Veruska Rodrigues Moreira

Plataforma em hardware reconfigurável para o ensino e pesquisa em laboratório de sistemas digitais a distância

Dissertação de Mestrado apresentada à Faculdade de Engenharia Elétrica e de Computação como parte dos requisitos para obtenção do título de Mestre em Engenharia Elétrica. Área de concentração: Telecomunicações e Telemática.

Orientador: Prof. Dr. Dalton Soares Arantes
Co-orientador: Dr. Fabbryccio Akkazzha Chaves Machado Cardoso

Banca Examinadora:
Prof. Dr. Dalton Soares Arantes (Orientador) - FEEC/UNICAMP
Prof. Dr. Ivan Luiz Marques Ricarte - FEEC/UNICAMP
Prof. Dr. Luciano Leonel Mendes - INATEL

Campinas, SP
2009

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

M813p

Moreira, Veruska Rodrigues
Plataforma em hardware reconfigurável para o ensino e pesquisa em laboratório de sistemas digitais a distância Veruska Rodrigues Moreira. – Campinas, SP: [s.n.], 2009.

Orientadores: Dalton Soares Arantes; Fabbryccio Akkazzha Chaves Machado Cardoso.
Dissertação de Mestrado - Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Ensino a distância. 2. FPGA. 3. Serviços na Web. I. Arantes, Dalton Soares. II. Cardoso, Fabbryccio Akkazzha Chaves Machado. III. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. IV. Título

Título em Inglês:	Reconfigurable hardware platform for research and distance learning on remote laboratories for digital systems
Palavras-chave em Inglês:	Distance learning, FPGA, Web services
Área de concentração:	Telecomunicações e Telemática
Titulação:	Mestre em Engenharia Elétrica
Banca Examinadora:	Ivan Luiz Marques Ricarte, Luciano Leonel Mendes
Data da defesa:	21/12/2009
Programa de Pós Graduação:	Engenharia Elétrica

COMISSÃO JULGADORA - TESE DE MESTRADO

Candidata: Veruska Rodrigues Moreira

Data da Defesa: 21 de dezembro de 2009

Título da Tese: "Plataforma em Hardware Reconfigurável para Ensino e Pesquisa em Laboratório de Sistemas Digitais a Distância"

Prof. Dr. Dalton Soares Arantes (Presidente): _____

Prof. Dr. Luciano Leonel Mendes: _____

Prof. Dr. Ivan Luiz Marques Ricarte: _____

Resumo

Esta dissertação apresenta a concepção e o desenvolvimento de uma plataforma em *hardware* reconfigurável denominada REDLART - *REconfigurable Digital Laboratory for Advanced Research and Teaching*, visando soluções de laboratório a distância aplicadas ao ensino e ao trabalho colaborativo em sistemas digitais. A plataforma é baseada em dispositivos FPGA (*Field Programmable Gate Array*) para desenvolvimento de circuitos digitais com aplicações em processamento digital de sinais, sistemas de comunicações digitais, sistemas de controle e áreas afins. Além da plataforma de *hardware*, também foi concebida e implementada uma arquitetura de sistema, composta por um conjunto de *softwares* cliente-servidor, com o objetivo de viabilizar o acesso remoto através da gerência e da configuração de experimentos desenvolvidos na REDLART. Tal sistema, incluindo a própria REDLART, possibilita o desenvolvimento de novos experimentos e sua disponibilização na *Web*, resultando em um *WebLab* reconfigurável para sistemas digitais. Testes foram realizados em nível de *hardware* e *software* para a validação da plataforma.

Palavras-chave: Ensino a distância, FPGA, Serviços na Web.

Abstract

This thesis presents a reconfigurable hardware platform called REDLART (REconfigurable Digital Laboratory for Advanced Research and Teaching), designed to enable laboratory applications in distance learning and collaborative work in digital systems. The platform is based on FPGA devices (Field Programmable Gate Array) to develop digital circuit applications for digital signal processing, digital communication systems, control systems and related areas. Besides the hardware platform, a system architecture consisting of a set of client-server software was also designed and implemented in order to enable the remote access through the management and configuration of experiments developed in REDLART. By using the client-server software with the REDLART platform, new experiments can be developed and made available on the Web, resulting in a WebLab for reconfigurable digital systems. Tests were performed at the hardware and software levels for the validation of the platform.

Keywords: Distance learning, FPGA, Web services.

Agradecimentos

A Deus, pela sua presença constante em minha vida.

A toda a minha família, em especial os meus queridos pais Francisco Custódio Moreira e Maria Alice Rodrigues Moreira e ao meu irmão Jefferson Rodrigues Moreira pelo apoio e suporte durante esta jornada. Obrigada por estarem sempre ao meu lado, me apoiando, me incentivando e nunca medindo esforços para que eu pudesse alcançar todos os meus objetivos.

Ao meu orientador Prof. Dalton S. Arantes e ao meu co-orientador Dr. Fabbryccio A. C. M. Cardoso, pelas sugestões e ensinamentos, pela confiança e oportunidade de realizar este trabalho. Eu os considero os meus orientadores tanto para a ciência quanto para a vida.

Aos amigos Cecilio C. Fraguas e Fábio A. G. Lisboa pelas revisões, críticas e sugestões.

A todos os meus amigos, especialmente os amigos do ComLab (Laboratório KyaTera em Comunicações Digitais), da Ujima Software, do DCLab (Laboratório de Comunicações Digitais), da Moradia Estudantil e aos conquistados durante o mestrado. Obrigada pela ajuda constante!

A Roger N. Nascimento, pelo carinho, compreensão e companheirismo.

Aos funcionários da Faculdade de Engenharia Elétrica e Computação (FEEC) da Unicamp pela prestatividade.

À Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) pelo apoio financeiro.

À Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP) pelo apoio ao ComLab.

A todos que contribuíram direta ou indiretamente para a realização deste trabalho, muito obrigada!

"Comece fazendo o que é necessário, depois o que é possível, e de repente você estará fazendo o impossível."São Francisco de Assis

Sumário

Lista de Figuras

Lista de Tabelas

Glossário

AD - Analógico Digital

Ajax - *Asynchronous Java Script And XML*

API - *Application Programming Interface*

ARP - *Address Resolution Protocol*

ASIC - *Application Specific Integrated Circuits*

CDMA- *Code Division Multiple Access*

CPLD- *Complex Programmable Logic Device*

DA - Digital Analógico

DAO - *Data Access Object*

DRAM- *Dynamic Random Access Memory*

EDA- *Eletronic Design Automation*

EDIF- *Electronic Data Interchange Format*

FIFO - *First In First Out*

FPGA- *Field Programmable Gate Array*

Gbps- Giga bit por segundo

GSM- *Global System for Mobile Communication*

HDL- *Hardware Description Languages*

HPIB - *Hewlett-Packard Interface Bus*

HQL - *Hibernate Query Language*

HTTP - *HyperText Transfer Protocol*

I/O - *Input/Output*

IC- *Integrated Circuits*

IEEE- *Institute of Electrical and Electronics Engineers*

IP - *Internet Protocol*

J2EE - *Java 2 Enterprise Edition*

JSF - *JavaServer Faces*

JSP - *JavaServer Pages*

KyaTera - Kya (rede em Tupi-Guarani) Tera (Terabits por segundo)

LUT - *Lookup Table*

Mbps - Mega bits por segundo

MCP - *Monitoring and Control Protocol*

MPEG- *Moving Picture Experts Group*

MVC - *Model-View-Controller*

NCF- *Netlist Constraints File*

NPI - Núcleo de propriedade intelectual

OOP - *Object Oriented Programming*

PDS - Processamento Digital de Sinais

PLD- *Programmable Logic Device*

PROM - *Programmable Read-Only Memory*

RECOLAB- *Remote Control Laboratory*

REDLART- *Reconfigurable Digital Laboratory for Advanced Research and Teaching*

RTL- *Register Transfer Level*

SATA - *Serial Advanced Technology Attachment*

SPLD- *Simple Programmable Logic Device*

SQL - *Structured Query Language*

SRAM- *Static Random Access Memory*

TCP - *Transmission Control Protocol*

TDPR- *Timing Driven Place and Route*

TIDIA - Tecnologia da Informação no Desenvolvimento da Internet Avançada

UCF- *User Constraints File*

UDP - *User Datagram Protocol*

USB - *Universal Serial Bus*

VHSIC- *Very High Speed Integrated Circuit*

WCDMA- *Wide-Band Code-Division Multiple Access*

WiMAX- *Worldwide Interoperability for Microwave Access*

Trabalhos Publicados Pelo Autor

1. V. R. Moreira, F. A. C. M. Cardoso, D. S. Arantes. “Plataforma reconfigurável para ensino e pesquisa em laboratório de sistemas digitais a distância”. *Anais XIX Simpósio Brasileiro de Informática na Educação* (SBIE’2008), Sociedade Brasileira de Computação, Fortaleza, Ceará, Brasil, pg. 542-551, Novembro 2008.

Capítulo 1

Introdução

Este capítulo apresenta uma visão geral e os objetivos e motivações para o desenvolvimento de uma plataforma em *hardware* reconfigurável para o ensino e pesquisa em laboratórios de sistemas digitais a distância. Também são apresentados os trabalhos relacionados, as contribuições e a organização do presente trabalho.

1.1 Visão Geral

A educação a distância é uma modalidade de ensino e aprendizado planejado, na qual alunos e professores se encontram em locais diferentes durante a fase de ensino. Por estarem em locais distintos, é necessário fazer uso de alguma tecnologia que proporcione um ambiente interativo de comunicação entre as partes. Nesse contexto, deve-se ter um cuidado especial no processo de criação de um curso a distância, levando-se em conta a instrução e comunicação por meio de várias tecnologias, bem como de disposições organizacionais e administrativas especiais [1].

Atualmente, os professores e as instituições de ensino passaram a incentivar com mais ênfase as atividades práticas extra-classe e extra-curricular de maneira planejada e orientada, com o uso de diversas tecnologias para facilitar a comunicação. As ferramentas de ensino a distância se aproveitaram das tecnologias disponíveis de gerenciamento de recursos, troca de mensagens, e-mails, vídeos, áudio, apresentação de *slides*, entre outras, para aplicar metodologias de ensino e disseminar o conhecimento para além das paredes da Escola e da Universidade. Isso possibilita a difusão da informação, o trabalho colaborativo e a sinergia entre professores e alunos, o que estimula o processo de ensino e promove novas metodologias de aprendizado.

No entanto, cursos que possuem a característica de conciliar o conhecimento teórico com as práticas de laboratório têm demonstrado limitações no que se refere à sua aplicação a distância na educação superior. Um dos grandes desafios na colaboração e no ensino a distância está na viabilização da prática de laboratórios. Apesar de já existir tecnologia disponível para isso, ainda não há uma solução integrada de *software* e *hardware* que emule o ambiente laboratorial e possibilite uma experiência real e de fato produtiva com o laboratório remoto. O laboratório remoto proporciona aos estudantes e usuários interação com experimentos reais via Internet ou rede dedicada [2].

É importante observar que a experiência com a prática de laboratório deve familiarizar o aluno com o uso de equipamentos e de procedimentos laboratoriais, além de possibilitar o trabalho colaborativo e explorar conceitos teóricos. Esses elementos são importantes para viabilizar a formação de pessoal técnico capacitado para o mercado de trabalho.

Outro desafio que se apresenta reside na linha da pesquisa laboratorial colaborativa. O objetivo é compartilhar recursos e equipamentos entre instituições de ensino e pesquisa através de acesso remoto, com a possibilidade de ampliação da faixa de utilização desses equipamentos, otimizando o uso dos recursos e ampliando o número de alunos e pesquisadores envolvidos no processo.

Apesar de todo o recurso tecnológico existente, ainda não é trivial construir uma solução de acesso remoto laboratorial sobre uma infraestrutura presencial já existente. Entre os requisitos que devem ser atendidos, destacam-se a segurança e a eficiência do acesso remoto ao aparato experimental. É provável que a maior dificuldade esteja no fato de que cada laboratório seja um caso específico que demande soluções específicas, difíceis de serem reaplicadas diretamente a outros laboratórios.

O conceito de laboratório remoto surgiu da necessidade de disponibilizar laboratórios reais a qualquer momento, sem restrição de tempo e espaço. Este conceito exige que seja adotada uma nova abordagem de ensino, permitindo a troca de informações, interação entre alunos, professores e tutores e a flexibilização de local e de tempo de estudo. Idealmente, o objetivo é minimizar os problemas de logística, facilitar o compartilhamento de recursos e possibilitar o trabalho colaborativo. Deseja-se que o meio de desenvolvimento de experiências de laboratório remoto seja um ambiente de aprendizado no qual o aluno possa se envolver remotamente em atividades laboratoriais convencionais, com a mesma eficiência que teria se estivesse presente fisicamente no laboratório real.

Em linhas gerais, a solução padrão para possibilitar o acesso remoto a laboratórios reais consiste em utilizar ferramentas de automação. Essas ferramentas são utilizadas para fazer a aquisição, o controle e a monitoração de dados do experimento. A aquisição e a monitoração são realizadas a partir de equipamentos de teste ou de cartões de aquisição específicos. De forma semelhante, o controle do experimento é possibilitado a partir de equipamentos de teste ou de controladores projetados especificamente para o experimento.

Este trabalho propõe o desenvolvimento de uma plataforma em *hardware* reconfigurável para o ensino e pesquisa a distância denominada REDLART (*REconfigurable Digital Laboratory for Advanced Research and Teaching*). Essa proposta permite que alunos e pesquisadores programem os seus próprios experimentos e os integrem na plataforma, aproveitando assim todos os benefícios de controle, monitoração e distribuição do experimento que a arquitetura da plataforma oferece. Esta plataforma foi projetada para viabilizar o conceito da prática de laboratório a distância, possibilitando a colaboração na pesquisa prática e no ensino de laboratório a distância.

A plataforma é baseada em dispositivos FPGA (*Field Programmable Gate Array*) para desenvolvimento de circuitos digitais com aplicações em processamento digital de sinais, sistemas de comunicações digitais, controle, entre outras. A plataforma foi projetada e desenvolvida para a prototipagem

rápida de laboratórios remotos nas áreas de engenharia, telecomunicações, computação e afins. Além da plataforma de *hardware*, também foi concebida e implementada uma arquitetura de sistema, composta por um conjunto de *softwares* cliente-servidor, para viabilizar o acesso remoto, a gerência e a configuração do experimento. Nesse contexto, a partir de provas de conceito, mostra-se que é possível acessar experimentos remotos que podem ser utilizados na prática de ensino de laboratório a distância. Além disso, mostra-se também que é possível viabilizar o trabalho colaborativo e a sinergia entre laboratórios de pesquisa pelo controle e monitoração de experimentos distribuídos em redes de pacotes de alta velocidade.

1.2 Trabalhos Relacionados

Alguns trabalhos correlatos motivaram de alguma maneira o desenvolvimento desta dissertação. Dentre eles podemos citar a disciplina Simulação, Co-simulação e Prototipagem de Sistemas de Comunicações Digitais (IE344B), a plataforma RECOLAB, o projeto Laboratório em Casa: Kits de *Hardware* para um Laboratório de Projeto Digital e o Laboratório Remoto Aplicado ao Ensino de Engenharia Eletrônica.

O projeto da plataforma REDLART surgiu da prática da disciplina de pós-graduação IE-344B [3] oferecida na Faculdade de Engenharia Elétrica e Computação da Universidade Estadual de Campinas. Essa disciplina aborda os paradigmas de implementação de sistemas de comunicação em *hardware* reconfigurável. Para testar os conhecimentos adquiridos, os alunos são convidados a desenvolver experimentos práticos de sistemas ou subsistemas de comunicações digitais e processamento digital de sinais utilizando FPGA. Durante o oferecimento dessa disciplina, constatou-se não ser necessário especificar uma arquitetura de *software* e de *hardware* que seja totalmente nova para cada novo projeto de experimento a ser desenvolvido. O mais racional neste caso é reutilizar uma infraestrutura básica que possa ser facilmente alterada através de um único componente funcional da arquitetura.

Uma proposta de laboratório remoto de controle denominada RECOLAB (*Remote Control Laboratory*) é apresentada. A plataforma RECOLAB visa proporcionar uma arquitetura geral para a execução remota de sistemas de controle automático de processo, referente à disciplina de engenharia de controle. O objetivo da plataforma é controlar os sistemas físicos de controle de processo em tempo real e de forma remota através da Internet. A plataforma foi desenvolvida utilizando as ferramentas Matlab e Simulink [4]. Através dessa plataforma é possível visualizar as informações associadas ao desempenho da simulação, assim como os gráficos, os dados e os resultados relevantes. Essa plataforma é empregada nos contextos de pesquisa colaborativa e ensino a distância [5]. Neste caso, o RECOLAB é uma plataforma de *software* que possibilita controlar e acessar processos mecânicos automatizados. Por outro lado, a REDLART é uma plataforma de *hardware* que possibilita desenvolver novos experimentos em uma plataforma já automatizada e integrada em um sistema para acesso remoto pela *Web*.

Uma inovadora metodologia de laboratório chamada Laboratório em Casa: Kits de *Hardware* para um Laboratório de Projeto Digital é apresentada para o curso introdutório de projeto digital. O

conceito de laboratório em casa permite ao estudante executar experimentos reais em sua casa, utilizando kits de *hardware* especialmente desenvolvido para esta finalidade. Os estudantes analisam determinados problemas, desenvolvem as soluções que julgarem mais apropriadas e testam as soluções desenvolvidas nos kits de *hardware*, cada um composto principalmente por uma placa lógica programável [6]. Embora possibilite que o experimento seja executado fora do ambiente do laboratório (em casa), essa metodologia não emprega de fato o conceito de laboratório a distância, não possibilitando uma interação remota com um laboratório real.

Um laboratório remoto interativo para aplicações educacionais na área de engenharia eletrônica é apresentado. O laboratório utiliza o serviço *Web* da Internet como infraestrutura de comunicação, aplicativos para instrumentação baseada em computador e uma placa eletrônica para controle de experimentos reais. O laboratório foi desenvolvido utilizando a ferramenta de automação LabView [7]. O laboratório remoto possibilita ao usuário atuar nos experimentos acionando chaves, potenciômetros digitais, entre outros e obter dados com as medidas de instrumentos comerciais [8]. Neste caso, o laboratório remoto consiste em uma solução de automatização de experimentos e não no desenvolvimento do experimento propriamente dito. É importante destacar que a REDLART possibilita desenvolver novos experimentos a partir de circuitos digitais com foco no problema. Finalizado o desenvolvimento do experimento, o mesmo já está automatizado e pronto para ser integrado no sistema de acesso.

1.3 Objetivos

A proposta desta dissertação consiste no estudo e desenvolvimento de uma plataforma experimental para a prática de laboratório a distância com aplicações em ensino e pesquisa colaborativa. Nesta linha de pesquisa, o objetivo é padronizar uma plataforma em *hardware* reconfigurável denominada REDLART, baseada em dispositivos semicondutores do tipo FPGA que incorpore soluções de controle e monitoração aos módulos de sua arquitetura. A plataforma proposta possibilita a criação de experimentos que possam ser acessados remotamente ou utilizados diretamente no laboratório.

