UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA MECÂNICA

IMPLEMENTAÇÃO E SIMULAÇÃO DE ALGORITMOS DE ESCALONAMENTO PARA SISTEMAS DE ELEVADORES USANDO ARQUITETURAS RECONFIGURÁVEIS

DANIEL MAURICIO MUÑOZ ARBOLEDA

ORIENTADOR: CARLOS HUMBERTO LLANOS QUINTERO

DISSERTAÇÃO DE MESTRADO EM SISTEMAS MECATRÔNICOS

PUBLICACÃO: ENM.DM- 11A/06

BRASÍLIA/DF: NOVEMBRO – 2006

UNIVERSIDADE DE BRASÍLIA

FACULDADE DE TECNOLOGIA

DEPARTAMENTO DE ENGENHARIA MECÂNICA

IMPLEMENTAÇÃO E SIMULAÇÃO DE ALGORITMOS DE ESCALONAMENTO PARA SISTEMAS DE ELEVADORES USANDO ARQUITETURAS RECONFIGURÁVEIS

DANIEL MAURICIO MUÑOZ ARBOLEDA

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

_____

Pedro de Azevedo Berger, Dr. (CIC-UnB)
(Examinador Externo)

_____

Leandro dos Santos Coelho, Dr. (PUCPR)
(Examinador Externo)

BRASÍLIA/DF, 24 DE NOVEMBRO DE 2006

FICHA CATALOGRÁFICA

MUÑOZ A., DANIEL MAURICIO	
Implementação e Simulação de Algoritmos de Escalonamento para Sistemas de Elevadores Usando Arquiteturas Reconfiguráveis [Distrito Federal] 2006.	
xvii, 128p., 210 x 297 mm (ENM/FT/UnB, Mestre, Sistemas Mecatrônicos, 2006).	
Dissertação de Mestrado – Universidade de Brasília. Faculdade de Tecnologia.	
Departamento de Engenharia Mecânica.	
1.Sistema de Elevadores	2.FPGAs
3.Lógica Nebulosa	4.Sistemas Inteligentes
I. ENM/FT/UnB	II. Título (série)

REFERÊNCIA BIBLIOGRÁFICA

MUÑOZ, DANIEL M. (2006). Implementação e Simulação de Algoritmos de Escalonamento para Sistemas de Elevadores Usando Arquiteturas Reconfiguráveis. Dissertação de Mestrado em Sistemas Mecatrônicos, Publicação ENM.DM-11A/06, Departamento de Engenharia Mecânica, Universidade de Brasília, Brasília, DF, 128p.

CESSÃO DE DIREITOS

AUTOR: Daniel Mauricio Muñoz Arboleda.

TÍTULO: Implementação e Simulação de Algoritmos de Escalonamento para Sistemas de Elevadores Usando Arquiteturas Reconfiguráveis.

GRAU: Mestre **ANO: 2006**

_____

Daniel Mauricio Muñoz Arboleda

CLN 407 Bloco C, sala 203.

70855-030 Brasília – DF – Brasil.

AGRADECIMENTOS

Agradeço, sinceramente, ao meu orientador, Prof. Dr. Carlos Humberto Llanos Quintero, pelos conhecimentos transmitidos, a competência, as acertadas decisões e, mais ainda, pela confiança, paciência e amizade.

Aos meus colegas e amigos, André Luiz Sordi Braga, Magno Batista Correa, Jones Yudi Mori e Renato Pereira pelas dicas de programação, apoio e sugestões ao longo do trabalho.

À CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior) e à FINEP (Financiadora de Estudos e Projetos) pelo apoio financeiro deste trabalho. Ao Grupo de Automação e Controle (GRACO) e todos meus professores pelo suporte e formação acadêmica.

Sou grato, de forma especial, a todos meus colegas de curso e amigos, dentro e fora da universidade, pelo carinho, disposição e os momentos de alegria atrás destas folhas.

À minha família, porque apesar da distância, seu apoio, estímulo e compreensão foram muitas vezes o motor deste trabalho.

RESUMO

Este trabalho propõe um sistema de elevadores que permite o transporte vertical de passageiros de uma forma eficiente. A abordagem é baseada na implementação de algoritmos de escalonamento usando arquiteturas reconfiguráveis. Um método baseado em lógica nebulosa foi proposto no intuito de identificar padrões de tráfego no edifício e despachar os elevadores, adotando diferentes estratégias de atendimento de chamadas.

Os sistemas de elevadores modernos para transporte vertical de passageiros são freqüentemente implementados por controladores microprocessados, no intuito de executar as tarefas de controle e ação. O estudo de estratégias para controle de elevadores tenta otimizar o desempenho do sistema, incrementando o fluxo de transporte e o conforto dos usuários. Ao mesmo tempo, o consumo de potência do sistema deve ser diminuído.

A arquitetura proposta para o Sistema de Controle Local (LCS) considera o uso de cinco algoritmos de escalonamento, os quais foram implementados em placas de desenvolvimento FPGA (*Field Programmable Gate Array*) do tipo Spartan3 numa abordagem integrada, reduzindo o consumo de área e otimizando o desempenho do circuito.

O Sistema de Controle de Grupo de Elevadores (EGCS), baseado em lógica nebulosa (FEGCS) foi desenvolvido em linguagem *Java*. Este sistema permite validar o desempenho dos algoritmos para diferentes situações de tráfego. Os resultados de simulação mostram que o tempo de espera é reduzido sempre que o consumo de potência é incrementado. O tempo de espera médio dos passageiros é aproximadamente de 36 segundos em um padrão de tráfego de descida.

O número de cálculos no controlador de grupo é reduzido, dado que o EGCS não esta diretamente envolvido em calcular o próximo andar a ser visitado. A implementação em *hardware* dos algoritmos de escalonamento permite melhorar o desempenho do cálculo do próximo andar a ser visitado por cada elevador.

ABSTRACT

This work proposes an elevator system that allows the vertical transport of passengers in a efficient way. This approach is based on the implementation of dispatching algorithms using *Reconfigurable Architectures*. A fuzzy logic method was proposed in order to identify traffic patterns in the building and schedule the elevators, which was carried out by implementing several strategies for attending the hall-calls.

Modern elevator systems for vertical transport of the passengers are frequently implemented by several microprocessed controllers in order to achieve the control and several action tasks. The study of elevator control strategies tries to improve the performance of the system, incrementing the transport flow and the comfort of the users. At the same time, the power consumption of the overall system must be reduced.

The proposed *Local Control System* (LCS) architecture considers the use of five dispatching algorithms for elevator systems, which were implemented on Spartan 3 FPGA (Field Programmable Gate Array) based boards in a integrated approach, reducing the area consumption of the overall circuit and improving its performance.

The *Elevator Group Control System* (EGCS) based on fuzzy logic (FEGCS) was developed on Java language. This system allows to validate the algorithms performance for different traffic situations. Simulation results show that the waiting time is reduced whenever the power consumption is incremented. The average waiting time of the passengers is about 36 seconds in a down traffic pattern.

The number of calculations in the group controller is reduced given that the EGCS is not directly involved in calculating the next floors to be visited. The hardware implementation of the dispatching algorithms allows to improve the performance calculation of the next floor to be visited by each elevator.

SUMÁRIO

LISTA DE TABELAS

LISTA DE FIGURAS

LISTA DE SIMBOLOS, NOMENCLATURAS E ABREVIAÇÕES

ABCI - Associação Brasileira de Construção Industrializada

ABNT - Associação Brasileira de Normas Técnicas

AI - *Artificial Intelligence* – Inteligência Artificial

ASIC - *Application Specific Integrated Circuits* – Circuito Integrado de Aplicação Específica

AWT - *Average Waiting Time* – Tempo de espera médio

BT - Padrão de tráfego de negócios

BTH - Padrão de tráfego combinação negócios / pesado

CAD - *Computer Aided Desig*n – Projeto Auxiliado por Computador

CITP - Porcentagem de tráfego centralizado

CMOS - *Complementary Metal Oxide Semiconductor*

CPLD - *Complex Programmable Logic Device* – Dispositivo Lógico Programável

DOTP - Porcentagem de tráfego distribuído

Down - Padrão de tráfego para pico de descida

EDA - *Electronic Design Automation* – Automatização de Desenho Eletrônico

EGCS - Sistema de Controle de Grupo de elevadores

FEGCS - Sistema Nebuloso de Controle de Grupo de Elevadores

FPGA - *Field Programmable Gate Array*

HC - Capacidade de transporte

HT - Padrão de tráfego pesado

I - Intervalo de tempo

IT - Padrão de tráfego inativo

ISE - *Integrated Software Environment* – Ambiente de *Software* Integrado

ISO - International Organization for Standarization

LB - *Logic Block – Bloco lógico*

LCS - *Sistema de Controle Local*

LTa - Padrão de tráfego para hora do lanche A

LTb - *Padrão de tráfego para hora do lanche B*

LTU - *LookUp Table* – Tabela de pesquisa

n - Número de andares

p - Número de passageiros

PAL - *Programmable Array Logic – Lógica de Programação de Registros*

PC - *Personal Computer* – Computador pessoal

PLA - *Programmable Logic Array* – Arranjo Lógico Programável

PLD - *Programmable Logic Devi*ce – Dispositivo Lógico Programável

O - Custo de área

Oc - Custo de área dos recursos combinacionais

RAM - *Random Access Memory* – Memória de Acesso Aleatório

SV - Valor de conveniência

TS - Tráfego de subida

TD - Tráfego de descida

TTL - *Transistor Transistor Logic* - Lógica Transistor Transistor

ULA - *Unit Logic Arithmetic* - Unidade Lógica Aritmética

Up - Padrão de tráfego pico de subida

VEI - Interface Virtual de Elevadores

VLSI - *Very Large Scale Integration* – Integração em Escala Muito Grande

x - Variável lingüística

X - Universo de discurso

1 – INTRODUÇÃO

Os sistemas de elevadores modernos para o transporte vertical de passageiros utilizam distintos componentes microprocessados que realizam as tarefas de sensoriamento, controle e ação. Geralmente, estes componentes encontram-se distribuídos no prédio e são interligados numa rede de comunicação. Os dispositivos de sensoriamento e de ação são especificados pelas normas de segurança e não apresentam mudanças radicais no princípio de funcionamento nos sistemas de elevadores. Por outro lado, as políticas de controle dos elevadores é uma área ainda em estudo que visa melhorar o desempenho do sistema, aumentar o fluxo de transporte, diminuir o consumo de potência e aumentar o conforto dos usuários.

Propor um sistema de controle de elevadores usando arquiteturas reconfiguráveis é um grande desafio visto que as arquiteturas convencionais microprocessadas dominam o mercado. Porém, a maioria destas arquiteturas não sempre se ajustam às condições exigidas e resulta difícil configurá-las para diferentes tipos de edifícios. Entretanto, os resultados alcançados em relação ao desempenho e confiabilidade não são satisfatórios.

1.1 – GENERALIDADES

A diversificação e evolução dos serviços oferecidos em edifícios inteligentes são baseadas no aproveitamento eficiente dos recursos tecnológicos atualmente existentes. Estes serviços consideram um estudo do comportamento e iteração dos usuários visando aumentar a produtividade dos mesmos, uma vez que o edifício ofereceria um ambiente adequado para atender às suas necessidades (Finley *et al.*, 1991).

Por outro lado, os edifícios inteligentes devem ser projetados de forma que possam incrementar a competitividade mediante a aplicação de tecnologias emergentes. Em conseqüência espera-se que estes edifícios possam adaptar-se a novos requerimentos objetivando serem produtivos ao longo do seu período de vida (Finley *et al.*, 1991).

De acordo com as anteriores considerações, um edifício inteligente deve incorporar sistemas mecatrônicos para o controle automatizado e a monitoração das interfaces

envolvidas. No entanto, devem ser incorporados sistemas de telecomunicação e segurança, considerando a integração, otimização e flexibilidade dos serviços possibilitando de uma forma simples e econômica a incorporação de novos sistemas (Finley *et al.*, 1991).

O presente trabalho faz parte de um projeto geral denominado *Sistema Integrado de Automação Predial* (SIAP) em desenvolvimento no *Grupo de Automação e Controle* (GRACO) na Universidade de Brasília. Este projeto propõe a integração dos diferentes serviços oferecidos em edifícios modernos, usando tecnologias existentes no mercado, mantendo o código fonte aberto e o uso de plataformas livres. Um dos objetivos do projeto SIAP é unificar, através de redes e protocolos de comunicação, as interfaces de aplicação do prédio num ambiente agradável aos usuários. A figura 1.1 apresenta a estrutura de desenvolvimento do projeto SIAP.

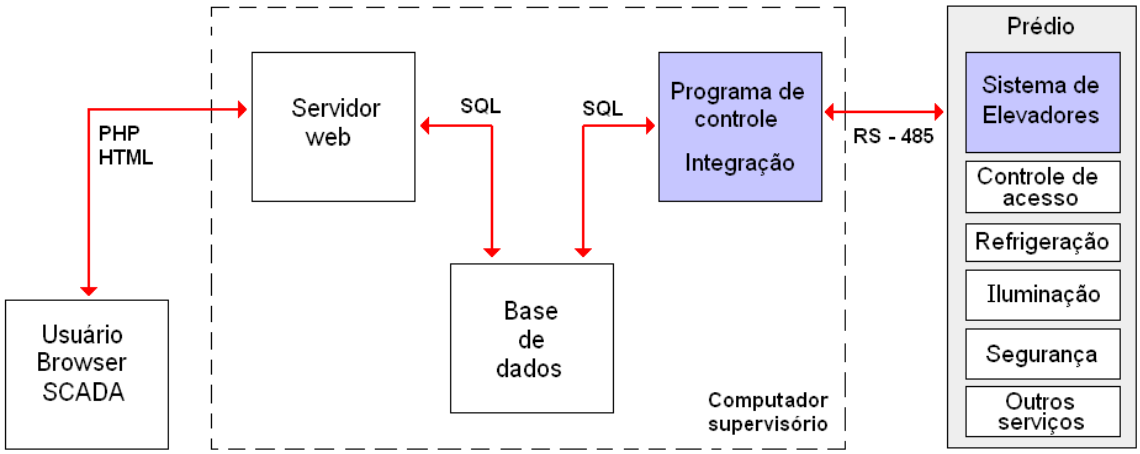


Figura 1.1: Ambiente de desenvolvimento do Projeto SIAP

O foco do presente trabalho em relação à estrutura do projeto SIAP é a implementação, controle e monitoração do sistema de elevadores usando dispositivos reconfiguráveis e tecnologias de fácil acesso.

Um sistema de elevadores é composto por um conjunto de interfaces: botões de chamada de pavimento, botões de chamada de cabina, sistemas de Controle Local de Elevadores (LCS), um Controle de Grupo de Elevadores (EGCS) e uma rede de comunicação. O controle de grupo administra múltiplos elevadores, enviando comandos de ação para os controladores locais. Entretanto, o EGCS analisa as informações sobre tráfego e distribuição de passageiros no edifício, visando tomar decisões baseadas em estratégias para atender uma nova chamada ou um grupo de chamadas. O controle local de elevador

implementa o sistema de movimentação de uma cabina, controlando o motor do elevador, motor da porta, controle de velocidade e aceleração, *displays* de apresentação de estado, entre outras interfaces. Os controladores locais de elevadores são comumente implementados por PLCs ou dispositivos microcontrolados (Otis, 2006, Siikonen, 1997).

A tarefa principal do controlador de grupo, comumente denominado Sistema de Controle de Grupo de Elevadores (EGCS), é administrar o despacho de múltiplos elevadores visando melhorar o desempenho no transporte vertical de passageiros. Os objetivos principais são a diminuição do tempo de espera dos usuários e a diminuição do consumo de potência do sistema.

1.2 – DESCRIÇÃO DO PROBLEMA

Algumas técnicas para a implementação de Sistemas de Controle de Grupo têm sido desenvolvidas, propostas e comercializadas por alguns fabricantes na indústria de elevadores (Otis, 2006, ThyssenKrupp, 2006). Porém, as ditas técnicas requerem que o controle de grupo resolva a totalidade do problema de despacho dos elevadores e o escalonamento das chamadas. Por outro lado, estas técnicas não são facilmente configuráveis, pois sempre que um novo componente quiser ser adicionado ao sistema de elevadores (novo andar ou cabina) é necessário programar os dispositivos microcontrolados e adicionar novas placas de controle. Isto resulta em trocar parcial ou totalmente o quadro de comando. O mesmo problema apresenta-se no caso de uma simples modernização do sistema.

No transporte vertical de passageiros, dois fatores principais devem ser diminuídos, o primeiro deles é o tempo de espera dos usuários nos pavimentos e dentro das cabinas. O segundo fator é o consumo de potência do sistema de elevadores. O resultado é o aumento do fluxo de passageiros no prédio oferecendo conforto e baixo consumo de energia elétrica.

De forma geral, o problema de encontrar a melhor solução para o atendimento de passageiros no transporte vertical, levando em conta os fatores mencionados, é considerado um problema computacionalmente complexo. Contudo, o desafio deste projeto é propor uma solução ao problema de transporte vertical de passageiros objetivando diminuir o

tempo de espera dos usuários e o consumo de potência do sistema de elevadores de forma que a implementação seja realizada de forma flexível, eficiente e facilmente configurável.

1.3 - OBJETIVOS

1.3.1 - Objetivo Geral

O objetivo geral deste trabalho é projetar, desenvolver e simular um Sistema de Elevadores, com a finalidade de servir para o transporte vertical eficiente de passageiros.

A abordagem considerada para este propósito baseia-se na implementação de algoritmos de escalonamento de elevadores usando arquiteturas reconfiguráveis. Por outro lado, é realizado o desenvolvimento de modelos heurísticos baseados em lógica nebulosa, para a identificação de padrões de tráfego no prédio, visando adotar estratégias de atendimento e despacho dos elevadores.

1.3.2 - Objetivos Específicos

Os objetivos específicos deste trabalho são os seguintes:

1. Estudar, projetar, simular e implementar diferentes algoritmos de escalonamento de elevadores, diretamente em *hardware,* mediante o uso de arquiteturas reconfiguráveis.

2. Projetar, simular e implementar o Sistema de Controle Local (LCS) baseado em arquiteturas reconfiguráveis, usando placas FPGA. Integrar as funcionalidades e interfaces necessárias para o funcionamento de um controle local convencional.

3. Projetar, simular e implementar um controle de grupo de elevadores baseado em lógica nebulosa, visando caracterizar o tráfego e distribuição de passageiros no prédio.

4. Construir uma rede de comunicação baseada no padrão RS-485 para interligar um grupo de controles locais de elevadores ao sistema de controle de grupo. Definir um protocolo de comunicação simples baseado no contexto mestre/escravo dentro do

padrão RS-485.

5. Construir uma interface flexível de monitoração e simulação de grupo de elevadores, baseada em linguagem de programação orientada a objetos. Validar os algoritmos de escalonamento em diferentes situações de tráfego e comparar os resultados de escalonamento.

6. Definir estratégias de divulgação de resultados, mantendo o código de fonte aberto e o uso aplicações livres de plataforma. Definir uma metodologia que permita estudar e comparar os resultados obtidos no escalonamento dos elevadores.

1.4 – JUSTIFICATIVA

Na área de edifícios inteligentes, especificamente em edifícios comerciais modernos, nos quais existe uma alta densidade de pessoas, a demanda dos negócios considera a capacidade de movimentar e re-agrupar pessoal de forma fácil e econômica. Os sistemas de automação predial fornecem capacidade de monitorar e automatizar funções que permitam economizar a operação e a manutenção dos serviços oferecidos no prédio. Entretanto, é possível trocar os sistemas análogos convencionais por ambientes de controle digital que permitem implementar algoritmos complexos visando diminuir o consumo de energia.

Conforme os estudos da Associação Brasileira de Construção Industrializada (ABCI) é observado um aumento aproximado de 10% na produtividade dos profissionais alocados em edificações com maiores graus de automação (ABCI, 1993, Mikkola e Grassman, 2003). Nos edifícios inteligentes, projetados para um tempo de vida médio de 50 anos, é justificada a inversão em automação predial (3% do total do investimento) devido ao custo relativamente baixo comparado com os custos de operação e manutenção (65 a 80% do total do investimento) (Han, 1997). Entretanto, a aquisição de sistemas de elevadores não constitui um gasto senão uma inversão, considerando que o valor patrimonial do imóvel é maior quando o prédio possui um sistema de elevadores moderno (Infolev, 2006).

O mercado mundial atual de elevadores estima cerca de 100 bilhões de reais por ano (35 milhões de euros), sendo que o 35% representa a aquisição de novos elevadores, 30% modernização e um 35% de manutenção. No Brasil existem ao redor de 250.000

elevadores instalados, dos quais 50% têm mais de 40 anos de uso, e a maioria destes, totalmente obsoletos (Infolev, 2006). A principal causa de acidentes reportados é causada pelo nivelamento errado da cabina o que compromete a segurança dos usuários. Segundo Infolev (2006), estipula-se que no ano 2010 entre em rigor um projeto de lei que exija um nivelamento aproximado de 10mm para todos os elevadores instalados, para o qual serão necessários investimentos em modernização do sistema de elevadores (uso de novas tecnologias de controle e automação).

A modernização de elevadores traz diversos benefícios, como por exemplo, melhoras em segurança e desempenho, eliminação de ruídos e vibrações produzindo viagens mais confortáveis e economia de energia de até 40% (Tianxiao, 2004, Sachs, 2005).

Num prédio residencial, além dos benefícios em desempenho e conforto dos usuários, existe um componente econômico relevante que um sistema moderno de elevador pode oferecer. Aproximadamente, o 7% do valor mensal do condomínio é pago somente na manutenção do elevador, e 6% do valor do condomínio é pago na energia consumida pelos elevadores. Em prédios residenciais, um sistema de elevadores eficiente e moderno estima uma economia de energia de aproximadamente de 5 a 7% do valor na fatura de condomínio (Roger, 2005).

Os elevadores para o transporte vertical de pessoas são considerados sistemas fundamentais de suporte às atividades de locomoção dos usuários de um edifício. Justifica-se estudar a integração destes sistemas com outros sistemas prediais, facilitando a instalação, manutenção e operação do elemento de automação. Dentro desta perspectiva, a integração dos sistemas prediais, deve ser analisada cuidadosamente em função da sua influencia no desempenho e produtividade das atividades dos usuários do edifício (Plönnigs *et al.* 2004, Bushby, 1997). Conseqüentemente, os serviços oferecidos pelo sistema integrado de automação predial (sistema de elevadores, controle de acesso, ar condicionado, entre outros) seriam mais eficientes (Wang e Xie, 2002, Wang e Xie, 2004).

Considerando a relevância que os sistemas de elevadores têm nos edifícios em geral, torna-se importante uma implementação que permita modificar ou selecionar alternativas de atendimento e que possa ao mesmo tempo ser configurável a diferentes tipos de situações, facilitando o uso de políticas de operação de forma a reduzir tempos e custos.

1.5 - CONTRIBUIÇÕES DO TRABALHO

Este trabalho realiza um aporte ao estado da arte na industria dos elevadores. Algoritmos de escalonamento para atendimento das chamadas solicitadas foram implementados diretamente em hardware, fornecendo maior flexibilidade e desempenho em relação a sistemas de elevadores convencionais microcontrolados. Uma arquitetura distribuída em que os Sistemas de Controle Local de elevadores (LCSs) são autônomos propõe diminuir os cálculos realizados pelo controlador de grupo. Entretanto, um módulo baseado em lógica nebulosa identifica padrões de tráfego no prédio e realiza o despacho dos elevadores visando diminuir o tempo de espera dos usuários e o consumo de potência do sistema.

1.6 – ORGANIZAÇÃO DO TRABALHO

O capítulo 2 apresenta a fundamentação necessária para o desenvolvimento deste trabalho abordando assuntos como funcionamento e características do sistema de elevadores, *hardware* reconfigurável e lógica nebulosa.

O capítulo 3 descreve os recursos de *hardware* e *software* utilizados na implementação e validação do sistema proposto.

O capítulo 4 descreve as arquiteturas desenvolvidas na implementação do sistema de elevadores, como também a construção da interface virtual de elevadores para validação, simulação e controle de grupo.

O capítulo 5 apresenta a aplicação particular desenvolvida a partir do sistema proposto, mostrando resultados de teste, simulação e escalonamento de elevadores. Finalmente, são relacionadas as conclusões e sugestões para trabalhos futuros no capítulo 6.

2 - FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta a fundamentação teórica dos temas envolvidos no presente projeto. Inicialmente é introduzida um breve histórico dos sistemas de elevadores, enfatizando a evolução do transporte vertical e o estado da arte. O segundo conceito descreve algumas considerações de importância na área de sistemas de elevadores aplicados aos edifícios em geral, visando explicar suas características, funcionamento, classificações de tráfego e tipos de escalonamento. Em seguida é abordado o tema relacionado à tecnologia de *hardware* adotada neste trabalho, com ênfase na tecnologia de *hardware* reconfigurável baseada em FPGA (*Field Programmable Gate Array*). Finalmente é apresentado o conceito de lógica nebulosa como mecanismo de controle para o despacho de um grupo de elevadores.

2.1 - HISTÓRIA E ESTADO DA ARTE NA INDÚSTRIA DE ELEVADORES

Quando os grandes grupos habitacionais se conformaram e as civilizações antigas interagiam entre guerras e comercio o homem criou as primeiras maquinas de transporte vertical. Em estes grupos populacionais o transporte de passageiros não era importante, o que em certa medida atrasou a chegada do que hoje se conhece como cidade moderna. Entretanto a necessidade de mover rapidamente produtos, mercancias e artefatos de um lugar mais baixo a um lugar mais alto, ou vice-versa, foi um problema que precisou grandes esforços.

No século III a.c aqueles elevadores primitivos eram acionados por energia humana ou animal e como resultado a segurança, o conforto e a eficiência do transporte estava comprometida. Durante séculos as comunidades foram agrárias e de vida horizontal e não foi até anos depois da revolução industrial quando os modernos elevadores para transporte de passageiros foram desenvolvidos (Otis, 2006, Siikonenb, 1997, Bronowski, 1992).

Na metade do século XIX quase não existiam prédios com mais de três pavimentos. Só a partir de 1852 quando Elisha Graves Otis fez a maior invenção na indústria dos elevadores, introduzindo um revolucionário mecanismo de segurança automático que permitia o elevador parar caso as cordas de tração falhem. A partir deste momento começaram a ser

construídos prédios de maior altura. Com aquela invenção, quando os cabos rompiam, uma mola acionava duas garras encima da cabina forçando-as a segurar-se sobre os suportes no poço do elevador. Esta invenção acelerou a construção de elevadores e o desenvolvimento na tecnologia para transporte vertical foi rápido.

Em 1853, Otis inaugurou a primeira fábrica de elevadores e em 1857 foi instalado o primeiro elevador para transporte de passageiros no prédio *Haughwout & Company* em Nova Iorque, Estados Unidos. No ano 1870 é introduzido o elevador hidráulico e em 1889 a construção da *Torre Eiffel* em Paris foi um evento histórico que afiançou esta indústria. Na *Torre Eiffel,* os elevadores hidráulicos operavam entre o solo e o primeiro pavimento, duas cabinas contra-balanceadas operavam entre o segundo e o terceiro pavimento. Os elevadores hidráulicos funcionavam com água a pressão fornecida por maquinas de vapor. Neste caso, a movimentação da cabina é realizada através de um embolo comprido que se movimenta para acima quando a água é bombeada no interior do cilindro e para abaixo por gravidade quando a água é jogada fora através de uma válvula hidráulica. Nas primeiras instalações a válvula principal para controlar o fluxo de água era aberta ou fechada manualmente através de um sistema de cordas dispostas verticalmente na cabina. Anos depois foram incorporadas válvulas piloto para o controle de aceleração e desaceleração.

Em 1880, Werner von Siemens inovou, com o uso do motor elétrico, a construção de elevadores. No primeiro modelo, a cabina, que suporta o motor elétrico, sobe pelo poço através de um conjunto de engrenagens que tracionam o elevador nos suportes ao lado do poço. Em 1887 foi criado o modelo de elevador elétrico que faz girar uma polia na qual se enrola um cabo de aço onde são suspensos o contrapeso e a cabina. As vantagens do elevador elétrico, rendimento, baixo custo de instalação, e velocidade quase constante sem reparar na carga, popularizou o uso de elevadores para o transporte de passageiros.

Os primeiros elevadores brasileiros só começaram a ser fabricados em 1918. Não eram movidos nem a vapor, nem a eletricidade. O cabineiro era quem girando uma manivela fazia com que o elevador subisse ou descesse. As portas eram também abertas e fechadas manualmente. Com a explosão demográfica e a construção de edifícios mais altos o movimento da cabina foi substituído por sistemas elétricos que dispensavam o serviço do cabineiro. Atender chamadas com o apertar de um botão aperfeiçoou a eficiência do serviço, reles e circuitos elétricos foram desenvolvidos e aplicados no Brasil.

Depois da 2ª Guerra Mundial, se apresentaram significativos avanços em sistemas eletrônicos. Em 1948 foram instalados computadores para analise automática da informação, permitindo que os elevadores trabalhassem isoladamente melhorando significativamente o tráfego e o rendimento operativo dos elevadores em grandes edifícios.

Os elevadores elétricos são usados em todo tipo de edifícios, residenciais, comerciais, hospitais e edifícios de usos múltiplos. A *Torre Sears* em Chicago possui 110 andares e 109 elevadores com velocidades de até 549 m/min, o *World Trade Center*, New York, tinha 110 andares em cada torre e 244 elevadores com velocidade de 488 m/min. Hoje em dia a construção de prédios de grande altura tem foco no sudeste Asiático e na costa da Arábia impulsionada pelo desenvolvimento financeiro, sendo em *Taipei,* capital de Taiwan, onde foi construído o edifício mais alto do mundo, o *Edificio Taipei 101,* no ano 2004.

A construção de edifícios de grande altura não só é imposta pela tecnologia construtiva senão também pelas vantagens em termos de eficiência que o sistema de elevadores pode oferecer. Por outro lado as condições de segurança e métodos de evacuação em casos de emergência devem ser analisadas cuidadosamente. Existem projetos realizáveis como a *Milenium Tower* em Tokio, desenhada por Norman Foster (1989) que alcançará 840 metros ou a *Cidade Vertical: Torre Bíonica* desenhada por Javier Pioz e Maria Rosa Cervera que alcançará os 1228 metros (Cervera, 2006).

O mercado atual na indústria de elevadores é diversificado. Existem empresas especializadas em fabricação de cabinas, de quadros de comando, ou de alguns componentes específicos. Particularmente, as empresas que investem em inovação e tecnologia, possuem a maior parte do mercado mundial, entre elas estão, a *Otis Corporation, Kone Corporation, ThyssenKrupp* e *Atlas Schindler*. Aportes, novos modelos de controle e melhoras no sistema de elevadores são introduzidas por estas empresas, ampliando o mercado e tornando-se mais competitivas. As publicações científicas encontradas estão relacionadas com melhoras e novos modelos de controle.

A maioria das empresas na indústria de elevadores encontradas no Brasil dedicam-se à comercialização dos produtos ou à manutenção dos sistemas de transporte vertical, sendo da *ThyssenKrupp* (ThyssenKrupp, 2006) e *Otis Corporation* (Otis, 2006) a maior demanda

no mercado de elevadores. Empresas nacionais, como é o caso de *Sectron* (Sectron, 2006) e *Infolev* (Infolev, 2006), possuem uma presencia demarcada e competitiva no mercado nacional e internacional, participando na fabricação dos quadros de comando. Indústrias locais investem na atualização dos seus produtos aplicando métodos e tecnologias emergentes.

O desenvolvimento do elevador moderno tem produzido mudanças profundas na arquitetura e há suposto uma maior evolução das cidades no sentido de permitir a construção de prédios de maior altura.

2.2 - SISTEMAS DE ELEVADORES

Estudos feitos pela Associação Brasileira da Construção Industrializada (ABCI) indicam que a produtividade dos profissionais alocados em edifícios com maior grau de automação melhora de 9% a 10% (ABCI, 1993).

Os sistemas de elevadores são instalados em edifícios para atender as necessidades de transporte vertical. A potência consumida por um grupo de 12 elevadores em um edifício comercial, de 8 andares, e tráfego de 2500 pessoas por dia, representa 15% do consumo total (Thumm, 1995, Roger, 2005). Um projeto do sistema de elevadores que permita selecionar alternativas ótimas de operação ajudará a reduzir os custos e tempos envolvidos. Por outro lado, os usuários destas edificações precisam um transporte eficiente, robusto, e que ao mesmo tempo atenda as necessidades de conforto. Isto está relacionado com o decremento dos tempos de espera e locomoção, garantindo assim as condições de trabalho e as exigências de produtividade.

O funcionamento destes sistemas é geralmente especificado pela natureza da demanda do transporte do edifício, como no modelo apresentado em Kim *et al.* (1998). Um sistema que permita atender uma demanda de tráfego utilizando diferentes configurações permitirá aos projetistas desenvolver estudos baseados em especificações e simulações, analisando as propostas que forneçam melhor desempenho.

2.1.1 - Funcionamento do sistema de elevadores

Em edifícios modernos o transporte vertical de passageiros é implementado por um Sistema de Controle de Grupo de Elevadores (EGCS) e por alguns subsistemas microprocessados que implementam o Sistema de Controle Local (LCS) de cada elevador (Sklyarov *et al.*, 2003, Becker e Hartenstein, 2003, Muñozb *et al.*, 2006). Basicamente, um sistema de controle de grupo de elevadores é formado pela quádrupla <*h,c,e,g*>, onde *h* é o conjunto dos botões de chamadas de pavimento, *c* é o conjunto de botões de chamada de cabina, *e* é o conjunto de elevadores do edifício e *g* é o controle de grupo (Muñoz *et al.*, 2006). As tarefas do EGCS são diversas, começando pela recepção de chamadas de pavimento e de cabina, seleção do melhor elevador para atender uma chamada ou grupo de chamadas de pavimento, aquisição e análise de dados de tráfego do edifício e a aplicação de estratégias de operação em caso de incêndio, falha ou manutenção no prédio. O funcionamento do EGCS é o seguinte:

1) Um passageiro no andar *x* que deseja ir para outro andar *y* pressiona um botão de chamada de pavimento (subir ou descer).
2) O EGCS registra a chamada, envia um sinal luminoso no botão indicando ao usuário que a chamada foi registrada e seleciona um elevador para servi-lo.
3) A ação determinada pelo EGCS é transmitida ao sistema de controle local (LCS) do elevador escolhido. O elevador executa as tarefas pertinentes até chegar ao andar onde foi registrada a chamada de pavimento.
4) O usuário pode conferir o estado do elevador (posição e direção) no edifício através de sinalizadores em cada um dos andares.
5) Quando o elevador alcança a chamada de pavimento, as portas são abertas, o usuário entra e registra o andar *y* através dos botões de cabina, sendo este o andar desejado. As portas se fecham e o elevador se movimenta até o andar de destino.
6) Chegando ao andar de destino (*y*), o elevador reduz a marcha, abre a porta e espera o desembarque do passageiro.

Na figura 2.1 são apresentados os elementos envolvidos no sistema de elevadores. Os controles locais estão conectados a um barramento central. As botoeiras e sinalizadores em cada andar estão conectados a um outro barramento, sendo o EGCS quem administra os protocolos de comunicação.

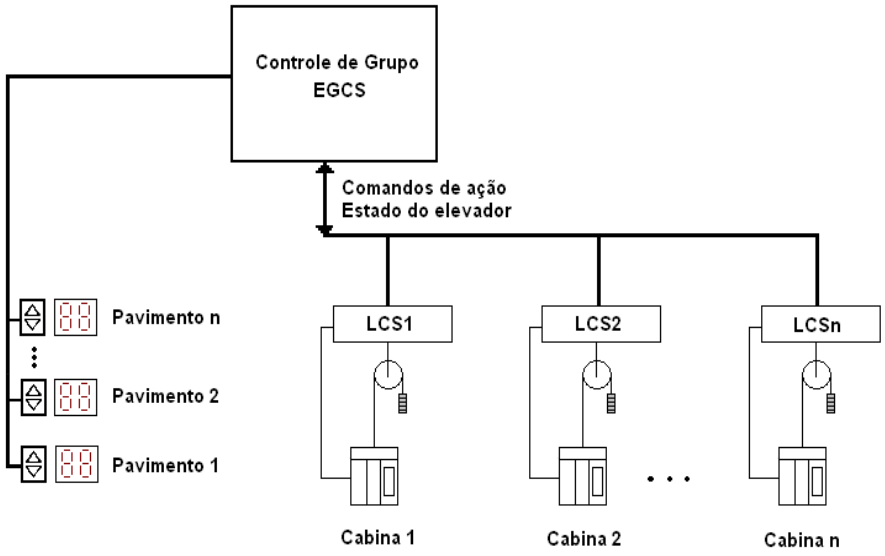


Figura 2.1: Elementos básicos em um sistema de elevadores.

Assim, a função do controle de grupo é supervisionar simultaneamente diversos elevadores garantindo que eles operem de maneira eficiente. Uma estrutura detalhada do controle local (LCS) de um elevador é mostrada na figura 2.2. O sistema de controle local executa as decisões que o controle de grupo tomou visando o melhor nível de serviço possível. Ditas decisões são enviadas em forma de *comandos de ação* e quando recebidos pelo controle local, são executadas diferentes tarefas, tais como acionar o "*driver* do motor", baseado na posição atual e a posição desejada, controle da porta, freio para travamento do movimento da cabina, controle velocidade e aceleração geralmente acionado por um inversor de freqüência com controle vetorial de fluxo. Sempre que mudar o estado do elevador deve ser enviado um *comando de estado* atual (Schneider, 2001, Bastidas, 1999).

Thumm (1995) e Sasaki *et al.* (1996) afirmam que um controle eficiente de um sistema de elevadores deve cumprir os seguintes requerimentos básicos:

a) Alta disponibilidade dos componentes;
b) Baixo custo;
c) Menor espaço consumido dentro do edifício;
d) Baixos custos de instalação e manutenção;
e) Maior conforto.

Entretanto, segundo estes requerimentos, os objetivos de um sistema de elevadores são:

- Minimizar o tempo de espera médio para os usuários;
- Minimizar o tempo médio de locomoção, isto é, o tempo gasto pelos passageiros desde

o momento que ingressam na cabina até chegar ao andar desejado;

- Reduzir o consumo de potência diminuindo o número de partidas dos atuadores (motores) e o tempo de locomoção;
- Maximizar a eficiência, isto é, maior quantidade de passageiros por unidade de tempo.

