate solutions for several problems in robotic and mechatronic areas. The tests and initial overall system validation were accomplished in both the simulator environment and in the real car. Therefore, this approach made possible to compare the simulation results with the movement variables of the real car, which were gathered in real time. This approach makes possible to test and to validate the control system with low cost and more safety.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIMBOLOS, SIGLAS E ABREVIATURAS

ABS – *Anti Block System* – Sistema anti bloqueio.

ASIC – *Application Specific Integrated Circuit* – Circuitos integrados para aplicações específicas.

AHS – *Automated Highway Systems* – Sistema de automação rodoviário.

BAS – *Braking Assitant* – Assistente do freio.

CDC – *Continuous Damping Control* – Controle continuo da suspensão.

CLB – *Configurable Logics Block* – Blocos lógicos reconfiguráveis.

CLP – Controlador Lógico Programável.

CPLD – *Complex Programmable Logic Device* – Dispositivo complexo de lógica programavél.

DARPA – *Defense Advanced Research Projects Agency* – Agencia da defesa em projetos de pesquisa avançada.

DSP – *Disigned System Processor* – Sistema projetado de processamento.

EDA – *Eletronic Design Automation* – Eletrônica para projeto de automação.

EDK – *Embedded Development Kit* – Kit de desenvolvimento embarcado.

EPROM – *Erasable Programmable Only Memory* – Memória programável apagável.

ESP – *Eletronic Estability Program* – Programa eletônico de estabilidade

FLS – *Fast Simplex Link* – Ligação rápida simpes.

FPGA – *Field Programmable Gate Array.*

FSM – *Finite State Machine* – Maquina de estados finitos.

GUI – *Graphical User Interface* – Interface Gráfica do Utilizador.

HDL – Hardware *Description Language* – Linguagem de descrição de Hardware.

HF – Hardware Fixo.

HP – Hardware Programável.

ILP – *Instruction Level Parallelism* – Paralelismo no nível de instrução.

INRIA – *Intitut National de Recherche en Informatique et en Automatique* – Instituto de pesquisas em informatica e automação.

I/O – *Input/Output* – Entrada e saída

IOB – *Input Output Block* – Bloco de entrada e saída

ISE – *Integrated Software Environment*.

LabVIEW – *Laboratory Virtual Instruments Engineering Workbench*

LMB – *Local Memory Bus* – Memória local de barramento.

LUT – *Lookup Table*

MIMD – *Multiple Instruction Stream Multiple Data Stream* – Fluxo múltiplo de instrução e fluxo múltiplo de dados.

MPGA – *Mask Programmable Gate Array*

OPB – *On-chip-Periferical Bus* – Barramento de periférico dentro do chip.

PC – *Personal Computer* – Computador pessoal.

PLA – *Programmable logic Array*

PLD – *Programmable Logic Device* – Dispositivo lógico programável.

PLP – *Processor Level Parallelism* – Paralelismo no nível do processador.

PWM – *Pulse With Modulation* – Modulação por largura de pulso.

SiAE – Sistema Automático de Estacionamento.

SIMD – *Single Instruction Stream Multiple Data Stream* – Fluxo simples de instrução e fluxo multiplo de dados.

RAM – *Random Access Memory* – Memória de acesso

RISC – *Reduced Instruction Set Computer*

SiP – *System in a Package*

SoC – *System-on-Chip*

SRAM – *Static Random Access Memory* – Memória estática de acesso.

TCS – *Traction Control System* – Sistema de Controle de Tração.

ULA – Unidade Lógica Aritimética.

VHDL – *Very Hight Description Language*

VI – *Virtual Intrumentation* – Instrumentação virtual.

VLSI – *Very Large Scale Integration*

XCL – *Xilinx CacheLink*

ϕ – Ângulo das rodas.

θ – Ângulo do vaículo em $_{XX}$ io ao eixo horizontal X no espaço bidimensional.

x – Distância do centro do carro até o eixo y.

y – Distância do centro do carro até o eixo x.

l – Distância das lateral entra as rodas.

V – Velocidade do veículo.

INTRODUÇÃO

Atualmente, os sistemas autônomos são utilizados em diversas áreas em variadas aplicações valendo-se em muitos casos de tecnologias que utilizam microprocessadores, que cada vez mais são exigidas a atender às crescentes necessidades de desempenho, baixo custo e com tempo de ciclo de projeto cada vez menor. Um sistema autônomo pode ser caracterizado por sua capacidade de execução das tarefas para as quais foi projetado, sem a interferência de qualquer controle externo, (Tzuu-Hseng S., et al., 2003) e (Baltes J. e Lin Y. Lin, 1999).

Os sistemas automatizados embarcados vêm ganhando grande espaço com o avanço da tecnologia, e variadas aplicações vêm sendo desenvolvidas em diversos laboratórios acadêmicos e de empresas, trabalhando em diferentes áreas como: automatização de processos fabris, sistemas de auxílio ao motorista, navegação robotizada em ambientes hostis, transporte de cargas, realização de tarefas repetitivas.

Essas aplicações, em particular o desenvolvimento de sistemas autônomos em diferentes níveis, têm gerado desafios cada vez maiores para os profissionais que trabalham no desenvolvimento de tecnologias aplicadas nessa área. Para tanto, técnicas baseadas em modelos matemáticos complexos como em (Yang E., et al., 2004), aplicações com redes neurais (Gu D. e Hu. H., 2002) e (Li, J. H. Lee e Li, P. M, 2005), algoritmos genéticos e lógica Fuzzy (Zhao Y. e Collins, Jr. E.G, 2005), entre outros são utilizadas de diferentes formas para o desenvolvimento desses projetos.

Técnicas cada vez mais elaboradas para o desenvolvimento de sistemas automatizados, como as FPGA's – *Field Programmable Gate Arrays*, estão sendo usadas no desenvolvimento de sistemas de automação e controle (aplicando-se a prototipação rápida de *circuitos integrados*). Isto permite tornar possível alcançar resultados de bom desempenho e custo (Dehon A., 2000).

A grande complexidade dos sistemas atuais de automação veicular envolvendo dispositivos mecânicos, eletrônicos e sistemas computacionais estimulam a introdução de novas metodologias de projeto. Um ponto importante é que as metodologias de desenvolvimento exigem um nível de abstração alto para elaboração, verificação e validação do sistema proposto.

Este trabalho descreve a implementação de um Sistemas de Controle de Movimentação Veicular baseado em técnicas de arquitetura reconfigurável e de Instrumentação Virtual. Desta maneira foi definido um novo fluxo de projeto que permite um nível alto de abstração para validação do sistema. Um ponto importante, é que o sistema a ser validado/simulado é composto de um controlador (implementado em arquiteturas reconfiguráveis) e vários outros elementos como atuadores, interfaces, sensores, etc. Objetivando o desenvolvimento da base de controle de movimentação do veículo para o projeto SiAE este sistema foi projetado e implementado aplicando-se técnicas que cada vez mais estão ganhando grande visibilidade no cenário mundial.

As técnicas de arquiteturas reconfiguráveis foram aplicadas no projeto de um microcontrolador embarcado em uma FPGA (*Field Programmable Gate Array*) com diferentes módulos responsáveis por executar diferentes funções.

Por outro lado, os conceitos ligados a Instrumentação Virtual foram aplicados na construção do ambiente de Simulação e Validação do Controle de movimentação do veículo usando a ferramenta Labview (Regazzi R. D., et al., 2005).

A implementação do sistema de controle de movimentação foi desenvolvido em etapas diferentes durante sua elaboração descrita neste trabalho. Inicialmente, foi projetado o sistema de controle de movimentação, utilizando os conceitos de processador embarcado em lógica reconfigurável, para o controle dos atuadores instalados no veículo de teste do projeto SiAE. A segunda etapa consistiu no desenvolvimento do ambiente de simulação o qual objetivou a validação e os testes do sistema de controle. A terceira e última fase consistiu no projeto e desenvolvimento da eletrônica para o acionamento e interface entre o controlador e os atuadores e transdutores.

Assim como o desenvolvimento do trabalho foi marcado por suas diferentes etapas, os resultados obtidos também foram distribuídos mediante a cada uma destas etapas alcançadas. Os resultados são apresentados no capítulo 4, capítulo 5 e capítulo 6, onde cada uma das etapas citadas é mostrada em detalhe.

1.1. CARACTERIZAÇÃO DO PROBLEMA

A abordagem tratando do controle de veículos autônomos vem sendo largamente pesquisada em diferentes níveis por todo o mundo.

A problemática proposta consiste no projeto e desenvolvimento de um sistema de controle de movimentação de um carro de passeio pequeno aplicando-se processamento embarcado em lógica reconfigurável no projeto do sistema. O sistema possui uma arquitetura dividida em módulos e blocos organizados de forma que cada qual realiza tarefas específicas de forma ordenada.

O sistema proposto nesse trabalho é parte do projeto SiAE (**S**istema **A**utomático de **E**stacionamento). A proposta do SiAE é tornar um veículo capaz de realizar o estacionamento de uma vaga paralela a sua direita. Na arquitetura do projeto SiAE este trabalho contribui para a implementação do sistema de controle de movimentação do veículo.

1.1.1. Projeto SiAE

O projeto SiAE teve início em 1999 com forte apoio da FIAT na doação do veículo de teste e financiamento da pesquisa. Desde seu início este projeto despertou grande interesse acadêmico e comercial, pois se trata de uma inovação tecnológica ainda não dominada no âmbito nacional e seu conceito arquitetônico contribui significantemente com os sistemas existentes.

Desde o surgimento do projeto, seu objetivo principal é o desenvolvimento de um sistema de estacionamento automático em uma vaga paralela. Esse estudo está inserido dentro do contexto de sistemas autônomos para o controle veicular já foi desenvolvido em diferentes trabalhos (Han-Shue, et al, 1999), (Wada M., et al, 2003) e (Shimazaki et al., 2004).

Esse projeto já passou por três fases distintas. Na primeira fase foram definidas quais as variáveis que deveriam ser automatizadas, projetando-se os atuadores que efetuariam o acionamento destas. Na etapa seguinte, foi definida uma arquitetura flexível e distribuída para o controle das variáveis automatizadas do veículo. Na terceira etapa e atual, foram projetados e embarcados os módulos de controle de movimentação em uma arquitetura reconfigurável.

Com o controle de movimentação do veículo operante, as fases seguintes do projeto objetivam um nível de autonomia maior, o qual é necessário para efetuar o estacionamento de forma autônoma.

As figura 1.1, abaixo, apresentam o veículo de teste onde a plataforma do sistema de controle de movimentação do projeto SiAE foi desenvolvido**.**

Figura 1.1 – Foto do veículo de testes do projeto SiAE (Bellardi, T., 2005).

A figura 1.1 mostra o veículo de teste no qual foi montada toda a plataforma de automação. Este veículo é um Palio com motorização 1000 cm^3 e injeção eletrônica de combustível, câmbio manual com seis marcha inclusive a ré, sistema de direção, acelerador e embreagem mecânicos.

Para a instalação dos dispositivos no veículo, foi necessário desmontar partes plásticas do acabamento do painel e o pára-choque dianteiro. A instalação dos dispositivos seguiu o conceito de modificar o mínimo possível os dispositivos instalados originalmente no veículo. Vale ressaltarmos que o objetivo do projeto é a utilização de dispositivos de baixo custo buscando-se o desenvolvimento de um produto acessível em sua comercialização.

O trabalho de (Bellardi, T., 2005) concentrou-se na definição da arquitetura flexível para o controle de movimentação do veículo e a definição dos módulos dessa arquitetura. Também foram projetados, construídos os circuitos eletro-eletrônicos para a primeira versão do *software* de controle da movimentação do veículo em um microcontrolador 8055 da família do 8051.

Na figura 1.2, observa-se a instalação de um motor de corrente contínua para atuar no sistema de direção do veículo.

Este atuador aciona a direção por meio de um redutor montado diretamente na cremalheira.

Figura 1.2 – Foto do acionamento da direção (Bellardi, T., 2005).

Para o acionamento da borboleta de aceleração foi instalado um motor de corrente contínua de deslocamento linear, figura 1.3. Originalmente este motor é utilizado para efetuar o controle da marcha lenta do motor (Bellardi, T., 2005).

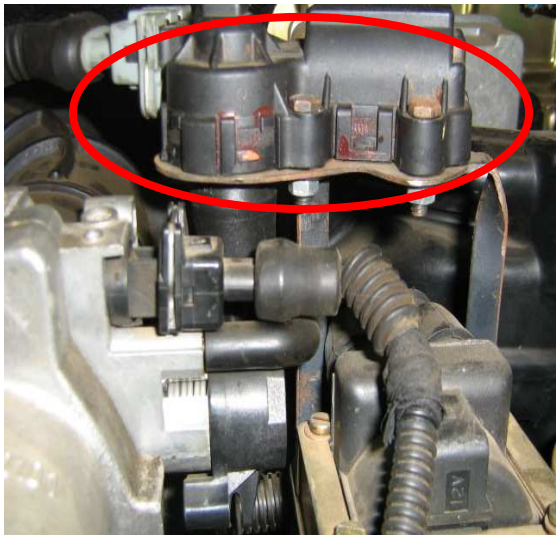


Figura 1.3 – Foto do acionamento do acelerador (Bellardi, T., 2005).

No sistema de freio, foi instalado também um motor de corrente contínua, como mostra a figura 1.4. Quando se necessita acionar o freio, o motor atua na direção horária e quando é necessário desacioná-lo, o motor gira no sentido anti-horário.

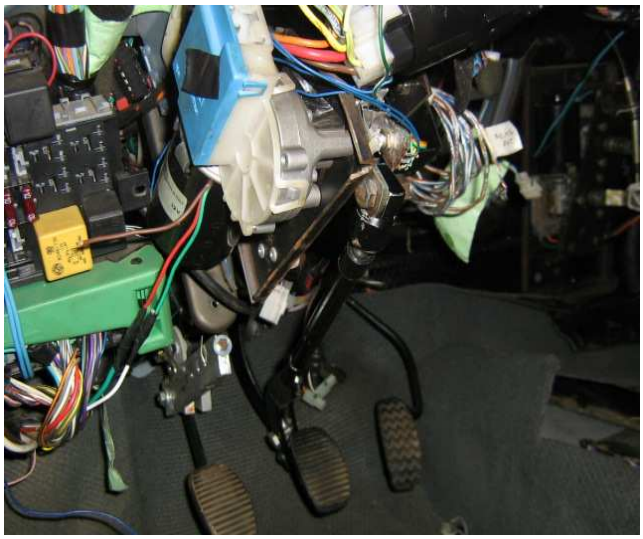


Figura 1.4 – Foto do acionamento do freio (Bellardi, T., 2005).

Para o acionamento do câmbio (figura 1.5), foram utilizados dois motores de corrente contínua com redução tipo coroa helicoidal e parafusos sem fim que atuam na alavanca do câmbio por meio de barras flexíveis.

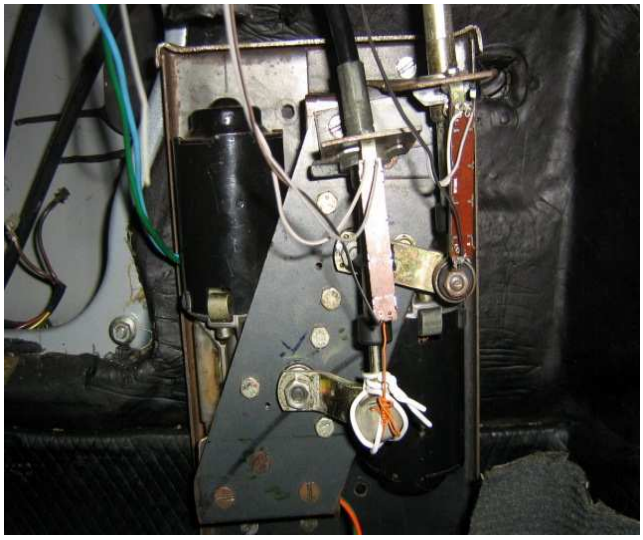


Figura 1.5 – Foto do acionamento do câmbio (Bellardi, T., 2005).

Na figura 1.6, para o acionamento da embreagem foi utilizado o sistema eletro-pneumático Autonomy® da FIAT. Este sistema atua diretamente no platô podendo acoplá-lo de duas formas: acionamento suave ou acionamento brusco.

Figura 1.6 – Foto do sistema de acionamento da embreagem (Bellardi, T., 2005).

Uma vez que o controle de movimentação esteja operante as fases que se seguem para que o projeto alcance o nível de autonomia estão fundamentadas no planejamento de trajetória e o controle de navegação bem com o sensoriamento.

1.2. MOTIVAÇÃO DESTE TRABALHO

Atualmente, os sistemas autônomos veiculares ganham cada vez mais espaço. Fatores relacionados à segurança, conforto e eficiência são os principais motivadores, os quais têm despertado o interesse de montadoras e laboratórios de pesquisa a redirecionarem suas pesquisas em sistemas autônomos.

Entretanto, grande parte desses sistemas são projetados em arquiteturas de sistemas microprocessados com diferentes conceitos (Shimazaki et al., 2004), (Wada M., 2003) e (Good et al., 1988). As aplicações desenvolvidas em sua grande maioria são desenvolvidas para carros com motores elétricos com diversos acessórios automáticos ou semi-automáticos.

Os sistemas desenvolvidos e comercializados possuem grande número de processadores espalhados por todo o veículo executando funções específicas. Outros sistemas foram desenvolvidos sobre plataforma de veículos elétricos os quais possuem inúmeros fatores simplificadores (Consoni F., 2004).

O projeto em questão por sua vez pode ser definido como diferenciado das aplicações já desenvolvidas, pois possui premissas que estabelecem a utilização de um veículo com motorização de 1000 cm^3 com câmbio e direção mecânicos, visando a obtenção de um sistema de baixo custo flexível o suficiente para adaptar-se a diferentes modelos de carros realizando a tarefa de estacionamento.

Todo esse trabalho envolve um detalhado estudo para o desenvolvimento de um sistema robusto, simples e barato que implique em modificações mínimas no veículo.

Os sistemas já desenvolvidos utilizam sistemas sofisticados e complexos instalados em veículos modernos que contam com inúmeros dispositivos de controle já instalados.

1.3. OBJETIVOS GERAIS

Este trabalho tem como objetivo principal desenvolver um controlador de movimentação para o veículo fazendo a leitura dos sinais dos sensores e acionando os atuadores de acordo com as necessidades, seguindo as especificações de arquitetura flexível em sistemas distribuídos (Bellardi, T., 2005).

A abordagem aplicada para este propósito baseia-se no projeto e implementação de um controlador de movimentação embarcado em lógica reconfigurável propiciando, nesta fase, a um veículo de passeio receber comandos para movimentar-se de forma segura e controlada. Por outro lado, este trabalho também visa desenvolver uma interface entre o controlador o operador para enviar os comandos para o ambiente de teste e de simulações.

1.3.1. Objetivos Específicos

Os objetivos específicos deste trabalho são:

1- Estudar, projetar, simular e implementar a programação dos módulos e seus respectivos blocos para o controle de movimentação em um microprocessador embarcado em arquitetura reconfigurável.

2- Projetar, testar e implementar os periféricos para o microprocessador embarcado em lógica reconfigurável e conectar ao barramento de comunicação.

3- Projetar um ambiente de simulação baseado em ferramentas de instrumentação virtual para a validação do sistema de controle.

4- Definir e projetar uma comunicação entre o operador e o controlador de forma semi-automática ou manual propiciando ao sistema receber comandos e executar as ações necessárias para a movimentação do veículo.

5- Projetar e construir uma interface flexível de monitoração e simulação do controle de movimentação para teste e validação do sistema em condições normais de operação.

1.4. JUSTIFICATIVA

Na área de veículos autônomos ou automação veicular, tem a cada dia aumentada a importância dos conceitos de segurança, conforto e eficiência. Visando maximizar cada vez mais estes princípios, muito se tem investido em novas pesquisas dentro dos conceitos de *drive-by-wire*, sistemas de apoio ao motorista e de automação de tarefas repetitivas dando maior conforto e segurança.

Estudos mostram o aumento da utilização de sistemas eletrônicos para o controle e monitoração de diversos dispositivos, sistemas e até tarefas.

Os automóveis representam os produtos que vêm cada vez mais incorporando equipamentos e dispositivos microeletrônicos, o que contribui para o aumento do consumo de insumos, em especial semicondutores. Os equipamentos incorporados aos automóveis representam a eletrônica embarcada na forma de sensores, computadores de bordo, circuitos eletrônicos, freios inteligentes, instrumentação, câmbios automáticos e entre outros (Consoni F., 2004).

Um exemplo importante é o BMW série 7, o qual possui 63 processadores embarcados (BMW série 7, 2005). Um ponto importante é que a capacidade para executar um projeto cresce mais lentamente se comparado com o aumento da complexidade do mesmo. Este ponto se reflete em um aumento lento da produtividade do ciclo de projeto de sistemas complexos (Harteinsten R., 2002).

O mercado tem respondido de forma positiva com o crescimento de unidades produzidas e comercializadas.

Estima-se que nos próximos anos o aumento do consumo destas tecnologias e novos conceitos em eletrônica embarcada nos veículos tragam maior segurança e conforto ao motorista e aos ocupantes.

Torna-se cada vez maior o relacionamento de computadores e da indústria de semicondutores com os automóveis. Analistas estimam que em 2000 o conteúdo médio de silício (que é base para os semicondutores) em um carro era de 239 dólares, com forte tendência de crescimento da utilização da eletrônica na indústria automotiva (Amato N. J., 2004).

Esta modernização está trazendo aos motoristas e passageiros diversos benefícios como: maior segurança ao dirigir, apoio ao motorista em situações de emergência, limitadores e controladores de velocidade e estabilidade do veículo e os sistemas de ajuda no estacionamento dos veículos.

Em países da Europa como a Alemanha, França e Itália existem rodovias com alto grau de automatização atuando de forma inteligente, são as chamadas AHS – *Automated Highway Systems*.

Essas rodovias, por sua vez, são equipadas com diversos dispositivos que auxiliam o motorista no controle e condução do veículo, podendo estabelecer o controle de velocidade eletronicamente, avisando o motorista das condições de alerta e emergência.

1.5. METODOLOGIA DE PROJETO

A metodologia aplicada neste trabalho dedicou-se ao projeto de um sistema de controle da movimentação para um veículo de passeio comum a partir dos conceitos da arquitetura flexível, modular e distribuída definida por (Bellardi, T., 2005). Esta arquitetura é dividida em módulos a fim de propiciar futuras modificações do sistema e o acréscimo de outras funcionalidades.

O sistema de controle de movimentação foi subdividido em três módulos: módulo de controle de movimentação, módulo de interface com o operador e o módulo de acionamento dos atuadores, sendo que todos eles serão detalhados posteriormente.

A sistemática de controle de movimentação foi projetada usando as ferramentas do EDK – *Embedded Development Kit*, do fabricante Xilinx, para a configuração do processador embarcado na FPGA Spartan-3 e o ISE – *Integrated Software Environment*, para a configuração do módulo de comunicação do teclado em VHDL – *Very Hight Description Language*. Na concepção do *software* desse controlador embarcado foi aplicada uma metodologia distribuída concebida por (Bellardi T., 2005), onde cada variável de controle de movimentação do veículo é controlada por um controlador local que responde a um controle central.

Também neste trabalho foi abordada uma metodologia de teste e validação em dois níveis. No primeiro nível de validação foram realizados testes no ambiente de simulação do controlador desenvolvido. No simulador foram utilizados conceitos de instrumentação virtual no ambiente do Labview da *National Instruments*.

A presente metodologia de projeto, envolvendo Instrumentação Virtual e Arquiteturas Reconfiguráveis, mostra-se promissora devido à grande flexibilidade das técnicas utilizadas. O projetista pode modelar diversas técnicas de navegação e controle para validar desempenho e viabilidade técnica/econômica do projeto.

1.6. LIMITAÇÕES

Este trabalho limita-se ao estudo, projeto, simulação, teste e implementação dos módulos e seus respectivos blocos para o controle de movimentação usando um microprocessador embarcado em lógica reconfigurável. Além do anterior, o trabalho inclui o projeto das placas de potência e de uma interface de comandos do operador para o sistema, bem como as modificações necessárias nos atuadores e transdutores para o bom funcionamento do sistema.

1.7. RESULTADOS ALCANÇADOS

Abaixo destacam-se os principais resultados obtidos neste trabalho:

1) Um controlador implementado em um processador e seus periféricos embarcados em uma FPGA. Neste trabalho são apresentados dados relativos à ocupação da FPGA e a utilização dos recursos do kit de desenvolvimento utilizado (ver capítulo 4).

2) Um ambiente de simulação para a observação e validação do comportamento do sistema veículo/controlador (ver figuras 3.4 e 3.5). São mostrados os resultados pertinentes aos testes de acionamento dos atuadores separadamente e o controle de movimentação no ambiente de simulação. (vide capítulo 5).

3) O projeto de placas de potência e placas de interfase FPGA/atuadores (ver capitulo 6).

4) O projeto e implementação de uma placa de aquisição de sinais baseado em um conversor analógico/digital (ver capitulo 6).

1.8. APRESENTAÇÃO DO TRABALHO

Este documento está organizado de forma a proporcionar ao leitor uma apresentação das tecnologias envolvidas neste trabalho relacionados ao projeto SiAE. Para apresentar estas informações de forma organizada e concisa, este trabalho foi distribuído em sete capítulos.

No primeiro capítulo, é apresentado ao leitor um panorama introdutório ao trabalho. Esta apresentação consiste na definição das especificações e características do problema a ser resolvido neste trabalho. Também é apresentado o projeto SiAE, seus objetivos e características, bem como os objetivos gerais e específicos deste trabalho dentro do projeto e os objetivos específicos. Nas seções seguintes, são apresentadas ainda as justificativas deste trabalho, suas limitações e a metodologia aplicada em sua elaboração.

O segundo capítulo dedica-se aos fundamentos tecnológicos aplicados neste trabalho. São apresentados os conceitos principais de computação reconfigurável, suas arquiteturas e funcionamento. Ainda neste tópico, o leitor encontra os conceitos de instrumentação virtual e suas ferramentas. Também pode encontrar aqui os conceitos principais de automação veicular e a tecnologia relacionada a esta área.

Já no terceiro é apresentado ao leitor a arquitetura do sistema desenvolvido no projeto SiAE, seu histórico e evolução nas diferentes etapas de sua elaboração.

No quarto capítulo, é apresentado o desenvolvimento da configuração do *Hardware* embarcado na FPGA bem como o se *Software* de controle, sendo apresentado os resultados e os resultados alcançados.

O quinto capítulo dedica-se à apresentação do desenvolvimento do ambiente de simulação, o funcionamento de seus módulos e os resultados obtidos com o desenvolvimento do ambiente de simulação.

O desenvolvimento do sistema de controle do veículo de testes está descrito no sexto capítulo. É apresentado o projeto dos circuitos de todas as placas, sua função e os resultados dos testes de acionamento e movimentação com o veículo de testes.

No sétimo e último capítulo o leitor encontra as conclusões obtidas com a elaboração implementação desse trabalho. Também são listados os futuros trabalhos mediante o estado atual do projeto tendo em vista as inovações tecnológicas.

Por fim nos apêndices o leitor encontra os desenhos das placas e dos PCB`s, relatório de utilização dos recursos de *hardware* do sistema de controle embarcado, os fluxogramas dos *software* de controle de cada um dos módulos do controlador embarcado em lógica reconfigurável.

2. FUNDAMENTOS TECNOLÓGICOS

Este capítulo apresenta a fundamentação teórica dos conceitos e tecnologias utilizados neste trabalho.

Inicialmente, descrevemos um breve histórico dos sistemas de automação veicular, enfatizando a evolução e a incorporação de sistemas automatizados. Também são abordadas as principais tecnologias aplicadas a veículos de passeio a evolução dos sistemas e os conceitos que revolucionaram a forma de dirigir um veículo possibilitando novos patamares de segurança, conforto e economia de combustível.

As pesquisas em automação veicular abriram as portas para o surgimento de sistemas que auxiliam o motorista na execução de tarefas pré-estabelecidas e no gerenciamento de funções específicas ou gerais como: manter a velocidade do veículo, manter a distância do veículo a sua frente, estacionamento automático ou até mesmo o controle de temperatura no interior do veículo.

Com o avanço das tecnologias relacionadas aos veículos automatizados, novos paradigmas na área de sistemas veiculares robotizados surgiram. Estes sistema ganham espaço em aplicações industriais (e.g. carga e descarda) e, adicionalmente, em aplicações urbanas objetivando cada vez mais isentar as pessoas de executarem tarefas repetitivas e/ou desgastantes.

Por outro lado, os sistemas de automação veícular objetivam o aumento de segurança, pois com o desenvolvimento desses sistemas não haveria mais excesso de velocidade, ultrapassagens perigosas, motoristas dirigindo embriagados e os acidentes causados por motoristas que dormem ao volante.

O desenvolvimento de sistemas veiculares robotizados cada vez mais avançados necessitam de ciclos e sistemas de teste, simulação e validação ainda mais próximos da realidade com a finalidade de exaurir a possibilidade de falhas. Com esse enfoque, os novos projetos na área de automação veicular têm utilizado variados recursos para os testes de simulação e validação.

A instrumentação virtual, mais propriamente a ferramenta LabVIEW da *National Instruments*, tem se mostrado atraente pois possibilita de forma visual a implementação de simuladores eficiente para a execução de testes preliminares para a validação dos sistemas automatizados.

A instrumentação virtual, surgiu com o intuito de suprir as necessidades de ferramentas de desenvolvimento de aplicações acadêmicas, mas atualmente com seu desenvolvimento, muitos projetistas estão aplicando seus conceitos em pequenas aplicações de automação industrial.

Os recursos oferecidos por esta ferramenta são aplicados para teste, verificação, automação e controle do sistema implementado por meio de modelos matemáticos teóricos ou modelos realísticos comportamentais. Também se mostra muito eficiente no desenvolvimento de aplicações voltadas a interface de sistemas automatizados (e.g sistemas de supervisório).

Juntamente, com o desenvolvimento dos sistemas veiculares automatizados o desenvolvimento de novas tecnologias de sistemas embarcados tem se mostrado necessário para dar suporte as necessidades cada vez mais pujantes. A quantidade de eletrônica embarcada nos veículos, atualmente, já está na casa de centenas no caso de veículos mais luxuosos.

Visando este crescimento as industrias de desenvolvimento de sistemas eletrônicos para aplicações veiculares está investindo na pesquisa de novas metodologias para o controle dos sistemas veiculares embarcados. Nessa dinâmica a computação reconfigurável vêm se mostrando uma tecnologia interessante, pois sua fundamentação está baseada em *hardware* no que tange a freqüência de funcionamento e *software* no que diz respeito a flexibilidade. A reconfigurabilidade do *hardware* materializada nas FPGA's tem se mostrado competitiva em termos de custo propriamente do *chip* e do seu ciclo de projeto.

Nesse capítulo são abordados os conceitos tecnológicos aplicados no desenvolvimento, teste e simulção do sistema de controle de movimentação de um veículo de passeio movido a motor de combustão interna. Estes conceitos estão relacionados com a aplicação de Sistemas Reconfiguréveis e Instrumentação Virtual para o projeto de Sistemas de Automação Veicular.

2.1. ASPECTOS TECNOLÓGICOS APLICADOS A AUTOMAÇÃO VEICULAR

Nesta seção são abordados os principais aspectos tecnológicos aplicados a automação veicular. Dentro dos conceitos relacionados a automação veicular são descritas as tecnologias aplicadas aos veículos de passeio seus conceitos e o estado da arte em sistemas inteligentes para o controle veicular.

2.1.1. Aspectos Gerais das Tecnologias Aplicadas a Sistemas de Automação Veicular

A automação veicular, além de proporcionar ao motorista segurança em situações criticas, pode também auxilia-lo dando-lhe informações necessárias para que possa executar manobras, ou mesmo automatizar alguma das tarefas relacionadas a dirigibilidade. As indústrias automotivas tem investido nesta área visando aumentar o conforto, segurança, estabilidade e rendimento dos veículos, utilizando-se de sistemas embarcados que controlam a ação de frenagem ABS – *Anti-lock-Breaking-System* (Kelber, 2003a), estabilidade do veículo ESP – *Eletronic Stability Program* (Kelber, 2003a), controle da velocidade, controle da autonomia, controle do consumo médio de combustível, controle de temperatura.

Com o aumento das necessidades de tornar os veículos mais inteligentes, soluções computacionais tornaram-se cada vez mais comuns. Em diversos paises os motoristas podem contar com computadores que possuem os mapas das ruas da cidade e recebem informações sobre as condições de trânsito escolhendo para o motorista o melhor trajeto.

As tecnologias atualmente desenvolvidas apresentam em sua maioria um grande conjunto de sensores, transdutores, atuadores, sistemas de comunicação modernos, que possibilitam a estes veículos cada vez mais executar tarefas mais complexas de forma segura. Para dotar estes veículos de comportamentos inteligentes são incorporados componentes de percepção garantindo maiores níveis de autonomia e robustez.

Cada vez mais os projetos de veículos inteligentes estão incorporando tecnologias de desenvolvimento de robôs autônomos, estudos da cinemática (Dudek G. and Jenkin M., 2000), comunicação, controle e inteligência para execução de tarefas.

A concepção de um veículo com graus de autonomia é uma idéia que vem sendo explorada e estudada por todo o mundo. A automação veicular, mesmo em seus diferentes níveis, proporciona aos motoristas e ocupantes maior segurança em condições adversas. Auxilia na condução do veículo executando tarefas de forma automatizadas, como por exemplo: manter o veículo na pista dentro da faixa correta, manter a distância dos veículos a sua frente, controlar a velocidade do veículo conforme o transito, achar o caminho mais curto e seguro para se chegar ao destino e até estacionar o veículo (Kelber C. R, et al., 2005).

Em 1997, o I.N.R.IA – *Institut National de Recherche en Informatique et en Automatique* apresentou ao público um veículo de fabricação em série, chamado de CyCab, com diversos modos de movimentação automatizados (Baille, Gérard et all., 1999). O CyCab já está sendo produzido pela empresa Robosoft, que já apresenta inclusive outros modelos, com o RoBUCAB, o RobuRIDE e o RoBUCAR (Robosoft, 2003).

Em 2003, a Toyota lançou, no Japão, o Prius, um modelo de veículo com capacidade de efetuar manobras de estacionamento autonomamente (Self-parking, 2003).

Em outubro de 2005, quatro veículos conseguiram completar o percurso da competição promovida pelo DARPA – *Defense Advanced Research Projects Agency*, o *DARPA Grad Chalenge Race*. 132 milhas (aproximadamente 212 km) foram percorridos pelo deserto de Nevada, nos Estados Unidos da América, de maneira completamente autônoma (CNN (a), 2005).

2.1.2. Tecnologias aplicadas a Veículos de Passeio

Historicamente, os conceitos tecnológicos no universo automobilístico vêm evoluindo ao longo dos últimos 100 anos. Em 1894 Vacheron lança o primeiro automóvel com um volante, em 1895 Panhard fabrica o primeiro carro fechado, o primeiro motor 4 cilindros em linha é lançado em 1898, 1899 surge o cambio em ‘H’ e o acelerador no pé. Em 1917 o primeiro veículo equipado com um velocímetro é produzido, o primeiro automóvel totalmente fechado e fabricado em aço é lançado em 1923.

Os conceitos têm evoluído trazendo consigo novas tecnologias que revolucionam o mundo até os dias de hoje. Estudiosos afirmam que os carros não serão mais somente um meio de transporte, mas uma extensão da casa e do escritório. Foi estipulado um prazo de cinco anos para que este conceito entre no mercado mundial.

A cada época novas necessidades surgem e com elas novos conceitos tecnológicos. Nas décadas de 50 e 60, os fabricantes buscavam velocidade, com motores cada vez mais potentes. Nos anos 70, foi a vez de reduzir o consumo de combustível, carros velozes, potentes e econômicos.

Em 1980 e 90, segurança foi colocada como meta em todo o mundo, trazendo novos aprimoramentos como controles de freios anti-bloqueio, ABS, o *drive-by-wire*, e o *Air Bags* (Kelber, 2003a).

Em 2000 os conceitos de conforto e acessibilidade a informação ganham importância. Utilizar a internet, digitar *email's*, obter ajuda de um computador de bordo para chegar ao destino desejado são exemplos típicos.

Com todos os avanços tecnológicos para auxílio na dirigibilidade do veículo, o crescimento de eletroeletrônicos embarcados tem sido exponencial com o surgimento de conceitos como: *break-by-wire, drive-by-wire e steer-by-wire* (Kelber C. R, et al., 2005).

O conceito *drive-by-wire* é originário da aviação onde foi desenvolvido o conceito *fly-by-wire* utilizado pela primeira vez no avião modelo F-16 com a finalidade de automatizar funções de controle de estabilidade o que segundo os engenheiros que participaram deste projeto, seria humanamente impossível pilotar esta aeronave por causa de sua instabilidade. Nos veículos terrestres este conceito de controle por sinais eletrônicos foi desenvolvido para auxiliar os pilotos dos carros da F-1 pois permite administrar os dispositivos de controle (acelerador, freio, tração e direção) do veículo.

Juntamente com os sistemas de auxiliar ao motorista priorizando a segurança, desempenho e redução dos índices de danos ambientais surgem também conceitos denominados de *Sistemas de Apoio ao Motorista*.

Estes sistemas baseiam-se em informações internas e externas por meio de sensores e transdutores verificando o estado atual do veículo e intervindo quando e se for necessário. Além deste, um sistema sofisticado de comunicação complementa a interação da estrutura de controle como um todo (Kelber C. R, et al., 2005).

2.1.3. Conceitos em Tecnologias de Rede Aplicados a Veículos

Atualmente, a grande maioria dos veículos fabricados em todo o mundo já está equipado com uma rede de comunicação entre módulos diferentes, espalhados por todo o veículo. A necessidade de se projetar uma rede de comunicação entre os módulos eletroeletrônicos, delimitava algumas restrições físicas para a utilização dos protocolos existentes. Problemas com temperaturas, vibrações e interferências prejudicavam consideravelmente a utilização de protocolos existentes, necessitando-se de um estudo para o desenvolvimento de um protocolo que atendesse os parâmetros de velocidade, segurança e confiabilidade nas trocas de informação.

Vendo esta necessidade Robert Bosch desenvolveu um protocolo chamado CAN – *Controller Area Network*, de comunicação serial para a aplicação em tempo real baseada na norma ISO 11898 e ISSO 11519-2, para a utilização em redes de comunicação serial em veículos, tendo recentemente evoluído para comunicar sensores discretos. O CAN consiste basicamente de um padrão de hardware com diferentes tipos de frames, regras de transmissão de dados e regras para detectar e corrigir erros, sua especificação define a camada física e o enlace do modelo de referência OSI/ISSO.

A figura 2.1 mostra um esquemático de um veículo inteligente tendo distribuídos os módulos de comunicação, sensoriamento interno e externo, interface entre homem e máquina e finalmente os sistemas mecatrônicos embarcados.

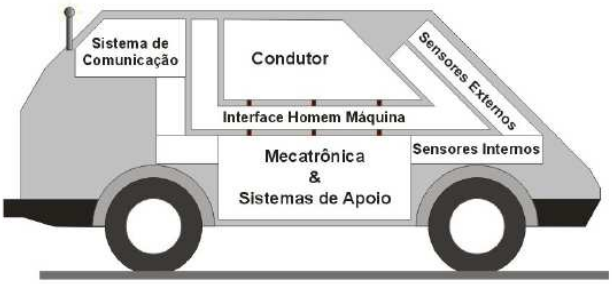


Figura 2.1 – Esquemático de um veículo inteligente. (Osório F. S., 2004)

Todos estes avanços contribuíram de forma significante para o aumento da segurança nos veículos devido os novos padrões e conceitos de segurança destes últimos anos.

2.1.4. Sistemas Inteligentes Veiculares

Um ponto importante é a introducão de sistemas inteligentes na industria automobilistica. Neste caso, na literatura o termo “Sistema Inteligente” é mais aplicado na introdução de novas tecnologias (eletrônica embarcada, sistema microprocessados, etc.) para viabilizar técnicas que aumentem a segurança, conforto e aspectos ambientais.

Nas últimas décadas os veículos de passeio têm deixado de ser máquinas essencialmente mecânicas e incorporado cada vez mais sistemas eletrônicos de controle e acionamentos controláveis usando informações desse sistema. A princípio foi uma “simples” substituição do sistema de ignição com platinado pela *Ignição Eletrônica*. O desenvolvimento dos sistemas de ignição eletrônica vem simultáneos com os *Sistemas de Injeção Eletrônica*.

Atualmente, veículos apresentados como carro conceito, dispõem de sistemas completos totalmente *drive-by-wire*. Exemplos destes são o Pivo, da Nissan (CNN (b), 2005) e o PM da Toyota (TOYOTA MOTOR, 2006).

Com o surgimento e a popularização cada vez maior dos conceitos de sistemas eletrônicos embarcados aplicados aos veículos de passeio, o mercado mundial responde cada vez mais de forma positiva, dando suporte para o desenvolvimento de novas técnicas e a realização de projetos e conceitos.

Com o crescimento das pesquisas na área de aplicação e projetos de sistemas surgem novas necessidades tecnológicas para aumentar a segurança e o conforto do motorista e ocupantes. Os projetos de sistemas inteligentes embarcados nos veículos vem evoluindo contribuindo para o desenvolvimento de vários ramos, tecnológico, acadêmicos e de mercado . Segundo (Kelber C. R, et all., 2005), um dos motivos principais do surgimento de carros inteligentes é propiciar ao motorista maior segurança e conforto em condições adversas de guiagem.

Os conceitos mais aplicados nos sistemas inteligentes embarcados nos veículos de passeio são:

- ABS – *Anti-lock-Breaking-System*;
- BAS – *Breaking-Assistant*;
- ESP – *Electronic Stability Program*;

- TCS – *Traction Control System*;
- *X-by-Wire*;
- *Cruise Control*;
- *Lane keeping Assistance*;
- *Parking Assistance*.

Estes sistemas foram desenvolvidos com o objetivo de proporcionar ao motorista maior segurança em condições críticas ou auxiliado na execução de tarefas relacionadas a condução do veículo. Cada dia os veículos inteligentes somam as novas tecnologias formando assim uma plataforma embarcada mais complexa. Para gerenciar todos estes sistemas estão sendo aplicados conceitos estruturais o que tem dado aos projetistas destes sistemas novas direções para projetar sistemas modulares responsáveis por controlar ações do veículo.

A figura 2.2, apresenta uma pirâmide que exemplifica os diferentes níveis de controle e atribuições de acordo com as necessidades e complexidade das tarefas a serem executadas em uma abordagem hierárquica.

Figura 2.2 – Estrutura da Arquitetura em Níveis de controle (Kelber C. R, et al., 2005)

Acompanhando esta tendência, esse trabalho possui uma arquitetura modular onde cada um dos módulos é responsável por ações sendo os módulos subdivididos em blocos formando uma arquitetura distribuída. Estas tecnologias tem apresentado bons resultados, no projeto *Autonomies Fahren*, (Becker, 1998), (Simon, 1999), (Sönitz, 1999), (Simon, 2000) e (Sönitz, 2001).

2.2. SISTEMAS DE AUTOMAÇÃO VEICULAR

Nesta seção são abordados os principais conceitos em sistemas de automação veicular. São apresentados o desenvolvimento e estado do sistemas de apoio ao motorista, bem como, o desenvolvimento da tecnologia em sistemas veiculares robóticos. Ainda nesta seção é descrita uma análise comparativa em os principais conceitos dos diferentes sistemas em automação veicular.

Nos últimos anos as pesquisas focadas na área de veículos inteligentes têm gerado conceitos que estão revolucionando o mundo no que se diz respeito a transporte pessoal. As linhas de pesquisas são as mais diversas possíveis, mas em comum, focam pontos como segurança, conforto, cuidados ambiental e adaptabilidade para diferentes circunstancias (por exemplo, soluções veiculares voltadas para deficientes). Por outro lado, os novos desenvolvimentos buscam cada vez mais substituir as soluções mecânicas por soluções eletromecânicas (Kelber, 2003a).

Nesta seção serão estudados os diferentes sistemas de automação veicular. Do ponto de vista dos objetivos e das técnicas aplicadas o sistemas de automação veicular podem ser divididos em: (a) Sistemas de Apoio ao Motorista (vide seção 2.1.5) e (b) Sistemas Veiculares Robóticos (vide seção 2.1.6). No primeiro caso, as técnicas tem como objetivo auxiliar o motorista na execução de tarefas determinadas melhorando o desempenho do usuário e a segurança. No segundo caso, trata-se de resolver o problema conhecido na literatura como *Hands-free Vehicle Driving* (Giove D., 2004). No caso do sistema envolver todos os controles parciais do veiculo o termo usado na literatura é *Automated Highway System* (AHS) (Han_Shue, et al., 1999).

2.2.1. Sistemas de Apoio ao Motorista

Os sistemas de apoio ao motorista surgiram na década de 80 com o intuito de proporcionar maior conforto e segurança ao motorista e os passageiros. São classificados em dois grandes grupos: *sistemas passivos* e *sistemas ativos* (Kelber C. R, et al., 2005).

No sistema passivo a eletrônica identifica os sinais das variáveis que indicam um cenário e alerta o motorista, que toma as devidas providências com a finalidade de contornar a situação.

Já no caso do sistema ativo ele funciona como um “co-piloto eletrônico” e em casos de situações perigosas ele atua diretamente sobre as variáveis para efetuar a correção necessária para auxiliar o piloto a contornar o problema. O princípio deste sistema nunca tira do motorista o controle do veículo, ou seja, mantém a responsabilidade do condutor atuando como auxiliar em situações de perigo.

Atualmente, no mercado é possível encontrar sistemas de apoio ao motorista dentro da arquitetura passiva. Os principais são: auxílio a estacionamento em vaga paralela, identificador de obstáculos em pontos cegos do carro, aviso de abandono de pista, sistema de auxílio na navegação e sistema de comunicação inter-veicular.

O auxilio de estacionamento em vaga paralela tem como princípio fundamental a utilização de uma câmera de vídeo instalada na traseira do veículo a qual gera imagens num monitor instalado no painel do veículo. O motorista pode então visualizar estas imagens e por meio as superposição de imagens pode manobrar o veículo até o ponto ideal para estaciona-lo na vaga.

Na figura 2.3 é possível visualizar dois destes sistemas instalados em veículos japoneses e europeus.

Figura 2.3 – Sistema de auxílio para estacionamento. (Osório F. S. e Heinen, F.; Fortes, L., 2000)

A identificação de obstáculos nos pontos cegos ao motorista também baseiam-se nas imagens geradas por câmeras ou por *scanners* a sistemas laser instalados na lateral do veículo. Este tipo de sistema auxilia os motoristas de grandes caminhões a executarem manobras de estacionamento (ou mesmo durante a condução do veículo) quanto a aproximação de outros veículos, pedestres ou obstáculos. Na figura 2.4 mostra a aplicação em carros de passeio leves e veículos de grande porte.

Figura 2.4– Sistema de identificação dos pontos cegos do veículo. (Osório F. S. e Heinen, F.; Fortes, L., 2000)

Os sistemas de identificação de obstáculos como visto podem ser aplicados em veículos de pequeno e grande porte. Um sistema muito útil para a segurança dos motoristas e passageiros em condições de pouca visibilidade ou de cochilo do motorista é o *Sistema de Aviso do Abandono de Pista*. Fontes de estudo das causas dos acidentes apresentam um elevado número de acidentes causados por motivos relacionados a perda de controle da direção do veículo pelo abandono da pista (Kelber, 2003a).