A arquitetura da plataforma REDLART possibilita a criação de experimentos de sistemas digitais a partir de um conjunto de serviços de *software* que isolam completamente a complexidade do *hardware* utilizado. Esses serviços são modularizados e organizados em uma arquitetura genérica e flexível que permite que os experimentos sejam desenvolvidos com foco no problema que se deseja resolver. A plataforma REDLART abstrai os detalhes exaustivos da implementação dos componentes do *hardware*. Essa proposta permite que alunos e pesquisadores implementem os seus próprios experimentos e os integrem à plataforma, aproveitando assim todos os benefícios de controle, monitoração e distribuição do experimento em rede que a arquitetura da plataforma oferece. Distribuir um experimento em rede significa implementar partes do mesmo em locais remotos e integrá-los automaticamente através da comunicação por fluxos de dados de tempo real em redes de computadores. Esse conceito é explorado nesta dissertação para viabilizar a colaboração em pesquisa aplicada.

A plataforma foi projetada e desenvolvida para a prototipagem rápida de novos experimentos em sistemas digitais nas áreas de engenharia, telecomunicações, computação e afins. Exemplos de tais

sistemas são módulos de compressão/descompressão de áudio e vídeo (MPEG, H.264, MP3, etc.), sistemas de transmissão digital para TV Digital, redes sem fio (Wi-Fi, WiMAX, etc.), telefonia celular (CDMA, WCDMA, GSM, etc.), entre muitas outras aplicações. A partir de provas de conceito, mostra-se que é possível utilizar a plataforma proposta como uma ferramenta complementar de auxílio e apoio à prática de ensino de laboratório a distância e ao trabalho colaborativo.

1.4 Motivações e Contribuições

O conceito de laboratório remoto permite que o usuário acesse e interaja com experimentos em tempo real. Os laboratórios remotos têm sido propostos, usualmente, como ferramentas de apoio e suporte ao ensino. Há um grande esforço da comunidade acadêmica nacional e internacional em disponibilizar laboratórios remotos em diversas áreas do conhecimento. Com o advento de novas infraestruturas de redes de alta velocidade para viabilizar o trabalho colaborativo, esses esforços têm se intensificado. Neste trabalho, além da aplicação em ensino, investiga-se também como viabilizar a colaboração na pesquisa aplicada. A solução empregada consiste em que partes do experimento possam ser desenvolvidas remotamente e então integradas, sem a necessidade de deslocamento. A proposta é que tais partes possam se comunicar através de fluxos de dados de tempo real em uma infraestrutura de rede de computadores como a Internet.

Em geral, as soluções apresentadas para prover a automatização dos experimentos são baseadas em *software* cliente-servidor ou em *software* para automação industrial. Existem também algumas propostas que são favoráveis à disponibilização do *hardware* necessário para que o aluno ou pesquisador desenvolva o experimento fora do ambiente da universidade. De modo geral, as propostas que estão relacionadas à aplicação da arquitetura de *software* cliente-servidor no contexto de laboratório remoto estão focadas no gerenciamento do usuário e nas metodologias que podem ser empregadas no aprendizado. Entretanto, possuem limitações no que diz respeito à prática de laboratório remoto. Essas limitações podem ser percebidas através da dificuldade de compartilhar recursos, fornecer o acesso ao experimento em tempo real, integrar novos experimentos ou equipamentos ou alterar os já existentes. Além disso, os experimentos que não são concebidos para serem compartilhados requerem automatização externa para que os usuários possam ter acesso.

A plataforma REDLART investe na questão do acesso em tempo real através da possibilidade de fluxo de dados de tempo real (*streaming*) para monitoração do experimento e para comunicação entre partes do experimento. A possibilidade de alterar um experimento na REDLART se deve à natureza reconfigurável dos dispositivos semicondutores utilizados, neste caso as FPGAs. A integração de novos experimentos é facilitada pela arquitetura sistêmica de *software-hardware-infraestrutura de rede* apresentada neste trabalho. O compartilhamento de recursos é abordado na plataforma não apenas pelo acesso remoto, mas também na possibilidade de reutilização de um experimento remoto para conceber um novo experimento também remoto. Tal abordagem é viabilizada pelo recurso de distribuição em rede do experimento.

De outra maneira, as soluções de automatização de experimentos que são desenvolvidas com base em ferramentas de *software* de automação possuem a característica de serem desenvolvidas através

da prototipagem rápida. No entanto, como fator limitante, essas soluções necessitam utilizar *software* proprietário no terminal do usuário. Isso pode inviabilizar o custo ou a execução dessa operação. Além disso, as soluções disponíveis até o momento são apresentadas como laboratórios remotos estáticos. Após o experimento estar disponível pouca ou nenhuma alteração pode ser realizada na estrutura ou na arquitetura do experimento. Conforme citado anteriormente, a plataforma utiliza dispositivos semicondutores reconfiguráveis (FPGAs) que possibilitam que o experimento seja recompilado. A arquitetura do sistema, por sua vez, possibilita que o experimento seja recarregado a distância.

A situação ideal, que é abordada neste trabalho, é ter disponível uma plataforma de *hardware* integrada a uma plataforma em *software* que possibilite a experiência de execução de laboratório remoto em condições semelhantes às encontradas em um laboratório presencial. O usuário deverá ter a liberdade de criar ou alterar os experimentos e executá-los sem restrição de tempo ou espaço. Para que a experiência com o experimento remoto emule a experiência com o laboratório presencial é importante que a infraestrutura de rede no qual se encontra o experimento seja robusta e confiável para evitar atrasos e perdas de pacotes. Também é importante que a plataforma de *software* que compõe o laboratório remoto forneça um ambiente amigável para possibilitar a interação entre alunos, tutores e professores. O ambiente de laboratório remoto deve disponibilizar recursos para auxiliar o processo de execução do experimento, como material de apoio, correio eletrônico, grupos de discussão, mural, entre outros.

Um dos esforços deste trabalho foi desenvolver e implementar uma plataforma em *hardware* reconfigurável que possa fornecer suporte para a prática de laboratório a distância, seja no ensino ou na colaboração em pesquisa aplicada. A arquitetura da plataforma proposta possibilita que alunos e pesquisadores desenvolvam seus próprios experimentos e os integrem na plataforma. A plataforma REDLART foi concebida para possibilitar o controle, a monitoração e a distribuição do experimento em rede. Ela foi desenvolvida parte em ambiente Matlab/Simulink através de ferramenta de desenvolvimento de sistemas para FPGA da Xilinx, o *System Generator* [9], parte em linguagem padrão de descrição de *hardware* chamada HDL (*Hardware Description Languages*). É importante destacar que experimentos podem ser concebidos utilizando apenas o ambiente de desenvolvimento sistêmico do *System Generator*.

Como plataforma de *hardware* para desenvolvimento de experimentos, é natural que a REDLART tenha limitações no que se refere a realizar o gerenciamento de usuário, gerenciamento do experimento e fornecer ferramentas que auxiliam na metodologia de ensino, como *chats*, mural, fóruns entre outros. Tais características são típicas de plataformas de *software*. Neste contexto, também é apresentada uma proposta de arquitetura de sistema, utilizando uma plataforma de *software* baseada na arquitetura cliente-servidor, que possibilita o gerenciamento do usuário e do experimento e que fornece uma interface de usuário que seja acessível para a execução do experimento.

A plataforma REDLART possibilita a flexibilização de horários para a execução de experimentos. O laboratório remoto pode ficar disponível 24 horas por dia. Os alunos possuem liberdade para gerenciar o tempo dedicado à prática do experimento, decidir o que praticar e ter a liberdade para cometer erros e aprender com eles, o que estimula a pró-atividade. A plataforma pode motivar os alunos a realizarem experiências práticas laboratoriais a distância, fornecendo a oportunidade de tes-

tar o conhecimento conceitual obtido nas aulas teóricas convencionais. Isso pode aumentar a sinergia entre professores, alunos e tutores em benefício do próprio aluno.

1.5 Organização da Dissertação

Neste capítulo, apresentou-se o tema da dissertação, uma visão geral do contexto no qual pode ser aplicado este trabalho e os trabalhos relacionados. Em seguida, foram descritos os objetivos, as motivações para o desenvolvimento da plataforma REDLART e a contribuição do presente trabalho. Posteriormente, será descrito sucintamente cada capítulo desta dissertação. Este trabalho está estruturado em sete capítulos e organizado da forma descrita a seguir.

O Capítulo 2 inicia apresentando os conceitos de ensino a distância e de laboratórios remotos. Durante o acesso a laboratório remoto, ao enviar um comando para o experimento, espera-se que tal comando tenha um reflexo praticamente imediato. Para que isso ocorra é importante que a rede de computadores, no qual o experimento está sendo executado, seja robusta. Por isso, são apresentadas algumas redes de alta velocidade que estão disponíveis para o ensino e pesquisa colaborativa. Alguns exemplos são apresentados para ilustrar o esforço da comunidade acadêmica nacional e internacional em viabilizar o desenvolvimento e implantação de laboratórios remotos. Para concluir, é apresentado o trabalho proposto no contexto educacional.

O trabalho proposto é baseado em uma plataforma em *hardware* reconfigurável que utiliza dispositivos semicondutores do tipo FPGA. Por isso, o Capítulo 3 inicia apresentando a história do surgimento das FPGAs, assim como os conceitos relacionados a esses dispositivos lógicos programáveis. Em seguida, são apresentadas as formas de desenvolvimento e implementação de um projeto que utiliza esse tipo de dispositivo. Por fim, são apresentados os principais fornecedores e desenvolvedores de ferramentas de *software* para a programação do *chip* FPGA.

O Capítulo 4 apresenta uma proposta de arquitetura de sistema para a aplicação de laboratório remoto. Essa arquitetura é composta por uma plataforma de *software* cliente-servidor que é responsável por fazer o gerenciamento de usuário, o gerenciamento de experimentos e a interação com a plataforma em *hardware* reconfigurável para a execução de experimento reais. A modelagem da arquitetura sistêmica, o desenvolvimento e a implementação da plataforma de *software* cliente-servidor é apresentada neste capítulo.

O Capítulo 5 apresenta a plataforma em *hardware* reconfigurável REDLART. Nesse capítulo são abordados os tópicos referente a especificação de requisitos, a modelagem, os cenários de utilização, o desenvolvimento e a implementação da plataforma REDLART. Para fornecer suporte à troca de informações entre a plataforma de *software* cliente-servidor e a plataforma REDLART foi desenvolvido um protocolo denominado protocolo MCP (*Monitoring and Control Protocol*). A especificação, a modelagem e a implementação desse protocolo também são apresentadas nesse capítulo.

O Capítulo 6 mostra três testes que foram realizados para validar a plataforma REDLART. A pro-

posta foi desenvolver experimentos de processamento digital de sinais disponibilizados para acesso remoto e no local do laboratório. Tais experimentos foram desenvolvidos utilizando a plataforma REDLART e a plataforma de *software* cliente-servidor para verificar o comportamento dessas plataformas e testar as funcionalidades por elas fornecidas.

Finalmente, o Capítulo 7 mostra um breve resumo dos principais tópicos abordados neste trabalho, assim como as considerações finais e a conclusão. Também são apresentados alguns tópicos que não foram implementados neste trabalho, mas serão estudados e analisados em trabalhos futuros.

Capítulo 2

Prática de Laboratório a Distância

O objetivo deste capítulo é introduzir o conceito de laboratório a distância e sua aplicação no ensino e na pesquisa colaborativa. Também é feita uma introdução sobre a infraestrutura necessária para viabilizar a prática de laboratório remoto. Uma revisão sobre exemplos de laboratórios a distância e uma breve descrição de como o trabalho proposto pode ser inserido no contexto educacional são também apresentadas.

2.1 Ensino a Distância

A educação a distância é a modalidade educacional na qual a mediação didático-pedagógica nos processos de ensino e aprendizagem ocorre com a utilização de meios e tecnologias de informação e comunicação, com estudantes e professores desenvolvendo atividades educacionais em lugares ou tempos diversos. Essa definição está presente no decreto 5.622, de 19.12.2005 que regulamenta o Art. 80 da Lei 9394/96 (Lei de Diretrizes e Bases da Educação Nacional).

Recentemente têm surgido diversas discussões políticas e pedagógicas de ação educativa no nível institucional e governamental sobre o tema educação a distância. A proposta de educação a distância surgiu como alternativa para suprir as necessidades de ensino que os métodos convencionais não contemplam. A educação a distância pode atender às necessidades de nivelar desigualdades entre grupos etários, proporcionar treinamento de emergência para públicos-alvo específicos, promover a redução de custos dos recursos educacionais, aumentar as oportunidades de aprendizado e estimular melhorias na qualidade das estruturas educacionais existentes.

Entretanto, cursos que possuem a característica de conciliar o conhecimento teórico com as práticas de laboratório, como engenharias e enfermagem, têm demonstrado limitações no que se refere à sua aplicação a distância na educação superior. Nas atividades laboratoriais convencionais, além da fundamentação teórica, são necessários materiais de leitura e a experiência de fazer (*hands-on*) proporcionada pelos laboratórios. Todos esses elementos em conjunto são vitais na formação de qualidade [10]. Desenvolver o ensino e a pesquisa em ambiente de laboratório fornece aos alunos a oportunidade de testar o conhecimento conceitual, trabalhar colaborativamente, interagir com equipamentos, aprender por tentativa e erro e fazer análises em dados experimentais [11].

A implantação de um curso a distância, com grade curricular mais voltada para a área técnica, pode ser viabilizada quando as disciplinas são realizadas de forma semi-presencial. Entretanto, a realização plena de tais cursos a distância também pode ser possível a partir do acesso remoto a laboratórios reais ou mesmo a partir do uso de laboratórios virtuais. Para alguns cursos, a possibilidade de acesso remoto é uma solução não apenas para viabilizar o ensino da prática a distância, mas também para contornar situações onde há falta de recursos, problemas logísticos, deficiência e risco ou condições perigosas de ambiente que tornam a prática de laboratório difícil [12]. Dessa necessidade de tornar os laboratórios disponíveis para que possam ser acessados a qualquer momento, desvinculando as restrições de tempo e espaço, junto com o desejo de minimizar os problemas logísticos e compartilhar recursos, surgiu o conceito de laboratório a distância.

2.2 Laboratório a Distância

A consolidação do uso da Internet como ferramenta de compartilhamento de informações, o desenvolvimento de ferramentas modernas de controle e aquisição e a distribuição de dados via rede de computadores vêm impulsionando o controle e o acompanhamento de ensaios experimentais remotos [13].

Segundo Casini et al., os laboratórios a distância dividem-se em duas classes: laboratórios virtuais e laboratórios remotos. Os laboratórios virtuais são sistemas que podem executar simulações remotamente com a possibilidade de animação do sistema controlado. Os laboratórios remotos são laboratórios onde os estudantes podem interagir com experimentos reais via Internet. As operações remotas são realizadas através de uma interface *Web* que possibilita alterar os parâmetros de controle, executar o experimento, ver os resultados e obter os dados [2].

Os laboratórios virtuais são formados por um conjunto de modelagem, simulações e visualizações do que seria uma experiência desenvolvida em um laboratório real. De fato, esses laboratórios não existem no mundo físico, real, o que para alguns pesquisadores é motivo de relutar em classificar de laboratório no sentido estrito da palavra. No entanto, eles podem ser desenvolvidos para cumprir alguns requisitos de laboratório e fornecer uma visão geral de como seria a experiência prática desse laboratório. Os laboratórios virtuais podem ser utilizados no sentido de ser mais uma ferramenta para colaboração no ensino de aulas de laboratório e não como substituição da mesma.

O conceito de laboratório remoto une a característica de laboratório real e laboratório virtual, pois permite que usuários remotos acessem experimentos reais. Os laboratórios remotos admitem a execução do experimento passo a passo em tempo real, possibilitam a análise dos dados em equipamentos reais, permitem a flexibilização de horário e local para realizar os experimentos e estimulam o compartilhamento de equipamentos.

Ao prover acesso a experimentos remotos, pode-se atender à demanda existente de ensino do uso de equipamentos técnicos complexos. Além disso, pode-se também suprir a necessidade de se introduzirem práticas consideradas estado da arte na área de estudo e ainda preencher as expectativas do

que é esperado de um profissional qualificado. Ao prover acesso a experimentos envolvendo equipamentos de custo elevado e algumas vezes sensíveis, pode-se ampliar a faixa de utilização desses equipamentos, o que de certo modo proporciona uma economia de escala [14].

A aplicação de laboratório remoto é muito diversificada. Pode ser utilizado no desenvolvimento de um único experimento ou em um conjunto pré-determinado de experiências ou até mesmo para integrar laboratórios remotos de alcance mundial, formados pela conexão de várias instalações menores. Por outro lado, para permitir a integração de vários laboratórios remotos ou simplesmente para executar um único experimento de forma satisfatória, muitas vezes é necessário ter disponível uma boa infraestrutura de rede de computadores, com requisitos de desempenho, como largura de banda, atrasos e perdas de pacotes, suficientes para suportar as aplicações desejadas.

2.3 Redes de Computadores

Durante o acesso a laboratórios remotos, dependendo do experimento, o usuário poderá ter acesso simultâneo a diversos sinais multimídia, tais como fluxo de dados de alta taxa em tempo real, monitoração de sensores, monitoração de equipamentos de teste (osciloscópio, analisador de espectro, etc.) e fluxos de áudio e vídeo. Além disso, o usuário poderá interagir com o experimento através de um aplicativo que enviará sinais de controle a partir de comandos de uma interface gráfica.

Ao enviar um comando para o experimento remoto, espera-se que tal comando tenha um reflexo praticamente imediato nos sinais de tempo real que estão sendo monitorados. Para atender a todos os requisitos mencionados, possibilitando que o usuário tenha uma dinâmica com o laboratório remoto semelhante à do laboratório presencial, é importante que a rede de computadores, na qual o experimento está sendo executado, ofereça largura de banda, confiabilidade, baixo atraso e baixa variação do atraso (*jitter*), de forma que minimize possíveis perdas e atrasos excessivos na transferência de dados.

Em Teitelman and Hanss [15] é definida largura de banda como a taxa de transmissão máxima que pode ser sustentada entre dois pontos. Atraso é o tempo necessário para um pacote percorrer a rede, medido no momento em que é transmitido pelo emissor até ser recebido pelo receptor. *Jitter* é a variação no atraso fim a fim. Quando é utilizado serviço de rede não orientado a conexão, um pacote enviado depois de outro pode chegar primeiro no receptor, caso o *jitter* tenha valores elevados. Isso pode levar a uma recuperação incorreta dos dados enviados.