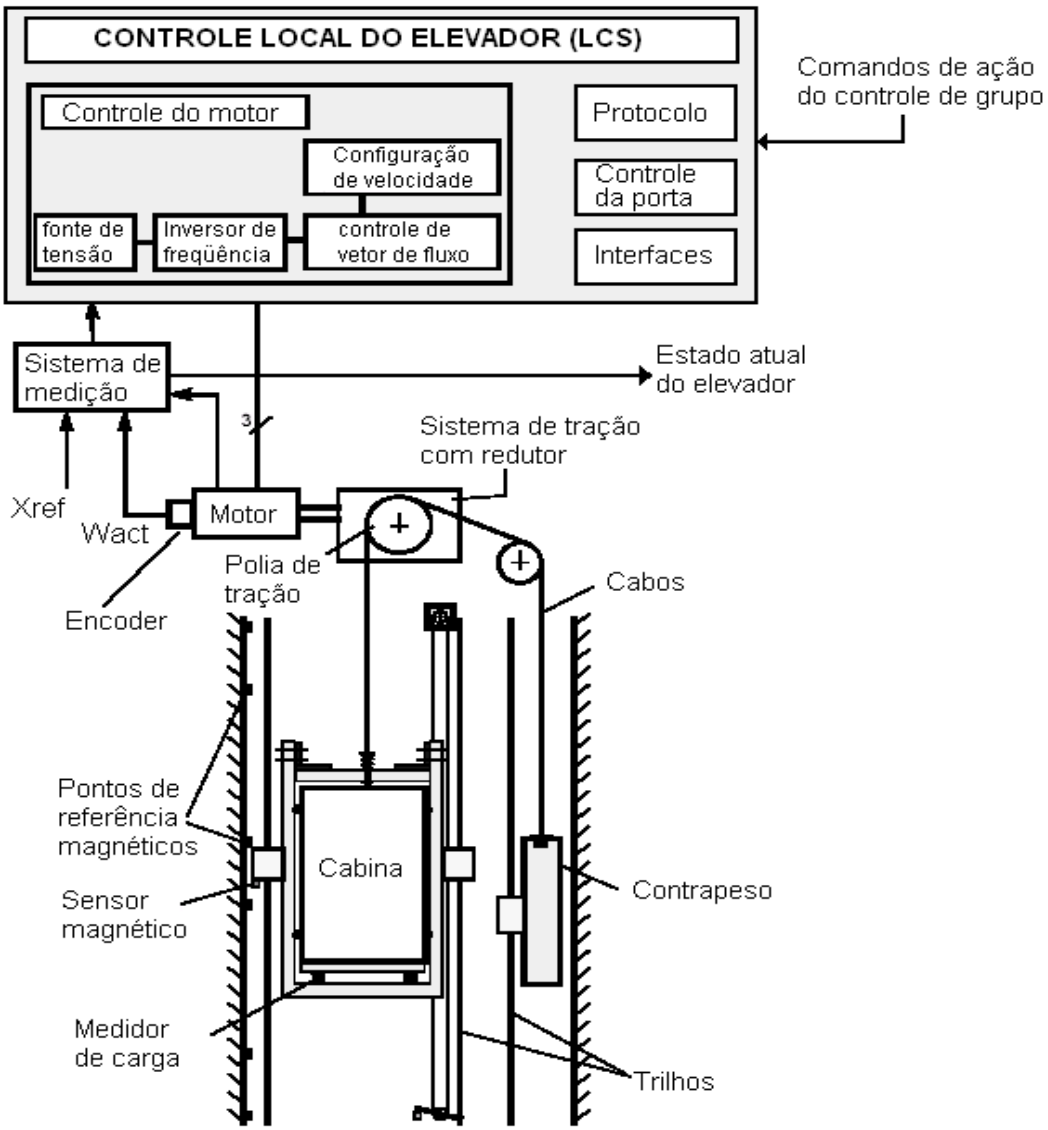


Figura 2.2: Estrutura de um elevador e controle local (modificado – Schneider, 2001).

Não sempre é possível cumprir simultaneamente os quatro objetivos. Como conseqüência, o conforto dos usuários demanda potência de consumo e por tanto, é necessário uma planificação de estratégias que permita satisfazer um dos critérios em determinadas circunstâncias que dependeram do tipo de edifício, do tráfego, e das necessidades dos passageiros em casos pouco usuais, como por exemplo, manutenção ou emergência.

No contexto deste trabalho, considera-se um enfoque nos mecanismos de controle de grupo e controle local para o atendimento eficiente de chamadas de pavimento dependendo do tráfego atual no edifício. Entretanto, as chamadas de cabina são tidas em conta na análise de distribuição de passageiros no edifício, porém, o atendimento delas é realizado segundo a chamada mais próxima no sentido de direção do carro.

2.2.2 - Classificação das edificações

O primeiro passo no planejamento do sistema de elevadores é a determinação do tipo e uso de edifício. Os parâmetros mais importantes para a classificação do edifício são o *Intervalo de tempo em horas pico de subida* (*I*) e a *Capacidade de transporte* (*HC*). O intervalo *I* é a média do tempo de espera dos usuários na entrada principal do edifício, em relação ao tempo médio de espera dos passageiros durante as horas pico de subida.

A Capacidade de Transporte *HC* é o número de passageiros que o sistema de elevadores transporta em cinco minutos durante as horas pico de subida (Siikonenb, 1997).

Segundo estes parâmetros as edificações são classificadas em quatro classes, a seguir:

- Prédios residenciais;
- Instalações de serviço público (metrô, centros comerciais, aeroportos);
- Hospitais e edifícios de usos diversos;
- Edificações comerciais de altura média, grande altura e arranha-céus.

Na figura 2.3 é mostrada a classificação dos tipos de edifícios em função dos valores do intervalo de tempo (*I*) e a capacidade de transporte (*HC*).

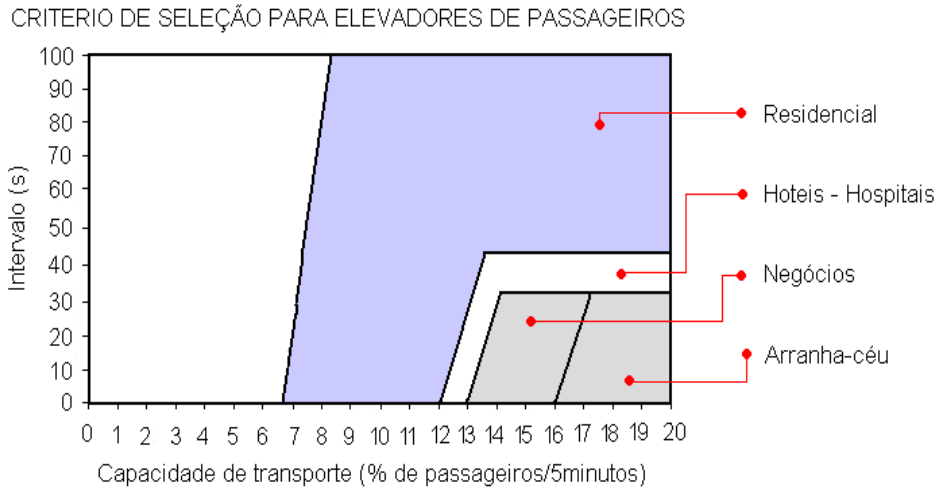


Figura 2.3: Critério de seleção para elevadores (modificado – Kone, 2006).

a) Prédios residenciais

Nos prédios residenciais a intensidade de tráfego é baixa. Os tempos de espera até duas vezes maior do que num prédio comercial são aceitáveis. Os elevadores para prédios residenciais podem ser escolhidos com normas ISO ou normas locais comparáveis. No Brasil existe a norma NBR10098 que padroniza elevadores elétricos destinados ao transporte de passageiros em edifícios residenciais, fixando dimensões e condições do projeto de construção necessárias à sua instalação (ABNT, 1987). Na figura 2.4 é mostrado o fluxo de tráfego habitual durante um dia num prédio residencial de 40 andares em Hong Kong. Os valores típicos de capacidade de transporte e intervalo de tempo são: *HC* = 5-7.5% / 5 min e *I* = 100 segundos.

b) Instalações de serviço público

Em aeroportos, os centros comerciais e outros lugares onde a densidade de tráfego é alta e o trajeto é de poucos andares, os elevadores são usados para transporte de mercancias ou para transportar pessoas portadoras de deficiências que podem locomover-se sem o auxílio de terceiros. Por tanto, nestes edifícios a capacidade de transporte das escadas elétricas é consideravelmente maior do que os elevadores.

c) Hospitais e edifícios de usos diversos

Nos hospitais, o sistema de elevadores precisa de um planejamento detalhado que deve considerar aspectos como a arquitetura e as necessidades especiais do edifício, por exemplo, transporte de emergências, serviços, camas, pacientes, visitantes e funcionários, além de fatores como a colocação das salas de operações e a unidade de cuidados intensivos que afeta diretamente as disposições do transporte. Os valores típicos de Capacidade de Transporte e de Intervalo de tempo são: *HC* = 11-13.5% / 5 min. Intervalo *I* = 40 segundos.

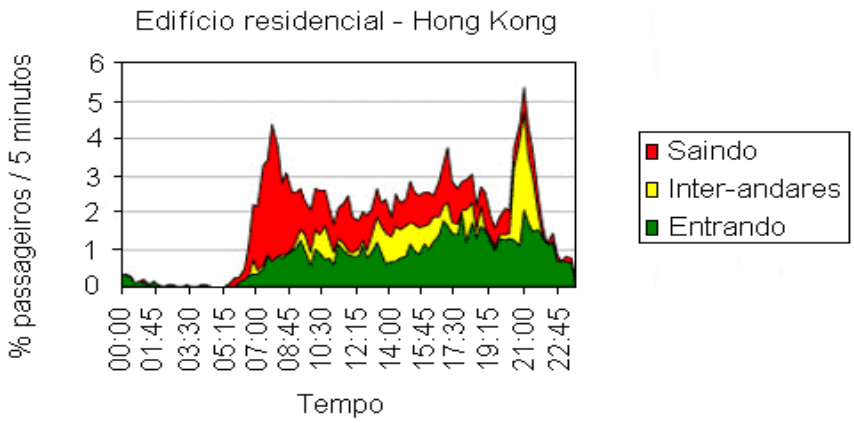


Figura 2.4: Fluxo de tráfego em edifícios residenciais (modificado – Siikonen, 1991).

d) Edificações Comerciais

- **Edifícios de altura média**: os edifícios comerciais de altura média são escritórios e hotéis. Nos hotéis a seleção do número de elevadores depende principalmente do número de quartos e camas. Geralmente são necessários elevadores adicionais para serviços diferentes ao transporte de passageiros. Em edifícios de escritórios existem três horas de tráfego intenso: tráfego de pico de subida nas horas de entrada pela manhã, de horas de comida e tráfego de pico de descida nas horas de saída no final da tarde. Geralmente o tráfego nas horas pico de subida (de manhã) é a referência para o planejamento do sistema de elevadores. Os Valores típicos de capacidade de Transporte e Intervalo de Tempo são: HC = 12.5–14% / 5 min e I = 30 segundos.
- **Edifícios de grande altura e arranha-céu**: em edifícios de grande altura, só um grupo de elevadores não satisfaz todas as necessidades de tráfego, por tanto distintos grupos de elevadores são divididos em zonas no edifício de forma que possam fornecer um serviço mais eficiente a todos os andares. Em edificações de 50 – 60 andares (50 a 200 metros), ou mais, são utilizados elevadores de cabina dupla ou blocos de elevadores um sobre outro viajando independentemente dentro de mesmo poço (ThyssenKruppb, 2006). Nos edifícios de grande altura o tráfego na hora da comida é comumente mais intenso do que o tráfego nas primeiras horas da manhã (vide figura 2.5), sendo a referência para o planejamento do sistema de elevadores em edificações de grande altura. Os valores típicos de capacidade de Transporte e intervalo de tempo são: HC = 15.5-17% / 5 min e I = 30 segundos.

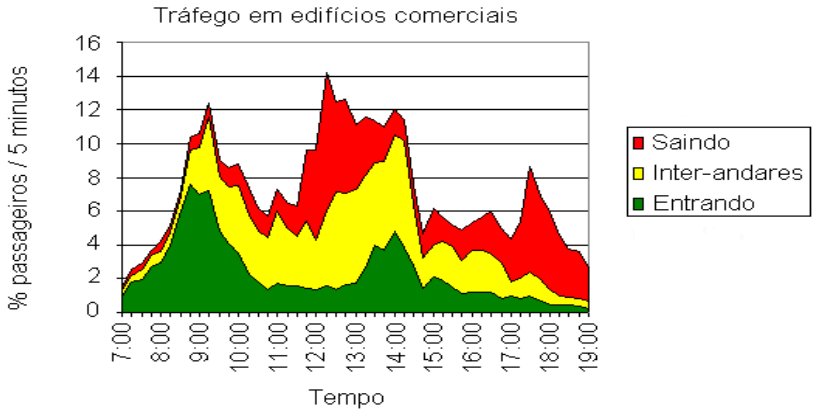


Figura 2.5: Fluxo de tráfego em edifícios comerciais (modificado – Siikonen, 1991).

As pesquisas sobre o tipo e uso do edifício são importantes no projeto de um sistema de elevadores, porém, os sistemas mais eficientes mantêm uma estimação constante do fluxo de tráfego durante as horas operacionais do prédio com o qual é possível aplicar distintas

estratégias de atendimento visando diminuir o tempo de espera dos usuários (Siikonenb, 1997). Estudos sobre a classificação do tráfego em edifícios visam melhorar a qualidade do serviço. Diferentes métodos têm sido aplicados, entre eles o uso da inteligência artificial (Zon e Yue, 2001, Kim *et al.*, 1998), entretanto, métodos de predição de tráfego são objeto de estudo nos últimos anos (Luo *et al.*, 2005). Por outro lado, estudos sobre estratégias de atendimento de chamadas são realizados desde começos dos anos 60 (Siikonenb, 1997, Sorsa *et al.*, 2003).

2.2.3 - Classificação do Tráfego

A classificação do tráfego é baseada nas características e a distribuição demográfica do edifício. Segundo esta classificação é possível adotar medidas que atendam cada um dos objetivos do sistema de elevadores por vez, mas que em forma geral, todos os objetivos sejam cumpridos durante a totalidade do tempo operacional do sistema.

Hoje em dia, as indústrias de elevadores que operam ao redor do mundo realizam um planejamento prévio à instalação do sistema de elevadores. Assim, pesquisas em torno da finalidade do prédio, estimação de número de andares, número de elevadores e fluxo de pessoas por dia são efetuadas.

Segundo a *Kone Corporation*, o planejamento de tráfego de um sistema de elevadores deve considerar os seguintes aspectos (Kone, 2006):

- Uma ótima distribuição e colocação dos aparelhos de transporte no edifício;
- Fácil acesso ao edifício e fluxo constante de pessoas e mercadorias;
- Sistema de controle para otimizar e sincronizar o fluxo de tráfego;
- Número e tipo correto de aparelhos de transporte;
- Tamanho e capacidade mais adequada dos aparelhos de transporte.

Os estudos sobre a determinação do tipo de tráfego têm sido realizados através de métodos diretos, como a utilização de câmeras (Kim e Monn, 2001, Haraguchi, 1990 *apud* Siikonen, 1997), métodos heurísticos baseados em lógica nebulosa (Tsuji e Amano, 1989, Siikonen, 1997) e redes neurais (So *et al.*, 1995). Um sistema de controle de grupo de elevadores (EGCS) deve considerar muitos fatores que afetam o estado atual e o estado futuro do sistema. Em este sentido, é possível atualizar dados da posição atual de cada elevador e do armazenamento de chamadas de pavimento e cabina. Entretanto, é difícil

conhecer informação sobre o numero de passageiros esperando nos pavimentos onde aconteceram as chamadas, além disto, informações futuras sobre chamadas de pavimento e cabina são incertas (Chenais e Weinberger, 1992, Kim *et al.*, 1998).

Geralmente, em edifícios comerciais, as mesmas características de tráfego se apresentam em horários similares e o mesmo numero de vezes por dia. Portanto, o tempo é um fator de importância na classificação do tráfego, porém, a maior peça de informação é o numero de chamadas de subida, de descida e de cabina (Siikonen, 1997, Kim *et al.*, 1998). A figura 2.6 mostra um padrão de tráfego de subida e descida típico de um edifício comercial. O número de chamadas de subida representa o tráfego de subida (TS) e o número de chamadas de descida representa o tráfego de descida (TD). A partir destes gráficos, característicos em cada edifício, é possível determinar as estratégias que o controle de grupo deve efetuar.

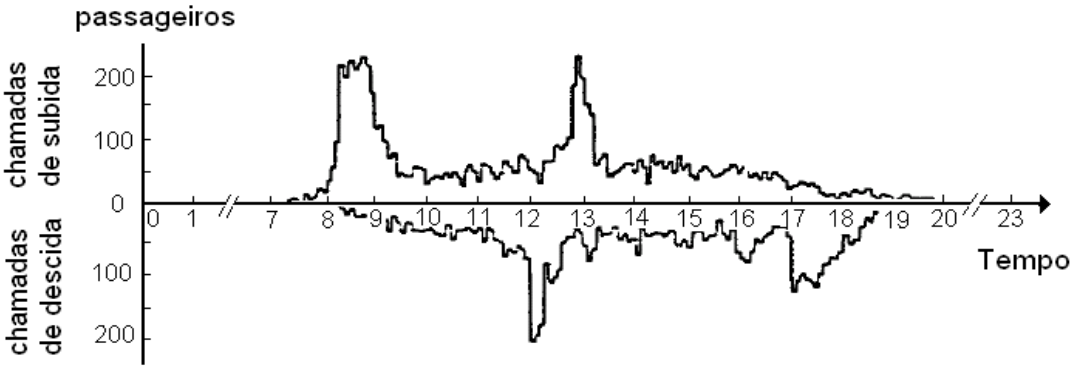


Figura 2.6: Tráfego de subida e descida em edifícios comerciais (modificado – Tsuji e Amano, 1989).

Na tabela 2.1 são apresentadas cinco principais características no tráfego de passageiros em um edifício a partir das quais é possível identificar um comportamento particular no transporte vertical.

Em Kim *et al.* (1998), destaca-se uma classificação em oito modos de tráfego. Esta classificação depende das características encontradas em um momento determinado. A tabela 2.2 resume a descrição para cada um dos oito padrões.

No contexto deste trabalho, distintos padrões de tráfego para o transporte vertical foram estudados, porém, será implementada a classificação de Kim *et al.* (1998), pela eficiência e flexibilidade de atender as características encontradas em diferentes tipos de edifícios.

Tabela 2.1: Características do tráfego de passageiros.

Característica	Definição
Tráfego de Subida (TS)	Número de chamadas de subida. É uma medida feita a partir dos botões subir em cada pavimento.
Tráfego de Descida (TD)	Número de chamadas de descida. É uma medida feita a partir dos botões descer em cada pavimento.
Porcentagem de Tráfego Centralizado (CITP)	Valor percentual da relação do numero de passageiros entrando no andar mais lotado respeito aos demais andares. É uma medida feita a partir das chamadas de cabina e indica qual dos andares é mais solicitado.
Porcentagem de Tráfego Distribuído (DOTP)	Valor percentual da relação do numero de passageiros saindo de todos os andares excepto o mais lotado, respeito os demais andares. É uma medida feita a partir das chamadas de cabina.
Tempo	Indica a hora atual em minutos e segundos.

Tabela 2.2: Padrões de tráfego de passageiros.

Padrão	Descrição / Tempo
(Up) Pico de subida	Passageiros entrando no edifício. Começo da jornada de trabalho.
(BT) Tempo de negócios	O tráfego total é médio. Antes e depois do meio dia.
(LTa) Tempo de almoço A	Passageiros indo ao restaurante. Começo da hora de almoço.
(LTb) Tempo de almoço B	Passageiros voltado aos escritórios. Fim da hora de almoço.
(BTH) Negócios e pesado.	O tráfego total é pesado. Qualquer hora.
(HT) Pesado	Passageiros saíndo e entrando em andares internos. Qualquer hora.
(Down) Pico de descida	Passageiros saíndo do edifício. Fim da jornada de trabalho.
(IT) Tempo inativo	O tráfego total é leve. Durante a noite.

2.2.4 - Estratégias de escalonamento

Os métodos para encontrar o escalonamento mais otimizado no transporte vertical é foco de pesquisa desde os anos 1960. Em um grupo de elevadores encontram-se n elevadores atendendo p chamadas de pavimento; se todas as combinações concernentes às chamadas de pavimento fossem calculadas, para encontrar a combinação de custo mínimo existe um grado de complexidade elevado. O problema radica em que o número de combinações cresce exponencialmente com p e, portanto, o número de possíveis casos é:

$$C = n^p \tag{2.1}$$

De forma geral, se n passageiros nos pavimentos devem ser atendidos por um único elevador, haverá $n!$ caminhos para atender-os. Se todos os caminhos são permitidos e são

levados em conta pelo EGCS o problema de atendimento é considerado um problema tipo *NP-hard,* isto para um único elevador (Pepyne e Cassandras, 1997, Nikovsky e Brand, 2003). Este problema é similar ao problema do *caixeiro viajante* considerado um problema tipo *DARP* (*Dial a Ride Problem*), no qual alguns objetos devem ser movimentados através de fontes de destinos numa rede de transporte. O sistema escalonador de elevadores é um caso especial aonde a rede de transporte forma conexões paralelas (minhoca) na qual também apresentam-se problemas *NP-hard* (Hauptmeirer *et al.*, 2001).

Nos primeiros modelos, as chamadas de pavimento e o atendimento por diferentes cabinas formam uma árvore de decisão, assim, para uma nova chamada é possível calcular o custo do serviço através da avaliação de uma função de custo. Finalmente o custo mínimo define o caminho ótimo para o atendimento das chamadas (Siikonen, 1993). Porém, a demanda do tempo de calculo é grande o que pode atrapalhar a eficiência do controle, por exemplo, usando um processador Intel Pentium 250 MHz, foram necessários 500 ms para encontrar a melhor solução no caso de três elevadores recebendo dez chamadas de pavimento. Em um edifício de oito elevadores e 60 andares o número de possíveis combinações é maior a $6.2*10^{57}$ (Siikonenb, 1997).

Distintos métodos visando diminuir o tempo de cálculo têm sido propostos. Em Closs (1970) *apud* Bastidas (1999), foi implementado o algoritmo ACA (*Adaptive Call Allocation*) no qual os elevadores são reservados em diferentes situações de atendimento (caminhos possíveis). Desta forma, apenas as novas chamadas são transmitidas ao melhor elevador. Com este método o tempo de calculo é diminuído consideravelmente, porém, as assinações de carro antigas perdem eficiência quando um novo caminho, similar ao anterior, é registrado a um carro.

O algoritmo OR (*Optimal Routing*) introduzido em Siikonen (1989) propõe uma otimização dinâmica através do controle coletivo no qual é procurada a chamada de pavimento mais próxima à cabina que se movimenta na mesma direção. Neste caso são consideradas chamadas de pavimento com dois botões, subir e descer.

Recentemente, os algoritmos genéticos (Tobita *et al.*, 1996, Fujino *et al.*, 1997, Sorsa *et al.*, 2003) e otimização dinâmica com aprendizado por reforço (Yuan-Wei e Li-Chen, 2000) têm sido estudados. As simulações numéricas para diferentes tipos de tráfego

utilizando algoritmos genéticos mostram uma otimização do serviço de elevadores entre 12 a 20% e um aumento da capacidade de transporte de 20%; ao mesmo tempo a exigência computacional do registro de de chamadas foi reduzido. No caso de algoritmos com aprendizado reforçado a otimização do serviço depende do tempo computacional ou do tempo de treinamento dos algoritmos. Nikovsky e Brand (2003), apresentam a dinâmica de um sistema de escalonamento de elevadores baseado em *cadeias de Markov* onde as simulações computacionais mostram uma redução do tempo de espera dos usuários até de 70% em edificações entre 8 e 30 andares e um grupo de 2 até 8 elevadores, porém, os algoritmos possuem um custo computacional maior do que os algoritmos convencionais.

As formulações anteriores são aplicáveis no suposto de que a velocidade dos elevadores alcance o seu maior valor em uma distância não maior do que a metade entre dos andares vizinhos, isto é, que o motor do elevador seja capaz de acelerar ao máximo e frear completamente durante a viagem entre dois andares vizinhos. Esta condição não é cumprida em elevadores modernos nos quais a velocidade máxima supera os 400m/min precisando pelo menos 3 ou 4 andares para acelerar e frear completamente. Algoritmos que utilizam uma quantidade considerável de cálculos computacionais que podem comprometer a eficiência no caso de elevadores de alta velocidade.

Como foi mencionado, o funcionamento do sistema de elevadores é geralmente determinado pela natureza da demanda de transporte do edifício, além de considerar fatores como futuras chamadas e fluxo de passageiros em determinado tempo. Estes fatores justificam o desenho de um controle flexível que permita a possibilidade de mudar dinamicamente a estratégia de atendimento.

2.2.5 - Princípios Coletivo e Seletivo

Os algoritmos de atendimento mais conhecidos são baseados nos princípios coletivo e seletivo, eles são controles automáticos caracterizados por existirem botões de chamadas internas para indicar o andar de destino (um para cada andar) e um único botão de chamada instalado em cada andar todos conectados ao controle local de forma que as chamadas fiquem nele registradas (Nikovsky e Brand, 2003).

No *princípio coletivo* o elevador efetua as paradas seqüencialmente procurando a chamada de pavimento mais próxima e na mesma direção do movimento, sem importar a ordem em

que as chamadas foram feitas (Strakosch, 1998 *apud* Nikovsky, 2003). Geralmente o principio de atendimento coletivo resulta em problemas de redundâncias (*bunching problem*), no qual múltiplos elevadores chegam no mesmo andar ao mesmo tempo.

No *princípio seletivo* o elevador seleciona o mais alto ou mais baixo pavimento que realizou uma chamada definindo qual será o próximo andar a ser visitado em um momento dado. No contexto do presente projeto, cinco estratégias baseadas segundo os princípios de atendimento coletivo e seletivo foram implementados.

Todos os algoritmos fazem referência às chamadas de pavimento, sendo que as chamadas internas (ou chamadas de cabina) sempre são atendidas segundo o princípio coletivo. Na definição dos primeiros quatro algoritmos consideram-se edifícios com um único botão de chamada em cada andar. O último algoritmo é aplicado em edifícios nos quais existem dois botões (subir e descer) nos andares intermediários permitindo ao usuário selecionar a direção de movimento.

2.2.6 - Algoritmos básicos de escalonamento

Os algoritmos básicos tratados neste trabalho são os seguintes:

a) **Algoritmo Coletivo de subida / Coletivo de descida (C):** este algoritmo implementa o principio coletivo em ambas as direções. O elevador efetua as paradas na seqüência de andares estabelecida pelas chamadas de pavimento durante a viagem de subida e de descida. Isto é feito calculando o andar mais próximo na mesma direção de movimentação do elevador, sem importar a ordem em que as chamadas foram realizadas. A figura 2.7 (a) mostra a forma de atendimento de andares durante a viagem de subida e descida em um edifício de 20 andares.

b) **Algoritmo Coletivo de subida / Seletivo de descida (U):** aplica o principio coletivo durante a viagem de subida. Na viagem de descida, seleciona o andar mais baixo que realizou uma chamada de pavimento, começando de novo a viagem de subida de forma coletiva. Aplica-se em caso de tráfego de pico de subida, transportando pessoas dos andares mais baixos até os andares mais altos. A figura 2.7 (b) mostra o comportamento deste algoritmo.

c) **Algoritmo Seletivo subida / Coletivo descida (D):** na viagem de subida o sistema escolhe o chamada de pavimento no andar mais alto (processo seletivo). Em seguida as chamadas são atendidas de forma seqüencial (processo coletivo) calculando a chamada de pavimento mais próxima ao andar atual na mesma direção de movimento, como é mostrado na figura 2.7 (c). Este algoritmo é aplicado em caso de tráfego de pico de descida, pois a função principal é transportar pessoas dos andares mais altos até os primeiros andares incluindo o térreo.

d) **Algoritmo Seletivo subida / Seletivo descida (S):** a figura 2.7 (d) mostra o comportamento deste algoritmo. Na viagem de subida é escolhido o andar mais alto que realizou uma chamada de pavimento, visitando-o primeiro. No passo seguinte é atendida a chamada de pavimento mais baixa, lembrando que as chamadas de cabina sempre são atendidas de forma coletiva. O objetivo deste algoritmo é transportar passageiros aos andares centrais no edifício, porém, a eficiência no transporte é baixa devido aos altos tempos de espera dos usuários e o alto consumo de potência.

e) **Algoritmo Coletivo - Seletivo / subida – descida (CS-ud):** na definição deste algoritmo, consideram-se edifícios com dois botões de seleção de chamada em cada andar para a seleção da direção de movimento. As chamadas de pavimento para subir são selecionadas separadamente das chamadas de pavimento para descer, sendo atendidas primeiramente todas as chamadas registradas em um dos sentidos para depois serem atendidas as de sentido oposto. O funcionamento é o seguinte: Inicialmente, na viagem de subida, as chamadas de pavimento para subir são atendidas segundo o principio coletivo e antes de começar a viagem de descida é escolhida a chamada de pavimento para descer mais alta, começando o processo coletivo de descida e atendendo seqüencialmente as chamadas de pavimento para descer. Finalmente, é escolhida a chamada de pavimento para subir mais baixa começando novamente a viagem coletiva na subida (figura 2.7 (e)). Este algoritmo é importante em casos de tráfego pesado ou quando o tráfego é intenso e equivalente nos dois sentidos.

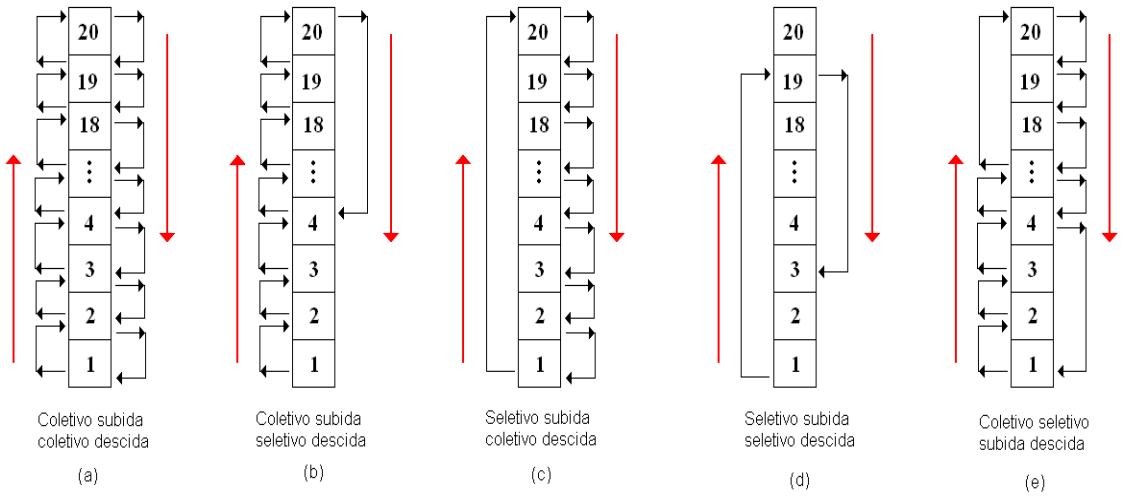


Figura 2.7: Comportamento dos algoritmos de escalonamento.

No âmbito deste trabalho, considera-se que a finalidade do planejamento de estratégias de escalonamento em um grupo de elevadores, envolve o estabelecimento de prioridades, entre o atendimento eficaz de passageiros e o consumo de potência, para diferentes situações de tráfego e características do edifício. Este planejamento esta baseado no contexto de um sistema de controle para otimização e sincronização do fluxo de tráfego no edifício.

2.3 - *HARDWARE* RECONFIGURÁVEL

Uma das motivações do presente trabalho é a avaliação do uso de dispositivos reconfiguráveis nas áreas de automação predial. Dada a alternativa plausível de implementar algoritmos diretamente em *hardware* usando estas tecnologias, existe a possibilidade de validar os ganhos em desempenho, custo, flexibilidade, entre outras. FPGAs permitem explorar o paralelismo intrínseco dos algoritmos obtendo ganhos claros em desempenho se comparados com implementações em *software* (por exemplo, usando microcontroladores). A flexibilidade encontrada nos dispositivos baseados em *hardware* reconfigurável fornece uma alternativa para a implementação das técnicas de controle relacionadas ao sistema de elevadores.

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 os 200 dólares americanos. A

complexidade dos microprocessadores, medida segundo o número de transistores dentro do *chip*, é dobrada cada 18 meses desde a aparição do 4004 (Moore, 1997). Com a evolução da tecnologia os circuitos integrados (*chips*), atualmente conseguem integrar 20 milhões de transistores por 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 2.8.

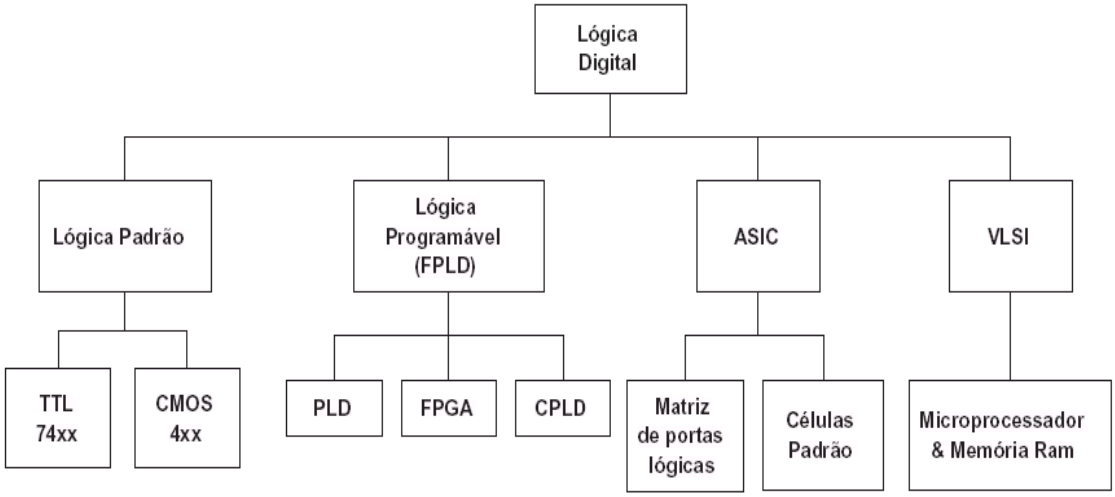


Figura 2.8: Circuitos integrados (modificado – Hamblen e Furman, 2001).

Os circuitos integrados tradicionais, TTL e CMOS, executam uma lógica padrão realizando operações pré-definidas pelo fabricante, portanto, o usuário deve conectar diferentes tipos de circuitos para executar uma lógica específica, trazendo inconvenientes como perda de desempenho, aumento da área e alto consumo de energia.

Os circuitos integrados, tais como PLD (*Programmable Logic Device*), CPLD (*Complex Programmable Logic Device*) e FPGA (*Field Programmable Gate Array*) possuem operações funcionais internas definidas pelo usuário, entretanto circuitos como ASIC (*Application Specific Integrated Circuits*) e VLSI (*Very Large Scale Integration*) permitem ao usuário determinar a lógica, porém, são implementados por fabricantes especializados utilizando alta tecnologia no processo, diminuindo o consumo de potência e aumentando o desempenho. Este é o caso dos microprocessadores e memórias RAM utilizadas em PCs, em contra partida, o custo da implementação é elevado e a funcionalidade é fixa, portanto, não é possível fazer *upgrades* ou modificar a lógica que vai ser executada.

Na figura 2.9 encontra-se uma relação do desempenho dos circuitos integrados em função

do custo e tempo de desenvolvimento.

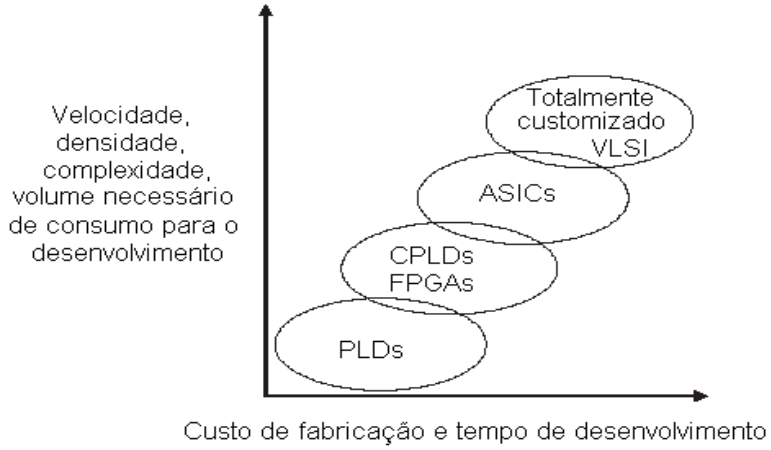


Figura 2.9: Relação do desempenho e o mercado na lógica digital.

A implementação de circuitos integrados do tipo ASIC demanda tempo (alguns meses) de fabricação e desenho, elevando o custo de desenvolvimento, sendo justificada a inversão para grande quantidade de circuitos integrados. Os usuários devem realizar testes de verificação e melhoras do desempenho antes do processo de manufatura em massa; erros de desenho podem aumentar o tempo e custo final. Isto faz com que empresas especializadas em fabricação de circuitos integrados utilizem FPGAs na fase de projeto e verificação, pois elas são programáveis pelo usuário (Hutton *et al.*, 2006).

Nas tecnologias ASIC e VLSI, as conexões entre os blocos lógicos são fixas (*hardwired*) e não programáveis o que permite uma velocidade de processamento algumas vezes maior que no caso de CPLDs e FPGAs. Ao mesmo tempo a área no circuito diminui e o custo de fabricação pode baixar. Entretanto, o custo de engenharia para o desenho dos ASIC e VLSI é consideravelmente alto comparado com o custo e a flexibilidade encontrados nas tecnologias CLPD e FPGA.

2.3.1 - Dispositivos Lógicos Programáveis

Os PLDs (*Programmable Logic Device*) são circuitos integrados de propósito geral que permitem a implementação da lógica definida pelo usuário. O PLD contém elementos lógicos, portas e chaves, programáveis. As chaves programáveis permitem conectar as portas lógicas internas para implementar o circuito lógico desejado. O primeiro tipo de PLD foi desenvolvido pela Philips, nos anos 80, e é baseado na expressão das funções lógicas na forma de uma equação Booleana de somas e produtos. Assim, o PLA

(*Programmable Logic Array*) contém uma matriz de portas AND (produtos) e portas OR (somas), sendo que a saída das portas AND são conectadas a um conjunto de portas OR e as conexões entre as portas lógicas são configuradas pelas chaves programáveis, implementando a função lógica desejada.

Os PLAs possuem um desempenho baixo devido ao atraso de propagação nas portas lógicas. Para solucionar este problema, foi introduzido um dispositivo similar chamado PAL (*Programmable Array Logic*), no qual as portas no plano AND são programáveis e as portas no plano OR ficam fixas, tornando o circuito mais simples de fabricar e diminuindo custos. Na maioria de PLDs a saída das portas OR é conectada a um *flip-flop* e depois realimentado como entrada no plano AND, permitindo ao PLD implementar maquinas de estado simples melhorando o desempenho (Hamblen e Furman, 2001, Bonato, 2004).

2.3.2 - FPGAs (*Field Programmable Gate Array*)

Internamente as FPGAs contém copias do mesmo elemento lógico (LB) básico organizados matricialmente. Cada bloco lógico contém um número pequeno de entradas e uma saída e 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 LTU (*LookUp Table*) que, através do controle de um grupo de multiplexadores e portas, permite o fluxo de dados desde as células de armazenamento até a saída do bloco lógico implementando a função lógica desejada.

O arranjo de células dentro da LTU é utilizado para armazenar a tabela de verdade da função lógica. A figura 2.10 (a) mostra a estrutura de um bloco lógico da FPGA StratixII da *Altera* (Hutton e Furman, 2006. A figura 2.10 (b) mostra um exemplo de LTU de duas entradas e uma saída.

Para realizar operações complexas, os elementos lógicos podem ser conectados uns com outros através de chaves de interconexão programáveis. Na figura 2.11 (a) é apresentada a estrutura geral dos FPGAs, contendo três tipos de recursos, a seguir: blocos de I/O, utilizados para realizar a interface com os pinos de entrada e saída do dispositivo, blocos lógicos para implementação de funções através da programação das células de armazenamento, e a rede de interconexão para roteamento horizontal e vertical entre as linhas e colunas dos blocos lógicos.