Outro sistema muito utilizado atualmente com a popularização cada vez maior dos GPS – *Global Positioning System*, são os de navegação. Baseados em mapas digitais das regiões estes sistemas funcionam como um guia instantâneo para o motorista auxiliando na definição das rotas a serem percorridas para alcançar seu destino. Atualmente, muitos destes sistemas estão ligados a sistemas mais complexos, o que possibilita ao motorista obter informações das condições de trânsito, se o combustível que está no tanque é suficiente para chegar até o destino, e oferece rotas alternativas mais rápidas e seguras (Kelber, 2003a).

Os sistemas de comunicação inter-veicular surgiram a partir da evolução dos sistemas de comunicação e informação. Estes foram desenvolvidos com o intuito de informar o motorista a ocorrência de acidentes, a aproximação do veículo de outro, agindo no controle da velocidade e impedindo colisões na traseira. Também, quando integrado com o sistema de navegação, permite que exista uma interação entre os veículos próximos, tornando possível um trafego de forma cooperativa e muito mais seguro, minimizando os riscos de acidentes e engarrafamentos. Na Alemanha este sistema tem sido utilizado nas rodovias de grande porte (Han-Shue, et al., 1999).

Nos sistemas ativos de apoio ao motorista o funcionamento pode ser chamado de "co-piloto" auxiliando o motorista durante condições extremas. Tais sistemas não podem evitar a ocorrência de um acidente por si só, mas podem reduzir os riscos dos mesmos ocorrerem pois atuam de forma preventiva a situações que podem oferecer risco. Outra aplicação dos sistemas ativos é a execução de tarefas repetitivas como o estacionamento em vagas paralelas.

O sistema de frenagem de emergência baseia-se em informações de sensores de proximidade, radares para atuar o sistema de frenagem milisegundos antes de uma colisão eminente, podendo minimizar a gravidade da colisão ou até mesmo evitá-la. Imediatamente é enviado para o *airbag* um sinal para aciona-lo antes da colisão. O principal desafio desse sistema é identificar a eminência de uma possível colisão.

Os sistemas de estacionamento automático tem sido muito aplicados, pois esta é uma tarefa que para muitos motoristas é de grande dificuldade. Neste caso, sensores de proximidade são instalados no veículo que identifica a vaga e executa as manobras necessárias para estacionar o veículo na vaga, sem colidir com os outros automóveis estacionados, deixando o veículo próximo ao meio fio.

Para a realização de tarefas autônomamente por veículo de forma ativa ou passiva existe um sistema de processamento e controle interligando aos transdutores. Estes sistemas possuem diferentes níveis de inteligência embarcada até porque o que se deseja é obter um veículo de execute o controle de navegação de forma autônoma. Desta forma é necessário chegar a um sistema de automação veicular complexo, com diferentes níveis hierárquicos.

Em (Han-Shue, et al, 1999) é apresentada uma proposta de automação veicular baseada em pontos magnéticos instalados nas rodovias. Magnetrômetros foram instalados na parte inferior do pára-choque dianteiro e traseiro do veículo, veja figura 2.5, os sinais dos transdutores são enviados ao controlador que realiza a ação na direção, freio e acelerador.

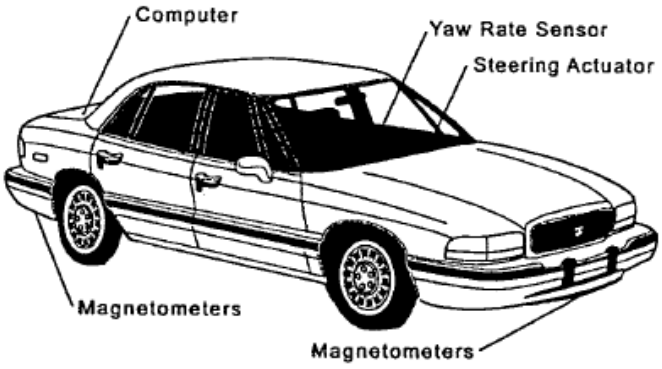


Figura 2.5– Sistema de Automação de pilotagem do veículo baseados em pontos magnéticos nas rodovias. (Han-Shue, et al, 1999)

Os resultados apresentados mostram que o sistema é robusto e seguro, mas possui um alto custo de implementação dos pontos magnéticos nos milhares de quilômetros das rodovias.

A proposta apresentada por (Shimazaki, 2004) consiste em um assistente para estacionamento. É classificado como um sistema passivo onde seu funcionamento está baseado em ajudar o motorista na tarefa de estacionamento por meio de linhas sobre postas nas imagens obtidas por uma câmera instalada na traseira do veículo. Sistema similar ao mostrado na figura 2.3.

A proposta do sistema de apoio ao motorista se mostra segura e de baixo custo mas sua robustez é comprometida quando a iluminação do ambiente é baixa e também depende da habilidade do motorista em executar as manobras.

O sistema proposto por (Wada M., 2003) faz uma comparação entre os sistemas de assistência ao motorista e os sistemas automatizados de estacionamento. O sistema de estacionamento proposto nesse trabalho possui alto nível de sofisticação pois é capaz de auxiliar o motorista na tarefa do estacionamento em diferentes posições do veículo, vaga paralela á direita, vaga paralela a esquerda, garagem.

Na proposta de Wada, destaca-se a interface do sistema com o motorista. O sistema se mostra robusto, seguro, mas complexo. Necessita de capacidade de processamento alta bem como armazenamento de informações. São aplicados paralelamente técnicas de posicionamento global por meio de GPS, *encoders* para identificar o seu deslocamento, e transdutores para captarem o ângulo das rodas.

2.2.2. Sistemas Veiculares Robóticos

A robótica é uma área que tem se desenvolvido rapidamente nos últimos anos. Inicialmente os autômatos tidos como verdadeiras obras de arte eram responsáveis por executarem tarefas repetitivas. A geração seguinte de sistemas robotizados foram os manipuladores robóticos de base fixa que são largamente utilizados nas indústrias (e.g. indústria automotiva).

Recentemente surgiram os robôs móveis destacando-se por sua mobilidade guiada, semi-autônoma ou totalmente autônoma. A tecnologia dos robôs móveis tornou-se grande foco de pesquisa sendo difundida em diferentes áreas e em diversas aplicações (e.g. exploração de ambiente hostil). Pesquisas também tem se intensificado na aplicação dos conceitos da robótica móvel em veículos de passeio.

Dentre os projetos que tem se destacado estão os participantes do *Grand DARPA Challenge 2004 – Autonomous Ground Vehicles*, este evento vem ocorrendo anualmente e tem como objetivo vencer uma distância de 300 quilômetros de forma completamente autônoma, a equipe vencedora recebe um prêmio de 1 milhão de dólares.

O setor de carga, descarga e transporte tem se destacado nos investimentos de apoio ao motorista. Projetos de carregadeiras, empilhadeiras autônomas foram desenvolvidos pelas empresas FOX GmbH e Goetting KG (ver figura 2.6).

Figura 2.6– Veículos de carga e descarda robotizados (Goetting, et al, 2001)

Outro projeto na área de sistemas veiculares robotizados foi realizado na Alemanha pela Volkswagen e consistiu no projeto de um robô (ver figura 2.7) capaz de executar a tarefa de dirigir um veículo por meio de um complexo sistema de sensoriamento do estado do veículo bem como analise de imagens capturadas por um sistema de visão computacional dedicado.

Figura 2.7– Projeto *Autonomes Fahren* (Volkswagen) – Robô motorista (Osório F. S., 2004)

Os sistemas veículares robotizados tem mostrado sua eficiência no que tange o aumento de segurança e conforto mas tem gerado o conflito de poder ir a qualquer lugar a qualquer hora de forma livre. A evolução dos sistemas veiculares robóticos vem trazendo um aumento significativo das tecnologias embarcadas nos veículos. O número de componentes eletrônicos vem aumentando expressivamente bem como o número de microcontroladores e sensores (Osório F. S., 2004).

2.2.3. Analise dos Diferentes Sistemas de Automação Veicular

A tabela 2.1 mostra uma comparação entre os sistemas de automação veicular estudados neste trabalho. Para analisá-los foram compará-los aspectos de implementação, características do controle, flexibilidade do sistema implementado, arquitetura entre outros.

Em (Good et al, 1988), foi proposto um *retrofit* num veículo de passeio com a finalidade de que o mesmo efetuasse manobras de estacionamento de forma automática. O sistema é composto por um circuito hidráulico que é comandado por um controlador. Esse sistema atua sobre o ângulo da direção do veículo e é classificado como assistente passivo pois auxilia na execução da tarefa de estacionamento quando é acionado.

A proposta de (Steven E. S. Charles A., 1991) aplica conceitos de AVC – *Automatic Vehicle Control* na implementação de PATH – *Program on Advanced Technology for Highway*. O princípio básico do sistema consiste no controle da direção e velocidade do veículo em sistemas rodoviários automatizados.

Tabela 1 – Principais Sistemas de Automação Veícular Implementados

Características	Good et al, 1988 (*)	Steven E. S. Charles A. (1991)	Han-Shue, et all, 1999	Wada M, 2003	Shimazaki et al., 2004 (*)	Tanaka et al, 2006 (*)
Controle da Direção	Controle hidráulico	sim	Pontos magnéticos instalados na rodovia	não	Não	Controle eletrônico
Controle da Embreagem	não	sim	não	não	Não	Não
Controle do Freio	não	sim	sim	não	Não	Controle eletrônico
Controle do Acelerador	não	sim	Controla a velocidade pela leitura dos pontos magnéticos da rodovia	não	Não	Controle eletrônico
IHM	não	não	Display de dados	Display para ângulo da direção; Freio; Acelerador.	Sim	Display mostra a imagem das câmeras e o estado do veículo
Arquitetura Modular	não	sim	não	sim	Sim	Sim
Motorização	Motor a combustão interna	Motor a combustão interna	Motor a combustão interna	Adaptável a qualquer tipo	Motor a combustão interna	Motor de combustão interna
Câmbio automático	não	não	Não informado	Adaptável a qualquer tipo	Adaptável a qualquer tipo	Sim
Conceitos *drive-by-wire*	sim	não	não	sim	Não	Sim
Controlador	Microcontrolador	Computador embarcado no veículo	Computador embarcado no porta-malas do veículo	Computador embarcado no porta-malas do veículo	Computador embarcado no veículo	Computador ECU
Sensoriamento	Ultra-som	ângulo das rodas; giro das rodas	Magnetrômetros	ângulo das rodas; giro das rodas; GPS	Ângulo das rodas, giro das rodas	Presença de obstáculos, captura de imagens, ângulo da direção
Unidade de controle	não	Não	não	não	Não	Sim
Flexibilidade	Somente do software	Somente software	Somente do software	Somente do software	Somente de software	Somente de software
Tipo de Sistema	Passivo	Ativo	Ativo	Passivo	Passivo	Passivo

(*) ***Patentes registradas no banco de patentes dos Estados Unidos da América.***

Já em (Han-Shue, et all, 1999), a proposta apresentada é um sistema ativo que combina os conceitos de AHS – *Automated Highway System*, com a utilização de pontos magnéticos distribuídos na pista para guiar o veículo.

O sistema controla a direção do veículo bem como sua velocidade durante o trajeto.

Wada M, 2003, apresenta um sistema de assistente para estacionamento que combina sinais de posicionamento global e hodometria para a definição da posição do veículo. O sistema proposto é passivo, o próprio condutor do veículo executa as tarefas com a ajuda do sistema que indica se o ângulo das rodas está correto e a distância do veículo em relação aos obstáculos.

Em (Tanaka et al., 2006) o sistema proposto utiliza técnicas avançadas de assistência para o estacionamento de veículos. A tecnologia aplicada consiste na sobreposição das imagens num display para o cálculo das variáveis importantes para o estacionamento do veículo em vagas paralelas ou garagens.

2.3. INSTRUMENTAÇÃO VIRTUAL

Nessa seção são apresentados os principais conceitos em instrumentação virtual e suas particularidades.

A instrumentação virtual pode ser entendida como sendo uma solução de medição e automação baseada em computador pessoal, sendo personalizada pelo usuário (Regazzi R. D., et al., 2005)**.**

O Labview – *Laboratory Virtual Instruments Engineering Workbench* (Regazzi R. D., et al., 2005) é o ambiente desenvolvido pela National Instruments que utiliza a linguagem G, projetada para o desenvolvimento de aplicativos como tantas outras linguagens, por exemplo: C, Basic e Delphi.

Embora todas estas sejam direcionadas a objetos (com compiladores que simplificam o processo de programação através de interfaces amigáveis com comandos e funções pré-definidas) existe uma diferença importante entre elas e a linguagem G. Ela foi desenvolvida pela *National Instruments* e possuí o padrão *G* (Gráfico). Sua principal característica é sua forma de programação que é altamente produtiva na construção de sistemas voltados para aquisição de dados, instrumentação, controle e outras aplicações.

A filosofia dos sistemas de programação, normalmente, é fundamentada no uso de linguagem texto com abreviações de palavras da língua inglesa para criar linhas de comandos que, quando processados, geram códigos de programação interpretados ou compilados.

A linguagem G por sua vez, permite que o programador utilize uma interface gráfica para a criação dos códigos de programação em blocos. Isto facilita o processo de aprendizagem permitindo que pessoas mesmo com pouco treinamento sejam capazes de realizar tarefas que em outras linguagens demandariam maior esforço e muito mais tempo (National I., 2007).

Os recursos de depuração do Labview auxiliam o programador a identificar as causas de diversos tipos de problemas na elaboração da programação. Isto permite ao projetista utilizar pontos de parada e animação na execução para visualizar a passagem dos dados através dos blocos. O anterior possibilita verificar os valores das variáveis pontualmente, facilitando o aprendizado e a visualização de possíveis problemas.

Os recursos de depuração do Labview auxiliam o programador a identificar as causas dos erros na compilação. Assim como o Delphi e o Visual Basic são sistemas de programação e desenvolvimento de aplicações de uso geral, o Labview possui diversas bibliotecas de funções e sub-rotinas para a execução de tarefas, principalmente relacionadas à área de simulação, automação e instrumentação.

Os programas em G são chamados de VI – *Virtual Instruments,* são formados por uma interface interativa com o usuário e um diagrama de fluxo de dados onde se encontra o código fonte.

De forma mais específica, a programação gráfica é estruturada da seguinte forma: O *Painel Frontal* permite que o usuário digite os valores de entrada e observe os valores de saída processados pelos blocos. Sendo análogo a um instrumento de medição, as entradas são chamadas de controle e as saídas de indicadores. O projetista pode utilizar uma gama de indicadores e controladores sendo estes referenciados por uma variável que auxilia na localização durante o processo de programação (veja figura 2.8).

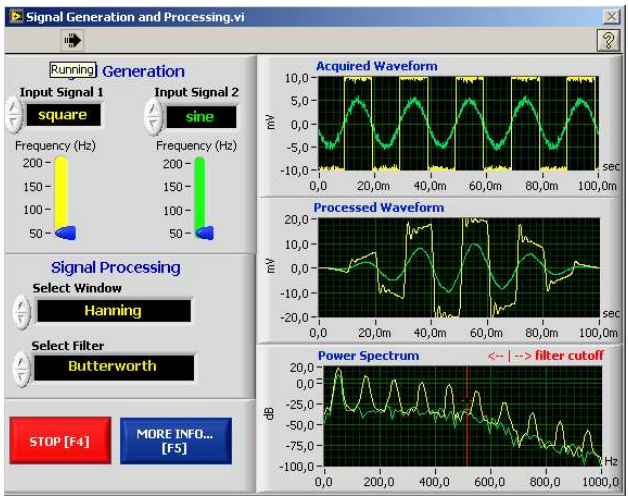


Figura 2.8 – Painel Frontal do Labview

O diagrama de blocos é o local onde fica o código fonte da aplicação e cada painel frontal é acompanhado de um diagrama de blocos. O projetista pode construir novos blocos utilizando os recursos do Labview. Os componentes do diagrama de blocos representam os nós onde as informações transitam seguindo a lógica do projetista. Neste caso, pode ser utilizada estruturas como: ciclos, laços de interações – *looping*, *for loop*, *while loop*, *case*, obedecendo a um fluxo de dados como num fluxograma (veja figura 2.9).

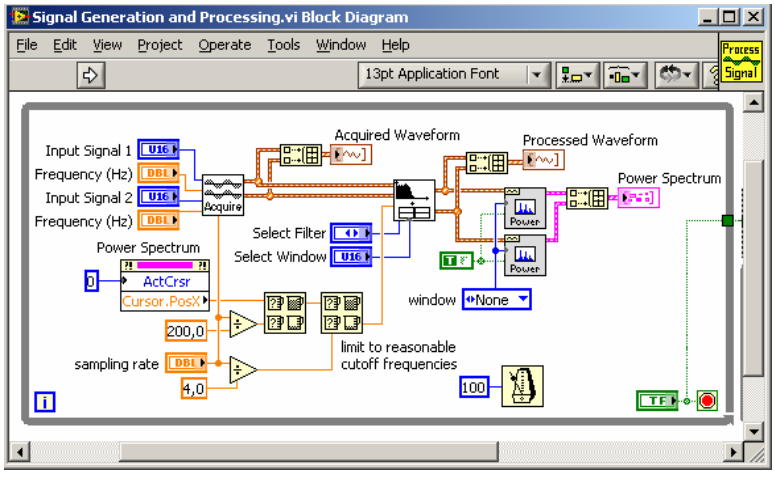


Figura 2.9 – Diagrama de Blocos do Labview

Os recursos de depuração do Labview auxiliam o programador a identificar as causas de diversos tipos de problemas por meio de uma janela que descreve qual o erro e onde se localiza. Outra função útil durante o projeto das aplicações no ambiente do Labview é o *Help Context* – ajuda contextualizada que mostra ao programador quais são as entradas e saídas de cada uma das ferramentas.

2.4. *HADWARE* RECONFIGURÁVEL E SEUS CONCEITOS

Nesta seção são abordados os principais conceitos relacionados a computação reconfigurável como: o funcionamento das FPGA's, especificações do kit de desenvolvimento utilizado neste trabalho, conceitos de sistemas embarcados e a ferramenta de desenvolvimento para embarcar microprocessadores e periféricos nas FPGA's das famílias superiores a Spartan-3.

Uma das motivações deste trabalho é a avaliação do uso de dispositivos reconfiguráveis com microprocessadores embarcados na área de automação veicular. Dada a evolução destes sistemas à possibilidade de se implementar algoritmos diretamente em hardware, embarcar microprocessadores ou implementar sistemas híbridos os quais podem ser compostos de periféricos desenvolvidos diretamente em hardware e ligados ao barramento interno.

A flexibilidade encontrada nos dispositivos baseados em hardware reconfigurável fornece uma alternativa para a implementação de variadas técnicas de controle para os veículos. Desde a introdução do primeiro microprocessador comercial, o Intel 4004, no final do ano 1971, os sistemas digitais têm evoluído de forma considerável. Este pequeno microprocessador, introduzido pela *Intel Corporation*, integrava 2300 transistores em uma única pastilha de silício cujo custo inicial oscilava por volta de duzentos dólares americanos.

A complexidade dos microprocessadores, medida segundo o número de transistores dentro do chip, é dobrada a cada 18 meses desde a aparição do 4004 (Moore, 1997). Com a evolução da tecnologia os circuitos integrados, atualmente integram 20 milhões de transistores por centímetro cúbico (cm^3), podendo atingir 100 milhões até 2012 (Brown e Vranesic, 2000). Uma classificação dos circuitos integrados que permitem a implementação de uma lógica digital é mostrada na figura 2.10 abaixo.

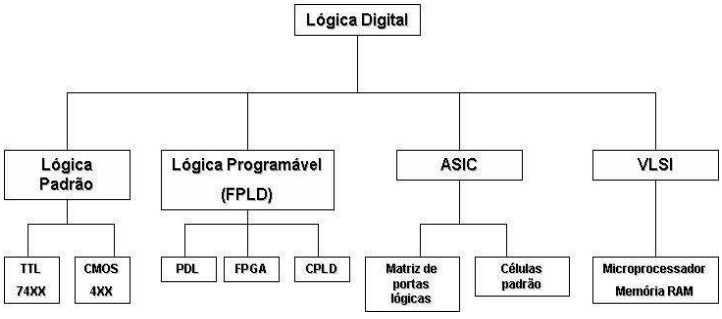


Figura 2.10 – Circuitos integrados – Modificado (Brown e Vranesic, 2000)

Circuitos integrados tradicionais, TTL e CMOS, executam lógicas padrão realizando operações pré-definidas pelo fabricante. Isso faz o usuário, dependendo da aplicação, conectar diversos tipos de circuitos para executar uma lógica específica ocasionando perda de eficiência, aumento de área e alto consumo de energia.

Já os circuitos integrados como os PLD – *Programmable Logic Device* e FPGA possuem operações lógicas internas definidas pelo usuário, enquanto os circuitos como: ASIC – *Application Specific Integrated Circuits* e VLSI – *Very Large Scale Integration* permitem ao usuário projetar a lógica sendo essa implementada pelo fabricante especializado. Este é o caso dos microprocessadores e das memórias RAM – *Random Access Memory* utilizadas nos computadores pessoais, o custo é elevado e a funcionalidade é fixa, não sendo possível a realização de atualizações ou modificações na lógica a ser executada.

2.4.1. Aspectos da Reconfiguração

As arquiteturas reconfiguráveis podem ser definidas como aquelas onde se podem aplicar os conceitos de reconfigurabilidade, citados na seção 2.4. São arquiteturas onde os componentes lógicos básicos que o constituem podem ser facilmente reconfigurados pela interligação de blocos lógicos. Estes componentes lógicos básicos, geralmente são as unidades de processamento, unidades funcionais de processamento, armazenamento, comunicação, entradas ou saídas de dados.

Uma arquitetura reconfigurável possui muito das características dos sistemas computacionais tradicionais, ver seção 2.4.1, mas a organização e implementação ocorrem de maneira muito diferente.

Segundo (Yabarrena J. M. S., 2006), ao invés do processamento de uma função ocorrer por meio de um conjunto de instruções executadas seqüencialmente ao longo do tempo, como em um processador, as arquiteturas reconfiguráveis geralmente processam a função por meio de unidades configuradas em blocos lógicos básicos, como ocorre nas FPGA’s, fato este que caracteriza uma computação paralela, envolvendo diferentes unidades funcionais com as quais obtem-se resultados intermediários.

O espectro atual dos sistemas reconfiguráveis é muito amplo e diversificado, com destaque para os seguintes tipos: co-processadores reconfiguráveis, processadores reconfiguráveis, computadores reconfiguráveis, sistemas embarcados reconfiguráveis e

sistemas híbridos, compostos de módulos reconfiguráveis e módulos não-reconfiguráveis. Uma análise da utilização e da evolução desses sistemas nos indica uma tendência de crescimento na amplitude e na complexidade desse espectro de sistemas computacionais reconfiguráveis e de arquiteturas reconfiguráveis.

Dentro da área de arquiteturas reconfiguráveis existem alguns possíveis problemas importantes que pode-se indicar e tentar analisar futuramente. Estes problemas estão relacionados principalmente com gargalos de desempenho, modos e métodos de programação das aplicações, modos de reconfiguração, complexidade e tempo de reconfiguração, desempenho das arquiteturas, flexibilidade das arquiteturas, adequação ou inadequação das arquiteturas aos problemas e principalmente com os modos eficientes de implementação dos conceitos de arquiteturas e computação reconfiguráveis.

Os sistemas reconfiguráveis são plataformas que permitem que o projetista realize modificações na aplicação. Desta forma, o sistema reconfigurável passa a trabalhar com uma arquitetura projetada exclusivamente para esta aplicação, o que faz com que a eficiência seja maior do que as encontradas em sistemas de uso geral. Isto ocorre porque o *Hardware* é projetado para executar os algoritmos necessários para aquela aplicação especificamente.

A tecnologia de computação reconfigurável é dada por meio das especificações:

- **granularidade** : fina, média e alta ou grossa;
- **topologia dos blocos construtivos básicos reconfiguráveis**: arranjo unidimencional, arranjo bi-dimensional, pipeline, *crossbar*;
- **programabilidade**: única, múltipla;
- **reconfiguração**: estática, dinâmica, parcial, total, local, remota, normal, rápida;
- **aspectos de implementação**: blocos de interligação reconfiguráveis, tipo de rota, topologia interligação e roteabilidade;
- **modelo de computação**: mono-processador, multi-processador, SIMD, MIMD, pipeline, VLIW;
- **modelo de implementação de solução**: reconfigurável, mista, fixa, programável;

- **propósito**: geral ou específico.

Devemos nos lembrar que nas arquiteturas dos sistemas computacionais reconfiguráveis a maioria dos conceitos e níveis de abstração arquiteturais tradicionais (não-reconfiguráveis) como: algoritmos, linguagens, compiladores, sistemas operacionais, arquiteturas, micro arquiteturas podem continuar existindo.

Deste modo, os conceitos e conhecimentos de arquitetura de computadores tradicionais também são muito importantes. Entre os diversos conceitos comuns aos sistemas computacionais tradicionais e os reconfiguráveis, onde pode-se destacar os seguintes tipos: paralelismo no nível de bits; paralelismo no nível de instruções ILP – *Instruction Level Parallelism* como o pipeline, paralelismo no nível de processadores PLP – *Processor Level Parallelism*, SIMD – *Single Instruction Stream Multiple Data Stream* muito utilizado nos laços de repetição e MIMD – *Multiple Instruction Stream Multiple Data Stream* muito usado dentro da e/ou entre as aplicações ou processos (Patterson D. A.; Hennessy J. L., 2000).

2.4.2. Funcionamento da FPGA

Existem diversos dispositivos programáveis em diferentes classes configuráveis com capacidade de funcionar com funções lógicas como: EPROM – *Erasable Programmble Only Memory*, PLA – *Programmable Logic Array*, PAL – *Programmable Array Logic*. Devido as necessidades mercadológicas de se projetar funções cada vez mais complexas, surgiram os dispositivos conhecidos como CPLD's – *Complex Programmable Logic Devices*, e outros tipos de dispositivos programáveis como o MPGA – *Mask Programmable Gate Array*, e o FPGA – *Field Programmable Gate Array*.

O FPGA é um *hardware* programável, ou seja, o projetista pode alterar a sua configuração sem desmontá-la do circuito ou retira-la do local instalado. Sua arquitetura estrutural é formada por uma matriz de blocos lógicos reconfiguráveis. Um conjunto de *Slaces* quando configurado e interligado executa operações computacionais. Para tal o projetista realiza a descrição do *hardware* a ser configurado, logo após esta descrição é compilada e obtem-se um arquivo de *bits* chamado de *bitstream* dos dispositivos configurados.

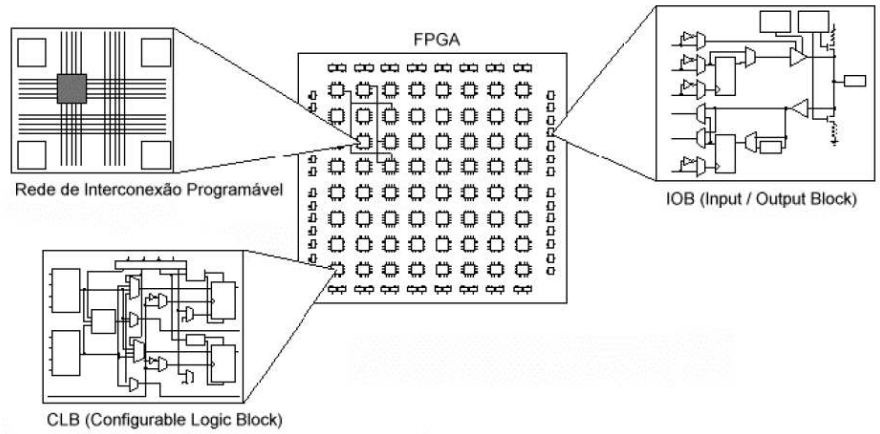


Figura 2.11 – Arquitetura interna de uma FPGA (Compton, K.; Hauck, 2002)

De modo geral as FPGA's possui uma arquitetura interna, como mostrado na figura 2.11, composta por uma matriz de blocos lógicos configuráveis chamados CLB (*Configurable Logics Block*) envoltos por uma rede de conexões programáveis.

Na periferia do circuito estão os blocos de entrada e saída – I/O *Blocks*, que também são programáveis. A arquitetura dos CLB pode variar de acordo com a família e fabricante, mas geralmente são formadas de pontos de entrada ligadas a blocos de funções puramente combinacionais chamados de LUT – *Look-up table*, multiplexadores e registradores que geralmente são *flip-flops*.

Uma FPGA pode ter seu comportamento redefinido de tal forma que sistemas completamente diferentes podem ser implementados na mesma pastilha. As FPGA's com granulidade mais fina permitem que o circuito seja definido no nível de portas lógicas, trabalhando-se em operações com até um bit de largura. Nas FPGA de granulidade grossa não é necessário informar detalhes no nível de portas lógicas, mas as operações em nível de palavras que variam na medida em que varia a granularidade.

Atualmente, está área apresenta-se como uma tecnologia extremamente interessante em aplicações industriais e acadêmicas. Sendo inúmeras as aplicações de computação reconfigurável em telefones celulares, controladores de dispositivos presentes em veículos terrestres, aeronaves, *Hardware* de alto desempenho para diversas aplicações, robótica móvel.

O arquivo contendo o *bit stream* é o responsável por determinar a função que o *hardware* irá desempenhar a partir do momento que o mesmo é configurado ou reconfigurado.

No momento da configuração, são especificadas, além da função que cada elemento reconfigurável irá desempenhar, as portas de entrada ou de saída de cada elemento reconfigurável da matriz. Desta forma, esta configuração das portas gera um outro tipo de configuração que é a configuração do roteamento dos dados.

Este roteamento é de grande importância para o desempenho e utilização do dispositivo reconfigurável, pois quanto melhor o roteamento, melhor a utilização da área do dispositivo reconfigurável e melhor o desempenho conseguido na execução das funções configuradas no dispositivo. É essa configuração do roteamento que vai influenciar o tempo que o dispositivo vai levar para responder a um sinal numa entrada e gerar uma saída. É nessa fase de roteamento, onde questões como: interferência de sinais, quando a integridade de um barramento pode ser comprometido devido à freqüência que o dispositivo esteja utilizando ser alta e interferir nos valores dos bits que trafegam por esse caminho.

Uma FPGA possui uma arquitetura interna como mostrado na figura 2.4, composta de uma matriz com milhares e em alguns casos milhões de blocos lógicos configuráveis CLB – *Configurable Logic Block*, cercado por uma rede de interconexões programáveis distribuída por todo o *chip* formando uma rede de interconexões configuráveis. Existem os blocos de entrada e saída IOB – *Input Output Block*, que também são programáveis, e que servem como interface do dispositivo.

A arquitetura com que a FPGA é roteada podendo ser entendida como sendo a maneira como os comutadores programáveis e os segmentos de trilha são posicionados de forma a permitir a interconexão dos blocos lógicos uns com os outros.

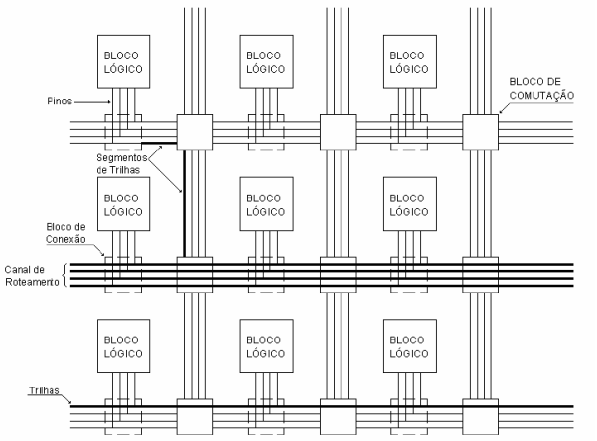


Figura 2.12 – Arquitetura geral de roteamento de uma FPGA (Compton, K.; Hauck, 2002)

Esta pode ser descrita a partir de um modelo geral como mostra a figura 2.12. Para melhor entendermos este modelo são necessários alguns conceitos:

a) **pinos**: são as entradas e saídas dos blocos lógicos, sendo importante ressaltar que são para ligação interna do FPGA, não podendo ser confundidos com os pinos externos do encapsulamento que são ligados aos blocos de I/O;

b) **conexão**: são as ligações elétricas entre um pino e um segmento de trilha, onde as conexões são realizadas pelos blocos de conexão.

c) **blocos de conexão**: são responsáveis por ligar eletricamente um pino e um segmento de trilha, estes dispositivos possuem tecnologia de programação.

d) **segmento de trilhas**: São os fios entre os blocos de comutação.

e) **trilhas**: São seqüências de segmentos direcionados, estendidos por todo o canal de roteamento compostos por segmentos de tamanho variável.

f) **blocos de comutação**: São utilizados para a conexão de dois segmentos de trilha e também possuem tecnologia de programação.

g) **canal de roteamento**: É a área entre duas linhas e colunas de blocos lógicos, sendo o canal formado por muitas trilhas paralelas.

Internamente, as FPGA's contém cópias do mesmo elemento lógico (LB) básico organizados matricialmente. Cada bloco lógico possui um número pequeno de entradas e uma saída. Dentro dele são encontradas pequenas células formadas por um ou dois *flip-flops* onde é possível armazenar valores de "0" ou "1". Os tipos de blocos lógicos mais comumente encontrados são baseados em LUT – *LookUp Table*, que por meio do controle de um grupo de multiplexadores e portas permite o fluxo de dados desde as células armazenamento até a saída do bloco lógico na função lógica desejada (Compton, K.; e Hauck, 2002).

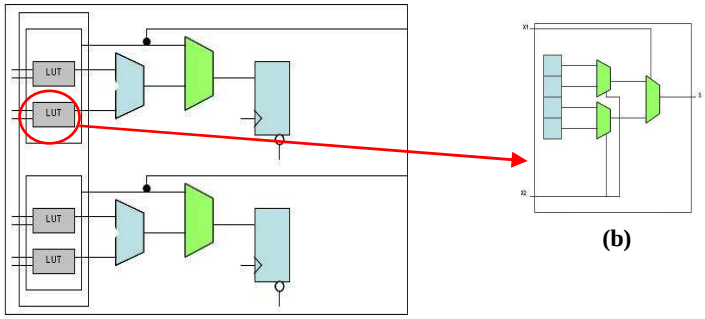


Figura 2.13 – Estrutura de um bloco lógico da FPGA Stratix II
(a) Bloco lógico Altera StratixII, (b) LUT de duas entradas e uma saída

O arranjo de células dentro da LUT é utilizado para armazenar a tabela verdade da função lógica a ser programada. A figura 2.13a mostra a estrutura de um bloco lógico da FPGA Stratix II da Altera, a figura 2.13b mostra um exemplo de LUT de duas entradas e uma saída.

Para realizar operações mais complexas, os elementos lógicos podem ser conectados uns com os outros por meio de chaves de interconexões programáveis. As FPGAs de última geração possuem canais de roteamento em planos diferentes, em outras FPGAs utilizam-se três planos (3D) o que permite um roteamento mais eficiente, tempos de programação menores entre os elementos lógicos e diminuição do consumo de potência (Mingjie et al., 2006).

A capacidade das FPGAs é definida pelo número de elementos lógicos ou LUTs. No gráfico da figura 2.14, vemos que a evolução desta tecnologia, em termos de capacidade e desempenho tem crescido de forma considerável nos últimos 8 anos (Taghavi et al., 2004).

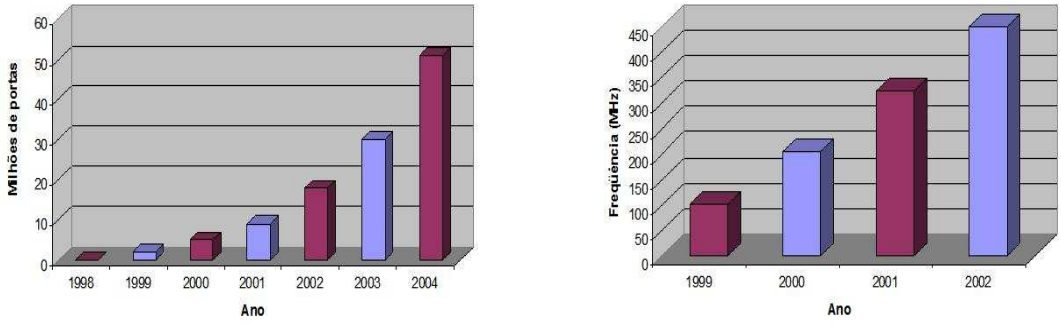


Figura 2.14 – Gráfico da evolução das FPGAs (Taghavi et al., 2004):
(a) Evolução em número de Gates; (b) Aumento da freqüência do relógio.

O mercado de fabricação das FPGAs atualmente é liderado por duas empresas: *Altera Corporation* e *Xilinx Corporation*, embora outros fabricantes também atuem neste mercado, sendo a contribuição destes é muito pequena. O campo de atuação das FPGA é grande e tem crescido cada vez mais, encontrando-se aplicações industriais, automotivas, telecomunicações, processamento de imagens, dispositivos médicos, entre outros produtos de consumo (Taghavi et al., 2004). Uma aplicação em particular das FPGAs é a possibilidade de reconfiguração dinâmica, o que visa aplicações revolucionárias num futuro próximo em aplicações gerais (Harteinsten, 2002).

No contexto deste trabalho, foram realizadas implementações numa FPGA da *Xilinx* da família Spartan-3, pois suas características de desempenho, capacidade e o seu custo são adequadas e atendem os requisitos da aplicação em questão. As placas de desenvolvimento baseadas na Spartan-3 (Digilent, 2006) possuem além do custo baixo, ótimos recursos para *interface* (PS2, porta serial).

Em termos gerais, as aplicações no mercado dos semicondutores mantêm uma tendência regular mudando o foco quando as novas tecnologias aparecem no mercado. Esta avaliação foi introduzida por Tsugio Makimoto em 1989 e está apresentada na figura 2.13 (Makimoto, 2002).

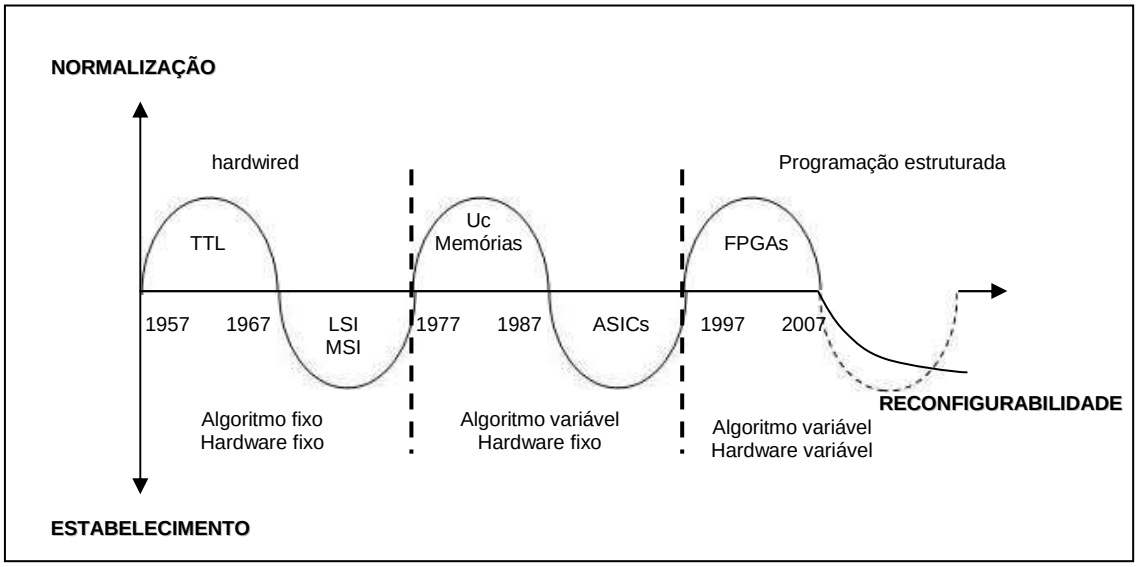


Figura 2.15 – Forma de onda de Makimoto, Modificado – (Makimoto, 2002)

Atualmente, as tendências tecnológicas preparam uma nova mudança. Porém, alguns estudos afirmam que não haverá uma próxima transição e sim um estabelecimento lento e prolongado que considera tecnologias avançadas na área de Sistemas Embarcados e Computação Reconfigurável (M. Herz et al., 2002).

2.4.3. Especificações Técnicas da Placa de Desenvolvimento

A placa de desenvolvimento Spartan-3 (utilizada neste trabalho) fabricada pela Xilinx em conjunto com a Digilent, fornece uma plataforma para o desenho e aplicação de diversos propósitos em *hardware*. Dispositivo FPGA XC3S200 com 200.000 gates em um encapsulamento BGA – *Ball Grid Array* de 256 pinos, da família Spartan-3. Suas principais características são citadas abaixo:

- 4.320 Elementos lógicos, 300 células lógicas equivalentes;
- 20 blocos de memória de 18Kb (216Kb);
- 20 multiplicadores em hardware 18x18 bits;
- 2 Mbit de memória *flash* (XCF02S);
- 1 Mbyte de memória SRAM – *Static Random Acesses Memory*;
- 01 Porta DB9 de comunicação RS-232;
- 4 Digital *Clock Managers* que fornecem flexibilidade e controle;
- 01 porta PS/2 para controle e leitura do *mouse* ou teclado.
- 01 porta JTAG para configuração e compilação da FPGA;
- 08 chaves tipo *slide-switches*;
- 04 *displays* de 7 segmentos ;
- 08 Led individuais;
- 04 botões tipo *bush-button*;
- 03 portas de expansão com 40 pinos cada.

Na figura 2.16 é apresentada um foto ilustrativa da placa de desenvolvimento Spartan-3 utilizada neste trabalho.

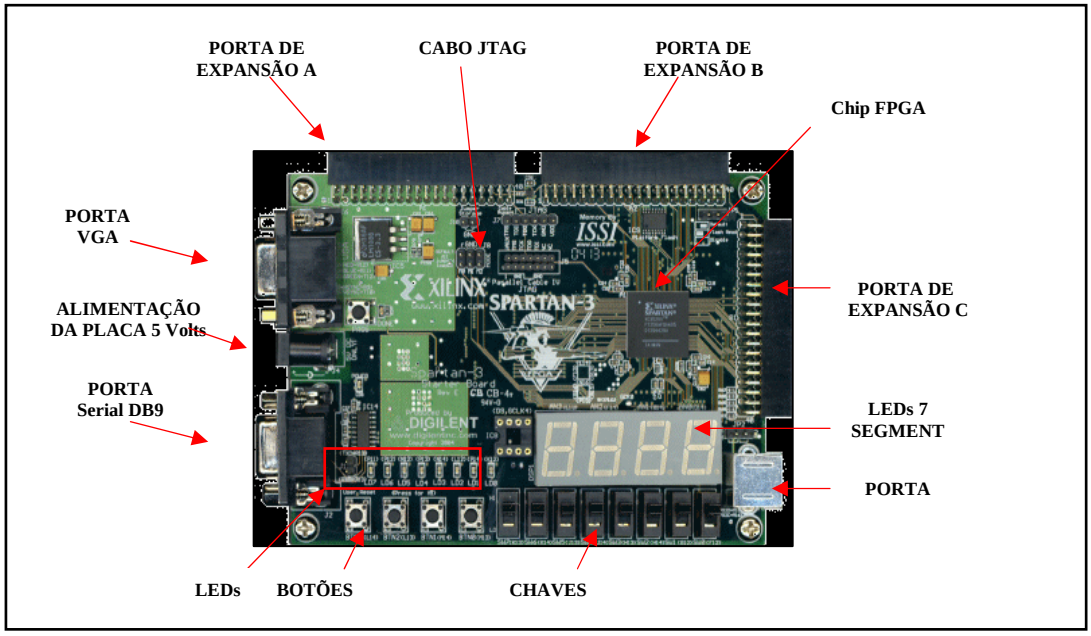


Figura 2.16 – Spartan-3 *Starter kit board* (Digilent, 2002)

Na figura 2.17 é apresentada um desenho esquemático dos recursos da placa de desenvolvimento Spartan-3, citados anteriormente na descrição dos recursos da placa de desenvolvimento.

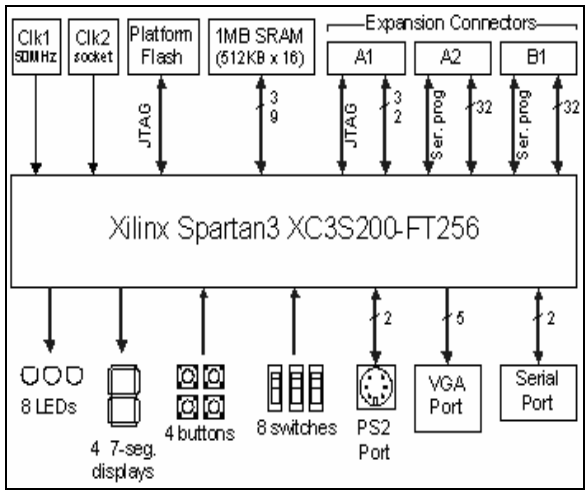


Figura 2.17 – Diagrama em blocos Spartan-3 *Starter kit board*

Nesta figura é possível observar que todos os dispositivos estão ligados ao *chip* XC3C200 que recebe e envia os sinais, fazendo o controle dos dispositivos que estiverem ligados aos conectores de expansão A, B e C.

2.4.4. Processamento Embarcado

O processamento embarcado está presente nos mais variados produtos, equipamentos e aplicações. Desde brinquedos de criança, celulares, automóveis e até em satélites.

O mercado e os centros de pesquisas observando padrões cada vez mais rigorosos têm intensificado os trabalhos de pesquisa nas arquiteturas dos sistemas projetados, os materiais e a estrutura de *hardware* e *software*.

Nem sempre o melhor projeto em termos de velocidade de processamento é baseado somente em uma estrutura do tipo *data flow* ou *signal flow*, conhecidos também como *data stream*, ou seja, quando o *hardware* é pré-definido com dados ou sinais passando por uma estrutura de processamento sem a necessidade de manipulação externa.

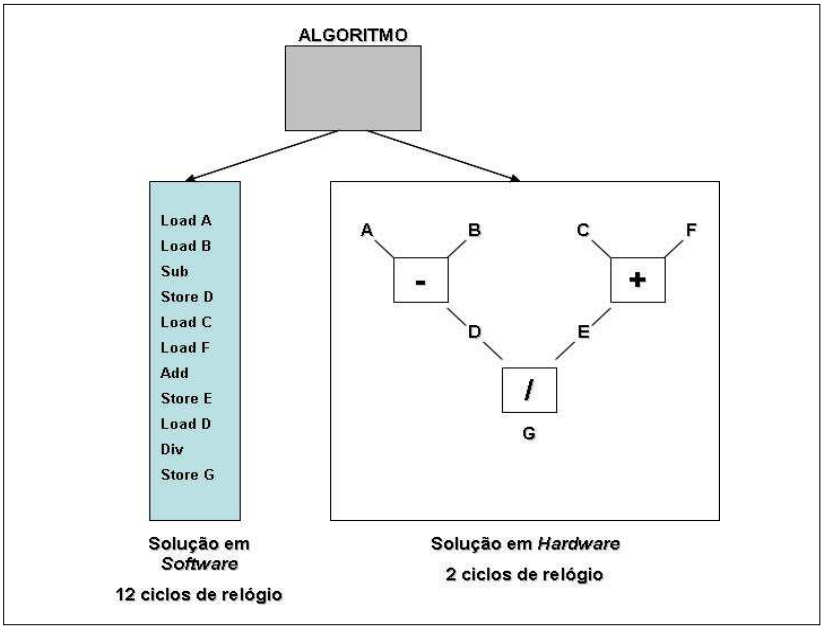


Figura 2.18 – Solução em *Hardware* x *Software* (Willians R., 2006)

Nota-se na figura 2.18 que a solução em *hardware* é mais rápida executando o processamento em 2 ciclos de relógio, o que a solução em *software* leva 12 ciclos de relógio para executar. Porém as aplicações em *hardware* nem sempre podem ser implementadas facilmente fazendo com que as soluções utilize a integração de um processador, realizando a execução de instruções paralelamente a um *hardware* para execução de uma ou mais tarefas de forma dedicada.