A recuperação incorreta dos dados não ocorre caso seja utilizado um serviço orientado a conexão. Nesses serviços os pacotes fora de ordem são reordenados quando chegam ao destino. Porém, a operação de ordenar os pacotes pode inserir atrasos na comunicação, dificultando a comunicação em tempo real. Por esse motivo, a comunicação em tempo real geralmente é feita utilizando um serviço não orientado a conexão.

Algumas redes de alta velocidade estão disponíveis para o ensino e pesquisa. Entre elas pode-se citar a rede KyaTera, a rede Giga e a rede Internet2, descritas com mais detalhes a seguir.

2.3.1 Projeto TIDIA-KyaTera

O programa TIDIA (Tecnologia da Informação no Desenvolvimento da Internet Avançada) foi lançado em São Paulo, Brasil, no segundo semestre de 2003. Esse programa é financiado pela FAPESP (Fundação de Amparo a Pesquisa do Estado de São Paulo) e inclui três programas colaborativos: KyaTera, Aprendizado Eletrônico e Incubadora Virtual.

Os projetos Aprendizado Eletrônico e Incubadora Virtual são responsáveis pela pesquisa e desenvolvimento colaborativo no ensino a distância e na geração de conteúdos, respectivamente, suportados por redes de alta velocidade. O projeto KyaTera tem por objetivo a construção de uma infraestrutura de rede de altíssima velocidade que possa operar como uma plataforma de testes para a pesquisa e desenvolvimento de *WebLabs*, de protocolos avançados e de novas tecnologias em infraestrutura/*hardware* de rede e em *software*.

Dentro do programa TIDIA-KyaTera, vale destacar a pesquisa e o desenvolvimento de *WebLabs*. O conceito de *WebLab* foi definido para aqueles serviços *Web* cujo objetivo seja prover acesso remoto a experimentos de laboratórios reais a partir de páginas da *Web*. Esse serviço oferece acesso comum a dados geograficamente distribuídos. Permite que os usuários possam utilizar todos os recursos de um equipamento remoto através de um navegador, sem a necessidade de investir grandes recursos na aquisição de equipamentos laboratoriais [16].

A FAPESP define o projeto KyaTera como um projeto cooperativo que consiste de uma rede de fibras ópticas projetada para a pesquisa e desenvolvimento de conexões de alta velocidade, integrando laboratórios de pesquisa que focam o estudo, desenvolvimento e demonstração de tecnologia e aplicações da Internet avançada.

A rede KyaTera interliga todos os laboratórios participantes por meio de fibras ópticas que chegam diretamente aos laboratórios associados (*fiber-to-the-lab*). Através de um enlace de 1,2 Gbps entre São Paulo e Miami, a rede conecta os pesquisadores do estado de São Paulo com outras redes experimentais de pesquisa no mundo, incentivando colaborações internacionais. Internamente, no estado de São Paulo, essa rede está subdividida em duas: Rede Experimental e Rede Estável.

A rede estável oferece, para cada laboratório, uma taxa de transmissão mínima de 1 Gbps, através de um *backbone*[1] de 10 Gbps que interconecta os três grandes centros concentradores da rede, localizados nas cidades de São Paulo, Campinas e São Carlos. A rede experimental oferece uma quantidade variável de fibras apagadas, monomodo e/ou multimodo, de acordo com as necessidades do laboratório. A rede experimental obteve uma taxa de transmissão máxima de 320 Gbps em testes realizados em abril de 2005 [17].

2.3.2 Projeto GIGA

O Projeto Giga consiste na implementação e no uso de uma rede óptica experimental voltada para o desenvolvimento de tecnologias de rede óptica, aplicações e serviços de telecomunicação associados

[1] fornece um caminho para a troca de informações entre redes diferentes.

à tecnologia e banda larga. Também prevê a transferência de tecnologia a empresas brasileiras [18].

A rede do Projeto Giga foi implantada em maio de 2004, com 735 km de extensão e capacidade de 2,5 Gbps, podendo chegar até 10 Gbps. Abrange os municípios de Campinas, São Paulo, São José dos Campos, Cachoeira Paulista, Rio de Janeiro, Niterói e Petrópolis. A rede interconecta 17 universidades e centros de pesquisa do eixo Rio - São Paulo. O projeto Giga é coordenado pela RNP (Rede Nacional de Ensino e Pesquisa) em parceria com o CPqD (Centro de Pesquisa e Desenvolvimento em Telecomunicações). O convênio foi assinado com a FINEP (Financiadora de Estudos e Projetos), em dezembro de 2002.

2.3.3 Rede Internet2

A rede Internet2 fornece aos Estados Unidos pesquisa e ensino em uma comunidade dinâmica, inovadora e com boa relação custo/benefício de uma rede óptica híbrida. A rede foi projetada para proporcionar a produção da próxima geração de serviço, assim como prover uma plataforma para o desenvolvimento de novas ideias e protocolos de rede. A rede Internet2 oferece infraestrutura de tecnologias de alta largura de banda. O *backbone* cruza o país com o apoio do Centro de Operações de Redes e Pesquisa Global e permite o fornecimento de 10 Gbps nos circuitos. A meta é oferecer 100 Mbps de conectividade de rede entre todos os *desktop* ligados à Internet2 [19].

É importante destacar que os laboratórios a distância, em princípio, devem ser projetados para serem acessados através da Internet atual. Como vimos anteriormente, algumas redes acadêmicas estão em processo de implantação e outras estão restritas a uma região de atuação. Por isso, é importante que no momento de projetar novos experimentos a distância os desenvolvedores pensem nas condições físicas da rede onde será executado o experimento. Redes congestionadas podem gerar perdas de pacote de dados e atrasos que prejudicam a interatividade com o experimento em tempo real.

2.4 Exemplos de Laboratório Remoto

Esta seção apresenta uma breve revisão para ilustrar o esforço da comunidade acadêmica nacional e internacional em viabilizar o desenvolvimento e implantação de laboratórios remotos para fins educacionais e de pesquisa colaborativa.

Em 1999 foi apresentado um módulo de curso a distância com aplicação na caracterização de dispositivos semi-condutores. O curso foi desenvolvido seguindo o modelo Cliente/Servidor, constituído de um servidor local no qual estavam instalados equipamentos que se comunicavam por meio do padrão HPIB (*Hewlett-Packard Interface Bus*) [20]. O público alvo do curso eram alunos da graduação e da pós-graduação que acessavam os instrumentos por meio de um navegador *Web* [21].

Casini et al [2] apresentou em 2002 um laboratório de física a distância constituído de sistemas hidráulicos, motores de corrente contínua e de um levitador magnético desenvolvido no ambiente Matlab/Simulink (MathWorks). O material didático foi fornecido para ajudar no aprendizado. Ao término do experimento um conjunto de dados é apresentado. A interação com o aluno ocorre por

meio de um navegador *Web*.

A *Trinity College Dublin* em parceria com a *Open University in Milton Keynes*, Reino Unido, desenvolveu um sistema chamado PEARL (*Practical Experimentation by Acessible Remote Learning*) que consiste de uma arquitetura genérica para acessar experimentos remotos. O PEARL é utilizado em cursos de visão computacional para inspeção visual e processamento de imagem digital. O sistema tem como base a arquitetura Cliente/Servidor implementado utilizando as tecnologias Java Applet (Sun Microsystems) e Corba (*Common Object Request Broker Architecture*) [22].

No *Massachussetts Institute of Technology* (MIT) foi desenvolvida uma arquitetura para laboratório remoto denominado iLabs. O iLabs é formado por diversos laboratórios remotos desenvolvidos no MIT nas áreas de microeletrônica, engenharia química, cristalização de polímeros, engenharia estrutural e processamento de sinal. Esses laboratórios remotos são casos de estudo para entender os complexos requisitos de operar experimentos remotos de equipamentos de custo elevado e escalar. Eles são utilizados para grandes grupos de estudantes do MIT e de outras universidades no mundo [23].

Na Universidade de Genova, Itália, foi desenvolvido o ISILab. O ISILab é composto de um módulo de experimento virtual e um módulo de experimento remoto para o ensino de eletrônica. Uma característica do ISILab é apresentar uma estrutura escalar e modular, o que permite incorporar outras experiências através do desenvolvimento de placas dedicadas ao sistema desenvolvido [24].

Na Universidade Federal de São Carlos (UFSCAR) foram desenvolvidos diversos *WebLabs*, disponibilizados através da rede KyaTera, para auxiliar no ensino de graduação. Entre eles, pode-se destacar o *WebLab* de hidrólise enzimática da sacarose ($C_{12}H_{22}O_{11}$), cujo objetivo é a execução remota de experimentos de inversão de sacarose em tempo real. Esse *WebLab* foi implementado através de uma solução híbrida que utiliza o Labview (*National Instruments*) e Java [25].

Na Universidade Estadual de Campinas (UNICAMP) encontram-se disponíveis, para acesso através da rede KyaTera, *WebLabs* na área de óptica, fotônica, engenharia química e bioquímica. Para integrar os *WebLabs*, pesquisadores do ComLab [26] desenvolveram um *framework* com base no paradigma Ajax (*Asynchronous Java Script And XML*). A proposta desse *framework* é uma arquitetura modular bastante flexível que permite adicionar, remover ou atualizar os *WebLabs* sem a necessidade de recompilar o núcleo da plataforma. Isso garante compartilhamento de recursos e facilidade de manutenção [27].

2.5 O Trabalho Proposto no Contexto Educacional

Os laboratórios a distância apresentados, de maneira geral, são *softwares* desenvolvidos para que os usuários tenham acesso a equipamentos reais. Alguns desses laboratórios remotos possuem a característica de serem modulares, ou seja, permitem que sejam integrados novos laboratórios remotos aos já existentes. Outros possuem arquitetura pouco flexível, o que dificulta a inclusão de novas funcionalidades. No geral, o foco de desenvolvimento desses laboratórios a distância está no *software*

cliente-servidor. Depois do *software* finalizado, pouca ou nenhuma reconfiguração no programa pode ser realizada.

A proposta da plataforma REDLART é o desenvolvimento de um laboratório remoto utilizando uma plataforma em *hardware* reconfigurável (FPGA) que possibilite a interação entre os usuários e a plataforma em tempo real através do controle do experimento e da monitoração de fluxos de sinais multimídia. O foco dessa proposta é fornecer uma plataforma modular e que possua interface de usuário adequada para facilitar a comunicação homem-máquina. O objetivo é proporcionar um ambiente colaborativo que seja genérico e flexível para que o experimento possa ser executado remotamente e no local do laboratório.

A plataforma REDLART pode ser utilizada como uma ferramenta adicional para auxiliar no processo de ensino-aprendizagem de práticas de laboratório de engenharia. A intenção não é substituir as aulas práticas de laboratório convencionais, mas fornecer uma ferramenta que possibilite estender o conceito de laboratório real para que as experiências possam ser realizadas a distância.

Cursos da área de ciências exatas possuem a grade curricular fundamentada em ferramentas matemáticas tradicionais que são utilizadas para consolidar a formação acadêmica do aluno. Entre essas ferramentas matemáticas podem ser mencionadas, por exemplo, as transformadas de Laplace e de Fourier. Nos últimos anos, com o advento das tecnologias digitais, é imprescindível o uso de tais ferramentas matemáticas em algoritmos e sistemas com aplicações diversas. O ensino dessas ferramentas no contexto das tecnologias digitais é geralmente agrupado sob o nome de Processamento Digital de Sinais (PDS). Atualmente, PDS é um campo imenso e diversificado. Há milhares de profissionais que consideram processamento de sinais como sua área de especialidade e outras centenas de milhares cujo trabalho envolve essa área [28].

As mudanças curriculares nos cursos de ciências exatas, principalmente nos cursos da área de engenharia, que vêm ocorrendo nos últimos anos devido à introdução de PDS, trazem uma série de desafios. A existência de poucas obras em língua portuguesa e a ênfase dada a ferramentas matemáticas de tempo contínuo podem trazer dificuldades [29]. Esses problemas podem ser refletidos como falta de motivação por parte dos alunos.

De forma genérica, entende-se que a motivação é aquilo que move uma pessoa ou que a põe em ação ou a faz mudar o curso. Em sala de aula, os efeitos imediatos da motivação do aluno consistem em envolvê-lo ativamente nas tarefas pertinentes do processo de aprendizagem. Tal envolvimento consiste na aplicação de esforço no processo de aprender, com a persistência exigida por cada tarefa [30].

Uma forma eficiente de capturar a atenção do aluno e motivá-lo é demonstrando que o conhecimento adquirido nas aulas teóricas é efetivamente empregado em uma série de tecnologias presentes em seu dia a dia. Isso pode ser conseguido de forma prática através de aulas laboratoriais, uso de recursos computacionais e através de ferramentas de ensino a distância. É nesse cenário que a plataforma REDLART pode ser introduzida para auxiliar no processo de ensino de laboratório a distância.

Nas práticas de laboratório de sistemas digitais envolvem-se problemas reais referentes a processamento digital de sinais. Um conceito bastante difundido em PDS é a teoria dos filtros digitais. Esse conceito pode ser aplicado em vários problemas reais. Por exemplo, após uma aula sobre filtros digitais, suponha que a prática de laboratório envolva o problema descrito a seguir. O controle remoto por rádio-freqüência é comum em portas de garagens e alarmes de carros. O princípio de funcionamento consiste em transmitir ondas de rádio que correspondam a um comando binário referente ao botão que o usuário pressionou. Com base no conhecimento adquirido sobre filtros digitais pode ser requerido que se projete e teste um sistema para detecção do sinal de controle para acionamento do dispositivo alvo.

A plataforma REDLART pode ser utilizada para aplicar na prática os conceitos adquiridos sobre PDS, como no exemplo de filtros digitais citado acima. A plataforma é flexível o suficiente para permitir que novos experimentos possam ser incorporados a ela, abstraindo-se os detalhes de implementação que não são o foco do problema. Para a resolução de um problema como o citado acima, podem-se desenvolver experimentos que utilizem, por exemplo, filtros digitais diversos.

Após a integração do experimento na plataforma, os alunos podem ter acesso a dados reais, executados em tempo real na plataforma de *hardware* REDLART para aferir medidas, analisar os resultados e executar o experimento sem restrição de tempo e espaço físico. Esse é apenas um cenário no qual a plataforma REDLART pode ser utilizada. No Capítulo 6 é descrito outro cenário. Além disso, cenários diferentes poderão ser explorados posteriormente.

Capítulo 3

Programação de Dispositivos FPGA

Há diversos tipos de circuitos integrados digitais. No entanto, o foco deste trabalho encontra-se nos dispositivos lógicos programáveis, especificamente nas FPGAs. Por esse motivo, este capítulo apresenta uma visão geral dos *chips* FPGAs assim como as formas de desenvolvimento e implementação dos mesmos.

3.1 Introdução a FPGA

Antes do advento das FPGAs as tecnologias disponíveis eram os Transistores, Circuitos Integrados (*Integrated Circuits* - IC), SRAMs (*Static Random Access Memory*) e DRAMs (*Dynamic Random Access Memory*), Microprocessadores, SPLDs (*Simple Programmable Logic Device*), CPLDs (*Complex Programmable Logic Device*) e ASICs (*Application Specific Integrated Circuits*). Para entender os motivos que levaram ao surgimento das FPGAs é importante considerá-las no contexto dessas tecnologias relacionadas. A Figura 3.1 mostra as datas aproximadas da cronologia das tecnologias relacionadas com FPGA [31]. A parte em cinza indica que embora a tecnologia estivesse disponível, no primeiro momento, não foi recebida com entusiasmo por parte dos profissionais que trabalhavam nesse setor [31]. Por exemplo, embora a primeira FPGA tenha surgido no início de 1984 os projetistas de *hardware* e os desenvolvedores não se motivaram a utilizar essa tecnologia de fato até o início de 1990.

Em 1947 pesquisadores da Bell Lab's conseguiram desenvolver o primeiro dispositivo transistor [32]. O transistor é um dispositivo semicondutor que controla ou amplifica sinais elétricos. Este dispositivo possui função semelhante às válvulas, de controlar o fluxo de corrente, mas com a vantagem de ter o tamanho reduzido em relação às mesmas. Os transistores não geram atrasos por aquecimento e podem funcionar como comutadores (chaves). Essa invenção representou um salto tecnológico na história. Isso substituiu definitivamente as válvulas e tornou possível reduzir consideravelmente o tamanho dos produtos eletrônicos [32].

Após a invenção dos transistores, as implementações de circuitos digitais tornavam-se cada vez mais complexas. O número de transistores empregados nos circuitos digitais aumentava rapidamente. Portanto, surgiu a necessidade de reduzir a eletrônica envolvida e desta forma surgiu o circuito in-

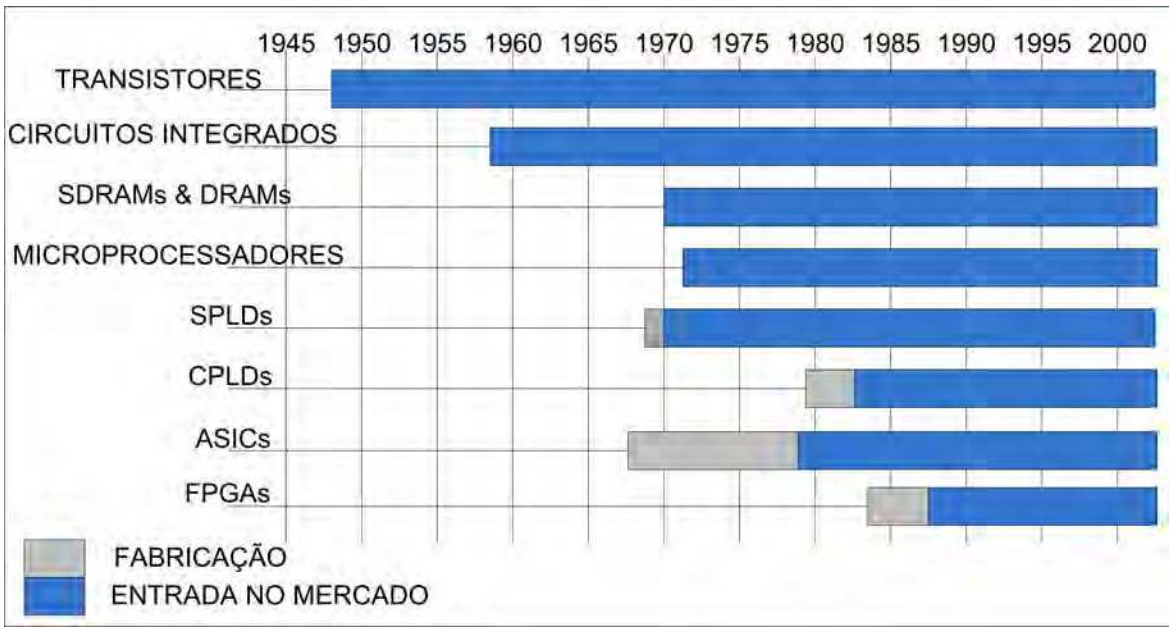

Fig. 3.1: Cronologia das tecnologias relacionadas com FPGA.

tegrado (CI). A cada novo CI lançado a complexidade dos projetos aumentava exponencialmente, possibilitando o desenvolvimento de aplicações que até então eram difíceis de serem implementadas [32].