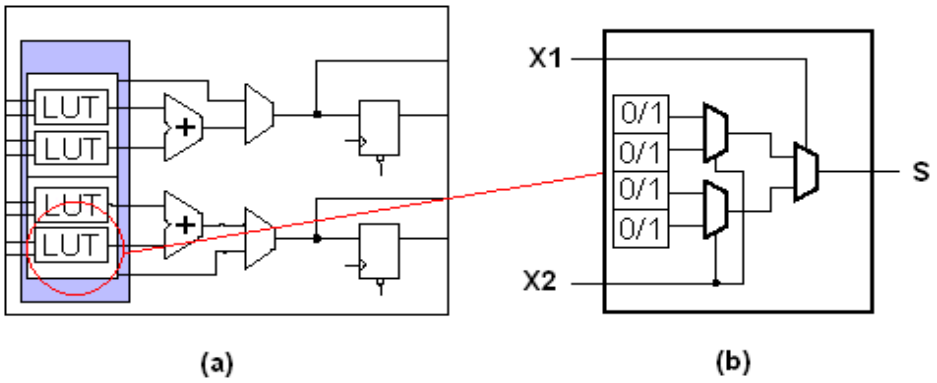


Figura 2.10: (a) Bloco lógico Altera StratixII (b) LTU de duas entradas e uma saída.

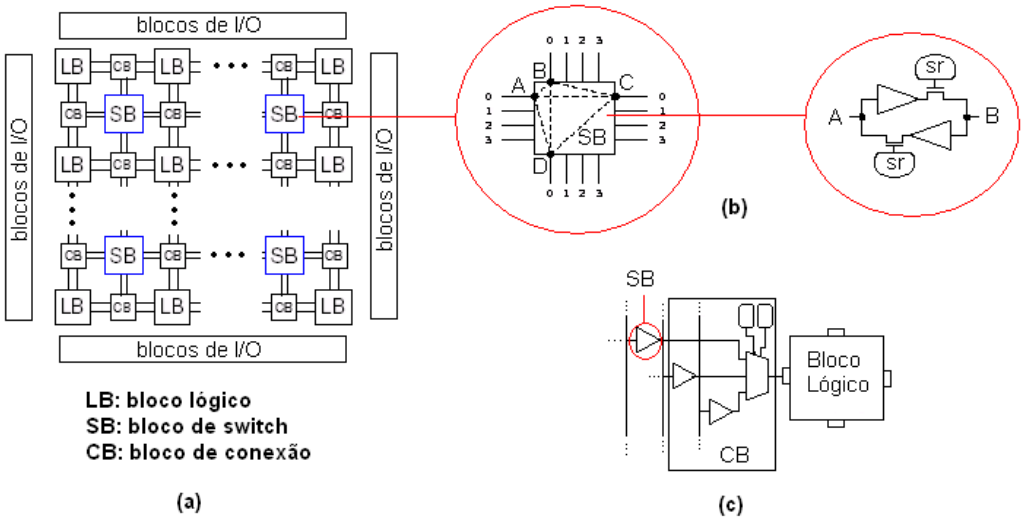


Figura 2.11: (a) Estrutura geral da FPGA (b) bloco de switch (c) bloco de conexão.

A figura 2.11 (b) mostra a estrutura dos blocos de switch (*switch block*), utilizados para conectar blocos lógicos não vizinhos. Entretanto, a figura 2.11 (c) mostra a estrutura dos blocos de conexão (*connection block*), que comunicam elementos lógicos vizinhos.

Nas FPGAs de última geração os canais de roteamento entre os componentes podem existir em planos diferentes. Na figura 2.11 é mostrada a estrutura de uma 2D-FPGA, na qual os blocos de *switch* estão num plano diferente aos blocos de conexão e blocos lógicos. Em outras FPGAs utilizam-se três planos (3D) o que permite um roteamento mais eficiente, tempos de propagação menores entre os elementos lógicos e diminuição do consumo de potência (Mingjie *et al.*, 2006, Yan Lin *et al.*, 2005).

A capacidade das FPGAs é definido pelo número de elementos lógicos ou LTUs, pois esta medida é considerada mais tangível que a medida expressa através de portas lógicas

(Hamblen e Furman, 2001, Altera, 2003). Conforme a figura 2.12 é constatada que a evolução desta tecnologia, em termos de capacidade e desempenho, tem crescido de forma considerável nos últimos 8 anos (Taghavi *et al.*, 2004).

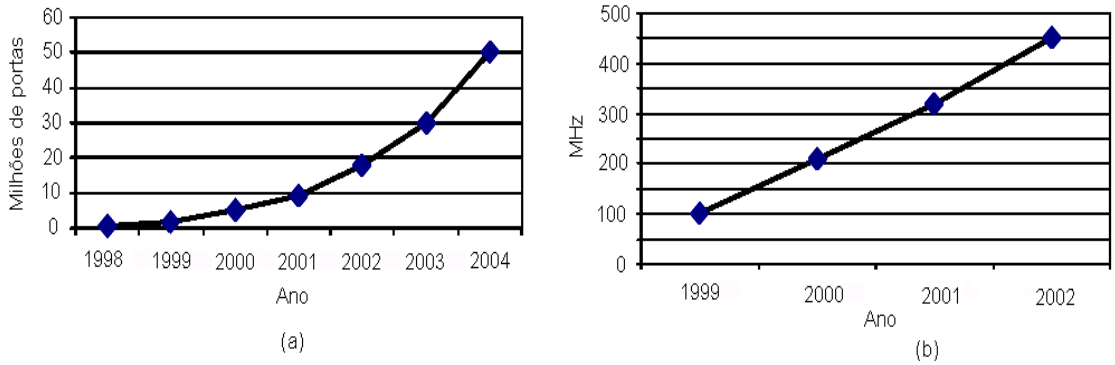


Figura 2.12: Evolução das FPGA: (a) Evolução em gates (b) Evolução em desempenho

O mercado de fabricação de dispositivos FPGA é dominado pelas empresas *Altera Corporation* (Altera, 2003) e *Xilinx Corporation* (Xilinx, 2005), embora existam outros fabricantes como *Actel Corporation* (Actel, 2006), *Cypress Semiconductor* (Cypress, 2006), *QuickLogic Corporation* (QuickLogic, 2006) e *Lattice/Vantis Corporation* (Latice, 2006).

O campo de aplicação das FPGAs é diverso, encontrando-se aplicações em indústria automotiva, redes de trabalho e telecomunicações, processamento de imagens, dispositivos médicos, entre outros produtos de consumo (Taghavi *et al.*, 2004). Uma aplicação em particular das FPGAs, que é caso de estudo atual, é a possibilidade de reconfiguração dinâmica, o que visa aplicações revolucionárias no futuro próximo em área de aplicação em telefonia celular, indústria automotiva e microprocessadores em geral (Harteinsten, 2002).

No contexto do presente trabalho, as implementações foram realizadas na FPGA Spartan3 2E, família mais recente do grupo Spartan da *Xilinx*, pois as suas características de desempenho e capacidade são adequados para o tipo de aplicação desejada. As placas baseadas na Spartan3 (Digilent, 2006) possuem as interfaces necessárias para a implementação do controle de escalonamento de elevadores. Uma ampliação da plataforma de trabalho é apresentada no capitulo 3.

Em termos gerais, as aplicações no mercado dos semicondutores mantêm uma tendência

regular mudando de enfoque cada vez que uma nova tecnologia aparece no mercado. Esta apreciação foi introduzida pela primeira vez por Tsugio Makimoto, no ano 1989 e esta representada na figura 2.13 (Makimoto, 2002).

Quando uma quantidade acentuada de novas arquiteturas, *software* e dispositivos aparecem, a indústria dos semicondutores sofre uma tendência de normalização surgindo a necessidade de diferenciar produtos através de valores agregados. Por outro lado, os progressos em desenho, automatização e avanços tecnológicos produzem uma tendência de estabelecimento do mercado na indústria dos semicondutores (Makimoto, 2002). Curiosamente aquelas mudanças nas tendências tecnológicas aconteceram de 10 em 10 anos, como mostrado na figura 2.13.

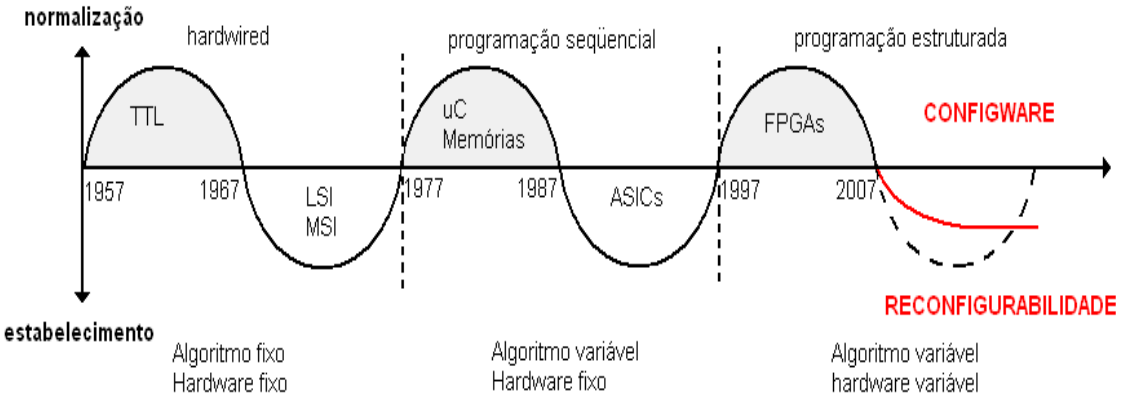


Figura 2.13: Onda de Makimoto (modificado – Makimoto, 2002).

Hoje em dia, 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 embebidos (*Embedded Systems*) e computação reconfigurável (*Configware*) fazendo de lado o tradicional *hardware* fixo (Harteinsten, 2000, Configware, 2005).

Os microcontroladores encontrados no mercado possuem uma arquitetura convencional tipo von Neumann, isto é, uma unidade lógica aritmética (ULA) executando uma única instrução à vez dentro de um conjunto de instruções, armazenando e lendo dados numa RAM, no qual um contador de programa controla o fluxo de instruções, como é mostrado na figura 2.14. Este tipo de arquitetura foi formulada por John von Neumann nos anos 1960 e constituiu durante anos o dispositivo computacional mais flexível sobre o qual diversas aplicações têm sido realizadas (Harteinsten, 2004).

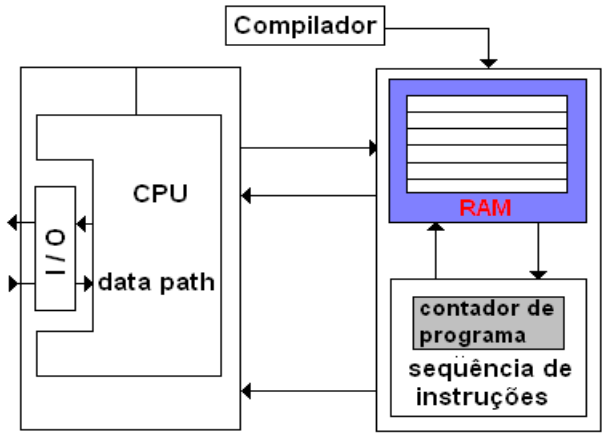


Figura 2.14: Arquitetura von Neumann.

A evolução de novas tecnologias nos processos de fabricação dos circuitos integrados permitiu que a complexidade e o desempenho aumentassem. No mercado são encontrados microcontroladores de alto desempenho que permitem uma velocidade maior no processamento de dados, calculo de operações e execução de instruções. Embora, a evolução da indústria de memórias de acesso aleatório (RAM) tenha crescido consideravelmente em desempenho e capacidade de armazenamento, a velocidade de funcional das memórias, isto é, a velocidade com que são escritos ou lidos os dados na memória, não consegue acompanhar a velocidade com que são processados os dados na unidade central.

A figura 2.15 mostra claramente a diferencia do desempenho dos processadores em relação às memórias durante os últimos 20 anos. Esta diferencia em desempenho, forma hoje em dia um dos maiores inconvenientes das arquiteturas convencionais e é conhecido como "gargalho de von Neumann" (Harteinsten, 2004). Em aplicações onde são necessários cálculos complexos é comum encontrar um conjunto de processadores atuando de forma paralela com o objetivo de aumentar a eficiência e diminuir o tempo de cálculo.

As arquiteturas modernas, como os CLPDs ou FPGAs, possibilitam a configuração dos seus blocos lógicos executando funções lógicas através de um fluxo de dados, sem a necessidade de ler seqüências de instruções. Por outro lado, não dependem da velocidade com que podem ser lidos ou escritos os dados dentro de uma memória. Neste novo modelo, as instruções não são mais controladas pelo contador de programa, senão que existe um fluxo de dados controlado por seqüenciadores e contadores de dados.

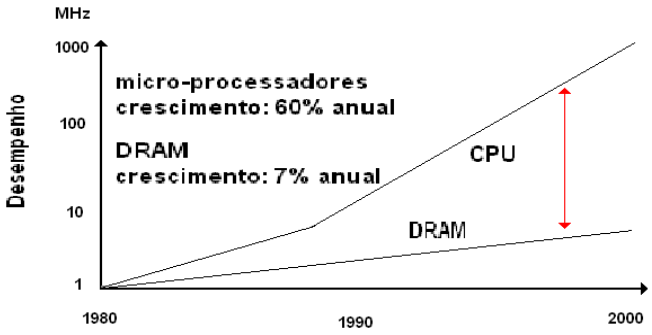


Figura 2.15: Paradigma de von Neumann (modificado – Harteinsten, 2002)

A grande vantagem desta arquitetura é que os blocos lógicos podem ser configurados para permitir o fluxo de dados de forma independente e paralela, além de permitir a reconfiguração dos mesmos em tempo de execução. Isto possibilita a execução, em uma única pastilha, de diferentes processos concorrentes, ampliando o espectro de aplicações das FPGAs. Atualmente são utilizadas ferramentas de CAD (*Computer Aided Design*) para reconfigurar os elementos lógicos que constituem a FPGA.

As empresas fabricantes de FPGAs possuem dispositivos para diferentes tipos de aplicações. Empresas como a *Altera* e *Xilinx* fornecem aportes, IP cores, e ferramentas de importante ajuda na elaboração de projetos, por outro lado, os desenvolvimentos de aplicações mostram um aumento constante de publicações cientificas (Harteinsten, 2000). Os dispositivos FPGA fornecidos pelas empresas *Altera* e *Xilinx* são classificados em famílias, conforme as suas principais características, como mostrado nas tabelas 2.3 e 2.4.

Tabela 2.3: Resumo das famílias e características das FPGA da *Altera.*

Família	FPGA	Elementos lógicos	Total de RAM bits	Blocos de DSP	PLL	Pinos de I/O para uso geral
Stratix	EP1S10	10.570	920.448	6	6	426
	EP1S80	79.040	7.427.520	22	12	1.238
Cyclone	EP1C3	2.910	59.904	--	1	104
	EP1C20	20.060	294.912	--	2	301
Excalibur	EPXA1	4.160	393.216	--	1	246
	EXPA10	38.400	4.194.304	--	--	711
Stratix DX	EP1SGX25D	25.660	1.944.576	10	4	462/614
	EP1SGX40D	41.250	3.423.744	14	8	638
Stratix II	EP2S15	15.600	419.328	12	2	365
	EP2S180	179.400	9.323.040	96	4	1173

Tabela 2.4: Resumo das famílias e características das FPGA da *Xilinx*.

Família	FPGA	Células lógicas	BRAM (kbits)	Processadores PowerPC	Multiplicador 18x18 Bits
Virtex-II Pro	XC2VP2	3.168	216	8	426
	XC2VP30	20.880	2.448	2	1.238
	XC2VP125	125.136	10.008	4	
Spartan3	XC3S50	1.728	72	--	104
	XC3S200	4.320	216	--	301
	XC3S5000	74.880	1.872	--	

2.4 - LÓGICA NEBULOSA

O desenvolvimento do sistema de elevadores, proposto neste trabalho, tem como objetivo melhorar a eficiência no transporte vertical de passageiros. A abordagem considerada para este propósito baseia-se na implementação de algoritmos de escalonamento de elevadores usando arquiteturas reconfiguráveis e o desenvolvimento de modelos heurísticos baseados em lógica nebulosa para a identificação de padrões de tráfego no prédio visando adotar estratégias de atendimento e despacho de elevadores (Muñozb *et al.*, 2006).

A lógica nebulosa (*fuzzy logic*) é uma técnica de IA baseada em conhecimento (sistemas especialistas), na qual um especialista (humano) gera uma base de conhecimento na forma de um banco de regras e o usuário (humano) consulta, ou valida, um sistema particular apresentando fatos à base de fatos. Os sistemas nebulosos incorporam a forma humana de pensar em um sistema de controle, porém, não pretendem entender ou explicar como é formado o raciocínio dedutivo do pensamento humano (Shaw e Godoy, 1999).

2.4.1 - Método de Modelagem Matemática

O objetivo principal em engenharia de controle é obter um modelo idealizado do processo a controlar para descrever como ele reage para várias entradas. Tradicionalmente, os modelos expressam-se em forma de equações diferenciais ou equações de diferença e são utilizadas ferramentas matemáticas como a *Transformada de Laplace* ou a *Transformada z*. Algumas restrições são feitas de forma a se obter um modelo matemático simples, entre elas podemos mencionar simplificações de processos *lineares* e *invariantes no tempo*.

Estas simplificações comumente feitas no controle de processos produzem certas dificuldades no desenvolvimento de uma descrição matemática significativa e realista do sistema (Shaw e Godoy, 1999).

2.4.2 - Método Heurístico

O método heurístico consiste em usar o "senso comum" na solução de problemas. Assim, a heurística considera realizar tarefas de acordo com a experiência prévia ou estratégias já freqüentemente utilizadas. Uma regra heurística associa conclusões (conseqüências) com condições (antecedentes ou fatos) e é construída como uma implicação lógica:

SE <*condição*> ENTÃO <*conseqüência*>

A vantagem do método heurístico é que possibilita a construção de uma função, não matemática, entrada-saída descrita ponto a ponto, na qual a restrição de linearidade não é mais necessária. A solução heurística de problemas não garante a solução ótima, no entanto, permite uma grande redução de custo e tempo e pode ser aplicada sempre que um modelo matemático equivalente for muito difícil ou complexo de se obter, considerando a presença de parâmetros incertos, desconhecidos ou variantes no tempo (Shaw e Godoy, 1999).

2.4.3 - Conjunto Nebuloso

Na teoria clássica de conjuntos um elemento qualquer pode pertencer ou não a um determinado conjunto, havendo só duas opções, sendo definidos os limites nítidos da relação elemento-conjunto. A teoria dos conjuntos nebulosos (*Fuzzy Sets)* foi introduzida em 1965 por Lotfi Zadeh quem afirmou que a transição desde "pertencer a um conjunto" até "não pertencer ao conjunto" é gradual (Zadeh, 1965 *apud* Shaw e Godoy, 1999).

Um elemento de um conjunto nebuloso não precisa, obrigatoriamente, pertencer ou não pertencer a este conjunto, podendo existir vários graus de pertinência, variando desde a completa exclusão (nível 0), até a pertinência máxima (nível 1). Quanto maior a pertinência de um objeto x com relação a um conjunto A, maior a compatibilidade deste objeto com a classe descrita por A. Assim, a definição de conjunto nebuloso generaliza a concepção de conjunto ordinário (Mamdani, 1975).

Seja X um conjunto de referência, também chamado de universo de discurso, expressado

de forma genérica por *x*. Um conjunto nebuloso *A* em *X* é definido como um conjunto ordenado de pares e por uma *função de pertinência* $\mu_A(x)$, que associa, para cada ponto em *X*, um número real no intervalo [0,1] que representa o grau de pertinência de *x* em *A*.

$$A = \{(x, \mu_A(x)) | X \rightarrow [0,1]\} \tag{2.2}$$

A figura 2.16 mostra de modo comparativo o conjunto ordinário (*A*) e nebuloso (*M*) que representam a idade de uma "pessoa adulta". Seja $X = R^+$ as idades possíveis para um ser humano. Define-se uma pessoa adulta (sobre os 50 anos) da forma:

$$M = \{(x, \mu_M(x)) | x \in X\} \tag{2.3}$$

onde, $\mu_M(x) = \dfrac{1}{1 + \left(\dfrac{x - 50}{10}\right)^4}$

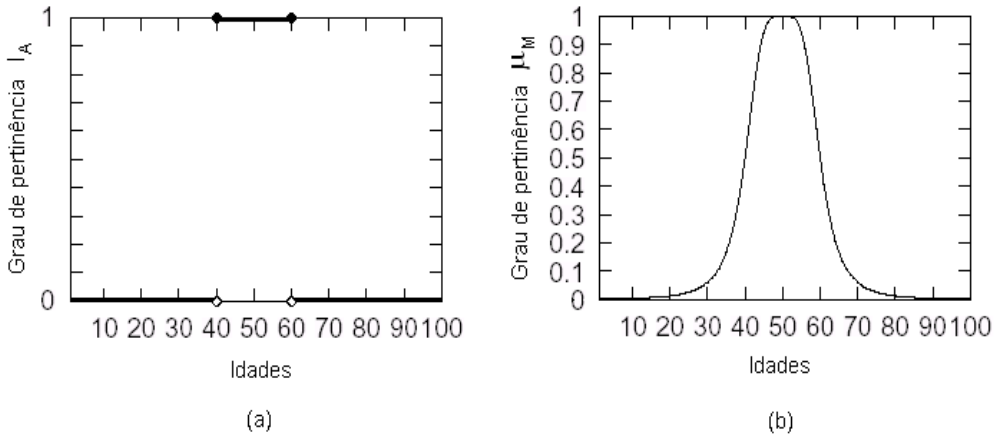


Figura 2.16 Conjuntos Ordinário e Nebuloso: (a) função do conjunto ordinário A "pessoa maior" (b) função gaussiana do conjunto nebuloso "pessoa maior".

O conjunto *A* não representa completamente o conceito de "pessoa adulta", pois uma pessoa com 39 anos e 11 meses seria considerada completamente incompatível com este conceito. Já o conjunto nebuloso *M* permite concluir que qualquer pessoa com idade entre 40 e 60 anos é adulta, acima dos 80 anos e abaixo dos 20 anos não é considerada uma pessoa adulta e, no intervalo [20,40] (respectivamente [60,80]), é considerado tanto mais adulto quanto mais próximo de 50 for a sua idade.

Na prática, sempre que o universo *X* seja contínuo é possível particionar *X* em vários conjuntos nebulosos de forma que as novas funções de pertinência μ_M abarquem todo *X*. A figura 2.17 mostra o mesmo universo *X* particionado em três funções de pertinência

representando três conceitos de idades diferentes: jovem, adulto e velho.

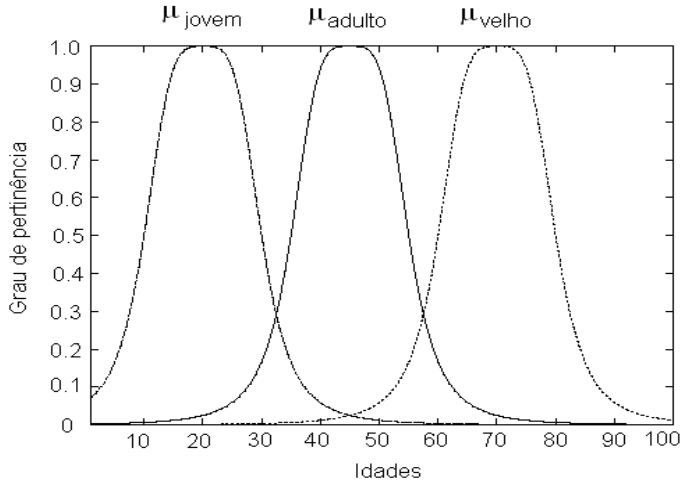


Figura 2.17: Particionamento do universo de discurso.

2.4.4 - Número nebuloso e funções de pertinência

Um número nebuloso é um conjunto nebuloso sobre *X*. Embora qualquer função de pertinência seja válida, existem famílias de números nebulosos com funções de pertinência parametrizáveis, alguns exemplos são relacionados a seguir:

- **Funções triangulares. Parâmetros {*a,b,c*}**

$$\text{triang}_{a,b,c}(x) = \max\left[\min\left(\frac{x-a}{b-a}\ \ \frac{c-x}{c-b}\right),0\right] \tag{2.4}$$

- **Funções trapezoidais. Parâmetros {*a,b,c,d*}**

$$\text{trap}_{a,b,c,d}(x) = \max\left[\min\left(\frac{x-a}{b-a}, 1, \frac{d-x}{d-c}\right),0\right] \tag{2.5}$$

- **Funções gaussianas. Parâmetros {σ,*c*}**

$$\text{gaussiana}_{\sigma,c}(x) = e^{-\left(\frac{x-c}{\sigma}\right)^2} \tag{2.6}$$

- **Funções Campana. Parâmetros {*a,b,c*}.**

$$Campana_{a,b,c}(x) = \frac{1}{1+\left|\frac{x-c}{a}\right|^{2b}} \tag{2.7}$$

Na figura 2.18 estão representadas as funções de pertinência: (a) triangular (20,60,80), trapezoidal (10,20,60,95), gaussiana (20,50), campana (20,4,50).

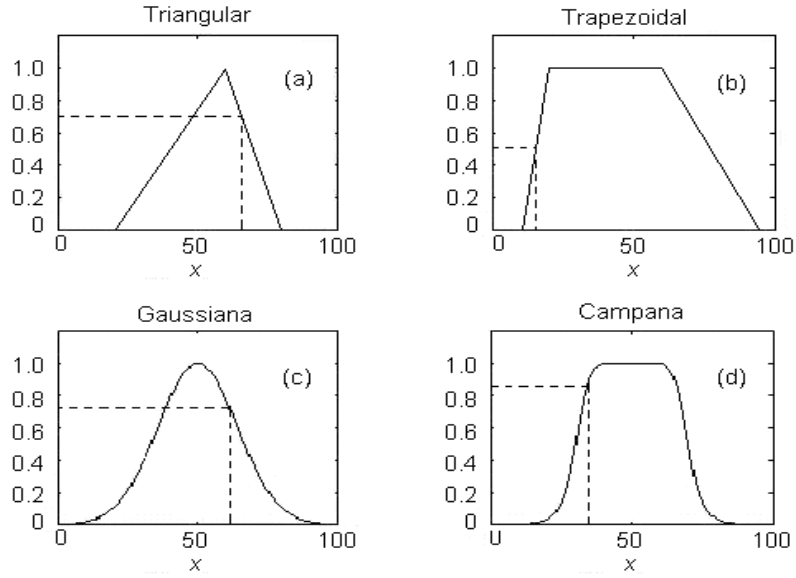


Figura 2.18: Funções de pertinência (a) Triangular (b) Trapezoidal (c) Gaussiana (d) Campana

2.4.5 - Variáveis lingüísticas

Como foi mencionado anteriormente, nos controladores convencionais o algoritmo de controle é descrito analiticamente por equações algébricas, diferenciais ou em diferenças, entretanto o controle nebuloso utiliza variáveis lingüísticas ao invés de variáveis numéricas. Uma variável lingüística x no universo de discurso X é definida em um conjunto de termos, nomes ou rótulos, $T(x)$, com cada valor sendo um número nebuloso (*fuzzy*) definido em X (Shaw e Godoy, 1999). Assim, as variáveis lingüísticas admitem apenas valores lingüísticos, por exemplo, os termos $T(a)$:

$$T(a) = \{\text{baixo, alto, muito alto, muitíssimo alto}\}$$

sobre o universo de discurso $X = [0,100]$, são termos lingüísticos de tipo altura, deste modo, *altura* é uma variável lingüística. Comumente cada termo lingüístico de uma variável é expressado por uma função de pertinência.

2.4.6 - Sistemas nebulosos, *fuzificação* e *defuzificação*

A configuração básica de um sistema nebuloso esta composta pelos blocos funcionais mostrados na figura 2.19. Esta estrutura representa a transformação que ocorre no domínio de mundo real, que usa números reais, para o domínio *fuzzy*, que usa números *fuzzy*.

Durante esta transformação um conjunto de inferências nebulosas é utilizado para tomar decisões e seguidamente é realizada uma transformação inversa do domínio *fuzzy* ao domínio do mundo real, acoplando as saídas f*uzzy* com as variáveis de atuação.

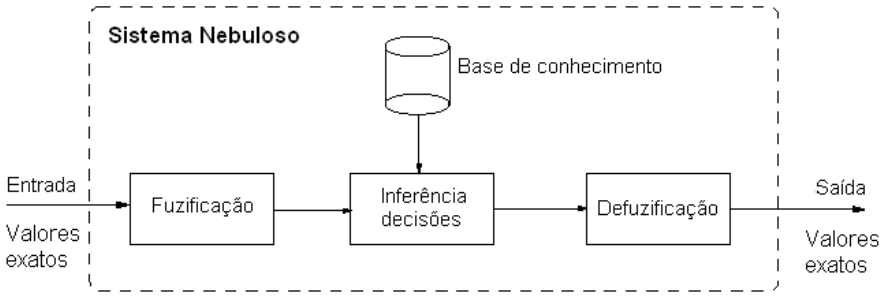


Figura 2.19: Sistema nebuloso.

- ***Fuzificação***

Os valores discretos (não-*fuzzy*) das variáveis de entrada geralmente são provenientes de sensores das grandezas físicas ou dispositivos de entrada computadorizados. O processo de *fuzificação* é um mapeamento do domínio de números reais para o domínio *fuzzy* considerando uma atribuição de valores lingüísticos, ou valores qualitativos, definidos por funções de pertinência às variáveis de entrada. O processo de *fuzificação* pode ser entendido como um pré-processamento de classes dos sinais de entrada, reduzindo consideravelmente o número de valores a serem processados, significando uma computação mais veloz. As funções de pertinência também podem ser um conjunto tabulado de valores numéricos e consultados em tabelas, procedimento conhecido como *fuzificação por tabela em memória*, útil quando os sistemas nebulosos são implementados em *hardware* (FPGAs) através de *LookUp-Tables* o que acelera o processo de *fuzificação*.

- **Base de conhecimento**

A base de conhecimento representa o modelo do sistema a ser controlado. Consiste em uma *base de dados* e uma *base de regras*. A base de dados contém as funções de pertinência lingüísticas utilizadas para representar os termos das variáveis de entrada e saída. A base de regras *fuzzy* lingüísticas contém o objetivo do controle armazenando a informação fornecida por *especialistas* na área de atuação do sistema nebuloso.

- **Inferência e tomada de decisões**

O processo de inferência *fuzzy* segue os seguintes passos:

1. Comparação dos fatos com as premissas (antecedentes).
2. Calculo do grau de compatibilidade de cada regra.
3. Calculo de crença em cada regra.
4. Agregação.

Após o processo de inferência, feito a partir do conhecimento especialista armazenado na base de regras, é possível utilizar implicações *fuzzy* para simular a tomadas de decisão humanas, gerando ações de controle. Os métodos de agregação mais utilizados são: o método de *Mamdani* (Mamdani, 1975, Mamdani, 1977), método de *Tsukamoto* (Tsukamoto, 1979) e o método de Takagi-Sugeno (Takagi e Sugeno, 1985). O método Mamdani é o modelo de implicação mais importante conhecido na literatura (Shaw e Godoy, 1999).

A figura 2.20 mostra um exemplo do modelo de raciocínio tipo Mamdani para um sistema de duas regras com dois antecedentes A e B cada uma. A implicação representada é:
Se A x B **então** C
Considerando as seguintes regras:
Se (a é A_1) e (b é B_1) **então** c é C_1.
Se (a é A_2) e (b é B_2) **então** c é C_2.

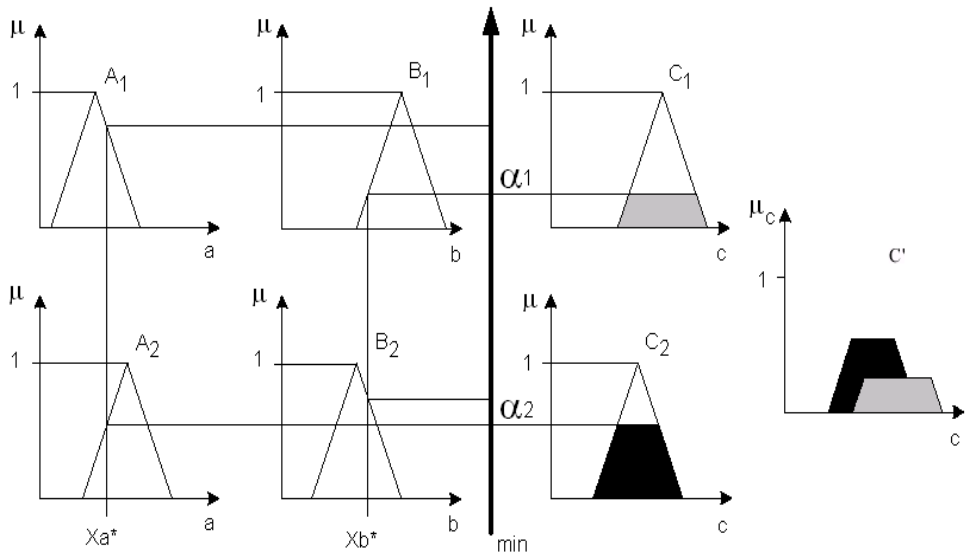


Figura 2.20: Método clássico de agregação Mamdani.

- ***Defuzificação***

Na *defuzificação*, o valor da variável lingüística de saída inferida pelas regras *fuzzy* será traduzido num valor discreto, com objetivo de obter-se um único valor numérico discreto

que executará uma ação de controle. Diferentes métodos de defuzificação são conhecidos, alguns deles exigem cálculos computacionais complexos.

1. Defuzificação Centro da Área (C-o-A)

Comumente denominado centro de gravidade, pois ele calcula o centróide da área do conjunto nebuloso de saída. O centróide é um ponto que divide a área em duas partes iguais. A equação 2.8 representa o calculo do centróide.

$$u^* = \frac{\sum_{i=1}^{N} u_i \, \mu_{OUT}(u_i)}{\sum_{i=1}^{N} \mu_{OUT}(u_i)} \tag{2.8}$$

O método do centróide apresenta problemas quando as funções de pertinência não possuem sobreposição e pode acontecer que o centro geométrico não tenha significado físico. Além disso, a necessidade de integração numérica exige esforços computacionais (Shaw e Godoy, 1999).

2. Defuzificação Centro do Máximo (C-o-M)

Neste método as áreas das funções de pertinência não desempenham nenhum papel e apenas os máximos (*pertinências singleton*) são usados. A saída discreta é calculada como uma média ponderada dos máximos, cujos pesos são os resultados da inferência conforme indica a equação 2.9, onde $\mu_{o,k}(u_i)$ indicam os pontos dos máximos das funções de pertinência de saída. Este método também é denominado defuzificação pelas alturas.

$$u^* = \frac{\sum_{i=1}^{N} u_i \cdot \sum_{k=1}^{n} \mu_{o,k}(u_i)}{\sum_{i=1}^{N} \sum_{k=1}^{n} \mu_{o,k}(u_i)} \tag{2.9}$$

3. Defuzificação Média do Máximo (M-o-M)

Em casos onde a função de pertinência tenha mais de um máximo a abordagem C-o-M não poderia ser utilizada, devido à necessidade de se escolher qual máximo utilizar, no entanto, pode tomar-se a média de todos os máximos, equação 2.10.

$$u^* = \sum_{m=1}^{M} \frac{u_m}{M} \tag{2.10}$$

onde M é o número de máximos encontrados, u_m é o m-ésimo elemento onde a função $\mu_{out}(u_i)$ tenha um máximo. No contexto do sistema nebuloso projetado neste trabalho foi utilizado o método M-o-M, pois é a solução mais plausível por desconsiderar o formato das funções de pertinência de saída além de não requerer cálculos computacionais complexos.

2.4.7 - Modelos de controle nebuloso em um grupo de elevadores

Pouco tempo depois da formulação dos conjuntos nebulosos, introduzida em 1965 por Lofti Zadeh, aplicações baseadas em lógica nebulosa começaram a ser desenvolvidas, principalmente em países de Asia e Europa (Shaw e Godoy, 1999). Os primeiros trabalhos foram realizados por Mamdani e Assilian. Os avanços nos sistemas de controle de grupo de elevadores (EGCS) foram desenvolvidos ao mesmo tempo que as técnicas de inteligência artificial começaram a ser exploradas em situações reais.

As técnicas de implementação dos sistemas nebulosos para o controle de grupo de elevadores são baseadas no uso de *software* sobre processadores programáveis de propósito geral. Os diferentes modelos e estratégias de sistemas nebulosos para o reconhecimento do tráfego de passageiros têm sido propostos desde começos dos anos 80, além de sistemas nebulosos aplicados a despacho de elevadores (Siikonen, 1997, Kim *et al.*, 1998, Xiong *et al.*, 2005).

No contexto do presente trabalho, o sistema de controle de grupo de elevadores via lógica nebulosa esta baseado nas implementações realizadas por Siikonen (1997) e Kim *et al.* (1998), pois os modelos propostos mostraram resultados satisfatórios em concordância com os objetivos deste projeto.

No sistema de controle de grupo de elevadores proposto por Siikonen (1997) (vide figura 2.21 (a)) a capacidade de transporte viu-se incrementada para diferentes padrões de tráfego, entretanto, o tempo de espera e de locomoção dos passageiros foi diminuído. No sistema nebuloso proposto por Kim *et al.* (1998), os padrões de tráfego são identificados a partir de características de tráfego que podem ser encontradas em diferentes tipos de edifícios considerando diversas situações e horários (vide figura 2.21 (b)).

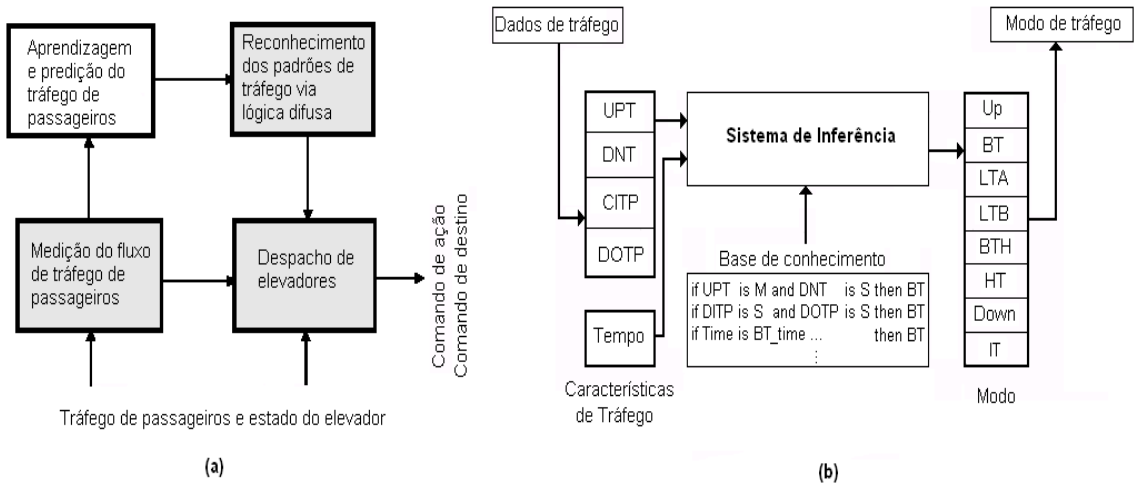


Figura 2.21: Modelos de referência (a) Blocos funcionais (modificado – Siikonen, 1997) (b) Estrutura do sistema nebuloso (modificado – Kim *et al.*, 1998).