Os projetos realizados em *hardware* com vários periféricos, com processador ou não, foi denominado de SoC – *System on chip*, ou no caso das arquiteturas reprogramáveis denominou-se SoPC – *System on programmable chip*.

Diversos fabricantes de FPGA tem trabalhado para disponibilizar no mercado de processadores implementáveis em FPGA's. Esses processadores por sua vez, são chamados de *soft-processors e* configurados como os *soft-cores*, sendo disponibilizados por seus fabricantes. Desta maneira, são classificados como IP's (*Intelectual Properties*).

A tabela 2 mostra os *soft-cores* disponibilizados no mercado pelos principais fabricantes de FPGA .

Alguns fabricantes como, por exemplo, a Xilinx além dos *soft-processors* fornecidos disponibilizam também um *hard-core* do processador *power PC* 405 para a família Virtex 4 e até 2 *cores* para a Virtex II Pro.

Tabela 2 – *Soft-cores* disponibilizados pelos fabricantes

Microprocessador Microcontrolador	**Fabricante**	**Famílias que suportam**
Nios II	Altera	Stratix, Cyclone, Stratix II, Stratix III
Microblaze	Xilinx	Virtex-5 LX, Virtex-5 FX, Virtex-5 SX, Virtex-4 FX, Virtex-4 LX, Virtex-4 SX, Virtex-II Pro, Virtex-II, Virtex-E, Spartan-3E, Spartan-3, Spartan-IIE, Spartan-II
Picoblaze - 8 bits	Xilinx	Virtex-4, Virtex-II Pro, Virtex-II, Spartan-3
8051 High-speed 8 - bits RISC Microcontroller (R80515)	CAST, Inc	Virtex-II Pro, Virtex-II, Spartan-3, Spartan-IIE , Stratix II, Cyclone, Stratix, Excalibur, APEX 20KE, APEX 20KC, ACEX 1K, FLEX 10KE
TMS32025 DSP Processor (C32025)	CAST, Inc	Virtex-II Pro, Virtex-II, Spartan-3, Spartan-IIE
C68000 Microprocessor	CAST, Inc	Stratix II, Cyclone, Stratix, APEX II, APEX 20KE, APEX 20KC, FLEX 10KE, Virtex-II Pro, Virtex-II, Virtex-E, Virtex, Spartan-3, Spartan-IIE
PIC1655x Fast RISC Microcontroller	Digital Core Design	Virtex-4 FX, Virtex-4 LX, Virtex-4 SX, Virtex-II Pro, Spartan-3L, Spartan-3E, Spartan-3, Spartan-IIE

O processador, *power PC* 405, fica embutido diretamente dentro da FPGA da Xilinx atingindo uma performance de até 700 DMIPs à 450 Mhz na virtex 4. Além dos processadores os fabricantes também disponibilizam os *cores* de vários periféricos como: *Timer*, UART, SPI, barramentos, controladores de memória, Ethernet, para o projeto e desenvolvimento de SoC's.

Para a integração dos *cores* os fabricantes disponibilizam suas ferramentas. A Altera oferece o Quartus II em forma de ambiente para o projeto e integração dos *cores* chamado de SoPC *Builder*. Já a Xilinx utiliza a ferramenta EDK – *Embedded Development Kit* para o projeto e desenvolvimento dos sistemas em lógica reconfigurável.

Estas ferramentas são *interfaces* GUI – *Graphical User Interface* onde o projetista não necessita de conhecimentos das linguagens de descrição de *hardware* VHDL – *Very High Description Language* e Verilog. É necessário somente saber instanciar e conectar tais periféricos, utilizando as ferramentas de desenvolvimento e programar as funções do processador.

2.4.5. A Ferramenta EDK – *Embedded Development Kit*

Nesta seção são mostrados as principais características da ferramenta da Xilinx EDK, utilizada neste trabalho.

2.4.5.1.Aspectos da Ferramenta

A utilização dos recursos dos hardwares reconfiguráveis, mais propriamente as FPGA's, por algum tempo estava restrito aos projetistas que trabalhavam com linguagens próprias para a descrição de *hardwares* como VHDL e Verilog. A ferramenta EDK –*Embedded Development Kit* disponibiliza a estes usuários a possibilidade de utilizar seus recursos sem a necessidade de qualquer conhecimento em linguagens de descrição de *hardware*. Para tal esta ferramenta auxilia o projetista a construir e configurar um microcontrolador ou um conjunto de microcontroladores Microblaze e seus periféricos.

O EDK gera arquivos com descrições do sistema que são utilizados para gerar os desenhos após a etapa de sínteses. Está composto por vários recursos para configurar o *hardware* e o projeto de diversos periféricos específicos. A configuração do Microblaze requer trabalhar com vários arquivos diferentes. Basicamente, para projetar o Microblaze é necessário trabalhar com três tipos de arquivos diferentes: arquivos de configuração de *hardware*, arquivos de descrição de *software* e os periféricos.

Para configurar um microprocessador de acordo com as necessidades o projetista pode utilizar o auxiliador que vai solicitando ao projetista as informações de configuração desejadas e ao final entrega o *hardware* configurado e pronto para ser programado e utilizado. Mas também possibilita ao projetista realizar todas as configurações do sistema de forma manual.

Na figura 2.19, pode-se observar uma das telas do auxiliador de configuração do Microblaze.

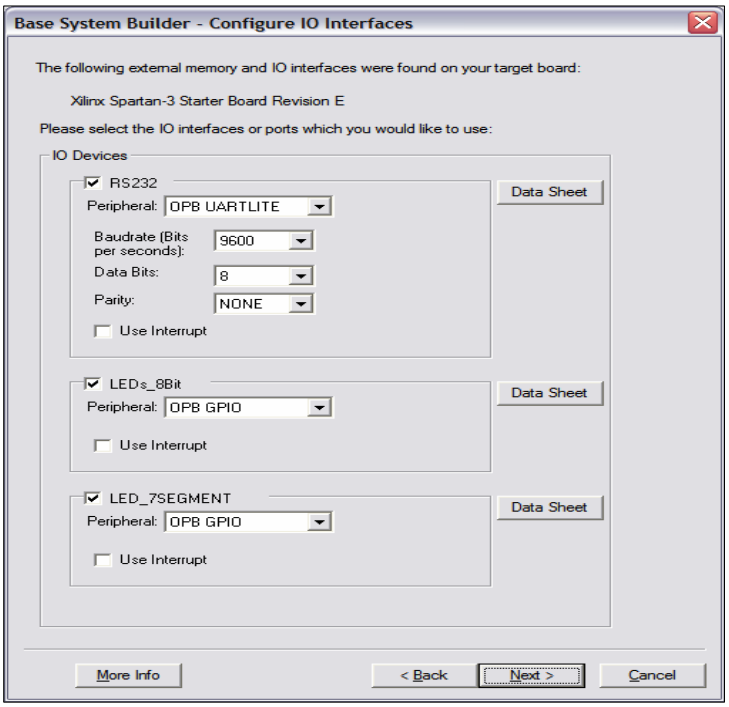
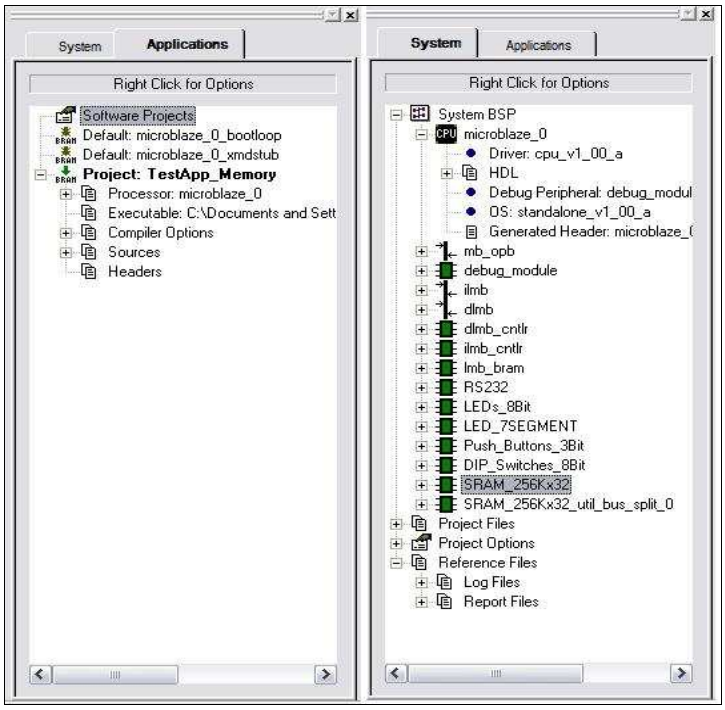


Figura 2.19 – Configuração das entradas e saídas de dados no Microblaze

Nesta tela apresentada na figura 2.19, o projetista pode selecionar os periféricos das entradas e saídas do microcontrolador, como: a interface RS-232, os LED's 8 *bits*, Display de 7 segmentos.

Após realizar as configurações o projetista já possui um *hardware* pronto para ser programado. Na parte esquerda da tela da figura 2.21, o usuário pode visualizar e editar os periféricos que foram configurados na aba *system* onde ficam todas as informações do *hardware*, figura 2.20a. A programação por sua vez, pode ser acessada, editada e configurada na aba *applications* onde o projetista pode visualizar editar e compilar o software de programação do microprocessador, veja figura 2.20b.



(a) **(b)**

Figura 2.20 – Ambiente de configuração do EDK – Configuração de *software*

Na figura 2.21, pode-se observar o ambiente para de adicionar e editar a especificação do *hardware*, suas ligações, suas portas, endereços, periféricos e parâmetros.

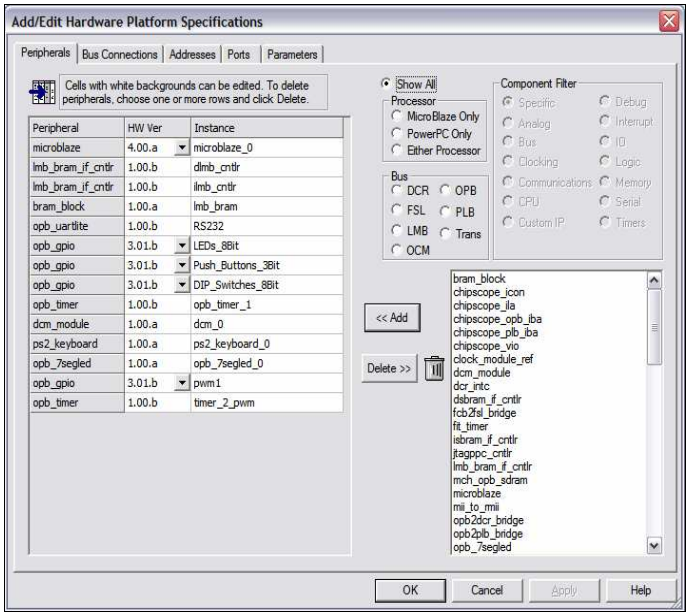


Figura 2.21 – Editor das especificações do Microblaze

Na aba mostrada da figura 2.21, pode-se acessar as configurações das concessões com o barramento, os endereços, as portas e os parâmetros.

2.4.5.2.Microprocessador MicroBlaze

A crescente utilização dos sistemas embarcados tem proporcionado uma evolução desta metodologia. Conceitos de processamento em lógica reconfigurável estão conquistando grande espaço nas aplicações em pesquisas e mercadológicas e apresentam bons resultados.

Diferentes ferramentas EDA – *Electronic Design Automation*, foram usadas para síntese das configurações na FPGA, sendo descritos comumente em VHDL. As FPGA's são aplicadas na implementação tipos diferentes de algoritmos, sendo muito atraente por causa do crescimento do seu uso como *hardware* de uso específico no lugar de ASIC e processadores de aplicações gerais.

O módulo de controle foi projetado na ferramenta de EDK, (EDK MicroBlaze tutorial, 2005) na qual o processador de Microblaze é o centro do sistema **(veja capítulo 4).** Este processador tem uma arquitetura RISC – *Reduced Instruction Set Computer*, com 32 bits registradores de uso geral, uma Unidade de Lógica de Aritmética (ULA), uma unidade de troca, e dois níveis de interrupção.

O Microblaze possui características bem definidas para todas as versões, abaixo são listadas as principais:

a) 32 registradores de propósito geral de 32 bits e registradores de propósito geral: PC – *Program Counter*, MSR – *Machine Status Register*, EAR –*Exception Address Register*, ESR – *Exception Status Register*, ESS – *Exception Specific Status* e FSR – *Floating Point Register*;

b) palavras de instrução de 32 *bits* com 3 operandos e 2 modos de endereçamento;

c) barramento de endereço de 32 *bits*;

d) *pipeline* de fluxo simples;

e) suporta *reset*, *interrupt*, *user exception*, *break* e *hardware exceptions*;

f) memória *cache* para dados e instrução;

g) *interface* de *debug* que suporta *softwares* de *debug* via JTAG.

A figura 2.22 é um diagrama que descreve o *core* (arquitetura) do Microblaze. Os blocos que estão na cor cinza são opcionais sendo configurados pelo projetista de acordo com suas necessidades.

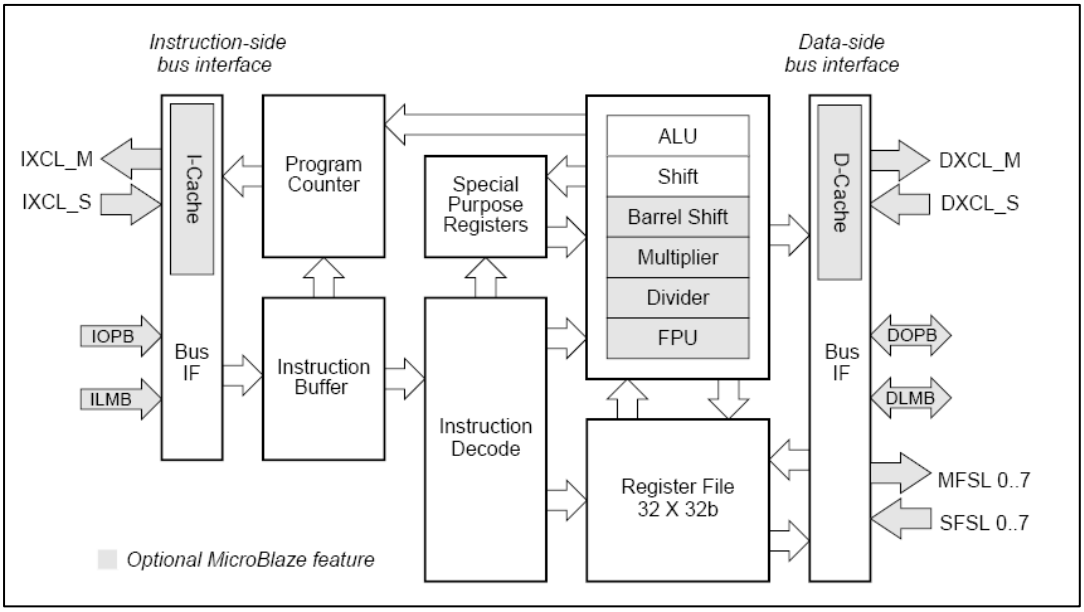


Figura 2.22 - Diagrama de blocos Microblaze (EDK MicroBlaze tutorial, 2005)

Os blocos em branco são as funções básicas que são oferecidas ao projetista como recursos mínimos para o funcionamento do sistema.

Além destes recursos, o Microblaze possui uma entrada para interrupção externa. Para o gerenciamento de diversas interrupções um periférico externo deve ser conectado ao sistema OPB – *Interrupt Controller*. Após a síntese, o Microblaze ocupa aproximadamente 900 a 2600 LUTs – *Look-up tables*) dependendo do nível de parametrização. O número de Microblaze que podem ser embarcados em uma FPGA depende exclusivamente de sua capacidade.

A última versão disponível no mercado do Microblaze é a *v 5.0*, que além das características comuns a todas as versões, possui diversas opções parametrizáveis conforme as necessidades para as quais o mesmo se dedica. A tabela 3 ilustra as opções de configuração para o Microblaze.

Tabela 3 - Opções de configuração das versões do Microblaze

Características	Versão do Microblaze			
	Versão 2.10	**Versão 3.00**	**Versão 4.00**	**Versão 5.00**
Estado da versão	Depreciada	Depreciada	Depreciada	Ativada
Processador do canal de comunicação	3	3	3	3
Barramento de periférico no circuito (OPB) dados de apoio à interface	Opcional	Opcional	Opcional	Opcional
Barramento de periférico no circuito (OPB) instruções de apoio à interface	Opcional	Opcional	Opcional	Opcional
Memória local de barramento (LMB) dados de apoio à interface	Opcional	Opcional	Opcional	Opcional
Memória local de barramento (LMB) instruções de apoio à interface	Opcional	Opcional	Opcional	Opcional
Hardware barrel shifter	Opcional	Opcional	Opcional	Opcional
Hardware para divisão	Opcional	Opcional	Opcional	Opcional
Hardware para debugação lógica	Opcional	Opcional	Opcional	Opcional
Link simples de *interface*(FSL)	0-7	0-7	0-7	0-7
Máquina de estados e limpeza de instruções	Opcional	Opcional	Opcional	Sim
Acumulador de instruções de *interface* (IOPB)	Opcional	Opcional	Opcional	Não
Acumulador de dados de *interface* (IOPB)	Opcional	Opcional	Opcional	Não
Acumulador de instruções de *interface CacheLink* (IXCL)	-	Opcional	Opcional	Opcional
Acumulador de dados de *interface CacheLink* (DXCL)	-	Opcional	Opcional	Opcional
4 ou 8 palavras para acumulação em linha	-	4	4	Opcional
Suporte de *hardware* execução	-	Opcional	Opcional	Opcional
Padrão de comparação das instruções	-	-	Opcional	Sim
Unidade de ponto flutuante (FPU)	-	-	Opcional	Opcional
Desabilita o *hardware* de multiplicação	-	-	Opcional	Opcional
Hardware de depuração legível ESR e EAR	-	-	Sim	Sim
Versão do processador do registrador	-	-	-	Opcional

As ferramentas do EDK formam um ambiente de desenvolvimento de sistemas embarcados que inclui uma biblioteca de periféricos, onde o EDK auxilia de forma a possibilitar uma configuração intuitiva. Adicionalmente, são incluídos o ambiente de desenvolvimento *software* Eclipse e um depurador GNU.

2.4.5.3.Arquivo de Configuração de *Hardware*

A configuração de *hardware* é armazenada no arquivo (.mhs) – *microprocessor hardware specification*, que contém as portas do sistema. Cada porta pode ser especificada por um nome como entrada, saída e é possível configurar o número de bis que compõe esta porta. A utilização de outros recursos da FPGA como: divisores de *hardware*, também devem ser habilitados para evitar erros na compilação.

No arquivo já citado, estão as instanciações dos periféricos do sistema, bem como a configuração dos mesmos.

Entre as opções de configuração estão o range de memória no qual estão configurados os periféricos, para tal é gerado um arquivo de opções de *hardware* (.mpd) – *microprocessor peripheral description*. A ferramenta EDK possui um editor para esse tipo de arquivo de fácil utilização.

2.4.5.4.Arquivo de Configuração do *Software*

O arquivo de configuração é gerado com a extensão (.mss) – *microprocessor software specification*, contendo as opções de compilação do *software* do sistema. Inclusive a especificação do modo de compilação do código, associação de bibliotecas e periféricos. Este arquivo depende exclusivamente do arquivo de descrição do *hardware* o qual especifica qual o controlador de cada um dos periféricos instanciados.

Os controladores dos periféricos encontram-se em outro arquivo com extensões (.mdd) *microprocessor driver definition*, onde estão os códigos necessários para o controle dos periféricos.

2.4.5.5.Código de Programação

Os três tipos de arquivos anteriormente apresentados, mhs, mpd e mdd, possuem uma sintaxe muito simples, pois, tratam da especificação das opções e de como instanciar os periféricos e seus controladores. Uma vez compilados e introduzidos os periféricos que serão utilizados no sistema em conjunto com o Microblaze, só resta carregar o código do programa. Estes arquivos podem estar em C, C++ ou *Assembler*. O compilador do Microblaze está baseado no compilador GCC.

2.4.5.6.A Arquitetura de Memória

O microblaze utiliza a arquitetura de memória tipo Havard (Hwang K., 2002), onde os dados e instruções são endereçados e acessados em espaços de memória distintos. Cada espaço de memória possui 32 bits para endereçamento, perfazendo um máximo de 4 GBytes para instruções e para dados.

O processador não diferencia acesso a dados ou às portas de entrada e saída (I/O), devido ao sistema de entradas e saídas ser mapeado em memória.

O acesso a memória é feito utilizando-se até três tipos de barramentos: *Local Memory Bus* (LMB), O*n-Chip Peripheral Bus* (OPB) e ou por meio do *Xilinx CacheLink* (XCL).

O Microblaze necessita de dois ciclos de relógio para acessar e ler o barramento LMB e normalmente dois ciclos de relógio para a escrita. As memórias *on-chip* conectadas diretamente ao barramento OPB requer 3 ciclos de relógio para a escrita e 4 ciclos para leitura.

2.4.5.7.Descrição das Interfaces de Sinais

O barramento LMB oferece acesso a memórias *on-chip* do tipo BRAM – *Block RAM*, *dual-port*. A interface OPB oferece conexão para periféricos e memória on-chip e off-chip. A interface *CacheLink* é destinada a periféricos especializados no controle de acesso à memória externa. Além das 3 interfaces de memória suportadas (LMB, OPB e XCL).

Tabela 4 – Características dos barramentos OPB e LMB

Característica	**Barramento**	
	OPB	**LMB**
Largura de endereço (bits)	32	32
Taxa de *clock* máxima (MHz)	125	125
Masters (max)	16	1
Masters (típico)	2-Aug	1
Slaves (max)	16	16
Slaves (típico)	2-Aug	1
Taxa de dados (pico)	500 MB/s	500 MB/s
Leitura/escrita concorrente	Não	Não
Bus locking	Sim	Não
Retry	Sim	Não
Timeout	Sim	Não
Suporte de compilador para *load/store*	Sim	Sim
Utilização de recursos da FPGA	Médio	Baixo

A tabela 4 mostra as principais especificações dos barramentos internos de comunicação disponível para o MicroBlaze.

a) Características da utilização das interfaces pelo Microblaze

O Microblaze pode ser configurado com estas interfaces das seguintes maneiras:

- uma versão 32-bits da interface OPB V2.0;

- LMB fornecendo um simples protocolo síncrono para eficientes transferências para os blocos de memória RAM (BRAMs);
- mecanismo de *streaming* sem arbítrio é suportados rapidamente pelo FSL;
- interface de *debug* para uso com o MDM – *Microprocessor Debug Module Core*.

b) Visão geral das interfaces de I/O do Microblaze

As interfaces mostradas na Fig. 5 são definidas da seguinte forma:

- *DOPB*: Interface de dados do barramento OPB;
- *DLMB*: Interface de dados do barramento LMB (Somente BRAM);
- *IOPB*: Interface de instruções do barramento OPB;
- *ILMB*: Interface de instruções do barramento LMB (Somente BRAM);
- *MFSL* 0..7: Interface master do FSL;
- *SFSL* 0..7 : Interface slave do FSL;
- *IXCL*: Interface de instruções do *CacheLink* para master/slave;
- *DXCL*: Interface de dados do *CacheLink* para master/slave;
- Core: Sinais diversos para *clock, reset, debug e trace*;

2.5. CONCLUSÕES DO CAPÍTULO

Neste capítulo foram estudados os conceitos tecnológicos e a fundamentação teórica em automação veícular, instrumentação virtual e computação reconfigurável.

O estudo do estado da arte dos sistemas de automação veícular justificando a crescente utilização desses sistemas aumentando a segurança e o conforto para o motorista e seus ocupantes. Diferentes implementações de sistemas de automação veícular estão sendo comercializadas no mundo todo com diferentes arquiteturas, grau de flexinilidade e controladores.

A tecnologia em instrumentação virtual apresentada visa inserir o usuário a novos ciclos de projetos aplicando novos técnicas de testes e validação, usando seus

conceitos na elaboração de ambientes de simulação e testes experimentais. Dentro o conceito de instrumentação virtual o LABview apresenta grande versatilidade no que tange sua depuração e a programação em linguagem G. Um grande número de ferramentas com grande utilizadade e de fácil aplicação é disponibilizada ao projetista.

As tecnologias baseadas em computação reconfigurável aplicadas para o desenvolvimento de sistemas de controle e automação mostram-se como boa alternativa para a implementação pois combinam flexibilidade, confiabilidade e baixo custo.

Com o desenvolvimento dos *cores* de microprocessadores embarcados em FPGA aumentou-se ainda mais a versatilidade da aplicação de *hardware* reconfigurável pois possibilitou ao projetista utilizar conceitos de *software* em *hardware* reconfigurável e o projeto de seus periféricos. Os conceitos introduzidos neste capítulo foram fundamentais para a elaboração deste projeto.

3. ARQUITETURA DO SISTEMA

Esse capítulo foca a apresentação da arquitetura proposta na elaboração do controle de movimentação e o controle do estacionamento automático do veículo. Inicialmente, é apresentada a arquitetura proposta em trabalhos anteriores (Bellardi T., 2005) e (Garrido R., 2001), sendo apresentadas partes mecânicas, eletroeletrônicas e computacionais propostas nesses trabalhos.

Sabe-se que para o projeto de sistemas de controle o estudo da arquitetura do sistema é um dos passos iniciais na concepção do projeto. É durante a elaboração da arquitetura do sistema que os projetistas dedicam sua atenção para a definição da comunicação, interação e o funcionamento entre os subsistemas.

Esse projeto não é diferente: para a elaboração do sistema de controle de movimentação dedicaram-se exforços na concepção de uma arquitetura que atendesse os requisitos de segurança, desempenho dentro das especificações do projeto em termos de custo, o que limita a aplicação de tecnologias de alto custo.

No tópico seguinte apresentamos as modificações e extensões que foram realizados na arquitetura proposta nos trabalhos anteriores, mostrando as vantagens da aplicação da computação reconfigurável por meio da tecnologia das FPGA's para o projeto e implementação do controle de movimentação do veículo.

3.1. ARQUITETURA PROPOSTA NOS TRABALHOS ANTERIORES

O projeto SiAE como citado no tópico 1.1.1, teve início em 1999 com o objetivo de se projetar um sistema automático de estacionamento para veículos de passeio movidos por motor a combustão interna. Desde seu surgimento até a elaboração desse trabalho esse projeto passou por duas fases diferentes.

Na primeira fase foram definidas os parâmetros principais do projeto SiAE e seus objetivos. A segunda fase por sua vez, consistiu no estudo e definição da arquitetura flexível e modular do sistema de controle.

As duas fases que antecederam esse trabalho contribuíram de forma importante para a obtenção de resultados consideráveis em termos tecnológicos e científicos na

elaboração e implementação da arquitetura. As implementações nessas fases realizadas trouxeram a possibilidade de se desenvolver soluções a partir do ponto até onde essas fases alcançaram dando seqüência ao projeto SiAE em busca do objetivo principal do projeto: o estacionamento automático do veículo em uma vaga paralela.

3.1.1. Arquitetura Proposta na Primeira fase do Projeto SiAE

Na primeira fase do projeto SiAE (Garrido R., 2001) foram definidos os parâmetros do projeto e seus objetivos. Esse trabalho também contribuiu na realização de vários estudos para se obter os modelos de aceleração, frenagem aerodinâmica, resistência ao avanço e resistência ao rolamento.

Uma vez definidos os parâmetros, objetivos e feito os estudos dos modelos obtidos, iniciaram-se os estudos para a elaboração e projeto da arquitetura de um sistema que realizasse o controle das variáveis responsáveis pela movimentação do veículo. As modificações foram propostas com o intuíto de realizarem o menor número de modificações possíveis no veículo a fim de possibilitar ao mesmo ser controlado pelo sistema ou por um motorista como um veículo comum.

Durante a elaboração da arquitetura do sistema foi seguido o fluxo de projeto saindo do acionamento mecânico para a interface com o operador. Uma vez definida a arquitetura de acionamento foram definidas as arquiteturas computacional e de acionamento eletro-eletrônico para o controle do veículo.

O controle de embreagem foi concebido baseando-se no sistema Autonomy ®, (Garrido, R., 2001) que possibilita o acionamento da embreagem de forma automática com baixo custo. Para a utilização desse sistema foram feitas modificações para que o mesmo pudesse ser controlado por comandos eletrônicos. O sistema Autonomy funciona paralelamente ao sistema original de acionamento e controle da embreagem que é realizado por meio do pedal (ver figura 3.1) .

O sistema de acionamento do freio foi concebido baseando-se no acréscimo de uma alavanca e um motor de corrente contínua ao sistema de acionamento original feito por meio do pedal (ver figura 3.1). O controle de aceleração da rotação do motor foi concebido partindo-se da utilização do sistema de controle de ignição e injeção eletrônica original do veículo de testes, (ver figura 3.1).

No sistema de controle do acionamento do câmbio implementou-se, em paralelo ao sistema original de acionamento, um sistema eletromecânico capaz de acionar com precisão o mecanismo de engate do câmbio por meio de sinais eletrônicos (ver figura 3.1).

Na arquitetura computacional de controle foi utilizado um microcomputador portátil (*Laptop*), um kit CW 552 com microcontrolador da família 8051 e seis placas de acionamento e condicionamento de sinais, sendo que uma placa para cada subsistema, por exemplo: acionamento do freio, acionamento da embreagem, acionamento do acelerador, acionamento do câmbio na direção de *X* e *Y* e o sensoriamento da rotação do motor e velocidade do veículo (ver figura 3.1).

Na figura 3.1 pode-se visualizar a arquitetura computacional e de acionamento eletro-eletrônicos proposta por (Garrido R., 2001).

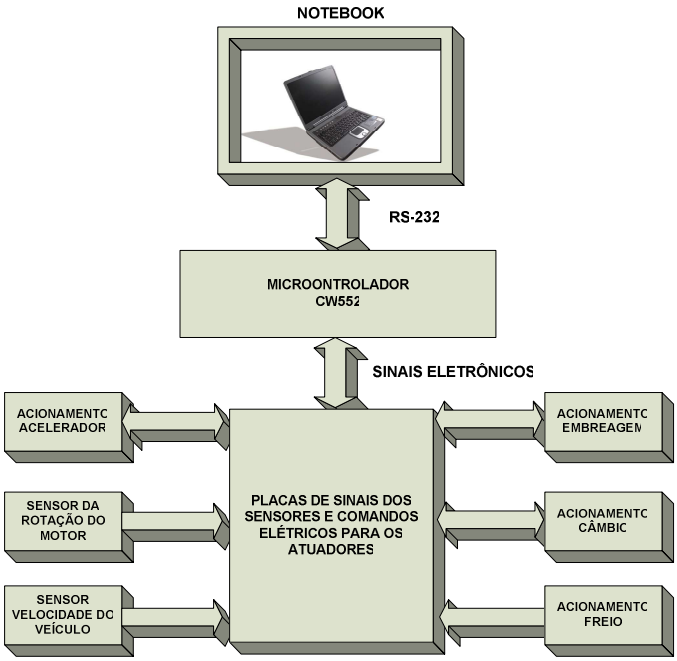


Figura 3.1 – Diagrama de blocos da arquitetura (Garrido R., 2001)

Nessa arquitetura do sistema proposta e implementado, o notebook era o responsável por processar os comandos e enviar ao microcontrolador.

As principais contribuições desse trabalho foram: o projeto e montagem do acionamento mecânicos e o controle das variáveis importantes para a movimentação do veículo, a montagem e os testes de controle da embreagem utilizando-se o sistema Autonomy® da FIAT.

3.1.2. Arquitetura proposta na segunda fase do projeto SiAE

Na segunda fase do projeto, foi definida uma arquitetura flexível para o controle da movimentação do veículo.

Inicialmente, aplicou-se a metodologia de se dividir o sistema e organizá-lo em diferentes módulos os quais possuem funções bem específicas para a realização das tarefas relacionadas ao controle da movimentação do veículo.

O desenho da arquitetura foi projetado se pensando em quatro módulos sendo esses: mapeamento, geração e controle de trajetória, controle de movimentação e interface com o usuário (veja a figura 3.2).

Essa estrutura apresentada por (Bellardi T., 2005) tem um caráter de orientar o desenvolvimento do projeto, não consistindo em um critério rígido (podendo sofrer modificações com os avanços do projeto).

Dentro do conceito definido o módulo de mapeamento tem por objetivo realizar o escaneamento do ambiente em que o veículo se encontra informando o posicionamento do alvo a ser alcançado e os possíveis obstáculos existentes, transmitindo essas informações ao módulo de geração da trajetória.

Uma vez que as informações chegam ao módulo de geração e controle de trajetória este realiza os cálculos necessários a partir da cinemática do veículo, as informações do ambiente e a manobra de movimentação que se deseja realizar.

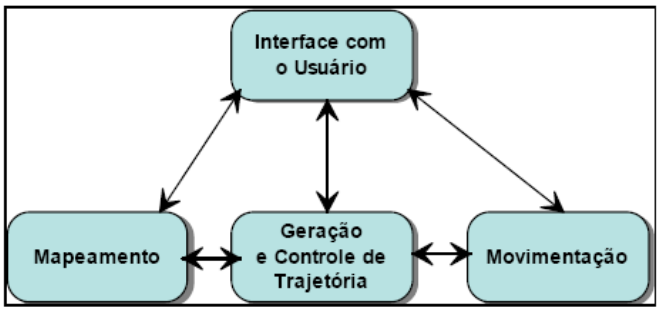


Figura 3.2 – Estrutura dos módulos do sistema (Bellardi T., 2005)

O módulo de movimentação recebe os comandos do módulo de geração e controle da trajetória e realiza o controle dos dispositivos atuadores correspondentes às variáveis de controle de movimentação do veículo. Além de receber os comandos do módulo de geração e controle da trajetória, esse módulo também pode receber comandos diretamente de um operador e realiza-los.

Finalmente, o módulo de interface com o usuário tem a como função levar as informações referentes ao sistema de forma inteligível até o operador, e também é responsável por incerir os comandos do operador ao sistema.

Nessa etapa do projeto, trabalhou-se na melhoria do sistema eletro-eletrônico para torná-lo mais confiável e robusto. Muito se dedicou ao desenvolvimento do módulo de movimentação trabalhando-se na programação do microcontrolador CW552 (Controlware, 1990), o mesmo utilizado na etapa anterior, juntamente com o microcomputador portátil (*Laptop*).

Na figura 3.3 temos a arquitetura detalhada do módulo de movimentação proposta por (Bellardi T., 2005).

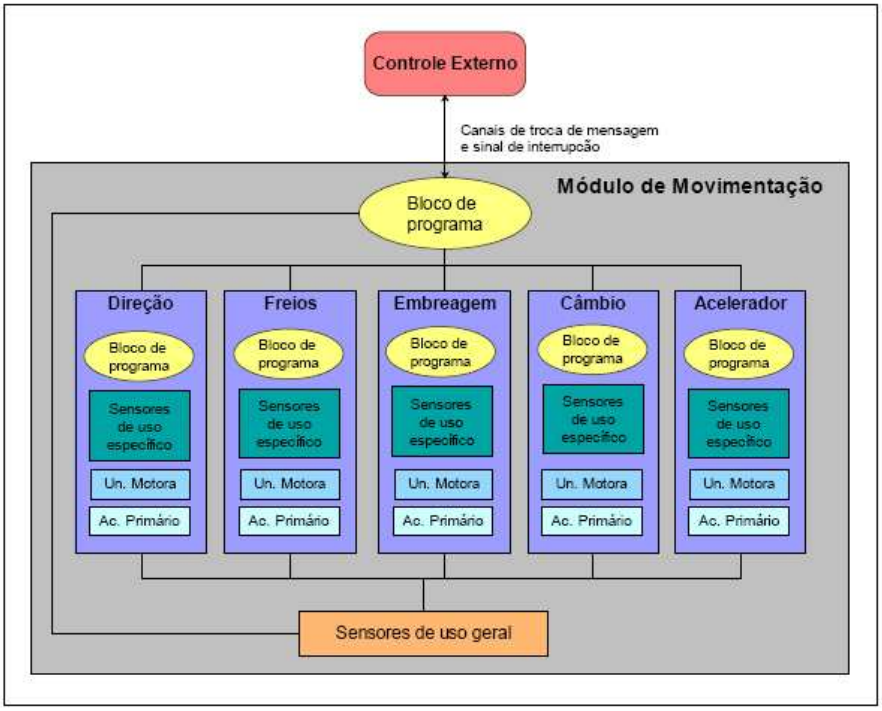


Figura 3.3 – Arquitetura conceitual do módulo de movimentação (Bellardi T., 2005)

Diferentemente da fase anterior, nessa pode-se verificar uma padronização descritiva da arquitetura do sistema onde é possível visualizar o módulo e seus blocos.

A principal contribuição dessa etapa para o projeto foi a definição de uma arquitetura distribuída e flexível, juntamente com a definição dos módulos e os blocos do módulo de movimentação.

3.2. A NOVA PROPOSTA PARA O SISTEMA

Como apresentado nos itens anteriores o projeto SiAE passou por duas fases que antecederam a atual e por este motivo apresentamos as arquiteturas das fases anteriores. Estas arquiteturas somaram grandes contribuições para o projeto e propiciaram o desenvolvimento da arquitetura atual que se utilizou dos principais conceitos desenvolvidos e aplicados anteriormente. Adicionando-se conceitos em *hardware* reconfigurável (veja capítulo 4), ambiente de simulação desenvolvido utilizando-se conceitos de instrumentação virtual (veja capítulo 5), e o projeto do sistema de controle de movimentação do veículo de testes real (veja capítulo 6).

Para desenvolver a arquitetura que atendesse os requisitos do projeto SiAE de baixo custo, desempenho e flexibilidade pensou-se na possibilidade de alteração na arquitetura proposta por (Bellardi T., 2005). Entretanto, foi visto que essa arquitetura possuía limitações para a implementação de um sistema de simulação e validação do controle de movimentação e a necessidade de tornar o sistema com um formato mais comercial tornou-se também um alvo a ser alcançado.

Dessa forma surgiu a necessidade de se projetar uma nova arquitetura aproveitando-se muito do que já se havia desenvolvido, acrescentando novas possibilidades e recursos visando romper novas fronteiras até o momento não superadas pelo projeto SiAE.

3.2.1. Conceitos envolvidos na Nova Proposta

Ao se pensar numa nova proposta para a arquitetura, partimos das definições do projeto SiAE para só então analisar os principais pontos onde o desenvolvimento de uma proposta contribuiria de forma consistente para o projeto SiAE servindo como base para os sistemas de navegação, apoio ao motorista, etc.

A proposta desenvolvida na fase atual desse trabalho objetivou a aplicação dos conceitos das arquiteturas anteriores em *Hardware* reconfigurável, o que trouxe ao projeto uma nova conceituação abrindo grande espaço para o desenvolvimento de sistemas de alto nível para o controle da trajetória, controle de navegação, caracterizando assim uma verticalização do projeto dentro dos conceitos da automação veicular no que tange os sistemas de apoio ao motorista e chegando até mesmo aos

sistemas veículares robotizados.

Busca-se com a arquitetura proposta alcançar o objetivo inicial do projeto SiAE e possibilitando ao veículo executar o estacionamento automático em uma vaga paralela a sua direita e podendo automatizar outras tarefas (Correia A., et al., 2007a).

Sendo assim, os principais conceitos desenvolvidos nesta fase foram concentrados em três frentes:

a) o desenvolvimento do controlador de movimentação em um microprocessador embarcado em lógica reconfigurável FPGA;

b) o desenvolvimento de um ambiente de simulação e validação do controlador;

c) o projeto e desenvolvimento das placas de interface de sinais e potência.

3.2.2. A arquitetura do Novo Sistema

A arquitetura do sistema de forma macro foi definida como pode ser visto na figura 3.4 e figura 3.5. Basicamente, o sistema é formado por três módulos: interface com o operador, controle da movimentação e simulação ou veículo de testes.

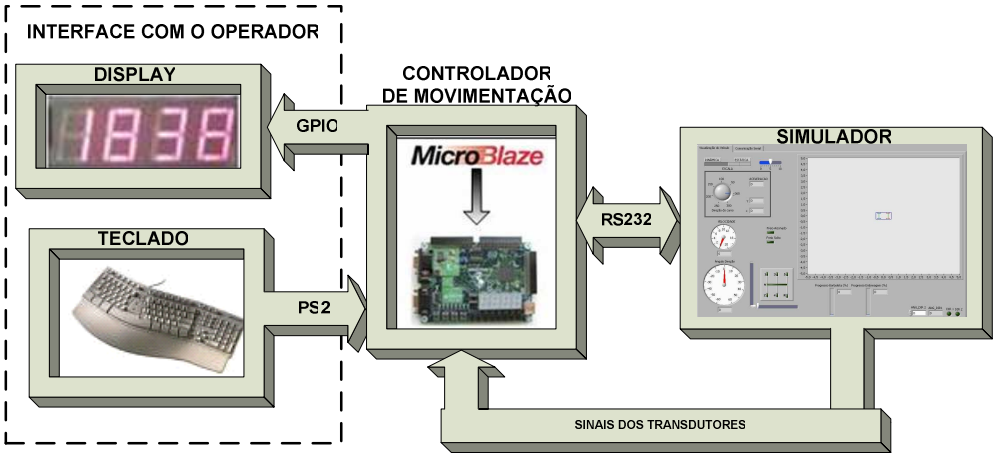


Figura 3.4 – Arquitetura macro do sistema implementado

O primeiro subsistema foi definido para a interface com o operador, contando com um display de sete segmentos (na FPGA) o qual possibilita o operador verificar os comandos e o *status* das variáveis. Os led's da FPGA também são utilizados para apresentar a condição das variáveis ao operador do sistema (Correia A., et al., 2007a).

O segundo subsistema é o controlador de movimentação onde fica todo o processamento e o controle de todos os subsistemas. Este foi implementado por meio de um *core* da Xilinx em um microprocessador e embarcado em uma FPGA modelo Spartan-3 (no capítulo 4 é detalhado o projeto do microprocessador embarcado) (Correia A., et al., 2007a).

O terceiro e último subsistema é formado pelo ambiente de simulação que reproduz de forma aproximada (usando intrumentação virtual, c om LabVIEW) a resposta cinemática das variáveis que são importantes para o controle de movimentação do veículo. O sistema, uma vez testado no ambiente de simulação, pode ser ligado no veículo de teste por meio de placas de interface para o acionamento dos atuadores e a leitura dos transdutores instalados no veículo (ver figura 3.5).

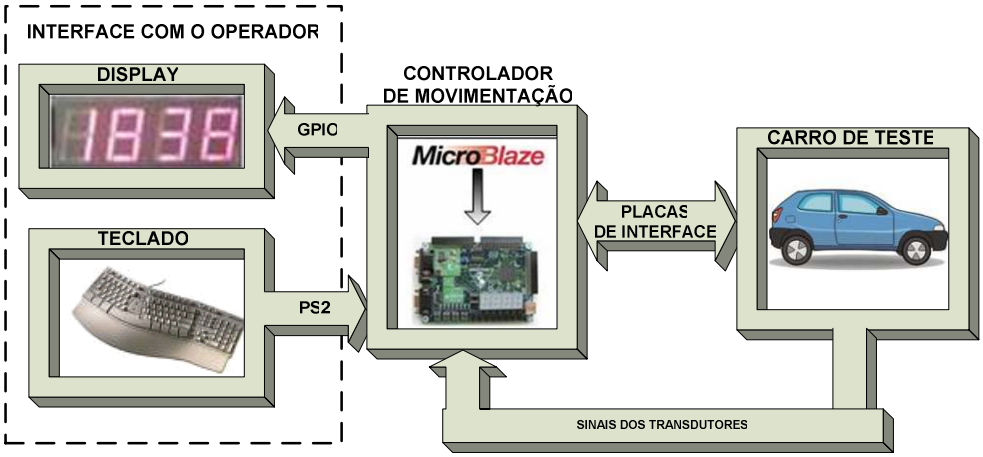


Figura 3.5 – Arquitetura do sistema implementado, no carro de testes

Esta nova arquitetura proposta nesse trabalho possui mudanças fundamentais para a evolução do sistema e do projeto. Uma das contribuições que podem ser destacadas é não utilização de qualquer computador ou processador ligado ao sistema para realização de um pré-processamento. Desta forma o controle fica totalmente inserido em uma FPGA que por sua vez gerência os sinais de comando do operador, lê os sinais dos transdutores e executa as ações necessárias (Correia A., et al., 2007b).

A segunda mudança é a criação de um ambiente de simulação que pode ser utilizado para testar e validar o sistema de antes de se testar no veículo real, diminuindo os riscos de acidentes e o dano do equipamento bem como do ambiente a sua volta. Em (Dario Giove, 2004) foi aplicada uma metodologia próxima da aplicada nesse trabalho, mas em menores proporções para o controle de acelerador, utilizado um sistema de simulação em instrumentação virtual e lógica reconfigurável para isso.

3.3. CONCLUSÕES DO CAPÍTULO

Neste capítulo foi estudada a arquitetura do sistema proposto para o projeto SiAE nas fases um e dois e a extensão da arquitetura proposta nesse trabalho.

A arquitetura proposta na primeira fase definiu parâmetros com o estudo dos modelos de aceleração, aerodinâmicos e frenagem do feículo de testes. Nessa fase foram projetados e implementados os controles de aceleração, freio, direção, câmbio e embreagem embarcados em um microcontrolador o qual recebia comandos via porta serial de um notebook. As principais contribuições dessa etapa foram: projeto e montagem dos atuadores, modelamento da aceleração e frenagem do veículo de testes.

Na segunda etapa dividiu-se o sistema em módulos os quais receberam atribuições específicas de controle dos atuadores correspondentes. A proposta da arquitetura foi projetada se pensando em quatro módulos: interface com o usuário, mapeamento, geração e controle de trajetória e movimentação. A principal contribuição dessa etapa concentrou-se no desenvolvimento e organização do sistema em uma arquitetura distribuída e flexível com a definição e dissociação do sistema em módulos com funções bem definidas.

A etapa atual do projeto SiAE objetivou em termos da arquitetura propor a extensão da arquitetura aplicando-se os conceitos de de *hardware* reconfigurável e a aplicação de instrumentação virtual para a elaboração do ambiente de simulação da movimentação do veículo. Na nova arquitetura proposta o sistema do controle de movimentação é programado em um microprocessador embarcado em lógica reconfigurável o que aumenta ainda mais a flexibilidade do sistema, pois além da flexibilidade dos módulos em software pode-se efetivar mudanças no *hardware*.

4. CONFIGURAÇÃO DO HARDWARE DE CONTROLE

Nesse capítulo são apresentadas ao leitor a descrição da configuração do *hardware*, as ferramentas utilizadas e os conceitos aplicados para se configurar um microprocessador embarcado em lógica reconfigurável. Também é apresentada a descrição do *software* desenvolvido, seu funcionamento no controle de cada um dos módulos e suas variáveis. Adicionalmente, são apresentados os resultados do desenvolvimento do *hardware* de controle, a ocupação dos recursos lógicos da FPGA e seu desempenho.