No final de 1960 e início de 1970 surgiram novos desenvolvimentos na área de circuitos integrados digitais o que impulsionou o processo de expansão dos mesmos. Em 1970 foi anunciada a primeira DRAM. A memória RAM dinâmica é formada por pares de transistores e capacitores que consomem pouco espaço no silício. O atributo dinâmico é utilizado porque o capacitor perde a informação ao longo do tempo, de modo que cada célula da DRAM deve ser periodicamente atualizada para manter a informação armazenada. A memória SRAM difere da memória DRAM porque uma vez que a informação foi carregada para dentro da célula SRAM os dados permanecerão inalterados, a menos que seja especificamente alterado ou a energia elétrica seja removida do sistema.

Em 1971 surgiu o projeto de um microprocessador que incorporava as funções de uma unidade de processamento central em um único circuito integrado. Os microprocessadores substituíram milhões de transistores o que diminuiu o tamanho dos dispositivos. Eles são utilizados principalmente para processamentos complexos e dentre algumas aplicações estão presentes nos computadores e aparelhos eletrônicos. As tecnologias SRAM e de microprocessadores são largamente utilizadas na maioria das FPGAs.

Os primeiros circuitos integrados programáveis foram genericamente chamados de dispositivos lógicos programáveis (*Programmable Logic Devices* - PLDs). O PLD é um circuito integrado cuja arquitetura interna é determinada pelo fabricante, mas que é projetado de tal forma que pode ser configurado ou programado em campo, no local de trabalho do desenvolvedor, para executar diversas

funções. O termo PLD pode ser referenciado em duas subcategorias: os SPLDs (*Simple Programmable Logic Devices*) e os CPLDs (*Complex Programmable Logic Devices*). Os SPLDs referem-se aos primeiros PLDs fabricados, os quais originalmente continham um número modesto de unidades lógicas equivalentes e eram bastante simples. Os CPLDs são dispositivos que contêm uma determinada quantidade de funções SPLD que compartilham uma matriz de interconexões programáveis. Se comparado às FPGAs, esses dispositivos contêm um número relativamente limitado de unidades lógicas que podem ser utilizadas para implementar funções menores e menos complexas.

Por volta do início de 1980, houve uma lacuna no processo de expansão dos circuitos integrados. De um lado havia dispositivos programáveis como os PLDs que são criados de maneira que podem ser programados para executar diversas funções. Os dispositivos PLDs eram altamente configuráveis e demandavam pouco tempo desde a fase de planejamento até a fase de implementação. No entanto, os PLDs não forneciam suporte a funcionalidades complexas. Do outro lado estavam disponíveis os ASICs. Os ASICs são circuitos integrados personalizado projetados para uma aplicação específica. Esses dispositivos podem conter centenas de milhares de unidades lógicas e podem ser utilizados para a construção de funções aritméticas de funcionalidades extremamente complexas. Em geral, o ASIC é concebido e construído para fim empresarial. Além disso, uma vez que o projeto foi implementado em um ASIC ele é permanentemente gravado no silício. Portanto, as opções disponíveis para a construção de circuitos integrados eram ou a utilização de PLDs, altamente configuráveis, mas sem suporte a funcionalidades complexas ou a utilização de ASICs para aplicações específicas.

Para preencher essa lacuna no processo de expansão dos circuitos integrados, em meados de 1985, uma companhia chamada Xilinx introduziu a ideia de combinar o controle de usuário e o *time-to-market*[1] do PLD com a densidade e os benefícios de arranjo de unidades lógicas. Isso originou uma nova classe de circuito integrado chamado de FPGA. As FPGAs ocupam o meio-termo entre PLDs e ASICs, porque suas funcionalidades podem ser personalizadas na "programação em campo"como acontece com os PLDs, além disso, podem conter milhares de unidades lógicas utilizadas para a execução de funcionalidades extremamente grandes e complexas que anteriormente só poderiam ser realizadas utilizando ASIC. O custo de um projeto em FPGA é menor do que o de um projeto em ASIC (apesar de componentes ASICs serem mais barato quando utilizados na produção de grande escala). Ao mesmo tempo, implementar mudanças no projeto é mais fácil em FPGA e o tempo necessário para a especificação, desenvolvimento e implementação até a finalização do projeto é menor.

As FPGAs são circuitos integrados digitais que contêm blocos configuráveis de lógica com interconexões programáveis entre eles. Os projetistas podem configurar esses dispositivos de maneira que executem diferentes tarefas. A parte do nome FPGA referenciada como "*Field Programmable*"menciona o fato de que sua programação acontece "em campo", ou seja, no local do usuário, diferentemente de dispositivos cuja programação é feita apenas pelo fabricante. Isso significa que as FPGAs podem tanto ser configuradas no laboratório como podem ser modificadas depois de já estarem em seus locais definitivos de operação [31]. O termo *gate array* está relacionado ao conceito de arranjo de unidade lógica. As unidades lógicas são componentes padrão formados por transistores que podem ser configurados independentemente e interconectados a partir de uma matriz de trilhas

[1] é o tempo de projeto e concepção de um produto até a disposição desse produto no mercado.

condutoras e *switches* programáveis [33].

Durante o início da década de 1990, o tamanho e a sofisticação das FPGAs começaram a aumentar. O grande mercado eram os consumidores das áreas de telecomunicação e redes, as quais envolviam o processamento de grandes blocos de dados. Mais tarde, em direção ao final da década de 1990, a utilização de FPGA em indústria automotiva e em aplicações industriais passou a ter um grande crescimento. No início do ano 2000, já se encontrava disponível FPGA de alto desempenho contendo milhares de unidades lógicas. Algumas das características desses dispositivos são a possibilidade de incorporar os núcleos de microprocessadores embarcados, além dos pinos de alta velocidade para serem usados como entradas e saídas [34] [31]. O resultado é que atualmente as FPGAs podem ser utilizadas para implementar praticamente qualquer circuito digital, incluindo *software* e dispositivos de comunicação como rádio, radar, processamento de imagem e outras aplicações de processamento digital de sinais.

O tipo de FPGA dominante tem base na tecnologia SRAM (*Static Random Access Memory*) e pode ser reprogramada quantas vezes for necessário [33]. Isso significa que, se as células lógicas SRAM forem programadas de maneira apropriada, cada bloco lógico do dispositivo pode ser configurado para executar uma função diferente. Uma FPGA que utiliza células SRAM é reprogramada toda vez que é ligada, pois o dispositivo FPGA é composto de uma memória volátil. Por essa razão, é necessário ter uma memória PROM (*Programmable Read-Only Memory*) em cada FPGA que utiliza a tecnologia SRAM.

Os primeiros dispositivos FPGA empregaram o conceito de bloco lógico programável. Os blocos lógicos programáveis podem ser vistos como componentes padrão que podem ser configurados independentemente. Atualmente, o bloco lógico programável é composto de vários recursos que podem implementar diversas funções aritméticas e lógicas. Esses recursos variam de acordo com o fabricante da FPGA, mas de maneira geral incluem LUTs (*Lookup Table*), registradores que podem atuar como um *flip-flops* ou como um *latch*, memórias, multiplicadores e acumuladores programáveis, gerenciamento de *clock* digital e lógica de transporte (*carry*) dedicado. Cada FPGA é formada por um grande número de blocos lógicos programáveis interconectados através de uma matriz de trilhas condutoras e *switches* programáveis. Para especificar a funcionalidade de cada bloco lógico programável, assim como seletivamente fechar os *switches* da matriz de interconexão, é necessário gerar um arquivo binário que indique as configurações da FPGA. Esse arquivo binário é gerado a partir de ferramentas de *software* seguindo um determinado fluxo de projeto. A hierarquia de *switches* programáveis em conjunto com os blocos lógicos programáveis permite que o sistema possa ser interconectado de acordo com a necessidade do projetista. A Figura 3.2 mostra a arquitetura geral de roteamento de uma FPGA.

Os blocos lógicos programáveis são conectados à matriz de trilhas condutoras através de circuitos responsáveis pelo interfaceamento das entradas e saídas. Estes circuitos são *buffers* que funcionam como pinos de entrada e saída bidirecionais de dados e de sinais. As trilhas são formadas por sequência de um ou mais segmentos de ligação em série. Os blocos de conexão permitem a conectividade das entradas e saídas de um bloco lógico com os segmentos de trilhas nos canais. Quando um canal é formado por um grupo de duas ou mais trilhas paralelas é chamado de canal de roteamento. Os *swit-*

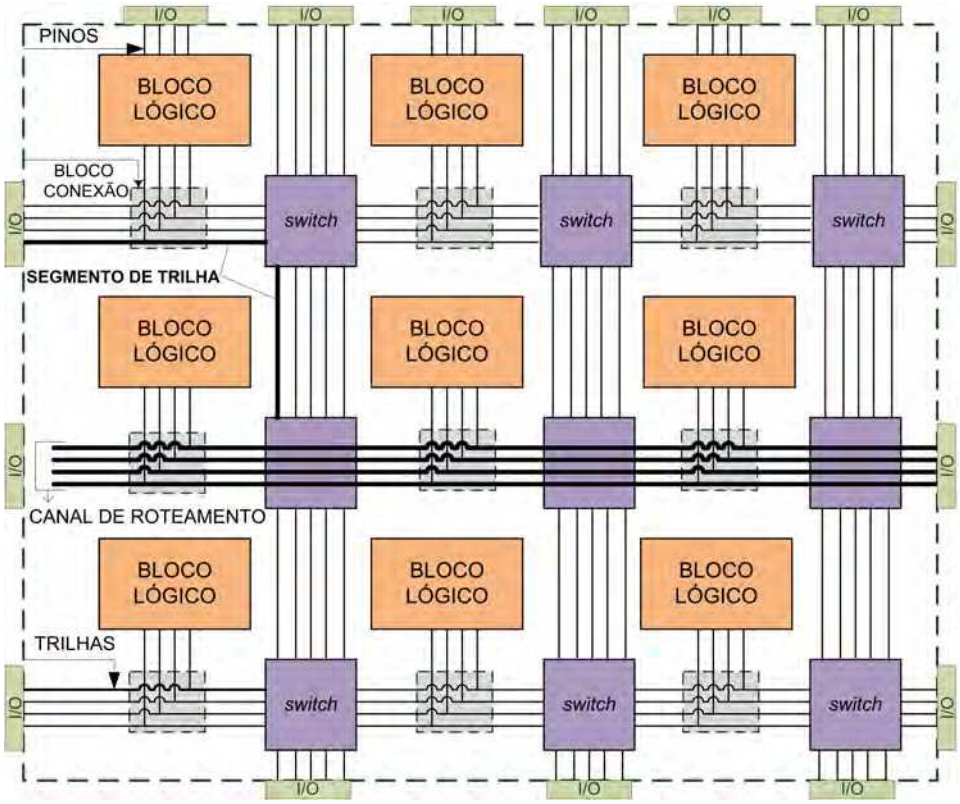

Fig. 3.2: Visão geral da arquitetura de roteamento de uma FPGA.

ches programáveis são organizados como trilhas de roteamento horizontal e vertical entre as linhas e colunas dos blocos lógicos programáveis. Em algumas arquiteturas tem-se que o bloco de conexão e os *switches* são encapsulados em uma mesma estrutura e em outras arquiteturas eles aparecem separados. Para que a interconexão aconteça de forma mais rápida, podem ser criados caminhos de interconexões globais os quais permitem transportar sinais através da FPGA, sem a necessidade de passar por múltiplos elementos de roteamento. Portanto, o arranjo de blocos lógicos programáveis e a matriz de interconexão formam a estrutura básica da FPGA para especificação de circuitos integrados complexos.

3.2 Formas de Desenvolvimento e Implementação

Há diversas maneiras de desenvolvimento e implementação de projeto de circuitos para FPGA. Os métodos tradicionais de fazer a programação em uma FPGA incluem o uso de linguagem esquemática ou linguagem de descrição de *hardware*.

3.2.1 Linguagem Esquemática

Antigamente, os circuitos eletrônicos eram projetados a mão. Os diagramas do circuito, também referenciado como diagrama esquemático ou esquemático, eram desenhados a mão utilizando caneta

e papel. Esses diagramas continham a representação gráfica de símbolos que indicavam as portas lógicas e as funções que deveriam ser utilizadas para implementar o projeto, juntamente com as conexões entre eles. É sabido que esse modo artesanal de desenvolver o circuito era demorado e propenso a erro. A Figura 3.3 a seguir mostra um simples diagrama esquemático desenvolvido nesse modelo.

Os diagramas esquemáticos são a primeira etapa do desenvolvimento de circuitos eletrônicos. Neles os componentes são desenhados levando apenas em consideração seus pinos, independente de seu formato físico. Esse diagrama demonstra as ligações físicas do circuito, independentemente da posição dos componentes e das trilhas [32].

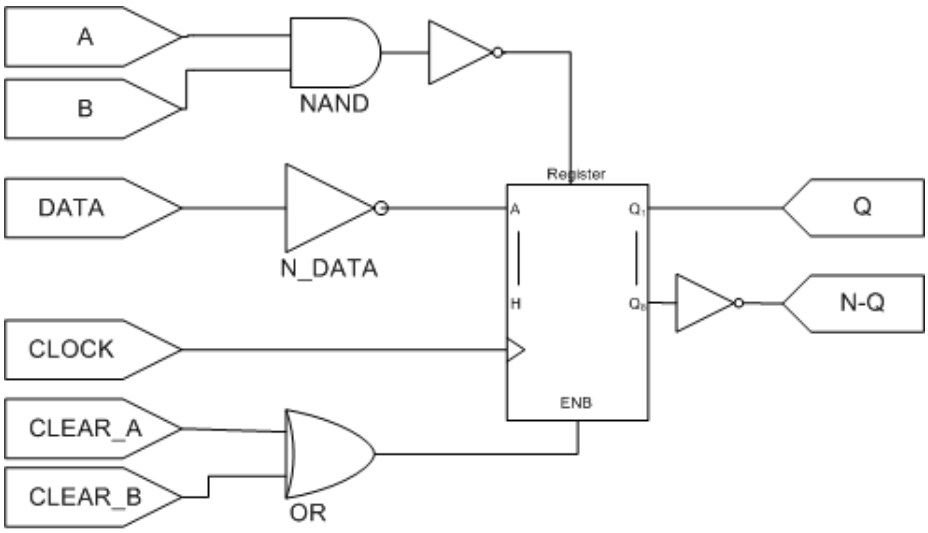

Fig. 3.3: Diagrama esquemático.

No final dos anos 70 e início dos anos 80, as empresas fabricantes de *hardware* começaram a fornecer programas gráficos para a captura do esquemático. Esses programas gráficos permitiam aos projetistas e aos desenvolvedores criar interativamente diagramas de circuitos (esquemáticos). Os desenvolvedores podiam selecionar a representação de símbolos como pinos de entrada/saída, unidades lógicas e funções de uma biblioteca de símbolos especiais, além de desenhar linhas que representavam as conexões entre os símbolos. Uma vez que o projeto do circuito lógico estava representado no programa gráfico, o pacote de ferramentas para capturar o esquemático poderia gerar uma descrição textual correspondente dos níveis de unidades lógicas.

A captura do esquemático é um método tradicional que os desenvolvedores utilizam para especificar os arranjos de unidades lógicas e os dispositivos lógicos programáveis. Consiste de uma ferramenta gráfica que permite explicitar as unidades lógicas necessárias e como essas unidades lógicas devem ser conectadas. Algumas etapas são necessárias para fazer a captura do esquemático, conforme mostra a Figura 3.4. Na primeira fase seleciona-se uma ferramenta específica para fazer a captura do esquemático e selecionam-se as bibliotecas de acordo com o modelo do dispositivo e do fabricante. Inicia-se o processo de construção do circuito carregando as unidades lógicas que serão utilizadas da biblioteca selecionada. O desenvolvedor tem a liberdade de utilizar diversas combina-

ções de unidades lógicas conforme forem necessárias. Na segunda fase são conectadas as unidades lógicas através de fios. O desenvolvedor tem o controle completo e pode conectar as unidades lógicas em qualquer configuração exigida pelo aplicativo. A terceira fase consiste em adicionar e rotular *buffers* ou pinos de entrada e saída. Isso irá definir o pacote de pinos de entrada e saída para o dispositivo. Na última fase é gerado o *netlist*.

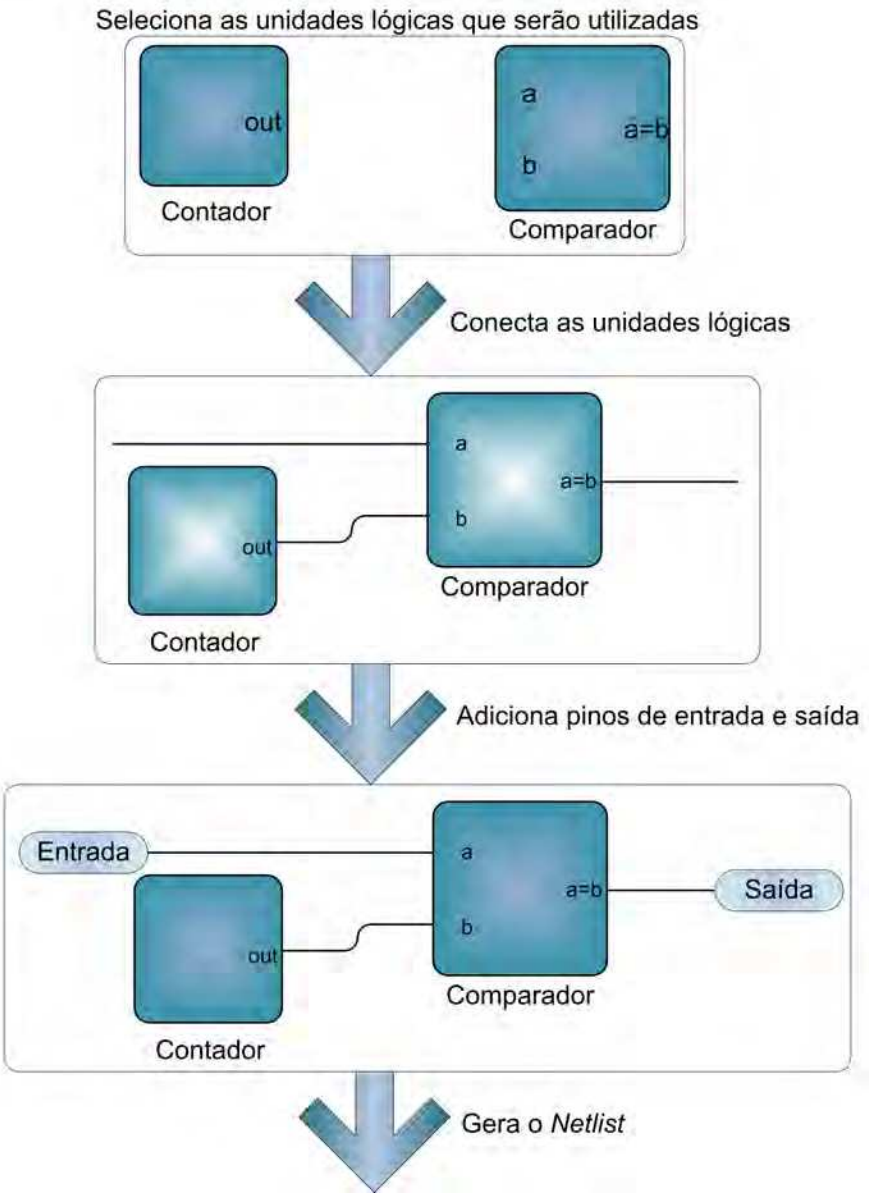

Fig. 3.4: Processo de captura do esquemático.