2.5 – CONCLUSÃO DO CAPÍTULO

Neste capítulo foram abordadas algumas definições consideradas relevantes para o desenvolvimento do presente trabalho. O estudo do estado da arte na industria de elevadores evidenciou uma evolução dos sistemas de transporte vertical, sendo justificado o uso de sistemas de elevadores em prédios comerciais e residenciais, pois estima-se uma economia no consumo de potência, além do aumento no conforto dos usuários. Em edifícios modernos os sistemas de elevadores são implementados por um Sistema de Controle de Grupo de Elevadores e por alguns subsistemas microprocessados que implementam o Sistema de Controle Local de cada elevador. As empresas nacionais e internacionais fabricam quadros de comando de elevadores e nos últimos anos têm desenvolvido Sistemas de Controle de Grupo e simuladores executados em computadores pessoais.

As técnicas de inteligência artificial baseadas em lógica nebulosa para o desenvolvimento dos sistemas de elevadores foram introduzidas a partir dos anos 1980. Algumas empresas têm aplicado estas técnicas e evoluíram no sentido de fornecer um melhor serviço no transporte vertical de passageiros. Entretanto, novas implementações são propostas e a aplicação de sistemas nebulosos para despacho de elevadores ainda é motivo de estudo por parte das empresas e a comunidade científica.

O objetivo principal de uma nova implementação de um sistema de elevadores é aumentar o conforto dos usuários ao mesmo tempo que o consumo de potência deve ser diminuído. Neste capítulo também foi apresentada a tecnologia de *hardware* reconfigurável baseada em FPGAs, examinando a sua potencialidade para implementar algoritmos de escalonamento de elevadores diretamente em *hardware.* Espera-se, com estas implementações, ganhar em desempenho, custo, flexibilidade e eficiência de forma que o sistema de elevadores proposto possa ser uma alternativa de estudo.

3 - PLATAFORMA DE DESENVOLVIMENTO

Este capítulo aborda uma descrição das diferentes plataformas de desenvolvimento consideradas na elaboração do presente trabalho. A plataforma de *hardware* é constituída principalmente pela placa de desenvolvimento FPGA *Xilinx* Spartan3, uma rede de conexão baseada no padrão RS-485 e uma placa de potência para controle do quadro de comando de elevadores. Os recursos de *software* apresentados é composto de ferramentas de desenho e automação eletrônico (EDA, *Electronic Design Automation*) para desenvolvimento de sistemas digitais programados em *hardware*, que são integradas no pacote *Xilinx ISE 7.1*, uma ferramenta de desenho assistido por computador (CAD) para desenvolvimento e validação de sistemas nebulosos, *Xfuzzy 3.0*, e a ferramenta *Eclipse 3.1* utilizado para embarcar e monitorar o controle de grupo no simulador virtual de elevadores baseado na linguagem de programação *Java*.

3.1 - DESENVOLVIMENTO DO SISTEMA DE CONTROLE LOCAL

A implementação e validação do sistema de controle local de elevadores (LCS – vide figura 2.1) é realizado utilizando-se placas FPGA (*Field Programmable Gate Array*) da família Spartan3 e por um pacote de ferramentas de *software* para desenvolvimento e simulação de sistemas digitais programados em *hardware* (*ISE 7.1*), ambos fornecidos pela empresa *Xilinx* (Xilinx, 2005). A seguir são explicadas algumas características, da placa Spartan3 e do pacote *ISE* 7.1, consideradas importantes no contexto do desenvolvimento dos LCSs.

3.1.1 - Características da placa Spartan3

A placa de desenvolvimento Spartan3 fornece uma plataforma para desenho de sistemas em *hardware*. Os recursos citados a seguir constatam as funcionalidades que podem ser programadas na placa Spartan3.

- Dispositivo FPGA XC3S200 da família Spartan3. Suas principais características são citadas na tabela 3.1;
- 2 Mbit de memória *flash* (XCF02S), não volátil para configuração da FPGA toda vez que o sistema for energizado;
- 1 Mbyte de memória SRAM (RAM estática assíncrona);

- Tensões de 3,3V, 2,5V e 1,5V disponíveis nos barramentos de expansão;
- Porta DB9 de comunicação RS-232;
- Oscilador de 50 MHz;
- Porta VGA de 8 cores;
- Porta PS/2 para controle de *mouse* e teclado;
- Porta JTAG para abaixar os dados de configuração da FPGA (*datastream*);
- Oito LEDs individuais, oito chaves tipo *slide-switches*, quatro *displays* de 7-segmentos e quatro botões tipo *push-button;* e
- Três portas de expansão, cada uma de 40 pinos.

Tabela 3.1: Características da FPGA Spartan3 XC3S200

Elementos lógicos	4320 LB
Memória RAM *on-chip*	216 Kbits
Multiplicadores em *hardware*	12 (18x18)
Pinos de I/O disponíveis ao usuário	173
Gerenciadores de relógio digital (*clock*)	4

A figura 3.1 apresenta a placa de desenvolvimento Spartan3 e a localização dos componentes relacionados. Todos os acessórios necessários para a conexão da placa, fonte de alimentação e cabo de programação, são fornecidos junto à placa de desenvolvimento.

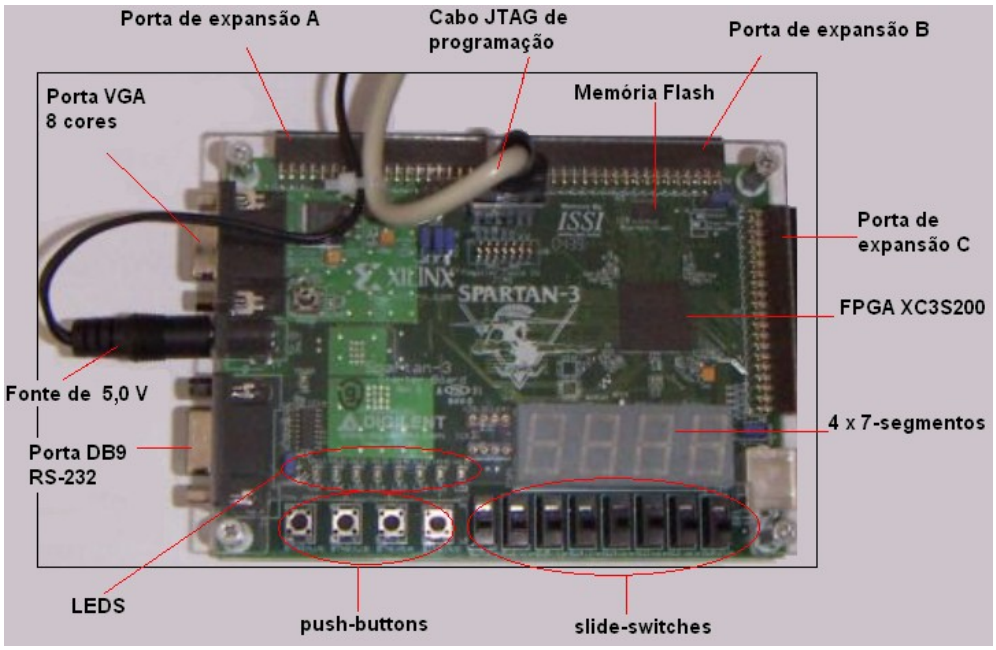


Figura 3.1: Localização dos componentes na placa Spartan3.

3.1.2 - Ferramentas de *software* para programação em *hardware*

Os recursos de *software* utilizados para o desenvolvimento dos sistemas descritos em *hardware* são integrados em um único pacote, *ISE 7.1*, fornecido pela empresa *Xilinx* (Xilinxb, 2005). Este ambiente compreende todas as fases de desenvolvimento de sistemas em FPGA considerando alta flexibilidade na programação de diferentes famílias de FPGAs da *Xilinx*. Entretanto, são fornecidos recursos e módulos que facilitam e otimizam o esforço do projetista. As etapas de desenho do pacote *ISE 7.1* envolvem primeiramente a descrição do projeto em linguagem de descrição de *hardware* (VHDL) e o mapeamento das entradas e saídas do sistema na placa FPGA. Posteriormente é realizada a síntese do projeto, o análise de consumo de recursos e desempenho. Por último, é realizada a etapa de simulação funcional baseada na ferramenta *ModelSim* (ModelSim, 2006) e a programação do dispositivo FPGA (Xilinxb, 2005).

3.2 - MÓDULO CONTROLADOR DE GRUPO VIA LOGICA NEBULOSA

O módulo de controle de grupo de elevadores (EGCS – vide figura 2.1) é descrito através de um sistema de inferência baseado em lógica nebulosa. Utilizou-se uma ferramenta de CAD para sistemas nebulosos, *Xfuzzy 3.0*, criada e fornecida livremente pelo grupo *Xfuzzy* do *Centro de Microelectrónica de Sevilla*, Universidad de Sevilla, Espanha (Xfuzzy, 2005).

O *Xfuzzy 3.0* é um entorno de desenvolvimento para sistemas de inferência baseados em lógica nebulosa. Esta constituído por diversas ferramentas que acompanham as diferentes etapas no projeto de sistemas nebulosos, desde a sua descrição inicial até a implementação final. A figura 3.2 mostra as etapas de projeto no *Xfuzzy 3.0*.

A etapa de descrição inclui ferramentas gráficas para a definição do sistema nebuloso. A etapa de verificação é composta por recursos de simulação, monitoração e representação gráfica do comportamento do sistema. A etapa de ajuste considera a aplicação de algoritmos de aprendizagem. Finalmente, a etapa de síntese inclui ferramentas para gerar descrições em linguagens de alto nível para implementações em *software* e *hardware* (Cabrera, 2003, Xfuzzy, 2005).

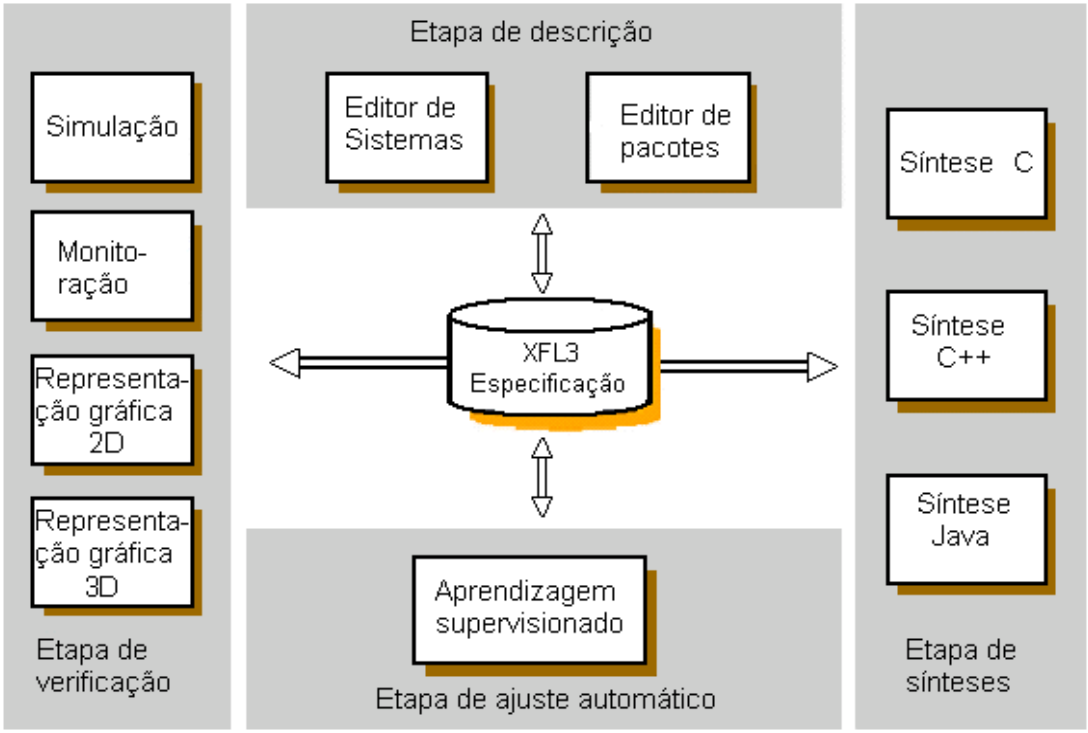


Figura 3.2: Desenho de sistemas nebulosos com *Xfuzzy* (modificado - Xfuzzy, 2005).

3.3 - REDE DE CONEXÃO DOS LCS COM O CONTROLE DE GRUPO

No sistema de elevadores, cada LCS irá controlar os recursos disponíveis para um elevador. Assim, o conjunto de sistemas de controle local e de controle de grupo são conectados para que os sinais possam ser enviados entre eles (vide figura 2.1). Para realizar esta tarefa, é necessário analisar a compatibilidade dos sinais, principalmente em relação aos níveis de tensão e consumo de corrente. A placa de desenvolvimento FPGA Spartan3 possui uma porta DB9 para comunicação serial RS–232 comumente utilizada para conexão da FPGA com um computador pessoal. Esta interface não permite conexões maiores do que 15 metros além de ser suscetível a problemas de comunicação devido às diferencias de potencial entre os sinais de terra e ao controle limitado dos tempos de picos e descidas do sinal, impossibilitando a sua aplicação no sistema de elevadores.

Aproveitando este recurso disponível e considerando as limitações que dita interface possui, implementou-se uma rede de conexão baseada no padrão RS-485, que permite a comunicação de dados seriais entre periféricos sobre uma interface mais robusta e eficiente, relevante na aplicação em sistemas de elevadores. Comumente, a rede de conexão de elevadores nos edifícios é baseada na interface RS-485 (Bushby, 1997), e em

sistemas mais modernos são encontradas interfaces de conexão de rede de controle de área (*CAN Open Lift – Control Area Network*) (Zeltwanger, 2004, CAN, 2005).

3.3.1 - Interface para o padrão RS-485

O RS-485 é um padrão físico de comunicação serial, tais como o RS-232 e o RS-422, onde cada sinal usa um par trançado, que se baseia no conceito de transmissão diferencial de dados balanceados.

O princípio da diferença de tensão é que o dado não está referenciado a um terra (0 Volt – terra), e portanto o nível lógico é definido pela diferença do potencial entre os fios, nomeados *A* e *B*. Por exemplo, caso no fio *A* se colocar um potencial alto, e no *B* um potencial baixo, tem-se uma diferença de tensão. Considerando *B* menos *A*, tem-se um resultado negativo o que resulta num nível lógico 0. Por outro lado, se for colocado um nível lógico alto no fio *B* e um baixo no *A*, tem-se um resultado positivo, nível lógico 1.

Utilizando um par trançado para a transmissão de dados por diferença de tensão, têm-se diversas vantagens. As interferências eletromagnéticas atuam simultaneamente nos dois fios, não aparecendo tensões induzidas do tipo efeito *Hall*, mantendo assim a diferença de tensão.

Algumas características do padrão RS485 são:

- Comunicação multiponto, permitindo ligar diversos componentes no mesmo barramento (até 31 dispositivos);
- Distancias de conexão até 1 Km;
- Arquitetura Mestre/Escravo. Essa abordagem é a usada na maioria das aplicações.

Existem duas versões do RS-485: Um par trançado simples (dois fios) e um par trançado duplo (quatro fios). Na versão par trançado simples, ilustrado pela figura 3.3, todos os dispositivos possuem três estados (incluindo o mestre): *transmitindo*, *recebendo* e em *espera*. A comunicação é bi-direcional, sendo o controle de fluxo feito por *software*. No par trançado duplo, ilustrado pela figura 3.4, o mestre não possui os três estados, sendo um dos canais para o mestre se comunicar com os escravos, e o outro para os escravos se comunicarem com o mestre.

Para a conexão das FPGAs com o barramento é necessário um conversor dos sinais provenientes da porta RS-232 da FPGA e os sinais provenientes do mestre (*host*), garantindo a compatibilidade entre os dois barramentos.

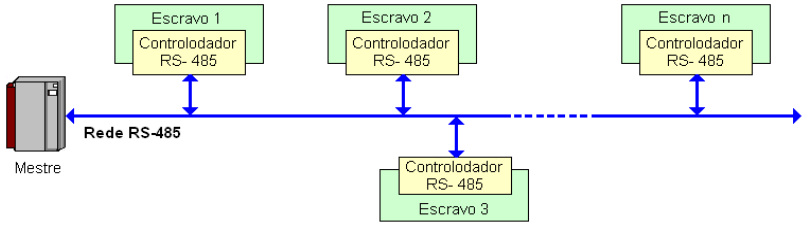


Figura 3.3: Barramento RS-485 com par trançado simples.

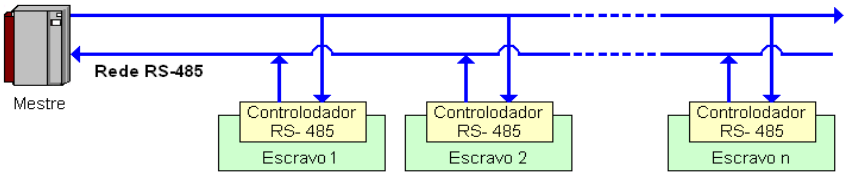


Figura 3.4: Barramento RS-485 com par trançado duplo.

3.3.2 - Placa WP05

O WP05 é uma placa que cria o meio físico para a interface RS-485 (figura 3.5) e realiza a conversão dos sinais do padrão RS-232 ao padrão RS-485 e vice-versa. Esta placa foi desenvolvida por uma empresa local, WP (WP, 2006) e faz parte do contexto de controle de acesso do projeto SIAP.

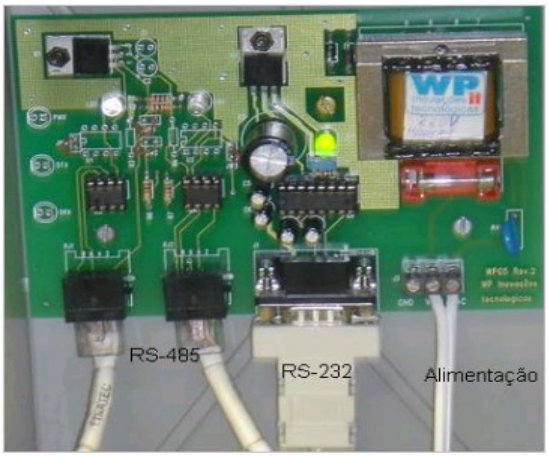


Figura 3.5: Placa WP05.

Para cada FPGA que implementa um sistema de controle local é necessário uma WP05, cabos de par trançado sem blindagem (UTP categoria 5) e conectores RJ45. São utilizados dois tipos de ligação. A primeira serve para conectar o WP05 mestre aos WP05 escravos. O segundo tipo de ligação serve para unir os escravos.

A topologia da rede RS-485 implementada no desenvolvimento deste projeto é par trançado duplo (figura 3.4). As ligações com os WP05 ficam de acordo como apresentado na figura 3.5.

O circuito de conversão do sinal RS-232 para RS-485 é feita em duas etapas. A primeira etapa é a recepção dos dados pelo integrado *Max232,* recebendo a informação em nível de tensão de -15 a +15 Volts e transforma para níveis entre 0 e 5 Volts (nível TTL). A segunda etapa é composta de uma dupla de circuitos integrados *SN75176BP* capaz de transformar o nível TTL em RS-485 para balanceamento das linhas de transmissão do barramento de comunicação multiponto (Texas, 1985).

Nas saídas *A* e *B* do chip *SN75176BP* são conectados os sinais de informação. A tabela verdade mostra o seu funcionamento (vide figura 3.6). Outro sinal importante é o **DE**, cuja função é habilitar a transmissão, abaixando a impedância da rede. Neste sentido, foi necessário acrescentar no projeto desenvolvido para os Sistemas de Controle Local um pino de controle de impedância na rede RS-485, ao mesmo tempo foi necessário habilitar um pino extra no cabo de transmissão serial RS-232 (vide figura 3.1). Quanto aos outros pinos não há relevância para o entendimento da interface implementada.

DRIVER

INPUT D	ENABLE DE	OUTPUTS A	OUTPUTS B
H	H	H	L
L	H	L	H
X	L	Z	Z

RECEIVER

DIFFERENTIAL INPUTS A-B	ENABLE $\overline{RE}$	OUTPUT R
V_{ID} >= 02.V	L	H
-0.2 v < v_{ID} < 0.2 V	L	?
V_{ID} <= -0.2 V	L	L
X	H	Z
Open	L	?

H = high level, L = low level, ? = indeterminate
X = irrelevant, Z = high impedance

Figura 3.6: Tabela de verdade do CI sn75176BP.

3.4 - MONITOREAMENTO DO SISTEMA DE ELEVADORES

Uma interface virtual de elevadores (VEI) foi construída com objetivo de simular e monitorar o comportamento do controle de grupo, os controles locais implementados nas FPGAs e de forma geral o sistema de elevadores. Esta interface foi desenvolvida em

linguagem *Java*, pensando na possibilidade de ser executada livre de plataforma, constituindo um dos objetivos do projeto SIAP. O recurso de *software* utilizado nesta tarefa é o *Eclipse SDK* versão 3.1.2 fornecendo suporte para a implementação gráfica do simulador, a conexão via RS-232 e o módulo de controle de grupo baseado em sistemas de inferência nebulosos (FEGCS – vide figura 2.1).
O sistema de monitoração integra as funcionalidades necessárias para o gerenciamento em PCs do barramento RS-485 e ao mesmo tempo da interface virtual de elevadores simulando os botões de chamada de pavimento e de cabina, atualizando o estado de cada elevador e apresentando resultados de escalonamento.

3.5 – CONCLUSÃO DO CAPÍTULO

Neste capítulo foram discutidas as ferramentas e plataformas de desenvolvimento usadas para a implementação do sistema de elevadores. A placa de desenvolvimento FPGA Spartan 3 cumpre os recursos técnicos necessários para abordar a descrição dos algoritmos de escalonamento de elevadores usando arquiteturas reconfiguráveis, além de fornecer a programação das funcionalidades requeridas para a conexão com o mundo exterior. Por outro lado, mediante a implementação em FPGAs dos Sistemas de Controle Local puderam ser discutidas as vantagens em relação ao desempenho, consumo de área e consumo de potência.

A rede de conexão baseada no padrão RS-485 fornece uma interface robusta e eficiente para a comunicação de dados seriais entre os periféricos e o Sistema de Controle de Grupo de Elevadores. Esta implementação permite o desenvolvimento de uma arquitetura Mestre/Escravo na qual é possível interligar 31 periféricos distribuídos numa distância até de 1 Km. Ditas características são relevantes e justificam o uso da rede RS-485 na implementação de sistemas de elevadores.

Os recursos de *software* escolhidos no desenvolvimento deste trabalho foram feitos tendo em conta fatores como a facilidade de acesso através do uso de plataformas livres, portabilidade e suporte. Ao mesmo tempo estes recursos fornecem a flexibilidade e os recursos técnicos necessários para o desenvolvimento do sistema de elevadores.

4 - IMPLEMENTAÇÃO DO SISTEMA DE ELEVADORES

Este capítulo descreve as arquiteturas implementadas para o funcionamento do sistema de elevadores proposto neste trabalho. Um detalhamento dos módulos de *hardware* utilizados na implementação dos algoritmos de escalonamento é apresentado, assim como as interfaces de conexão com o controle de grupo e o barramento RS-485. Todas estas funcionalidades foram desenvolvidas utilizando-se a ferramenta EDA (*Electronic Design Automation*) *Xilinx ISE* 7.1 e a linguagem de descrição de *hardware* (VHDL) realizando a programação e configuração do sistema de controle local (LCS) nas placas Spartan3. Em seguida é abordada a construção do sistema de inferência nebuloso e a interface virtual para monitoração, simulação e validação em tempo real do comportamento dos algoritmos de escalonamento num grupo de elevadores.

Nos sistemas convencionais o sistema de controle de grupo (EGCS) calcula o próximo andar a ser visitado por cada elevador, usando uma estratégia determinada (Kim *et al.*, 1995, Siikonenb, 1997, Kim *et al.*, 1998). Na proposta deste trabalho, o cálculo do próximo andar é realizada pelo controle local de cada elevador. A implementação do sistema proposto é mais flexível dado que diferentes algoritmos de escalonamento para cada elevador podem ser escolhidos, dependendo do tipo de trafego num instante determinado. Por outro lado, o EGCS é determinará o algoritmo mais apropriado a ser rodado em cada um LCS. Igualmente o EGCS calculará o elevador que realizará o atendimento mais eficiente das chamadas de pavimento, tendo em conta o tráfego atual no prédio (vide seção 2.2.3).

As chamadas de cabina são tidas em conta no analise de distribuição de passageiros no edifício. O atendimento das chamadas de cabina, também tratadas nos LCSs, é realizado segundo a chamada mais próxima no sentido de direção do carro (princípio coletivo, vide seção 2.2.5).

4.1 - ARQUITETURA IMPLEMENTADA NO SISTEMA DE ELEVADORES

O sistema de elevadores é composto por três módulos principais, conforme as características explicadas na seção 2.1.1:

a) O sistema de controle de grupo (EGCS);
b) Os sistemas de controle local (LCSs);
c) Uma interface de conexão baseada no padrão RS-485.

A figura 4.1 apresenta a arquitetura geral. O EGCS é um sistema de alto nível cuja tarefa é escolher a melhor estratégia de funcionamento de cada LCS. Por outro lado, a tarefa principal de cada LCS é escolher qual é o próximo andar a ser visitado por seu elevador, acionando o sistema para a movimentação do elevador. A interface de conexão baseada no padrão RS-485 é composta por alguns nós de *hardware* que implementam a camada física da rede. Cada LCS é conectado com cada nó da rede através de uma interface RS-232.

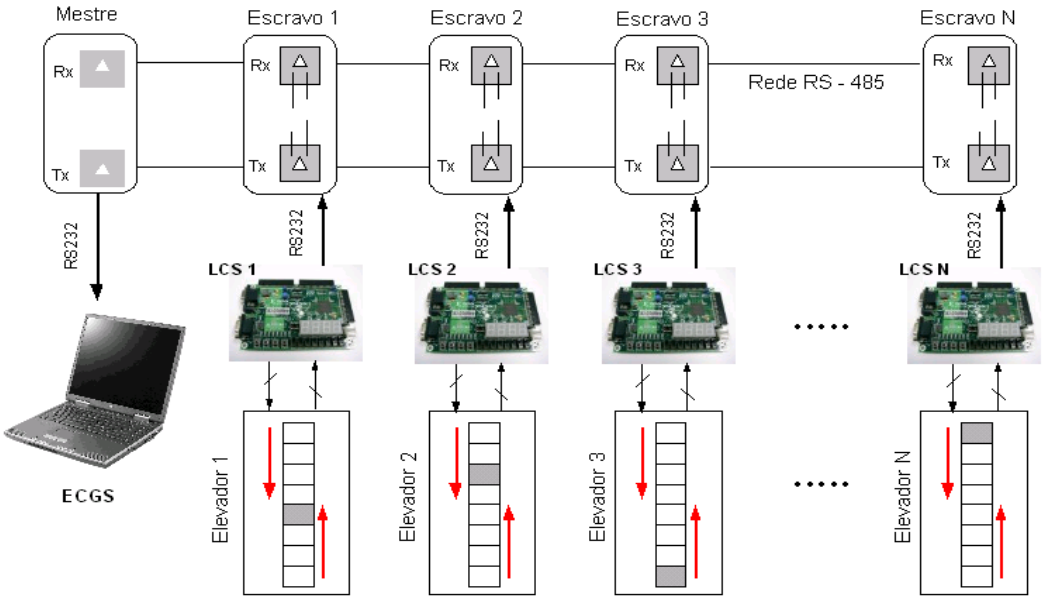


Figura 4.1: Arquitetura implementada para o sistema de elevadores.

O funcionamento do sistema considera *comandos de ação* enviados desde o EGCS aos controles locais (LCS) e *comandos de estado* enviados desde cada LCS ao controle de grupo (EGCS). Para esta tarefa, cada controle local, que no contexto da interface RS-485 é um escravo, possui um endereço de elevador que o identifica. Assim o controle de grupo (implementado pelo mestre ou *host*) realiza tarefas de execução/monitoração conforme descrito a seguir:

a) Pode enviar *comandos de ação* a um elevador em particular;
b) Identificar qual dos elevadores mudou de estado;
c) Captura de dados;
d) Transmitir os dados recebidos à interface virtual para que possam ser apresentados

no monitor;

e) Analisar o tráfego e distribuição atual dos passageiros no edifício;

f) Retornar mudanças de estratégias com base no tipo de tráfego atual.

4.2 - ARQUITETURA DO SISTEMAS DE CONTROLE LOCAL

Os diferentes componentes do sistema de controle local implementado em *hardware* foram descritos em VHDL e sintetizados utilizando a ferramenta *EDA ISE 7.1* da *Xilinx*. O LCS pode ser visto como uma caixa preta, ou entidade, que implementa algumas funcionalidades, comunicando-se de forma serial com o barramento externo através de dois pinos *din – dout.* Adicionalmente, o LCS controla os componentes físicos do elevador (porta, motor, *leds* e *displays*). Um detalhamento das portas de entrada e saída requeridas na construção do LCS é mostrado na figura 4.2.

Os sinais de *reset* e *clock* são utilizados para sincronismo do sistema. Entretanto, o sinal *pinhighZ* é usado para abaixar a impedância do barramento solicitando atenção do mestre para receber um comando. O sinal de entrada *endereço* permite identificar cada LCS facilitando a comunicação. Os outros sinais de I/O serão explicados nas seções a seguir.

```
 1  ENTITY ControleLocal IS
 2   GENERIC (numandar : positive := 8);   --Número de andares
 3   PORT (
 4        reset       : IN  std_logic;
 5        clk50M      : IN  std_logic;
 6        din         : IN  std_logic;
 7        endereço    : IN  std_logic_vector(2 DOWNTO 0);
 8        IR          : OUT std_logic;
 9        pinhighZ    : OUT std_logic;
10        dout        : OUT std_logic;
11        motor       : OUT std_logic_vector(1 DOWNTO 0);
12        porta       : OUT std_logic;
13        Registrador : OUT std_logic_vector (numandar-1 DOWNTO 0);
14        Anodo       : OUT std_logic_vector(3 DOWNTO 0);
15        RES         : OUT std_logic_vector(7 DOWNTO 0));
16  End ControleLocal;
17
18  ARCHITECTURE Comportamental OF ControleLocal IS
19    component SerialComm
20      (...)
21    end component;
22    component ProtocoloELE
23      (...)
24    end component;
25    component Elevador
26      (...)
27    end component;
28
29  BEGIN
30    (...) -- Conexao das instancias
31  End Comportamental;
```

Figura 4.2: Descrição das portas de entrada e saída do LCS.

4.2.1 - Funcionalidades implementadas no sistema de controle local (LCS)

O sistema de elevadores proposto considera a capacidade de cada Sistema de Controle Local (LCS) mudar as estratégias de atendimento, segundo o tipo de tráfego atual de passageiros no edifício.

Conforme explicado na seção 2.2.3, o fluxo e a distribuição de passageiros dentro do edifício pode ser caracterizado levando em conta medidas realizadas a partir dos botões de seleção de chamada de pavimento e de cabina. A figura 4.3 aclara os possíveis movimentos dos passageiros, encontradas em edifícios comerciais, e que definem as características de tráfego usualmente encontradas nos mesmos horários e situações (vide tabela 2.1).

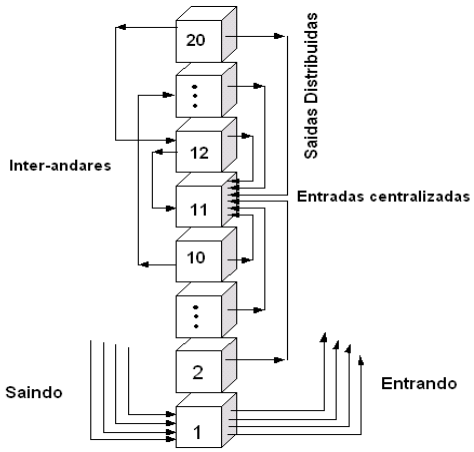


Figura 4.3: Comportamento do tráfego de passageiros em edifícios comerciais.

No contexto deste trabalho, considerou-se a classificação de tráfego para o transporte vertical de passageiros realizada por Kim *et al.* (1998) e relacionada na tabela 2.2. Para cada uma destas situações de tráfego é possível executar uma estratégia de atendimento específica. A escolha da estratégia vai depender, basicamente, do objetivo desejado em determinado momento:

a) Diminuição do tempo de espera dos usuários;
b) Diminuição da potência de consumo do sistema de elevadores.

Os cinco algoritmos de escalonamento implementados em *hardware* podem atender qualquer um dos oito padrões de tráfego relacionados na tabela 2.2, porém, em determinado momento, só um deles alcança maior eficiência no atendimento, melhorando a qualidade do serviço. Em casos específicos de tráfego pode ser escolhido um algoritmo

que permita diminuir o consumo de energia. Por exemplo, em casos de tráfego em tempo de negócios de pouca intensidade (*BT – Business Time*) ou tráfego em tempo inativo (*IT – Inactive Time*), é possível manter desligados, ou em espera, a maioria dos elevadores.

Conforme explicado na seção 2.2.4, os algoritmos de escalonamento seguem os princípios coletivo e seletivo, lembrando que as implementações fazem referência unicamente às chamadas de pavimento, pois o atendimento das chamadas de cabina, também tratado nos LCS, é realizado segundo a chamada mais próxima no sentido de direção do carro (principio coletivo). As chamadas de cabina também fornecem importante informação sobre a distribuição de passageiros no edifício.

Em base a estes conceitos, os seguintes algoritmos, descritos a partir de máquinas de estados finitos, foram implementados (vide seção 2.2.4):

a) Algoritmo Coletivo subida / Coletivo descida (C).
b) Algoritmo Coletivo subida / Seletivo descida (U).
c) Algoritmo Seletivo subida / Coletivo descida (D).
d) Algoritmo Seletivo subida / Seletivo descida (S).
e) Algoritmo Coletivo - Seletivo / subida – descida (CS-ud).

Nas seguintes seções são descritos, detalhadamente, cada um dos módulos funcionais da arquitetura proposta para o sistema local de elevadores.

4.2.2 - Implementação do registrador de chamadas

O componente básico do Sistema de Controle Local (LCS) é um subsistema composto de um registrador de *n-bits* e *n* multiplexadores de duas entradas e uma saída, onde a *i-ésima* célula do registrador representa o *i-ésimo* andar no edifício. Este sistema é denominado *Subsistema_Registrador* e a sua função principal é armazenar os andares para os quais é referenciada uma chamada (armazenamento de chamadas de pavimento e de cabina).

Cada multiplexador controla a transmissão dos sinais provenientes das chamadas de pavimento e cabina E_i e do sinal *bit_rst.* Uma chamada pode ser registrada escrevendo um "0" na célula respectiva. Entretanto, os multiplexadores, ou unidades de controle, podem apagar o registro da chamada escrevendo "1" no bit de clareio (*bit_rst*), selecionando a

célula correspondente através do sinal de controle C_i (*ControlMux*). A figura 4.4 apresenta a arquitetura implementada para o registrador de chamadas.

O vetor de saídas S_i contém informação do andar ou andares solicitados. Portanto, este vetor será o objeto de inspeção por parte dos algoritmos de escalonamento.

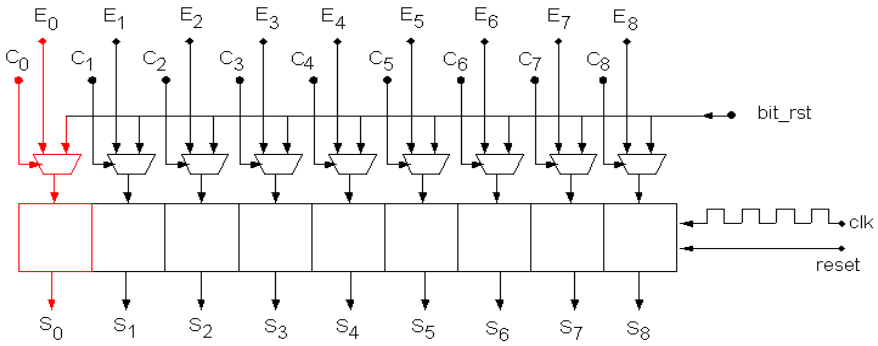


Figura 4.4: Arquitetura básica do registrador de chamadas.

Para permitir a implementação do Algoritmo *Coletivo - Seletivo / subida – descida* (CS-ud) foram construídos dois subsistemas para registro de chamadas: um para armazenar chamadas de subida e outro para chamadas de descida. Deve ser lembrado que para este tipo de implementação o sistema possui 2 botões de seleção de chamada em cada pavimento, indicando a direção de movimento desejada, seja subida ou descida.

4.2.3 - Implementação dos algoritmos C, U, D e S

Na descrição dos primeiros quatro algoritmos implementados em *hardware* (vide seção 4.2.1), é usado unicamente um *Subsistema_Registrador*. Portanto, a especificação dos algoritmos objetiva a aplicação em edifícios com um botão de seleção de chamadas em cada pavimento.

No intuito de testar e validar os quatro algoritmos, cada um deles foi implementado independentemente segundo a arquitetura mostrada na figura 4.5 diferençando-se apenas na descrição da lógica do funcionamento, ou seja, a estratégia de atendimento das chamadas.

A figura 4.6 apresenta o código VHDL simplificado que descreve as portas de entrada e saída de uma entidade chamada *Elevador* e a estrutura do componente denominado *CoreElevador*. A descrição mostrada implementa o controle de um único algoritmo para um edifício de 8 andares.

A arquitetura implementada possui duas máquinas de estados finitos *Fsm_reg e Fsm_ele*. A *Fsm_reg* determina o próximo andar a ser visitado, recebendo como entradas o vetor de chamadas SR (vide figura 4.5) e um sinal do sensor da porta (IR, indicando quando a porta encontra-se aberta ou fechada). Sempre que um andar é atendido, o módulo *Fsm_reg* apaga o registro da chamada através do sinal de saída *ControlMux*, que representa o controle dos multiplexadores do *Subsistema_Registrador,* permitindo a transmissão do valor do sinal *bit_rst* ao registrador de andares.