A configuração do *Hardware* de controle representa uma das etapas primordiais desse trabalho e objetiva o projeto e implementação de um sistema para o controle de movimentação do veículo que atenda as necessidades de segurança e os objetivos definidos para o controle do veículo dentro das especificações do projeto SiAE.

4.1. CONFIGURAÇÕES DO *HARDWARE* EMBARCADO

A configuração dos recursos de *hardware* da FPGA necessários à implementação do controlador de movimentação passou pela primeira fase que foi a definição da arquitetura do controlador de movimentação e o ambiente de simulação e validação, apresentada no capítulo anterior. Após a definição dos recursos necessários, iniciou-se propriamente o estudo das ferramentas de configuração desses recursos e sua implementação.

No projeto do sistema embarcado de controle de movimentação foi utilizada a plataforma XPS do EDK (ver tópico 2.4.5), sendo utilizada esta ferramenta para formatar os recursos disponíveis na Spartan-3. A configuração de *hardware* foi realizada em duas etapas. Na primeira utilizou-se a opção *Base System Builder Wizard* (EDK MicroBlaze tutorial, 2005) na configuração dos recursos padrão disponíveis pelo Microblaze para a Spartan-3. Na segunda etapa foi utilizado o ISE – *Integrated Software Environment*, para a descrição do *hardware* de controle de comunicação de teclado utilizando o protocolo PS2.

4.1.1. Configurações da Arquitetura do Sistema Embarcado e seus Periféricos

Com o *Base System Builder Wizard* a configuração dos recursos de *hardware* que o projetista necessita se torna uma tarefa simples e rápida. Na figura 4.1, é mostrada a tela inicial de configuração onde o projetista seleciona o fornecedor da FPGA que está sendo utilizada, o modelo da placa e a sua revisão. No caso foi selecionado fornecedor Xilinx e a placa Spartan-3 que foi a placa utilizada nesse trabalho.

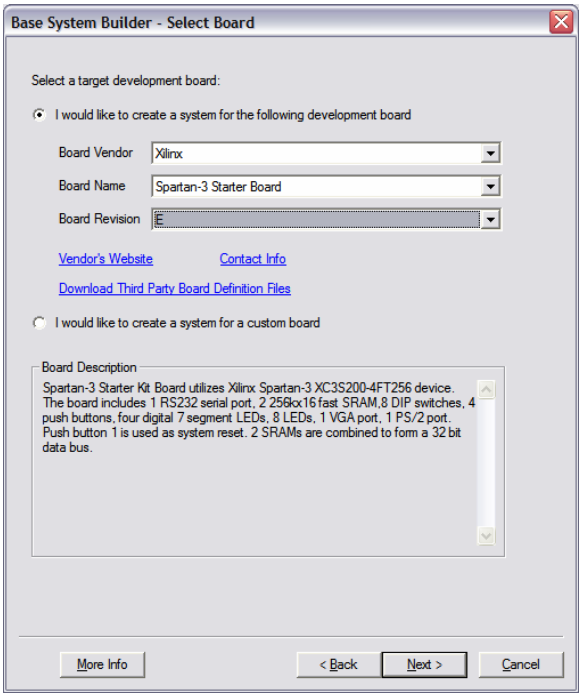


Figura 4.1 – Tela de seleção do modelo da placa utilizada – *Base System Builder Wizard*

Uma vez selecionada a placa, o projetista pode selecionar o processador com o qual deseja trabalhar, MicroBlaze ou PowerPC. No caso da placa selecionada só é disponibilizado o processador MicroBlaze.

Na fase seguinte foram configuradas as freqüências do relógio, do sistema e a freqüência do relógio do barramento, o tamanho das palavras, a forma de teste do processador e como será utilizado o bloco de memória *Cache*.

A figura 4.2, mostra o diagrama do *hardware* configurado utilizando o *Base System Builder Wizard*, suas ligações e a estrutura do sistema. Cada um dos blocos é considerado um periférico que uma vez configurado, foi ligado ao barramento *OPB – On-chip Periferical Bus*, que leva as informações de cada um dos periféricos ao processador onde são processadas.

O sistema possui 3 barramentos: um barramento OPB e dois barramentos do tipo LMB (dados e instruções). O Microblaze é *master* dos 3 barramentos enquanto os demais periféricos são escravos (*slaves*).

Os periféricos apresentados na figura 4.2, são conectados ao barramento OPB e possuem até 3 sinais externos que o projetista deve conhecer para ter acesso ao controle do trafego de informação. *opb_clk*, *opb_rst* e um sinal de interrupção.

Quando os periféricos (*chaves*, *botões*, *display*, *timer*, *RS 232* e *leds*) não estiverem previamente conectados, devem ser ligados aos sinais de *clock* (*opb_clk*), *reset* (*opb_rst*). No caso do uso de interrupção, o sinal referente à interrupção deve ser ligado ao controlador de interrupção.

Cada periférico irá possuir um endereço base (*c_baseaddr*) para utilização na escrita e leitura de registradores do periférico. A geração de endereços é automática e feita utilizando o *XPS*.

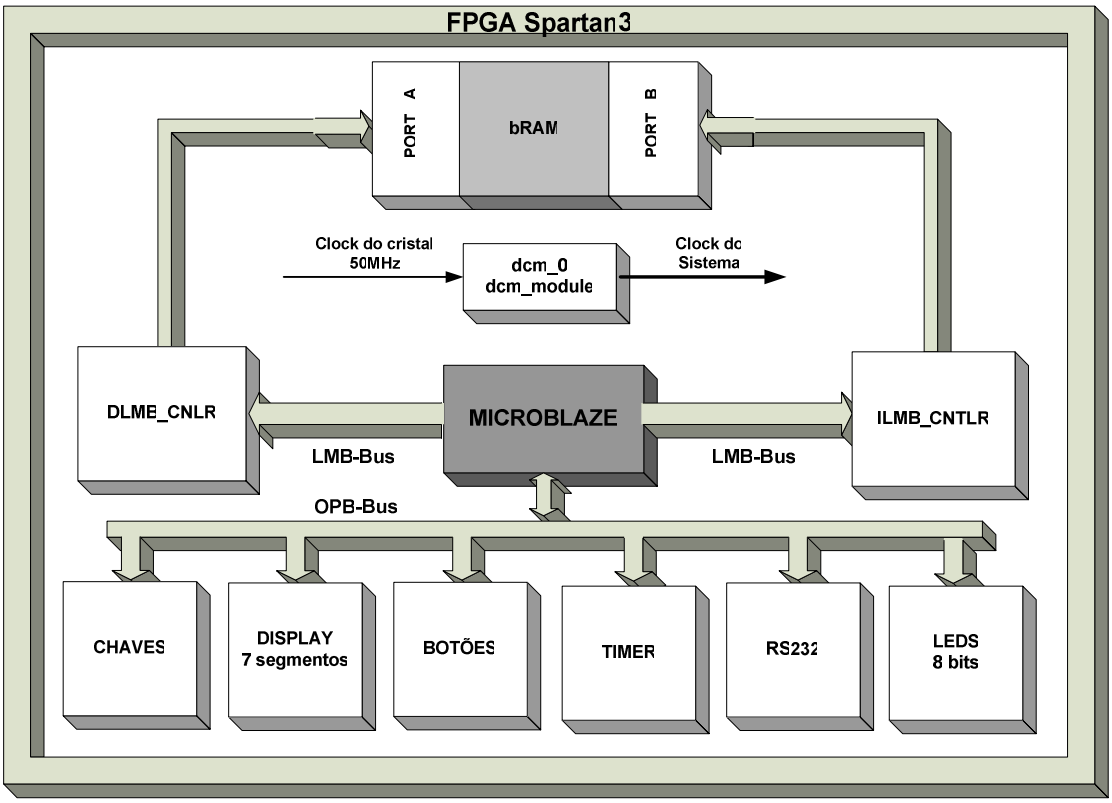


Figura 4.2 – Diagrama do *hardware* configurado utilizando o *Base System Builder Wizard*

O *lmb_bram_if_cntlr* é um controlador de interface para o bloco de memória *RAM* (*BRAM*). Este controlador possui uma interface para conexão com o barramento

LMB e com a memória *BRAM* . O sistema possui 2 instâncias deste periférico: uma instância controla o barramento de dados (*dlmb_cntlr*), enquanto a outra instância (*ilmb_cntlr*) controla o barramento de instruções. O MicroBlaze possui uma arquitetura tipo *Havard* (Patterson, D. A. e Hennessy, J. L., 2000) com um conjunto de instruções RISC.

O *bram_block* é uma memória totalmente parametrizável com dois portos de acesso (*dual_port*). As FPGA's da Xilinx possuem blocos de memória imersos em sua estrutura re-configurável. Assim o tamanho da memória instanciada irá depender da capacidade da família da FPGA. A Spartan-3 XC3S200 possui 12 blocos BRAM de 2,25 KB cada, totalizando 27 KB. O sistema possui 1 instância *bram_block* contendo 8 BRAMs, totalizando 16MB**.**

O *OPB_GPIO* (*General Purpose Input/Output*) é um periférico de entrada e saída paralela com interface de 32 bits para conexão com um barramento OPB. O OPB GPIO pode ser configurado com 1 ou 2 canais. Cada canal pode ser configurado como entrada, saída ou bidirecional.

Caso o pino de entrada exclusivo seja escolhido, a quantidade de lógica necessária para implementação do *GPIO* será menor. Na utilização do pino bidirecional, sua funcionalidade deve ser configurada dinamicamente via *software*.

Estes parâmetros, como de outros periféricos, podem ser configurados diretamente no arquivo MPD – *Microprocessor Peripheral Description*, ou pelo menu "*Parameters"* da janela "*Add/Edit Hardware Platform Specifications*" no XPS. O periférico possui uma interface para conexão com o barramento OPB e uma interface para mapeamento externo. A Tabela 4 descreve os sinais da interface externa.

4.1.2. Configurações dos Registradores

De acordo com a funcionalidade de cada canal do GPIO, até 4 registradores de 32 bits podem estar presentes no *core*. Dois registradores de dados (*gpio_data* e *gpio_data1*) e dois registradores com três estados (*gpio_tri* e *gpio2_tri*). Os registradores de dados estão presentes no modo saída e os registradores três estados no modo três estados.

No modo entrada de um canal, nenhum registrador do canal está presente.

Os registradores de dados *gpio_datax* contém os dados atuais da saída e os registradores *gpio_tri* configuram o pino bidirecional para o funcionamento como entrada ou como saída de dados.

Quatro instâncias do *core opb_gpio* estão presentes no sistema Microblaze:

1) ***leds_8bit***: interface de saída de dados com os 8 leds presentes na placa S3-SB, possui largura de 8 bits;

2) ***dip_switches_8bit***: interface de entrada de dados com os 8 *switches* presentes na placa S3-SB. Possui largura de 8 bits;

3) ***push_buttons_3bit***: interface de entrada de dados com os 3 *push-buttons* presentes na placa S3-SB. Possui largura de 3 bits;

4) ***pwm_io***: interface que foi necessária para a recuperação dos sinais PWM (*Pulse Width Modulated)*. Estes sinais PWM são sinais externos de outro periférico, o opb_timer, e para uma manipulação destes dados via *software* foi necessária a introdução de um opb_gpio. A largura dos dados foi configurada inicialmente para 5 bits, ou seja, até 5 sinais PWM podem ser recuperados.

   O *opb_timer* é um *timer*/*contador* de 32 bits com interface para o barramento OPB.

   As Características do registrador de configuração do timer são as seguintes:

   - interface de 32 bits para o barramento OPB com suporte a *byte-enable*;
   - 2 *timers* programáveis com interrupção e geração e captura de eventos;
   - largura do contador configurável;
   - saída para PWM;

4.1.3. Descrição dos Módulos Timer's/PWMs

O *core* do Microblaze possui dois módulos *timers* idênticos, como mostrado na figura 4.3. Cada *timer* possui um registrador de *load* (*tlrx*), um registrador contador (*tcrx*) e um registrador de estado de controle do timer (*tcsrx*).

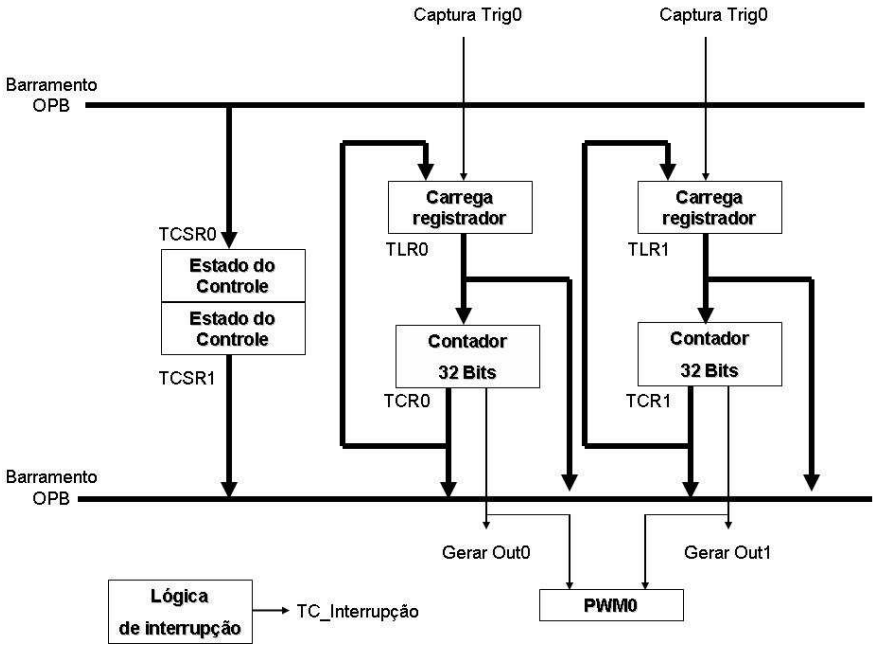


Figura 4.3 – Diagrama de blocos opb_timer

O *timer* possui 3 modos de operação:

a) **Modo *generate***: ***n***este modo o valor do registrador *TLR* é carregado no contador *TCR*. O contador, quando habilitado, começa a contagem crescente ou decrescente, dependendo do valor do bit *UDT* no registrador *TCSR*.

Na transição do *carry out* do contador, o contador pode parar a contagem ou automaticamente re-carregar o valor do registrador *TLR* e continuar a contagem (bit *ARHT* do *TCSR*). Se o bit *TINT* em *TCSR* estiver habilitado, a saída *GenerateOut* será colocada em nível alto por 1 ciclo de *clock.* Se a interrupção também estiver habilitada (*ENIT* em *TCSR*), o sinal de interrupção também será colocado em nível alto por 1 ciclo de clock. A função generate foi utilizada na configuração do PWM – *Pulse Width-Modulation*.

b) **Modo *captura*:** no modo captura o valor do contador é armazenado no registrador *load* (*TLR*) quando um sinal de captura é detectado. Na ocorrência do evento de captura, o bit *TINT* também é habilitado.

Esse modo não foi utilizado na configuração do timer por não haver necessidade de sua aplicabilidade.

c) **Modo PWM**: Esse modo foi utilizado para a geração de pulsos PWM para o controle de velocidade dos motores de corrente contínua – CC, instalados para realizarem o acionamento do acelerador, freio, câmbio e direção.

Neste modo os dois timers presentes no OPB_TIMER são combinados para produzir um sinal modulado em largura de pulso PWM com uma freqüência e ciclo de trabalho específico. O Timer (0) determina o período, enquanto o Timer (1) determina o ciclo de trabalho.

4.1.4. Configuração dos Timers no Modo PWM's

Para a ativação do modo PWM, os seguintes detalhes devem ser observados:

- O modo *generate* deve ser habilitado para ambos timers (bit *MDT* no *TCSR*);
- Os bits *pwma0* em *tcsr0* e *pwmb0* em *tcsr1* devem ser habilitado para '1' para habilitar o modo PWM;
- Os sinais *GenerateOut* devem ser habilitados no TCSR (bit GENT = '1'), porque o sinal PWM é gerado a partir dos sinais *GenerateOut* do Timer0 e do Timer1;
- O nível lógico ativo dos sinais *GenerateOut* devem ser habilitados para '1' em ambos timers. Isto é feito setando os parâmetros *c_gen0_assert* e *c_gen1_assert* para '1';
- O contador pode ter contagem crescente ou decrescente (bit *UDT*).

A freqüência do PWM é determinada pelo registrador *TLR0*, e no caso do contador em contagem decrescente, o valor de *TLR0* é calculado da seguinte maneira:

$$TLR0 = \frac{FREQ\_OPB\_CLOCK}{FREQ\_PWM} - 2 \qquad (1)$$

Já o ciclo do PWM é determinado pelo registrador *TLR1*, e no caso do contador em contagem decrescente, o valor de *TLR1* é calculado da seguinte maneira:

$$TLR1 = \frac{DUTY\_CICLE \bullet FREQ\_OPB\_CLOCK}{100 \bullet FREQ\_PWM} - 2 \qquad (2)$$

Para nosso caso, FREQ_OPB_CLOCK = 50MHz (*clock* de operação), então:

$$TLR0 = \frac{50000000}{FREQ\_PWM} - 2 \qquad (3)$$

$$TLR1 = \frac{50000 \bullet DUTY\_CYCLE}{FREQ\_PWM} - 2 \qquad (4)$$

Com o valor de *FREQ_PWM* em [hz] e DUTY_CYCLE valendo de 0 a 100.

Tabela 5 – Descrição bits registrador TCSR

Bit	Nome	Descrição
21	ENALL	'0' – Sem efeitos nos timers '1' - Habilita todos os timers
22	PWMAx	'0' – Desabilita PWM '1' - Habilita PWM
23	TINTx	**Leitura:** '0' – Não ocorreu interrupção **Leitura:** '1' – Ocorreu interrupção **Escrita :** '0' – Sem mudança no estado de TxINT **Escrita :** '1' – Zera T0INT
24	ENTx	'0' – Desabilita timer (contador pára) '1' – Habilita timer (contador ligado)
25	ENITx	'0' – Desabilita sinal de interrupção '1' – Habilita sinal de interrupção
26	LOADx	'0' – Nada '1' – Carrega contador com valor de TLRx
27	ARHTx	No *overflow* do contador ou na detecção do modo captura: '0' – Mantém contador ou valor de captura '1' – Recarrega valor de TLRx ou sobrescreve valor de captura
28	CAPTx	'0' – Desabilita sinal de *trigger* da captura '1' – Habilita sinal de *trigger* da captura
29	GENTx	'0' – Desabilita sinal externo de *generate* '1' – Habilita sinal externo de *generate*
30	UDTx	'0' – Contagem crescente '1' – Contagem decrescente
31	MDTx	'0' – Modo *generate* '1' – Modo captura

Para o sistema de controle de movimentação do simulador, foi definida uma freqüência em torno de 100 Hz para a operação do PWM. A freqüência deve ser baixa devido ao limite de velocidade do padrão serial protocolo RS 232. Para o controlador de movimentação do veículo de teste a freqüência de operação foi definida em 20Khz.

As instâncias do periférico OPB_TIMER foram as seguintes:

- ***pwm_timer_*1** – gera sinal PWM para controle dos módulos *Acelerador* e *Freio* do carro;

- ***pwm_timer_2*** – gera sinal PWM para controle do módulo *Câmbio* do carro.

O *opb_uartlite* é um periférico de transmissão/recepção assíncrona de dados seriais (UART) para conexão com um barramento OPB. Dentre suas características pode-se destacar:

- interface de 8 bits com o barramento OPB;
- um canal para transmissão e outro para recepção (*full duplex*);
- *buffer* FIFO de 16 caracteres para recepção e 16 caracteres para transmissão;
- número de bits dos caracteres configurável (5 a 8 bits);
- paridade par ou ímpar;
- *baud rate* configurável.

A instância de nome RS232 no Microblaze utiliza uma taxa de transmissão (*baud rate*) de 19600 bps (bits por segundo), sem paridade e usando palavras de 8 bits.

Os sinais RX e TX do *core* foram ligados apropriadamente nos pinos respectivos na interface serial padrão RS-232 da placa Spartan-3.

4.1.5. Inserção do Módulo Teclado no Sistema

O periférico *opb_keyboard* não é um *core* disponível na suíte do EDK. Este *core* foi desenvolvido separadamente utilizando a ferramenta de descrição e *hardware* ISE e posteriormente feito a importação do periférico e sua conexão ao barramento OPB do sistema embarcado. O *opb_keyboard* tem como função realizar a interface via porta padrão PS/2 com um teclado de PC comum. Este periférico possui os seguintes sinais que estão listados na Tabela 5.

Tabela 6 – Descrição dos sinais do periférico opb_keyboard

Sinal	Interface	I/O	Descrição
keyboard_data	Externa	I	Sinal de dados da porta PS/2
Keyboard_clk	Externa	I	Sinal de sincronismo da porta PS/2
Reset	OPB	I	Sinal de *reset*
Scan_code	OPB	O	*Scan_code* da tecla pressionada do teclado
Scan_ready	OPB	O	Sinal de *scan_code* disponível
read	OPB	I	Sinal de *acknowledge* de leitura do *scan_code*

A Figura 4.4 ilustra o registrador de status e leitura do teclado (KSR – *Keyboard Status Register*):

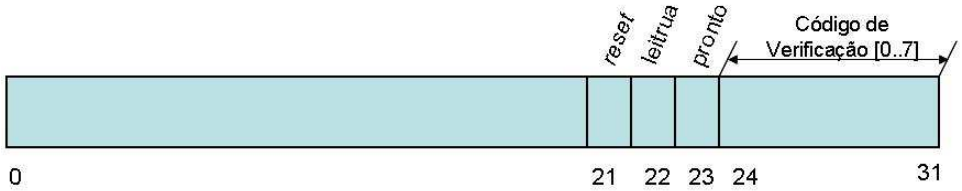


Figura 4.4- Registrador KSR – opb_keyboard

Os procedimentos para um teclado conectado a uma porta PS/2 são:

a) verificar o status do bit 23 ready, caso seja ‘1’ alguma tecla foi pressionada e o respectivo *scan_code* se encontra na posição 24 – 31 do registrador KSR;

b) recuperar o código da tecla (*scan_code*) do teclado lendo o *byte* menos significativo de KSR;

c) realizar um reconhecimento da tecla (*acknowledge*) selecionando o bit 22 – leitura em ‘1’;

O periférico *opb_7segled* também é um *core* que foi criado com a ferramenta *Create/Import Peripheral Wizard* do XPS. Esse *core* é um controlador do display de 7 segmentos disponível na placa S3-SB. Ele tem como função realizar a multiplexação do display e sua respectiva descrição.

Tabela 7 – Descrição dos registradores opb_7segled

Registrador	Endereço	Descrição
reg0	C_BASEADDR + 0x00	Caractere do dígito das unidades do display
reg1	C_BASEADDR + 0x01	Caractere do dígito das dezenas do display
reg2	C_BASEADDR + 0x02	Caractere do dígito das centenas do display
reg3	C_BASEADDR + 0x03	Caractere do dígito do milhar do display

O *core dcm_module* tem como entrada o sinal *clock* vindo do oscilador da placa Spartan-3 e sua saída é utilizada como *clock* do nosso sistema embarcado. Sua principal função é reduzir o *clock skew* do circuito, ou seja, reduzir a diferença do tempo de propagação do sinal de *clock*.

As configurações foram feitas nos periféricos e em sua forma de acesso ao barramento OPB. Foram definidos o tamanho da memória do programa, o funcionamento de cada um dos periféricos acrescentando: o protocolo PS2 e os módulos PWM (que fazem parte do sistema de controle de movimentação, mas que não foram configurados por meio do *Base System Builder Wizard*).

4.2. ASPECTOS GERAIS DESENVOLVIMENTO DO *SOFTWARE* DOS MÓDULOS DE CONTROLE

O projeto do *software* partiu da arquitetura flexível e distribuída definida por (Bellardi T., 2005). Na arquitetura descrita no tópico 3.1 desse trabalho, foi definido que cada um dos módulos seria responsável por controlar as variáveis de movimentação separadamente, respondendo aos comandos do operador.

Uma vez definidos os módulos do sistema de controle de movimentação, cada um deles passou por um processo de detalhamento o qual contou com uma analise anterior do funcionamento das variáveis de movimentação do veículo. No processo de detalhamento dos módulos, foram definidos quais os recursos de *hardware* seriam necessários, bem como a relação entre esses módulos. Também foram definidas as funções a serem executadas dentro dos módulos do sistema.

O *software* foi desenvolvido utilizando a linguagem C e compilado pelo GNU GCC com diretivas para o processador Microblaze. Foi realizada uma programação em módulos, divididos em arquivos que depois foram ligados e compilados por meio do GCC. Essa programação também foi realizada de forma a facilitar a migração do *software* desenvolvido do ambiente de simulação no controlador de movimentação para a aplicação no veículo real de teste apenas com a troca do arquivo *protocolo.c*, uma vez que o hardware configurado na FPGA é o mesmo.

A diferença fundamental entre os dois sistemas (citada no capítulo 3 nas figuras 3.4 e 3.5) é que na aplicação de controle de movimentação do veículo de teste não existe um protocolo de comunicação, ou seja, os dados são passados paralelamente para o

carro e os sinais de entrada vindos do conversor A/D de oito entradas devem ser multiplexados por meio do arquivo *interface_dados.c*. Já na aplicação desenvolvida para o simulador cinemático, implementado em LavVIEW, os dados são enviados e recebidos por meio do protocolo de comunicação no padrão RS232 por meio do arquivo *protocolo.c*.

Nas figuras 4.5 e 4.6, pode-se observar os diagramas de blocos do software do controlador de movimentação no microprocessador embarcado na FPGA para o veículo de testes e para o simulador cinemático, respectivamente (Correia A., et al., 2007a).

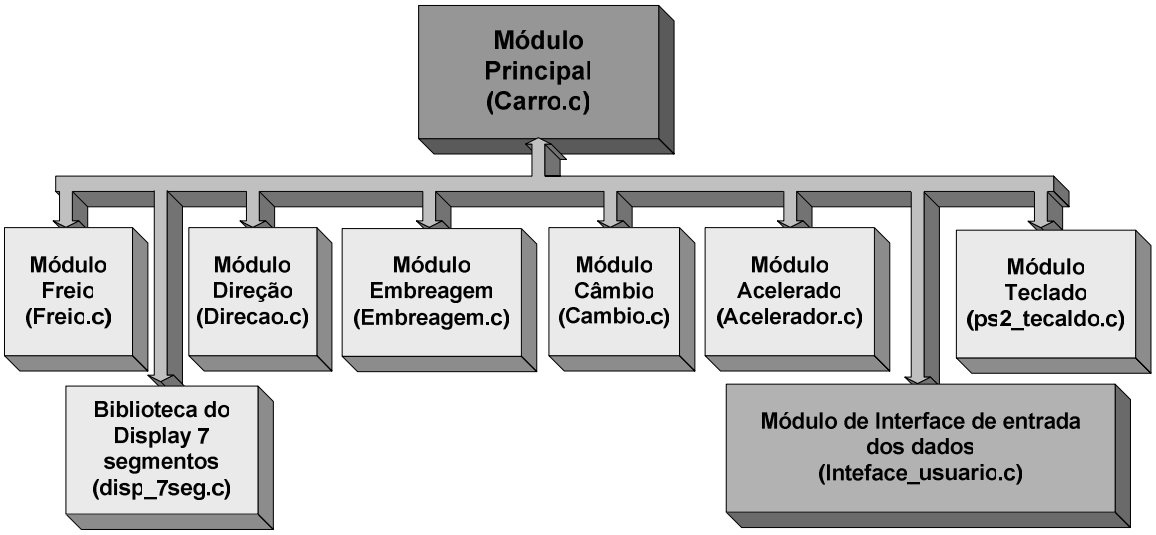


Figura 4.5 – Diagrama do *software* do controlador de movimentação (Carro de testes)

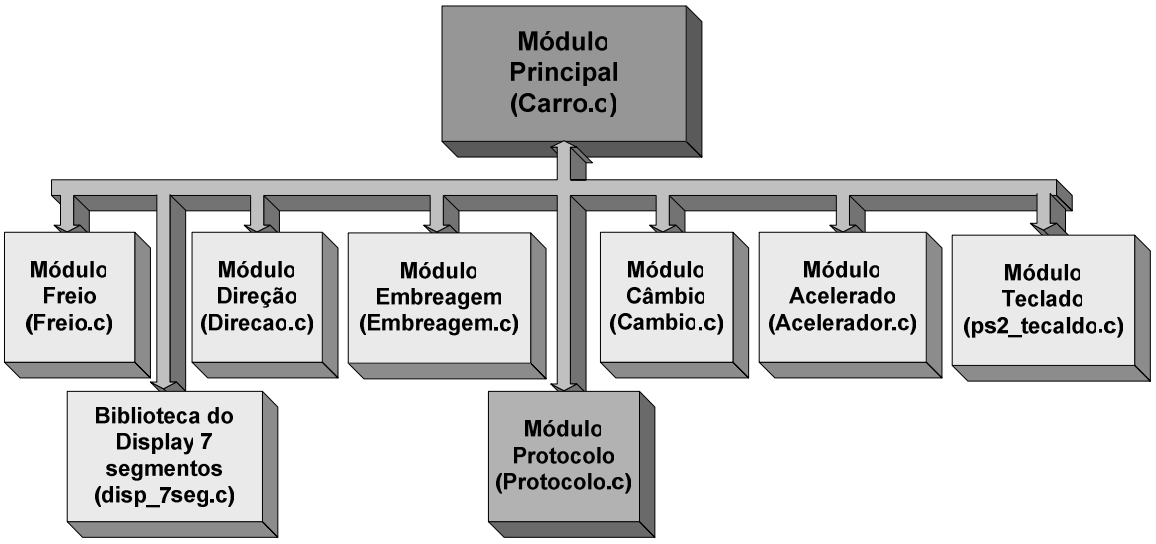


Figura 4.6 – Diagrama do *software* do controlador de movimentação (Simulador)

A tabela 8 mostra a lista dos módulos em software e descreve a função de cada um dos módulos da arquitetura do sistema, implementados em software e responsáveis por controlar a movimentação do veículo de teste e do veículo no simulador.

Tabela 8 – Descrição da função dos módulos do software

Módulo	Descrição
carro.c	Módulo principal de execução que contém a função main ( ). È responsável por chamar as funções de entrada de dados e do protocolo de comunicação.
protocolo.c	Módulo que realiza o protocolo de comunicação serial via RS-232 com o Labview.
freio.c	Funções que acionam o motor DC do freio e analisam o sinal vindo do respectivo potenciômetro.
embreagem.c	Funções de acionamento do sistema Autonomy® do Palio.
cambio.c	Funções de acionamento dos dois motores DC presentes no câmbio e seu sensoriamento.
acelerador.c	Funções de acionamento do motor DC presente na borboleta do carro de acordo com o sinal de retorno do potenciômetro e da rotação.
direcao.c	Funções de acionamento do motor do sistema de direção do carro de acordo com a posição desejada.
interface_usuario.c	Realiza a interface com o usuário através da porta PS/2 e do display de 7 segmentos. Também realiza a interpretação e execução dos comandos chamando cada função específica presentes em outros módulos.
ps2_teclado.c	Biblioteca com as funções básicas para leitura dos *scan_code* do teclado.
disp_7seg.c	Biblioteca com as funções básicas para escrita de caracteres alfas-numéricos no display de 7 segmentos.

Antes da descrição detalhada de cada módulo são apresentadas as bibliotecas e funções utilizadas para cada periférico do sistema do microprocessador embarcado na FPGA. As funções são em sua maioria de escrita e leitura em memória, ou seja, escrita e leitura de ponteiros, no endereço base do periférico.

4.2.1 Funções e Bibliotecas dos Periféricos utilizadas na Programação dos Módulos

As bibliotecas de cada periférico são automaticamente geradas pela ferramenta *LIBGEN* do XPS.

Dois cabeçalhos básicos são utilizados pelos módulos do programa:

- O ***xbasic_types.h***: realiza a redefinição dos tipos básicos de dados como *char*, *int* e *float* para outras denominações (veja a Figura 4.7).

- O ***xparameters.h***: esta biblioteca é gerada automaticamente pela *LIBGEM* e possuem constantes definidas para os endereços dos periféricos do sistema, auxiliando na utilização destes endereços como parâmetros de funções no

software. A definição de um endereço base de um periférico de nome RS232: ***#define XPAR_RS232_BASEADDR 0x40600000***, pode ser citada como por exemplo (veja a figura 4.7).

```
typedef unsigned char   Xuint8;     /**< unsigned 8-bit */
typedef char            Xint8;      /**< signed 8-bit */
typedef unsigned short  Xuint16;    /**< unsigned 16-bit */
typedef short           Xint16;     /**< signed 16-bit */
typedef unsigned long   Xuint32;    /**< unsigned 32-bit */
typedef long            Xint32;     /**< signed 32-bit */
typedef float      Xfloat32;   /**< 32-bit floating point */
typedef  double     Xfloat64;     /**<  64-bit  double  precision
floating point */
```

Figura 4.7 – Utilização do *xbasic_types.h*

Além dos endereços bases, são definidas constantes para as ID's dos periféricos, máscaras de interrupção e parâmetros importantes.

a) Periférico Opb_Gpio – *xgpio_l.h*

Foram utilizadas funções de baixo nível, ou seja, funções de leitura e escrita em registradores do periférico. Assim para saída de dados, uma escrita no registrador é feita no periférico, e para entrada de dados, uma leitura do registrador é feita.

Abaixo seguem as funções de baixo nível utilizadas:

- **xgpio_mgetdatareg –** *BaseAddress, Channel*

  Realiza a leitura do registrador de dados do canal *Channel* do periférico GPIO com endereço base *BaseAddress*. Retorna o valor lido do registrador.

- **xgpio_msetdatareg** – *BaseAddress, Channel, Data*

  Realiza a escrita do valor Data no registrador de dados do canal *Channel* do periférico GPIO com endereço base *BaseAddress*;

b) periférico opb_uartlite – *xuartlite_l.h*

Essa biblioteca é utilizada para o envio e recepção de caracteres pela interface serial UART. Assim utilizamos somente três funções: uma para envio, outra para recepção e uma para verificar se há dados no buffer FIFO de recepção do periférico.

Funções utilizadas na elaboração do periférico opb_uartlite:

- **xuartlite_misreceiveempty –** *baseaddress*: Verifica se há dados no buffer FIFO de recepção do periférico. Retorna ‘0’ caso haja dados e ‘1’ caso contrário.
- **xuartlite_sendbyte –** *xuint32 baseaddress,xuint8 data*: Envia a palavra de 8 bits *Data* para o *buffer* de entrada do periférico UART do endereço base *BaseAddress*;
- **xuartlite_recvbyte –** *xuint32 baseaddress*: Retorna a palavra de 8 bits presente no *buffer* de recepção do periférico UART do endereço base *BaseAddress*.

Caso o periférico opb_uartlite esteja configurado como interface STDIN e/ou STDOUT, então funções padrões de entrada e saída como *printf()* e *scanf()* podem ser usadas. Porém estas funções ao serem compiladas, resultam em um grande número de instruções e, portanto, não são adequadas para um sistema embarcado. A Xilinx fornece alternativas para a função *printf()*. A função: *xil_printf()* resulta em apenas 1KB de programa, porém não suporta a impressão de números em ponto flutuante (%f) ou do tipo *long* de 64 bits.

c) **periférico opb_timer –** ***xtmrctr_l.h***: São utilizadas 2 funções: uma para escrita no registrador de status (TCSR) e outra para escrita no registrador de *load* (TLR).

Funções utilizadas na implementação dos registradores:

- ***xtmrctr_msetcontrolstatusreg*:** Escreve o valor *RegisterValue* no registrador TCSR do timer de número *TmrCtrNumber* (0 ou 1) no OPB_TIMER do endereço base *BaseAddress*. Esse valor escrito irá configurar o modo de operação do timer/contador. Para auxílio na configuração de cada bit do registrador, existem máscaras (constantes) definidas no cabeçalho *xtmrctr_l.h*, ou seja, realizando vários *OU* Lógicos das máscaras consegue-se configurar o timer da maneira apropriada;
- ***xtmrctr_msetloadreg***(*BaseAddress,TmrCtrNumber,RegisterValue*)
  Escreve o valor *RegisterValue* no registrador TLR do *timer* de número

*TmrCtrNumber (*0 ou 1) no OPB_TIMER do endereço base *BaseAddress*. Essa função foi utilizada para configuração da freqüência e ciclo de trabalho dos sinais de PWM utilizados no sistema.

d) periférico opb_7segled – *disp_7seg.c*: Foram mapeadas diversos caracteres alfa-numéricos para exibição no display de 7 segmentos da placa S3-SB.

Os caracteres mapeados são os seguintes:

A, B, C, D, E, F, H, I, J, L, O, P, S, U, R, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

As funções utilizadas foram as seguintes:

- **dispLED –** *Xuint32 Val, Xuint8 DP*: Faz a conversão de um número *Val* de 32 bits para a forma BCD com 4 números de 8 bits (milhar, centena, dezena e unidade) e realiza a escrita apropriada no registrador do periférico opb_7segled. O parâmetro DP indica a posição do *dot point* no display de 7 segmentos, com o valor de 0 sendo relativo ao dígito menos significativo e 3 ao mais significativo;
- **disp_char_7seg –** *char texto,char pos*: Faz a conversão de um caractere para sua respectiva representação em *display* de 7 segmentos e em seguida escreve esta representação na posição *pos* do display. *Pos = 0*, posição menos significativa, *pos = 3*, posição mais significativa.

e) periférico opb_keyboard – *ps2_teclado.h*: A biblioteca possui funções para realizar o *pooling*, ou seja, verifica se há dados (*scan_code* do teclado). Basicamente são funções que fazem leitura e escrita no registrador do *opb_keyboard*.

As funções utilizadas foram as seguintes:

- **boolean tem_dado_ps2 –** *endereco*: Retorna *true* caso haja *scan_code* disponível no periférico OPB_KEYBOARD do endereço *endereco*, caso contrário retorna *false*;
- **unsigned char get_scan_code –** long endereço: Retorna como *scan_code* o valor do registrador de dados do periférico OPB_KEYBOARD do endereço *endereço*.

4.2.2 Descrição das Variáveis de Controle de Movimentação

Abaixo foram descritas as variáveis de Controle de Movimentação implementadas:

a) **variáveis de sensoriamento do tipo potenciômetro**: são variáveis de 8 bits do tipo inteiro sem sinal (0 a 255).

- *ang_dir1* e *ang_dir2* – Posição das colunas de direção;
- *posx_cambio* e *posy_cambio* – Posição do eixo x e do eixo y do câmbio respectivamente;
- *pos_borboleta* – Posição da borboleta do acelerador;
- *pos_pedal_embreagem* – Posição do pedal da embreagem;
- *pos_freio* – Posição do pedal do freio.

b) **variável dos sinais de rotação, deslocamento hodométrico e contato da borboleta**:

- sinal_rdb – 3 bits para os sinais de rotação do motor, deslocamento hodométrico e contato da borboleta (veja a Figura 4.8).

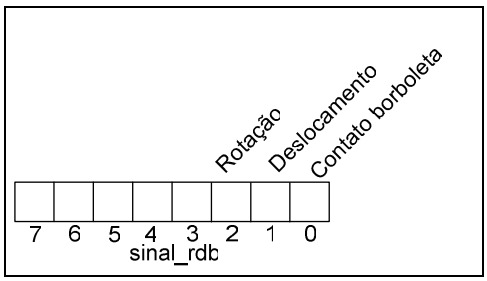


Figura 4.8 - Variável *sinal_rdb*

c) **variáveis de ativação dos atuadores**: são variáveis de 8 bits do tipo inteiro sem sinal (0 a 255):

- *saida_embreagem*: são os sinais para controle do sistema Autonomy® , que é composto de 2 eletro-válvulas (2 bits) e um motor de passo (4 bits);
- *saida_cambio*: são os sinais para os dois motores presentes no câmbio: 1 no eixo X e outro no eixo Y. Cada motor possui 1 sinal PWM e 1 de controle de sentido de rotação ou direção;

- *saida_fda*: são os sinais de controle dos motores DC presentes no freio, na coluna de direção e na borboleta do acelerador. Cada motor possui 1 sinal PWM e 1 de controle de sentido de rotação ou direção.

A Figura 4.9 ilustra a posição de cada bit em cada uma destas variáveis, o que facilita fazer a leitura dos valores da cada uma de forma precisa.

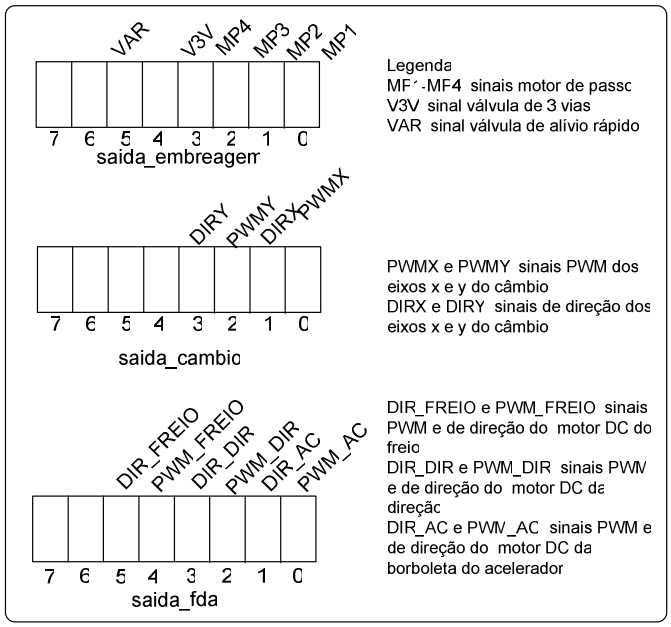


Figura 4.9 – Variáveis de controle dos atuadores

4.3 O MÓDULO CONTROLADOR GERAL (*carro.c*)

É o bloco que contém a função *main()* e portanto é o bloco básico e de execução inicial do *software.* O ciclo de execução é mantido por meio de um laço de repetição (*while*), presente na função *main().* A Figura 4.10 ilustra um fluxograma de execução do módulo *carro.c*.

A execução começa com a inicialização de variáveis e dos temporizadores que são então configurados para o modo PWM. Então o programa entra em um laço de execução na espera de um sinal vindo do ambiente de simulação Labview para realizar a sincronização da comunicação serial. Este sinal é enviado ao iniciar a execução do ambiente de simulação.

Ao receber o sinal vindo do ambiente de simulação desenvolvido no Labview, o programa sai do laço e continua seu fluxo de execução. Em seguida faz uma atualização dos *leds* para indicação de execução:

a) **led(0) acesso**: esperando sinal do Labview;

b) **led(1) acesso**: sinal recebido e sistema executando normalmente.

Desta maneira, o módulo *carro.c* realiza uma chamada às funções *serial_send()* e *serial_receive()* do *protocolo.c* para realizar o envio e recepção dos dados serialmente.

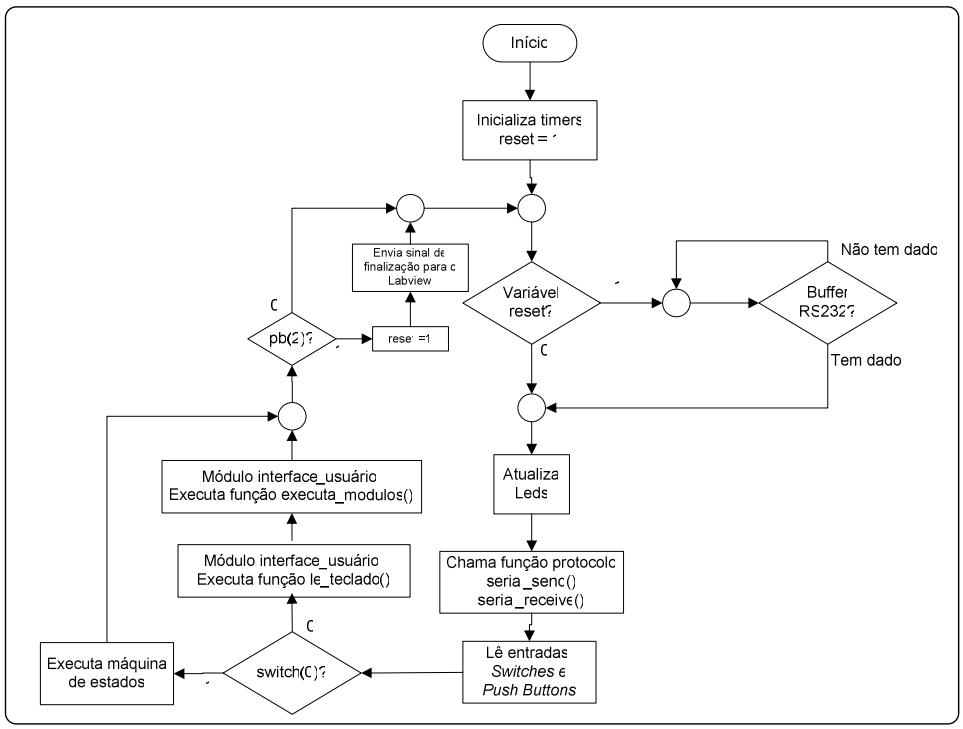


Figura 4.10 **–** Fluxograma módulo *carro.c*

As variáveis referentes aos dados dos *switches* e *push buttons* são atualizadas. O valor do *switch(0)* é testado, e caso esteja ativado, o fluxo de execução passa por uma máquina de estados onde cada estado realiza uma chamada à uma função de execução. Caso não esteja ativado, o módulo *carro.c* irá realizar uma chamada à função *le_teclado()* e em seguida uma chamada para o módulo *executa_modulos()*. A *função le_teclado(),* que será detalhada mais adiante, realiza a leitura e interpretação de dados vindos do teclado e o módulo *executa_modulos()* realiza o *pooling* de um vetor de execução de módulos. Na seção 4.10 deste capítulo as duas funções serão melhor detalhadas**.**

Em seguida o *push-button*(2) é testado, e caso esteja ativado a variável *reset* será setada (‘1’) e envia um sinal de requisição de finalização para o Labview. Então o Labview recebe o sinal, identifica a requisição e finaliza sua execução. O Labview também pode finalizar a execução e enviar um sinal, que será identificado como uma requisição de finalização e a variável *reset* será habilitada.

O fluxo de execução principal do sistema pode ser resumido na figura 4.11. O fluxo se resume a: (a) envio de variáveis de controle dos atuadores, (b) atualização das variáveis referentes aos potenciômetros e demais sinais de sensoriamento, (c) identificação de comandos e processamento.

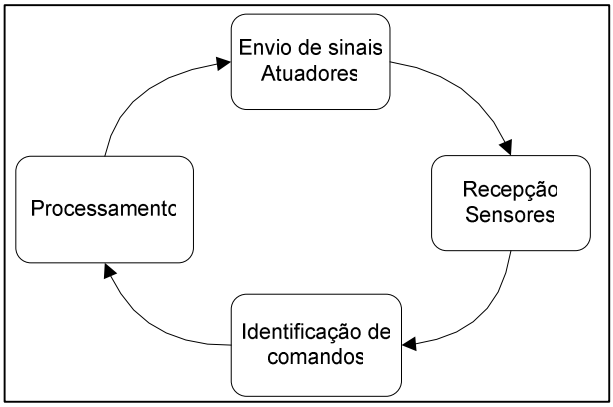


Figura 4.11 – Fluxo de execução resumido - Módulo *carro.c*

4.4 MÓDULO CONTROLADOR DA DIREÇÃO (*direção.c*)

Este módulo é responsável pela ativação do atuador presente na estrutura de direção do carro e posicionamento das rodas de acordo com a posição desejada. A função única deste módulo é a função *é descrita a seguir:*

- *direção(Xuint8 sentido,Xuint8 angulo*): utilizando as posições atuais das rodas nas variáveis *ang_dir1* e *ang_dir2*, a função realiza a ativação do atuador até o ângulo do *sentido*. O sentido *D* (direita) ou *E* (esquerda) é referente à direção das rodas. A figura 4.12 ilustra melhor essa definição de posicionamento da roda.