O *netlist* é uma descrição textual do circuito. Ele é gerado pelas ferramentas de desenvolvimento de projeto, como por exemplo, um programa de captura do esquemático. O *netlist* é uma maneira compacta de descrever o circuito para que outros programas possam entender quais unidades lógicas

estão no circuito, como elas estão conectadas, e quais são os nomes dos pinos de entrada e saída. Na Figura 3.5, o *netlist* reflete a sintaxe atual do projeto do circuito no esquemático. As linhas descrevem cada um dos componentes e cada uma das ligações. Na geração do *netlist* são atribuídos nomes aos componentes e às ligações. Além disso, quando esse projeto for implementado, terá um pacote de pinos de entrada e pinos de saída.

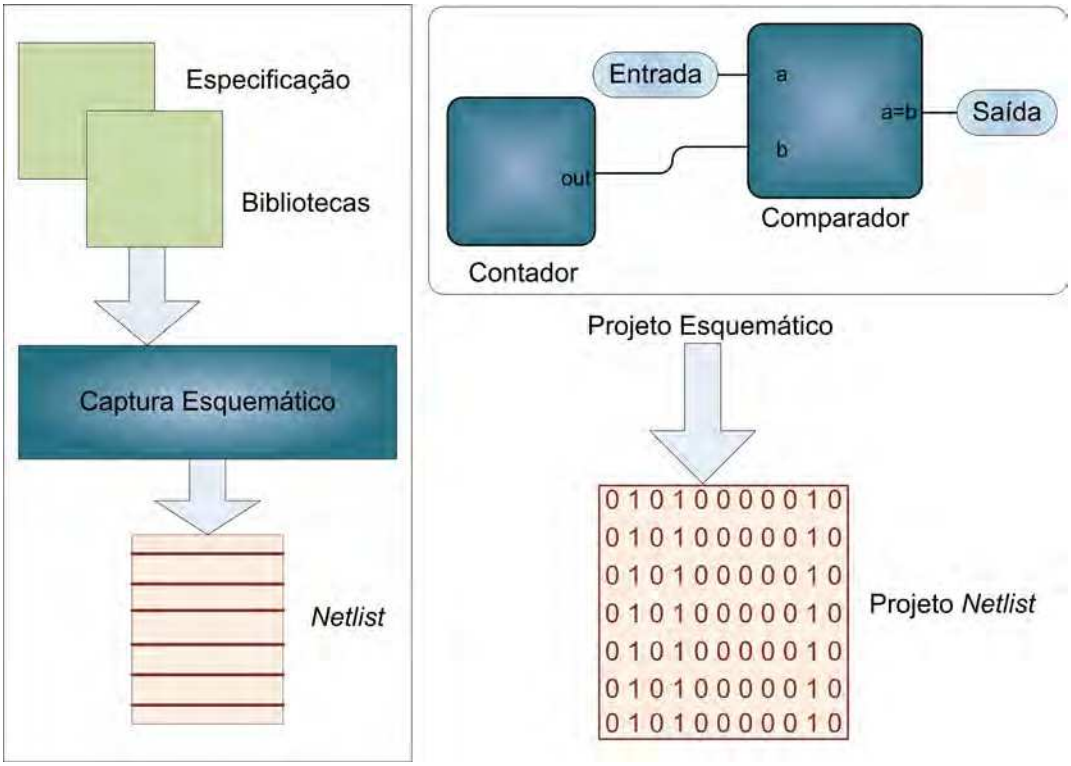

Fig. 3.5: Especificação do Projeto - *Netlist*.

Considere o exemplo de um projeto que utiliza 2000 unidades lógicas. Uma típica página de esquemático inclui aproximadamente 200 unidades lógicas [31]. Portanto, seriam necessários 10 páginas de esquemático para criar o projeto que utiliza 2000 unidades lógicas. Para criar cada página é preciso passar por todas as quatro fases mencionadas anteriormente que são iniciar o processo de projetar o circuito escolhendo os componentes que serão utilizados, conectar os componentes, acrescentar os pinos de entrada e saída e, por último, gerar o *netlist*. O tempo necessário para gerar o *netlist* varia de acordo com o número de componentes utilizados no projeto.

Um problema inerente à utilização do método de captura do esquemático é a dificuldade na migração entre os modelos de FPGA e os fabricantes. Se o projeto foi desenvolvido utilizando 2000 unidades lógicas de um fabricante x de FPGA e o desenvolvedor deseja migrar o projeto para utilizar a FPGA do fabricante y, seria necessário modificar todas as páginas do esquemático gerado para utilizar as bibliotecas dos componentes do fabricante y.

3.2.2 Linguagem de Descrição de Hardware

Nos últimos anos, os projetos desenvolvidos em *hardware* reconfigurável cresceram no tamanho e na complexidade, consequentemente, era difícil entender e manter um projeto que utilizava mais de 5000 unidades lógicas o que representava algumas dezenas ou centenas de páginas de esquemático. Além disso, o processo de fazer a captura do esquemático de um grande projeto no nível de abstração de unidades lógicas era passível de erros e também consumia muito tempo para a geração do esquemático. Por isso, os vendedores de *software* de automação de projetos eletrônicos (*Eletronic Design Automation* - EDA) iniciaram o desenvolvimento de ferramentas de projeto e o estudo de fluxo de desenvolvimento com base no uso de linguagem de descrição de *hardware* (*Hardware Description Languages* - HDL).

O objetivo é utilizar uma linguagem de alto nível para descrever o circuito integrado em um arquivo de texto em vez de uma descrição gráfica no nível de unidades lógicas. A linguagem de descrição de *hardware* é utilizada para descrever uma representação funcional e comportamental do circuito através de notação adequada e padronizada que independe do fabricante de *hardware*.

Por exemplo, considere um simples projeto que precisa especificar um multiplicador onde dadas 2 entradas de 16 bits cada, produzam-se uma saída de 32 bits. Para efeito de comparação serão utilizados os dois métodos apresentados até o momento, que são a captura do esquemático e o arquivo HDL. Um multiplicador é um componente regular formado por somadores e registradores que necessita de várias unidades lógicas para ser implementado. O multiplicador possui duas entradas de 16 bits (A e B) e produz um resultado de 32 bits, conforme mostra a Figura 3.6.

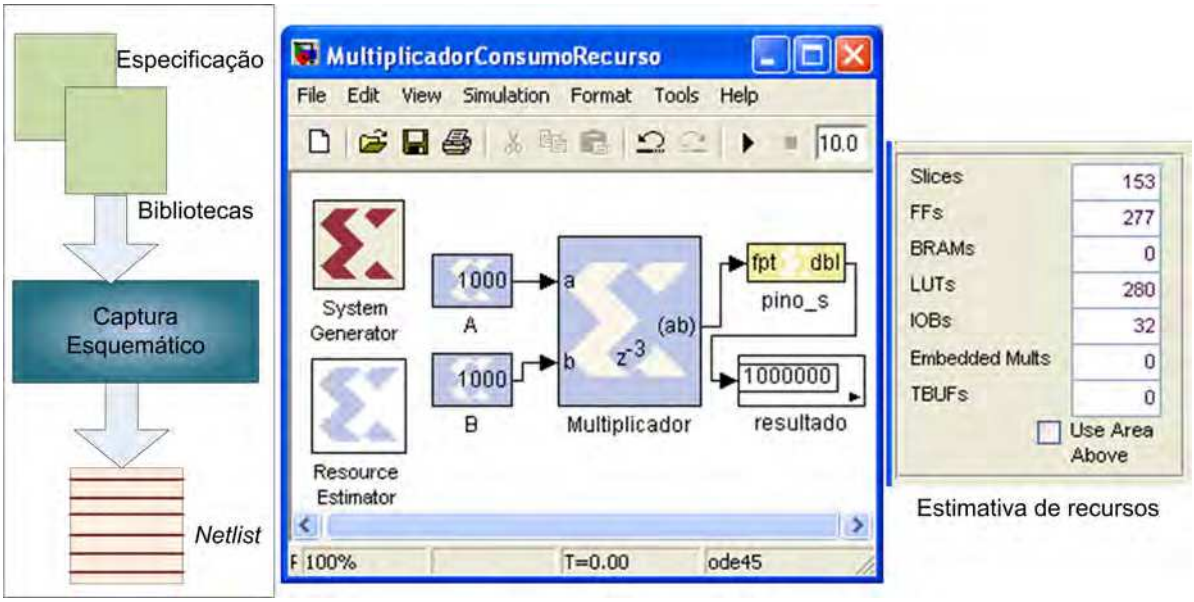

Fig. 3.6: Projeto esquemático.

A Tabela estimativa de recursos, apresentada na Figura 3.6, indica a quantidade de recursos do

chip FPGA que são necessários para implementar o circuito do projeto multiplicador. Esse circuito necessita de aproximadamente 153 unidades lógicas, 280 *lookup tables*, 277 *flip-flops* e 32 *buffers* de entrada e saída. Para realizar a captura do esquemático as unidades lógicas que serão utilizadas devem ser carregadas, posicionadas na página e conectadas com os *buffers* de entrada e saída. Na implementação HDL será necessário a mesma quantidade de recursos utilizada pelo método de captura do esquemático, no entanto, o desenvolvimento em HDL pode ser feito de maneira mais simples e mais rápida se comparado com o método anterior. A Figura 3.7 mostra a implementação em HDL do projeto multiplicador de 32 bits.

```
Entity MULTIPLICADOR Is
Port (A,B: In std_logic(15 downto 0);
resultado: out std_logic(31 downto 0));
end MULTIPLICADOR;

Architecture BEHAVE of MULTIPLICADOR Is
Begin
resultado<= A * B;
end BEHAVE;
```

Fig. 3.7: Projeto multiplicador escrito em HDL.

Na implementação HDL, caso o projetista do circuito deseje alterar a especificação do multiplicador para que a saída produzida seja um valor de 16 bits, é necessário apenas alterar a definição da variável resultado de std_out(31 downto 0) para std_out(15 downto 0) considerando os *bits* mais significativos. Isso provavelmente exigiria do projetista menos tempo de trabalho do que se utilizasse o método de captura do esquemático, uma vez que essa modificação no método de captura do esquemático exigiria uma nova geração do mesmo.

3.2.3 Níveis de abstração HDL

A linguagem de descrição de *hardware* é bastante versátil, por isso é importante entender de que maneira essa linguagem pode ser utilizada como parte do fluxo de projeto de um circuito digital. As funcionalidades de um circuito digital podem ser representadas em três diferentes níveis de abstração e o HDL pode fornecer suporte a esses níveis de abstração, conforme mostra a Figura 3.8.

O nível mais baixo de abstração de um HDL é o estrutural, que consiste em uma representação do circuito semelhante a um *netlist* de portas lógicas ou de *switches* [35]. Esse nível de abstração é formado por dois subníveis: *switches* e unidades lógicas, respectivamente. O nível de *switch* remete a capacidade de descrever um circuito como um *netlist* de *switches* transistores. O nível imediatamente superior é o nível de unidades lógicas que indica a capacidade de descrever o circuito como um *netlist* de funções e portas lógicas primitivas. No entanto, deve-se observar que a palavra "estrutural"pode ter uma conotação diferente, pois também pode ser utilizada para se referir a um *netlist* de nível de bloco hierárquico, em que cada bloco pode ter seu conteúdo especificado utilizando qualquer um dos

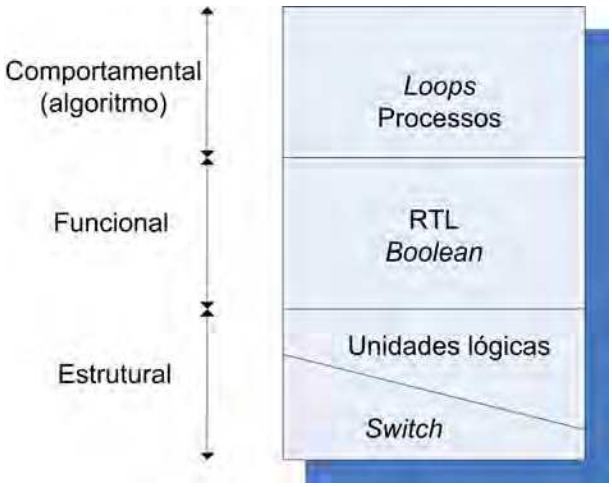

Fig. 3.8: Diferentes níveis de abstrações.

níveis de abstração.

O próximo nível de abstração de HDL é o funcional, que indica a capacidade de descrever o funcionamento do circuito em termos de lógica combinacional e booleana. A lógica combinacional indica construções ou estruturas cujo resultado depende exclusivamente da variável de entrada, por exemplo, é o caso das estruturas *if*, *then* e *else*. O nível funcional de abstração também incorpora a representação de nível de registro de transferência (*Register Transfer Level* - RTL). O nível RTL é responsável por indicar o funcionamento do circuito através de registradores interligados por lógica combinacional.

O nível de abstração mais alto do HDL tradicional é conhecido como comportamental. Esse nível refere-se a maneira de descrever o comportamento do circuito utilizando estruturas como *loops* e processos. O nível de abstração comportamental permite que seja possível utilizar a infraestrutura de algoritmo para compor equações através de multiplicadores e somas.

Os conceitos apresentados até o momento são conceitos genéricos e comuns a todos os HDLs, uma vez que existem diferentes HDLs padronizados como VHDL e Verilog além dos HDLs proprietários. O VHDL e o Verilog serão abordados adiante.

3.2.4 Verilog

Em meados dos anos de 1980, um novo HDL chamado de Verilog foi desenvolvido [31]. A definição formal de Verilog descreve a sintaxe e a semântica dessa linguagem. Nesse contexto, o termo sintaxe refere-se à gramática da linguagem, por exemplo, a ordem das palavras e os símbolos relacionados. A análise sintática faz o reconhecimento das declarações, definições, blocos de comandos e da estrutura do programa como um todo. A análise semântica refere-se ao significado das palavras, dos símbolos e acepção dos relacionamentos entre eles. A análise semântica verifica os relacionamentos entre os fragmentos do código do programa. O Verilog foi projetado para ser simples, intuitivo

e eficaz em vários níveis de abstração em um formato padrão de texto para ser utilizado por várias ferramentas de projeto de circuitos integrados.

O Verilog original mostrou-se consideravelmente forte quanto ao nível de abstração estrutural, especialmente no que diz respeito à capacidade de modelagem de atraso, nível de abstração funcional (equações booleanas e RTL), além de suportar algumas construções ao nível de comportamento [31]. Portanto, o Verilog se tornou popular. Contudo, as empresas começaram a manifestar interesse em desenvolver o seu próprio HDL e isso ocasionou um problema porque começaram a estender a linguagem em diferentes direções.

A popularidade do Verilog continuou crescendo e o IEEE formou uma comissão de trabalho para estabelecer o padrão Verilog. Em meados de 1995 o Verilog foi oficialmente estabelecido como padrão IEEE 1364-1995. Desde então, algumas modificações e correções foram realizadas na especificação da linguagem. A versão mais atual é o padrão IEEE 1364-2005 [36].

3.2.5 VHDL

Em 1980, o departamento de defesa dos Estados Unidos lançou o programa de circuito integrado de alta velocidade (VHSIC - *Very High Speed Integrated Circuit*) cujo objetivo era incentivar o avanço de pesquisa em tecnologia digital. Durante o desenvolvimento desse projeto percebeu-se que os componentes empregados utilizavam diferentes linguagens de simulação e ferramentas de projeto que eram incompatíveis. Em 1981, a fim de resolver essa questão, foi lançado um projeto para desenvolver uma nova linguagem de descrição de *hardware* chamada VHSIC HDL ou VHDL (*Very High Speed Integrated Circuit Hardware Description Language*). Esse projeto contou com o apoio de diversas empresas e em 1987 a linguagem VHDL tornou-se padrão IEEE 1076 [37].

A principal motivação para a utilização da linguagem VHDL (ou Verilog) é que VHDL é uma linguagem padrão independente do fornecedor de *hardware* ou da tecnologia empregada, portanto é portátil e reutilizável. Uma vez que o código VHDL foi escrito, ele pode ser utilizado para implementar um circuito em um dispositivo programável ou pode ser utilizado no processo de fabricação de um *chip*. Atualmente, muitos *chips* comerciais são projetados utilizando essa abordagem [38].

A linguagem VHDL é muito forte nos níveis de abstrações funcional e comportamental e também fornece suporte a construções mais complexas. No entanto, o VHDL é um pouco fraco quando se trata do nível estrutural de abstração, especialmente no que diz respeito à capacidade de modelagem de atraso. Contudo, quando comparado com o Verilog, o VHDL possui a vantagem de facilitar a programação em *hardware* porque fornece ênfase ao aspecto comportamental da linguagem HDL.

3.2.6 Vantagens de Utilizar a Linguagem de Descrição de Hardware

A descrição de um sistema em linguagem de descrição de *hardware* apresenta inúmeras vantagens. Por exemplo, facilita o intercâmbio de projetos entre grupos de pesquisa. Isto favorece que os projetos possam ser modificados sem dificuldade porque a linguagem HDL independe da tecnologia atual de circuitos integrados. Além disso, permite ao projetista considerar no seu projeto os atrasos comuns

aos circuitos digitais, o que deixa a implementação do projeto menos suscetível a erros. A utilização da linguagem HDL pode reduzir consideravelmente o tempo de projeto e implementação.

3.3 Fluxo de Projeto para FPGA

A crescente complexidade do projeto em FPGA, assim como o aumento da sofisticação das ferramentas de projeto, propiciaram um aumento na demanda para modelar um projeto com níveis de abstração mais elevados. A mudança em desenvolver um projeto com base na captura do esquemático para o desenvolvimento com base no HDL trouxe melhorias significativas. Essa mudança permitiu aos desenvolvedores implementar os módulos em um padrão comportamental que era independente da tecnologia. No entanto, havia muitos aspectos no projeto digital com o uso do HDL que poderia ser melhorado. A fim de aperfeiçoar o desenvolvimento de projeto em circuitos integrados houve um esforço por parte da comunidade acadêmica e da indústria para definir um fluxo principal de projeto. O fluxo tradicional de projeto de circuitos para FPGA pode ser dividido em quatro fases distintas: especificação, verificação, implementação e depuração de sistema, conforme mostra a Figura 3.9 [33]. É importante salientar que pode haver variação na especificação de cada fase do fluxo de projeto de acordo com o fabricante da FPGA.