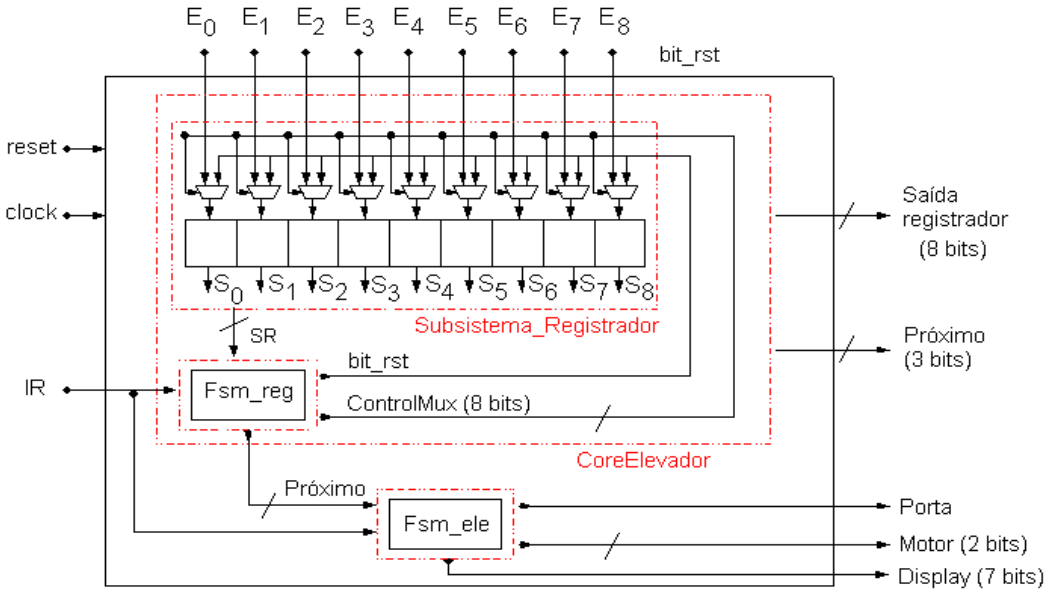


Figura 4.5: Arquitetura básica dos algoritmos C, U, D e S.

```
1  ENTITY Elevador IS
2    GENERIC (numandar : positive := 8);  --Número de andares
3    PORT (Reset       : IN  std_logic;
4          Clk         : IN  std_logic;
5          E0,E1,E2,E3,E4,E5,E6,E7 : IN std_logic;
6          IR          : IN  std_logic;
7          Proximo     : OUT std_logic_vector (2 DOWNTO 0);
8          Motor       : OUT std_logic_vector (1 downto 0);
9          Porta       : OUT std_logic;
10         Registrador : OUT std_logic_vector (numandar-1 DOWNTO 0);
11         Display     : OUT std_logic_vector (6 DOWNTO 0);
12 END Elevador;
13 ARCHITECTURE Comportamental OF Elevador IS
14 COMPONENT CoreElevador
15   (...)
16 END COMPONENT;
17
18 COMPONENT FSM_ele
19   (...)
20 END COMPONENT;
21
22 BEGIN
23   (...)
24 END Comportamental;
```

```
1  ENTITY CoreElevador IS
2    GENERIC (numandar : positive := 8);  --Número de andares
3    PORT(Reset       : IN  std_logic;
4         Clk         : IN  std_logic;
5         E0,E1,E2,E3,E4,E5,E6,E7 : IN std_logic;
6         IR          : IN  std_logic;
7         andatend    : IN  std_logic;
8         Registrador : OUT std_logic_vector (numandar-1 DOWNTO 0);
9         Proximo     : OUT std_logic_vector(2 DOWNTO 0));
10 END CoreElevador;
11 ARCHITECTURE Comportamental OF MiniElevador IS
12   (...)
13 COMPONENT RegComp
14   (...)
15 END COMPONENT;
16 COMPONENT Fsm_reg
17   GENERIC (numandar : positive := 8);  --Número de andares
18   PORT (IR          : IN  std_logic;
19         Clk         : IN  std_logic;
20         reset       : IN  std_logic;
21         SR          : IN  std_logic_vector(numandar-1 DOWNTO 0);
22         bit_rst     : OUT std_logic;
23         ControlMux  : OUT std_logic_vector(numandar-1 DOWNTO 0);
24         Proximo     : OUT std_logic_vector(2 DOWNTO 0));
25 END COMPONENT;
26 BEGIN
27   (...)
28 END Comportamental;
```

Figura 4.6: Código VHDL para os algoritmos C,U,D e S.

A figura 4.7 (a) apresenta o código VHDL simplificado que descreve as portas de entrada e saída da entidade que implementa um algoritmo específico. Os estados dos sinais de

entrada do sensor da porta (IR) e do registrador de chamadas (SR) são examinados através de condições específicas que determinam o funcionamento do algoritmo. A descrição dos quatro algoritmos implementados é baseada em máquinas de estados finitos de três estados (*Calcula, Acima e Abaixo*), como mostrado na figura 4.7(b). Neste caso, a diferença entre um algoritmo e outro é a forma e a ordem em que é examinado o registrador de chamadas.

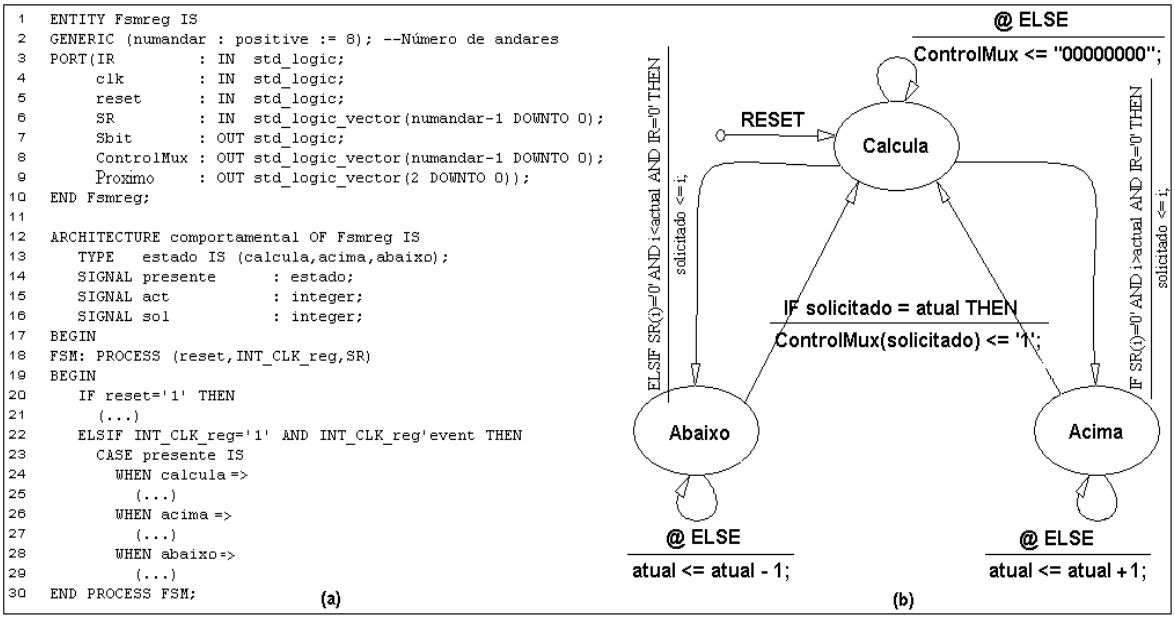


Figura 4.7: Implementação *Fsm_reg*: (a) Código VHDL (b) Máquina de estados finitos.

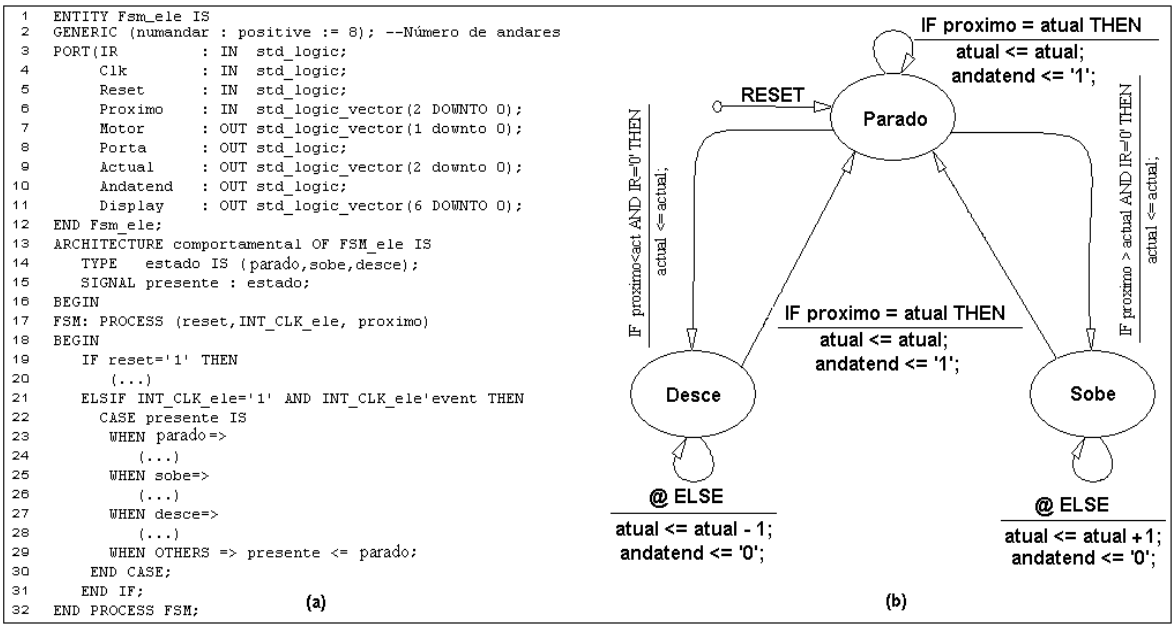
No algoritmo *Coletivo subida/Coletivo descida* (algoritmo C, vide seção 4.2.1) a busca de chamadas no registrador SR esta baseada numa estrutura cíclica, sendo realizada de forma seqüencial e alternada em ambas as direções (acima e abaixo).

No algoritmo *Coletivo subida/Seletivo descida* (algoritmo S) a estrutura cíclica implementa uma busca seqüencial de abaixo para acima começando da primeira célula na qual foi encontrado um registro de chamada. Isto é, um valor de "1" na célula, capturando a posição das células encontradas durante o percurso de subida pelo registrador. Logo em seguida uma nova procura é iniciada selecionando a célula mais baixa com valor "1".

A implementação do algoritmo *Seletivo subida/Coletivo descida* (algoritmo D) é inverso do algoritmo U, portanto a busca seqüencial é realizada de acima para abaixo. No algoritmo *Seletivo subida/Seletivo descida* (algoritmo S) não é realizada uma procura seqüencial, porém, são capturadas as posições das células dos extremos do registrador, para o qual é procurado o sinal "1" a partir da primeira célula do registrador (mais baixa) para depois alternar a busca desde a última célula (mais alta).

Uma vez que o *Fsm_reg* calcula qual é o próximo andar a ser visitado (sinal *próximo* na figura 4.5) dita informação é transmitida para o módulo *Fsm_ele.* Este módulo é uma máquina de estados (FSM) que implementa a execução da ordem, controlando o motor do elevador, o fechamento e abertura da porta e a apresentação em *displays* do andar atual e a direção do movimento.

A função da *Fsm_ele* é levar a cabina até o andar indicado pelo sinal *próximo.* A descrição considera três estados, (*Parado*, *Sobe* e *Desce*). O funcionamento, basicamente, consiste em comparar o andar atual com o próximo andar a ser atendido, previamente calculado pela *Fsm_reg.* A figura 4.8(a) apresenta a implementação simplificada deste módulo (em VHDL) para um caso edifício de oito andares. A representação da máquina de estados é mostrado na figura 4.8(b).



```
1  ENTITY Fsm_ele IS
2  GENERIC (numandar : positive := 8); --Número de andares
3  PORT(IR          : IN  std_logic;
4       Clk         : IN  std_logic;
5       Reset       : IN  std_logic;
6       Proximo     : IN  std_logic_vector(2 DOWNTO 0);
7       Motor       : OUT std_logic_vector(1 downto 0);
8       Porta       : OUT std_logic;
9       Actual      : OUT std_logic_vector(2 downto 0);
10      Andatend    : OUT std_logic;
11      Display     : OUT std_logic_vector(6 DOWNTO 0);
12 END Fsm_ele;
13 ARCHITECTURE comportamental OF FSM_ele IS
14    TYPE   estado IS (parado,sobe,desce);
15    SIGNAL presente : estado;
16 BEGIN
17 FSM: PROCESS (reset,INT_CLK_ele, proximo)
18 BEGIN
19    IF reset='1' THEN
20       (...)
21    ELSIF INT_CLK_ele='1' AND INT_CLK_ele'event THEN
22      CASE presente IS
23       WHEN parado=>
24          (...)
25       WHEN sobe=>
26          (...)
27       WHEN desce=>
28          (...)
29       WHEN OTHERS => presente <= parado;
30     END CASE;
31    END IF;
32 END PROCESS FSM;
```

(a)

(b)

Figura 4.8: Implementação *Fsm_ele*: (a) Código VHDL (b) Máquina de estados finitos.

4.2.4 - Implementação do algoritmo CS-ud

A função deste algoritmo é calcular qual é o próximo andar a ser visitado a partir dos sinais armazenados em dois subsistemas para armazenamento das chamadas (*Subsistema_Registrador)*. Um deles é usado para armazenar as chamadas de *botão-subir* e outro para as chamadas de *botão-descer*. Existem também dois componentes de *hardware* (*CoreElevador_S e CoreElevador_D*).

A figura 4.9 apresenta a arquitetura utilizada na implementação deste algoritmo, destacando-se a descrição de duas máquinas de estados finitos (*Fsm_reg_S e Fsm_reg_D*).

Estas máquinas determinam de forma concorrente (neste caso, com execução em paralelo) qual é o próximo andar a ser visitado segundo o estado do seu registrador.

Um terceiro módulo chamado *filtro* recebe e administra as informações provenientes das FSMs. A tarefa principal do módulo *filtro* consiste em determinar qual dos dois sinais provenientes da descrição das maquinas de estados finitos (*Fsm_reg_S* e *Fsm_reg_D*) será o próximo andar a ser visitado pelo carro do elevador. Desta forma é possível priorizar as chamadas num dos sentidos de atendimento. Desta maneira, quando todas as chamadas numa direção foram atendidas, então o módulo *filtro* transmite os sinais da outra FSM atendendo as chamadas no sentido oposto.

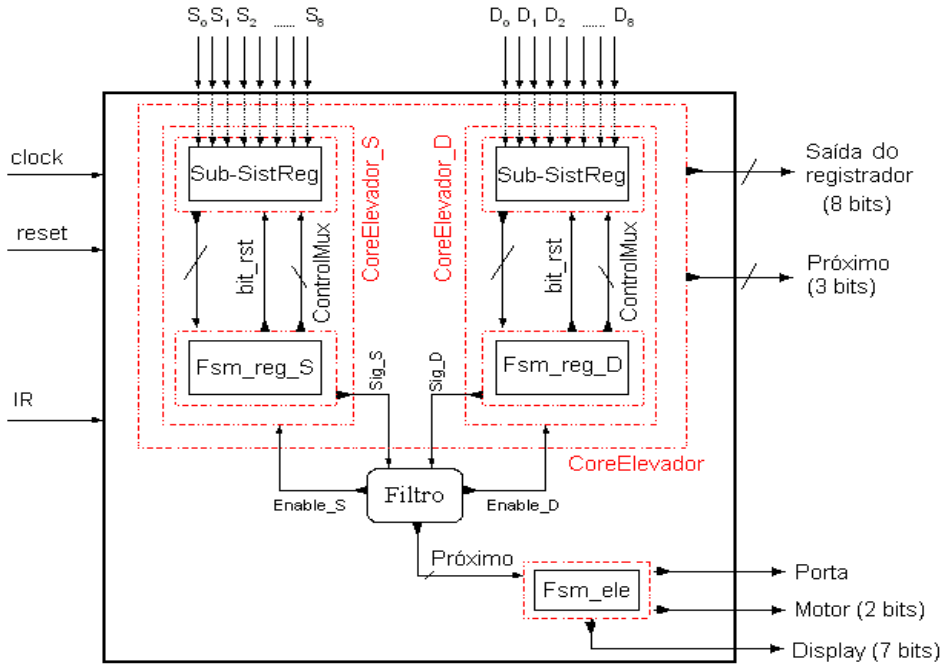


Figura 4.9: Arquitetura básica do algoritmo CS-ud.

A estratégia utilizada pelo módulo *filtro* permite que o algoritmo possa ser configurado para atender de forma eficiente os tráfegos majoritários de subida e de descida. Os sinais *enable_S e enable_D* habilitam as FSMs para transmitir o sinal que indica o próximo andar. Assim, quando uma das máquinas de estado esteja habilitada, o seu sinal *próximo* é transmitido ao módulo *Fsm_ele*. Entretanto a outra FSM está desabilitada para transmissão.

O módulo *Fsm_ele* deste algoritmo executa as mesmas funções que na arquitetura dos quatro algoritmos anteriormente explicados e a cuja descrição e implementação é mostrada na figura 4.8.

4.2.5 - Arquitetura geral do Sistema Integrado

A estrutura geral do sistema de controle local de elevadores é apresentada no diagrama de blocos da figura 4.10.

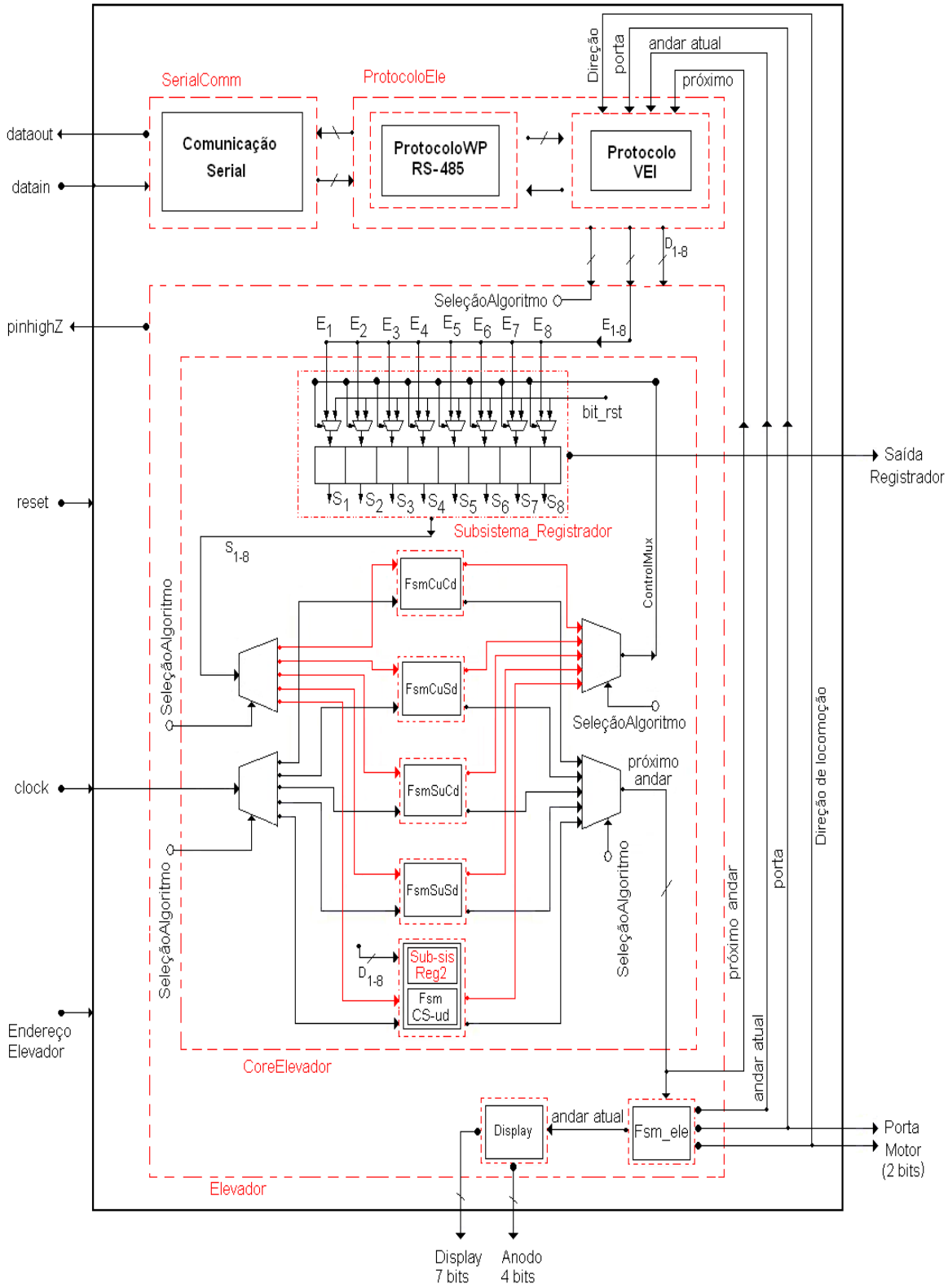


Figura 4.10: Arquitetura geral do LCS.

Na arquitetura existem três componentes principais que formam um nível hierárquico de fluxo de dados. Assim, o primeiro componente, chamado *SerialComm,* implementa a comunicação serial para a transmissão bidirecional de dados com o barramento externo. O componente *ProtocoloEle*, interpreta as informações convertendo-las em *comandos de ação* enviando-as ao terceiro componente (*Elevador*). Este último executa as tarefas pertinentes ao escalonamento do elevador.

O componente *Elevador* descreve os recursos de *hardware* necessários para a implementação do sistema de controle de movimento da cabina, do controle da porta e dos *displays* de apresentação do estado atual. Adicionalmente, um componente chamado *CoreElevador* implementa os algoritmos de escalonamento, os *subsistema_Registrador* para armazenamento de chamadas e um grupo de multiplexadores e demultiplexadores que permitem compartilhar os recursos de *hardware* descritos anteriormente.

Uma vez que acontece uma mudança no estado atual do elevador, o componente *Elevador* envia a informação necessária pelo barramento. Estes dados são transmitidos primeiramente ao módulo *ProtocoloEle*, quem constrói um *comando de estado* (compreensível pelo hospedeiro da rede RS-485), sendo finalmente enviado ao barramento pelo componente *SerialComm*.

Existem cinco módulos de escalonamento implementados em cada LCS. Cada um deles executa uma estratégia de atendimento de andares diferente. Entretanto, só é permitido que em cada LCS seja executado um tipo de escalonamento por vez. Para isto um conjunto de multiplexadores ativam uma das máquinas de estados segundo o sinal de controle *SeleçãoAlgoritmo* (vide figura 4.10). Desta forma é possível reconfigurar o algoritmo de atendimento a partir dos comandos enviados pelo barramento. Um conjunto de demultiplexadores direcionam os sinais de controle e de ação indicadas pelo algoritmo selecionado.

4.2.6 - Comunicação com o barramento externo

Como foi apresentado nas figuras 4.2 e 4.10, o sistema de controle local (LCS) contém, além do módulo *Elevador,* outros dois componentes denominados *ProtocoloEle* e *SerialComm*. A função destes dois componentes é interpretar os dados que entram no módulo *Elevador* e preparar os dados enviados ao barramento externo.

- O componente *ProtocoloEle* integra o entendimento dos comandos recebidos e a construção dos comandos a serem enviados pelo barramento. Este componente é composto por dois módulos *ProtocoloWP* e *ProtocoloVEI*. O primeiro deles administra os comandos correspondentes à função do LCS como escravo no contexto do padrão RS-485 (vide figura 4.1). Entre as funções principais que realiza este módulo é a de responder ao comando de requisição do estado do elevador. Para realizar isto, o primeiro passo é abaixar a impedância da rede, através do sinal de saída *pinhighZ*, indicando ao mestre que o LCS possui um dado pronto e que finalmente é enviado pelo barramento. Outra função do módulo é a de transmitir *comandos de ação* provenientes do mestre para o módulo de *ProtocoloVEI*.

  O *ProtocoloVEI* tem como função a distribuição dos *comandos de ação* dentro do componente *Elevador*. Para isto muda o estado das entradas, tais como, o estado da porta, registros de chamadas de subida e descida, entre outros. Uma segunda tarefa deste módulo é a construção dos comandos que são enviados pelo barramento no contexto da comunicação com a interface virtual de elevadores (VEI). Uma explicação mais detalhada é apresentada na seção 4.3.3.

- Os *comandos de ação e de estado* que trafegam pelo barramento são recebidos ou enviados de forma serial pelo componente *CommSerial*. Este componente implementa a comunicação serial de 8 bits de dados, 9600 bits por segundo (bps), sem paridade. A figura 4.11 mostra o pacote de transmissão bidirecional enviados pela interface serial implementada. A figura 4.12 apresenta a descrição da entidade e os processos da implementação.

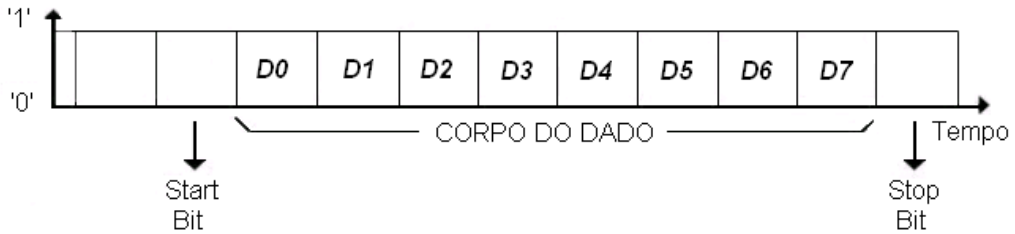


Figura 4.11: Pacote transmitido utilizando pela interface RS-232.

O primeiro processo realiza uma divisão de *clock* que ajusta a velocidade de transmissão em 9600 bps. O segundo processo utiliza os eventos do novo *clock* gerado (clk9600) com objetivo de paralelizar o dado serial proveniente do barramento externo. Finalmente, o

terceiro processo utiliza o mesmo evento de *clock* para enviar via serial o *comando de estado* do elevador encontrado em 10 vetores (registradores) (d1 até d10) cada um de 8 bits.

```
 1  ENTITY ComunicacionSerial IS
 2   PORT (
 3        din, clk50M, rst: IN  STD_LOGIC;
 4        d1,d2,d3,d4,d5,d6,d7,d8,d9,d10: IN STD_LOGIC_VECTOR (7 DOWNTO 0);
 5        load            : IN  STD_LOGIC;
 6        dado            : OUT STD_LOGIC_VECTOR (7 DOWNTO 0);
 7        dado_valido     : OUT STD_LOGIC;
 8        dout            : OUT STD_LOGIC);
 9  END ComunicacionSerial;
10
11  ARCHITECTURE rtl OF ComunicacionSerial IS
12    (...)
13  BEGIN
14  Gera9600bps: PROCESS(clk50M, rst)
15     BEGIN
16       (...)
17     END PROCESS Gera9600bps;
18  Paraleliza: PROCESS (rst, clk9600)
19    BEGIN
20      IF (rst='1') THEN
21        (...)
22      ELSIF (clk9600'EVENT AND clk9600='1') THEN
23        (...)
24      END IF;
25  END PROCES Paraleliza;
26  Serializa: PROCESS (clk9600, rst,load)
27    BEGIN
28      IF (rst='1') THEN
29        (...)
30      ELSIF (clk9600'EVENT AND clk9600='1') THEN
31        (...)
32      END IF;
33  END PROCESS Serializa;
34  END rtl;
```

Figura 4.12: Código VHDL da comunicação serial.

4.2.7 - Análise da implementação do sistema de controle local de elevadores

Conforme o explicado nas anteriores seções, considera-se que a implementação do sistema de elevadores proposto neste trabalho é baseado numa arquitetura distribuída (vide figura 4.1). Neste contexto cada LCS não só executa as tarefas referentes à locomoção do elevador, senão que também é capaz de configurar a estratégia de atendimento, mudando assim a ordem dos andares a serem visitados.

Esta arquitetura difere das implementações convencionais (Siiknen, 1997, Kim *et al.,* 1998, Yuan-Wei, 2000), sendo que o controlador de grupo (EGCS) não precisa realizar os cálculos do próximo andar a ser atendido. O sistema proposto é mais flexível, pois permite validar em tempo real o comportamento dos algoritmos de escalonamento dentro de um grupo de elevadores.

A tabela 4.1 resume de forma comparativa as vantagens do sistema proposto em relação às arquiteturas propostas por outros autores. São apresentadas as características das diferentes implementações mencionando o tipo de lógica para cálculo do próximo andar a ser visitado, os métodos usados para identificação de tráfego, o lugar no qual a lógica é desenvolvida (EGCS ou LCS), entre outras características.

Tabela 4.1: Diferenças Básicas na Implementação do Sistema de Elevadores.

Características, Empresa-Autor, Referência / Implementação	Mitsubishi (Tsuji, 1989)	Hitachi (Tobita, 1996)	Siikonen (Siikonen, 1997)	Kim (Kim, 1998)	Yuan Ho (Yuan Wei, 2000)	Sistema Proposto
Método e local da identificação de padrões de tráfego.	Não	Não	Lógica nebulosa EGCS	Lógica nebulosa EGCS	Redes Neurais EGCS	Lógica nebulosa EGCS
Método e local do cálculo do melhor elevador. (despacho).	Lógica nebulosa EGCS	Algoritmos Genéticos EGCS	Função de Custo EGCS	Função de Custo EGCS	Redes Petri Coloridas EGCS	Lógica nebulosa EGCS
Local do cálculo do próximo andar. (escalonamento).	Baseado em *Software* EGCS	Baseado em *Software* EGCS	Baseado em *Software* EGCS	Baseado em *Software* EGCS	Baseado em *Software* EGCS	Baseado em *Hardware* LCS
Aplicação e local da implementação de múltiplos algoritmos.	Não	Não	Não	Não	Sim *Software* EGCS	Sim *Hardware* LCS
Cálculo do melhor algoritmo de atendimento	---	---	---	---	Sim EGCS	Sim EGCS
Predição de Tráfego	Não	Não	Não	Não	Sim	Não

Existem algumas diferencias básicas na arquitetura do sistema de elevadores proposto e as implementações encontradas na literatura. Aparte da descrição em *hardware* do Sistema de Controle Local (LCS), a diferencia básica consiste em um processamento distribuído que permite diminuir os cálculos realizados pelo controlador de grupo e ao mesmo tempo a quantidade de dados que trafegam pela rede.

4.2.8 - Análise do custo/desempenho da implementação baseado em complexidade

Os recursos de *hardware* utilizados e o custo de área da implementação estão diretamente relacionados com a complexidade da arquitetura (Akl, 1997). Neste caso, o custo de área da implementação do registrador de chamadas é $O(n)$, onde n é o número de andares. Entretanto, o custo de área da descrição das máquinas de estados finitos (O_{FSM}) é proporcional à codificação dos estados, isto é $O(\log n)$, mais o custo de área dos recursos combinacionais (O_c). Portanto:

$$O_{FSM} = O(\log(n)) + O_c, \tag{4.1}$$

No caso específico das maquinas de estado que descrevem os algoritmos de escalonamento (*Fsm_reg)*, o custo de área é maior devido à utilização de um laço de repetição que implementa uma busca seqüencial no registrador de andares. Neste caso, os recursos de *hardware* consumidos são um comparador de 1 bit, *O(1)*, e um multiplexador de oito entradas com um custo de área proporcional à expressão:

$$O(n \log(n)) \tag{4.2}$$

Enquanto à análise de desempenho (*timming)* é possível concluir que o custo de recolher as chamadas de pavimento é *O(1)*. Entretanto, numa implementação em *software* este custo poderia ter uma complexidade de *O(n),* sendo *n* o tamanho do registrador. O tempo necessário para determinar o próximo andar a ser visitado é $\Omega(n)$, sendo *n* é o número de andares (devido a que foi implementado um laço *FOR* para realizar a procura no registrador de andares).

Dado que os três módulos (*subsistema_Registrador*, *Fsm_reg* e *Fsm_ele*) rodam de forma concorrente, a complexidade geral será o pior caso. Neste caso, o custo da *Fsm_reg* é maior do que o custo do *Fsm_ele* dado que a primeira máquina de estados (*Fsm_reg*) deve calcular o próximo andar a ser visitado.

A busca seqüencial implementada na descrição das *Fsm_reg* dos primeiros quatro algoritmos considera um laço *FOR* aninhado com uma condição *IF*. Já no caso do último algoritmo (CS-ud), o custo de área dos dois registradores é 2 *O(n)* e o custo de área de cada FSM e de cada conjunto comparador – multiplexador, é similar ao indicado nas expressões 4.1 e 4.2 respectivamente. Uma possível implementação em *hardware* para a descrição da busca seqüencial, é mostrada na figura 4.13.

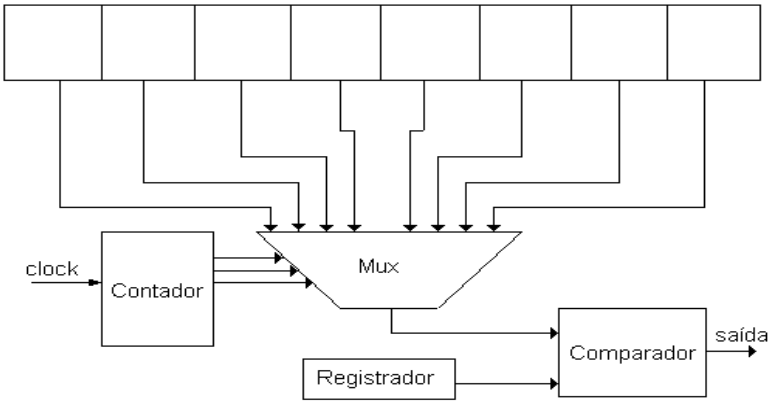


Figura 4.13: Estrutura em *hardwa*re para uma busca seqüencial.

4.3 - INTERFACE VIRTUAL DE ELEVADORES E CONTROLE DE GRUPO

A *Interface Virtual de Elevadores* (VEI) foi desenvolvida em linguagem *Java* e roda em PCs independente da plataforma. O VEI é usado para realizar as tarefas de controle de grupo e emular o sistema de elevadores, sendo uma ferramenta flexível em relação à configuração dos parâmetros do edifício e dos elevadores, tais como o número de andares, numero de elevadores, velocidade dos elevadores, tipo de edificação, entre outros.

O VEI é conectado à rede RS-485 através do conversor RS 232/485, como foi apresentado na figura 4.1. Inicialmente, o VEI envia *comandos de estado* atual do sistema, através de eventos que acontecem na sua interface. Logo em seguida cada LCS processa esta informação e envia de volta o estado atual do elevador respectivo. Finalmente, os dados recebidos pelo VEI são utilizados para desenhar a animação e apresentar em tempo real os tempos de espera e locomoção. Um protocolo de comunicação administra os dados que trafegam pela rede, entre os LCSs e o VEI.

De acordo com as funcionalidades, o sistema VEI é dividido em três módulos principais: módulo *Tráfego*, módulo *Desenho Prédio* e módulo *Comunicação Serial*, como mostrado na figura 4.14.

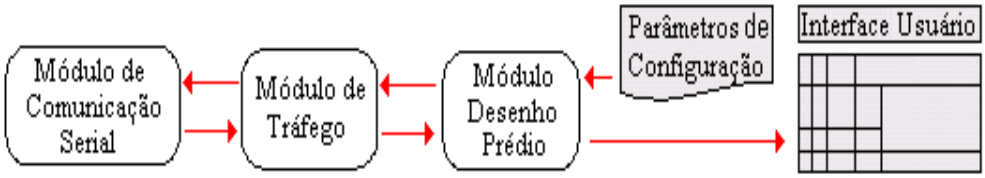


Figura 4.14: Estrutura funcional do VEI.

4.3.1 - Módulo de tráfego: sistema de inferência nebuloso para o controle de grupo

Um método heurístico, baseado em lógica nebulosa, foi implementado neste módulo com o objetivo de identificar o padrão de tráfego atual no edifício e selecionar o elevador mais conveniente para atender uma chamada, ou grupo de chamadas, de pavimento.

Foram implementados alguns métodos para preparar os dados de entrada do sistema de inferência nebuloso. Estes métodos utilizam, além dos eventos gerados na interface de usuário, os valores atuais de posição, direção e chamadas já designadas para cada elevador. Isto é, o estado do elevador, assim como também o padrão de tráfego atual, a distribuição

de passageiros no edifício e o tempo expressado em horas e minutos (vide figura 4.4). A finalidades destes métodos consideram a estimação das seguintes variáveis:

a) **Chamadas de subida:** indica o número de chamadas de pavimento na direção subir.
b) **Chamadas de descida:** indica o número de chamadas de pavimento na direção descer.
c) **Porcentagem de entradas centralizadas:** indica qual/quais do(s) andar(es) no edifício esta/estão sendo centralizado(s). Esta medida é realizada a partir das chamadas de cabina e é um valor percentual da razão de passageiros entrando no andar mais lotado com respeito aos demais andares no edifício. Isto é uma medida de qual andar é mais solicitado.
d) **Porcentagem de saídas distribuídas:** é um valor percentual da razão dos passageiros saindo de todos os andares, exceto do mais lotado, com respeito aos demais andares no edifício. Esta medida é realizada a partir das chamadas de cabina que indica o grau de distribuição dos passageiros no edifício.
e) **Tempo de espera:** é um cálculo estimado do tempo desde o momento em que é realizada uma chamada de pavimento até o elevador chegar ao pavimento indicado, abrindo as portas e permitindo a ingresso dos passageiros. Esta medida depende diretamente dos parâmetros de configuração do sistema de elevadores tais como, velocidade do elevador, relação de aceleração, altura dos andares e o tempo de abrir e fechar a porta. O *tempo de espera* depende também do número de paradas que deve realizar o elevador antes de chegar ao pavimento solicitado.
f) **Número de paradas:** este é o número de paradas que deve realizar o elevador antes de chegar num pavimento solicitado.

As seis variáveis calculadas formam parte das entradas do sistema de controle de grupo de elevadores nebuloso (FEGCS – *Fuzzy Elevator Group Control System*) implementado. A figura 4.15 apresenta a configuração do FEGCS proposto, o qual é composto por duas máquinas de inferência, *Identificador de Tráfego* e *Despacho de Elevadores*. Existe um terceiro módulo (não nebuloso), denominado *módulo de mapeamento*, que acopla as saídas do sistema nebuloso com a interface virtual de elevadores.

Quando é realizada uma chamada de pavimento, as variáveis *Número de paradas* e *Tempo*

de espera, são calculadas para cada elevador. Assim, mantendo o valor das demais entradas, todo o conjunto de variáveis são apresentadas ao sistema nebuloso. Desta forma é calculado um valor de conveniência para cada elevador.

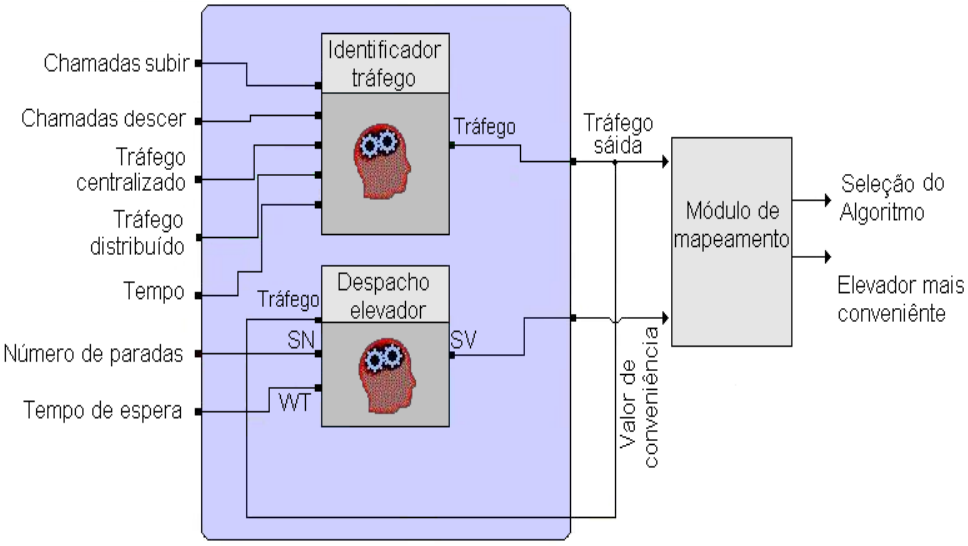


Figura 4.15: Módulos do FEGCS.

- *Identificador de tráfego*: esta máquina de inferência identifica o tráfego atual, em tempo real, selecionando um dos oito possíveis padrões de tráfego descritos na tabela 2.2: tráfego de pico de subida (*Up*), tráfego em tempo de negócios (*BT*), tráfego em tempo de almoço A e B (*LTa*) e (*LTb*), tráfego de combinação negócios/pesado (*BTH*), tráfego pesado (*HT*), tráfego de pico de descida (*Down*) e tráfego em tempo de inatividade (*IT*). Só um daqueles oito padrões poderá ser transmitido na saída *tipo de tráfego*. Este módulo utiliza como variáveis lingüísticas de entrada as características de tráfego apresentadas na tabela 2.1.

- *Despacho de Elevadores:*esta máquina de inferência entrega qual é o elevador mais indicado para atender uma chamada ou grupo de chamadas de pavimento. Para esta tarefa são importantes três variáveis lingüísticas: o *Tráfego atual,* realimentado do módulo Identificador de Tráfego, o *Tempo de espera e Número de paradas*.