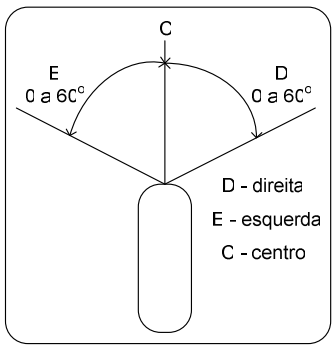


Figura 4.12 – Ângulo de posicionamento da roda do carro

A ativação do motor é feita através do ajuste dos bits 2 e 3 da variável *saida_fda,* lembrando que o bit 3 é referente ao sinal de direção e o bit 2 ao sinal de PWM. Como foi observado no veículo, a atuação do motor de acionamento da direção é de baixa velocidade, não necessitando controle de velocidade por meio do PWM. Assim o PWM da direção sempre terá ciclo de trabalho de 100%, resumindo temos:

- atuação do motor sentido “direita”: *saida_fda* (3..2) = “11”;
- atuação do motor sentido “esquerda”: *saida_fda* (3..2) = “01”;
- motor parado: *saida_fda* (3..2) = “00” ou “10”.

4.5 MÓDULO CONTROLADOR DO ACELERADOR (*acelerador.c*)

Este módulo é responsável pela ativação e controle do atuador de deslocamento linear presente na borboleta. Pode ser feito um controle por posição do curso da borboleta ou pelo número de pulsos por segundo referentes ao número de rotações do motor desejada. A descrição da função é dada a seguir:

a) **Função *Acelerador (char opcao,Xuint16 valor)***: permite controlar a posição da borboleta de 0% aceleração mínima a 100% aceleração máxima. O percentual escolhido é determinado pelo parâmetro *valor* e a *opcao* deve ser igual a ‘B’.

A função também pode atuar na borboleta até que a rotação do motor atinja o valor dado pelo parâmetro *valor* (em RPM). Para isso o parâmetro *opcao* deve ser igual a ‘R’.

Os bits 0 e 1 da variável *saida_fda* são utilizados como sinais de controle do motor. O bit 0 é o sinal de PWM e o bit 1 o sinal de direção. A variável que contém o PWM do motor é a variável *pwm_ac*. Portanto:

- atuação positiva na borboleta (aumento no giro do motor): *saida_fda(0) = pwm_ac* e *saida_fda(1) = ‘1’*;
- atuação negativa na borboleta (redução no giro do motor): *saida_fda(0) = pwm_ac* e *saida_fda(1) = ‘0’*;
- borboleta parada (giro idealmente constante): *saida_fda(0) = ‘0’*.

4.6 MÓDULO CONTROLADOR DO FREIO (*freio.c*)

Este módulo é responsável pela ativação do atuador de acionamento do pedal do freio, sua função é descrita a seguir:

a) **Função *Freio (int freia)*:** ativa o freio até chegar à sua posição máxima caso o parâmetro *freia = '1'* ou desativa o freio até chegar em sua posição inicial caso o parâmetro *freia* = '0'. A posição é lida através da variável *pos_freio*.

   A ativação do motor é feita através do ajuste dos bits 4 e 5 da variável *saída_fda*, lembrando que o bit 5 é referente ao sinal de direção e o bit 4 ao sinal de PWM controlado pela variável *pwm_ac*. O mesmo PWM é utilizado pelo módulo do acelerador (o sistema não freia e acelera ao mesmo tempo).

   Abaixo são descrita as variáveis de saída da função do freio:

   - ativação do freio: *saida_fda(4) = pwm_ac* e *saida_fda(5) = '1';*
   - desativação do freio: *saida_fda(4) = pwm_ac* e *saida_fda(5) = '0';*
   - freio inoperante: *saida_fda(4)* = '0'.

4.7 MÓDULO CONTROLADOR EMBREAGEM (*embreagem.c*)

O módulo que controla a embreagem recebe os sinais do controlador de movimentação e executa a atuação da embreagem por meio do controle do sistema Autonomy®.

Segundo (Bellardi T., 2005) e (Garrido R., 2001), para a ativação da embreagem o sistema deve ativar a eletro-válvula da válvula de 3 vias: *saida_embreagem(3) = '1'*. Para liberação rápida o sistema deve ativar a válvula de descarga rápida: *saida_embreagem(5) = '1'*.

Para a liberação gradual o sistema deve ativar um motor de passo obturador, que irá aumentar o reduzir uma abertura para liberação do ar contido na câmara do sistema Autonomy®. Os sinais para o controle do motor de passo são os bits 0 a 3 da variável *saida_embreagem*. Para o correto funcionamento do motor de passo, devem seguir a seqüência da variável *saida_embreagem(3..0)*: '0101', '0110', '1010', '1001', '0101'.

A Função principal do módulo da embreagem é descrita a seguir:

- ***embreagem (char opcao, unsigned char pos):*** esta função realiza a ativação da embreagem (*opcao = 'A')*, liberação rápida da embreagem (*opcao = 'R')* e liberação controlada da embreagem (*opcao = 'C')* de 0 a 100% (parâmetro *pos*) utilizando o sinal de retorno do potenciômetro presente na variável *pos_embreagem*.

Além da função principal, o módulo *embreagem.C* controla diretamente o motor de passo, aumentando ou diminuindo a quantidade de ar que sai da câmara posterior do cilindro de acionamento pneumático do sistema Autonomy® .

4.8 MÓDULO CONTROLADOR CÂMBIO (*cambio.c*)

Módulo responsável pela ativação dos 2 atuadores instalados para o acionamento da direção X e Y da caixa de câmbio. Um motor controla o deslocamento do câmbio no eixo-X e o outro controla no eixo-Y (ver figura 4.13).

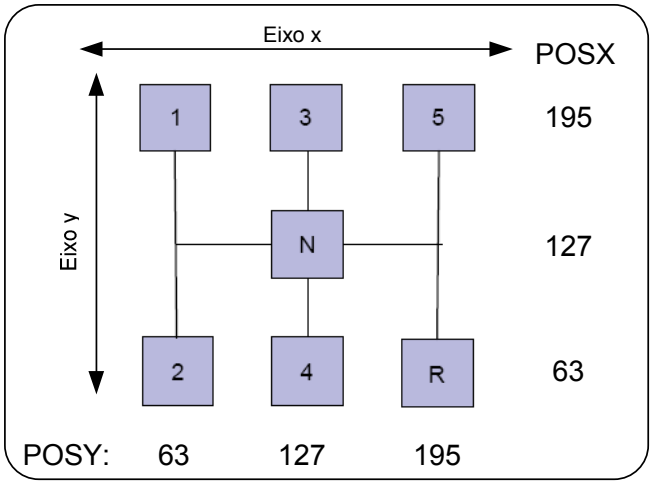


Figura 4.13 – Posições e eixos do câmbio

A função principal para o controle do câmbio é descrita a seguir:

- ***câmbio (char marcha)***: esta função utiliza os sinais de retornos dos potenciômetros *posx_cambio* e *posy_cambio* para realizar o controle de posição de cada eixo e assim posicionar a alavanca do câmbio corretamente na marcha desejada (parâmetro *marcha*).

Além do anterior, foram pré-definidas constantes para indicação das posições críticas da caixa de câmbio como mostrado na figura 4.13. Os valores não foram baseados nas leituras dos potenciômetros do carro, prevendo uma futura adaptação dos valores reais presentes no carro. Estes valores também são os utilizados na plataforma de simulação do Labview. Somente as posições “Ré”, “Neutro” e “1ª marcha” foram implementadas no módulo, prevendo inicialmente somente o uso destas, sendo que são as necessárias para estacionar.

Os bits de 0 a 3 da variável *saida_cambio* são utilizados como sinais para o controle dos 2 motores DC’s (2 bits para cada). Os bits 0 e 2 são referentes aos sinais PWM dos motores da posição X e Y, respectivamente. Os bits 1 e 3 são referentes aos sinais de direção dos motores da posição X e Y respectivamente. As variáveis *pwmx* e *pwmy* referenciam os sinais PWM de cada motor.

A descrição das saídas da função *cambio.c* são dadas a seguir:

- alavanca do câmbio movendo para a direita: *saida_cambio(0) = ‘1’ e saida_cambio(1) = ‘1’;*
- alavanca do câmbio movendo para a esquerda: *saida_cambio(0) = ‘1’ e saida_cambio(1) = ‘0’;*
- alavanca do câmbio movendo para cima: *saida_cambio(2) = ‘1’ e saida_cambio(3) = ‘1’;*
- alavanca do câmbio movendo para baixo: *saida_cambio(2) = ‘1’ e saida_cambio(3) = ‘0’;*
- alavanca parada: *saida_cambio(0) = ‘0’ e saida_cambio(2) = ‘0’;*

4.9 O MÓDULO PROTOCOLO (*protocolo.c*)

O protocolo deve ser capaz de enviar três variáveis para o ambiente de simulação de forma serial: (a) *saída_embreagem*, (b) *saída_cambio* e (c) *saída_fda*. Além do anterior recebe as oito palavras referentes aos sinais dos transdutoresO: (a) *ang_dir1*, (b) *ang_dir2*, (c) *pos_embreagem*, (d) *pos_freio*, (e) *pos_borboleta*, (f) *posx_cambio*, (g) *posy_cambio* e (h) *sinal_rdb*.

A solução encontrada foi: enviar sempre três palavras de oito bits por meio do MicroBlaze e receber duas palavras de 8 bits. Esses bits fazem a requisição de uma das

variáveis de sensoriamento, com exceção do *sinal_rdb*. A figura 4.14 ilustra com detalhe o envio das 3 palavras:

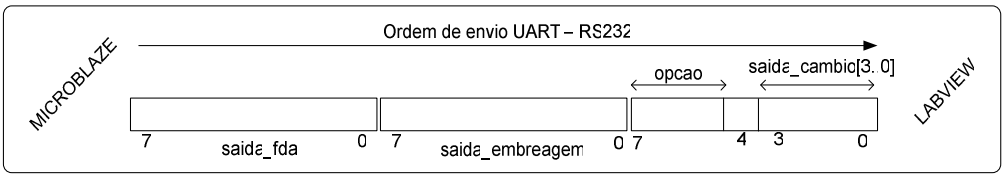


Figura 4.14 – Protocolo para envio das palavras pelo Microblaze

Os três bits de requisição são controlados pela variável *opcao*. E foram definidos como constantes no *HEADER protocolo.h*:

- *rq_dir1*: Valor 32 e faz a requisição da variável *ang_dir1*;
- *rq_dir2*: Valor 64 e faz a requisição da variável *ang_dir2*;
- *rq_xcambio*: Valor 96 e faz a requisição da variável *posx_cambio*;
- *rq_ycambio*: Valor 128 e faz a requisição da variável *poxy_cambio*;
- *rq_borboleta*: Valor 160 e faz a requisição da variável *pos_borboleta*;
- *rq_pedal_embreagem*: Valor 192 e faz a requisição da variável *pos_embreagem*;
- *rq_pos_freio*: Valor 214 e faz a requisição da variável *pos_freio*.

O ambiente de simulação recebe as três palavras, processa-as e então retorna duas palavras, uma contendo o código de requisição e os três bits do *sinal_rdb* e uma palavra contendo a variável requisitada.

O código de requisição é retornado solicitando confirmação para o Microblaze de que a palavra recebida corresponda à palavra que foi solicitada.

Ao receber a variável requisitada o Microblaze incrementa a variável *opção* para requisição da próxima palavra a qual contém a variável.

A Figura 4.15 ilustra as palavras recebidas pelo Microblaze.

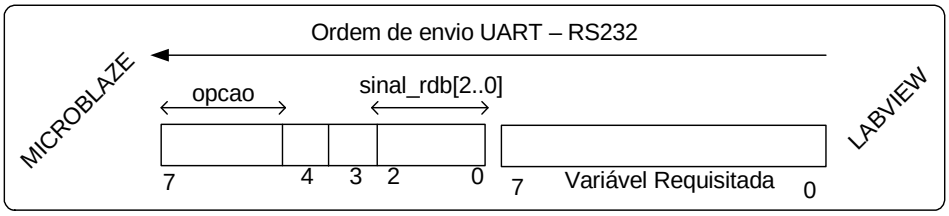


Figura 4.15 – Protocolo - Recepção das palavras pelo Microblaze

4.10 O MÓDULO INTERFACE COM O USUÁRIO (*interface_usuario.c*)

Este módulo implementa um modo de entrada de comandos pelo usuário através do teclado e disponibiliza os caracteres recebidos no display de 7 segmentos. Somente caracteres de possível representação são identificados. Por exemplo: o caractere 'E' está mapeado, enquanto o caractere 'M' não está mapeado pois sua representação em 7 segmentos não é possível. Os caracteres utilizados foram descritos na tabela 10.

As Funções principais do módulo de interface são as seguintes:

a) ***le_teclado()***: Verifica se alguma tecla do teclado foi acionada e realiza a identificação e tratamento (função trata_scan_code(*scan* code)) do *scan_code*.

Abaixo são descritos os dois modos de operação do módulo interface:

- modo de entrada de comandos em linha (variável *comando_manual = '0'*): neste modo o usuário deve entrar com comandos pré-definidos, que são automaticamente disponibilizados no display de 7 segmentos. Assim o usuário entra com o comando, normalmente composto de uma palavra chave de 1 ou 2 parâmetros, e então confirma pressionando a tecla ENTER. O comando é identificado e executa o módulo de controle. Na função *le_teclado* o *scan_code* é recebido e identificado sendo a função *trata_scan_code* chamada para tratamento adequado do *scan_code*.
- modo de comandos manuais (variável *comando_manual = '1'*): Este modo implementa a execução dos módulos de controle através do simples pressionar das teclas do teclado. Exemplo: ao pressionar a tecla "seta para cima" (↑) a função irá acionar o módulo do acelerador, aumentando a rotação do motor. A variável *flag* permite executar ações resultantes do pressionamento inicial da tecla ou do pressionamento contínuo da tecla (em cada recebimento do *scan_code*) ou da liberação da tecla.

b) ***trata_scan_code****(char sc)*: Trata o *scan code sc* recebido. Para armazenar os caracteres recebidos e disponibilizá-los no display, uma variável do tipo vetor buffer_7seg[8] implementa um *buffer* de 8 caracteres (char). Caso o *scan code* seja alfa-numérico, o *buffer* sofre um deslocamento (*shift*) para a esquerda e o caractere é adicionado ao *buffer* na posição 0. Caso o *scan_code* seja referente a tecla ENTER, a função interpreta_comando() é chamada. E caso o *scan_code* seja proveniente da tecla BACKSPACE, o último caractere inserido no *buffer* é removido (posição 0) e um deslocamento para a direita é realizada no vetor .

c) ***interpreta_comando()*:** Reorganiza a variável *buffer_7seg*[8] em outra variável *comando*[8]. Exemplo: se o comando ACB34 for digitado pelo usuário, o vetor *buffer_7seg[]* será 4, 3, B, C, A, e o vetor *comando[]* será A, C, B, 3, 4. A função também identifica a posição do 1º algarismo numérico, identificando o parâmetro e guardando seu valor na variável *param_num*. Se não houver erros na composição do comando, a função é finalizada chamando outra função: *exec_comando().*

d) ***exec_comando()***: Realiza a identificação da sintaxe do comando e aloca chamadas aos módulos de execução em uma matriz de *pooling*. Essa matriz é composta por 3 vetores: as variáveis *exec_modulo[0..1]*, *param1[0..1]* e *param2[0..1]* de 2 posições cada, portanto, uma matriz 3 x 2. O pooling é realizado pela função *executa_modulos()* que é sempre chamada no laço de repetição do módulo principal.

A tabela 9 descreve as sintaxes para os comandos que o operador pode enviar ao sistema por meio da interface juntamente com os seus parâmetros:

Tabela 9 – Sintaxes dos comandos suportados

Sintaxe	Param1 x	Param2 y	exec_modulo[k]	param1[k]	param2[k]
DIxy	‘D’,‘E’ ou ‘C’	Inteiro de 0 a 60	‘D’	x	y se x != ‘C’ 0 se x = ‘C’
FREx	‘0’ ou ‘1’	-	‘F’	x	-
ACxy	‘B’ ou ‘R’	0 a 100 se x = ‘b’ 0 a 9999 se x = 'r'	‘A’	x	y
CABxy	‘0’ a ‘6’	-	‘C’	‘N’ se x = 0 ‘R’ se x = 6 x se 0<x<6	-
EBxy	‘A’,’R’ ou ‘C’	0 a 100 se x = ‘C’	‘E’	x	0 se x != ‘C’ y se x= ‘C’
Cx ou Fx	‘A’,’C’ ou ‘F’	0 a 100 para Cx 1 a 9999 para Fx	-	-	-
Pxy	‘R’ ou ‘A’	Inteiro positivo	‘P’	x	y

A Tabela 10 descreve todas as linhas de comandos disponíveis, sua sintaxe e parâmetros para a execução de ações.

Tabela 10 – Descrição das linhas de comandos e suas ações

Linha de comandos			
Sintaxe	**Param1 x**	**Param2 y**	**Ação**
DIxy	‘D’,‘E’ ou ‘C’	Inteiro de 0 a 60	Posiciona as rodas em x graus à direita (x = D) ou à esquerda (x = E) ou (x = C) alinha as rodas.
FREx	‘0’ ou ‘1’	-	FRE1 - Ativa o freio FRE0 - Desativa o freio
ACx	‘B’ ou ‘R’	0 a 100 se x = ‘b’ 0 a 9999 se x = ‘r’	ACBy – Aciona a borboleta y% de seu curso total. ACRy – Aciona a borboleta até atingir a rotação y.
CABxy	‘0’ a ‘6’	-	CABx – Posiciona o câmbio na marcha desejada
EBxy	‘A’,’R’ ou ‘C’	0 a 100 se x = ‘C’	EBA – Ativa a embreagem EBR – Libera rapidamente a embreagem EBCy – Libera a embreagem até y % de seu curso.
Cx ou Fx	‘A’,’C’ ou ‘F’	0 a 100 para Cx 1 a 9999 para Fx	Cxy – Altera o ciclo de trabalho do sinal PWM para y% do freio (x = F), do acelerador (x = A) ou do câmbio (x = C) Fxy – Altera a freqüência do sinal PWM do freio (x = F), do acelerador (x = A) ou do câmbio (x = C) para y Hz (Labview) ou y kHz (Carro direto).
Pxy	‘R’ ou ‘A’	Inteiro positivo	PAy – Avança y o motor de passo da embreagemPRy – Retrocede y passos do motor de passo da embreagem

A Tabela 11 descreve todos os comandos manuais disponíveis.

Tabela 11 – Descrição dos comandos manuais e suas ações

Comandos Manuais	
Sintaxe	**Ação**
↑	Aciona borboleta positivamente (aumento da rotação do motor)
↓	Aciona borboleta negativamente (redução da rotação do motor)
←	Aciona direção para a esquerda
→	Aciona direção para a direita
W	Movimenta alavanca do câmbio para cima
S	Movimenta alavanca do câmbio para baixo
A	Movimenta alavanca do câmbio para esquerda
D	Movimenta alavanca do câmbio para direita
Espaço pressionada	Ativa freio
Espaços liberada	Desativa freio
E	Aciona embreagem
Q	Libera embreagem rapidamente
1	Posiciona câmbio na marcha 1
N	Posiciona câmbio no ponto morto ou neutro
R	Posiciona câmbio na marcha ré
>	Avança motor de passo
<	retrocede motor de passo

e) ***executa_modulos()***: Essa função faz parte do laço de repetição no módulo principal carro.c e tem a função de executar o *pooling* na matriz de *pooling*. O vetor *exec_modulo[0..1]* é checado e se a posição k estiver setada, a função realiza as chamadas apropriadas das funções com os parâmetros que estiverem presentes em *param1[k]* e *param2[k]*. Atualmente o tamanho da matriz de *pooling* é de 2 posições, ou seja, somente 2 módulos podem ser executados ao mesmo tempo. De acordo com a necessidade esse parâmetro pode ser alterado através da constante *tam_pooling*.

4.11 RESULTADOS OBTIDOS COM O SISTEMA DE CONTROLE DE MOVIMENTAÇÃO

Os resultados obtidos com a implementação do sistema de controle de movimentação são mostrados nas figuras 4.16 à 4.28, representam as ligações das LUT´s da Spartan-3 formando assim o controlador de movimentação.

A figura 4.16 mostra a rota das ligações das LUT's que formam o microblaze. Observando essa figura pode se ter idéia da ocupação dos recursos lógicos demandados

com o microprocessador microblaze.

No apêndice C, a tabela utilização dos recursos da FPGA – Microblaze, mostra a utilização dos recursos da FPGA Spartan-3. Os percentuais são relativos aos recursos disponíveis da Spartan-3 e os recursos utilizados para a implementação do Microblaze.

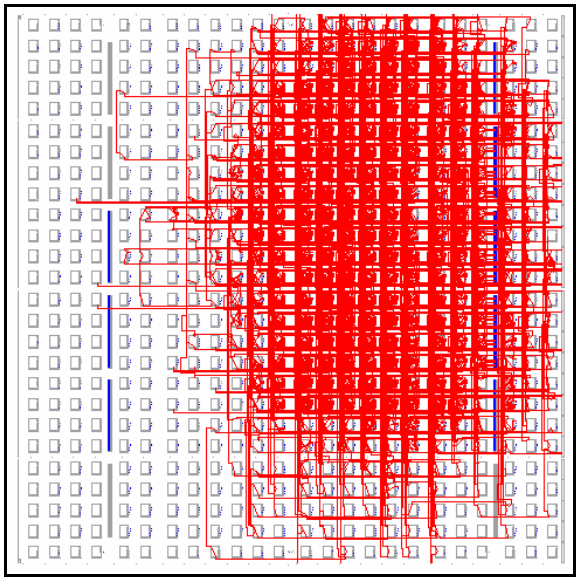


Figura 4.16 – Ligações das LUT's – Microprocessador Microblaze.

A tabela limite de freqüência do *clock* em cada módulo no apêndice C mostra a freqüência máxima de cada módulo. O clock máximo é calculado em condições normais de funcionamento, temperatura, vibração e alimentação.

No caso do Microblaze o clock máximo calculado é de 91,128Mhz. O clock limite da Spartan-3 é de 50Mhz, mostrando que o Microblaze não limitará a velocidade do sistema implementado. Na figura 4.17, observa-se a rota das ligações das LUT's das entradas e saídas, *Input Output* (I/O) da FPGA para o Microprocessador Microblaze. A maior concentração das ligações estão localizadas na periferia da área útil da FPGA e a área central.

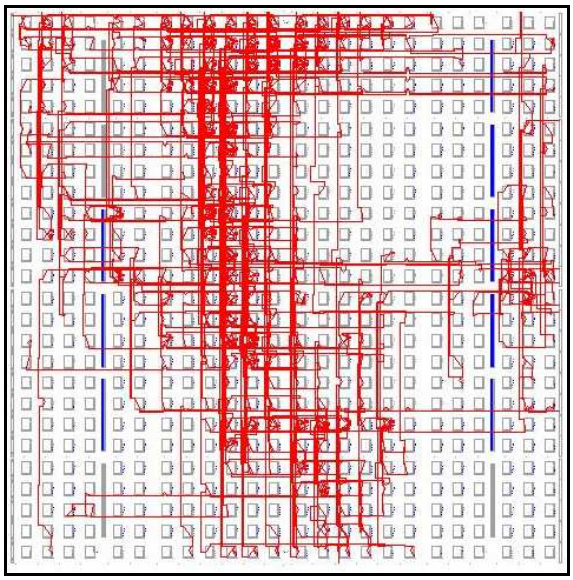


Figura 4.17 – Ligações das LUT's – Interface I/O.

A figura 4.18 mostra a rota das ligações das LUT's formando o periférico PS2 responsável por receber os comandos do teclado. Na figura 4.18, vê-se a ocupação dos recursos lógicos demandados para a implementação do periférico PS2 na FPGA.

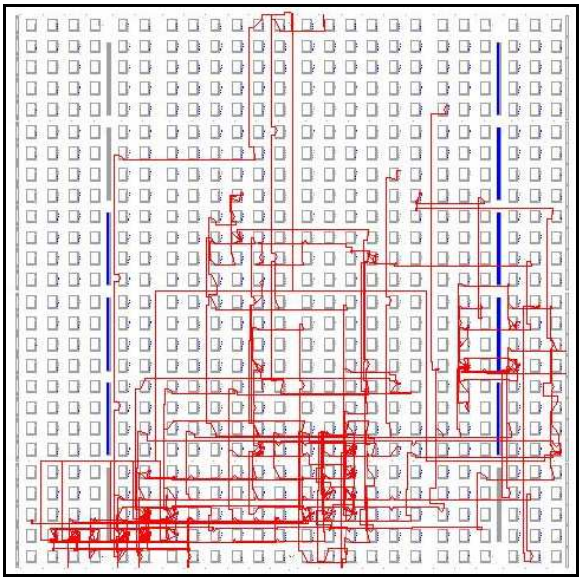


Figura 4.18 – Ligações das LUT's – PS2 teclado.

No apêndice C, no tópico periféricos, a tabela utilização dos recursos da FPGA – Teclado PS2, mostra a utilização dos recursos da FPGA Spartan-3. Os percentuais são relativos aos recursos disponíveis da Spartan-3 e os recursos utilizados para a implementação da interface PS2.

A figura 4.19 mostra a rota das ligações das LUT's formando o barramento de ligação dos periféricos.

Na figura 4.19, vê-se a ocupação dos recursos lógicos demandados para a implementação do barramento de ligação dos periféricos.

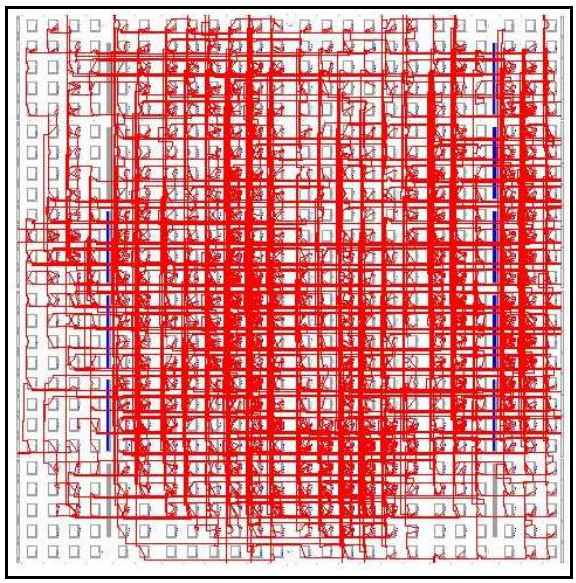


Figura 4.19 – Ligações das LUT's – Barramento dos Periféricos (OPB).

A figura 4.20 mostra a área ocupada da FPGA Spartan-3 pela implementação do controlador de movimentação e as ligações do MicroBlaze (Correia A., et al., 2007a).

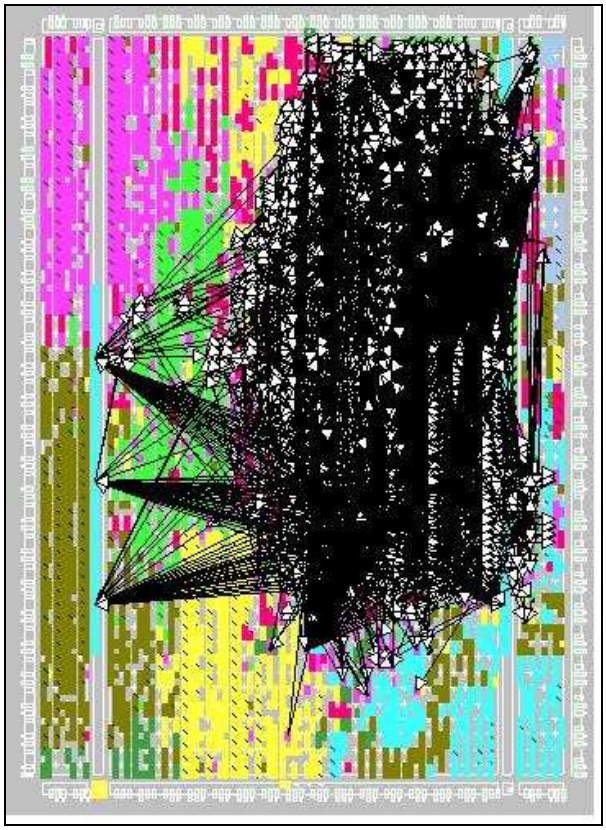


Figura 4.20 – Área de ocupação da FPGA Spartan-3.

4.12 CONCLUSÕES DO CAPÍTULO

Neste capítulo foram abordado os processos de referentes ao projeto e a configuração do controlador de movimentação embarcado em lógica reconfigurável. Também foram apresentadas as ferramentas aplicadas para a configuração do microprocessador responsável pelo controle de movimentação.

Durante a implementação do controlador diferentes fases foram realizadas desde a definição do desenho da arquitetura do controlador de movimentação. Após a definição da arquitetura iniciou-se o estudo para a definição da implementação dos periféricos necessários para que com o processador Microblaze controlasse os atuadores de movimentação obedecendo aos comandos enviados por meio de um teclado.

Os recursos utilizados na elaboração do sistema de controle de movimentação foram aproveitados ao máximo dentro do oferecido pelo kit de desenvolvimento da Xilinx da família Spartan-3. Foram apresentados os conceitos aplicados na configuração do processador Microblaze e os periféricos básicos oferecidos pela ferramenta EDK.

Outros recursos foram utilizados além dos já oferecidos basicamente durante a configuração dos periféricos do MicroBlaze. Um dos recursos não básicos que foram aplicados foi a configuração do PWM para controlar a velocidade dos motores de acionamento dos atuadores.

Um conceito muito importante que foi utilizado durante a configuração dos recursos de hardware foi a utilização de um periférico configurado em VHDL para o controle do protocolo de comunicação com o teclado. Uma vez configurado em VHDL este periférico foi incerido ao sistema e ligado ao barramento de periféricos. Esta função se mostrou importante pois abre precedentes para a configuração de periféricos com especificações conforme as necessidades do projetista com as vantagens de ser um hardware e assim podendo funcionar de forma paralela.

Uma vez configurado o hardware e seus periféricos, assim como um microcontrolador, foram programados os módudos em software. Cada um desses módulos programados seguiram as especificações definidas na arquitetura do sistema.

Possuem funções próprias recebendo parâmetros e estímulos do controlador geral executando suas funções de controle de forma local**.** Na programação foram

utilizadas funções especificas para o microprocessador Microblaze da Xilinx como: acesso ao barramento de periféricos, acesso aos registradores.

Adicionalmente neste capítulo, foi descrito o funcionamento e a arquitetura cada um dos módulos da arquitetura do sistema de controle de movimentação. As variáveis que compõe cada um dos módulos para a entrada, saída e controle dos sinais são descritas e especificadas. Neste tópico também foram descritos os comandos por meio dos quais o operador controla de forma manual ou semi-automática as ações de movimentação do veículo tanto no ambiente de simulação como com o veículo real de testes.

A implementação do microprocessador Microblaze embarcado na FPGA Spartan-3, sem contar com os periféricos, aplicada nesse trabalho utilizou 43% das Slices disponíveis e 29% das LUT's de quatro entradas e com um limite de freqüência de trabalho em 91 Mhz (veja Apêndice C).

Finalmente, foram apresentados resultados relativos as rotas das ligações das LUT's do microprocessador, a interface do microprocessado e a área de ocupação do sistema de controle de movimentação embarcado no kit de desenvolvimento Spartan-3.

5. DESENVOLVIMENTO DO AMBIENTE DE SIMULAÇÃO

Esse capítulo apresenta ao leitor informações pertinentes ao projeto e a implementação do ambiente de simulação da cinemática de um veículo de passeio comum, movido por um motor à explosão e equipado com câmbio totalmente mecânico. Mostra detalhadamente a arquitetura e o funcionamento do sistema de simulação em cada um de seus módulos.

O objetivo inicial de projetar e implementar um simulador buscou-se a obtensão de um ambiente capaz de simular as respostas do veículo aos estímulos enviados de um controlador de movimentação, por meio de sinais eletrônicos. Isto é fundamental para testar e validar o controlador de movimentação antes da montagem e teste do mesmo no veículo real.

5.1. A UTILIZAÇÃO DA INSTRUMENTAÇÃO VIRTUAL PARA O DESENVOLVIMENTO DO AMBIENTE DE SIMULAÇÃO

A utilização de uma nova metodologia de ciclo de projeto (empregando-se testes e simulação do controle da cinemática do veículo em um ambiente programado com uma ferramenta para projetos baseado em instrumentação virtual) mostrou-se promissora.

O ambiente de simulação criado no *software* LabView, de forma geral, realiza as seguintes tarefas:

- Obter os sinais recebidos pela porta serial (padrão RS-232). Estes sinais devem ser extraídos do pacote enviado pelo Microblaze usando o padrão de protocolo de comunicação serial que foi especificado conforme padrão em (Bushby S., 1997).
- Tratar os sinais recebidos para enviar comandos ao simulador (para o veículo virtual) de tal forma que o carro simulado se comporte de acordo com os atuadores presentes no carro real. Exemplo: a simulação de um dos atuadores o qual recebe o sinal PWM e de direção, e então realiza o incremento/decremento de uma variável que simula a posição de um

atuador e envia essa variável de volta para o Microblaze usando uma palavra de 8 bits (como um conversor A/D real).

- Relacionar as variáveis de controle de velocidade, posição do veículo, ângulo das rodas e posição dos atuadores para o controle do veículo no ambiente de simulação.
- Simular o movimento do veículo no ambiente de simulação de acordo com os valores de aceleração, velocidade, posição do veículo e o ângulo das rodas.
- Permitir a visualização das variáveis por meio de uma interface na qual o operador pode verificar os valores e seu estado;

5.2 DISTRIBUIÇÃO DOS MÓDULOS NO SISTEMA DE CONTROLE DE MOVIMENTAÇÃO COM O SIMULADOR

A programação realizada no ambiente Labview foi projetada e implementada em módulos, chamados de VI's (*Virtual Instruments*). Cada módulo realiza funções diferentes e comunicam-se por meio de uma VI projetada especificamente para isso, onde estão todas as variáveis globais do sistema.

A figura 5.1 ilustra a arquitetura do sistema e as ligações entre cada um de seus módulos (*módulo serial, módulo de trajetória do carro, módulo variáveis globais, módulo trata sinais*)

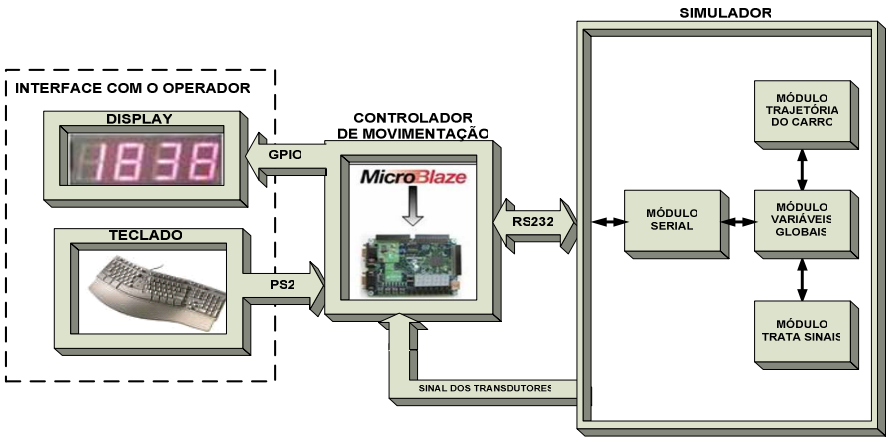


Figura 5.1 – Diagrama de blocos da plataforma do ambiente de simulação no Labview

Cada um dos módulos apresentados na figura 5.1 possuem função específica dentro do funcionamento do ambiente de simulação.

Abaixo é descrita a função de cada um dos módulos projetados e implementados no ambiente de simulação :

- **módulo serial**: recebe os sinais de comando do controlador, enviando os valores das variáveis para o módulo variáveis globais;
- **módulo trajetória do carro**: realiza todas as funções relacionadas à cinemática do veículo na janela do simulador;
- **módulo variáveis globais**: realiza a ligação das variáveis com todos os módulos;
- **módulo trata sinais**: realiza o tratamento dos sinais das variáveis e a apresentação dos valores dos sinais mais importantes para o controle do veículo no ambiente de simulação.

Foram criados quatro módulos VI's formando o ambiente de simulação. Cada qual possui uma funcionalidade definida objetivando dar ao sistema a distribuição dos módulos afim de não sobrecarregar o processamento, tornando o sistema mais rápido.

A figura 5.2 apresenta o detalhamento da estrutura dos módulos programados no ambiente de simulação bem como os seus blocos (cada qual executando uma função específica).

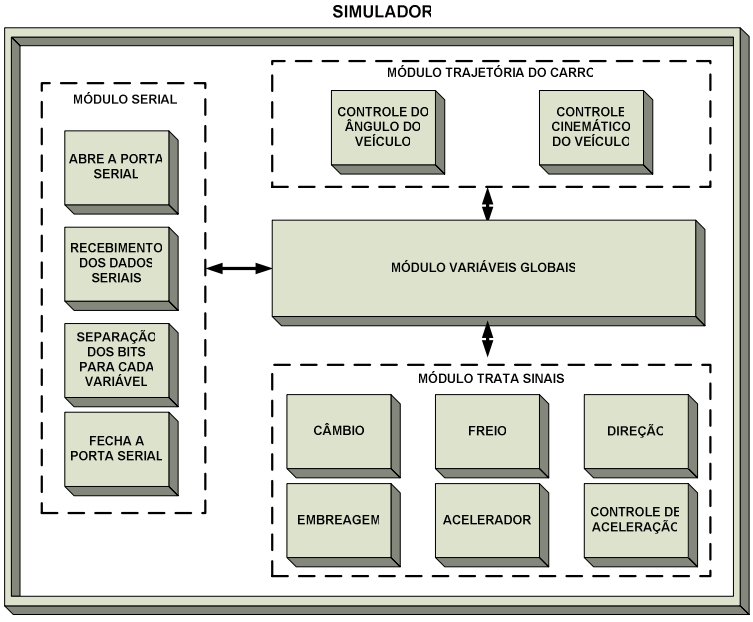


Figura 5.2 – Diagrama dos módulos do ambiente de simulação e seus blocos

A figura 5.2 mostra o diagrama detalhado do ambiente de simulação com seus módulos e os blocos.

Cada módulo é formado por blocos os quais realizam as seguintes funções:

- **abre a porta serial**: responsável por abir a porta serial dando início a recepção dos dados;
- **recebimento dos dados seriais**: responsável por realizar o recebimento e tratamento dos dados enviados para serem separados em palavras e bits;
- **separação dos bits para cada variável**: realiza a separação das palavras em bits (dos dados enviados por meio serial);
- **fecha a porta serial**: responsável por realizar o fechamento da comunicação serial;
- **controle do ângulo do veículo**: bloco responsável pelo controle do ângulo de direção do veículo em relação ao plano cartesiano *x* e *y*;
- **controle cinemático do veículo**: responsável pela parte gráfica do simulador. Este módulo controla todas as ações de movimento do veículo dentro da janela do simulador;
- **câmbio**: responsável por controlar as ações relativas ao câmbio do veículo. Este módulo simula o posicionamento das marchas conforme comandos enviados pelo operador;
- **freio**: responsável pelo controle de acionamento do freio do veículo dentro do ambiente de simulação;
- **direção**: controla todas as funções relacionadas ao controle do atuador da direção, trabalhando em conjunto com o controle do ângulo do veículo. Este módulo recebe o comando do controlador contendo a direção esquerda ou direita e a velocidade do acionamento e muda o ângulo conforme a necessidade;
- **acelerador**: realiza o controle do acionamento do atuador do acelerador conforme os comandos enviados pelo controlador;
- **embreagem**: responsável pelo controle do acionamento do atuador da embreagem, recebendo os comandos do controlador;

- **controle de aceleração**: O controle de aceleração realiza todo o cálculo lógico e matemático para o controle da aceleração do veículo no ambiente de simulação. Recebe e executa os cálculos a partir das informações relativas a posição dos atuadores do acelerador, embreagem, freio e câmbio.

Outra razão pela qual o sistema foi projetado em módulos é o fato de que com poucas modificações esse simulador pode executar outras funcionálidades como: sistema supervisório para o operador do veículo, ambientes de simulação para o teste de diferentes arquiteturas de controladores de movimentação, entre outras. Essas funcionalidades podem ser realizadas com a configuração/adição de um ou mais módulos. Esta característica faz o sistema flexível a modificações e implementações posteriores.

5.3 MÓDULOS DO SISTEMA DE SIMULAÇÃO

5.3.1 Módulo Serial

O módulo serial é responsável pela comunicação do ambiente de simulação com o controlador de movimentação. Este recebe e envia as informações do estado das variáveis e o acionamento dos atuadores, enviando os sinais a cada um dos módulos responsáveis por meio do módulo variáveis globais.

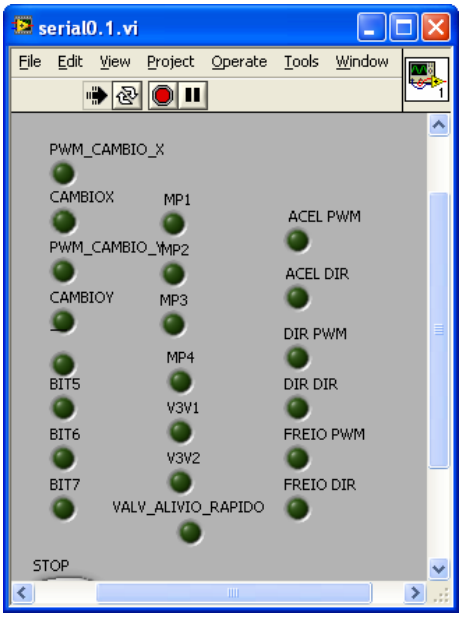


Figura 5.3 – Painel frontal do Módulo de comunicação serial do Simulador no Labview

No painel frontal o operador pode acompanhar a comunicação visualizando os *leds* identificados de acordo com cada variável (acendendo e apagando). A figura 5.3, apresenta o painel frontal do módulo de comunicação serial com suas variáveis.

Além do painel frontal, cada módulo possui o seu diagrama de blocos onde está a programação do módulo VI em questão. O módulo serial possui blocos de configuração da porta serial estabelecendo a sua velocidade da comunicação (que nesse caso é 19.200 bps Bits por segundo). Adicionalmente, realiza a comunicação da UART via porta serial padrão RS-232 com o controlador de movimentação configurado no Microblaze.

Esse módulo também realiza a atribuição de cada bit à sua função específica (PWM e direção) e repassa bit a bit para a VI *Trata Sinais*. Além disso, recebe os valores simulados de potenciômetros e envia para o Microblaze, de acordo com o código de requisição recebido do processador (veja o módulo *protocolo.c*, no capítulo 4).

O diagrama de blocos do módulo de comunicação serial do simulador (figura 5.4) é formado por um seqüenciador com quatro estágios. Cada estágio do seqüenciador contém um ou mais blocos que executam uma função, esse seqüenciador é executado como um laço de repetição onde é executado do primeiro até o quarto índice voltando logo ao primeiro.

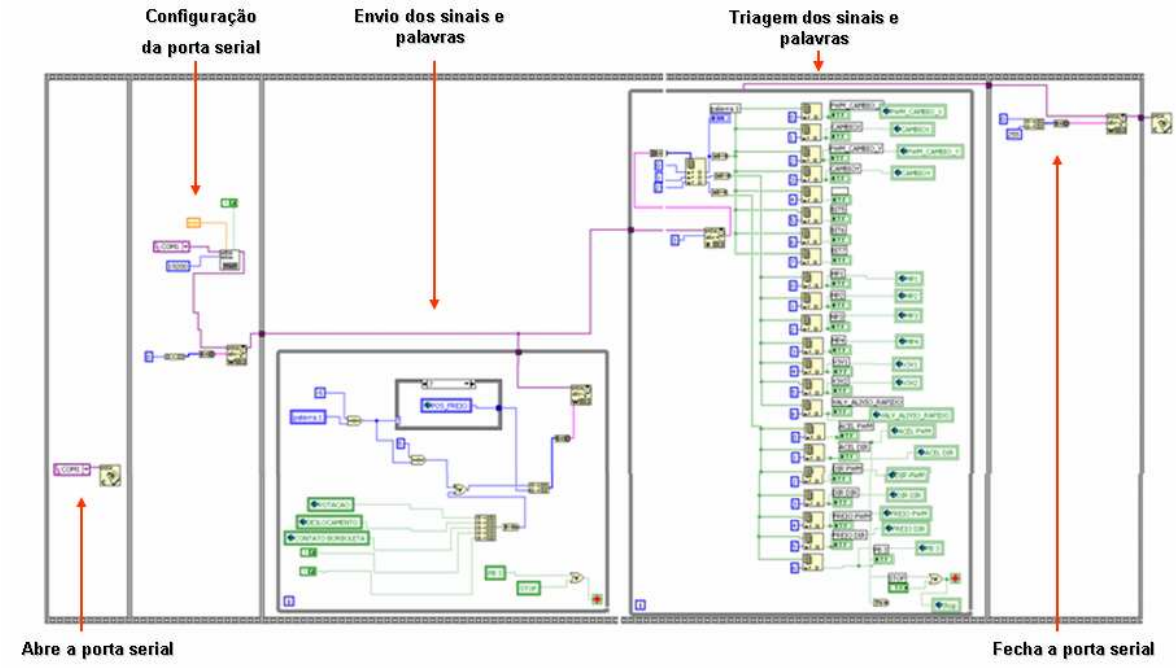


Figura 5.4 – Módulo de comunicação *serial* do Simulador no Labview

No primeiro estágio pode-se configurar qual a porta está sendo utilizada para a realização da comunicação. Nesse caso, está sendo utilizada a porta COM1 que está configurada no primeiro e segundo passo do seqüenciador. Este estágio também abre a porta serial e a janela para a configuração da porta serial (ver figura 5.4).

O segundo estágio do seqüenciador possui dois blocos. Estes blocos realizam o envio dos sinais e palavras e a triagem dos sinais e palavras separando os bits e escrevendo os dados nas variáveis globais relacionadas aos mesmos. A porta de comunicação é fechada no terceiro estágio completando assim o ciclo torna a ser repetido indefinidamente até que o ciclo de comunicação seja finalizado.

As Figuras (5.5 a 5.7) mostram os blocos do Módulo *Serial* de forma mais detalhada, cada um com sua estrutura e funcionalidade.

O bloco de configuração da porta serial desempenha a função de possibilitar ao projetista modificar, a taxa de transmissão dos dados e escolher qual a porta que se deseja utilizar. No caso dessa aplicação, está sendo utilizada a porta COM1, como descrito anteriormente, e a taxa de transmissão dos dados é de 19200 bps (veja a figura 5.5).