Antes de descrever as fases do fluxo de projeto é necessário destacar que o conjunto de ferramentas de *software* associado a um fluxo de projeto provê ao desenvolvedor um nível de abstração que o permite focar no algoritmo a ser desenvolvido, ao invés de se preocupar com os circuitos que serão implementados. Dessa forma, a programação do dispositivo pode ser feita através de uma linguagem de programação (VHDL) ou mesmo através de modelagem de sistemas [35]. As empresas fabricantes de FPGA geralmente disponibilizam ferramentas para fazer a modelagem do sistema. A Xilinx, por exemplo, oferece uma ferramenta chamada *System Generator*. A modelagem do sistema utilizando essa ferramenta será vista em detalhes no Capítulo 5.

3.3.1 Especificação

Na fase de especificação ou geração do *netlist* é obtida uma descrição compacta do circuito. Conforme visto anteriormente, o *netlist* fornece uma descrição compacta dos componentes do circuito e de como estes componentes estão interconectados incluindo os pinos de entrada e saída. A descrição do circuito realizada pelo *netlist* é dependente do fabricante e do modelo do dispositivo porque os componentes utilizados na descrição são provenientes de bibliotecas específicas deste fabricante.

A geração do *netlist* pode ser feita através do método de captura do esquemático ou da síntese de código HDL. O esquemático é uma representação gráfica que permite explicitar as unidades lógicas necessárias e como estas unidades lógicas devem ser conectadas. O esquemático facilita o desenvolvimento de projetos de circuitos por que não é necessário trabalhar diretamente com a descrição textual. No entanto, não provê portabilidade porque o projeto é concebido para uma família de dispositivos de um determinado fabricante. Caso seja necessário migrar para outro fabricante, será necessário desenvolver o projeto utilizando as bibliotecas deste fabricante. O código HDL apresenta uma especificação funcional e comportamental do circuito independentemente do modelo do dispositivo ou do

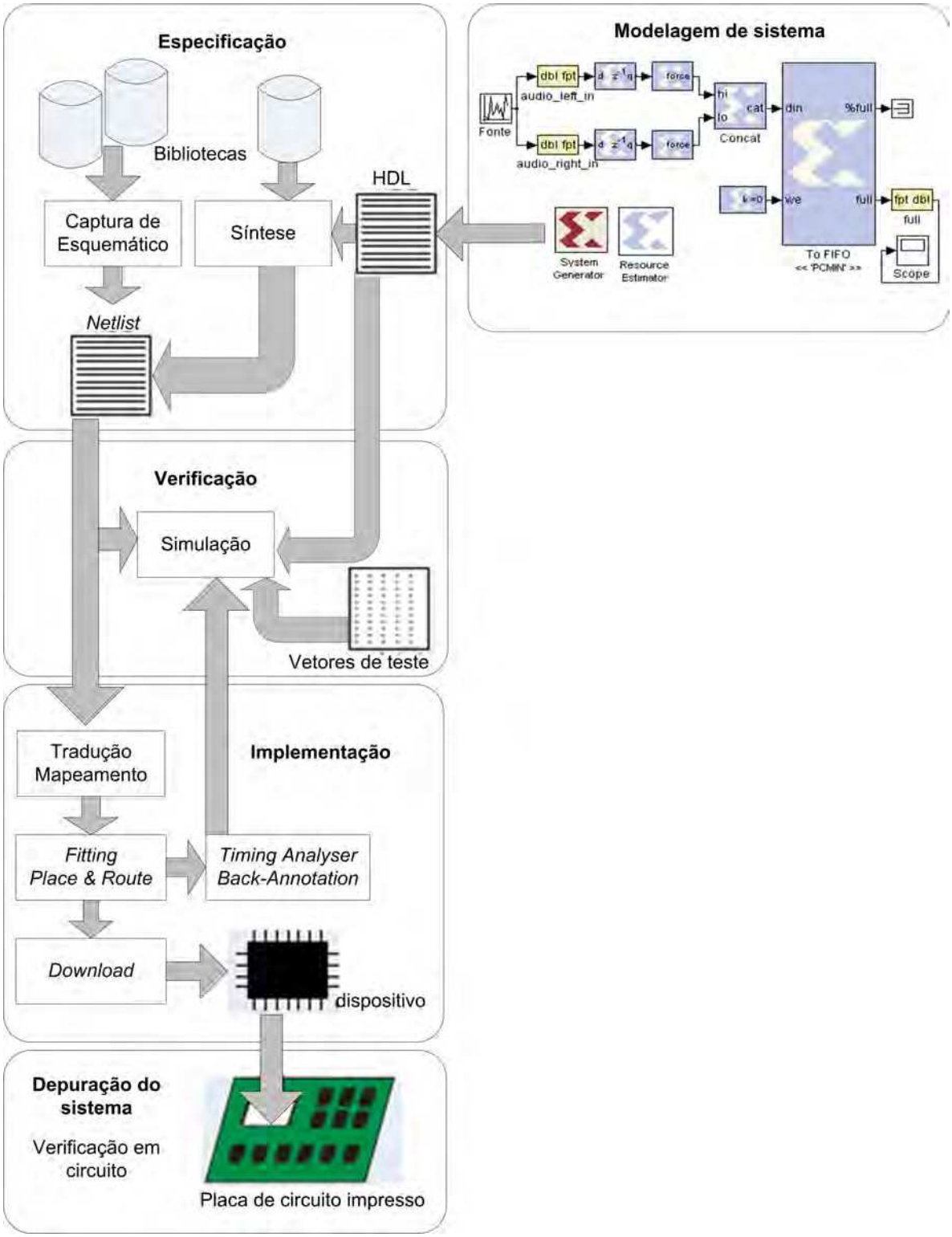

Fig. 3.9: Fluxo de projeto de uma FPGA.

fabricante de *hardware*. Portanto, a partir do código HDL utiliza-se uma ferramenta de síntese que interprete esse código e gere um *netlist* aperfeiçoado. As ferramentas de síntese de HDL obtêm a representação RTL do projeto junto com um conjunto de restrições e geram o *netlist* correspondente. Durante esse processo, a ferramenta de síntese realiza a minimização de área e otimização de tempo.

Os editores de restrições são integrados às ferramentas de síntese e permitem aplicar restrições ao projeto HDL. Pode-se adicionar restrição de frequência de *clock*, de sincronismo de entrada e saída, de sinal, entre outras. As restrições definidas na fase de síntese podem ser encaminhadas para implementação através de um arquivo de restrição de *netlist* (*Netlist Constraints File* - NCF) ou através de um arquivo de formato de troca de dados eletrônico (*Electronic Data Interchange Format* -EDIF). O EDIF é um arquivo textual que representa a hierarquia das interconexões dos *netlists* ou dos circuitos esquemáticos. No entanto, é recomendado especificar as restrições separadamente em um arquivo de restrições do usuário (*User Constraints File* - UCF). O UCF fornece controle sobre as especificações gerais, possibilita ter acesso a mais tipos de restrições e permite priorizar restrições de sinais. Uma vez especificado o *netlist* inicia-se a fase de verificação.

3.3.2 Verificação

Os projetos são verificados utilizando um simulador. O simulador é um *software* que confirma as funcionalidades de um circuito. Quando um projeto utiliza uma linguagem padronizada, por exemplo, VHDL, há garantias de que o projeto pode ser reutilizado. Se um fornecedor de FPGA alterar suas bibliotecas, apenas é necessário fazer a compilação da fase de síntese. Depois de completar a fase de síntese é necessário verificar se o circuito projetado está trabalhando conforme o esperado. Esse é o objetivo da fase de verificação. Para simular o circuito é preciso fornecer informações sobre o projeto via o *netlist*, gerado através da captura do esquemático ou da síntese HDL, e um vetor de teste que será utilizado para verificar se o circuito está funcionando de maneira apropriada. O simulador utiliza essas informações para determinar a saída do circuito.

No caso do simulador ter encontrado algum problema, o projeto deve voltar para a etapa de especificação. O projetista deverá corrigir o esquemático ou o arquivo HDL, gerar novamente o *netlist* e executar a etapa de verificação. Estimativas indicam que os projetistas dispendem 50% do tempo de desenvolvimento nas fases de especificação e verificação até que o projeto funcione como o esperado [33]. Após o projeto funcionar apropriadamente, o *netlist* é especificado e inicia-se a fase de implementação.

3.3.3 Implementação

O *netlist* descreve o projeto utilizando as unidades lógicas de um fabricante específico de FPGA. Uma vez que o *netlist* foi verificado e não foram encontrados erros é hora de colocá-lo em um *chip*. Essa fase é denominada de implementação. A etapa de implementação compreende as fases de tradução, posicionamento e roteamento, analisador de tempo e *download*.

A fase de tradução inclui vários programas que são utilizados para fazer a importação do *netlist* e prepará-lo de acordo com o *layout* da FPGA. Esses programas variam de acordo com o fornecedor da

FPGA. Alguns dos programas utilizados nesta fase obtêm os componentes e, como os mesmos estão interconectados através do *netlist*, é feito o mapeamento de tais componentes para células lógicas, que podem ser configuradas como *lookup tables* (LUT), registradores ou mesmo RAM [35]. Outros programas verificam se as regras específicas são atendidas, por exemplo, não exceder o número de *buffers* do dispositivo. Resumindo, a fase de tradução traduz o *netlist* de componentes lógicos para um *netlist* de primitivas da família do *chip* FPGA. O processo de tradução geralmente termina com um relatório completo contendo os resultados de todos os programas executados. Esse relatório contêm uma lista de erros e alertas encontrados. A fase posterior é a de posicionamento e roteamento.

Na fase de posicionamento e roteamento (*place & route*) os programas são executados depois da compilação. Posicionamento é o processo de selecionar módulos específicos ou blocos lógicos, onde as unidades lógicas projetadas residirão na FPGA. Roteamento diz respeito ao roteamento físico das interconexões entre os blocos lógicos. A maioria dos fabricantes de FPGA fornecem ferramentas de posicionamento e roteamento, de modo que não é necessário conhecer os detalhes intrínsecos da arquitetura do dispositivo. Alguns fabricantes que fornecem ferramentas para esta fase permitem que os usuários mais experientes configure a ferramenta de modo que possam obter melhor desempenho do que a configuração automática oferecida. Essa fase consome mais tempo para ser concluída porque executa uma tarefa muito complexa que é determinar a localização dos componentes do projeto no dispositivo e assegurar que todos estarão corretamente conectados, além de atender o desempenho desejado. As ferramentas de roteamento e posicionamento só poderão funcionar bem se a a arquitetura do dispositivo possuir faixas de roteamento suficientes para o projeto.

As simulações podem refletir o funcionamento do circuito de forma bastante realista através da realimentação de informações relacionadas a atraso, desempenho ou velocidade através da etapa *Timing Analizer*, que é posterior a etapa de posicionamento e roteamento. Um programa relacionado a esta etapa é chamado *Timing Driven Place and Route* (TDPR). O TDPR fornece informações de atrasos e variações de atrasos sobre os caminhos do projeto. Esta informação é muito importante e pode ser vista de diferentes maneiras, por exemplo, mostrar todos os caminhos utilizados no projeto e indicar qual caminho gera o maior atraso e qual gera o menor atraso. Portanto, a etapa *Timing Analizer* reflete os atrasos dos blocos lógicos e das interconexões entre eles. Após a fase de tradução, posicionamento, roteamento e análise de tempo é gerado um arquivo *bitstream*. O arquivo *bitstream* contém todas as informações necessárias para definir a lógica e a interconexão do projeto, portanto, é específico para cada projeto. A etapa final da implementação é o *download*.

A fase de *download* é necessária, entre outros motivos, porque a maioria das FPGAs utilizam tecnologia SRAM. Fazer o *download* de um programa para o dispositivo significa baixar as informações de configurações contidas no arquivo *bitstream* para a memória deste dispositivo. Como os dispositivos SRAM perdem a suas configurações quando a energia é desligada, o *bitstream* deve ser armazenado em local cuja tecnologia empregada seja não volátil. Geralmente, esta informação é armazenada em uma memória PROM. Para que o arquivo *bitstream* seja transferido do local onde se encontra o projeto para a FPGA é necessário utilizar a interface JTAG. JTAG é um padrão IEEEANSI 1149i_1190 e descreve um conjunto de regras de projeto que facilitam a análise, a programação do dispositivo e a depuração do *chip* ou da placa.

3.3.4 Depuração do Sistema

Depois que o *bitstream* é transferido para o dispositivo o mesmo começa a funcionar. Nessa fase, é preciso verificar se o projeto está funcionando no dispositivo real conforme havia sido planejado. Este processo é chamado de depuração do sistema. Encontrar problemas nessa fase significa que a especificação do projeto para o dispositivo em uso não foi realizada de maneira apropriada ou que alguns aspectos de sinais, cuja origem ou destino sejam o dispositivo, não foram considerados. Assim sendo, deve-se coletar os dados do problema e corrigir a modelagem do projeto.

3.4 Principais Fornecedores

O objetivo deste trabalho não é concentrar-se exclusivamente no estudo das FPGAs. No entanto, as seções apresentadas neste capítulo tornam-se úteis para a consulta ou para o conhecimento de como surgiram as FPGAs, os conceitos fundamentais relacionados e quais as formas de desenvolvimento e implementação destes dispositivos. O fluxo de desenvolvimento de um projeto em FPGA, no geral, atende aos requisitos do fluxo de projeto citado anteriormente, embora haja algumas variações nas etapas de desenvolvimento de acordo com o fabricante da FPGA. A Tabela 3.1 mostra alguns dos principais fabricantes.

Tab. 3.1: *Principais fabricantes e fornecedores de FPGA.*

Companhia	Website
Actel Corporation	http://www.actel.com
Altera Corporation	http://www.altera.com
Anadigm Incorporated	http://www.anadigm.com
Atmel Corporation	http://www.atmel.com
Cypress Semiconductor Corporation	http://www.cypress.com
Lattice Semiconductor Corporation	http://www.latticesemi.com
QuickLogic Corporation	http://www.quicklogic.com
Xilinx Incorporated	http://www.xilinx.com

Em geral, os fabricantes oferecem soluções de ferramentas completas para o desenvolvimento e implementação em FPGA. Além disso, algumas empresas se especializaram em desenvolver ferramentas completas de automatização de projetos eletrônicos para FPGAs, conforme mostra a Tabela

3.2 [34].

Tab. 3.2: *Desenvolvedores de ferramentas completas para FPGA.*

Companhia	**Website**	**Solução**
Altium Limited	http://www.altium.com	Ambiente de desenvolvimento em software e hardware
Cadence Design Systems	http://www.cadence.com	Ferramentas de simulação e ambiente de desenvolvimento inicial
Mentor graphics Corporation	http://www.mentor.com	Ferramentas de simulação e síntese
Synopsys Incorporated	http://www.synopsys.com	Ferramentas de simulação e síntese

Capítulo 4

Arquitetura de Sistema para Aplicações de Laboratório a Distância

Este capítulo apresenta uma proposta de arquitetura de sistema para a aplicação de laboratório remoto. Esta proposta visa proporcionar um ambiente genérico e flexível através de uma arquitetura orientada a serviço. Essa arquitetura possibilita a interação com a plataforma em *hardware* reconfigurável para a execução de experimento reais.

4.1 Introdução a Metodologia de Sistema

De acordo com o padrão ISO/IEC 42010 IEEE Std 1471-2000 [39], o termo sistema pode ser definido como um conjunto de componentes organizados para realizar uma função específica ou um conjunto de funções. Na proposta de arquitetura de sistema para aplicação de laboratório remoto, o termo sistema engloba os subsistemas, as aplicações, o sistema no sentido tradicional do termo e outras funcionalidades que podem ser agregadas. A aplicação difere do sistema no sentido que a aplicação é uma descrição de um grupo lógico de competências que gerencia os objetos de dados necessários para processar os dados e fornecer suporte ao modelo de negócio do sistema.

Para que um sistema interaja com outro sistema ou outra aplicação, geralmente se estabelece um acordo ou contrato entre o provedor e o consumidor do sistema. Esse acordo é uma descrição entre um provedor do sistema e um cliente, que pode ser um usuário final ou outro provedor, especificando o serviço a receber. Ele delimita o escopo da aplicação do sistema além de descrever o tipo e a natureza do serviço a ser fornecido.

Cada sistema possui uma arquitetura que o descreve. A arquitetura é definida como a organização fundamental do sistema, englobando seus componentes e as relações entre eles e o ambiente, assim como os princípios que regem a sua concepção e evolução [39]. De acordo com o contexto onde é empregado, o termo arquitetura é definido como uma descrição formal do sistema, ou um plano detalhado do sistema em nível de componentes para orientar sua implementação.

A arquitetura descreve as funcionalidades e o comportamento do sistema. Para chegar nessa des-

crição é importante responder a algumas perguntas sobre o sistema, tais como: Quais são os dados importantes para o sistema? Como os processos são executados? Onde estão as pessoas que utilizarão esse sistema? Quem são as pessoas mais importantes para o sistema? Durante quanto tempo o sistema estará disponível? Por que o sistema deve ser concebido, ou seja, qual a motivação para a construção do sistema? [40]

4.2 A Arquitetura Proposta para Aplicações de Laboratório Remoto

Um dos objetivos deste trabalho é apresentar uma arquitetura de sistema para a aplicação de laboratório remoto. Em sistema de execução remota para experimentos laboratoriais, os dados mais importantes para o sistema são os dados referentes ao experimento. Tanto os dados necessários para a execução do experimento quanto os dados resultantes são de grande importância para o sistema devido ao seu valor agregado e à informação nele contida. Também os dados referente às pessoas que utilizam o sistema são importantes porque através dessa informação é possível desenvolver experimentos remotos que atendam a um público alvo específico.

Os processos definidos no sistema são executados para realizarem as tarefas de gerar, processar, armazenar, executar e apresentar as informações do experimento remoto. Também devem executar as tarefas de controle e gerenciamento de usuários.

A princípio, os usuários do sistema são os professores, os alunos e os pesquisadores. Portanto, o público alvo do sistema está em universidades, faculdades, centros de pesquisa ou em suas residências trabalhando remotamente. Por isso, a situação ideal é que o sistema esteja disponível para acesso remoto 24 horas por dia para que o usuário tenha a liberdade de escolher o horário que lhe for mais conveniente para acessá-lo. A princípio o sistema estará disponível para acesso através da rede KyaTera, descrita na seção 2.3.1, e da rede Internet convencional.

A concepção do sistema possibilita a interação entre os usuários e os experimentos remotos através de uma plataforma de *software* integrada a uma plataforma de *hardware* reconfigurável, que executa em tempo real e permite o controle e o monitoramento de experimentos laboratoriais.