- *Módulo de mapeamento:* A partir do padrão de tráfego identificado, o módulo de mapeamento escolhe entre os cinco possíveis algoritmos de escalonamento implementados nos LCSs. Além disto, é escolhido o elevador com maior valor de conveniência, sendo que o sistema nebuloso entrega um valor de conveniência para cada elevador.

As variáveis lingüísticas e as funções de pertinência que descrevem o sistema nebuloso são apresentadas na figura 4.16 e são relacionadas a seguir:

- *Quantidade de tráfego (TA):* Representa valores de quantidade, neste caso o número de chamadas para subir e número de chamadas para descer, portanto é utilizada para calcular o grau de pertinência das entradas *chamadas de subida* e *chamadas de descida*. O universo de discurso abrange valores desde zero até 10 chamadas de pavimento. Possui três funções de pertinência com termos lingüísticos pequeno, mediano e grande.

- *Porcentagem de tráfego (TP):* Representa valores percentuais, é utilizadas para calcular o grau de pertinência das entradas *tráfego centralizado* (*CITP*) e *tráfego distribuído* (*DOTP*). O *universo de discurso* compreende valores entre 0 e 100 %. Possui três funções de pertinência com termos lingüísticos pequeno, mediano e grande.

- *Tempo:* Esta variável lingüística representa o tempo atual medido em horas-minutos. É utilizada para calcular o valor de pertinência da variável de entrada *tempo*. O universo de discurso considera o tempo operacional do prédio, neste caso desde as 5 até as 22 horas. Entretanto, estes valores dependem da funcionalidade do edifício. Esta variável possui sete funções de pertinência: *tempo inativo um* (Tin1), *tempo de subida* (Tup), *tempo de negócios um* (Tbt1), *tempo de comida* (Tln), t*empo de negócios dois* (Tbt2), *tempo de descida* (Tdown) e *tempo inativo dois* (Tin2).

- *Número de paradas (SN):* O número de paradas é uma medida da potência de consumo do sistema do elevador. Esta variável lingüística representa o número de paradas que um elevador deve realizar antes de atingir o andar onde foi realizada uma nova chamada de pavimento. É utilizada para calcular o grau de pertinência da entrada (*Número de paradas*). O universo de discurso compreende valores entre 0 e 10 paradas, sendo este valor alterado pelo número de andares do edifício. Esta variável possui três funções de pertinência com termos lingüísticos pequeno, mediano e grande.

- *Tempo de espera (WT):* O tempo de espera é uma medida do conforto dos usuários no edifício. Esta variável representa o tempo estimado desde o momento em que é realizada uma chamada de pavimento até o momento em que o elevador abre

completamente a porta no pavimento indicado. É utilizada para medir o grau de pertinência da entrada *Tempo de espera*. O universo de discurso foi estipulado entre um valor mínimo de 15 segundos (tempo total de locomoção até um andar vizinho) e o tempo máximo de 400 segundos, sendo que este valor depende diretamente do número de andares e das características físicas do elevador. Possui três funções de pertinência com termos lingüísticos pequeno, mediano e grande.

- *Tráfego:* A variável de saída da primeira máquina de inferência que define o tipo de tráfego no edifício através de oito padrões de tráfego identificados por oito funções de pertinência: Up, BT, LTA, LTB, BTH, HT, Down e IT. O universo de discurso considera os oito padrões [1 até 8].

- *Valor de Conveniência (SV):* É a variável lingüística de saída da segunda máquina de inferência. Representa o valor de conveniência de cada elevador quando é realizada uma nova chamada, ou novas chamadas. Esta variável é utilizada para calcular o grau de pertinência da variável de saída *valor de conveniência*. O universo de discurso compreende valores entre 1 e 10, portanto esta variável caracteriza o elevador através de uma menção numérica (entre 1 e 10), assim, entre maior a menção, maior será a conveniência de um elevador para o atendimento das novas chamadas. Possui dez funções de pertinência: SV1, SV2, SV3, SV4, SV5, SV6, SV7, SV8, SV9 e SV10.

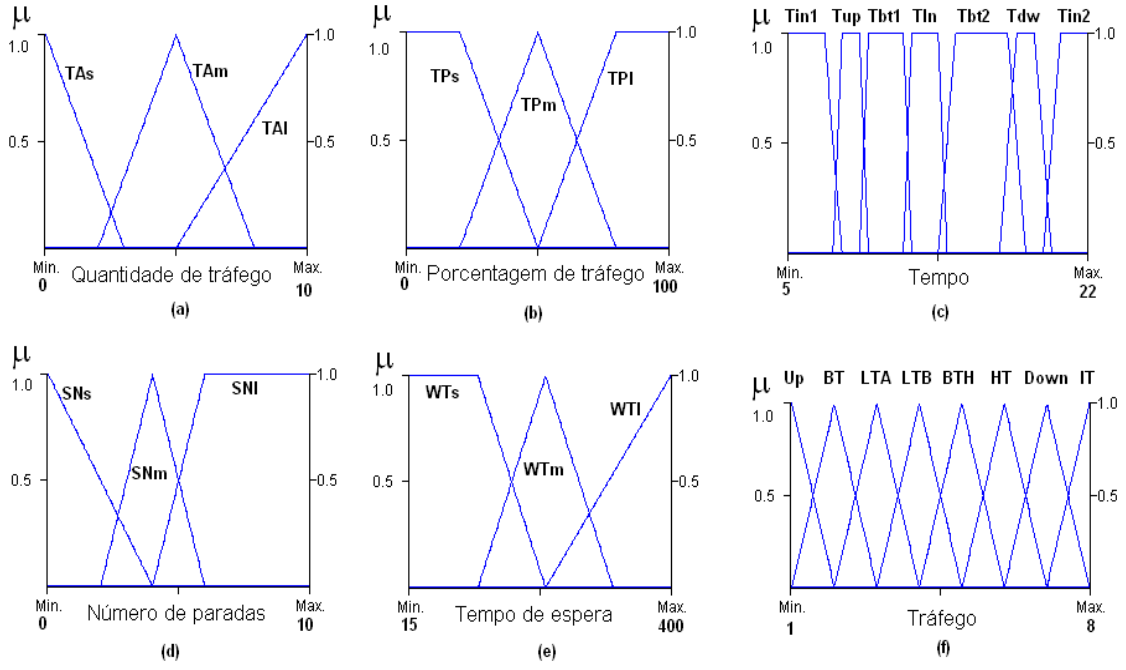


Figura 4.16: Funções de pertinência do FEGCS.

O conjunto de saídas, *tipo de tráfego* e *valor de conveniência,* compõem a decisão tomada pelo controlador de grupo visando diminuir o tempo de espera e/ou a potência de consumo do sistema de elevadores do edifício. Por exemplo, quando o tráfego é leve, isto é uma saída identificada pelos padrões BT (*Business Time*) ou IT (*Inactive Time*), o FEGCS procura manter parados o maior número possível de elevadores, diminuindo assim a potência de consumo. Já no caso de ser identificado um padrão de tráfego diferente, o FEGCS procura manter ligados o maior número possível de elevadores, rodando em cada LCS um algoritmo específicos, no intuito de aumentar a eficiência do serviço de transporte.

4.3.2 - Módulo de desenho do prédio

O módulo de *Desenho do prédio* é uma interface entre o *Módulo de tráfego* e o usuário (vide figura 4.14. O objetivo desta interface é emular as características encontradas em um sistema de elevadores real. Esta implementação considera a construção de pavimentos, botões de pavimento, cabinas, botões de cabina, *displays* de apresentação e outros componentes presentes em situações reais.

Um dos métodos implementados neste módulo é a construção de um prédio virtual definido através da informação fornecida nos parâmetros iniciais de configuração: número de andares, número de elevadores, tipo de edifício (comercial ou residencial), entre outros. Outra tarefa relevante é a apresentação do estado atual do sistema em uma *Matriz de Estado,* a qual armazena a informação necessária, na execução de outros métodos.

As informações adquiridas a partir da *Matriz de Estado*, são úteis no desenho do estado atual de cada elevador, apresentando aos usuários o estado do sistema de elevadores através de uma interface gráfica. Esta interface contem objetos que simulam os componentes do sistema e utiliza métodos de tratamento de eventos para emular as ações geradas num sistema real. Todos estes eventos geram informações que são transmitidas ao *Módulo de Tráfego* e ao mundo exterior através do módulo que implementa a comunicação serial (vide figura 4.14).

4.3.3 - Módulo de comunicação serial

- Todos os eventos tratados nas interfaces do sistema de elevadores têm uma ação em um ou mais controles locais (LCS). Parâmetros de configuração e eventos nos botões

de chamadas compõem uma interface que interatua entre o usuário e o LCS. Para realizar esta interação de elementos funcionais utiliza-se o barramento externo RS-485 e um módulo de comunicação serial.

Foram utilizadas as bibliotecas padrão de *Java* e a utilização de classes para a descrição da comunicação serial, de forma que a configuração da transmissão dos dados fosse flexível, permitindo ajustar a taxa de transmissão, o número de bits de dados e a paridade. Neste caso, a configuração utilizada na interface virtual de elevadores (VEI) deve coincidir com a configuração implementada em *hardware* nos controle locais (LCS), garantindo a compatibilidade entre as duas partes. Portanto, a configuração padrão é uma transmissão a 9600 bps, 8 bits de dados, sem paridade e um bit de parada (vide figura 4.10).

A comunicação entre os LCSs e a interface virtual é realizada através do protocolo VEI, implementado em ambas as partes no contexto mestre/escravo. Este protocolo foi construído da forma mais simples possível, perguntas e respostas, considerando *comandos de ação* e *comandos de estado*, lembrando que a transmissão é baseada em pacotes de 1 *byte* de dados.

- ***Comandos de ação***: são comandos enviados desde o VEI para o LCS. A sua função é codificar as rotinas das possíveis ações que o controle local é capaz de executar. Existem sete *comandos de ação* no protocolo VEI, cada um representa uma tarefa para ser realizada. Os sete comandos são compostos de 5 bytes, cada um, onde o primeiro byte indica o começo da transmissão, o segundo byte identifica o LCS ao qual é dirigida uma ação específica que é representada pelos terceiro e quarto bytes. O último byte indica o fim da transmissão. Estes comandos são relacionados na tabela 4.2.

- ***Comando de estado*:** é um comando enviado desde os LCS para o VEI. Existe um único *comando de estado* que codifica o estado atual de um elevador, incluindo informação do andar atual, qual é o próximo andar calculado (no caso de existir), o estado da porta (aberta ou fechada) e a direção de movimento do elevador (parado, subindo ou descendo). O *comando de estado* é enviado pelos LCSs quando é solicitada a petição, por parte do VEI, ou quando acontece alguma mudança no estado de um ou mais elevadores. Este comando é composto de 10 bytes:

Os primeiros dois *bytes* identificam a fonte do comando, isto é o elevador que esta realizando a transmissão. O terceiro e quarto *bytes* representam o andar atual, os quinto e sexto *bytes* indicam qual é o próximo andar a ser atendido e últimos quatro *bytes* representam o estado da porta e a direção de locomoção do elevador. <ExAxNxPxDx>

Tabela 4.2: Comandos de ação enviados desde o VEI para o LCS.

Comando	Exemplo	Ação
Comando de chamada de subida	*<ExSw.>*	Elevador *x* registra chamada de subida no andar *w*.
Comando de chamada de descida	*<ExDw.>*	Elevador *x* registra chamada de descida no andar *w*.
Comando de chamada de cabina	*<ExCw.>*	Elevador *x* registra chamada de cabina para o andar *w*.
Comando de mudança algoritmo	*<Exuu.>*	Elevador *x* roda algoritmo *u*.
Comando abrir porta	*<Exaa.>*	Abre a porta do elevador *x*, sempre que o elevador *x* esteja parado.
Comando fechar porta	*<Exff.>*	Fecha a porta do elevador *x*.
Comando de reset do LCS	*<Exrr.>*	Reinicia o elevador *x*, apagando registradores e variáveis de estado.
Comando de petição de estado	*<Exss.>*	O VEI solicita o comando de estado do elevador *x*.

A figura 4.17 apresenta a interface de usuário VEI que permite a interação do sistema virtual de elevadores com os controles locais.

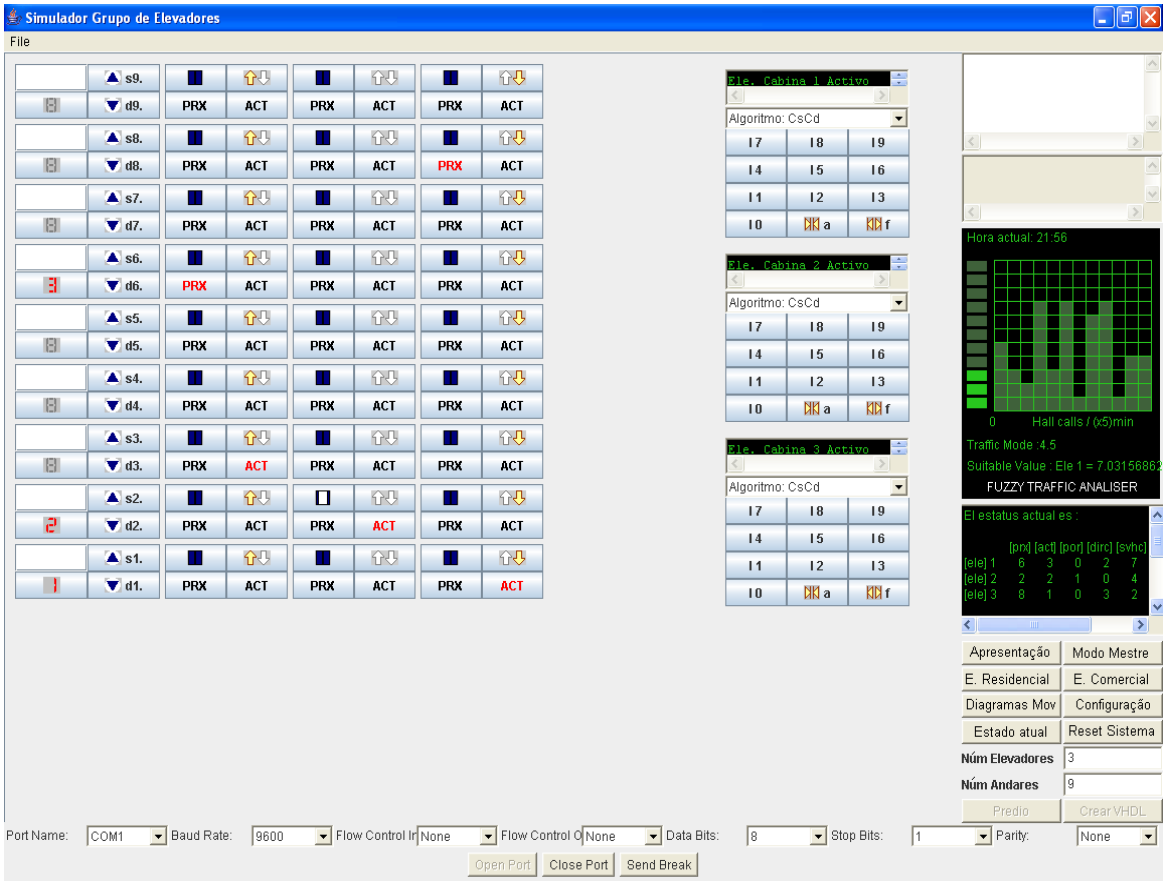


Figura 4.17: Interface de usuário do VEI.

4.4. - CONCLUSÃO DO CAPÍTULO

Neste capítulo foi descrita a arquitetura proposta para a implementação do Sistema de Controle Local (LCS). Cinco algoritmos de escalonamento de elevadores foram implementados diretamente em *hardware*, num ambiente integrado que permite compartir alguns recursos de *hardware*, reduzindo a área e o consumo de potência.

O Sistema de Controle de Grupo de Elevadores (EGCS), baseado em lógica nebulosa (FEGCS) e desenvolvido em linguagem *Java*, permite validar em tempo real o comportamento dos algoritmos de escalonamento dentro de um grupo de elevadores.

As vantagens do sistema de elevadores proposto sobre outras implementações são relacionadas a seguir:

a) Na arquitetura proposta cada LCS não só executa as tarefas referentes à locomoção do elevador, senão que também é capaz de configurar a estratégia de atendimento, mudando assim a ordem dos andares a serem visitados. Isto possibilita que o EGCS escolha o algoritmo mais apropriado para rodar em cada LCS, conseqüentemente, a implementação resulta mais flexível.

b) O sistema de controle de grupo não precisa realizar os cálculos do próximo andar a ser atendido, isto resulta na diminuição dos cálculos realizados pelo EGCS.

c) O sistema proposto realiza um processamento distribuído implementando mecanismos de despacho para gerenciar um grupo de elevadores. Entretanto, a implementação em *hardware* dos algoritmos de escalonamento oferece maior desempenho no tempo de cálculo do próximo andar a ser visitado por cada elevador.

A comunicação entre os Sistemas de Controle Local (LCSs) e o Sistema Nebuloso de Controle de Grupo de Elevadores (FEGCS) é realizada através do padrão RS-485, para o qual alguns módulos de protocolo foram desenvolvidos em *hardware*.

5 - TESTES E APLICAÇÃO DO SISTEMA DE ELEVADORES

Os componentes funcionais do sistema de elevadores proposto e implementado neste trabalho permite a construção de grupos de elevadores considerando a sua aplicação em edifícios de configuração particular. Este capítulo apresenta a aplicação do sistema proposto para o caso de um edifício de oito pavimentos (padrão arquitetônico de Brasília), com um grupo de três elevadores. Primeiramente, são considerados os resultados de síntese lógica do sistema e simulações funcionais do controle local de elevadores (LCS) prévios à implementação final. Em seguida, são analisadas as curvas de controle do sistema nebuloso de controle do grupo de elevadores (FEGCS). Finalmente são apresentados os mapas comparativos de escalonamento obtidos para cada um dos algoritmos e as gráficas de tempo de atendimento dos usuários.

5.1 - SÍNTESE E SIMULAÇÃO FUNCIONAL DO CONTROLE LOCAL

O desenvolvimento dos sistemas de controle local de elevadores (LCS), foi realizado utilizando a ferramenta *ISE7.1* e placas FPGA XC3S200 da família Spartan3, fornecidos pela empresa *Xilinx* (Xilinxb, 2005). A linguagem de descrição de *hardware* (VHDL) foi utilizada para construir o projeto de um LCS segundo as implementações apresentadas na seção 4.2. Foi realizada a síntese do projeto e os sinais de entrada e saída foram mapeados na placa FPGA Spartan3.

Os recursos de *hardware* utilizados na implementação do sistema de controle local de elevadores para um edifício de oito andares, são apresentados na tabela 5.1. Nesta tabela é possível observar o número de elementos lógicos consumidos pelo LCS e a freqüência máxima de trabalho do sistema embarcado numa FPGA XC3S200 da família Spartan3 da *Xilinx*, onde a taxa de ocupação dos elementos lógicos é referente ao consumo de *Lookup Tables* (LTU) de 4 entradas. As informações obtidas na tabela foram obtidas a partir dos relatórios de compilação fornecidos pelo *software* de desenvolvimento *ISE* 7.1 da *Xilinx.*

A implementação do sistema de controle local de elevadores consume 46% dos recursos disponíveis. Na tabela 5.2 é relacionado o custo e o desempenho (consumo de recursos e freqüência máxima do *clock*) de cada um dos componentes que fazem parte da

implementação do LCS. Pode ser observando que, o consumo de LTUs do componente registrador é igual a oito, no caso da implementação do LCS de 8 andares, portanto o consumo de este tipo de recurso é proporcional ao número de andares.

Tabela 5.1: Resultados de síntese para implementação do LCS

Dispositivo	**4 input LUT** ***Max:* 3,840**	**Slice FFs** ***Max:* 3,840**	**IOBs** ***Max:* 173**	**Freqüência máxima**
XC3S200 Spartan 3	1767	507	25	70,641 MHz

Tabela 5.2: Resultados de síntese dos componentes do LCS

Componente	**4 input LUT** ***Max:* 3,840**	**Slice FFs** ***Max:* 3,840**	**IOBs** ***Max:* 173**	**Freqüência máxima (MHz)**
Subsistema_Registrador	8	8	27	_
Fsm_CuCd (Coletivo subida / Coletivo descida)	338	53	23	84.717
Fsm_CuSd (Coletivo subida / Seletivo descida)	372	52	23	73.992
Fsm_SuCd (Seletivo subida / Coletivo descida)	377	52	23	77.375
Fsm_SuSd (Seletivo subida / Seletivo descida)	358	81	23	81.679
Fsm_CSud (Coletivo - Seletivo / subida – descida)	821	168	120	73.899
Fsm_ele	60	36	13	134.264
SerialComm	123	115	78	158.028
Protocolo VEI	111	66	37	103.455
Protocolo _ RS-485	45	19	28	271.098

Entretanto, a taxa de consumo em LUTs dos algoritmos de escalonamento é maior em relação aos outros componentes devido à implementação de um laço *FOR* aninhado em uma condição *IF* como estrutura de calculo do próximo andar a ser visitado.

A implementação do Sistema de Controle Local (LCS) em FPGAs permite reduzir o consumo de área, dado que os algoritmos de escalonamento compartilham vários recursos de *hardware* (vide figura 4.10). O *subsistema_Registrador*, a máquina de estados para movimentação do elevador (*Fsm_ele*) e os módulos de comunicação são compartilhados pelos algoritmos de escalonamento. Por outro lado, quando um algoritmo é selecionado, os outros quatro algoritmos são desabilitados parando os sinais de *clock* correspondentes, o que permite reduzir o consumo de potência.

No intuito de um primeiro teste de funcionamento, foram realizadas as simulações funcionais dos componentes principais: registrador de chamadas, algoritmos de

escalonamento e finalmente do sistema integrado. Utilizou-se a ferramenta *ModelSim* (ModelSim, 2006) para este fim.

A figura 5.1 apresenta a simulação funcional do *Sub-sistema_Registrador* implementado segundo a arquitetura mostrada na figura 4.4. As chamadas de pavimento correspondem ao sinal de entrada "*andar*". A entrada de controle dos multiplexadores é representada pelo sinal "*bit_controle*" e a saída do registrador de chamadas pode ser conferida no sinal "*saida_reg*".

O sinal "*bit_controle*" possui um valor predeterminado de "0" permitindo que sempre que uma chamada de pavimento for realizada, isto é um pulso ativo em "0", o andar correspondente seja armazenado no registrador. Neste caso, os pavimentos [8, 6, 4, 3] foram solicitados e devidamente armazenados. Durante o tempo total da simulação o sinal "*Apaga_bit*" permanece em "1" permitindo apagar o registro de um andar sempre que o sinal "*bit_controle*" selecionar o andar correspondente.

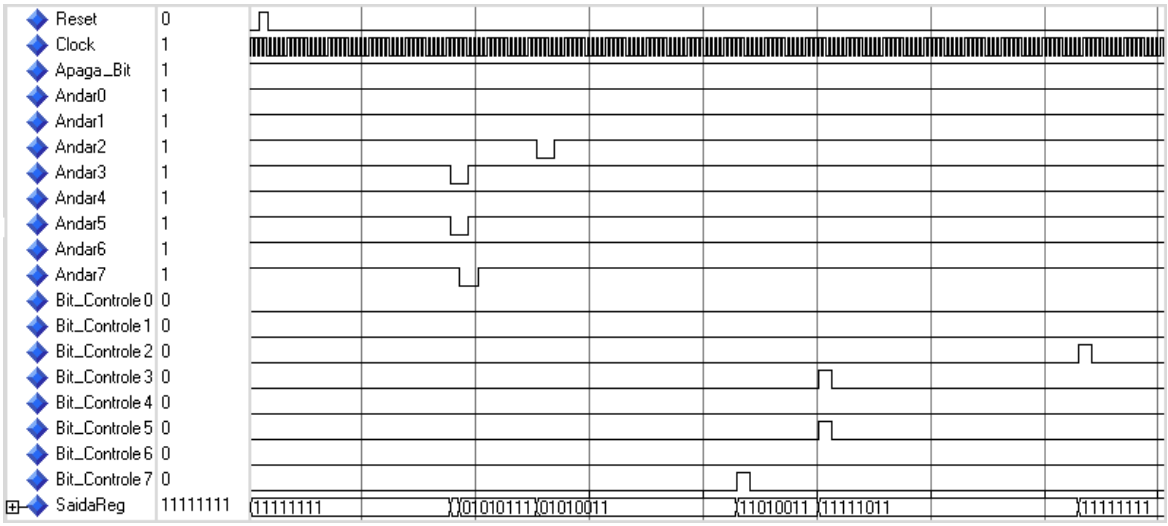


Figura 5.1: Simulação Funcional *subsistema_Registrador.*

Foi realizada a simulação funcional dos primeiros quatro algoritmos (C, U, D e S), os quais utilizam apenas um botão para seleção de chamada. Estes algoritmos foram implementados conforme a estrutura apresentada na figura 4.5, que basicamente é composta por um *Sub-sistema_Registrador* e duas maquinas de estados finitos, *Fsm_reg* e *Fsm_ele*.

A figura 5.2 apresenta a simulação do algoritmo *Coletivo subida/Seletivo descida* (algoritmo U). A entrada do sistema é composta por um vetor de 8 bits (sinal *pavimentos*),

que representa os oito botões de pavimento (oito andares).

Inicialmente, com o elevador no primeiro andar e a porta aberta, são selecionados os andares quarto e sexto, sendo eles atendidos durante a viagem de subida utilizando o principio coletivo. Uma vez no sexto andar, são realizadas as novas chamadas de pavimento, terceiro e quinto andar, sendo o terceiro andar atendido durante a viagem de descida aplicando o principio seletivo. Finalmente o sistema aplica novamente o principio coletivo atendendo o quinto andar durante a próxima viagem de subida.

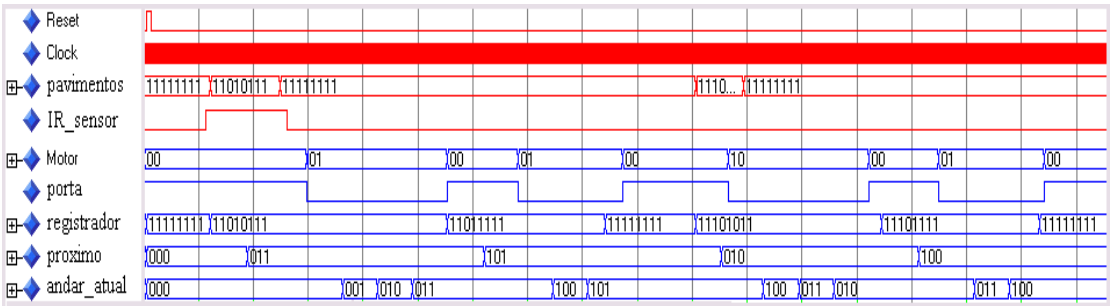


Figura 5.2: Simulação Funcional do algoritmo U.

A figura 5.3 apresenta a simulação funcional do algoritmo *CS-ud*, baseado na arquitetura mostrada na figura 4.9. Neste caso existem dois elementos *sub-sistema_Registrador* para armazenar as chamadas de pavimento nas direções subir e descer, representadas pelos sinais *chamada_sobe* e *chamada_desce,* respectivamente.

Inicialmente o sistema espera as solicitações nos andares, entretanto o elevador permanece parado no primeiro andar com a porta fechada. No intuito de comparar as diferenças de escalonamento entre chamadas para subir e para descer, foram solicitadas ao mesmo tempo as chamadas representadas nos sinais *chamada_sobe* e *chamada_desce.* O segundo e o sexto andar fizeram uma solicitação de subida. Entretanto, os andares quarto e oitavo fizeram uma solicitação de descida. Neste caso, foram atendidas primeiramente as chamadas para subir a traves do principio coletivo, posteriormente o sistema escolhe (parte seletiva) a chamada mais alta do prédio que realizou uma solicitação de descida, isto é, o oitavo andar. Finalmente o elevador atende as chamadas de descida de forma coletiva, terminando a simulação no atendimento do quarto andar.

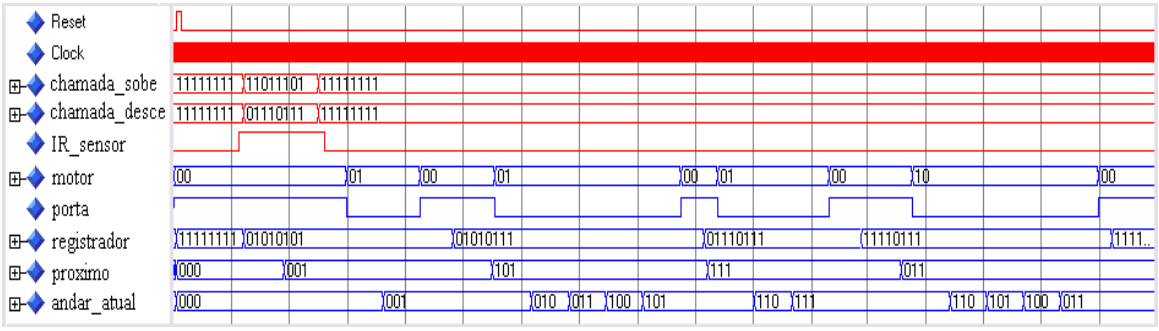


Figura 5.3: Simulação Funcional CSud.

A simulação funcional do sistema de controle local de elevadores (LCS) desenvolvido neste trabalho é apresentada na figura 5.4. Nesta simulação, baseada na arquitetura do LCS (figura 4.10), o sinal "*endereço elevador*" foi configurada no valor "001" avisando ao elevador 1 a execução dos *comandos de ação*. Entretanto, o sinal de entrada "*seleção algoritmo*" foi configurado no valor "000" selecionando o algoritmo *Coletivo subida/Coletivo descida.*

Para efeitos demonstrativos as primeiras três chamadas de pavimento foram feitas no estado de porta aberta (sinal *estado_porta).* As chamadas são na sua ordem de execução: quarto, sexto e sétimo andar, sendo armazenadas no registrador (sinal *registrador*). Sempre que uma nova chamada de pavimento acontecer, o sistema calcula o próximo andar a ser visitado (sinal *próximo*) segundo o algoritmo *Coletivo subida/Coletivo descida.* Porém, por efeitos de segurança, o motor (sinal *motor*) permanece parado até o fechamento da porta.

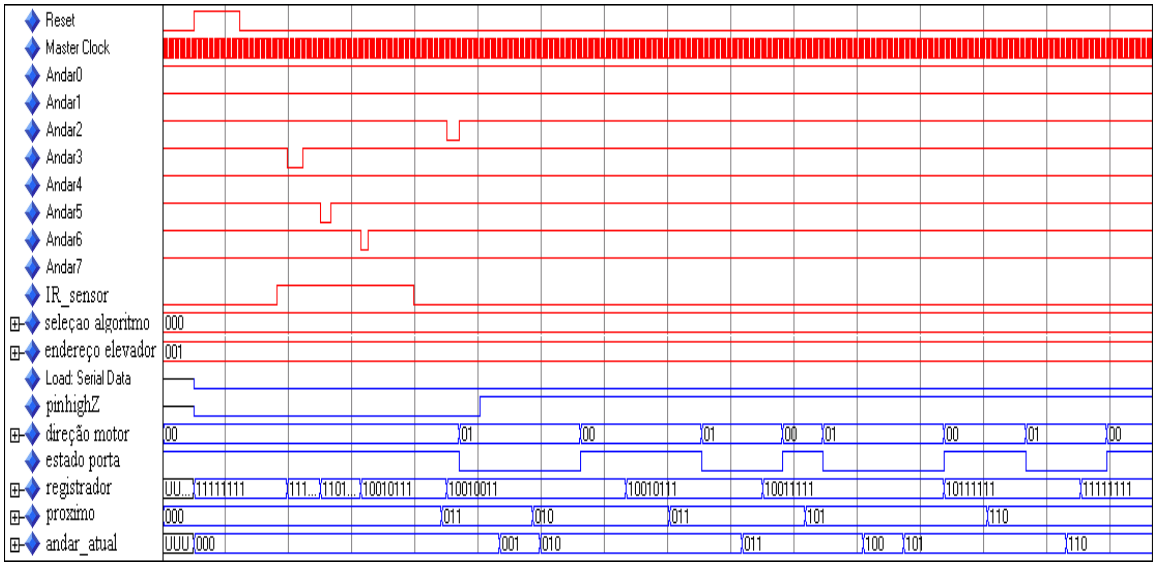


Figura 5.4: Simulação Funcional sistema integrado LCS.

Após o fechamento da porta e durante a movimentação do motor, uma nova chamada, em um andar inferior ao calculado previamente, foi realizada. Neste caso, o terceiro pavimento. O sistema reconhece esta nova chamada e muda o próximo andar a ser visitado. Na medida do atendimento de cada pavimento solicitado, o registro do andar é apagado e a porta é aberta para a saída dos passageiros.

A estrutura do sistema de elevadores proposto na seção 4.1 considera uma arquitetura distribuída na qual os controles locais rodam diferentes algoritmos de atendimento de chamadas. Por outro lado, o controle de grupo realiza as tarefas de gerenciamento, escolhendo a melhor estratégia de atendimento para cada controle local, o que possibilita que ao mesmo tempo sejam implementadas diversas técnicas de atendimento dependendo das necessidades de tráfego do prédio.

5.2 - SIMULAÇÃO DO CONTROLE DE GRUPO

Conforme explicado na seção 4.3 o controle de grupo realiza a tomada de decisões a partir dos resultados obtidos pelo analisador de tráfego e o módulo escalonador, os quais foram desenvolvidos utilizando-se sistemas baseados em lógica nebulosa (vide figura 4.15).

A implementação e posterior simulação dos sistemas nebulosos considera, além das funções de pertinência de entrada e saída, a especificação de uma base de regras que constituem o conhecimento especialista no transporte vertical de passageiros.

5.2.1 - Identificador de tráfego

A função principal desta maquina de inferência é identificar o tipo de tráfego no prédio, dependendo do estado de cada uma das características de tráfego de passageiros e da hora atual. A tabela 5.3 apresenta a base de regras utilizadas no sistema de inferência nebuloso.

Com base nas regras de conhecimento foram realizadas as simulações do sistema nebuloso visando monitorar o comportamento do *Identificador de tráfego,* figura 5.5. Neste caso, segundo os valores das variáveis de entrada, existem três regras ativas, porém a primeira regra possui um grado de crença maior. Conseqüentemente, o padrão de tráfego identificado corresponde ao tráfego majoritário de subida. Foi utilizado o método de defuzzificação *média dos máximos* (seção 2.4.6).

Tabela 5.3: Base de regras do identificador de tráfego.

Número de Regra		1	2	3	4	5	6	7	8	9
ENTRADAS	Chamadas subir	TAl	TAs	TAs	TAl ou TAm	TAm	TAl	-	TAs	TAs
	Chamadas descer	-	TAs	TAl ou TAm	TAs	TAm	TAl	TAl	TAs	TAs
	Centralizado	-	TPs	TPm	TPs	TPl ou TPm	TPm ou TPs	-	TPs	TPs
	Distribuído	-	TPl ou TPm	TPm	TPs	TPl ou TPm	TPm ou TPs	-	TPs	TPs
	Tempo	Tup	Tbt1 ou Tbt2	Tln	Tln	Tbt1 ou Tbt2	Tbt1 ou Tbt2	Tdown	Tin1 ou Tin2	< Tin1 ou > Tin2
SAIDA	**Tipo de tráfego**	Tráfego de subida	Tráfego negócios	Tráfego lanche A	Tráfego lanche B	Tráfego negócios/ pesado	Tráfego pesado	Tráfego de descida	Tráfego inativo	Tráfego inativo

Finalmente, após o monitoramento, são analisadas as curvas de controle do sistema de inferência *Identificador de tráfego*. A figura 5.6 apresenta o comportamento da variável de saída *Tráfego em* função das variáveis de entrada *Tempo*, *Chamadas subir* e *Chamadas descer*. As outras variáveis de entrada permanecem constantes como mostrado na legenda da figura 5.6.

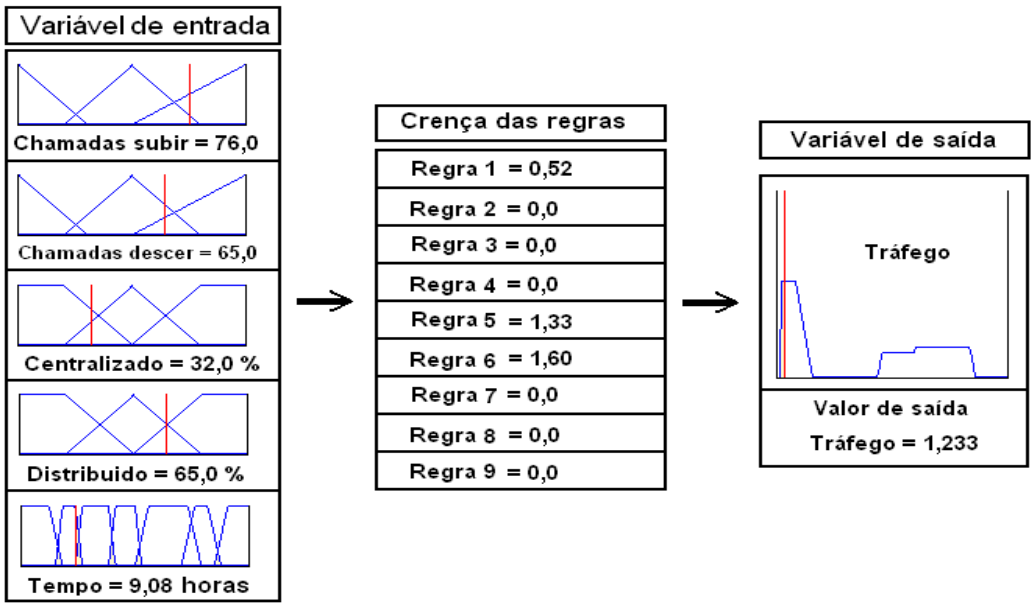


Figura 5.5: Simulação Identificador de tráfego.

Cabe mencionar que a identificação de tráfego é um processo não linear. No caso do sistema nebuloso proposto existem oito padrões definidos, por tanto, sempre que o valor da variável de saída é obtido após da defuzzificação é calculado o inteiro mais próximo.

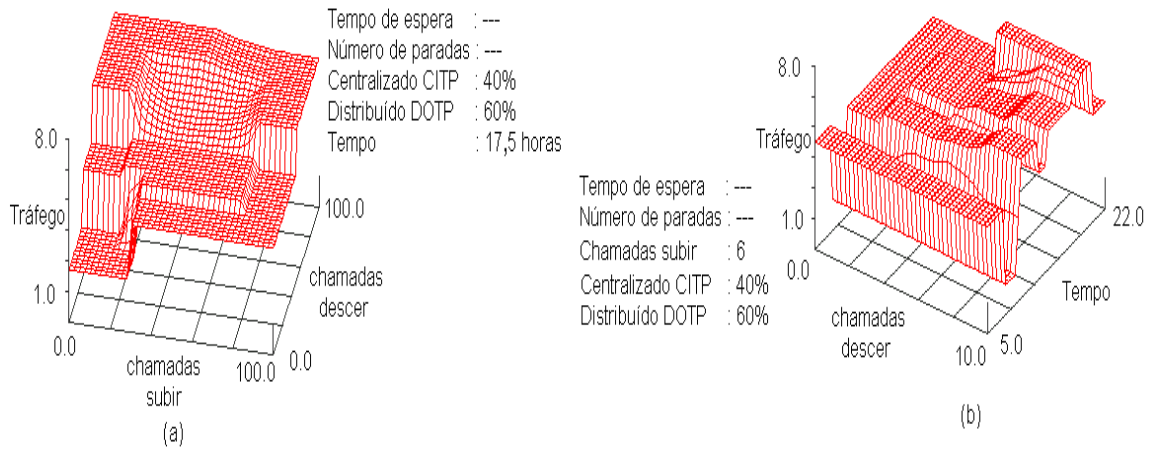


Figura 5.6: Curvas de controle Identificador de tráfego.