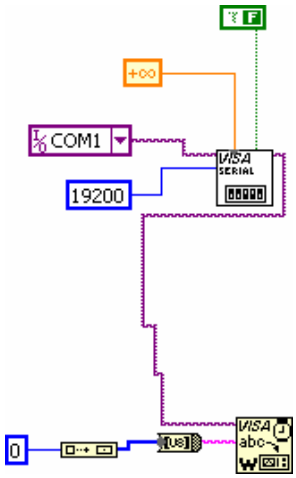


Figura 5.5 – Blocos de configuração da porta serial – VI *Serial*

A figura 5.6, apresenta o laço responsável pelo envio dos sinais das variáveis do estado de movimentação do veículo no ambiente de simulação, como por exemplo: *posição-acelerador, velocidade, posição-freio, posição-câmbio, posição-embreagem e ângulo-direção.*

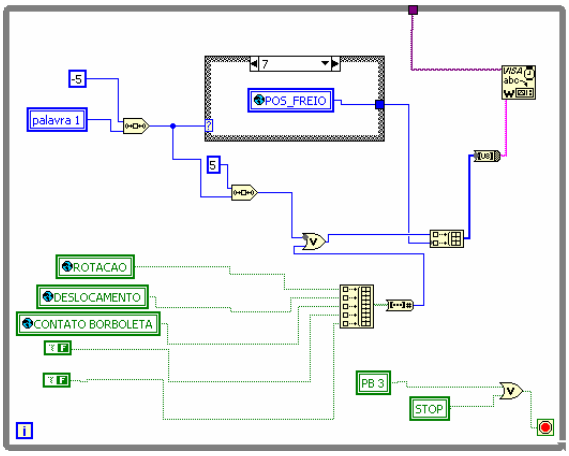


Figura 5.6 – Laço de repetição para o envio dos sinais – VI *Serial*

Os sinais que são enviados pelo controlador de movimentação por meio do protocolo RS-232 serialmente, chegam ao laço de recepção dos sinais (figura 5.7) e nesse são separados em três palavras de oito bits cada. Os oito bits das palavras são acessados separadamente sendo que cada um desses traz os comandos do controlador de movimentação.

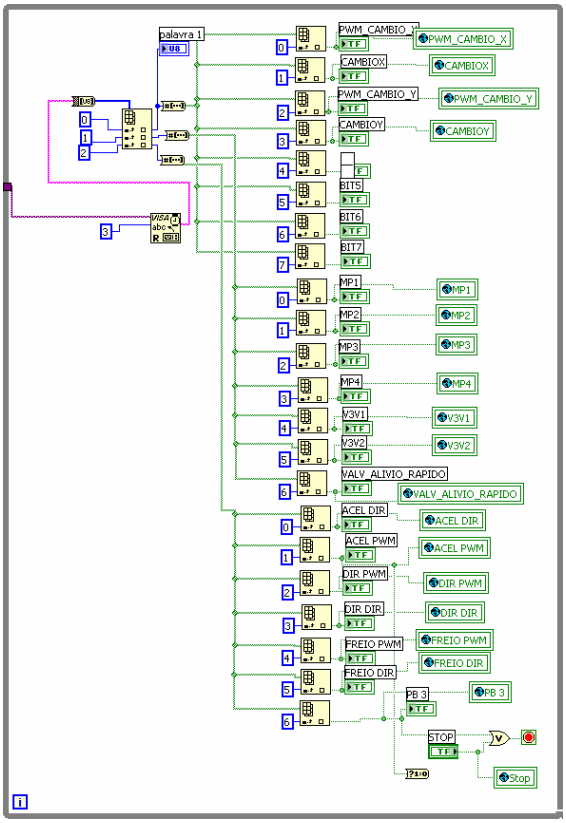


Figura 5.7 – Laço de repetição para recepção dos sinais – VI *Serial*

Os sinais enviados do controlador de movimentação chegam ao laço de recepção dos sinais onde são escritos nas variáveis globais relacionadas.

5.3.2 Módulo *Trata-Sinais*

O módulo *Trata-Sinais* foi projetado com o objetivo de simular a resposta de cada um dos atuadores aos estímulos dos comandos enviados por meio da comunicação serial. Possui os blocos de simulação dos atuadores de cada uma das variáveis presentes no veículo: *acelerador, freio, direção, câmbio* e *embreagem*.

Os motores que atuam essas variáveis são representados por um bloco que responde adequadamente ao sinal PWM e de direção, realizando o incremento ou decremento em uma variável específica. O valor dessa variável é enviado de volta à VI *Serial* para posterior envio ao Microblaze.

Nos blocos de execução, os laços são temporais, ou seja, são executados a cada 1 mili-segundo**.** Assim pode-se determinar parâmetros para velocidade de acionamento dos atuadores no ambiente de simulação. Por exemplo, se desejar-se que uma variável seja incrementada em 50 unidades/segundo, incrementa-se a variável com um valor de 0,05 unidades por ciclo de execução. Lembrando que as variáveis incrementadas são simuladas como se fossem posições dos atuadores e, portanto, devem ser valores inteiros de 8 bits vindos de um potenciômetro (0 a 255).

Na figura 5.8 é mostrado uma o painel de controle com os instrumentos que indicam o estado atual das variáveis relacionadas. O operador tem acesso ao ângulo das rodas, posição do freio, marcha engatada, posição da embreagem e os sinais do PWM do câmbio.

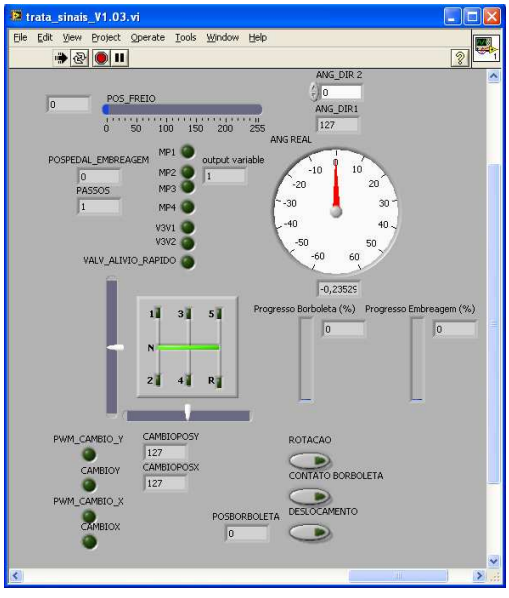


Figura 5.8 – Painel frontal do Módulo de Trata sinais do Simulador no Labview

As figuras 5.9 a 5.14, ilustram os blocos de execução das variáveis no diagrama de blocos da VI Trata Sinais. Cada bloco simula o funcionamento e a resposta de cada um dos atuadores das variáveis de controle de movimentação do veículo no ambiente de simulação.

O bloco *acelerador* (figura 5.9), como o nome sugere, recebe os estímulos do Módulo *Variáveis-Globais*, simula a reação dos atuadores de um veículo real e apresenta ao operador o valor de progresso da borboleta de aceleração. No projeto dos blocos foi projetado uma sub-VI chamada *Motor-DC*, que simula a resposta de incremento e decremento posicional de um motor de corrente contínua que aciona grande parte das variáveis de controle de movimentação do veículo real

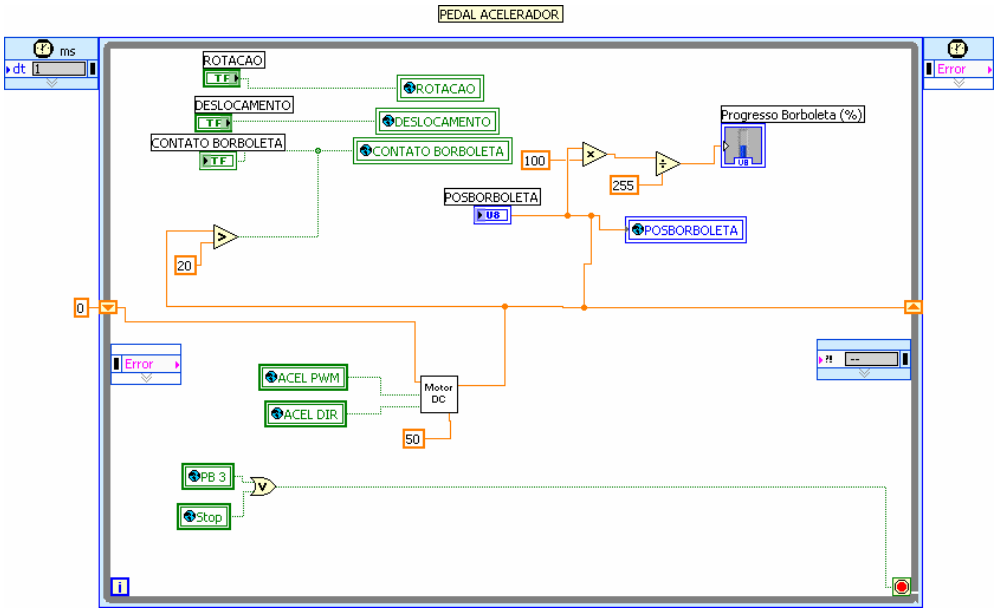


Figura 5.9 – Bloco Aceleração – VI *Trata Sinais*

A figura 5.10 apresenta o laço que forma o bloco de simulação do freio. Esse bloco recebe os sinais *freio_pwm* e *freio_dir* que controlam a velocidade e a direção do acionamento do motor de corrente contínua no simulador, e envia o sinal de posição do freio respectivamente.

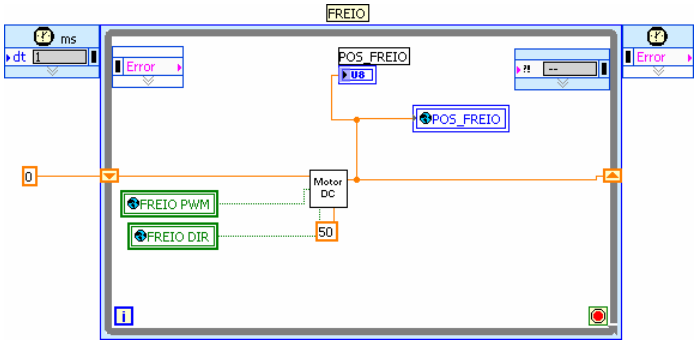


Figura 5.10 – Bloco Freio – VI *Trata Sinais*

De forma similar aos blocos anteriores, o bloco da *Direção* (figura 5.11), recebe os sinais *dir_pwm* e *dir_dir* da VI variáveis globais e vindos do controlador de movimentação, e apresenta ao operador o valor do ângulo das rodas.

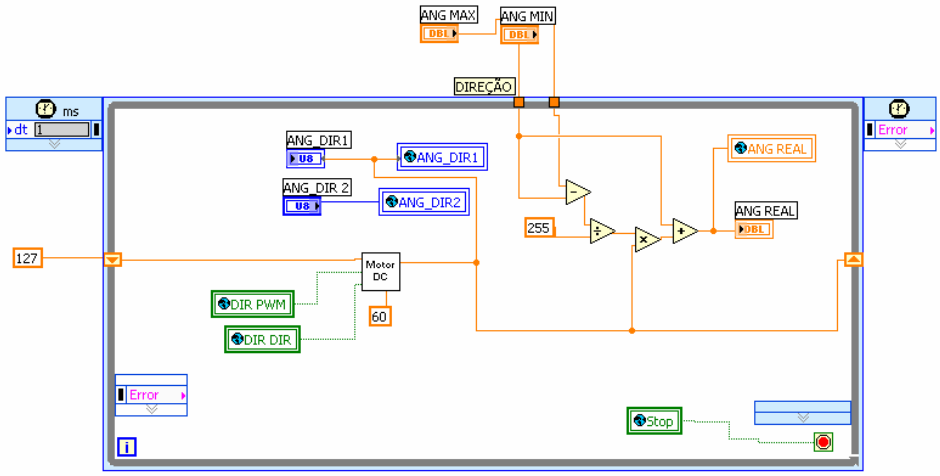


Figura 5.11 – Bloco Direção – VI *Trata Sinais*

O bloco *câmbio* recebe os sinais de pulso PWM e a direção para os dois motores, *acionamento-axial* e *acionamento-radial*, compara com os valores de posição pré-estabelecido para apresentar ao operador qual a marcha está engatada. Esse módulo possui duas sub_VI de simulação dos motores de acionamento chamada de *Motor_DC*, veja a figura 5.12.

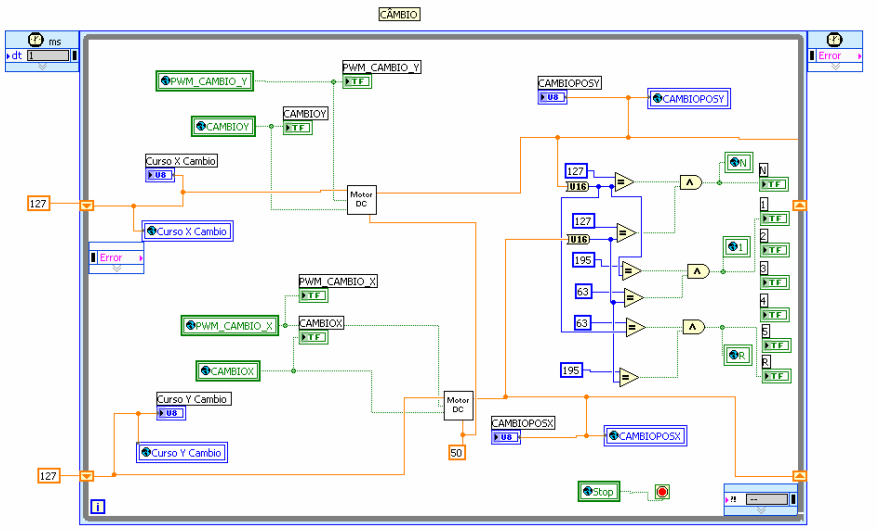


Figura 5.12 – Bloco Câmbio – VI *Trata Sinais*

A figura 5.13 apresenta o bloco de simulação do acionamento da embreagem. Diferentemente dos blocos anteriores, esse bloco não possui um motor de corrente contínua para o acionamento da variável.

O acionamento do sistema da embreagem no carro de teste é realizado por meio de um conjunto de atuadores eletro-pneumáticos com duas válvulas (direcional e escape rápido) e um motor de passo para o acionamento da válvula reguladora de fluxo.

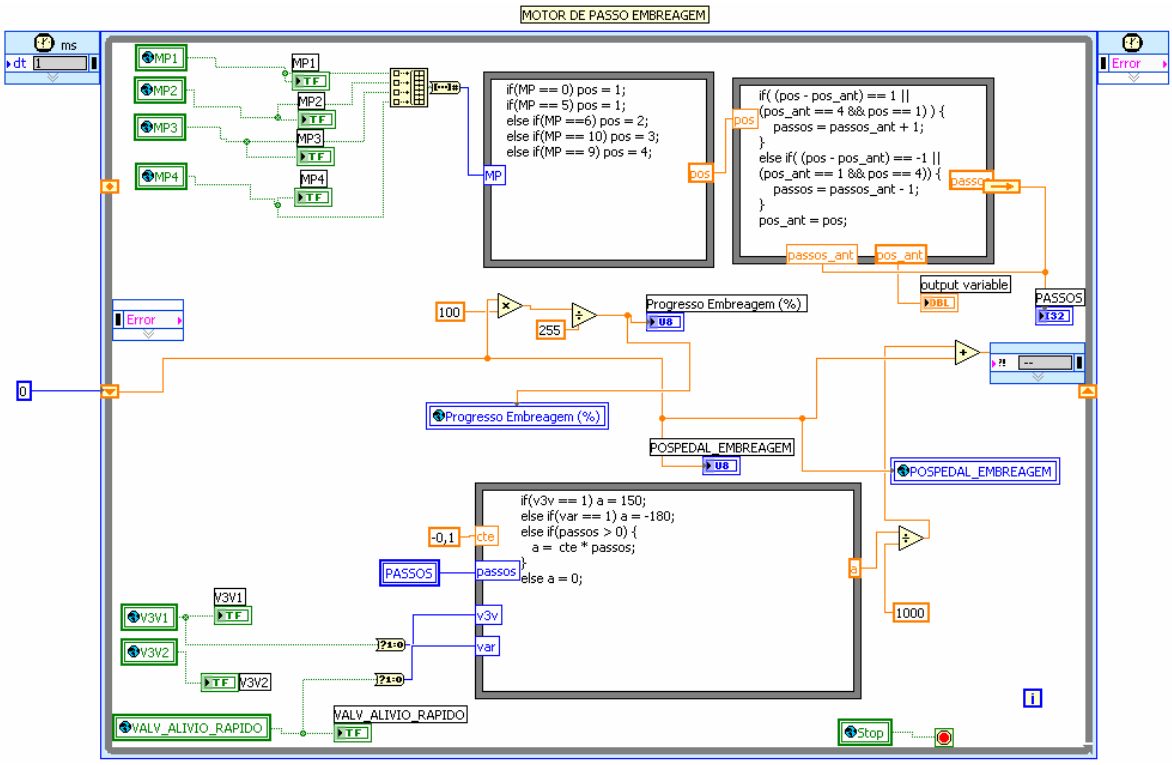


Figura 5.13 – Bloco Motor de passo Embreagem – VI Trata Sinais

Para a simulação da resposta de acionamento do motor de passo e das válvulas, além dos recursos aplicados anteriormente foi utilizado a ferramenta *fórmula-note* que permite a entrada de variáveis e a execução de funções na forma de texto estruturado utilizando-se o padrão ANSI.

O bloco *motor-de-passo-embreagem* recebe o valor das variáveis de acionamento das bobinas do motor de passo, válvula de *escape-rápido* e *válvula direcional* da VI *Variáveis-Globais*, trata os sinais e os envia novamente para o módulo *embreagem***,** onde os sinais de *posição do pedal* e *progresso da embreagem* em resposta aos estímulos do controlador.

Finalmente, o bloco de controle de desaceleração e aceleração (figura 5.14), realiza o processo de equacionamento das variáveis que interferem diretamente na aceleração e na desaceleração do veículo.

As variáveis de *posição-embreagem*, *velocidade-atual-veículo*, *marcha-atual* e *posição-freio* são as variáveis que definem o sentido da aceleração do veículo no ambiente de simulação e sua intensidade.

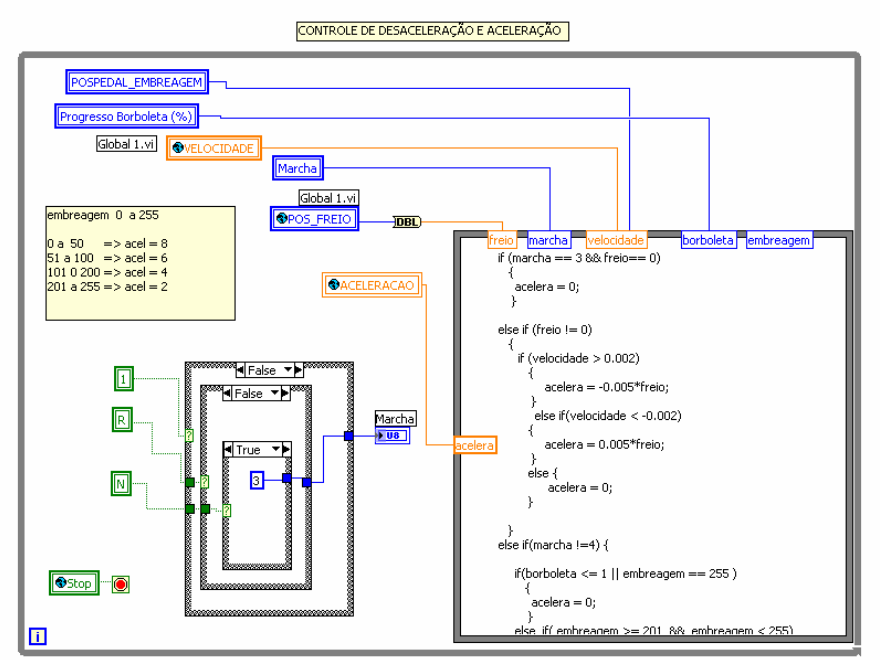


Figura 5.14 – Bloco Motor de Controle de Desaceleração e Aceleração – VI Trata Sinais

Uma vez que as variáveis entram no laço *formula-note*, mostrado na figura 5.14, são executados diversos testes para definir o sentido e direção com que o veículo deverá acelerar ou desacelerar.

5.3.3 Módulo *Trajetória-carro*

Esse módulo realiza a simulação da cinemática do veículo com direção tipo Ackerman (Osório, F. S., 2002) em um sistema de coordenadas (*X* e *Y*). O Módulo VI *Trata Sinais* envia parâmetros de aceleração e ângulo das rodas. Neste caso, o módulo gera os movimentos do veículo baseado nas três equações 5, 6 e 7 descritas em (Baille, G. 1999).

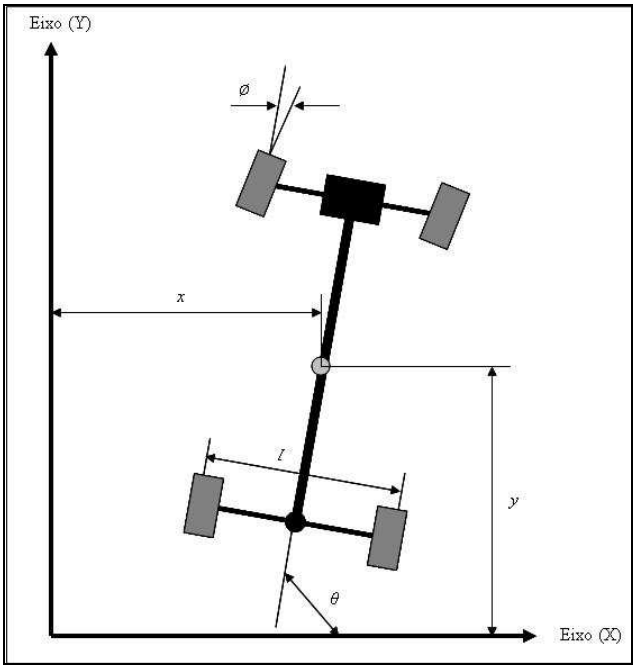
$$\dot{x} = v.\cos(\theta).\cos(\phi) \tag{5}$$

$$\dot{y} = v.\sin(\theta).\cos(\phi) \tag{6}$$

$$\dot{\theta} = v.\frac{\sin\phi}{l} \tag{7}$$

Onde:

- ϕ é o ângulo das rodas;
- θ é o ângulo do veículo com relação ao eixo horizontal *X* no espaço bidimensional;
- *x* é a distância do centro do carro até o eixo-y;
- *y* é a distância do centro do carro até o eixo-x;
- *l* é a distância lateral entre rodas do veículo a ser simulado;
- *v* é a velocidade do veículo (veja figura 5.15).



Figuras 5.15 – Variáveis das equações cinemáticas

As constantes dimensionais definidas não foram baseadas no veículo real.

A figura 5.16 ilustra o painel frontal do módulo VI *Trajetória Carro*

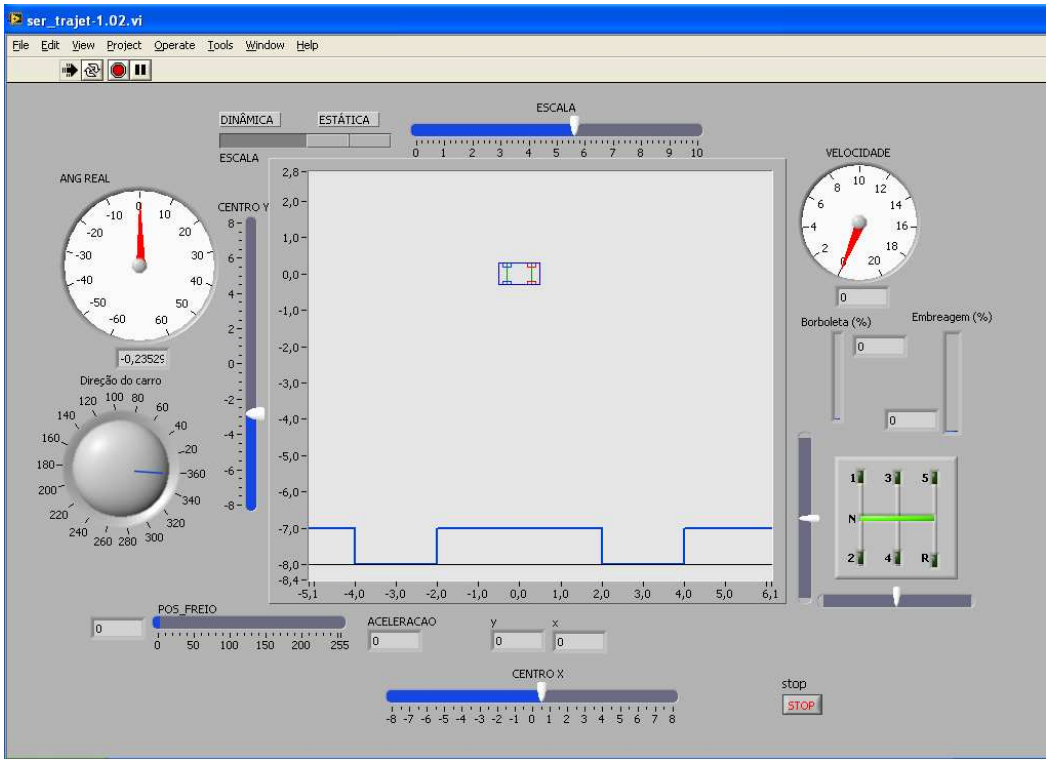


Figura 5.16 – Painel Frontal – VI *Trajetória Carro*

O display gráfico que realiza a simulação de movimentação do veículo no espaço bidimensional é construído por meio do desenho das retas que formam o contorno do veículo. A cada ciclo de execução os segmentos que compõe o veículo são desenhados novamente rotacionados e/ou transladados conforme o seu deslocamento no plano.

A Figura 5.17 ilustra um trecho da estrutura do laço de repetição responsável por realizar as operações de rotação e translação das retas que formam o contorno do veículo, redesenhado o veículo dentro do plano dando a sensação de movimentação do veículo no ambiente de simulação.

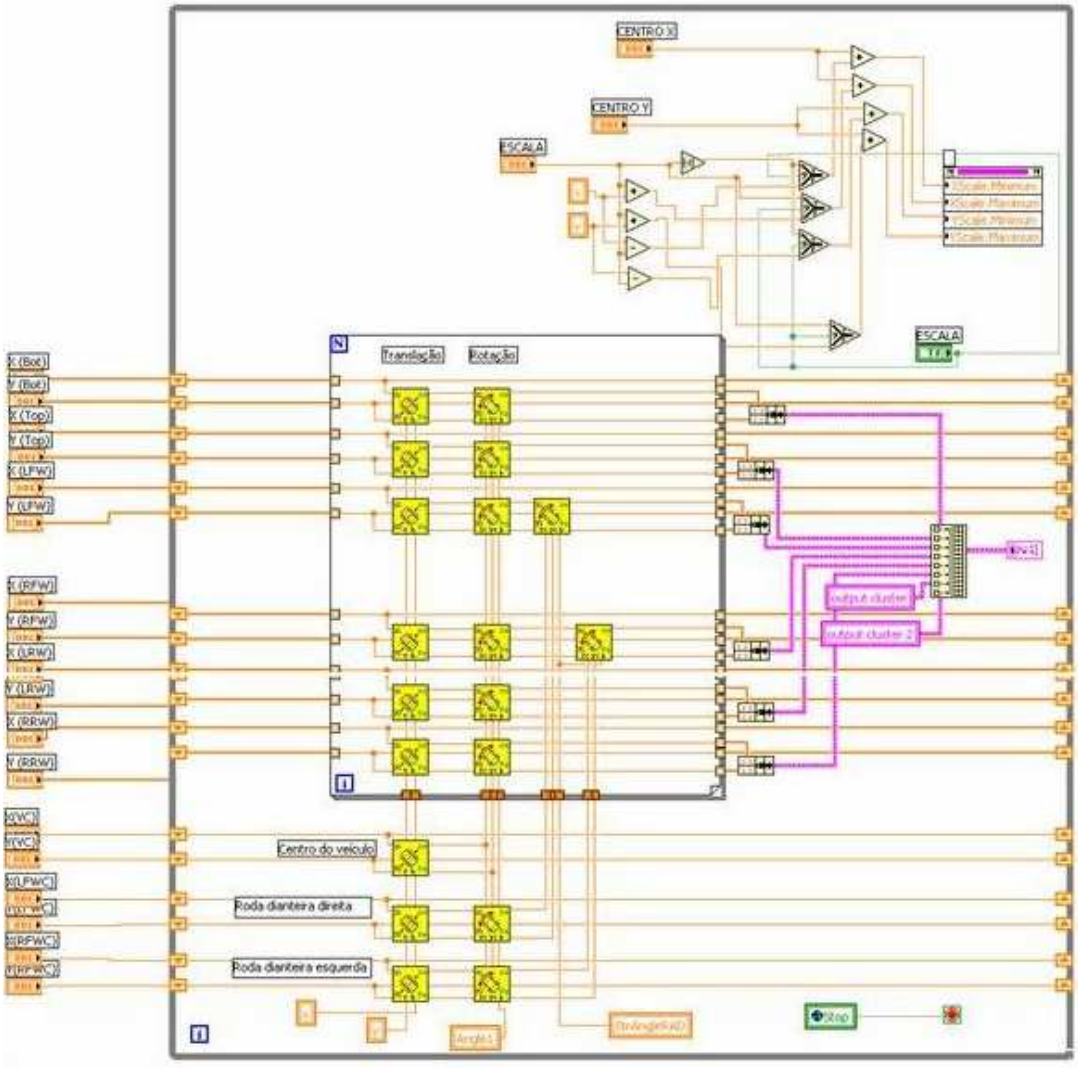


Figura 5.17 – Diagrama de blocos com o laço de repetição da simulação de movimentação do veículo – VI *Trajetória Carro*

Cada um dos ícones em amarelo no centro da figura são sub VI responsável por desenhar uma reta de contorno ou detatalhe que forma o veículo na tela de simulação de movimentação. Cada sub VI executa os cálculos da posição atual e calcula a próxima posição conforme o movimento que se deseja executar. Os dados referentes ao posicionamento e direção de cada uma das retas são enviados a um *cluster* que agrupa todas as informações e as envia a um gráfico tipo *x, y* para seja simulada a movimentação do veículo na tela.

5.4 RESULTADOS DO AMBIENTE DE SIMULAÇÃO

O sistema proposto nesse trabalho para o controle de movimentação de um veículo de passeio em ambiente de simulação foi desenvolvido e testado.

Os resultados foram obtidos por meio da simulação do controle de movimentação do veículo no ambiente de simulação realizando-se o acionamento dos atuadores que acionam o acelerador, freio, direção, câmbio e embreagem, por meio de comandos de um operador enviados ao controlador de movimentação.

A figura 5.18 mostra a leitura do sinal de posicionamento do acelerador no ambiente de simulação. O eixo da amplitude do gráfico é a escala de posição da borboleta variando de o a 255, ou seja um byte. Já o eixo do tempo está em milesegundos. Portanto, o gráfico da figura 5.18 mostra a combinação da amplitude do sinal de posição do acelerador no tempo.

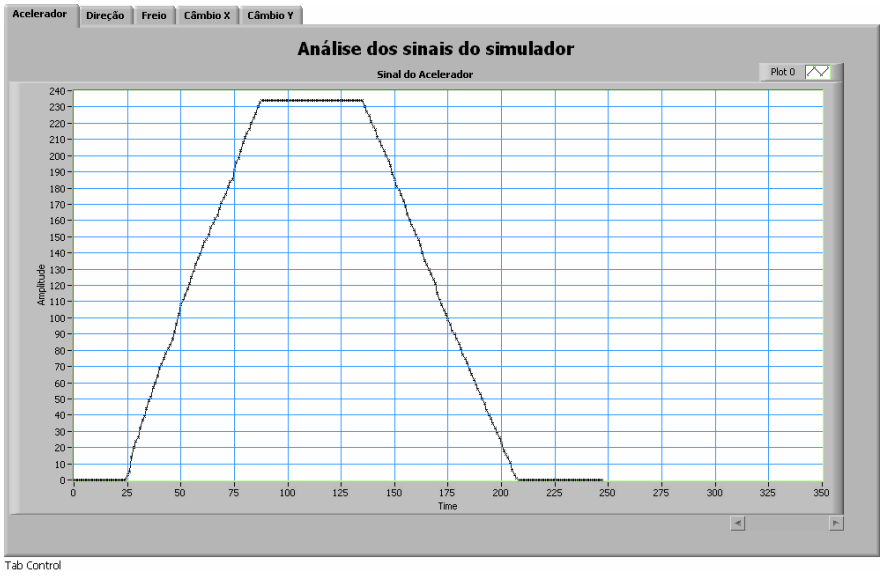


Figura 5.18 – Gráfico da resposta do sinal do Acelerador no Simulador.

Com esse gráfico podemos verificar o acionamento da aceleração de forma linear tanto na crescente quanto na decrescente.

A figura 5.19 por sua vez, mostra a leitura do sinal de posicionamento da direção no ambiente de simulação. O eixo da amplitude do gráfico é a escala de posição ângular das rodas dianteiras do carro no simulador.

A posição das rodas varia de forma angular de 0 a 120 graus, sendo que definimos que quando as rodas estão no fim do curso para a esquerda -60 graus e quando estão posicionadas ao fim do curso para a direita 60 graus.

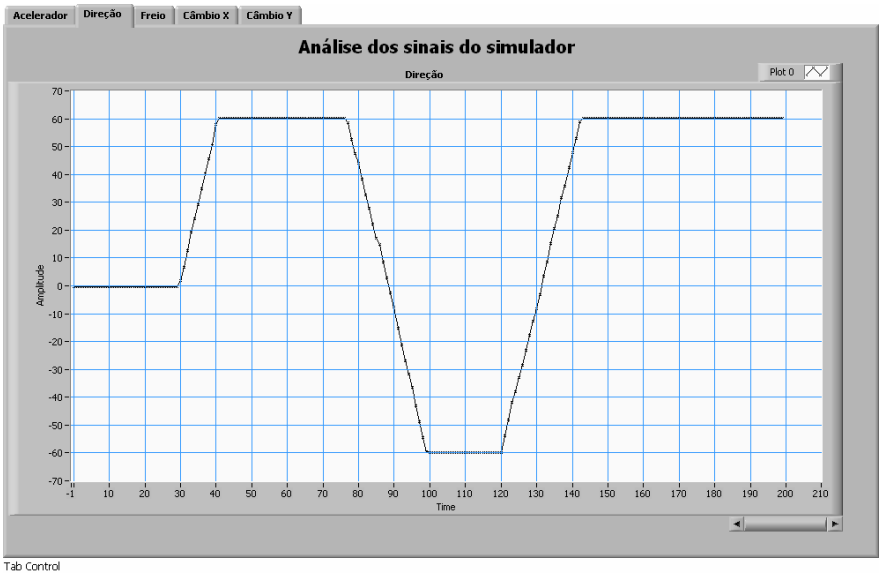


Figura 5.19 – Gráfico da resposta do sinal da Direção no Simulador.

Com esse gráfico podemos verificar o acionamento da direção de forma linear tanto na crescente quanto na decrescente.

No gráfico mostrado na figura 5.20, podemos verificar a posição do freio no tempo. O eixo da amplitude do gráfico é a escala de posição do freio do carro no simulador.

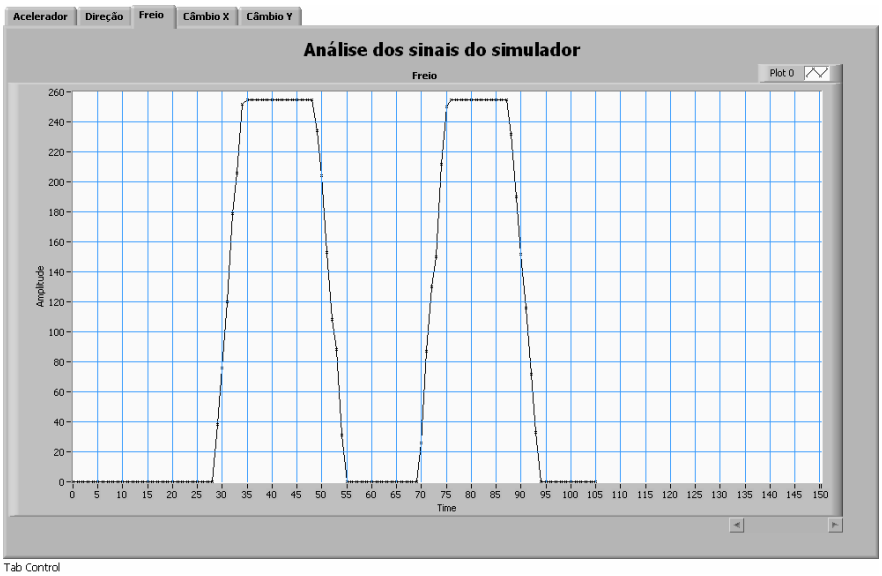


Figura 5.20 – Gráfico da resposta do sinal do Freio no Simulador.

A figura 5.21 mostra a leitura do sinal de posicionamento longitudinal do câmbio no ambiente de simulação. O eixo da amplitude do gráfico é a escala de posição da alavanca de câmbio variando de o a 255, ou seja um byte. Já o eixo do tempo está em milesegundos, portanto o gráfico da figura 5.21 mostra a combinação da amplitude do sinal de posição do câmbio no sentido longitudinal no tempo.

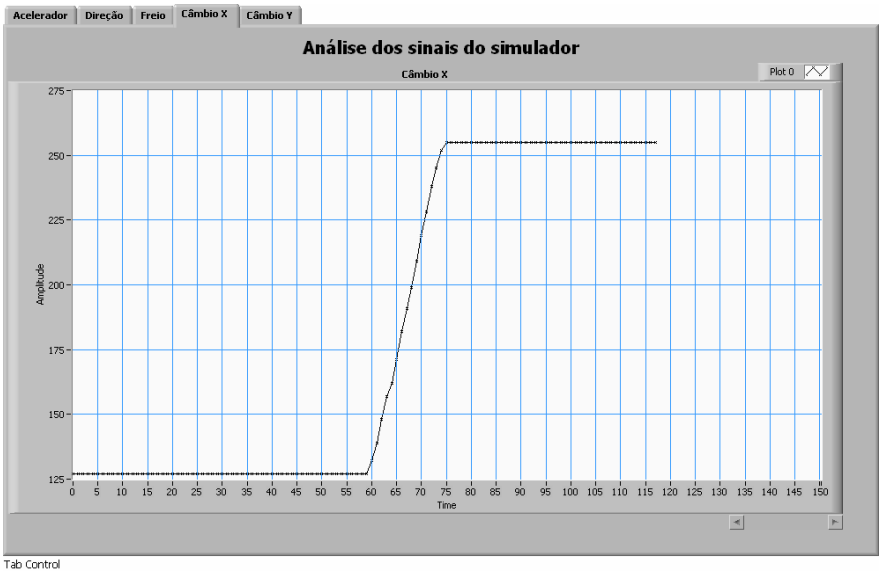


Figura 5.21 – Gráfico da resposta do sinal do Câmbio *X* Simulador.

Na figura 5.22, é mostrada a leitura do sinal de posicionamento transversal do câmbio no ambiente de simulação.

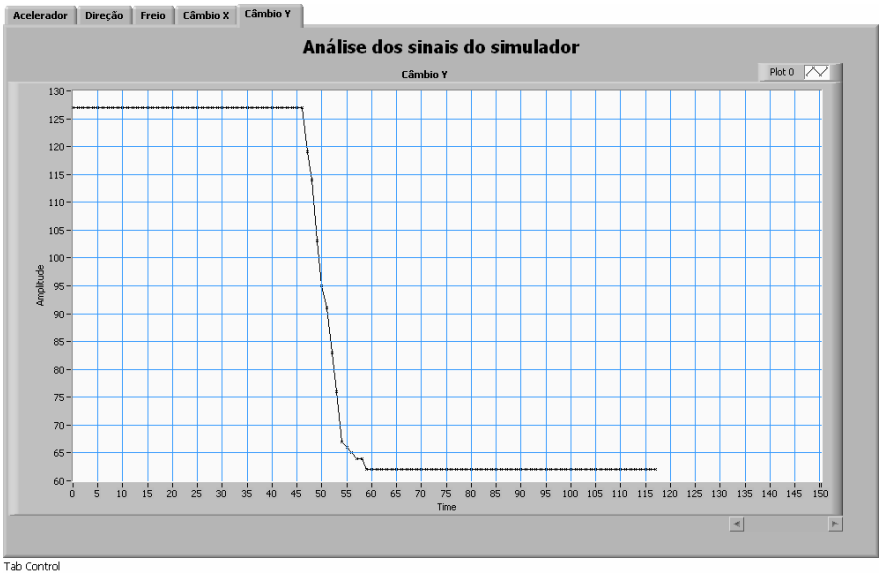


Figura 5.22 – Gráfico da resposta do sinal do Câmbio *Y* Simulador.

Já o eixo do tempo está em milesegundos, portanto o gráfico da figura 5.22 mostra a combinação da amplitude do sinal de posição do câmbio no sentido transversal no tempo.

A figura 5.23 mostra o teste de simulação do controle de movimentação do veículo.

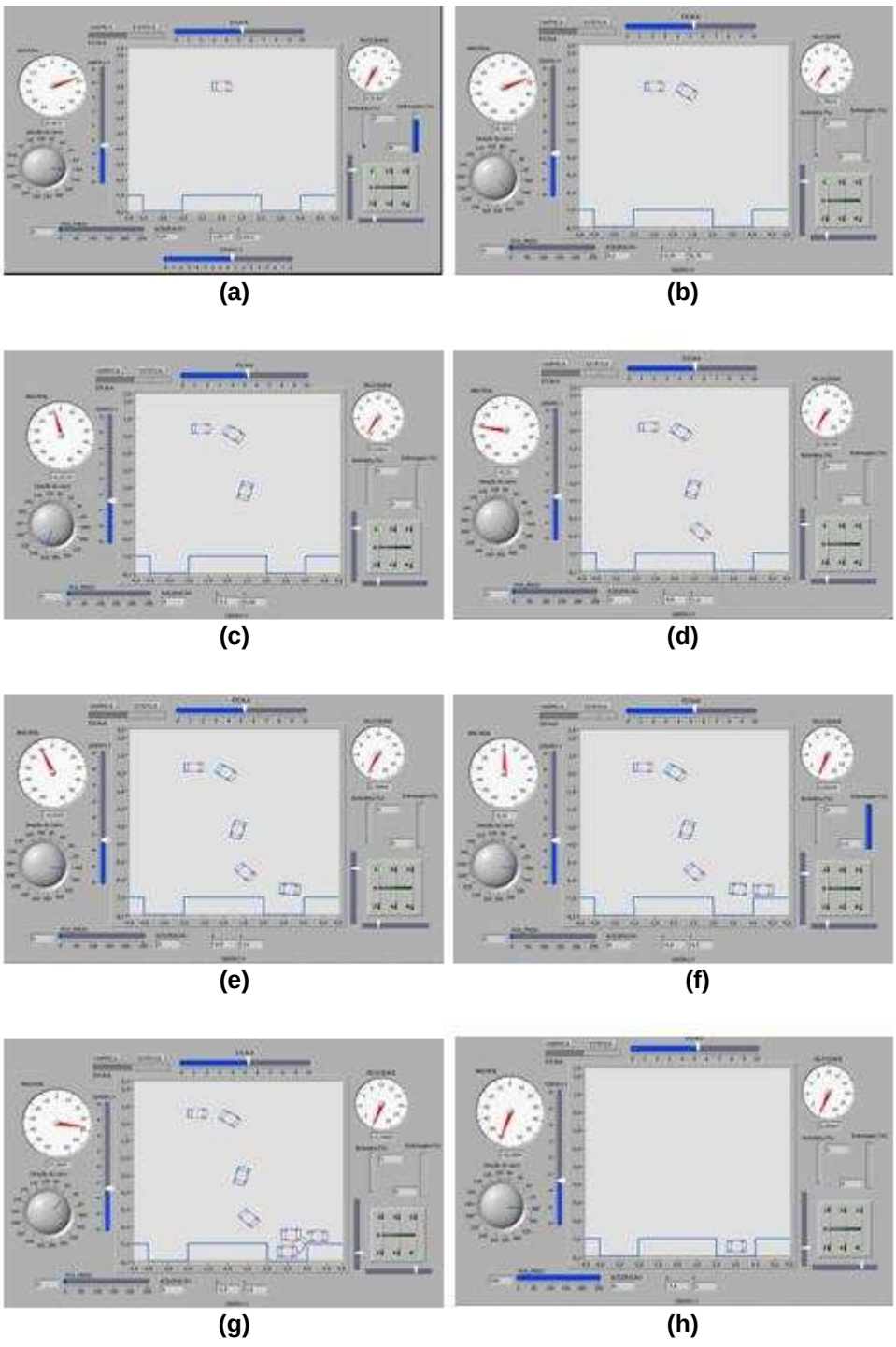


Figura 5.23 – Seqüência de movimentação do veículo no Simulador.

O teste mostrado na seqüência da figura 5.23 de (*a*) à (*h*) mostra o carro no ambiente de simulação saindo de um ponto qualquer, sendo controlado por um operador por meio de comandos enviados via teclado até estacionar numa vaga paralela a sua direita.

5.5 CONCLUSÕES DO CAPÍTULO

Neste capítulo foram abordados assuntos pertinente ao projeto e a construção do ambiente de simulação aplicando-se os conceitos de instrumentação virtual. Na elaboração do ambiente de simulação foram aplicadas equações que descrevem matematicamente o modelo cinemático da trajetória do veículo.

A utilização de uma nova metodologia de projeto empregando-se testes e simulação do controlador da movimentação se mostrou flexível o que permitiu a elaboração de uma plataforma de simulação modular. Em cada um dos módulos foram criados blocos com funções bem estabelecidas dentro das funções de cada um dos módulos.

O formato modular mostrou-se alinhado com a arquitetura projetada no controlador de movimentação embarcado em lógica reconfigurável. O módulo serial é a porta de entrada e saída para a troca dos sinais de forma serial entre o controlador de movimentação e o ambiente de simulação e os distribui ao módulo variáveis globais que realiza a ligação das variáveis entre o módulo trajetória do carro e o módulo trata sinais.

A aplicação dos conceitos da instrumentação virtual dentro da arquitetura modular apresentou vantagens em aspectos da possibilidade da criação de sub-blocos com funções específicas nos blocos como, por exemplo, motor de corrente contínua.

Finalmente, os resultados obtidos do ambiente de simulação foram divididos em duas etapas. Na primeira foram realizados testes de simulação com o acionamento de cada um dos atuadores, responsáveis pelo controle de movimentação do veículo no ambiente de simulação, dos quais foram obtidos gráficos que mostram a posição do atuador no tempo. Já na segunda etapa foi realizado o teste de controle de movimentação propriamente dito.

O teste de simulação com o controlador de movimentação consistiu em controlar o veículo no simulador por meio dos comandos enviados pelo operador via teclado. A simulação objetivou a validação do controlador de movimentação quanto à sincronização dos tempos de envio dos comandos e resposta aos estímulos do operador e das variáveis controladas.

6. DESENVOLVIMENTO DO SISTEMA DE CONTROLE DE MOVIMENTAÇÃO DO VEÍCULO DE TESTES

No capítulo 5 foi descrito o projeto e a implementação do ambiente de simulação e os resultados alcançados. Neste capítulo será detalhado o projeto dos circuitos de acionamento e controle do carro de teste, movido por um motor a explosão e equipado com câmbio totalmente mecânico. Adicionalmente, apresenta-se o desenvolvimento das placas de interface de sinal e de potência responsáveis por receber os sinais vindos dos terminais de extensão da FPGA para acionar os atuadores.