A arquitetura proposta para aplicações de laboratório remoto descreve o sistema com base nas funcionalidades a serem fornecidas, evitando a redundância de implementação e a perda de foco na missão do sistema. A missão do sistema é prover acesso remoto ou no local do laboratório em tempo real a professores, alunos ou pesquisadores para a execução de experimentos laboratoriais de sistemas digitais utilizando a plataforma REDLART.

As macro-funcionalidades do sistema são modularizadas e apresentadas como serviços. Os serviços são módulos com baixo acoplamento que possuem uma interface independente de sua implementação. A descrição da arquitetura em módulos permitiu diminuir o nível de complexidade do sistema, facilitou a tomada de decisões sobre o desenvolvimento e possibilitou identificar a sinergia entre os

módulos que compõem o sistema. A definição da arquitetura ajudou na escolha das soluções técnicas para desenvolver o sistema. A Figura 4.1 mostra uma visão em camada da arquitetura proposta.

Fig. 4.1: A arquitetura proposta para aplicações de laboratório remoto.

A arquitetura proposta é composta por 4 camadas: Aplicação *WebLab*, *Middleware*, Infraestrutura de Rede e REDLART. Cada camada possui uma função específica e executa sua função independentemente das outras camadas. Cada camada possui uma interface de programação de aplicação (*Application Programming Interface* - API) que estabelece o contrato ou a maneira como uma camada pode comunicar-se com a outra.

As camadas se comportam como provedoras ou consumidoras de serviços. A camada REDLART fornece serviços para a camada de *middleware*. A camada de *middleware*, por sua vez, usa os serviços da camada de REDLART e fornece serviços para o camada de infraestrutura de rede. A camada de infraestrutura de rede usa os serviços fornecidos pela camada de *middleware* e fornece serviços para a aplicação *WebLab*. A camada aplicação *WebLab* utiliza os serviços fornecidos pela camada infraestrutura de rede. Observe que existe um contrato bem definido que impede que aplicação *WebLab* se comunique diretamente com o REDLART, por exemplo. Este contrato é definido através da API especificada em cada camada. A camada aplicação *WebLab* não possui todas as informações necessárias especificada na API da camada REDLART, portanto, não poderá acessar esta camada diretamente. Para acessá-la é necessário, primeiramente, acessar a camada *middleware* que por sua vez irá acessar a camada infraestrutura de rede e finalmente acessará a camada REDLART. As camadas serão vistas em detalhes nas seções seguintes.

4.3 Aplicação WebLab

Os *WebLabs* são laboratórios onde experimentos, equipamentos e sistemas reais podem ser controlados a distância através da Internet. Desenvolvidos com o intuito de viabilizar o compartilhamento de equipamentos e ideias entre pesquisadores de diferentes laboratórios, os *WebLabs* abrem um leque

de possibilidades inovadoras na pesquisa colaborativa e no ensino a distância [16].

O objetivo do *WebLab* é prover acesso remoto a experimentos de laboratório, permitindo que os usuários possam utilizar todos os recursos de um equipamento remoto. Os *WebLabs* podem ser desenvolvidos por pesquisadores, professores ou alunos. A situação ideal é que os *WebLabs* estejam disponíveis para acesso remoto 24 horas, por dia 7 dias por semana e que sejam acessados através de uma rede de banda larga estável.

O módulo aplicação *WebLab* representa o experimento do laboratório que será executado. Experimentos de filtros digitais, de processamento de áudio, de codificação/decodificação de vídeo, entre outros. Por exemplo, suponha que um usuário está executando um experimento que envolva a captura e a reprodução de fluxo de sinais de áudio. O usuário pode selecionar a fonte do sinal, o tipo de processamento, controlar o volume e determinar onde o sinal de áudio processado deve ser reproduzido. Esse módulo é responsável pela ordem da execução, ou seja, a sequência do processo. Neste caso, a sequência do processo, com base nas informações fornecidas pelo usuário, seria primeiro fazer a captura do fluxo de sinal de áudio, depois processar o fluxo de sinal capturado, obter o resultado do processamento e por último reproduzir o fluxo de sinal resultante. Observe que essa camada não faz de fato a captura, processamento e reprodução do fluxo de sinal de áudio. Porém, essa camada conhece a camada que poderá executar estes serviços.

A camada aplicação *WebLab* então encaminha os valores dos parâmetros do experimento definido pelo o usuário para a camada de *middleware*. A camada de *middleware*, por sua vez, verificará se possui todos os serviços necessário para atender à requisição da camada aplicação *WebLab*. Em caso afirmativo, executará os serviços e retornará o resultado para a camada de aplicação *WebLab*. Em caso negativo, encaminhará os dados recebidos para a camada REDLART. É o que acontece neste caso. A camada de *middleware* não tem os serviços responsáveis por acessar os dispositivos de *hardware* para realizar a captura, o processamento e a reprodução do fluxo de sinal de áudio. Então a camada de *middleware* enviará uma solicitação para a camada REDLART que executará os serviços necessários para atender à requisição e encaminhará a resposta para a camada de *middleware*. A camada de *middleware*, por sua vez, encaminhará a resposta para a camada aplicação *WebLab* através da camada de infraestrutura de rede.

Portanto, a camada de aplicação *WebLab* possui o conhecimento do contexto e a ordem de execução. Logo, é nessa camada que são definidos os processos ou as regras de negócio do experimento. O usuário interage diretamente com esta camada. A execução dessa camada pode exigir dados do usuário para iniciar ou continuar o processo. Consequentemente, a natureza da execução não é atômica, ou seja, a execução pode parar, retroceder e continuar.

4.4 A camada de *Middleware*

A camada de *middleware* é constituída por uma plataforma de *software* que é responsável por fazer a comunicação entre a camada de aplicação *WebLab* e a camada REDLART através da rede local, da rede KyaTera ou da Internet convencional. Além disso, realiza o gerenciamento do usuário,

o gerenciamento de controle da aplicação *WebLab* e o gerenciamento dos serviços necessários para executar o experimento de laboratório. O gerenciamento do usuário e o gerenciamento de controle da aplicação *WebLab* utilizam soluções consolidadas de segurança para fazer o gerenciamento de contas do usuário e o acesso a aplicação, respectivamente.

Para implementar a camada de *middleware* foi utilizado um projeto de *software* livre denominado AppFuse [41]. Esse projeto integra as mais recentes tecnologias J2EE[1] (*Java 2 Enterprise Edition*) para construção de aplicações *Web*. O código-fonte está disponível em diferentes combinações de *frameworks*, incluindo os de persistência e apresentação. O *AppFuse* utiliza o padrão de projeto *Model-View-Controller* (MVC) e já oferece um protótipo inicial com funcionalidades que são comuns em toda aplicação *Web*, tais como *login*, segurança, envio de e-mail e camada de acesso a dados. Outros pontos positivos do projeto AppFuse são: a possibilidade de gerar automaticamente as telas e as tabelas do sistema a partir do diagrama de classes da aplicação, ser fortemente fundamentado em testes e de ser integrado a um gerenciador de dependências [41].

Atualmente, o padrão de projeto MVC é muito utilizado em aplicações *Web* porque facilita o desenvolvimento e a manutenção do projeto como um todo e emprega o conceito de arquitetura multicamadas. A arquitetura multicamadas diminui a complexidade da implementação e permite separar e padronizar o código fonte das camadas. Em [42] encontram-se mais informações sobre o MVC. A estrutura do *middleware* é formada pelas camadas de modelo, de controle e de apresentação ou visão seguindo o padrão de projeto MVC.

4.4.1 Modelo

A descrição do modelo representa os dados da aplicação e as regras de negócio que gerenciam o acesso e a atualização desses dados. O modelo é uma camada do *middleware* que é responsável por controlar os dados da aplicação, responder às consultas da camada de apresentação a respeito do estado e atualizar o seu estado quando solicitado pela camada de apresentação [27].

A camada de modelo contém os dados comuns e os dados que são específicos da aplicação. Os dados comuns são aqueles utilizados pelo *middleware* para executar a interação com a aplicação *WebLab* e a camada REDLART. Para interagir com a aplicação *WebLab* é necessário ter uma classe específica para armazenar os dados do usuário. Cada usuário da aplicação pode assumir um papel diferente (papel de usuário ou de administrador da aplicação) por isso se faz necessário armazenar em uma determinada classe qual o papel que determinado usuário possui na aplicação. Os dados específicos referem-se às informações que são peculiares do experimento. Esses dados devem ser encaminhados para a camada REDLART realizar o processamento desses dados ou serem obtidos da camada REDLART para serem visualizados na aplicação *WebLab*. A Figura 4.2 mostra o diagrama de classes do *middleware*.

[1] define o padrão para o desenvolvimento de aplicações empresariais. J2EE é uma plataforma de programação voltada para aplicações multicamada com base em componentes modulares. Tais módulos são executados em um servidor de aplicação utilizando a linguagem de programação Java.

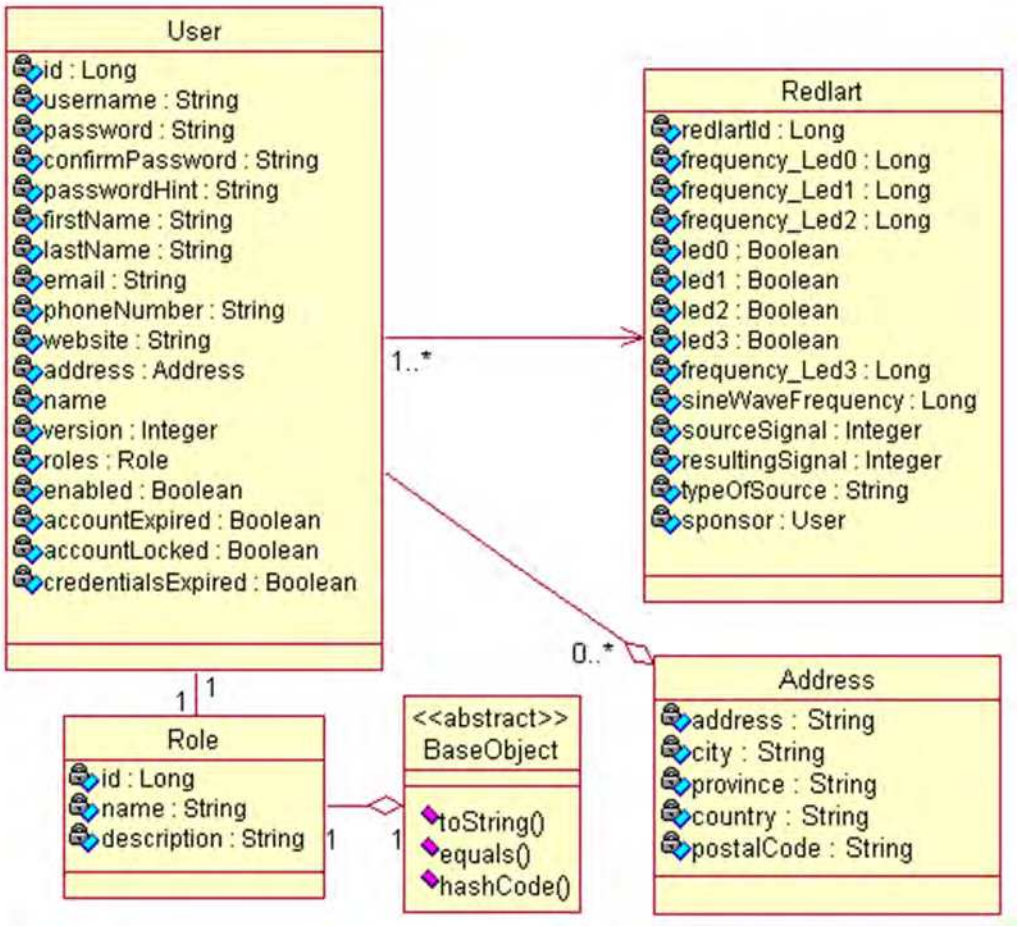

Fig. 4.2: Diagrama de classes do middleware.

As classes *User*, *Role*, *BaseObject* e *Address* fazem parte dos dados comuns da aplicação. A classe *User* contém os dados pessoais e da conta de acesso do usuário da aplicação. Essa classe está relacionada com a classe *Address* que encapsula as informações de endereço do usuário. A classe *Role* contém informações referentes ao papel que o usuário pode assumir. Os possíveis papéis são *user* ou *admin*. O usuário que possua o papel de *user* pode editar suas informações pessoais, executar e consultar os experimentos *Web* disponíveis, entrar em contato com o administrador para fazer críticas, sugestões e indicar novos experimentos *WebLabs*. O usuário com o papel de *admin* pode gerenciar as contas dos usuários cadastrado na aplicação e fazer as operações de inserção, atualização, remoção e consulta dos experimentos. Todo usuário cadastrado na aplicação deve possuir um papel associado. É considerada uma boa prática de programação que cada objeto do modelo contenha os métodos *toString()*, *equals(Object o)* e *hashCode()*. Esses métodos estão definidos na classe abstrata *BaseObject*. Cada objeto do modelo deve implementar esta classe.

A camada de modelo contém os dados que são específicos da aplicação. Um experimento *WebLab* que realiza o controle da frequência de *leds*, de volume de áudio, de parâmetros de equalização, entre outros, deve armazenar apropriadamente os valores dos parâmetros em uma classe. Por isso, no

modelo de dados há a classe Redlart. A classe Redlart é responsável por armazenar informações dos parâmetros do experimento *Web*.

A camada de modelo também é responsável por persistir as propriedades e os atributos definidos nas classes. Para fazer a persistência dos dados utiliza-se o padrão de projeto DAO (*Data Access Object*). Esse padrão de projeto encapsula a lógica de acesso a dados e define uma interface que pode ser implementada para cada nova fonte de dados utilizada. Isso viabiliza a substituição de uma implementação por outra. Além disso, gerencia a conexão com a fonte de dados para obter e armazenar os dados. Também permite a utilização de um *pool* de conexões gerenciado internamente pelo DAO, sendo, portanto, transparente para as outras partes do sistema [27]. Nos DAOs são realizadas as operações CRUD (*Create*, *Read*, *Update* e *Delete*) típicas de um banco de dados [42].

A camada de *middleware* utiliza o banco de dados PostgreSQL. O PostgreSQL é uma ferramenta de *software* livre para sistema de base de dados. Esse banco de dados possui uma arquitetura que fornece confiabilidade e integridade dos dados [43]. O acesso a base de dados PostgresSQL é feito através do *framework* Hibernate. O Hibernate é uma plataforma de alto desempenho para persistência de dados e serviços de consulta. Ele permite que o projetista desenvolva classes persistentes seguindo o paradigma de programação orientado a objeto (*Object Oriented Programming* - OOP)[2] [44]. O Hibernate permite realizar consultas em sua própria linguagem HQL (*Hibernate Query Language*), assim como em linguagem nativa SQL (*Structured Query Language*) [45].

Na camada de *middleware* existe uma classe DAO associada para cada classe User e Redlart. O DAO comunica com a base de dados satisfazendo o contrato proposto por uma interface de programação de aplicativos (API) que é especificado em cada método DAO. As operações básicas do DAO de inserir, atualizar, remover e consultar estão definidas na classe GenericDao. As classes DAOs devem herdar a classe GenericDao para executar aquelas operações. As operações de consultas também são definidas nos DAOs respectivamente, UserDao, RedlartDao e RoleDao. A Figura 4.3 mostra a definição dessas classes.

A classe UserDaoHibernate é responsável por manipular as informações dos usuários na aplicação. A classe RoleDaoHibernate manipula os papéis que o usuário pode assumir. A classe RedlartDaoHibernate gerencia as informações dos experimentos *WebLabs* que são executados no módulo REDLART. Cada experimento será representado na base de dados por um identificador único chamado *redlartId*. Associados ao *redlartId* estarão os dados referente àquele experimento. Toda vez que for inserido um novo experimento na aplicação será gerado um novo identificador e, caso seja removido um experimento, o registro contendo as informações será excluído da base de dados. A classe RedlartDao também permite realizar buscas específicas. Por exemplo, dado o valor do parâmetro frequência do *led* é possível obter a lista de experimentos que atendem a esse requisito. Seguindo a abordagem da construção da camada de *middleware*, a camada seguinte da estrutura é a camada de serviço. A camada de serviço está no âmbito da camada de modelo segundo o padrão de projeto MVC.

[2]é um paradigma de análise, projeto e programação de sistemas de *software* com base na composição de um conjunto de objetos que interagem entre si.

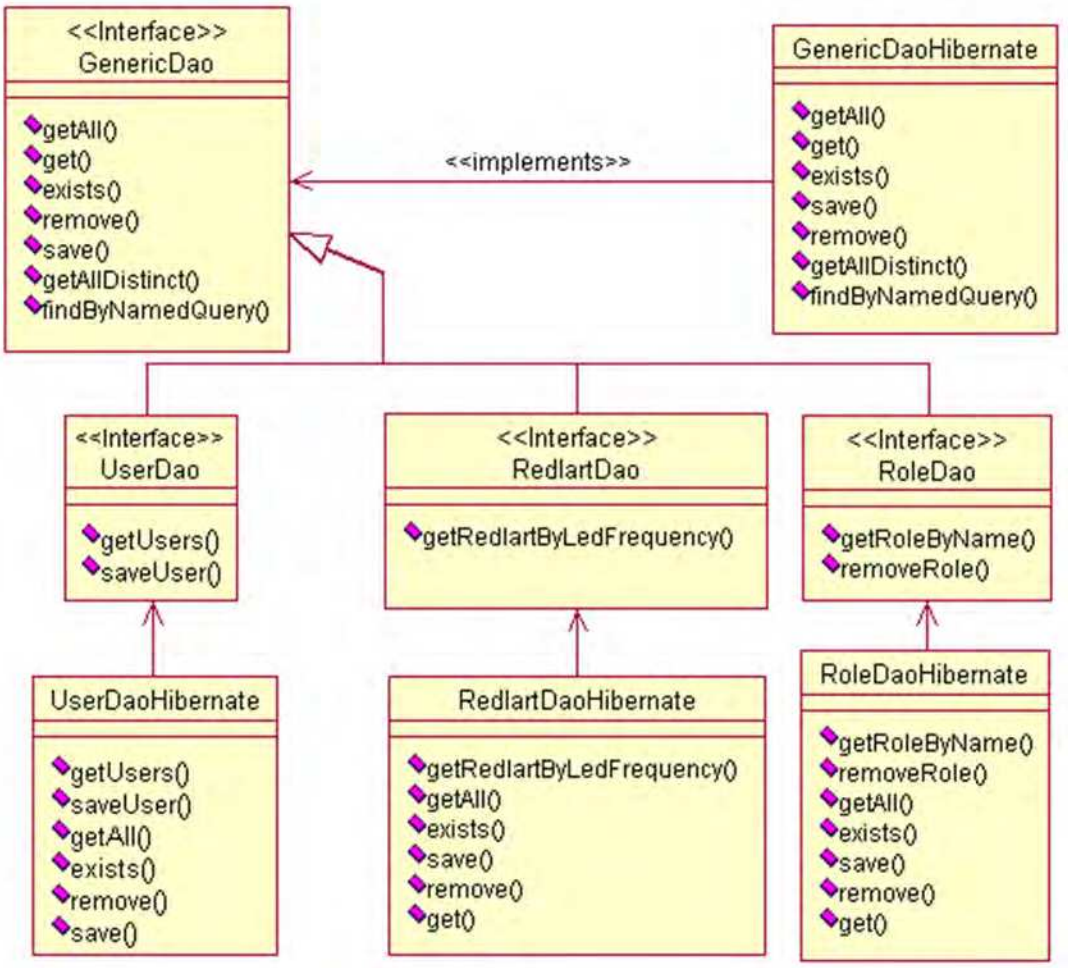

Fig. 4.3: Representação das classes Daos utilizadas no middleware.