5.2.2 - Despacho de elevadores

A função principal deste sistema de inferência nebuloso é identificar qual é o elevador mais conveniente para receber um novo conjunto de chamadas de pavimento (escalonamento dos elevadores). Como explicado na seção 4.3.1, o módulo *despacho de elevadores* recebe como entradas as variáveis lingüísticas: *Tráfego atual, Tempo de espera* e *Número de paradas.*

O tempo de espera esta relacionado com o conforto dos usuários, entretanto, o número de paradas relaciona-se com o consumo de potência do sistema de elevadores. Sempre que uma chamada de pavimento for realizada, o módulo escalonador dará uma menção numérica (entre 1 e 10) para cada elevador do grupo. Esta menção faz referência à conveniência do elevador atender a chamada, ou chamadas, realizadas. Assim, as menções são armazenadas num vetor de possibilidades, e o elevador com maior valor de conveniência receberá os *comandos de ação* pertinentes.

A tabela 5.4 apresenta a base de regras utilizadas por este sistema nebuloso, mostrando o *valor de conveniência* (SV) em função do *número de paradas* (SN) e do *tempo de espera* (WT) em diferentes situações de tráfego. Basicamente, quando for identificado um padrão *tráfego de negócios* (BT) ou tráfego de *tempo inativo* (IT) então o escalonador dará prioridade a elevadores com maior número de paradas e tempos de espera curtos (SV: 10, 9, 8 e 7). Isto faz possível que durante os períodos de inatividade ou de tráfego leve, o sistema possa manter o maior número possível de elevadores parados, resultando num baixo consumo de potência.

Por outro lado, quando o tráfego atual não for um tráfego leve então o sistema priorizará os elevadores com poucas paradas programadas (SV: 10, 8 e 7), resultando num atendimento de usuários mais eficiente durante os horários de pico de subida, pico de descida e nos períodos de tráfego pesado. Em contrapartida poderá existir um aumento do consumo de potência.

Tabela 5.4: Base de regras do escalonador de elevadores.

Tráfego: BT ou IT			
SN \ WT	WTs	WTm	WTl
SNs	4	3	1
SNm	8	6	5
SNl	10	9	7

Tráfego: ~BT & ~IT			
SN \ WT	WTs	WTm	WTl
SNs	10	8	6
SNm	8	7	5
SNl	6	2	1

A figura 5.7 apresenta uma simulação do sistema de inferência nebuloso no caso de um tráfego majoritário de descida (*Down*: padrão de tráfego 7.0). O sistema de elevadores corresponde a um grupo de três elevadores cada um com oito paradas (oito pavimentos), então o universo de discurso da variável *número de paradas* (SV) será o intervalo [0 até 8]. O universo de discurso da variável *tempo de espera* (WT) é o intervalo [15 até 400] segundos (vide seção 4.3.1).

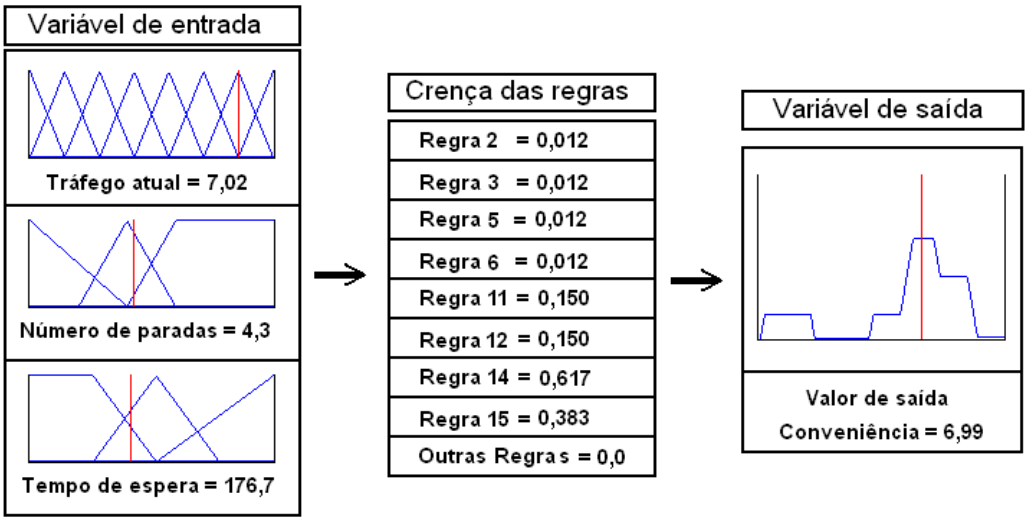


Figura 5.7: Simulação Escalonador de elevadores.

Pode-se observar na figura 5.8 que as funções de pertinência das variáveis *número de paradas* e *tempo de espera* possuem uma distribuição não simétrica no universo de discurso. Especificamente, os números de paradas grandes e tempos de espera curtos abrangem maior área no seu universo de discurso. Em conseqüência, os valores médios no valor de número de paradas também podem ser interpretados como grandes, fazendo com

que o sistema de inferência seja sensível aos elevadores com maior número de andares programados. Isto favorece o baixo consumo de potência, pois os elevadores com poucas paradas programadas ficariam inativos.

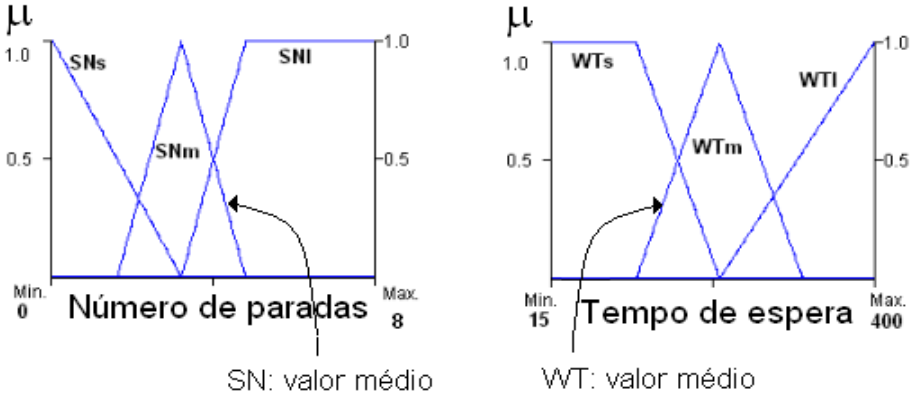


Figura 5.8: Funções de Pertinência (a) Número de paradas (b) Tempo de espera.

Por outro lado, valores médios de tempos de espera podem ser interpretados como pequenos. Isto resulta numa maior sensibilidade nos tempos de espera curtos. Desta forma, o elevador que forneça o menor tempo de espera dos usuários poderá atender uma nova chamada de pavimento. Estas implicações também podem ser conferidas nas curvas de controle do sistema nebuloso de *despacho de elevadores* (figura 5.9), nas quais se observa o comportamento da variável de saída *valor de conveniência* (SV) de um elevador em função das entradas. Neste caso as entradas do sistema *identificador de tráfego* permanecem constantes.

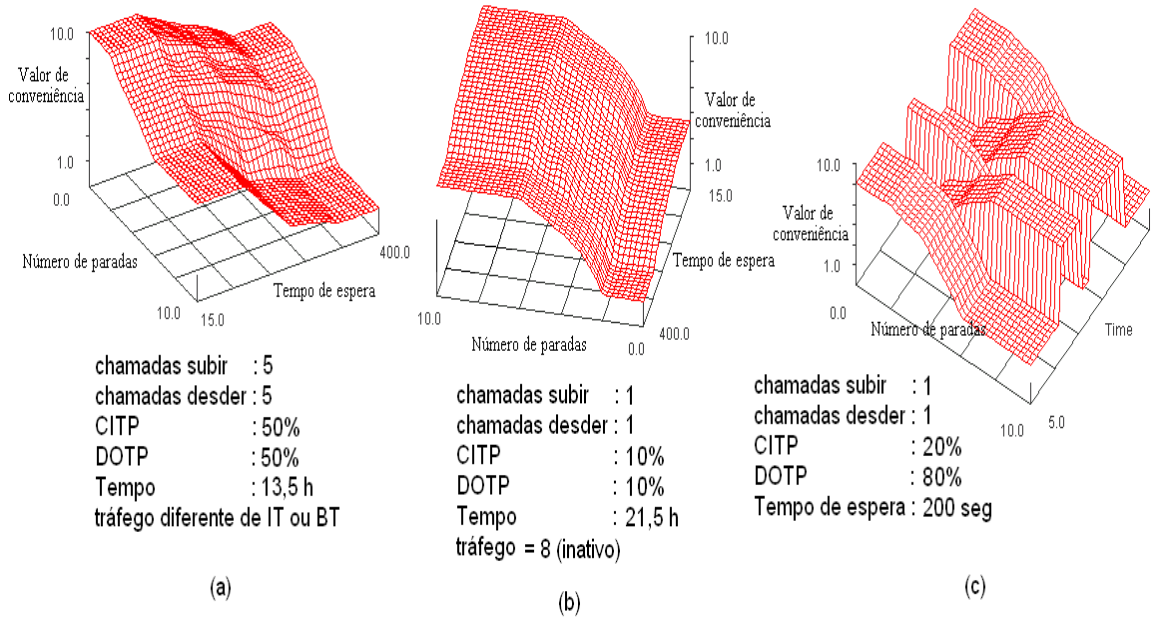


Figura 5.9: Curvas de controle do despacho de elevadores.

Observa-se na figura 5.9 que no caso de tráfego diferente do tráfego leve (BT ou IT), o valor de conveniência é priorizado para elevadores com número de paradas pequenos e tempos de espera grandes. No caso de ser identificado um tráfego leve o sistema muda de estratégia, dando mais prioridade aos elevadores com número maior de paradas programadas dano pouca importância ao valor de tempo de espera.

5.3 - INTEGRAÇÃO E SIMULAÇÃO DO SISTEMA DE ELEVADORES

Após as simulações do sistema nebuloso de controle de grupo e do controle local, foi realizada a integração das partes funcionais do sistema de elevadores, segundo a arquitetura proposta na seção 4.1. O sistema foi configurado para o caso de um prédio de 8 andares e um grupo de três elevadores. Utilizando-se conversores RS232-RS485 (WP05) foi construída a interface de conexão entre os controles locais e a interface virtual de elevadores (VEI) contendo o controle de grupo (vide figura 5.10).

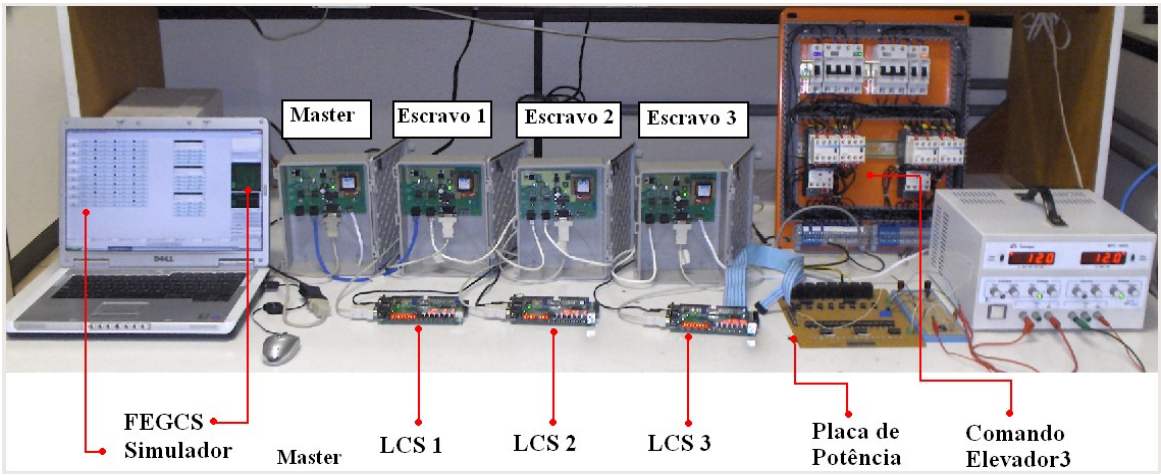


Figura 5.10: Integração das partes funcionais do sistema de elevadores.

Cada um dos sistemas de controle local (implementados nas FPGAs), possui um endereço para identificação que é utilizado pela interface virtual de elevadores (VEI). Os passos envolvidos na interação entre o usuário e as tarefas executadas pela interface virtual são descritos a seguir:

- O usuário realiza uma ou mais chamadas de pavimento ou chamadas de cabina.
- O controle de grupo recebe esta informação e analisa a possibilidade de mudar o algoritmo de escalonamento atual em um ou mais sistemas de controle local. Logo em seguida o controle de grupo seleciona o elevador ou elevadores mais convenientes para

atender as chamadas de pavimento.

- No caso em que o algoritmo de escalonamento tenha mudado, o sistema de controle local (LCS) atualiza o algoritmo correspondente e recebe as chamadas de pavimento. Entretanto, quando o estado do elevador em qualquer um dos LCS tenha mudado, o comando correspondente será transmitido ao VEI através da rede RS485. Finalmente, o dado é atualizado numa matriz de estado e a interface de usuário desenha o novo estado na tela. O usuário pode validar em tempo real o comportamento dos algoritmos de escalonamento num grupo de elevadores.

5.3.1 - Diagramas de Movimentação

Os diagramas de movimentação dos primeiros quatro algoritmos (só um botão por andar) são mostrados na figura 5.11. Visando comparar o escalonamento de cada algoritmo, foi implementado um sistema de oito andares no qual cada elevador roda um algoritmo diferente.

O primeiro elevador roda o algoritmo *Coletivo subida / Coletivo descida* (algoritmo C), o segundo elevador roda o algoritmo *Coletivo subida / seletivo descida* (algoritmo U), o terceiro controle local implementa o algoritmo *Seletivo subida / Coletivo descida* (algoritmo D), finalmente, o quarto elevador usa o algoritmo *Seletivo subida / Seletivo descida* (algoritmo S).

As mesmas chamadas de pavimento mostradas na figura 5.11(a) são recebidas pelos sistemas de controle local (LCS) e o comportamento de atendimento de cada um dos algoritmos é apresentado na figura 5.11 (b).

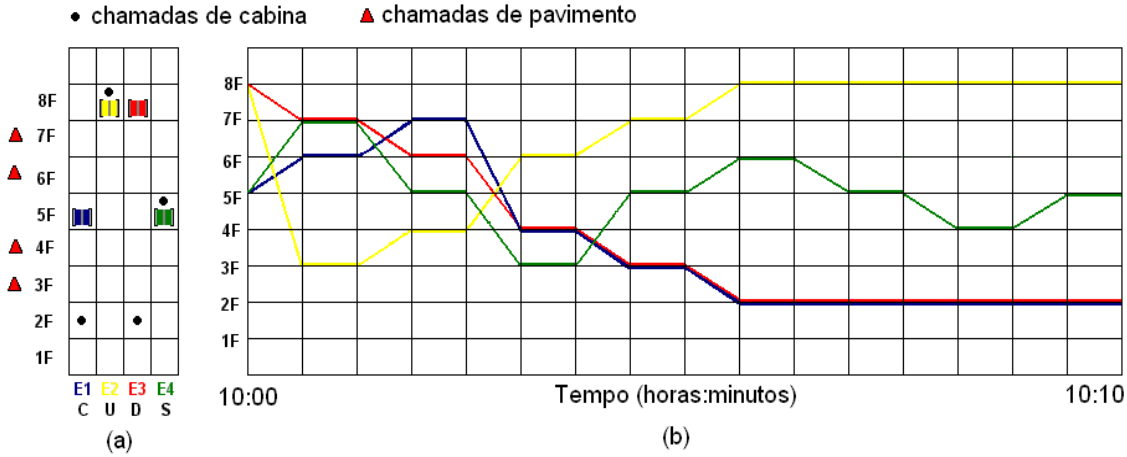


Figura 5.11: (a) Chamadas de pavimento e cabina (b) Diagramas de movimentação.

As chamadas de pavimentos são representadas pelos triângulos da figura 5.11(a). As chamadas de cabina são representadas pelos círculos pretos, que neste caso são iguais para cada elevador. Este é o caso em que todos os passageiros dentro da cabina desejam ir para o mesmo andar, permitindo observar as diferencias no atendimento dos pavimentos. Deve-se lembrar que as chamadas de carro sempre são atendidas usando o principio coletivo. As mesmas chamadas de pavimento (3F, 4F, 6F e 7F) foram transmitidas para cada controle local.

O *Algoritmo Coletivo* (Algoritmo C), implementado no primeiro elevador, atende as chamadas de pavimento da mesma forma durante a viagem de subida e de descida. Pode-se observar que este elevador atenderá as chamadas mais próximas na direção de movimento. Este comportamento é comum nos casos de tráfego leve, no quais os passageiros movimentam-se entre os andares sem muita intensidade (*inter-floors*). Em conseqüência, este tipo de algoritmo é conveniente quando são identificados os padrões de tráfego de *tempo de negócios* (BT) ou de *tempo inativo* (IT).

No caso do segundo elevador, observou-se que o *Algoritmo de Subida* (Algoritmo U) atendeu primeiramente o andar mais baixo que realizou uma chamada. Posteriormente, atendeu os pavimentos solicitados segundo o principio coletivo durante a viagem de subida. Este elevador tenta levar as pessoas dos andares mais baixos aos andares mais altos. Em edifícios comerciais este tipo de comportamento é encontrado nos mesmos horários por dia, isto é, entre as 07:30 e as 08:30 a.m, horário em que comumente os usuários começam a jornada de trabalho (tráfego majoritário de subida (*Up*)) e às 13:30 – 14:30 p.m quando os usuários voltam ao prédio depois do lanche (tráfego LTB).

O terceiro elevador implementa o *Algoritmo de Descida* (Algoritmo D). Neste caso o fluxo de passageiros é na direção de descida. Pode-se observar que, inicialmente, o elevador atendeu a solicitação mais alta e depois aplicou o principio coletivo, atendendo as chamadas mais próximas durante a viagem de descida. Deve-se lembrar que não importa a ordem em que apareceram as chamadas. Em conseqüência, o elevador transporta os usuários desde os andares mais altos até os andares mais baixos. Este comportamento é encontrado no começo do horário de lanche (tráfego LTA), entre as 11:30 a.m e as 12:30 p.m, e também entre as 17:00 e as 18:30 p.m, horário em que os passageiros dirigem-se à saída do prédio terminando a jornada de trabalho (tráfego de descida (*Down*)).

O quarto elevador rodando o *Algoritmo Seletivo* (Algoritmo S) tenta levar as pessoas aos andares intermediários do prédio. O critério se seleção é o mais alto ou o mais baixo dos andares solicitados. Assim, quando a chamada de cabina for a mesma, este algoritmo transporta passageiros no caso de tráfego centralizado, por exemplo, em horários prévios a conferencias. Em contrapartida, o tempo de viagem e de espera é alto e conseqüentemente produz desconforto aos usuários.

A figura 5.12 apresenta o diagrama de movimentação do quinto algoritmo implementado em *hardware*: *Coletivo - Seletivo / subida - descida*. Este algoritmo utiliza dois botões por pavimento indicando a direção de deslocamento dos usuários. O mesmo pode ser configurado para atender primeiramente as chamadas de subida (modo subida) ou primeiramente as chamadas de descida (modo descida).

Visando comparar as diferenças de atendimento, as mesmas chamadas de pavimento são consideradas nos dois modos de implementação. Entretanto, a chamada de cabina para os passageiros na direção de subida é o oitavo andar (8F) e a chamada de carro para os passageiros na direção de descida é o primeiro andar (1F).

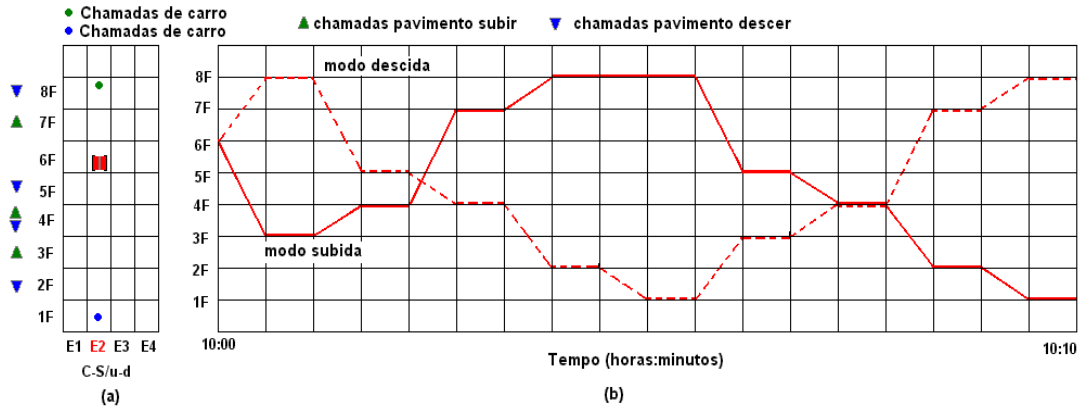


Figura 5.12: (a) Chamadas de pavimento e de cabina (b) Diagrama de movimentação do algoritmo CS-ud no modo subida e no modo descida.

No modo subida (linha continua) o elevador atenderá primeiramente os andares que fizeram uma solicitação de subida. Desta forma, o sistema aplica o principio seletivo escolhendo o andar mais baixo que realizou uma chamada de subida, isto é o terceiro andar. Posteriormente, o sistema atende de forma coletiva todas as chamadas nesta direção durante a viagem de subida. Uma vez todas as chamadas de subida foram atendidas, o elevador escolhe o andar mais alto que realizou uma chamada de descida (princípio seletivo) e atende estas chamadas, de forma coletiva, durante a viagem de descida.

A linha pontada na figura 5.12 (b) apresenta o comportamento do mesmo algoritmo configurado no modo de descida, isto é, um atendimento preferencial das chamadas de descida. Este algoritmo pode ser utilizado para atendimento de passageiros durante tráfego pesado ou combinações de tráfegos de subida / descida.

5.3.2 - Tempo de Espera dos Usuários e Potência de Consumo

A *Interface Virtual de Elevadores* (VEI) possui um ambiente de simulação que permite manipular os botões de seleção de chamadas de pavimento e de chamadas de carro. Desta forma, é possível validar o comportamento do despacho dos elevadores em diferentes situações de tráfego e dos Sistemas de Controle Local (LCS) para os diferentes algoritmos de escalonamento.

Conforme explicado nas seções 2.2.1 e 2.2.2 o objetivo de um sistema de elevadores é aumentar a capacidade de transporte de passageiros, diminuindo o tempo de espera dos usuários e o consumo de potência do sistema geral. O ambiente de simulação considera que o edifício tem apenas uma entrada principal. Isto é importante devido a que em edifícios com mais de uma entrada, o tempo de espera médio pode ultrapassar o valor do intervalo de tempo (seção 2.2.2). Nestes casos, o tempo de espera nas entradas principais pode ser maior do que o tempo de espera em horário de pico de subida (Siikonenb, 1997).

As curvas de tempo de espera nos pavimentos e o tempo de espera nos carros para diferentes fluxos de passageiros é uma forma de avaliar o desempenho do sistema de elevadores. A partir destas curvas é possível conferir a eficiência do sistema de controle de grupo e o funcionamento dos diferentes algoritmos implementados em *hardware*.

O sistema de elevadores proposto (Sistema Nebuloso de Controle de Grupo (FECGS) e Sistemas de controle Local (LCS)) foi submetido a simulações de tráfego de descida. O sistema, configurado no caso de um prédio de oito andares e três elevadores, foi carregado no ambiente de simulação. Cada simulação consiste em calcular, para diferentes fluxos de passageiros, o número de paradas dos elevadores e o tempo de espera nos pavimentos e dentro dos carros. O fluxo de passageiros varia entre 10 até 150 passageiros cada 5 minutos. Para cada simulação o tipo de algoritmo é previamente escolhido.

As figuras 5.13 até 5.18 apresentam os resultados de simulação do sistema proposto no

caso de um tráfego majoritário de descida, segundo as seguintes especificações:

- Fluxo de Passageiros: [10 até 150] passageiros cada 5 minutos;
- Velocidade = 50 m/min. Carga máxima do elevador = 10 pessoas;
- Hora de simulação: [17:30 até 18:30];
- Não é considerado o tempo de entrada e saída dos passageiros da cabina.

Visando comparar o desempenho dos algoritmos de escalonamento, as simulações foram realizadas em quatro grupos. Assim, foram obtidos os dados, mantendo o mesmo tipo de algoritmo em todos os LCSs e mudando o fluxo de passageiros em intervalos de quinze em quinze, começando em 15 e finalizando em 150 passageiros cada 5 minutos. Posteriormente, muda-se o algoritmo de escalonamento que será executado nos LCSs e é começada uma nova simulação.

Cada simulação considera a contagem de tempo gasto pelos usuários nos pavimentos, dentro das cabinas e do número de paradas que cada elevador realiza. Este processo é realizado a partir de chamadas de pavimento aleatórias. Sempre que um pavimento é atendido, é realizada uma chamada de cabina para o primeiro andar (tráfego de descida). Para intensidades de tráfego pequenas, [10, 15 e 30 passageiros cada 5 minutos] são poucas as chamadas realizadas, e portanto, são poucos os dados obtidos. Conseqüentemente, os valores de tempo para intensidades de tráfego pequenas são desconsiderados do análise.

Dado que foi realizada uma simulação no caso de tráfego majoritário de descida, é esperado um melhor desempenho do sistema quando os controles locais (LCS) executem o algoritmo de descida, *Seletivo descida / Coletivo subida* (Algoritmo D). A partir das figuras 5.13 (a), 5.14 (a), 5.15 (a) é 5.16 (a) é possível observar a incidência dos algoritmos no conforto dos usuários (tempo de espera).

Nestas gráficas detalham-se as diferencias nos valores de tempo máximo de espera dos usuários nos pavimentos (em vermelho) e dentro dos carros (em verde) em função do fluxo de passageiros. No caso do ultimo algoritmo, *Coletivo Seletivo / subida descida* , as chamadas na direção descer são atendidas de forma preferencial (*modo descida*), porém, dado que todas as simulações foram realizadas no caso de tráfego de descida, o comportamento do *Algoritmo CS-ud* é idêntico ao comportamento do *Algoritmo de Descida*. Portanto, serão consideras apenas as simulações dos primeiros quatro algoritmos.

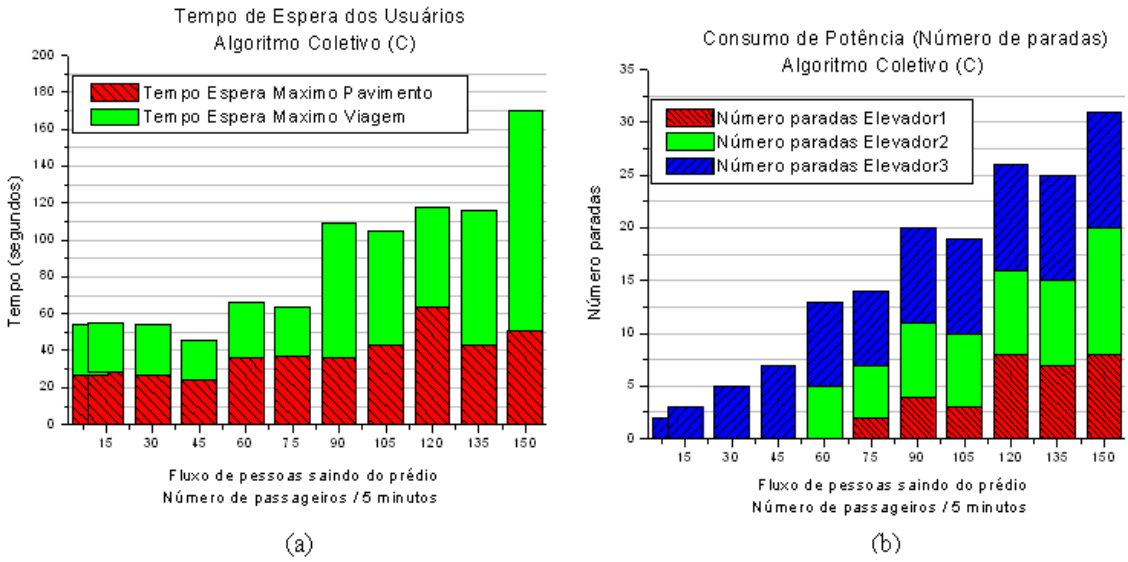


Figura 5.13: Simulação Algoritmo C (a) Tempo Máximo (b) Consumo de Potência.

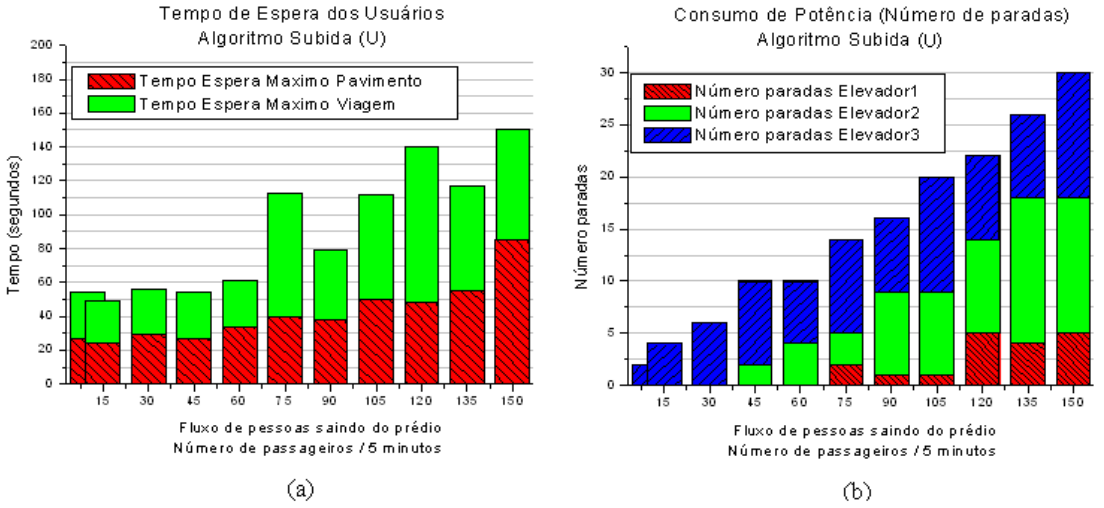


Figura 5.14: Simulação Algoritmo U (a) Tempo Máximo (b) Consumo de Potência.

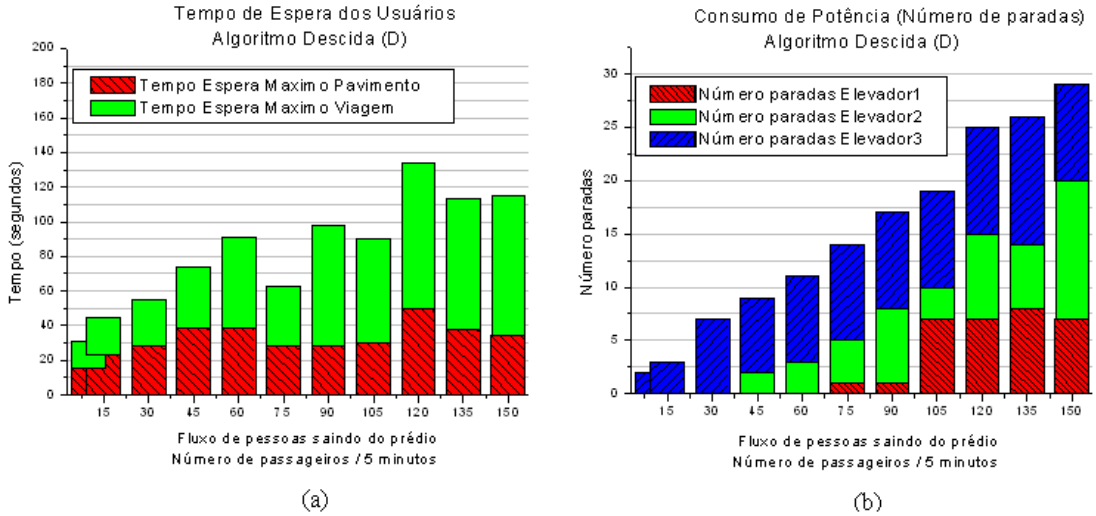


Figura 5.15: Simulação Algoritmo D (a) Tempo Máximo (b) Consumo de Potência.

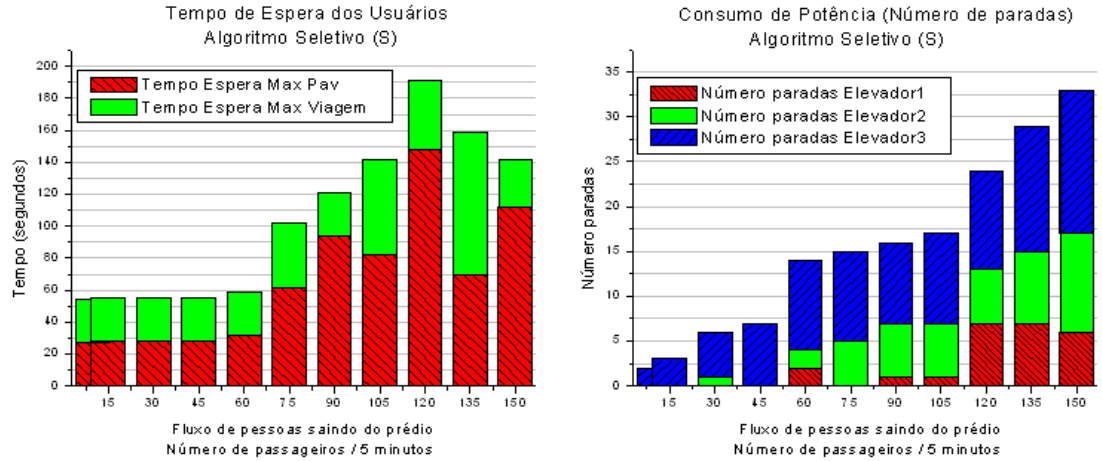


Figura 5.16: Simulação Algoritmo S (a) Tempo Máximo (b) Consumo de Potência.

O tempo de espera nos pavimentos depende da chamada de pavimento selecionada, sendo estas chamadas realizadas de forma aleatória. Nas tabelas 5.5 e 5.6 apresentam-se os dados de tempo de espera nos pavimentos e dentro da cabina, respectivamente. São relacionados apenas os tempos a partir de intensidades de 45 passageiros cada 5 minutos. Os valores para intensidades pequenas são desconsiderados do análise dos dados, devido a que nestes casos são poucas chamadas realizadas e qualquer um dos algoritmos poderá atender com o mesmo desempenho os mesmos pavimentos solicitados. Já no caso de intensidades maiores do que 45 passageiros cada 5 minutos, são realizadas um número maior de chamadas, portanto, existe diferencia no desempenho quando selecionado um algoritmo específico.

Tabela 5.5: Tempo espera nos pavimentos.

Algoritmo / HC	**45**	**60**	**75**	**90**	**105**	**120**	**135**	**150**
Algoritmo C	24	36	37	36	43	64	43	51
Algoritmo U	27	34	40	38	50	48	55	85
Algoritmo D	39	39	28	28	30	50	38	34
Algoritmo S	28	32	61	94	82	148	70	112

Tabela 5.6: Tempo espera dentro da cabina.

Algoritmo / HC	**45**	**60**	**75**	**90**	**105**	**120**	**135**	**150**
Algoritmo C	22	30	27	73	62	54	73	119
Algoritmo U	27	27	73	41	62	92	62	65
Algoritmo D	35	52	35	70	60	84	75	81
Algoritmo S	27	27	41	27	60	43	89	30

Os tempos de espera nos pavimentos são inferiores de 60 segundos nos casos do algoritmo *Coletivo subida / Coletivo Descida* (algoritmo C) e do algoritmo *Seletivo subida / Coletivo Descida* (algoritmo D). Já no caso dos algoritmos *Coletivo subida / Seletivo Descida* (algoritmo U) e do algoritmo *Seletivo subida / Seletivo Descida* (algoritmo S) são encontrados tempos superiores de 80 segundos.

Desta forma o melhor desempenho do sistema de elevadores enquanto a tempo de espera máximo nos pavimentos é garantido mediante a aplicação do *Algoritmo Coletivo* e do *Algoritmo de Descida.* Entretanto, o tempo máximo de espera dentro da cabina é consideravelmente alto (maior a 180 segundos) no caso do *Algoritmo Coletivo* (algoritmo C). Isto é devido a que o *Algoritmo Coletivo* atende as chamadas de pavimento mais próximas na direção atual de movimento. Em conseqüência, quando é realizada uma chamada nos andares mais baixos, é provável que os usuários sejam transportados para os andares superiores antes de alcançar o primeiro andar (térreo - andar desejado), aumentando assim o tempo de espera dentro da cabina.

As figuras 5.17 e 5.18 (a) apresentam os valores de tempo total, isto é, o tempo máximo de espera nos pavimentos mais o tempo máximo de espera dentro da cabina. É observado que quando o tráfego é intenso (maior do que 90 passageiros / 5 minutos) o *Algoritmo de Descida* fornece maior conforto dos usuários, no caso particular de simulação de tráfego de descida. Para intensidades muito pequenas [10, 15 e 30 passageiros cada 5 minutos] o tempo total não apresentada maiores mudanças quando selecionados diferentes algoritmos.

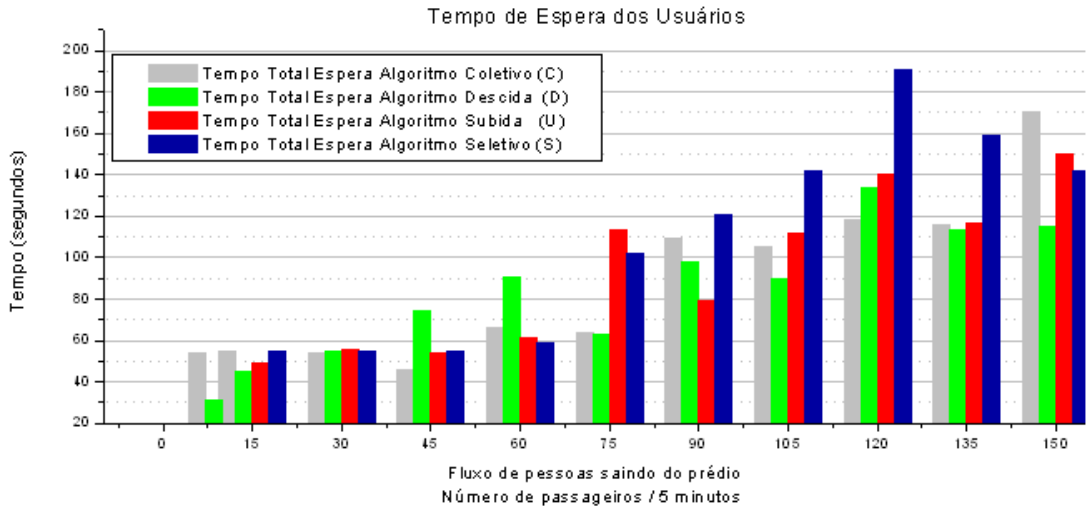


Figura 5.17: Tempo de Espera Total.