O desenvolvimento do sistema de controle de movimentação do veículo de teste objetiva: (a) dotar um veículo de um sistema de controle e acionamento de atuadores (b) permitir o controle de movimentação por meio de comandos eletrônicos. Com esse sistema pronto e em funcionamento o projeto SiAE contará com uma plataforma flexível de controle da movimentação em funcionamento e confiável para que se possa dar seqüência ao desenvlvimento do sistema de baliza automática.

Finalmente, são apresentados os resultados obtidos no acionamento dos atuadores que controlam as variáveis de movimentação do veículo de testes

6.1. DESENVOLVIMENTO DAS PLACAS DE INTERFACE E POTÊNCIA

Nas etapas anteriores do projeto SiAE foram projetadas e implementadas as placas para acionamento dos módulos. As citadas placas por sua vez possuíam limitações tecnológicas, pois em sua concepção, foram utilizados relés para realizar o acionamento dos motores de corrente contínua. Este problema técnico impossibilitava o controle de velocidade dos motores usando um sinal eletrônico apropriado (por exemplo PWM). Além do anterior, nem todos os módulos possuíam placas apropriadas para seu funcionamento.

Outro ponto que motivou a projetar e implementar novas placas de acionamento e condicionamento foi a necessidade de que o kit de desenvolvimento baseado em FPGA Spartan3 é muito sensível a correntes superiores a 100 mA. Para garantir que não circulariam surtos de corrente pelo kit de desenvolvimento Spartan-3 foi necessário

que isolar os sinais de acionamento dos atuadores e os sinais de realimentação dos transdutores.

Além das placas de acionamento dos atuadores, foi necessário o projeto de uma placa com um conversor A/D – Analógico digital. Além do anterior foi projetada uma placa de interface com a FPGA, utilizando opto-acopladores para o devido isolamento e proteção da placas.

Por esses motivos citados acima, foi realizado um novo projeto das placas para o acionamento dos motores e condicionamento dos sinais. Para projetar os circuitos de acionamento dos atuadores utiliza-se uma metodologia de separar os circuitos em módulos lógicos (de controle) e de potência.

Os módulos foram distribuídos conforme sua função, ou seja, foi projetada uma placa de potência para o acionamento de cada atuador: placa para a direção, placa para o freio, acionamento transversal e uma outra para o acionamento longitudinal do câmbio.

O projeto dos circuitos (lógicos e potência) foi realizado por (Willians R., 2006) utilizando o *software* PROTEL DXP 2004. O sistema está composto de quatro (4) placas de controle de potência de acionamento, uma (1) placa de acionamento da embreagem, uma (1) placa para o conversor analógico digital e uma (1) placa de interface com a FPGA.

6.1.1. Controle de Potência de Acionamento dos Atuadores (Ponte H)

A placa *ponte H* contém um circuito para acionamento de motores de corrente contínua. O circuito contem transistores *mosfet* de rápido chaveamento (ideal para aplicação com PWM) e suporte a correntes elevadas. Os transistores canais P (IRF4205) suportam uma corrente de dreno contínua (25ºC) de 74A e os transistores canais N (IRF3205) suportam uma corrente de dreno contínua (25ºC) de 110A.

A Figura 6.1 ilustra a parte principal do circuito: a ponte H com os transistores. O circuito completo pode ser encontrado no Apêndice A deste documento (vide placa de potência ponte H, neste apêndice).

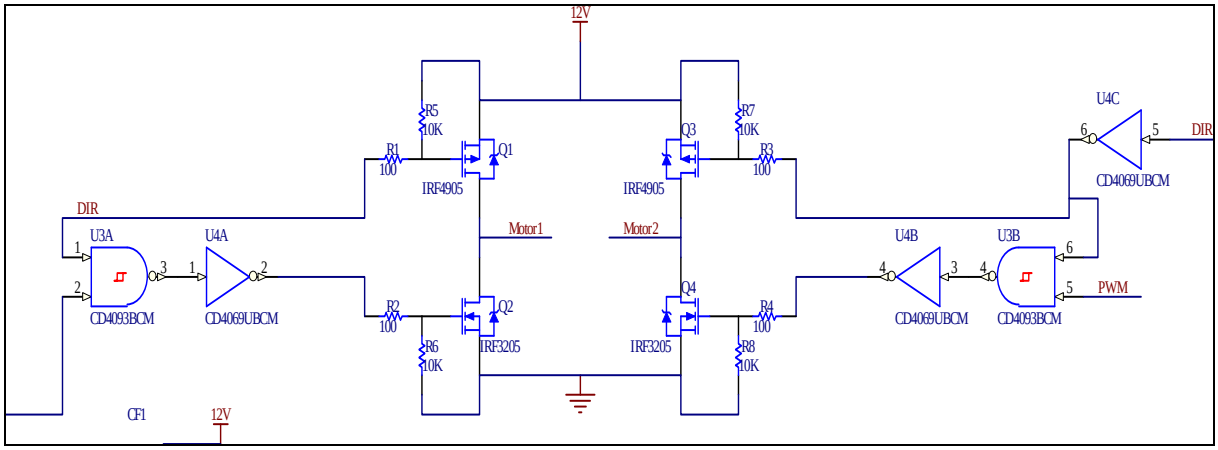


Figura 6.1 – Esquemático do Circuito Ponte H

A figura 6.2 mostra uma imagem em três dimensões da placa com o *circuito de potência Ponte H*. Esta imagem foi obtida após o desenho do circuito e a configuração de suas trilhas.

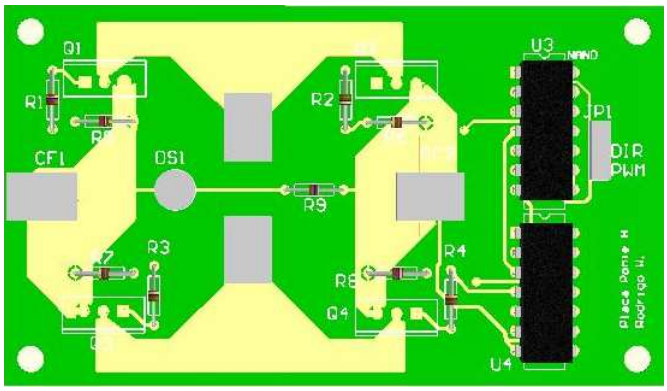


Figura 6.2 – Imagem da placa Ponte H – *Circuito placa Ponte H*

Ao realizarem-se os testes com uma placa experimental o projeto da placa foi enviado para a empresa *Stick* circuitos impressos, (Stick, 2007), para realizar a prototipação da mesma.

Após a fabricação das placas foi realizada a montagem dos componentes no laboratório de soldagem e os testes de continuidade do circuito. Após a montagem e dos testes de continuidade foram realizados os testes em bancada com cada uma das placas conectadas à FPGA.

6.1.2. Controle de Acionamento da Embreagem

A placa da embreagem ativa duas eletro-válvulas do sistema Autonomy® e o motor de passo. Para ativação das eletro-válvulas foram utilizados transistores MOSFET do tipo N (IRFZ48N) que suporta uma corrente de dreno contínua (25ºC). O circuito para ativação de cada válvula é mostrado na figura 6.3.

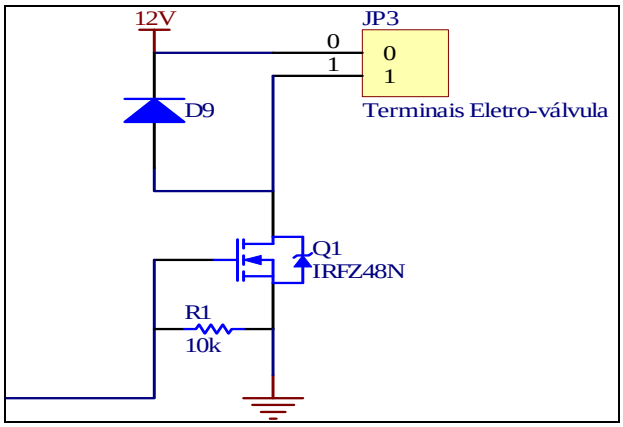


Figura 6.3 – Esquemático do Circuito de Acionamento da eletro-válvula

Para a ativação do motor de passo foi utilizado o *driver* L298N (veja a figura 6.3).

A figura 6.4 mostra uma imagem em três dimensões da placa com o circuito de acionamento da eletro-válvula.

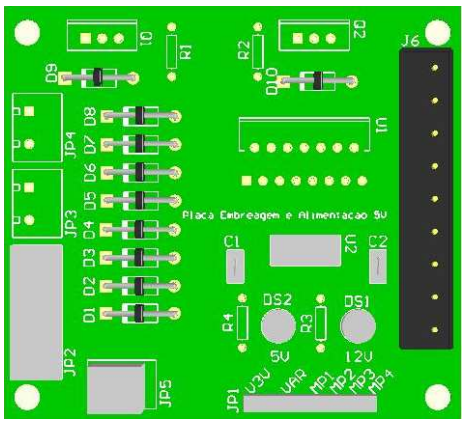


Figura 6.4 – Imagem tridimensional da *placa de acionamento da eletro-válvula*

A figura 6.5 mostra o circuito de potência projetado para o acionamento do motor de passo responsável pelo controle de velocidade de acionamento da embreagem.

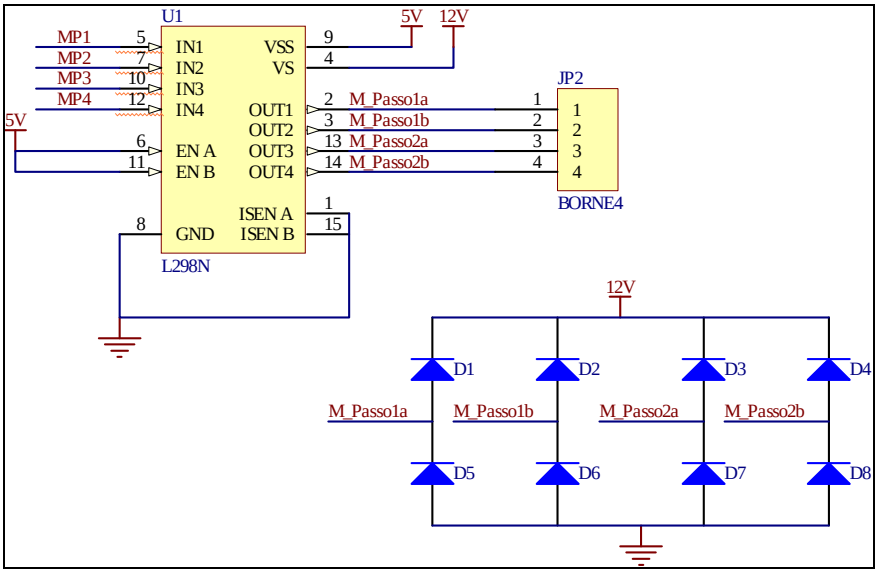


Figura 6.5 – Esquemático do Circuito de acionamento Motor de Passo

A placa de potência da embreagem também realiza a regulação de tensão 5V necessária para esta e as outras placas (com exceção da ponte H). O circuito utiliza o regulador de tensão 7805 (veja a figura 6.6).

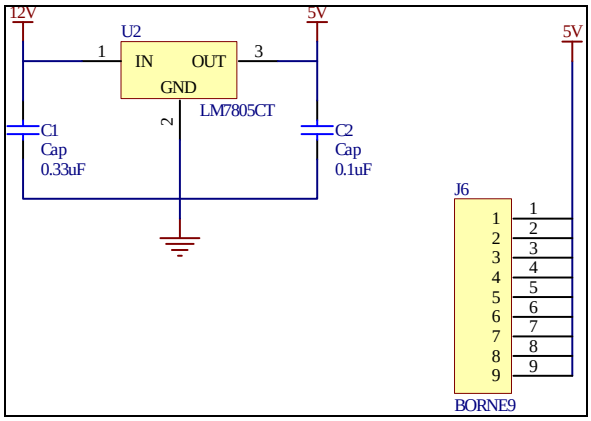


Figura 6.6 – Esquemático do Circuito Regulador de tensão 5V

O circuito completo da *Placa Embreagem* encontra-se disponível no Apêndice A (vide *placa de acionamento da embreagem*, no citado apêndice).

6.1.3. Conversor Analógico Digital (A/D)

A *Placa Conversor Analógico Digital* – A/D realiza a conversão e multiplexação de 8 sinais utilizando um CI ADC0808 de 8 entradas. Os sinais EOC (*End of Conversion*) e START/ALE são usados pelo Microblaze para a correta operação e multiplexação dos 8 sinais analógicos que são convertidos em palavras digitais de 8 bits.

A figura 6.7 mostra o circuito do conversor Analógico Digital implementado.

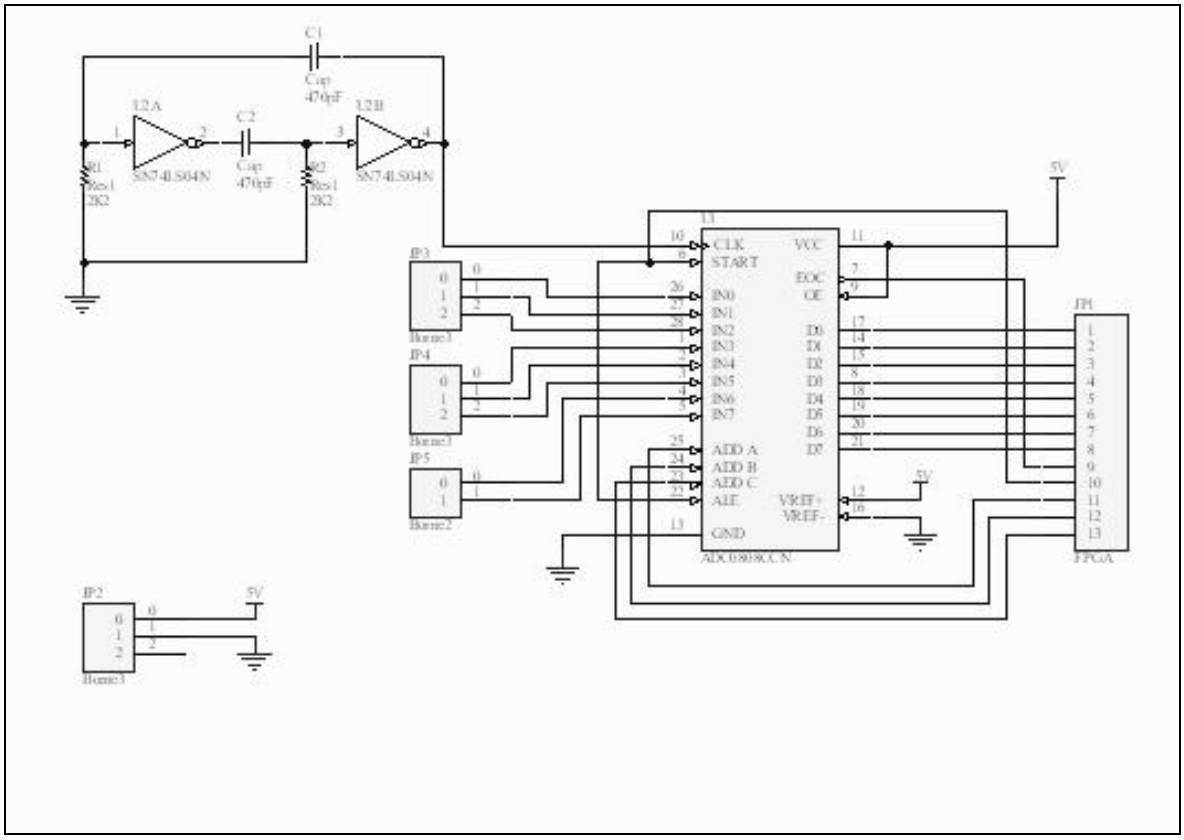


Figura 6.7 – Esquemático do Circuito *Conversor dos Sinais Analógico/Digital* (A/D).

Na figura 6.8 é mostrada uma imagem tridimensional da placa com o circuito do conversor analógico digital.

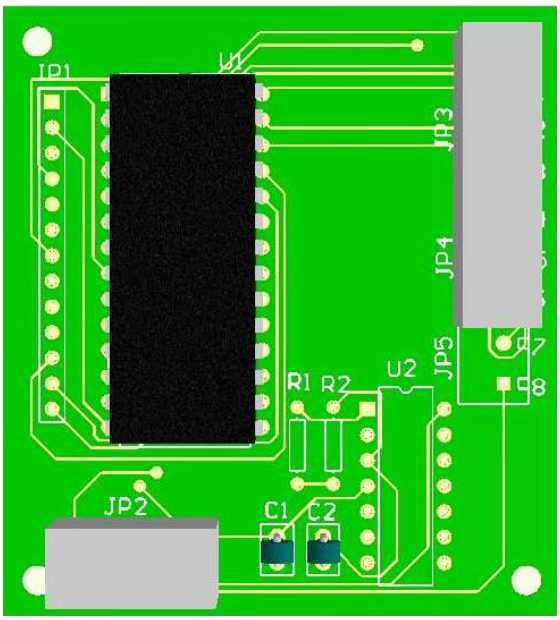


Figura 6.8 – Imagem da placa com o *Circuito do Conversor Analógico Digital* (A/D).

6.1.4. Interface de Sinais e Comando entre a FPGA e o Circuito dos Atuadores

O nível de tensão do sinal nos pinos de saída da FPGA é de 3,3V. Para compatibilidade deste nível de tensão com os necessários (5V e 12V) foram utilizados acopladores ópticos de rápido chaveamento (do tipo 6N137).

A figura 6.9 mostra parte do circuito opto-isolado projetado e implementado para isolar os sinais de acionamento e os sinais digitais dos transdutores utilizados para o controle de movimentação.

Cada sinal de saída da FPGA é ligado a um opto-acoplador da seguinte maneira: para os sinais de entrada, somente um resistor de 1kΩ foi utilizado em série com o sinal, pois a placa é trabalha com uma tensão de 5V. E para os de saída foi utilizado um resistor de 330 Ω.

Os *jumpers* JW1, JW2 e JW3 são utilizados para selecionar o nível de tensão do *pull-up* externo na saída dos opto-acopladores. Cada *jumper* está associado a um grupo de sinais:

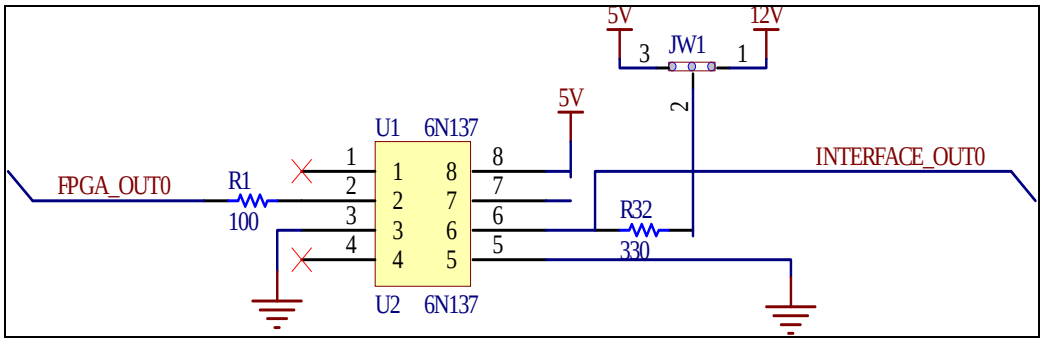


Figura 6.9 – Opto-acoplagem do sinal *fpga_out0* – Placa Interface

- JW1: Sinais de 0 a 12;
- JW2: Sinais de 13 a 19;
- JW3: Sinais de 20 a 30.

A placa interface dá suporte a 31 sinais de saída e 31 sinais de entrada. Foi projetada para utilização com a placa Spartan-3 (Digilent, 2006). A conexão deve ser feita utilizando cabos flats de 40 vias conectadas aos conectores A2 e B1 da placa S3-SB e aos conectores JP1 e JP2 respectivamente da (*Placa Interface FPGA*).

Na figura 6.10 é mostrada uma imagem tridimensional da placa com o circuito da placa de *interface com a FPGA*.

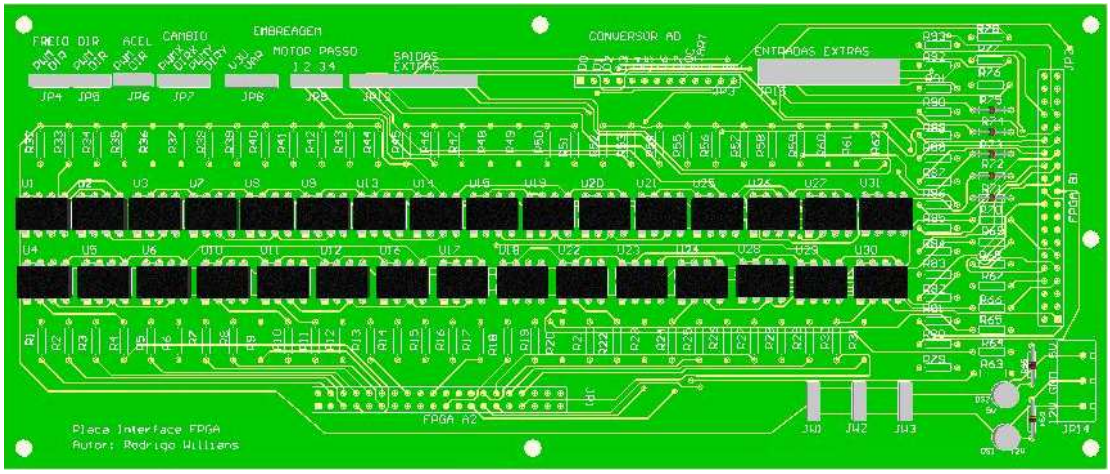


Figura 6.10 – Imagem da placa com o Circuito de interface.

Após a geração dos esquemáticos, *software* DXP 2004 gera automaticamente todo o *footprint* e roteamento da PCB final. Os arquivos gerados pelo DXP foram então enviados para o processo de fabricação para a empresa especializada. Os desenhos das PCB's encontram-se disponíveis no Anexo B deste documento.

6.2. RESULTADOS OBTIDOS COM O CONTROLE DE MOVIMENTAÇÃO NO VEÍCULO DE TESTES

Os resultados obtidos com a implementação do controle de movimentação proposto foram classificados em duas diferentes etapas.

Na primeira etapa foram realizados testes com cada um dos atuadores instalados no veículo. O teste consistiu em acionar os atuadores por meio de comandos enviados via teclado pelo operador ao controlador de movimentação.

Os resultados alcançados com este teste podem ser comparados com os testes realizados no simulador (vide seção 5.4 do capitulo 5).

A figura 6.11 mostra o sinal de posição da borboleta do acelerador do veículo de teste. Por meio desse sinal pode-se controlar a aceleração do veículo.

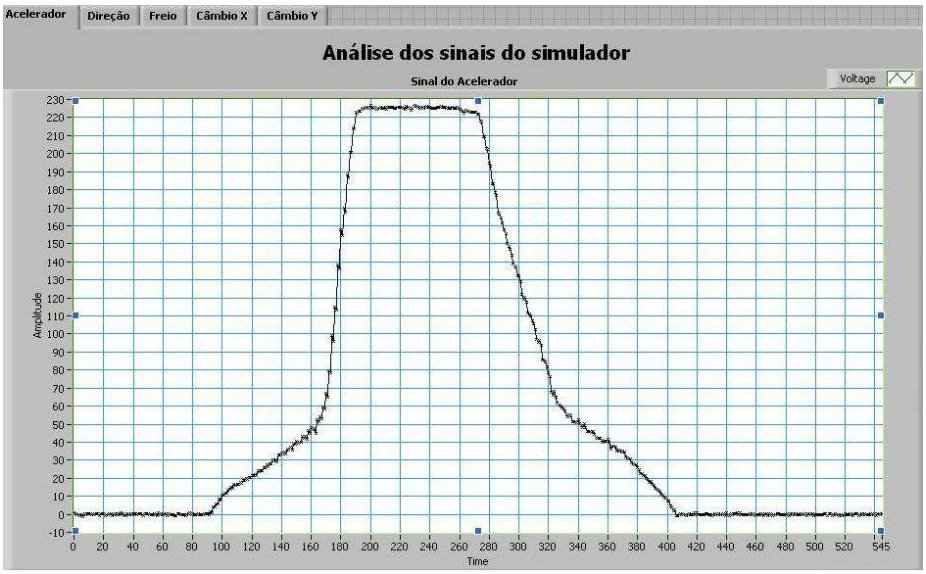


Figura 6.11 – Sinal de posição da borboleta do acelerador – Captura da posição do atuador no Carro de testes real.

O gráfico na figura 6.11, mostra a curva de aceleração máxima, o que é indicado pelo valor posicional 225, e a curva de desaceleração até zero.

A figura 6.12 por sua vez, mostra o sinal de posição angular da direção do veículo. O gráfico (ver figura 6.12) mostra a curva de deslocamento angular das rodas até o fim de curso mecânico da barra de direção. Pode-se visualizar a posição saindo de 0 graus e chegando até 53 graus para a direita (escala positiva) e chegando até a – 60 graus (escala negativa, fim de curso para a esquerda.

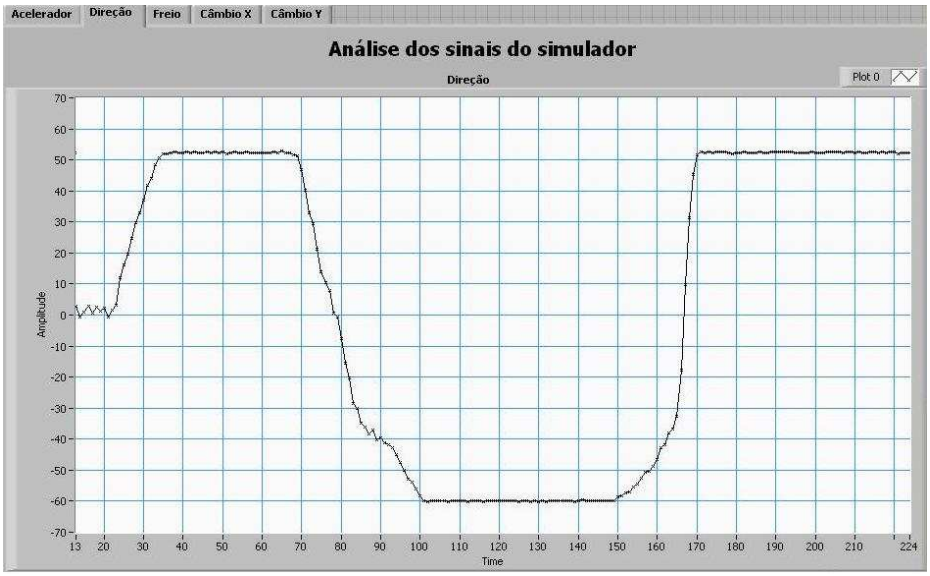


Figura 6.12 – Sinal de posição da Direção – Captura da posição do atuador no Carro de testes real.

Na figura 6.13 é mostrado o sinal relativo à posição do freio, sendo que esta varia de 0 a 255. Quando a leitura do transdutor indica (0) significa que o freio está desativado, e quando indica 255 o freio está totalmente acionado.

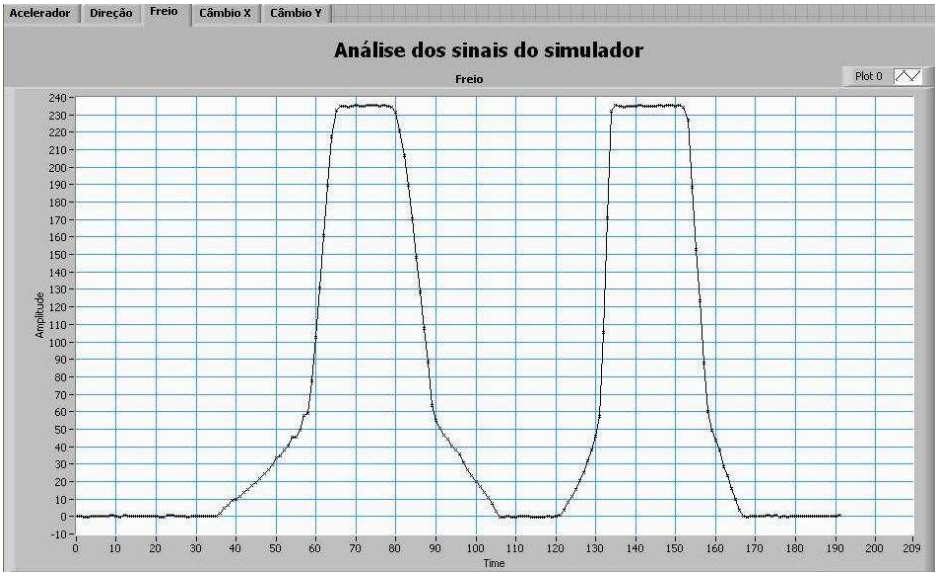


Figura 6.13 – Sinal de posição do Freio – Captura da posição do atuador no Carro de testes real.

Na figura 6.14 é mostrado o sinal relativo à posição do câmbio no eixo *X* (ver figura 4.13 no capítulo 4) sendo que esta varia de 0 a 255. A combinação da posição do câmbio no eixo *X* e no eixo *Y* indica que uma marcha está ou não engatada.

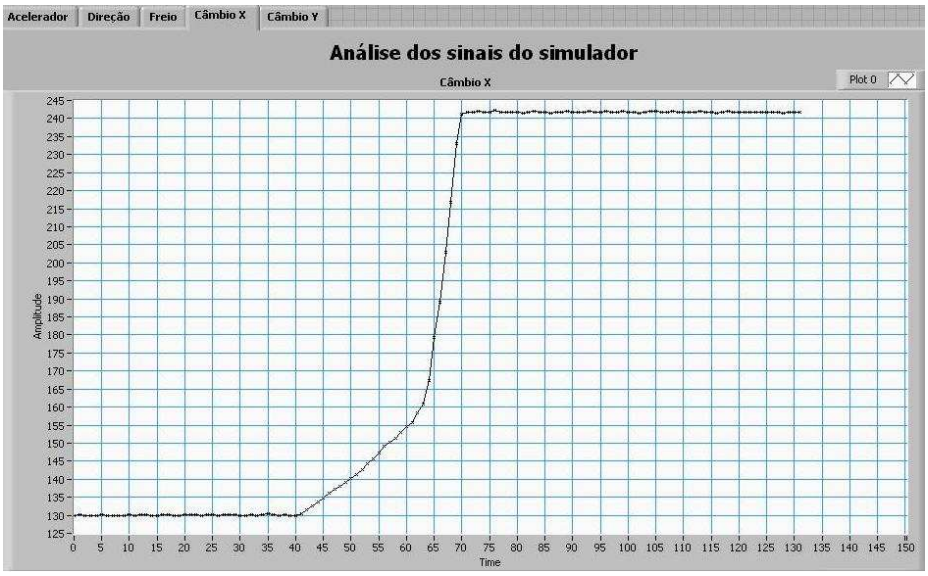


Figura 6.14 – Sinal de posição do Câmbio em X – Captura da posição do atuador no Carro de testes real.

Na figura 6.15 é mostrado o sinal relativo à posição do câmbio no eixo *X*, sendo que esta varia de 0 a 255.

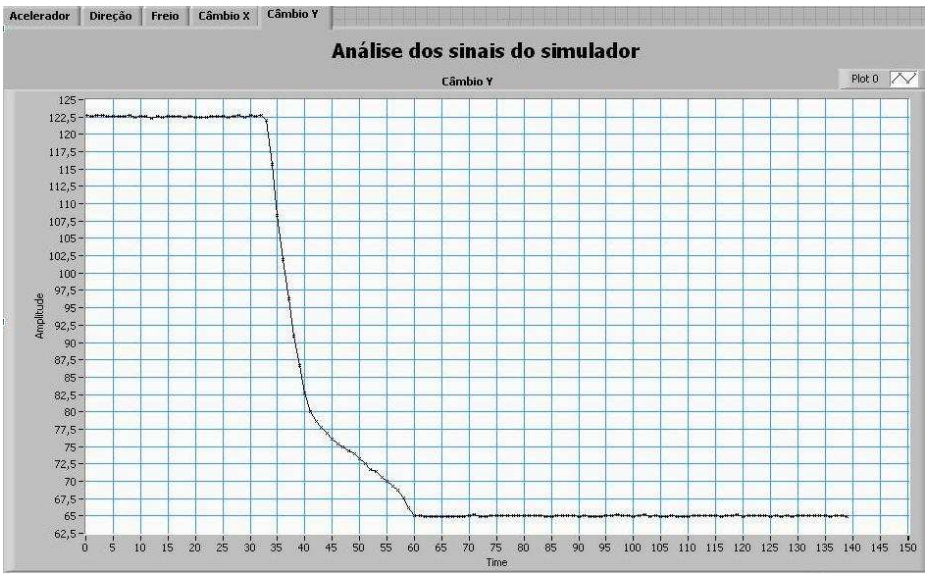


Figura 6.15 – Sinal de posição do Câmbio em Y – Captura da posição do atuador no Carro de testes real.

Já na segunda etapa foram realizados os testes de movimentação do veículo. O teste de movimentação do veículo foi realizado num estacionamento. Este teste consistiu em controlar a movimentação do veículo por meio de comandos enviados ao controlador por meio do teclado.

A figura 6.16 mostra uma seqüência de imagens de um vídeo onde se registrou o teste de movimentação do veículo. A realização do teste teve por finalidade a validação do controle de movimentação do veículo por meio de comandos enviados por meio de comandos ao controlador de movimentação.

O teste consistiu em engatar a primeira marcha, soltar o freio e a embreagem e controlar o veículo em movimento numa área de 800 m² num estacionamento vazio (vide as seqüências, figura 6.16(a) até (q).

Figura 6.16 – Seqüência de cenas do vídeo do teste de movimentação do veículo de teste.

6.3. CONCLUSÕES DO CAPÍTULO

Neste capítulo foram abordados os principais pontos relativo ao projeto do sistema eletrônico de controle de movimentação do veículo real.

O projeto e desenvolvimento das placas de controle e acionamento dos atuadores e os transdutores é de grande importância ao trabalho como um todo, pois desempenha um papel muito importante no que tange o acionamento eletro-eletrônico.

Os resultados obtidos também foram distribuídos em duas etapas. Na primeira etapa foram executados testes com os atuadores no veículo real. Os gráficos obtidos mostram os atuadores sendo acionados descrevendo a posição no tempo. A segunda etapa por sua vez mostra uma seqüência de *frames* da filmagem do teste de movimentação realizado com o veículo real.

7. CONCLUSÕES E TRABALHOS FUTUROS

7.1. CONSIDERAÇÕES GERAIS

Este trabalho propõe a implementação de uma arquitetura de controle de movimentação de um veículo que atenda as necessidas de segurança, conforto e baixo custo, dentro dos parâmetros estabelecidos no projeto SiAE para o estacionamentm automático.

O sistema de controle embarcado em lógica reconfigurável proposto explora o uso dos conceitos de arquitetura modular para o controle dos atuadores de acionamento da movimentação do veículo de passeio comum com câmbio totalmente mecânico.

Neste trabalho verificou-se a flexibilidade da implementação do sistema de controle de movimentação (com controle implementado em FPGA), assim como também, foi desenvolvido um ambiente de simulação (baseado em uma ferramenta de instrumentação virtual) para a execução de testes e validação do sistema antes da realização dos testes com o veículo real.

7.2. O CONTROLADOR DE MOVIMENTAÇÃO EM HARDWARE RECONFIGURÁVEL

O projeto e implementação do controlador de movimentação em hardware reconfigurável trouxe ao projeto SiAE novo cenário tecnológico, pois abriu um novo caminho para a elaboração de sistemas envolvendo o controle da navegação seguindo uma trajetória. Adicionalmente, a tecnologia envolvida pode ser adaptada para técnicas conhecidas em robótica (implementação/validação) como, por exemplo, a navegação evitando obstáculos. O projeto em si é um passo prévio para a elaboração de um sistema de estacionamento automatizado (baliza automática).

A extensão da arquitetura modular (proposta nesse trabalho) foi potencializada com a flexibilidade da implementação do controle de movimentação usando hardware reconfigurável.

No que tange os resultados obtidos com a implementação do sistema de controle de movimentação embarcado em lógica reconfigurável motra-se promissor em dois principais aspectos: tempo de projeto e flexibilidade em hardware/software.

Foram utilizados 85% dos recursos lógicos do kit de desenvolvimento empregado. Não houveram perdas na freqüência do *clock* de cada um dos módulos e do sistema como um todo possibilitando ao mesmo funcionar até na freqüência máxima oferecida pelo kit Spartan-3.

As vantagens da arquitetura proposta para o sistema de controle de movimentação aplicando-se sistema reconfigurável com relação aos sistemas convencionais são:

- a utilização de periféricos configurados em hardware por meio da linguagem VHDL;
- flexibilidade em software e hardware;
- ciclo de projeto testes e validação mais rápidos e concisos .

7.3. O SISTEMA DE SIMULAÇÃO

O ambiente de simulação trouxe ao projeto uma nova metodologia de ciclo de projeto aplicando-se testes de simulação para a validaçào do controle de movimentação.

Os resultados obtidos e apresentados na forma de gráficos da resposta do sinal das variáveis de controle da movimentação mostram a atuação ao longo do tempo (veja seção 5.4 do capítulo 5). Também foram apresentados resultados no âmbito do controle de movimentação no ambiente de simulação por meio de uma seqüência de fotos do veículo movendo-se na janela do simulador bem como a indicação de todas as variáveis controladas.

A arquitetura modular do sistema de simulação mostrou-se alinhado com a arquitetura projetada e implementada no controlador. A utilização dos conceitos da instrumentação virtual contribuiram para a concepção da arquitetura modular apresentando recursos como a criação de sub-VI’s com funções específicas em cada um dos módulos e blocos do sistema (vide seção 5.3 do capítulo 5).

O controle de movimentação no ambiente de simulação mostrou-se fundamental para a execução dos testes e validação do controlador de movimentação antes dos testes com o veículo real.

7.4. INTEGRAÇÃO DO SISTEMA DE CONTROLE DE MOVIMENTAÇÃO NO VEÍCULO DE TESTES

Tabela 12 – Principais Sistemas de Automação Veícular Implementados comparados ao sistema proposto neste trabalho

Características	Good et al, 1988 (*)	Steven E. S. Charles A. (1991)	Han-Shue, et all, 1999	Wada M, 2003	Shimazaki et al., 2004 (*)	Tanaka et al, 2006 (*)	Correia A., et al, 2007
Controle da Direção	Controle hidraulico	sim	Pontos magnéticos instalados na rodovia	não	Não	Controle eletrônico	Controle eletrônico
Controle da Embreagem	não	sim	não	não	Não	Não	Controle eletrônico
Controle do Freio	não	sim	sim	não	Não	Controle eletrônico	Controle eletrônico
Controle do Acelerador	não	sim	Controla a velocidade pela leitura dos pontos magnéticos da rodovia	não	Não	Controle eletrônico	Controle eletrônico
IHM	não	não	Display de dados	Display para ângulo da direção; Freio; Acelerador.	Sim	Display mostra a imagem das câmeras e o estado do veículo	Display e leds mostram o estado do Sistema
Arquitetura Modular	não	sim	não	sim	Sim	Sim	Sim
Motorização	Motor a combustão interna	Motor a combustão interna	Motor a combustão interna	Adaptável a qualquer tipo	Motor a combustão interna	Motor de combustão interna	Motor de combustão interna
Câmbio automático	não	não	Não informado	Adaptável a qualquer tipo	Adaptável a qualquer tipo	Sim	Não
Conceitos *drive-by-wire*	sim	não	não	sim	Não	Sim	Não
Controlador	Microcontrolador	Computador embarcado no veículo	Computador embarcado no porta-malas do veículo	Computador embarcado no porta-malas do veículo	Computador embarcado no veículo	Computador ECU	Processador embarcado em FPGA
Sensoriamento	Ultrason	ângulo das rodas; giro das rodas	Magnetômetros	ângulo das rodas; giro das rodas; GPS	Ângulo das rodas, giro das rodas	Presença de obstáculos, captura de imagens, ângulo da direção	Variáveis de controle de movimento
Unidade de controle	não	Não	não	não	Não	Sim	Sim
Flexibilidade	Somente do software	Somente software	Somente do software	Somente do software	Somente de software	Somente de software	Hardware e software
Tipo de Sistema	Passivo	Ativo	Ativo	Passivo	Passivo	Passivo	Passivo

(*) ***Patentes registradas no banco de patentes dos Estados Unidos da América.***

O desenvolvimento do sistema de controle de movimentação do veículo de testes objetivou a implementação de um sistema capaz de acionar os atuadores que controlam a movimentação do veículo de testes.

O projeto e implementação do sistema eletro-eletrônico foi baseado na distribuição proposta no sistema projetado nas fases anteriores (Bellardi, T., 2005) e (Garrido R., 2001), mas foi reprojetado com a utilização de transistores para o controle de velocidade dos atuadores.

A tabela 12 apresenta uma comparação com os principais características de outros trabalhos com este trabalho. Destacam-se contribuições quanto a flexibilidade do sistema proposto e a utilização de processamento embarcado em lógica reconfigurável.

Os resultados obtidos com a implementação do controle de movimentação foram obtidos em duas etapas.

Na primeira fase foram testados todos os acionamentos das variáveis de controle de movimentação e a obtenção dos gráficos com os sinais da posição no tempo de cada um dos atuadores mostrados nas figuras 6.10 até a figura 6.14.

Comparando com os gráficos obtidos com o acionamento dos atuadores no simulador (veja as figuras 5.17 até 5.21) podemos ver que o sistema de controle respondeu satisfatoriamente no ambiente de simulação e no controle do veículo real.

Já na segunda fase foram realizados testes de controle de movimentação com o veículo de testes. A figura 6.15 mostra o teste realizado que consistiu em controlar o veículo por meio de comandos via teclado enviados ao controlador embarcado em lógica reconfigurável.

Os resultados obtidos mostraram-se satisfatórios dentro dos objetivos traçados inicialmente, relativos ao projeto de um sistema capaz de receber comandos e realizar o controle dos atuadores responsáveis pela movimentação de um veículo de passeio movido por um motor a combustão interna.

7.5. SUGESTÕES PARA TRABALHOS FUTUROS

As perspectivas de trabalhos futuros relacionados objetivam atingir um nível funcional do sistema para atender as espectativas do projeto SiAE bem como o controle de trajetória e de navegação.

Ao adicionar os transdutores de posicionamento do veículo e localização dos obstáculos, os módulos para a geração e controle de trajetória o sistema estará completo para executar a tarefa de estacionamento automático.

7.5.1. Controle de trajetória

Para a realização do estacionamento automático o projeto de um módulo que gere e controle a trajetória é necessário. O módulo de geração de trajetória recebe os sinais relativo as distâncias entre o veículo e a vaga de estacionamento e realiza os cálculos para gerar a trajetória que levará o veículo dentro da vaga com segurança. O controle da trajetória se dá com a avaliação em tempo real da posição do veículo e as veriáveis de controle frente a trajetória pré-estabelecida.

A arquitetura do sistema proposto possui a flexibilidade necessária para acoplar o módulo de geração e controle da trajetória sendo necessário a utilização de um kit com maior número de unidades lógicas.

7.5.2. Controle de navegação

O Controle de navegação é uma proposta para dotar o veículo de inteligência capaz de alcançar um objetivo posicional de forma a desviar de obstáculos dinâmicos ou estáticos dentro de regras de otimização de caminho.

REFERÊNCIAS BÍBLIOGRÁFICAS

Amato João Neto (2004) – Relatório setorial FINEP – Semicondutores – http://www.finep.gov.br/PortalDPP/relatorio_setorial - acessado em 2004

Baille, Gérard at al. (1999).: Le CyCab de 'I'NRIA Rhône-Alpes. INRIA. Relatório técnico no. 0229.

Baltes, J., and Lin, Y. Lin (1999) .: Path- Tracking Control of a Non-Holonomic Car-like Robot with Reinforcement Learning. CITR, Tamaki Campus, University of Auckland, Private Bag 92019 pg 1–17

Bellardi Thiago C., (2005) .: Definição de uma arquitetura flexível para controle de movimentos de um veículo de passeio. Publicação DM – 002 Dissertação de mestrado apresentada no departamento de mecânica da Universidade de Brasília, novembro.

Bellows, P.; Hutchings, B. "JHDL - An HDL for Reconfigurable Systems", Proceedings of the IEEE Symposium on Field-Programmable Custom Computing Machines, Abril 1998.

Becker, J. Simon, A., Söhnitz, I., Göllinger, H., Schumacher, W., (1998) "A Decentralized Path Planning and Control Structure for an Autonomous Vehicle", In *Proceeding of the IV '98. IEEE International Conference on Intelligent Vehicles*. Stuttgart, Alemanha - Outubro 1998.

BMW série 7 – www.automotor.xl.pt/0505/900.shtm - Acessado em março de 2005

Buyya, R. (1999a) .:"High Performance Cluster Computing: Architectures And Systems", Prentice Hall Nj, USA.

Buyya, R. (1999a) .: "High Performance Cluster Computing: Programming And Applications" , Prentice Hall Nj, USA.

Brown S.; Vranesic Z. (2000) .: Fundamentals of Digital Logic with VHDL Design. Mc Graw Hill, Toronto.

Bushby S. (1997).: BACnet: a Standard Comunication Infrastrutures for intelligent Buildings. In: Elsevier, Automation in Construction 6, pg. 529-540.

CNN (a) - Four vehicles finish in $2 million robot race. Acessado em setembro 2005. http://www.cnn.com/2005/TECH/science/10/08/robot.race.ap/.

CNN (b) - Don't like parking? Try Pivô, - Acessado em setembro 2005. http://www.cnn.com/2005/AUTOS/09/30/nissan_concept.reut/

Compton, K.; Hauck, S. (2002) .:"Reconfigurable Computing: A Survey of Systems and *Software*", ACM Computing Survey, Vol. 34, No. 2, pg. 171-210.

Consoni Flavia (2004) – Relatório setorial FINEP – Automóveis – acessado em 2004 http://www.finep.gov.br/PortalDPP/relatorio_setorial

Correia A.; Llanos C. H. Q.; Carvalho R. W.; Alfaro S. A. (a) (2007): A Design/Testing Platform Based on Reconfigurable Architectures and Virtual Instrumentation Applied to the Hands-free Driving Automobile Problem, WSEAS transactions on systems and control. Issue 3, vol. 2, Março 2007.

Correia A.; Llanos C. H. Q.; Carvalho R. W.; Alfaro S. A. (b) (2007): A Platform Based on Reconfigurable Architectures and Virtual Instrumentation Applied to the Driving Automobile Problem. 6th WSEAS International Conference on SIGNAL PROCESSING, ROBOTICS and AUTOMATION (ISPRA '07). Ilha de Corfu, Grécia, Fevereiro, 2007. pg. 242 a 248.

Digilent - http://www.digilentinc.com/ - Acessado em dezembro de 2006.

Dehon A. (2000) .: "The Density Advantage of Configurable Computing", IEEE Computer. Vol. 33, No. 4.

Dudek, G. and Jenkin, M. (2000). *Computational Principles of Mobile Robotics*. Cambridge University Press, Cambridge, UK.

EDK MicroBlaze tutorial - Xilinx 2005.

Garrido R. (2001).: Automação e Controle Aplicados em Veículo de Combustão interna Convencional em Deslocamento Linear. Dissertação de mestrado UnB.

Giove D. (2004).: Reconfigurable Hardware Resource in Accelerator Control Systems. EPAC, Lucerne Switzerland.