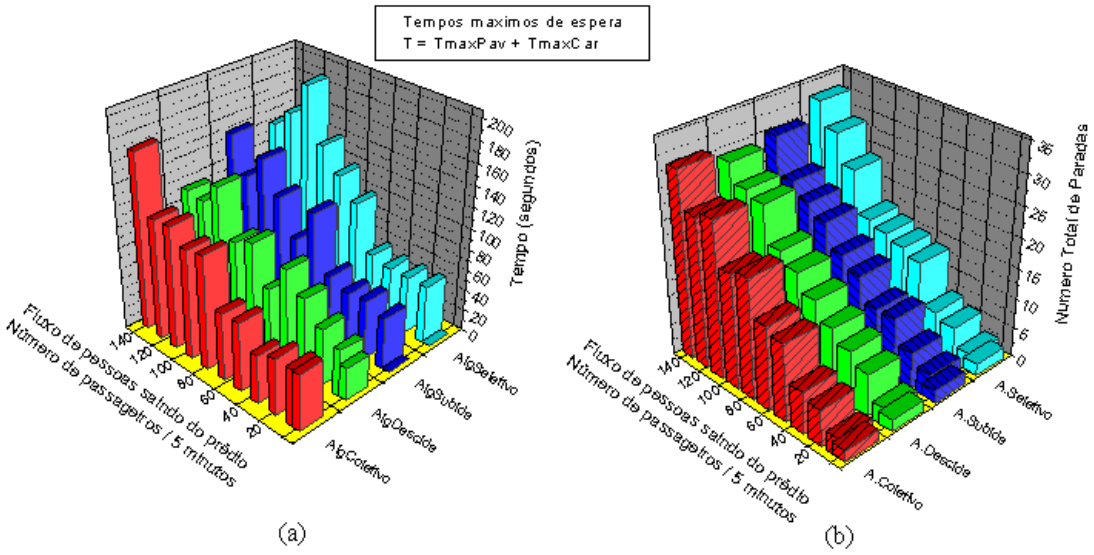


Figura 5.18: Simulação Tempo Espera - Potência (a) Tempo Total (b) Consumo Potência.

Observa-se que para as intensidades medianas de 45 e 60 passageiros cada 5 minutos o tempo total de espera no caso do *algoritmo de Descida* é maior em relação com os outros algoritmos, porém, o consumo de potência mantém valores normais. Este comportamento, é explicado a partir do caracter aleatório do número de chamadas a ser realizadas e dos andares selecionados em cada simulação, o que caracteriza as situações reais e de aplicação em prédios comerciais.

Visando caraterizar o comportamento dos algoritmos, nesta situação de tráfego, foi calculado o tempo de espera médio (AWT) para intensidades de tráfego altas (a partir de 90 passageiros cada 5 minutos). O tempo de espera médio é um valor médio do tempo de espera nos pavimentos, sendo uma medida do desempenho no atendimento dos usuários. A tabela 5.7 apresenta os valores de AWT dos usuários quando o tráfego é intenso. O *Algoritmo de Descida* possui o menor valor médio, justificando assim, a sua aplicação no caso de tráfego majoritário de descida.

Tabela 5.7: Valores médio de tempo total de espera dos usuários

Algoritmos	**Tempo médio espera total**	**AWT: Tempo médio espera nos pavimentos**
Algoritmo C	123,6 segundos	47,4 segundos
Algoritmo U	119,6 segundos	55,2 segundos
Algoritmo D	110,0 segundos	36,0 segundos
Algoritmo S	151,0 segundos	101,2 segundos

O Sistema Nebuloso de Controle de Grupo (FEGCS) manteve um comportamento adequado em relação ao despacho dos elevadores durante todas as simulações. Assim, sempre que é iniciada uma simulação, o FEGCS registra a maioria das chamadas ao terceiro elevador. Desta forma, quando o tráfego é leve (pouco intenso) o sistema de elevadores mantém desligados os elevadores número 1 e 2, sendo o elevador 3 quem atende a maioria das chamadas. Quando o elevador 3 tiver registrado uma quantidade de chamadas de forma que possa comprometer o tempo de espera dos usuários, entra em funcionamento o elevador 2. Entretanto, o elevador 1 permanece desligado. Esta situação apresenta-se em todas as simulações para um fluxo entre 45 e 60 passageiros/5 minutos.

Esperava-se que na medida em que o fluxo de passageiros aumenta, os tempos de espera fossem maiores. Em alguns casos é observado que o tempo de espera total diminui em relação ao fluxo de passageiros imediatamente anterior. Por exemplo, no caso do *Algoritmo Coletivo* e do *Algoritmo de Descida* para os fluxos [75 105 135], os tempos de espera diminuem em relação aos valores obtidos para os fluxos [60 90 120] respectivamente (figuras 5.13 (a) e 5.15 (a)). Este comportamento pode ser explicado com o aumento do consumo de potência nos mesmos pontos. De forma geral, foi encontrado que o tempo de espera dos usuários diminuiu quando foram registradas chamadas aos elevadores que permaneciam em espera.

O decremento do tempo de espera dos usuários tem como custo o aumento de consumo de potência. Nas simulações esperava-se um incremento no consumo de potência na medida do aumento do fluxo de passageiros**.**

As figuras 5.13 (b), 5.14 (b), 5.15 (b) é 5.16 (b) apresentam os resultados de número de paradas de cada elevador em função do fluxo de passageiros para cada um dos algoritmos de escalonamento. O número de paradas de cada elevador esta relacionado com o número de vezes que o motor deve iniciar e frear a marcha. Isto é proporcional ao consumo de potência (Kim *et al.*, 1998). A partir destas gráficas é possível conferir o aumento de número de paradas nos pontos em que o tempo de espera diminuiu (figuras 5.13 e 5.15 nos pontos de intensidade: [75 105 135]).

Em todas as simulações são consideradas apenas as paradas para atendimento das chamadas de pavimento. As chamadas de cabina são sempre as mesmas (primeiro andar –

térreo) e em forma geral, como observado nos mapas de escalonamento (figura 5.11), o *Algoritmo Seletivo* realiza um maior número de paradas em relação aos outros algoritmos. Deve-se lembrar que não sô é importante as chamadas que são registradas a cada elevador, senão também, a ordem em que ditas chamadas são atendidas.

Na simulação do *Algoritmo de Subida* (Algoritmo U), o tempo de espera para um fluxo de 90 passageiros/5 minutos diminuiu consideravelmente em relação ao tempo para um fluxo de 75 passageiros/5 minutos (figura 5.14 (a)). Na figura 5.14 (b) é possível observar que a quantidade de chamadas registradas ao elevador 1 diminuiu levemente. Entretanto, o número de chamadas registradas ao elevador 2 aumentou abruptamente. Em conseqüência, a potência de consumo aumentou, os três elevadores entraram em movimento, sendo o elevador 2 quem atendeu a maiorias das chamadas. Esta mesma situação pode ser observada para uma intensidade de tráfego de 135 passageiros/5 minutos.

De forma geral, é possível observar que, no caso de fluxos intensos (maior do que 90 passageiros/5minutos), o número de chamadas registradas ao elevador 1 no *Algoritmo de Descida* (Algoritmo D) é maior do que as chamadas registradas ao mesmo elevador nos outros algoritmos. Isto explica o fato de que os tempos de espera para este algoritmo sejam menores. Ao mesmo tempo o *Algoritmo de Descida* possui o menor número médio de paradas, conseqüentemente o consumo de potência é menor, justificando assim, a sua aplicação no caso de tráfego majoritário de descida.

5.4 - VANTAGENS DA IMPLEMENTAÇÃO EM *HARDWARE* DO LCS

No intuito de comparar o desempenho da arquitetura proposta para o Sistema de Controle Local (LCS) em relação às implementações convencionais, varias medidas do tempo gasto pelo sistema para realizar o cálculo do próximo andar a ser visitado foram realizadas no osciloscópio. Foi realizada a implementação de um prédio de oito andares usando duas arquiteturas diferentes. A primeira, baseada em dispositivos microprocessados, implementa o algoritmo coletivo na subida e na descida. A segunda, baseada em dispositivos FPGA, implementa a descrição do LCS mostrada na figura 4.10 e permite a escolha de cinco algoritmos de escalonamento.

Na implementação do algoritmo coletivo foi utilizado um dispositivo microcontrolador

Philips 80552 e *clock* de 12MHz. Encontrou-se que o tempo gasto pelo microcontrolador depende diretamente do andar, ou andares selecionados. O menor tempo de cálculo foi de 50 milesegundos e corresponde à solicitação de um único andar, sendo que a solicitação vem de um andar vizinho ao andar atual (Jalorretto, 2005).

Na implementação do LCS em *hardware* encontrou-se que o tempo de cálculo do próximo andar não depende do número de andares nem do tipo de algoritmo selecionado. O dispositivo utilizado foi o dispositivo FPGA XC3S200 Spartan3 da *Xilinx*. O valor do *clock* é de 50MHz e o tempo de cálculo do próximo andar é de 0.1 microssegundos.

A figura 5.19 apresenta os tempos de cálculos obtidos para cada um dos casos. A partir deste dados é possível concluir que, no pior dos casos, o tempo gasto para uma implementação em *hardware* rodando a 12MHz seria de 0,0042 milisegundos. Entretanto, o tempo gasto na implementação em microcontroladores rodando a 12 MHz foi de 50 milisegundos. Em conseqüência, o fator de desempenho da descrição do sistema de controle local em *hardware* é 11.900 vezes maior do que o desempenho da mesma descrição usando arquiteturas convencionais microcontroladas.

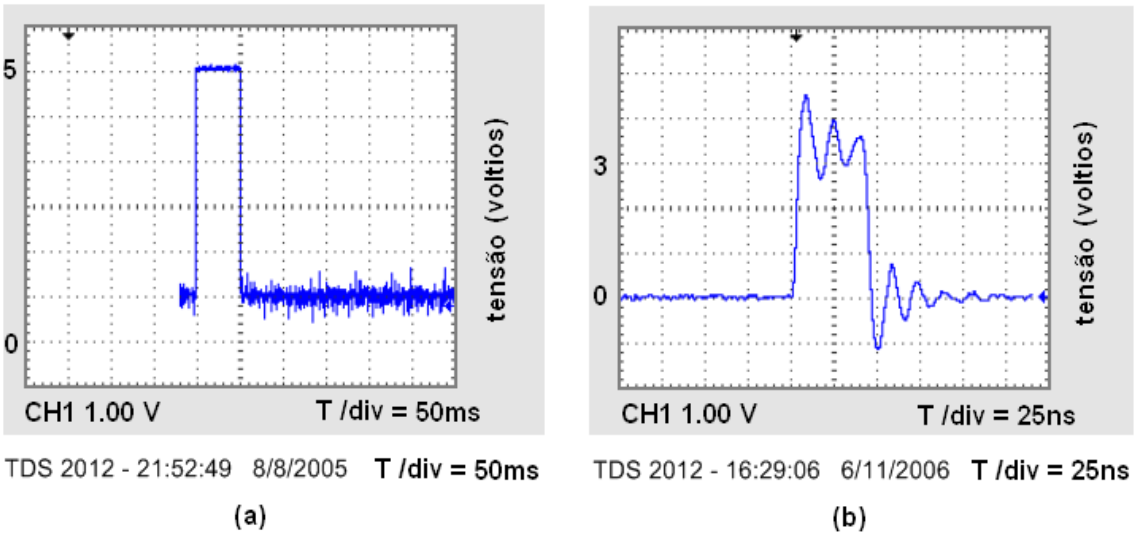


Figura 5.19: Tempo de cálculo: (a) Dispositivo microcontrolado (b) Dispositivo FPGA.

Conforme explicado nas análises do custo/desempenho da implementação do sistema de controle local baseado em complexidade (vide seção 4.2.8), é possível justificar o valor do tempo de cálculo no próximo andar por um custo de área devido à utilização de um laço de repetição que implementa uma busca seqüencial no registrador de andares.

5.5 – CONCLUSÃO DO CAPÍTULO

Neste capítulo foram apresentados os resultados de sínteses e simulação dos Sistemas de Controle Local (LCS), assim como também, as simulações do sistema de inferência nebuloso para controle de grupo (FEGCS), e do sistema integrado no caso específico de um prédio de oito andares e três elevadores.

A implementação do Sistema de Controle Local (LCS) em FPGAs permitiu reduzir o consumo de área, dado que os algoritmos de escalonamento compartem os mesmos recursos de *hardware.* A descrição do Sistema de Controle Local (LCS) de elevadores consume 46% dos recursos disponíveis da FPGA. A taxa de consumo em LUTs dos algoritmos de escalonamento é maior em relação aos outros componentes devido à implementação de um laço *FOR* (aninhado com uma condição *IF* como estrutura de calculo do próximo andar a ser visitado).

O sistema de inferência nebuloso para controle de grupo foi projetado de forma que o despacho de elevadores seja mais sensível aos elevadores com maior número de paradas programadas e aos tempos de espera curtos. Isto deve diminuir o consumo de potência e aumentar o conforto dos usuários. Estas considerações foram conferidas na simulação do sistema integrado. Nas figuras de tempo máximo de espera dos usuários e de número de paradas (figuras 5.13 até 5.18) pode-se observar que o Sistema Nebuloso de Controle de Grupo (FEGCS) favorece os baixos consumos de potência, ainda quando o tráfego é intenso, comprometendo assim, o conforto dos usuários.

As simulações para um tráfego majoritário de descida (*Down*) confirmaram os resultados esperados em relação ao maior desempenho do *Algoritmo de Descida* (Algoritmo D). Conseqüentemente, espera-se que no caso de tráfego majoritário de subida (*Up*) e de tráfego de combinação negócios/pesado (BTH) o maior desempenho seja do *Algoritmo de Subida* (Algoritmo S) e do Algoritmo *Coletivo-Seletivo/Subida-Descida* (CS-ud), respectivamente.

6 – CONCLUSÕES E SUGESTÕES PARA TRABALHOS FUTUROS

6.1 - CONSIDERAÇÕES GERAIS

Este trabalho propõe uma arquitetura para a implementação de um sistema de elevadores que atende as necessidades do transporte vertical de passageiros, considerando a descrição de algoritmos de escalonamento usando dispositivos reconfiguráveis.

O sistema de elevadores proposto explora o uso de modelos heurísticos baseados em lógica nebulosa como uma alternativa para o controle de um grupo, objetivando identificar padrões de tráfego e adotar estratégias de despacho de elevadores.

Este trabalho verificou a utilidade e flexibilidade da implementação em *hardware* dos algoritmos de escalonamento de elevadores, assim como também, foram confirmadas as vantagens em relação ao desempenho, consumo de área e consumo de potência dos sistemas de controle local quando são descritos através de arquiteturas reconfiguráveis.

6.2 - SISTEMA DE CONTROLE LOCAL DE ELEVADORES

A implementação do Sistema de Controle Local (LCS) em FPGAs permitiu reduzir o consumo de área, dado que os algoritmos de despacho compartem os mesmos recursos de *hardware.* A descrição final do Sistema de Controle Local de elevadores consume 46% dos recursos disponíveis na FPGA XC3S200 da placa Spartan3 da Xilinx.

Cinco algoritmos de escalonamento de elevadores foram implementados diretamente em *hardware*, num ambiente integrado que, além de compartilhar alguns recursos, reduzindo o consumo de LTUs, permite uma redução do consumo de potência. Isto acontece dado que quando um algoritmo está ativo, os outros algoritmos são desabilitados parando o sinal de *clock* de entrada de cada um deles.

A vantagem da arquitetura proposta para o sistema de controle local, com relação aos sistemas convencionais, é que a implementação em *hardware* dos algoritmos de escalonamento possibilitou a redução do tempo gasto para o cálculo do próximo andar a

ser visitado por cada elevador, o que pode resultar útil em ambientes de tráfego intenso e elevadores de alta velocidade.

Na implementação do LCS em *hardware* encontrou-se que o tempo de cálculo do próximo andar não depende do número de andares nem do tipo de algoritmo selecionado. É possível justificar o valor de tempo de cálculo do próximo andar por um aumento na taxa de consumo de LTUs, devido ao uso de um laço *FOR* que implementa a busca dos sinais solicitados no registrador de andares. No procedimento de síntese, esta descrição é repetida *n* vezes, sendo *n* o número de andares, de forma que o tempo de busca no registrador é o menor possível, em contrapartida existe um alto o aumento de uso de área na FPGA.

6.3 - SISTEMA NEBULOSO DE CONTROLE DE GRUPO

O Sistema de Controle de Grupo de Elevadores (EGCS), baseado em lógica nebulosa (FEGCS) e desenvolvido em linguagem *Java*, permitiu validar em tempo real o comportamento dos algoritmos de escalonamento dentro do grupo de elevadores. As seguintes considerações são concluídas:

- O sistema de inferência nebuloso para controle de grupo foi projetado de forma que o escalonamento de elevadores seja mais sensível aos elevadores com maior número de paradas programadas e aos tempos de espera curtos. As simulações realizadas mostraram, em certos casos, uma diminuição do consumo de potência e um aumento do conforto dos usuários. No entanto, a partir das gráficas de tempo máximo de espera dos usuários e de número de paradas é possível concluir que o Sistema nebuloso de Controle de Grupo (FEGCS) favorece os baixos consumos de potência, ainda quando o tráfego é intenso. Isto compromete o conforto dos usuários em situações em que o sistema deveria priorizar a diminuição dos tempos de espera.

- As simulações para um tráfego majoritário de descida (*Down*) confirmaram os resultados esperados em relação ao maior desempenho do *Algoritmo de Descida* (Algoritmo D). De forma qualitativa, observou-se que quando é identificado um trafego majoritário de subida (*Up*) e no caso de um prédio de 3 elevadores e 8 andares, foi encontrado um maior desempenho quando dois LCSs executam o *Algoritmo de Subida* (algoritmo S) e um LCS executa o *Algoritmo Coletivo* (Algoritmo C). Espera-

se que no caso de tráfego majoritário de combinação negócios/pesado (BTH) o maior desempenho seja alcançado pelo *Algoritmo Coletivo-Seletivo/Subida-Descida* (Algoritmo CS-ud).

6.4 - INTEGRAÇÃO DO SISTEMA DE ELEVADORES

O sistema de elevadores proposto realiza um processamento distribuído implementando um grupo de sistemas de controle locais autônomos e independentes entre si. Desta forma, cada um dos LCSs pode gerenciar o funcionamento um único elevador através da implementação de diferentes estratégias (algoritmos) de escalonamento de elevadores. Esta metodologia permite que o sistema de elevadores continue em funcionamento caso o sistema central quebrar ou ficar fora de conexão.

Os testes realizados confirmaram que o desempenho de cada algoritmo de escalonamento é diferente segundo a situação de tráfego atual no edifício. Através da arquitetura proposta foi possível validar o desempenho de cada algoritmo, e o desempenho de configurações particulares num grupo de elevadores, em situações diversas e para distintas intensidades de tráfego.

Na arquitetura proposta cada LCS não só executa as tarefas referentes à locomoção do elevador, como também é capaz de configurar a estratégia de atendimento, mudando assim a ordem dos andares a serem visitados. Isto possibilita que o EGCS escolha o algoritmo mais apropriado para rodar em cada sistema de controle local, o que resulta na flexibilidade da implementação.

A vantagem da arquitetura proposta para o sistema de elevadores, em relação com sistemas convencionais, considera que o sistema de controle de grupo não precisa realizar os cálculos do próximo andar a ser atendido. Isto resulta na diminuição dos cálculos realizados pelo EGCS.

Em termos de flexibilidade, através da implementação em FPGAs dos Sistemas de Controle Local fica aberta a possibilidade de acrescentar futuras interfaces ou componentes funcionais no sistema, sem precisar projetar, fabricar e acrescentar uma novas placas para este fim. Ainda mais importante, é que o desenho destes novos componentes dentro da

mesma FPGA não compromete o desempenho geral do dispositivo, devido à possibilidade de executar processos de forma concorrente.

6.5 - SUGESTÕES PARA TRABALHOS FUTUROS

As perspectivas de trabalhos futuros relacionadas a seguir, visam alcançar um nível funcional do sistema de elevadores proposto. Melhorar as funcionalidades existentes e acrescentar novas interfaces permitirá que o desenvolvimento alcançado seja comparado com sistemas comerciais de alto nível.

6.5.1 - Construção de um protocolo de comunicação

Uma das dificuldades encontradas no sistema foi a perda de dados enviados desde o Sistema de Controle de Grupo (EGCS) aos Sistemas de Controle Local (LCS) quando o tráfego no prédio é intenso. O protocolo de comunicação atual considera apenas a função do LCS como escravo no contexto do padrão RS-485. Deve-se ampliar a funcionalidade deste protocolo de forma que quando um comando de ação seja enviado (do mestre ao escravo), dita informação possa ser interpretada no LCS, mesmo que nesse momento esteja-se enviado um comando de estado (do escravo ao mestre).

6.5.2 - Criação de um seqüenciador na arquitetura dos LCSs

Na figura 4.10 apresentou-se a arquitetura do sistema de controle local (LCS), na qual, cada maquina de estados finitos que descreve os algoritmos de escalonamento executa um laço *FOR* aninhado com uma condição *IF* para a busca dos andares solicitados. Esta descrição é replicada *n* vezes, sendo *n* o número de andares, ainda para cada algoritmo. Isto resulta no alto consumo de área na FPGA. Planeja-se a criação de um seqüenciador que permita eliminar o uso do laço *FOR*, e que ao mesmo tempo seja compartido como recurso de *hardware* pelos algoritmos de escalonamento. Com isto, espera-se diminuir o consumo de LTUs, em contrapartida, o desempenho poderia estar comprometido.

6.5.3 - Implementação do sistema moderno de elevadores

Planeja-se a implementação de modelos de elevadores modernos. Nestes modelos as chamadas de cabina são realizadas antes que o usuário entre no carro. Desta forma, é possível realizar uma estimação do número de pessoas esperando nos pavimentos. Por outro lado, estratégias de despacho de elevadores podem ser executadas com maior

objetividade, devido a que as chamadas do andar desejado são conhecidos antes que a cabina pare nos pavimento que solicitaram o elevador.

6.5.4 - Implementação de posicionamento de elevadores por ultra-som (USP)

Com o intuito de simplificar a implementação do sistema de elevadores, pretende-se incluir um sistema de posicionamento de elevadores por ultra-som, eliminando a fiação dentro do poço do elevador. A flexibilidade dos dispositivos FPGA permite que dentro do mesmo circuito integrado, e sem perdas de desempenho, seja incluído um componente que interprete os sinais do USP, conhecendo a posição exata do elevador.

6.5.5 - Criação de uma rede *CANOpen Lift*

Visando acrescentar robustez no sistema de elevadores, planeja-se a construção de uma interface de conexão de rede de controle de área (*CAN Open Lift – Control Area Network*). Este tipo de interface de conexão é dedicado ao sistema de elevadores e são encontradas em edifícios modernos. O principal atrativo da rede *CANOpen Lift* é a capacidade de interligação de interfaces: quadros de cabina e pavimento, *drivers* de cabina e porta, sensores de peso, *encoders*, *displays* e controladoras, além de que o tamanho da rede é flexível possibilitando o controle de 8 elevadores em paralelo num prédio de 254 andares.

6.5.6 - Incorporação em *hardware* do Controlador de Grupo (EGCS)

Finalmente, estuda-se a possibilidade de implementar o controle de grupo diretamente em *hardware* através da implementação de microcontroladores embarcados e da descrição de modelos de inferência nebulosa para despacho de elevadores e identificação de padrões de tráfego.

REFERENCIAS BIBLIOGRÁFICAS

ABCI (1993). Associação Brasileira da Construção Industrializada. "*As Edificações do Terceiro Milênio.*" In: Construção No. 2355, pp. 25-26.

ABNT (1987). Associação Brasileira de Normas Técnicas. "*Normas Técnicas para Elevadores em Brasil.*" In: NBR10098. Disponível em: http://www.abnnet.com.br/

Actel Corporation. (acessado em 06/2006).

Disponível em: http://www.actel.com/

Akl, S.G. (1997). "*Parallel Computation: Models and Methods*." In: Prentice-Hall, New York, USA.

Altera Corporation. (acessado em 02/2005)
Disponivel em: http://www.altera.com/

Bastidas, G. (1999). "*Aplicação de Redes de Petri Interpretadas na Modelagem de Sistemas de Elevadores em Edifícios Inteligentes.*" In: Dissertação (Mestrado), Escola Politécnica, Universidade de São Paulo, São Paulo, Brasil.

Becker, J., Hartenstein, R. (2003). "*Configware and Morphware Going Mainstream*." In: Journal of Systems and Architectures, Vol. 49, pp.127-142.

Bonato, V. (1999). "*Projeto de um Módulo de Aquisição e Pré-processamento de Imagem Colorida Baseado em Computação Reconfigurável e Aplicado a Robôs Móveis.*" In: Dissertação (Mestrado), Universidade de São Paulo, São Paulo, Brasil.

Bronowski, J. (1992). "*A Escalada do Homem*.". In: Martins Fontes, São Paulo, Brasil.

Brown, S., Vranesic, Z. (2000). "*Fundamentals of Digital Logic with VHDL Design.*" In: Mc Graw Hill, Toronto, Canadá.

Bushby, S. (1997). "*BACnet: a Standard Communication Infrastructures for Intelligent Buildings.*" In: Automation in Construction, Vol. 6, pp. 529-540.

Cabrera, A.J., Sanchez-Solano, S., Jiménez, C.J., Barriga, A.I., Baturone. (2003). "*Arquitectura Eficiente para la Implementación en Hardware de Sistemas de Inferencia Difusos.*" In: Ingeniería Electrónica, Automática e Comunicaciones, Vol. 23(1), pp. 59-61.

CAN in Automation – *CANopen Lift.* (acessado em 02/2006).

Disponível em: http://www.can-cia.org/canopen/lift/en/profile.html

Cervera & Pioz Architects. (acessado em 06/2006).

Disponível em http://cerveraandpioz.com/

Chenais, P., Weinberger, K. (1992). "*New Approach in the Development of Elevator Group Control Algorithms."* Proceedings of the ELEVCON92, 4, pp. 48-57.

Closs, C. (1970). "*The Computer Control of Passenger Traffic in Large Lift Systems*." In: The Victoria University of Manchester, Ph.D. Thesis.

Configware - *Reconfigurable Computing.* (acessado em 06/2006).

Disponível em: http://www.configware.org/

Cypress Semiconductor Corporation. (acessado em 06/2006).

Disponível em: http://www.cypress.com/

Digilent Inc. (acessado em 06/2006).

Disponível em: http://www.digilentinc.com/

Finley, M., Karakura, R., Nbogni, R. (1991). "*Survey of Intelligence Buildings Concepts*." In: IEEE Communications Magazine, Vol. 29(4), pp. 18-23.

Fujino, A., Tobita, T., Segawa, K., Yoneda, K., Togawa, A. (1997). "*An Elevator Group Control System with Floor-Attribute Control Method and System Optimization Using Genetic Algorithms."* In: IEEE Transactions on Industrial Electronics, Vol. 44(4), pp. 546-552.

Hamblen, J.O., Furman, M.D. (2001). "*Rapid Prototyping of Digital System."* In: Ed. Kluwer Academic Publishers, Boston, USA.

Han, C. (1997). "*An OOP Bases Intelligent Building Information System in New and Technologies in Planning and Construction of Intelligent Building II."* In: Proceedings of IB/IC Intelligence Building Congress, pp. Tel-Aviv, Israel.

Haraguchi, H. (1990). "*Load Distribution Detecting Systems for Elevators."* In: US Patent No. 4951786.

Harteinsten, R. (2000). "*Configware / Software Co-Design: Be Prepared for the Next Revolution!"* Keynote, France.

Harteinsten, R. (2002). "*Enabling Technologies for Reconfigurable Computing and Software / Configware Co-Design."* In: CNRS Internal Workshop, ENST, Paris, France.

Harteinsten, R. (2004). "*The Digital Divide of Computing."* In: Proceedings of the 1st Conference on Computing Frontiers CF04, pp. 357-362, Ischia, Italy.

Hauptmeirer, D., Krumke, S.O., Rambau J., Wirth, H.C. (2001). "*Euler is Standing in Line Dial-a-Ride Problems With Precedence-constraints."* In: Discrete Applied Mathematics, 113, pp. 87-107.

Ho, Yuan-Wei., Fu, Li-Chen. (2000). "*Dynamic Scheduling Approach to Group Control of Elevator Systems with Learning Ability."* In: IEEE Proceedings of International Conference on Robotics & Automation, pp. 2410-2415, San Francisco, USA.

Hutton, M., Yuan, R., Schleicher, J., Chua, K., Phoon, H. (2006). "*A Methodology for FPGA to Structured-ASIC Synthesis and Verification."* In: Proceedings of the Conference of Design, Automation and Test in Europe, pp. 64-69, Munich, Germany.

Infolev Elevadores. (acessado em 08/2006).

disponível em: http://www.infolev.com.br/

Jalorretto, M. (2005). Projeto de Iniciação Científica, Pivic, Brasília, Brasil.

Kim, Jung-Hwan., Monn, Byung-Ro. (2001). "*Adaptive Elevator Group Control With Cameras."* In: IEEE Transactions on Industrial Electronics, 12(2), pp. 377-382.

Kim, C.B., Seong, K.A., Lee-Kwang, H., Kim, J.O. (1995). "*A Fuzzy Approach for Elevator Group Control System."* In: IEEE Transactions on Systems Man and Cybernetics Part A 25(6), pp. 985-990.

Kim, C.B., Seong, K.A., Lee-Kwang, H., Kim, J.O. (1998). "*Design and Implementation of a Fuzzy Elevator Group Control System."* In: IEEE Transactions on Systems Man and Cybernetics Part A 28(3), pp. 277-287.

Kone Corporation. (acessado em 04/2006).

Disponível em: http://www.kone.com/

Latice Semiconductor Corporation. (acessado em 06/2006).

Disponível em: http:/www.latticesemi.com/

Luo, F., Xu, Y., Cao, J. (2005). "*Elevator Traffic Flow Prediction With Least Squares Support Vector Machines."* In: IEEE Proceedings of the Fourth International

Conference on Machine Learning and Cibernetics, pp. 4266-4270, Guangzhou, China.

Makimoto, T. (2002). "*The Hot Decade of Field Programmable Technologies."* In: IEEE Proceedings of the International Congress on Field Programmable Technologies FPT, pp. 3-6, Hong Kong.

Mamdani, E. H., Assilian, S. (1975). "*An Experiment in Linguistics Synthesis With a Fuzzy Logic Controller."* In: International Journal Man-Machine Studies, Vol. 7(1), pp. 1-13.

Mamdani, E. H. (1977). "*Application of Fuzzy Logic to Approximate Reasoning Using Linguistic Synthesis."* In: IEEE Transaction on Computers, Vol. C(26), pp. 1182-1191.

Mikkola, J., Grassman, O. (2003). "*Managing Modularity of Product Architectures: Toward an Integrated Theory."* In: IEEE Transaction on Engineering Management, Vol. 50(2), pp. 204-218.

Mingjie, Lin., Gamal, A., Chang, Y., Wong, S. (2006). "*Performance Benefits of Monolithically Stacked 3D-FPGA."* In: Field Programmable Gate Array FPGA06, pp. 113-122, USA.

ModelSim. (acessado em 06/2005).

Disponível em: http://www.model.com/

Moore, G.E. (1997). "*The Microprocessor: Engine of the Technology Revolution."* In: Communications of the Association for Computer Machinery AMC, Vol. 40 (2), pp. 112-114, USA.

Muñoz, D.M., Llanos, C.H., Ayala-Rincon, M., Van-Els, R. (2006). "*Implementation of Dispatching Algorithms for Elevator Systems Using Reconfigurable Architectures."* In: Simposio Brasileiro de Concepção de Circuitos Integrados, SBCCI, pp. 32-37, Ouro Preto, Brasil.

Muñozb D.M., Llanos, C.H., Ayala-Rincon, M., Van-Els, R. (2006). "*Implementation, Simulation and Validation of Dispatching Algorithms for Elevator Systems."* In: Congreso Internacional de Reconfigurabilidad Reconfig06, pp. 290-297, San Luis Potosí, México.

Nikovsky, D., Brand, M. (2003). "*Decision-Theoretic Group Elevator Scheduling*." In: AAAI Proc. 13th Int. Conference on Automated Planning and Scheduling, pp. 133-142, Trento, Ilaty.

Otis Corporation. (acessado em 04/2005).

disponível em: http://www.otis.com/

Pepyne, D.L., Cassandras, C.G. (1997). "*Optimal Dispatching Control for Elevator Systems during Up-peak Traffic*." In: IEEE Transactions on Control Systems Technology, Vol. 5(6), pp. 629-643.

Plönnigs, J., Neugebauer, M., Kabitzsch, K. (2004). "*A Traffic Model for Networked Devices in the Building Automation."* In: IEEE Proceedings of the International Workshop on Factory Communication Systems, pp. 137-145, Vienna, Austria.

QuickLogic Corporation. (acessado em 06/2006).

Disponível em: http://www.quicklogic.com/

Roger, R. (2005). "*Industrialized Building System: Reproduction Before Automation and Robotics."* In: Automation in Construction, Vol. 14, pp. 442-451.

Sachs, H. (2005). "*Opportunities for Elevator Efficiency Improvements."* In: American Council for an Energy-Efficient Economy, pp. 1-11, Washington, USA.

Sasaki, K., Markon, S., Makagawa, M. (1996). "*Elevator Group Supervisory Control System Using Neural Networks."* In: January Elevator World Magazine.

Schneider, P., Huck, E., Schwarz, P., (2001). "*A Modeling Approach for Mechatronic Systems – Modeling and Simulation of an Elevator System."* In: XI Intern. Symposium in Theoretical Electrical Engineering, pp. 19-22, Linz, Austria.

Sectron Elevadores (acessado em 08/2006).

disponível em: http://www.sectron.com/

Shaw, I.S., Godoy, M. (1999). "*Controle e Modelagem Fuzzy."* In: Ed. Edgar Blücher Ltda, FAPESP, São Paulo, Brasil.

Siikonen, M.L. (1989). "*Computer Modeling of Elevator Traffic and Control."* In: Licientiate Thesis, Helsinki University of Technology, Helsinski, Finland.

Siikonen, M.L., Leppala, J. (1991). "*Elevator Traffic Pattern Recognition*." In: Proccedings of 4th International Conference on Fuzzy Systems Association, pp. 195-198, Brussels, Belgium.

Siikonen, M.L. (1993). "*Control Methods for an Elevator Group."* In: Proceedings of the

Nordic Operations Research Conference, NOAS93.

Siikonen, M.L. (1997). "*Elevator Group Control with Artificial Intelligence*." In: Research Report A67. System Analysis of Technology, Helsinki University of Technology, pp. 1-32, Helsinski, Finland.

Siikonenb, M.L. (1993). "*Planning and Control Models for Elevator in High-Rise Buildings*." In: Research Report A68. System Analysis of Technology, Helsinki University of Technology, pp. 1-39, Helsinski, Finland..

Sklyarov, V., Skliarova, I., Almeida, P., Almeida, M. (2003). "*Desing Tools and Reusable Libraries for FPGA-Based Digital Circuits."* In: IEEE Proc. Euromicro Symposium on Digital System Desing, pp. 255-263, Aveiro, Portugal.

So, A., Beebe, J., Chan, W., Liu, S. (1995). "*Elevator Traffic Pattern Recognition by Artificial Neural Networks."* In: Elevator Technology, Vol. 6, pp. 122-131.

Sorsa, J., Siikonen, M.L., Ehtamo, H. (2003). "*Optimal Control of Double-deck Elevator Group Using Genetic Algorithms."* In: International Transactions in Operational Research, Vol. 10, pp. 103-114.

Strakosch, G.R. (1998). "*Vertical Transportation: Elevators and Escalators."* In: Ed. John Wiley & Sons, New York, USA.

Taghavi, T., Ghiasi, S., Ranjan, A., Raje, S., Sarrafzadeh., M. (2004). "*Innovative or Perish: FPGA Physical Design."* In: International Symposium on Physical Design ISPD04, pp. 148-155, Phoenix, USA.

Takagi, T., Sugeno, M. (1985). "*Fuzzy Identification of Systems and its Applications to Modeling and Control."* In: IEEE Transactions Systems, Man, and Cybernetics, Vol. SMA-15(1), pp. 116-132.

Texas Instruments. (1985). "*Differential Bus Transceivers SN75176B",* SLLS101D, USA.

Thumm G. (1995). "*Elevators and Escalators Within Intelligent Buildings. New Methods and Technologies in Planning and Construction of Intelligent Buildings."* In: I Proceedings of IB/IC Intelligent Buildings Congress, Ed. A. Lustig, Tel-Aviv, Israel.

ThyssenKrupp Corporation (acessado em 04/2006).

Disponivel em: http://www.thyssenkruppelevadores.com/

ThyssenKruppb. (acessado em 08/2006).

Disponivel em: http://www.twin-elevator.com/

Tianxiao R. (2004). "*The Chinese Elevator Market and Elevator Industry."* In: Elevator World, Vol. 52(7), pp. 140, 142, 144, 146.

Tobita, T., Fujino, A., Segawa, K. (1996). "*A Parameter Tuning Method Using Genetic Algorithms for an Elevator Group Control System."* In: IEEE Proceedings of the 22nd International Conference on Industrial Electronics, Control, and Instrumentation, Vol. 2, pp. 823-828, Taipei, Taiwan.

Tsukamoto, Y. (1979). "*An Approach to Fuzzy Reasoning Method."* In: Advances in Fuzzy Set Theory and Applications, pp. 137-149, The Netherlands.

Tsuji, S., Amano, M. (1989). "*Application of the Expert System to the Elevator Group-Supervisory Control."* In: IEEE Proceedings of the 5th Conference on Artificial Intelligence Applications, pp. 287-294, Miami, USA.

Wang, S., Xie, J. (2002). "*Integrated Building Management System and Facilities Management on the Internet."* In: Automation in Construction 11, pp. 707-715.

Wang, S., Xie, J. (2004). "*Investigation on Intelligent Building Standard Communication Protocols and Application of IT Technologies."* In: Automation in Construction 13, pp. 607-619.

Xfuzzy 3.0. *Ferramenta de CAD para Lógica Difusa.* (acessado em 04/2005).

Disponível em: http://www.imse.cnm.es/Xfuzzy/Xfuzzy_3.0/Xfuzzy3.0_sp.pdf

WP Inovações Tecnológicas. (acessado em 03/2006).

Disponível em: http://www.wpautomacao.com.br/

Xilinx Corporation. "*Spartan3 Starter Kit User Guide*." (acessado em 06/2005).

Disponível em: http://www.xilinx.com/bvdocs/userguides/ug130.pdf

Xilinxb. "*Integrated Software Environment, ISE 7.1i."* (acessado em 06/2005).

Disponível em: http://toolbox.xilinx.com/docsan/xilinx7/books/docs/cgd/cgd.pdf

Xiong, Bo., Peter, B.L., Chang, S.C. (2005). "*Group Elevator Scheduling with Advanced Traffic Information for Normal Operations and Coordinated Emergency Evacuation.*" In: IEEE Proceedings of the International Conference on Robotics and Automation, pp.

1419-1424, Barcelona, Spain.

Lin, Yan., Li, Fei., He, Lei. (2005). "*Routing Track Duplication with Fine-Grained Power-Gating for FPGA Interconnect Power Reduction."* In: Proceedings of the 2005 Conference on Asia South Pacific Design Automation, pp. 645-650, Shanghai, China.

Zadeh, L.A. (1965). "*Fuzzy Sets."* In: Information and Control 8, University of California, pp. 338-353, USA.

Zeltwanger, H. (2004). "*CANopen Lift People."* In: CAN Newsletter Vol. 1, pp. 8-12.

Zong Q., Yue, Y. (2001). "*A Traffic Flow Forecasting Method of Elevator Group Control System."* In: Systems Engineering and Electronics, Vol. 23(7), pp. 102-104.