Goetting, H.H. (2001) .: FOX GmbH: Automatisierte Fahrzeuge. http://www.foxit.de

Good Warren T.; Vilhauer Jr., Jacob E (1988).: Automatic parallel parking system- Patent No 4735274 – United States

Gu D., Hu. H. (2002).: Neural Predictive Control for a Car-like Mobile Robot. International Journal of Robotics and Autonomous Systems, Vol. 39, No. 2-3, pg 1-15

Hamblen J. O.; Furman M. D. (2001).: Rapid Prototyping of Digital System. Ed. Kluwer Academic Publishers.

Han-Shue; Tan Jürgen Guldner; Satyajit Patwardhan; Chieh Chen an Bénédicte Bougler (1999) .: Development of on Automated Steering Vehicle Based on Roadway.

Harteinsten R. (2002).: Enabling Technologies for Reconfigurable Computing and Software/Configure Co-Design. CNRS International Workshop ENST, Paris.

Hutton M.; Yuan R.; Schleicher J.; Chua K.; Phoon H. (2006) .: A methodology for FPGA to Structured_ASIC Synthesis and Verification. Proceeding of the Conference of Design, Automation and test in Europe, pg. 64-69.

Hwang, K.; Xu, Z. (1998) .:“Scalable Parallel Computing: Technology, Architecture, Programming”, McGraw-Hill.

I.N.R.I.A institute – Acessado em 2007 http://www.inria.fr/index.en.html

Kelber Christian R., Cláudio R. Jung, Farlei Heinen (2005) .: Computação Embarcada; Projeto e Implementação de Veículos Autônomos Inteligentes. XXV Congresso da SBC, Janeiro.

Kelber, C.R.; Osório, F.S.; Jung, C.R.; Heinen, F.J.; Dreger, R.S.; Gules, R.; Mello Jr., C.D.; Silveira, M.A.; Schumacher, W.; (2003a) "Tecnologias para Automação Veicular - Soluções em Mecatrônica e Sistemas de Apoio ao Motorista"; *Engenharia- Estudos Tecnológicos*; ISSN 1517-3615, Vol. XXIII, No. 24, p.37-47.

Kelber, C.R.; Dreger, R.S.; Gomes, G.K.; Webber, D.; Schirmbeck, J.; Netto, R.H.;

Borges, D.A.; (2003b) "Cell-Phone Guided Vehicle, an Application based on a Drive-by-Wire Automated System"; In *Proceeding of the 2003 IEEE Intelligent Vehicles Symposium*; Ohio, USA, 9-11 June.

Lewis, T. G.; El-Rewini, H. (1998) .: “Distributed and Parallel Computing”, Manning.

Li J. H, Lee, Li, P. M. (2005) A Neural Network Adaptive Controller Design for Free-Pitch-Angle Diving Behavior of an Autonomous Underwater Vehicle. Robotics and Autonomous Systems. Elsevier, 52 pg 132 - 147

Magnets – A Case Study of Mechatronic System Design. IEEE/ASME Transaction on Vehicle Technology, Vol., 4, No 3, Fevereiro.

Makimoto T. (2002) .: The Hot Decade of Field Programmable Technologies. IEEE Proceedings of the International Congress on Field Programmable Technologies.

Mingjie Lin.; Gamal A.; Chang Y.; Wong S. (2006) .: Performance Benefits of Monolithically Stacked 3D-FPGA. FPGA06 - USA.

Moore G. E. (1997) .: The Microprocessador: Engine of the Technology revolution. Communications of the AMC, Vol. 40 (2), pg. 112-114.

M. Herz et al. (2002) .: Memory Organization for Data-Stream-based Reconfigurable Computing; Proc. IEEE ICECS 2002, Dubrovnik, Croatia.

National I. - http://zone.ni.com/devzone/cda/tut/p/id/5053 - acessado em 21/02/2007.

Osório, F. S.; Heinen, F.; Fortes, L. (2002). Controle da Tarefa de Estacionamento de um Veículo Autônomo através do Aprendizado de um Autômato Finito usando uma Rede Neural J-CC. In: *VII Simpósio Brasileiro de Redes Neurais, 2002*, SBC – Porto de Galinhas - Recife.

Osório, F. S.; Musse, S. R.; Santos, C. T.; Heinen, F.; Braun, A. e Silva, A.T. S. (2004). Ambientes Virtuais Interativos e Inteligentes: Fundamentos, Implementação e Aplicações Práticas. *XXIV Congresso da SBC – JAI 2004* (Jornadas de Atualização em Informática). Tutorial. Salvador, Bahia. Web: http://inf.unisinos.br/~osorio/palestras/jai04-avii.html (acessado: maio de 2005)

Patterson, David A.; Hennessy, John L. (2000).:Organização e projeto de computadores: A interface Hardware/Software.2 Ed. Rio de Janeiro:LTC, pg. 551.

Petko, M., Uhl, T. (2001) .: Embedded controller design-mechatronic approach. IEEE, Second Workshop on Robot Motion and Control. 195-200

Pfister, G. (1998).:“In Search Of Clusters”, Prentice Hall.

Regazzi, Rogério Dias; Pereira, Paulo Sérgio e Silva Jr; Manoel Feliciano (2005) .: Soluções práticas de instrumentação – Utilizando a programação gráfica Labview.

Robosoft S.A. Automated People Transportation – Applications, Technologies and Perspectives. Sep. 2003 - Acessado Disponível em http://www.robosoft.fr/Brochures/wp-transports_EN.pdf

Sanchez, E. et al. (1999) .:"Static and Dynamic Configurable Systems", IEEE Transactions of Computers, pp. 556-564.

Self-parking car hits the shops. BBC News, 01 setembro 2003 - Disponível em http://www.news.bbc.co.uk/2/hi/technology/3198619.stm.

Shaladove S. E. e Desoer C. A., 1991

Shimazaki Kazunori; Kimura Tomio; Yamada Satoshi (2004).: Parking assisting device – Patent No US 6711473 B2 - United States

Simon, A.; Becker, J., (1999) "Vehicle Guidance for an Autonomous Vehicle", In *Proceeding of the IEEE International Conference on Intelligent Transportation Systems*. Tokyo, Japan; 5-8 October.

Simon, A., Söhnitz, I., Becker, J., Schumacher, W.; (2000) "Navigation and Control of an Autonomous Vehicle", In *Proceeding of the 9th IFAC Symposium on Control in Transportation Systems*. Braunschweig; 13-15 June.

Sipper, M.; Sanchez, E. (2000) .:"Configurable Chips Meld *Software* and *Hardware*, IEEE Computer, pp. 120-121.

Söhnitz, I., Schwarze, K., (1999) "Lateral Control of an Autonomous Vehicle: Design and First Practical Results", In *Proceeding of the IEEE International Conference on Intelligent Transportation Systems*. Tokyo, Japan; 5-8 October.

Söhnitz, I. (2001); "Querregelung eines autonomen Strassenfahrzeugs"; *Forschritt-Berichte* VDI, Reihe 8, Nr. 882.

Steven E. Shaladover; Charles A. Desoer (1991) .: Automatic Vehicle Control Developments in the PATH Program. IEEE Transaction on Vehicle Technology, Vol., 40, No 1, Fevereiro.

Stick - http://www.stick.ind.br/ - acessado em 21/02/2007.

Tanaka Yuu; Iwata Yoshifumi; Satonaka Hisashi; Endo Tomohiko; Kubota Yuichi; Matsui Akira; Iwakiri Hideyuki; Sugiyama Toru; Kawakami Seiji; Iwazaki Katsuhiko; Kataoka Hiroaki (2006) .: Vehicle backward movement assist device and vehicle parking assist device - Patent No 7039504 – United States

TOYOTA MOTOR CORPORATION.: The PM concept vehicle – Acessado em http://www.toyota.com/vehicles/future/pm.html

Taghavi T.; Ghiasi S.; Ranjan A.; Raje S.; Sarrafzadeh M. (2004) .: Innovative or Perish: FPGA Physical Design. ACM, ISDP04, USA.

Tan, H.S., Guldner, J., Patwardhan, S., Chen, C., Bougler, B. (1999) .: Development of an Automated Steering Vehicle Based on Roadway Magnets—A Case Study of Mechatronic System Design. IEEE/ASME Transactions on Mechatronics, Vol. 4, No. 3. 258 – 271.

Turley, J. (1998).:"Triscend E5 Reconfigures Microcontrollers", Microprocessor Report, Nov. 16, pg. 12-13.

Tzuu-Hseng, S., Chang, S-J., Chen, Y-X (2003).: Implementation of Autonomous Fuzzy Garage-Parking Control by an FPGA-Based Car-Like Mobile Robot Using Infrared Sensors. International Conference on Robotics & Automation, Taipei, Taiwan, pg 3776 – 3781

Villasenor, J.; Mangione-Smith, W. H. (1997) .:"Configurable Computing", Scientific American, pg. 54-59.

Yang E., Gu, D., Mita, T., Hu, H (2004) .: Nonlinear Tracking Control of A Car-Like-mobile Robot via Dynamic Feedback Linearization. Control 2004, University of Bath, UK, Setembro.

Yabarrena, Jean Mimar Santa Cruz (2006) .: Tecnologias system on chip e CAN em sistemas de controle distribuído. Dissertação de mestrado – USP.

Wada M., Kang Sup Yoo e Hideki Hashimoto (2003) .: Development of Advanced Parking Assistance System. IEEE Transaction on Industrial eletronics, Vol., 50, No 1, Feb. 2003.

Willians R. (2006) .: Projeto de um Sistema Embarcado em Arquitetura Reconfigurável e instrumentação Virtual Aplicados a Veículos Autônomos – Trabalho de Graduação - UnB.

Wolf, W. (1994) .:“Modern VLSI Design: A system Approach”, Englewood Cliffs, Prentice Hall.

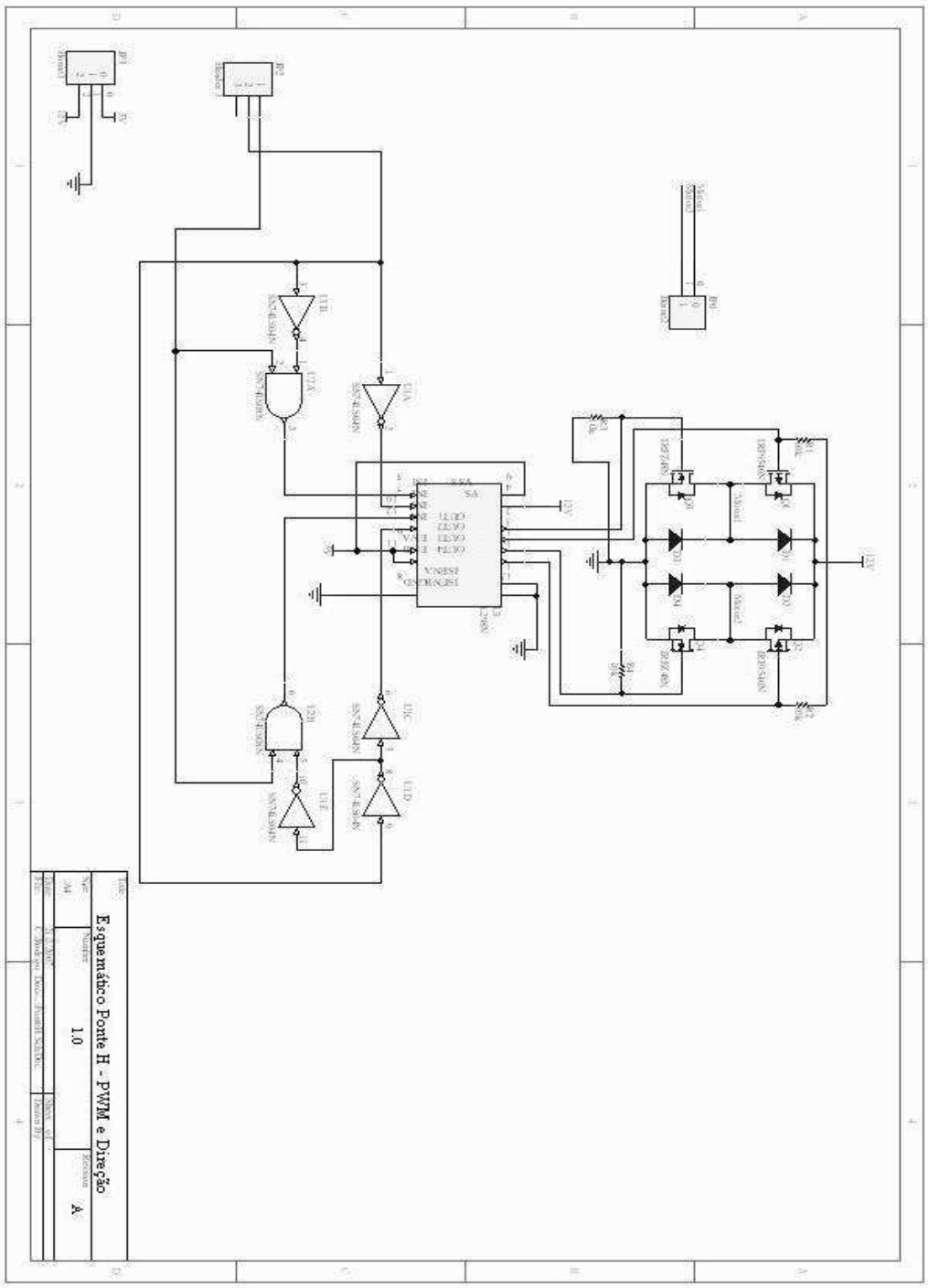
Xilinx : Spartan-3 Starter Kit Board User Guide. UG130 (v1.1) Maio 2005.

Zhao Y., Collins, Jr. E.G (2005).: Robust Automatic Parallel Parking in Tight Spaces via Fuzzy Logic. Robotics and Autonomous Systems. Elsevier, 51 pg.111 – 127.

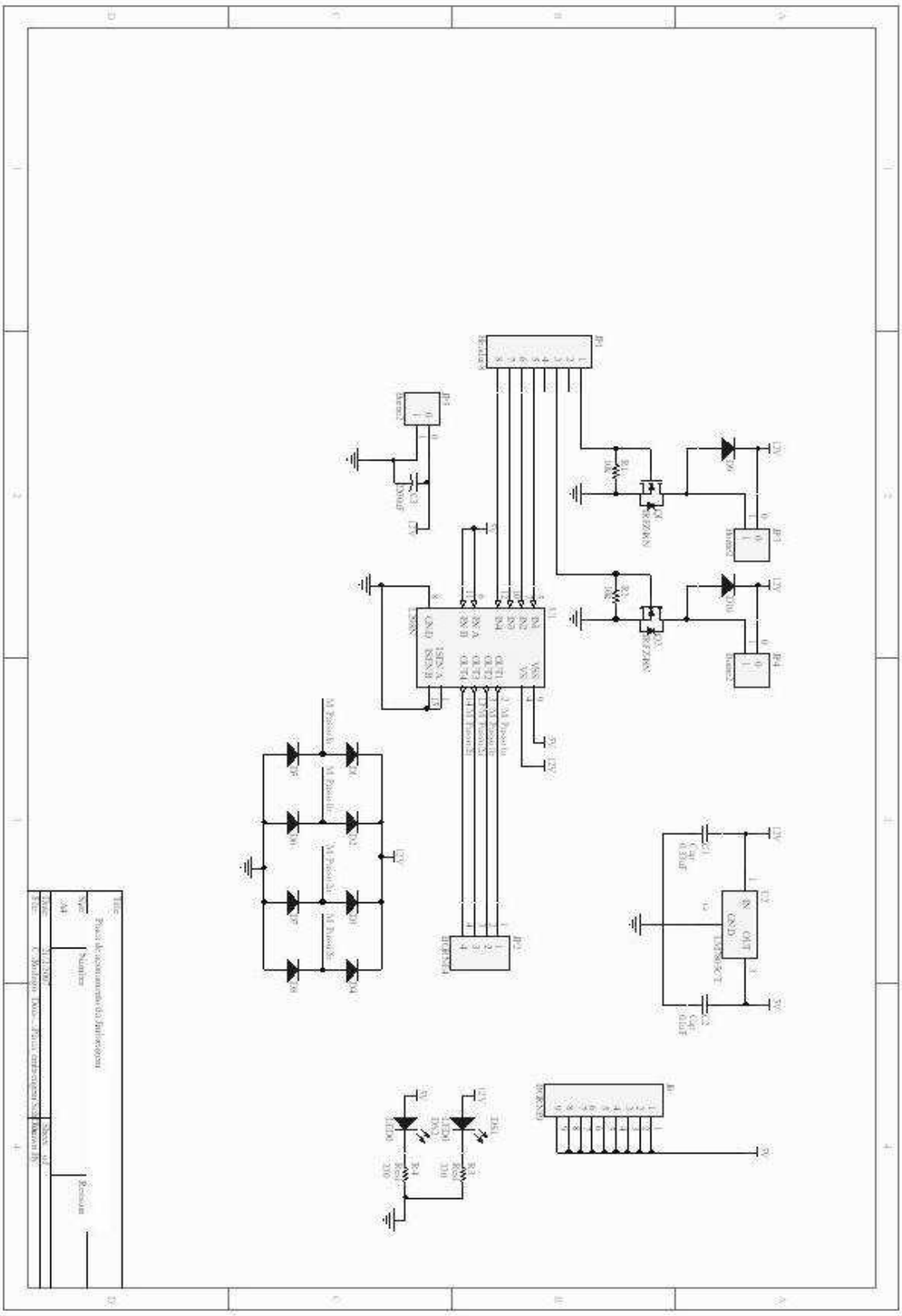
APÊNDICES

APÊNDICE A – DESENHOS DOS CIRCUITOS DAS PLACAS DE POTÊNCIA E SINAL

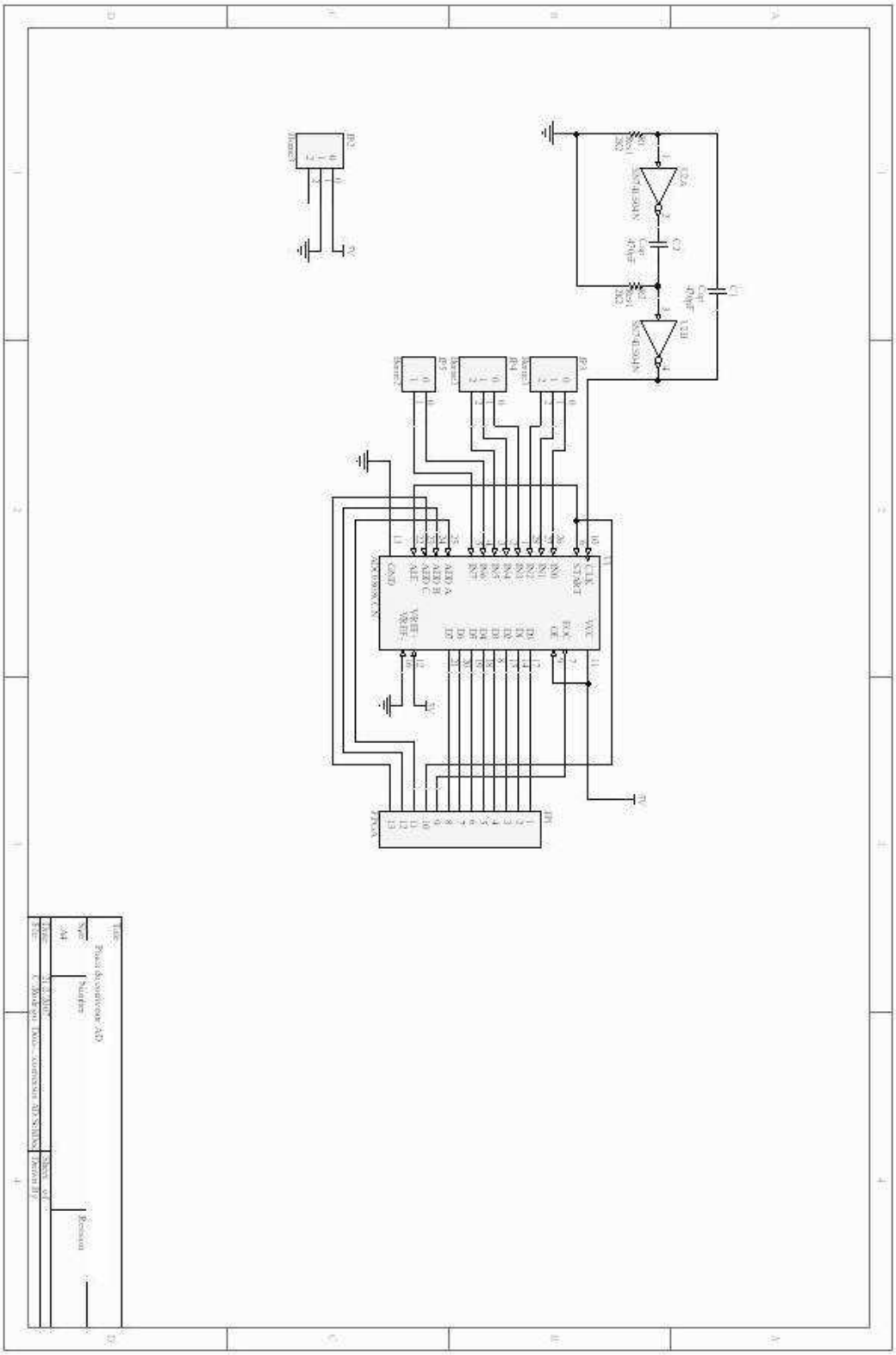
- **CIRCUITO DA PLACA DE POTÊNCIA PONTE H**



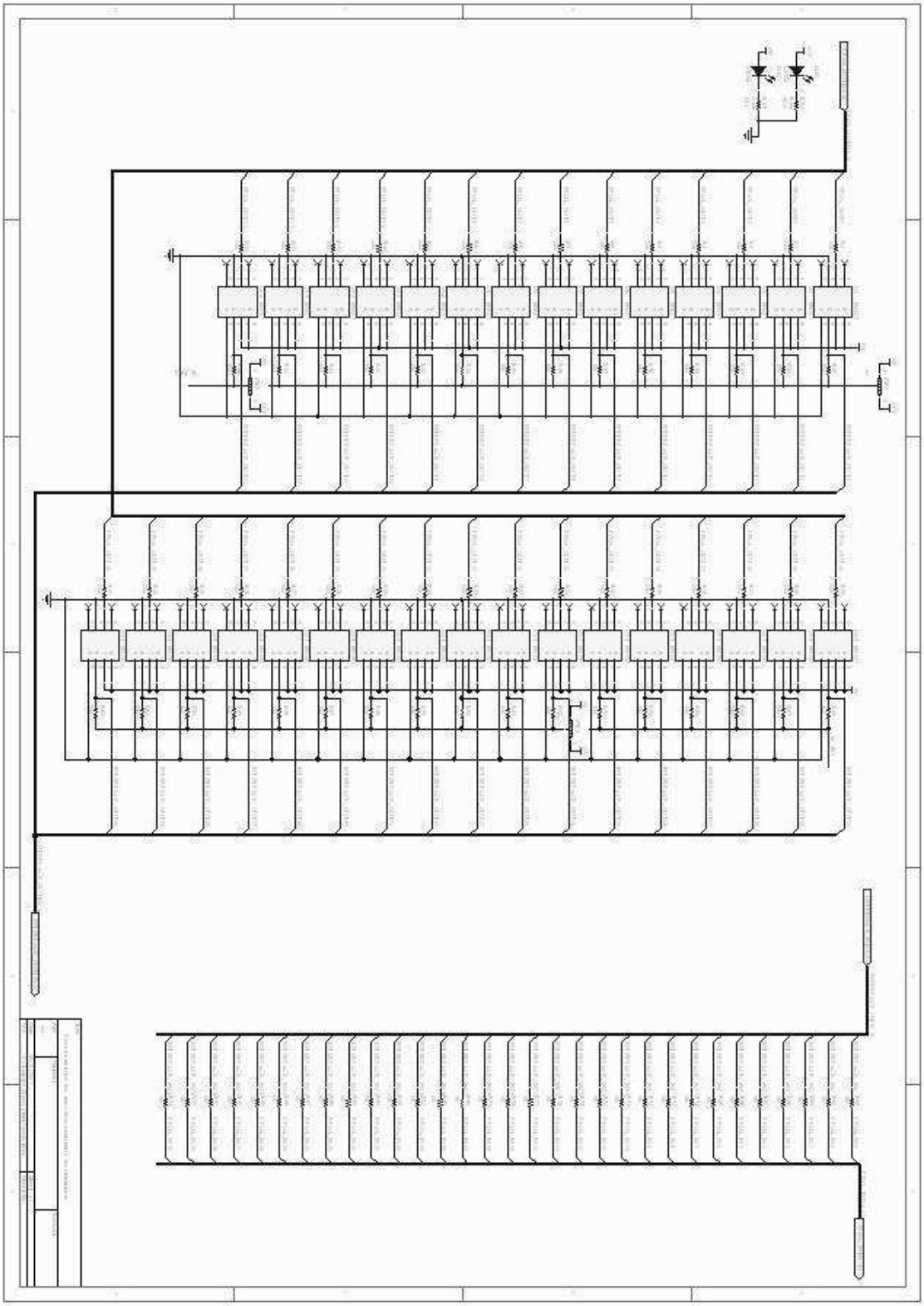
➢ CIRCUITO DA PLACA DE ACIONAMENTO DA EMBREAGEM



- **CIRCUITO DA PLACA DO CONVERSOR ANALÓGICO DIGITAL – A/D**

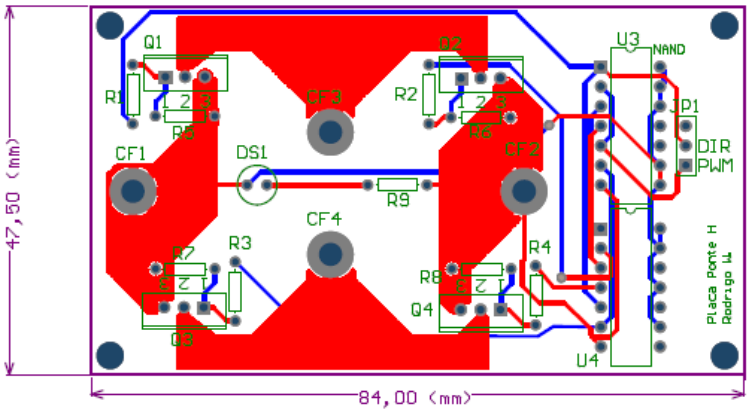


- **CIRCUITO DA PLACA DE INTERFACE DE POTÊNCIA E SINAIS**

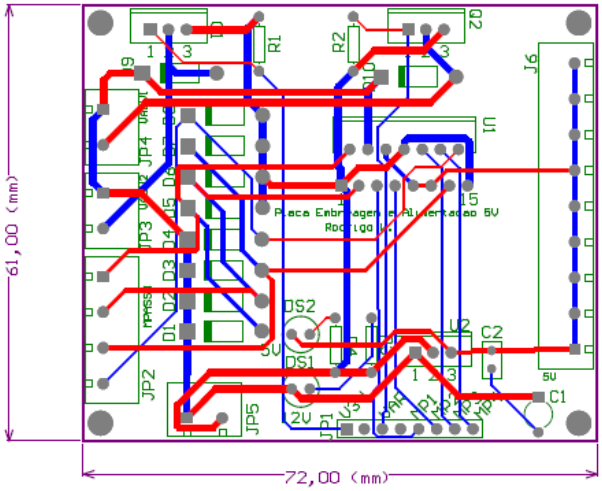


APÊNDICE B – DESENHO DOS PCBs DAS PLACAS DE POTÊNCIA E SINAL

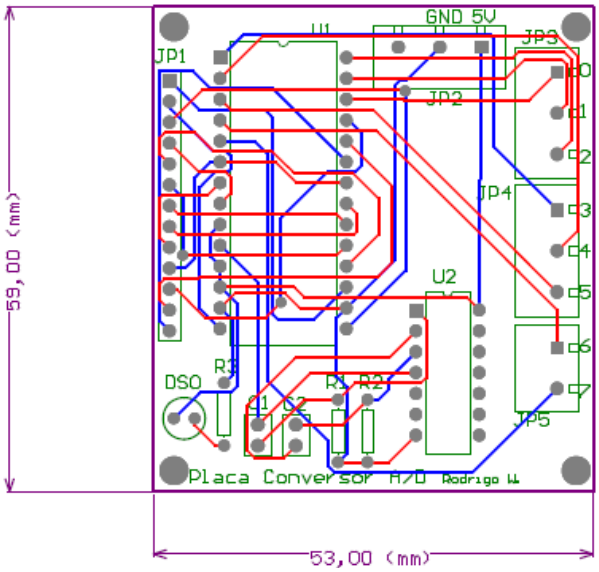
- **PCB DA PLACA DE POTÊNCIA PONTE H**



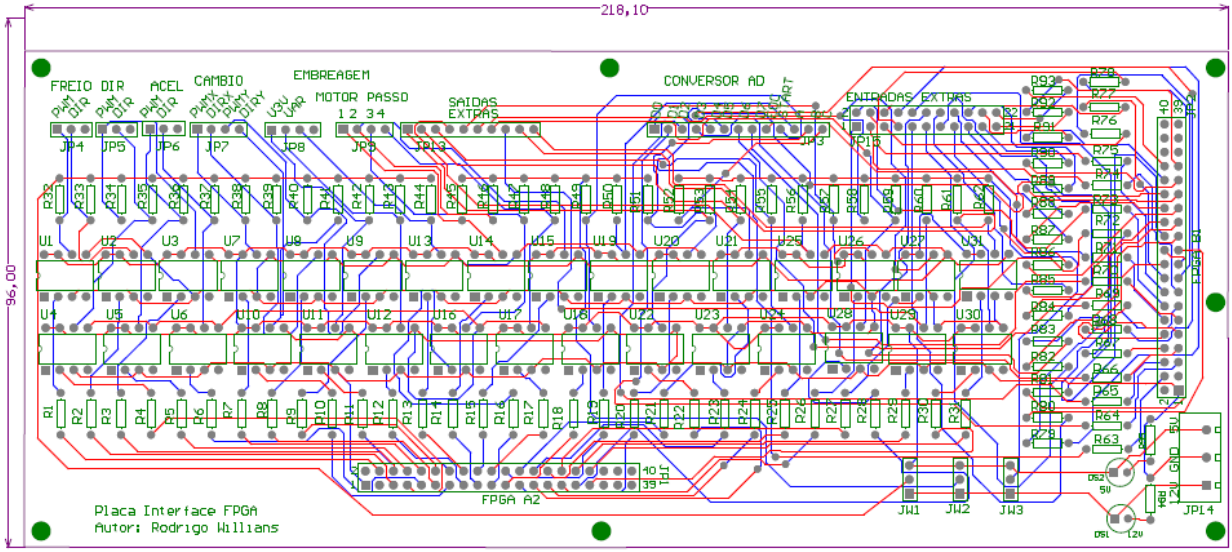
- **PCB DA PLACA DE ACIONAMENTO DA EMBREAGEM**



- **PCB DA PLACA DO CONVERSOR ANALÓGICO DIGITAL – A/D**



- **PCB DA PLACA DE INTERFACE DE POTÊNCIA E SINAIS PARA A FPGA**

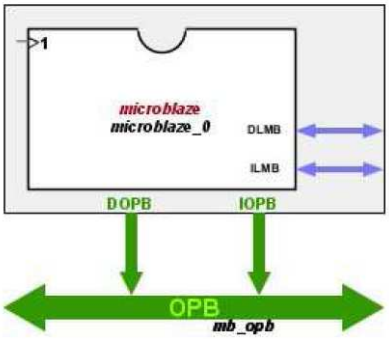


APÊNDICE C – RELATÓRIO DO SISTEMA DE CONTROLE

➢ PROCESSADOR

o MicroBlaze

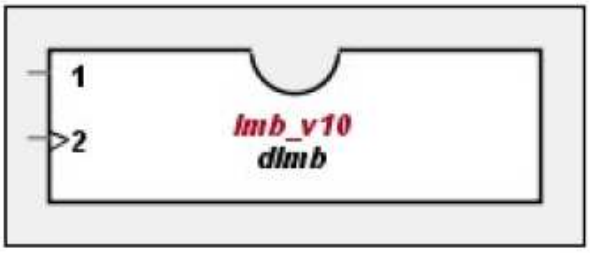
Utilização dos recursos da FPGA - Microblaze			
Recursos	**Usado**	**Disponível**	**% Usado**
Slices	827	1920	43.07%
Slice Flip Flops	557	3840	14.51%
LUTs de 4 entradas	1119	3840	29.14%
MULT18X18s	3	12	25.00%



➢ BARRAMENTOS DE COMUNICAÇÃO

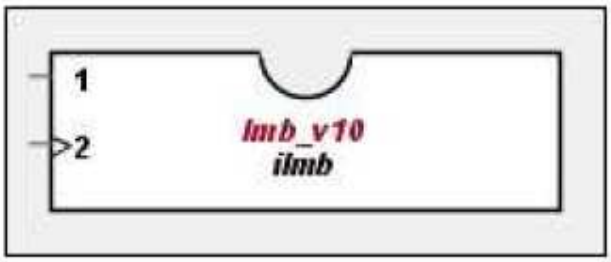
o dlmb

Utilização dos recursos da FPGA - dlmb			
Recursos	**Usado**	**Disponível**	**% Usado**
Slices	1	1920	0.05%
Slice Flip Flops	1	3840	0.03%
LUTs de 4 entradas	1	3840	0.03%
MULT18X18s	-	-	0.00%



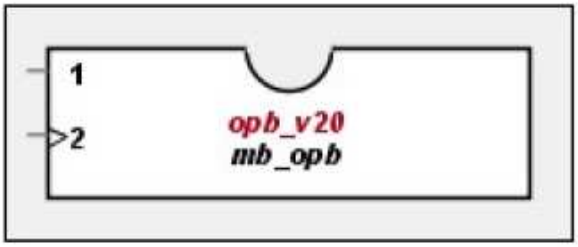
- **ilmb**

Utilização dos recursos da FPGA - ilmb			
Recursos	**Usado**	**Disponível**	**% Usado**
Slices	1	1920	0.05%
Slice Flip Flops	1	3840	0.03%
LUTs de 4 entradas	1	3840	0.03%
MULT18X18s	-	-	0.00%



- **mb_opb**

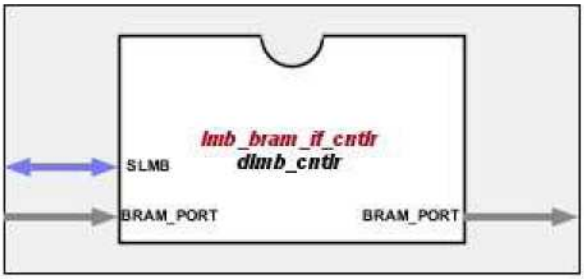
Utilização dos recursos da FPGA - mb_opb			
Recursos	**Usado**	**Disponível**	**% Usado**
Slices	145	1920	7.55%
Slice Flip Flops	11	3840	0.29%
LUTs de 4 entradas	249	3840	6.48%
MULT18X18s	-	-	0.00%



- **Controle de MEMÓRIA**

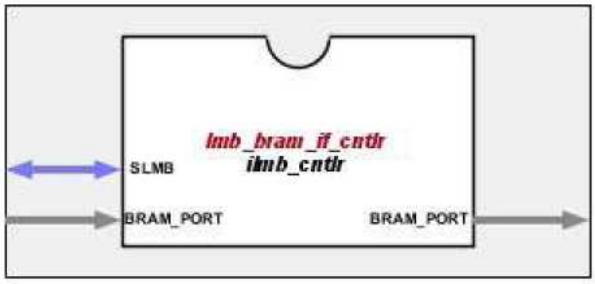
  - **dlmb_bram**

Utilização dos recursos da FPGA - dlmb_bram_cntlr			
Recursos	**Usado**	**Disponível**	**% Usado**
Slices	3	1920	0.16%
Slice Flip Flops	1	3840	0.03%
LUTs de 4 entradas	5	3840	0.13%
MULT18X18s	-	-	0.00%



  - **ilmb_bram**

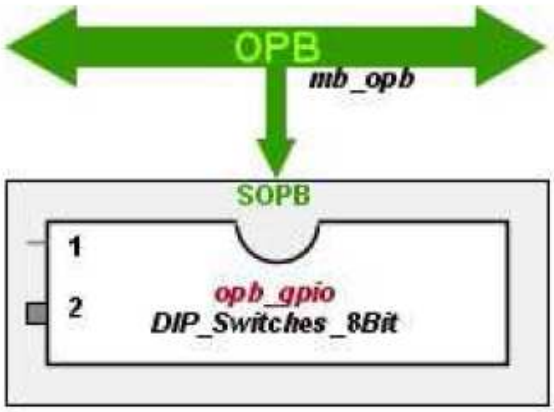
Utilização dos recursos da FPGA - dlmb_bram_cntlr			
Recursos	**Usado**	**Disponível**	**% Usado**
Slices	3	1920	0.16%
Slice Flip Flops	1	3840	0.03%
LUTs de 4 entradas	5	3840	0.13%
MULT18X18s	-	-	0.00%



➢ PERIFÉRICOS

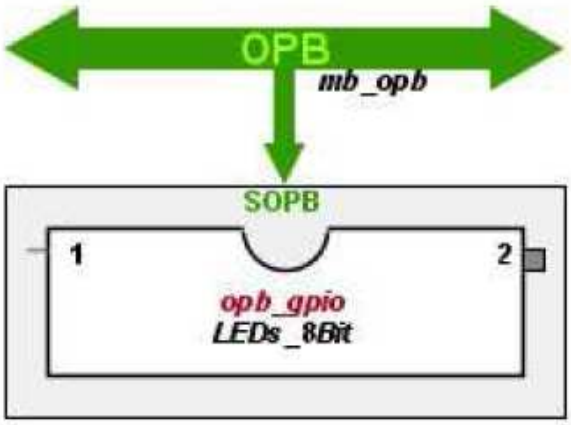
o Chaves – 8 bits

Utilização dos recursos da FPGA - Chaves			
Recursos	**Usado**	**Disponível**	**% Usado**
Slices	49	1920	2.55%
Slice Flip Flops	72	3840	1.88%
LUTs de 4 entradas	27	3840	0.70%
MULT18X18s	-	-	0.00%



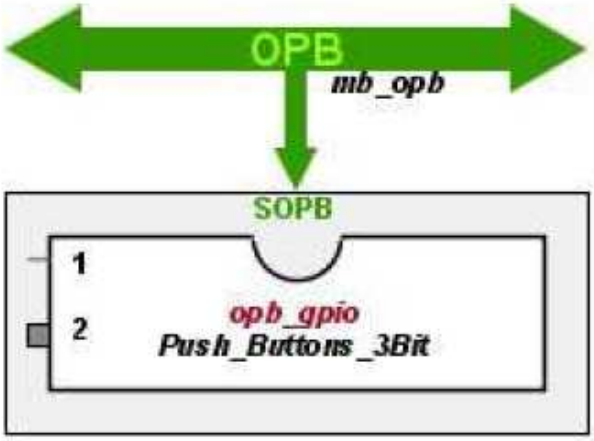
o Leds – 8 bits

Utilização dos recursos da FPGA - leds			
Recursos	**Usado**	**Disponível**	**% Usado**
Slices	67	1920	3.49%
Slice Flip Flops	105	3840	2.73%
LUTs de 4 entradas	37	3840	0.96%
MULT18X18s	-	-	0.00%



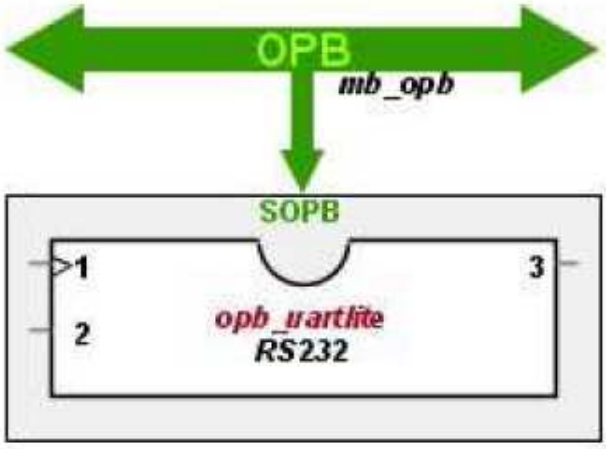
- **Botões – 3 bits**

Utilização dos recursos da FPGA – Botões			
Recursos	**Usado**	**Disponível**	**% Usado**
Slices	42	1920	2.19%
Slice Flip Flops	57	3840	1.48%
LUTs de 4 entradas	27	3840	0.70%
MULT18X18s	-	-	0.00%



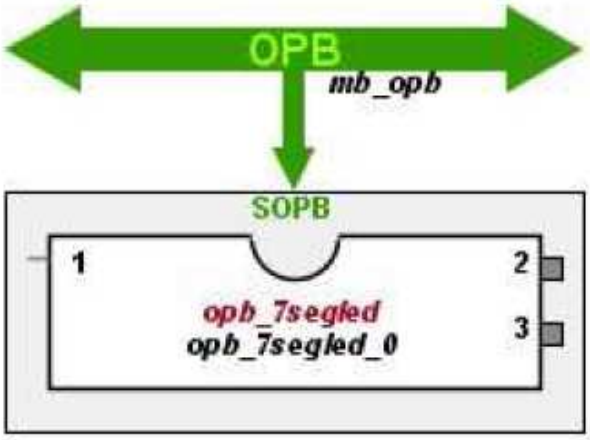
- **RS-232**

Utilização dos recursos da FPGA - rs232			
Recursos	**Usado**	**Disponível**	**% Usado**
Slices	54	1920	2.81%
Slice Flip Flops	62	3840	1.61%
LUTs de 4 entradas	88	3840	2.29%
MULT18X18s	-	-	0.00%



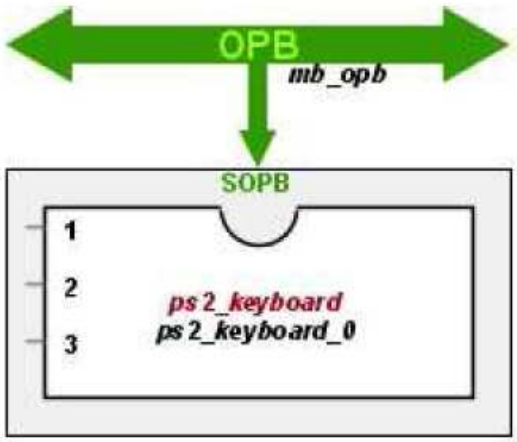
- **Display de sete segmentos**

Utilização dos recursos da FPGA - display			
Recursos	**Usado**	**Disponível**	**% Usado**
Slices	185	1920	9.64%
Slice Flip Flops	165	3840	4.30%
LUTs de 4 entradas	194	3840	5.05%
MULT18X18s	-	-	0.00%



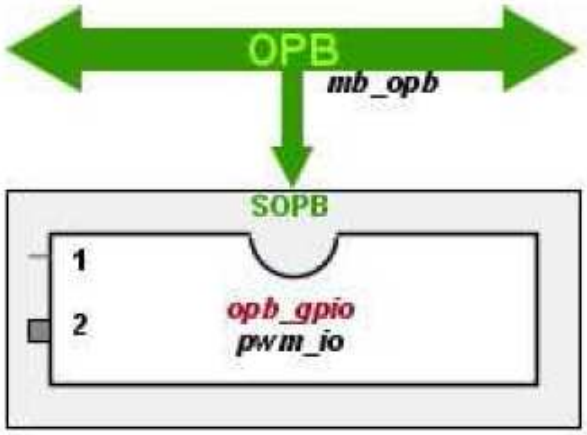
- **Teclado PS 2**

Utilização dos recursos da FPGA - ps2_teclado			
Recursos	**Usado**	**Disponível**	**% Usado**
Slices	54	1920	2.81%
Slice Flip Flops	76	3840	1.98%
LUTs de 4 entradas	46	3840	1.20%
MULT18X18s	-	-	0.00%



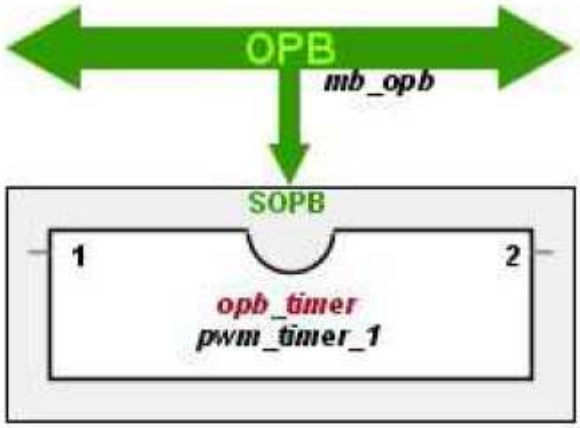
- **PWM_I/O**

Utilização dos recursos da FPGA - pwm/IO			
Recursos	**Usado**	**Disponível**	**% Usado**
Slices	45	1920	2.34%
Slice Flip Flops	63	3840	1.64%
LUTs de 4 entradas	27	3840	0.70%
MULT18X18s	-	-	0.00%



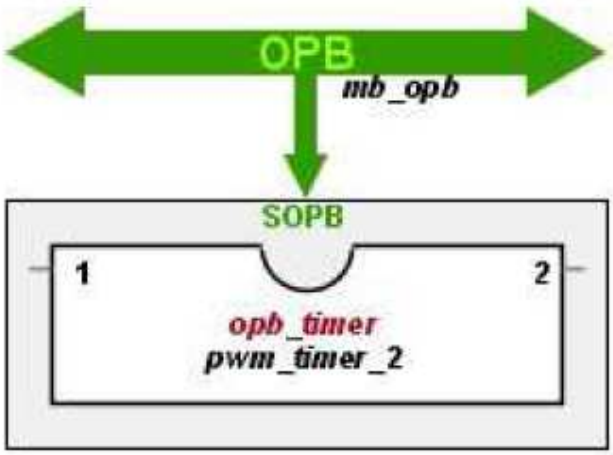
- **PWM_TIMER_1**

Utilização dos recursos da FPGA - pwm_timer1			
Recursos	**Usado**	**Disponível**	**% Usado**
Slices	265	1920	13.80%
Slice Flip Flops	313	3840	8.15%
LUTs de 4 entradas	291	3840	7.58%
MULT18X18s	-	-	0.00%



- **PWM_TIMER_2**

Utilização dos recursos da FPGA - pwm_timer2			
Recursos	**Usado**	**Disponível**	**% Usado**
Slices	265	1920	13.80%
Slice Flip Flops	313	3840	8.15%
LUTs de 4 entradas	291	3840	7.58%
MULT18X18s	-	-	0.00%



- **Freqüência de funcionamento dos módulos**

Limite da freqüência do clock em cada módulo		
Módulo	**Porta do Clock**	**Freqüência máxima (MHz)**
Display 7 segmentos	OPB_Clk	68.362
Microblaze	CLK	91.128
PWM_timer_1	OPB_Clk	98.348
PWM_timer_2	OPB_Clk	98.348
PS2_Teclado	Keyboard_Clk	100.120
PS2_Teclado	OPB_Clk	100.120
PS2_Teclado	Clock_25Mhz	100.120
PS2_Teclado	ready_set	100.120
Leds_8 bits	OPB_Clk	135.612
Chaves	OPB_Clk	135.612
PWM_O/I	OPB_Clk	138.658
Botões	OPB_Clk	138.927
RS232	OPB_Clk	151.676
mb_opb	OPB_Clk	203.707
ilmb	LMB_Clk	238.152
dlmb	LMB_Clk	238.152