A camada de serviço é conhecida como camada lógica, de negócio, ou camada de transação. Nessa camada são especificadas todas as regras de negócio que devem ser satisfeitas. As regras de negócios representam as macro-funcionalidades da aplicação. Essas macro-funcionalidades são modularizadas e apresentadas como serviço. Os serviços são módulos com baixo acoplamento, cujas interfaces são independentes da implementação. As vantagens de utilizar uma arquitetura orientada a serviço é que estes são independentes e podem ser providos independentemente da plataforma tecnológica. Além disso, possuem interfaces bem definidas que promovem a troca de informações. Essas características proporcionam o reuso de serviços e facilitam a manutenção da camada de *middleware*.

A camada de serviço é formada por gerenciadores de serviço. O gerenciador de serviço é um grupo organizado e integrado de atividades correlatas. Ele pode utilizar vários DAOs. Os gerenciadores de serviço são criados para fornecer serviços mais complexos e também estabelecer a comunicação com os DAOs.

A estrutura das camadas de *middleware* vista até o momento define que na primeira camada

encontra-se o modelo de dados e o DAO associado a esses dados para fazer a persistência dos mesmos. É o caso do UserDao, RoleDao e RedlartDao. A camada de serviço gerencia transações, sendo capaz de manter várias chamadas DAOs dentro de uma mesma transação. Também provê os serviços que serão utilizados pelo módulo aplicação *WebLab*. A Figura 4.4 mostra os gerenciadores de serviço do *middleware*.

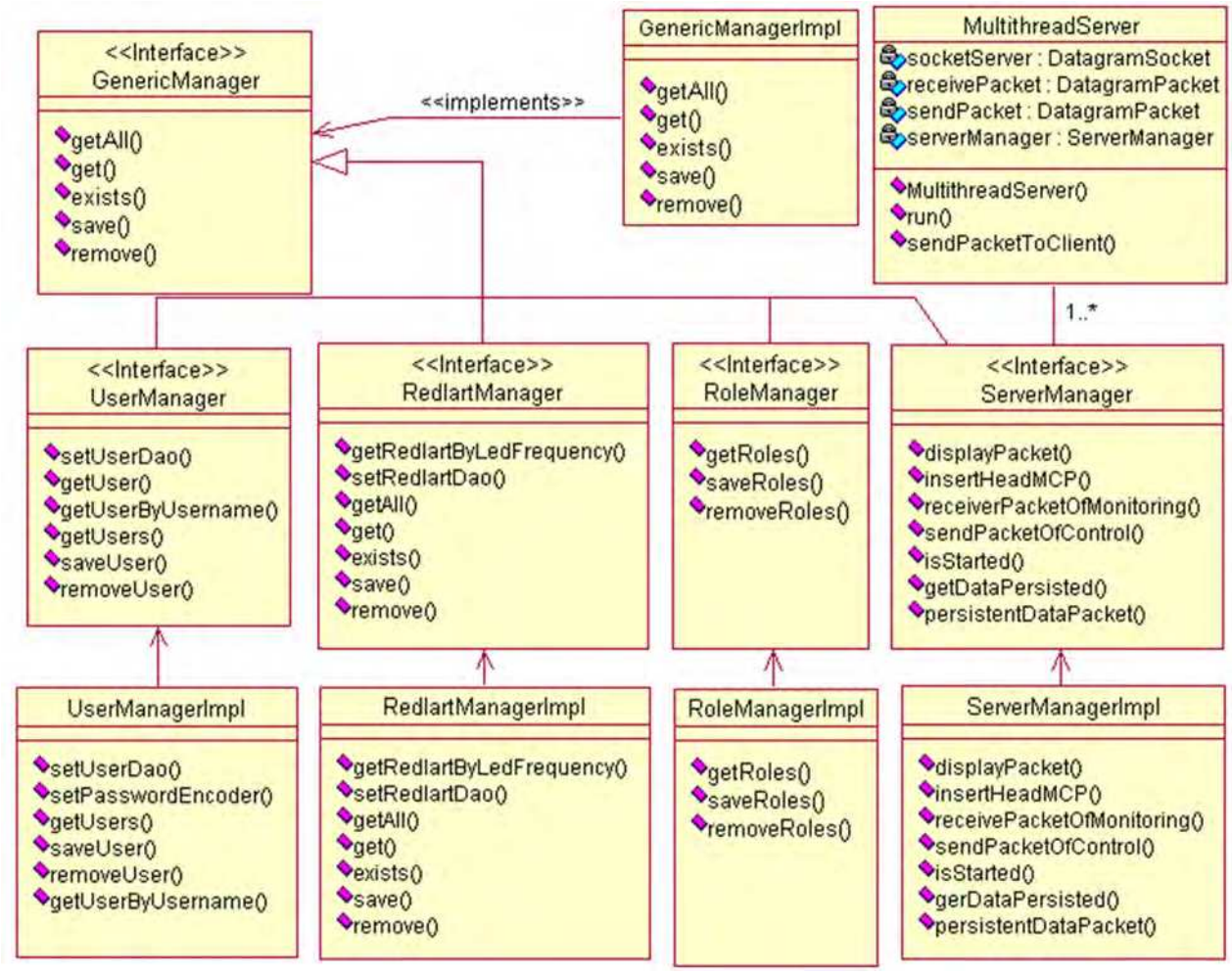

Fig. 4.4: Representação dos principais serviços utilizados no middleware.

O UserManager, o RoleManager e o RedlartManager definem as regras de negócio através de um contrato de serviços de seus respectivos modelos de dados. Os contratos de serviços são especificados através da interface e as regras de negócio são representadas pelos métodos. Esses gerenciadores de serviços também determinam a comunicação entre a camada de modelo e a camada de apresentação. Por exemplo, um dos serviços definidos no RedlartManager é obter a lista de Redlart, dada uma determinada frequência do *led*. Essa lista de Redlart obtida é então encaminhada para a camada de apresentação que fará a formatação desses dados para serem mostrados de forma mais apropriada para o usuário.

A classe MultithreadServer é responsável por fazer a comunicação entre os módulos REDLART e o *middleware*. Essa comunicação é estabelecida através de um *DatagramSocket*[3]. Quando um usuário altera um parâmetro do experimento, é enviado um pacote UDP através do *socket* para indicar a alteração do parâmetro. O módulo REDLART recebe esse pacote, faz o processamento necessário e envia um pacote UDP para o *middleware* indicando o novo valor desse parâmetro. A comunicação entre os módulos REDLART e o *middleware* é ponto a ponto e deve permanecer estável indefinidamente, a menos que ocorra algum problema na rede. A classe MultithreadServer emprega o conceito de *multithread*[4]. Isso garante que o *middleware* poderá aceitar várias conexões de REDLART distintas e poderá gerenciar essas conexões paralelamente. Por exemplo, suponha um experimento distribuído em dois módulos REDLART. Cada camada REDLART é formada por uma plataforma de *hardware* REDLART. Para cada plataforma REDLART a camada de *middleware* deverá possuir uma conexão particular para gerenciar os dados oriundos ou destinados da mesma e apresentá-los ao usuário que está executando o experimento. Isso deve ser feito paralelamente de maneira que atenda a todos os requisitos do experimento de maneira satisfatória.

A classe MultithreadServer utiliza os serviços definidos na interface ServerManager, implementada pela classe ServerManagerImpl. Esta classe é responsável por gerenciar a comunicação entre o *middleware* e o módulo REDLART, através dos seguintes serviços: *displayPacket*, *insertHeadMCP*, *sendPacketOfControl* e *receivePacketOfMonitoring*. Além disso, oferece serviços que interagem diretamente com os DAOs para fazer a persistência dos dados na base de dados.

Quando o pacote de dados chega na camada de *middleware* oriundo da camada REDLART significa que esse pacote contém os dados atualizados do experimento. Portanto, esses dados devem ser apresentados na interface do experimento de maneira apropriada. Essa é a função do serviço *displayPacket*. O serviço *insertHeadMCP* é responsável por inserir o cabeçalho do protocolo MCP (*Monitoring and Control Protocol*) no campo de carga útil do protocolo UDP. Esses protocolos serão abordados no Capítulo 5. Por enquanto, é importante destacar que esses protocolos determinam uma maneira padronizada de transferir dados na rede. Os três tipos de pacotes de dados que trafegam entre a camada REDLART e a camada de *middleware* são os pacotes de controle, de monitoração e de requisição. O pacote de controle é enviado do *middleware* para a REDLART contendo todos os valores dos parâmetros do experimento sempre que ocorre uma alteração por parte do usuário. O serviço responsável por gerar este pacote é o *sendPacketOfControl*. Após o módulo REDLART receber o pacote de controle e fazer o processamento, ele envia um pacote de monitoração para indicar que a alteração dos parâmetros foi executada com sucesso. O serviço *receiverPacketOfMonitoring* é responsável por receber o pacote de monitoração e atualizar os dados na base de dados. O pacote de requisição é enviado do *middleware* para a REDLART para solicitar que a plataforma REDLART envie pacotes de monitoração.

A camada de serviço pode fornecer para a camada de apresentação serviços que são considerados de propósito geral ou específico para a aplicação. Por exemplo, conversor de data e conversor de hora são considerados serviços utilitários, enquanto que os serviços de *receiverPacketOfMonitoring*

[3]representa um *socket* para enviar e receber pacotes UDP (*User Datagram Protocol*). Um *socket* é um serviço ponto a ponto de envio e entrega de pacotes.

[4]permite que múltiplos programas executem no mesmo ambiente do processo de forma paralela.

e *sendPacketOfControl*, citados anteriormente, são específicos da aplicação. A camada de serviço interage diretamente com a camada de dados e com a camada de controle.

4.4.2 Controle

A camada de controle, assim como a camada de modelo é uma subcamada do *middleware*. A camada de controle traduz a interação do usuário com a camada de apresentação em ações a serem executadas pela camada de modelo. Em uma aplicação com interface gráfica de usuário, interações do usuário podem significar o *click* de um botão, a seleção de um menu, entre outros eventos. A ação executada pela camada de modelo inclui ativar as regras de negócio ou alterar o estado do modelo. De acordo com a interação do usuário e de acordo com a resposta da ação executada pela camada de modelo, a camada de controle responde selecionando a visão (*view*) apropriada [42].

Na camada de middleware, a subcamada de controle é representada por JavaBeans[5] gerenciáveis. Os *beans* gerenciáveis são também conhecidos por *backing beans*. Eles são responsáveis por intermediar a comunicação entre a camada de apresentação e a camada de modelo.

Para criar um *backing bean* é necessário obedecer algumas convenções. Entre essas convenções ele deve possuir um construtor nulo e para cada atributo definido deve ter os métodos acessores (*get* e *set*) associados. Em alguns casos, como por exemplo, quando o estado da aplicação for armazenado no lado cliente, também é necessário estender a classe *serializable*[6]. A Figura 4.5 mostra os *beans* utilizados no *middleware*.

Os *backing beans* que são definidos com o nome *List*, UserList e RedlartList, são utilizados para obter os dados da camada de modelo e repassar para a camada de apresentação. Quando a aplicação mostra a lista de experimentos que estão em execução é porque o *backing bean* acionou o serviço correspondente. O serviço, por sua vez, retornou para o *backing bean* a lista de experimentos disponíveis sendo apresentado pela camada de apresentação.

Os *backing beans* que são definidos com o nome *Form*, por exemplo UserForm, RedlartForm e SignupForm, são utilizados para obter os dados fornecidos pelo usuário (através da camada de aplicação). Quando o usuário acessa a página de *login* ele insere os seus dados de acesso (*nome do usuário* e *senha*). O *backing bean* irá capturar essa informação e chamará o serviço responsável por fazer a validação do usuário. O serviço verificará se o usuário está cadastrado na base de dados e se os dados fornecidos são válidos. Caso o usuário seja validado o *backing bean* indicará à camada de apresentação para mostrar a página principal da aplicação. Caso o usuário não seja validado ele indicará a camada de apresentação que mostre uma mensagem indicando que o nome do usuário ou a senha estão incorretos.

[5] JavaBeans são componentes de *software* reutilizáveis escrito na linguagem de programação Java. Eles são utilizados para encapsular muitos objetos dentro de um único objeto (o *bean*).

[6] a serialização é o processo de converter um objeto em uma sequência de bits para que ele possa ser armazenado em um arquivo ou um *buffer* de memória ou ainda ser transmitido através de um enlace de conexão de rede.

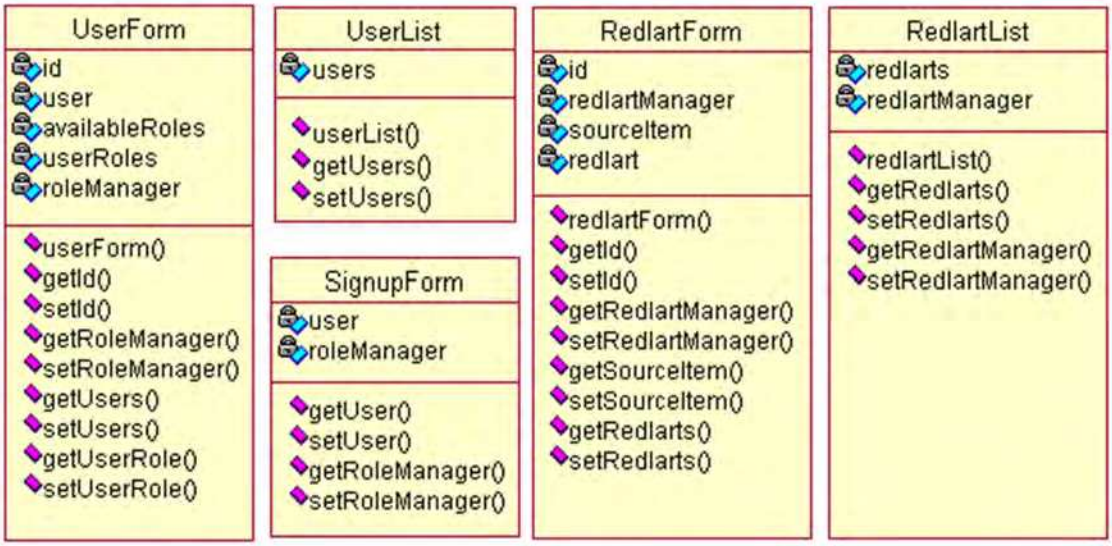

Fig. 4.5: Diagrama dos beans gerenciáveis utilizados no middleware.

4.4.3 Apresentação

A camada de apresentação mostra a visão do modelo de dados e responde às entradas dos usuários através da camada de controle, instruindo o modelo para atualizar seus dados [27]. Essa camada acessa os dados da aplicação através do modelo e mantém a consistência da apresentação dos dados quando o modelo é alterado.

As aplicações *Web* são hospedadas em servidores. Esses servidores são chamados de *container*[7] *servlet*[8]. O *container servlet* precisa dos arquivos compilados de uma aplicação *Web* para prover acesso à aplicação. O *container* também pode ser considerado um ambiente contextual que executa as aplicações *Web* desenvolvidas em linguagem JAVA. Todas as aplicações *Web* Java são hospedadas por um programa Java executável que internamente cria *threads* para cada requisição do navegador *Web* do usuário. Isso garante que vários usuários podem acessar o servidor da aplicação com a sensação de simultaneidade, pois o servidor atenderá aos pedidos de requisição de maneira paralela.

As aplicações *Web* com base na tecnologia Java estão organizadas em coleções de componentes, páginas *Web*, classes Java e arquivos de propriedades. Essas configurações são necessárias para o servidor *Web* hospedar um *servlet* ou (*JavaServer Pages* - JSP)[9] utilizando um navegador *Web*. Quando um usuário acessa uma página da Internet e clica em um componente desta página, por exemplo, um

[7] em programação orientada a objetos o *container* é um objeto que contém outros objetos que podem ser incluídos ou removidos em tempo de execução.

[8] O *servlet* é um *container* específico para aplicações *Web*. O *servlet* é formado por uma classe Java que processa requisições e respostas dinamicamente, proporcionando novos recursos aos servidores.

[9] é uma tecnologia Java empregada no lado do servidor que permite aos desenvolvedores de *software* criar páginas *Web* geradas dinamicamente com HTML ou XML. O JSP interage com os clientes *Web* utilizando o modelo requisição/resposta.

botão, o navegador *Web* submete uma requisição HTTP[10] (*HyperText Transfer Protocol*) para o servidor Java. O *servlet* aceita a requisição e utiliza o protocolo HTTP para enviar a resposta da requisição. A resposta pode ser, por exemplo, uma página *Web*, uma imagem, um vídeo ou um arquivo de texto. A aplicação *Web* deve conter todos os elementos que o servidor precisa para executar esta operação.

Para o desenvolvimento de aplicações *Web*, o *middleware* utiliza a plataforma *JavaServer Faces* (JSF)[11]. O JSF é uma arquitetura orientada a componente de interface de usuário com base no padrão de projeto MVC [41]. O estado do componente é salvo quando o cliente faz uma requisição de uma nova página e restaurado quando o servidor envia a resposta. As Figuras 4.6 e 4.7 mostram alguns exemplos de interface de usuário utilizadas no *middleware*.

Para que o usuário tenha acesso aos experimentos remotos é necessário que ele esteja cadastrado no *middleware*. Atualmente, o cadastro é feito mediante o fornecimento de algumas informações pessoais como nome e senha que são cadastrados previamente. Futuramente, essas informações poderão ser obtidas de uma base de dados de uma universidade, por exemplo. O usuário acessa a página do experimento e fornece o seu *nome de usuário* e sua *senha* para se autenticar na aplicação. O sistema apresentará a tela inicial da aplicação como mostra a Figura 4.6.

Fig. 4.6: Interface página principal da aplicação WebLab.

Na página principal são apresentadas as opções de editar o perfil do usuário e visualizar a lista de experimentos disponíveis. Além disso, é possível monitorar e controlar um experimento que está em

[10] é um protocolo da camada de aplicação do Modelo OSI utilizado para transferência de dados e hipermídia.

[11] é uma plataforma de aplicação *Web* que utiliza a tecnologia Java para estabelecer um padrão de desenvolvimento de interfaces do usuário no lado servidor [46].

execução sob a plataforma REDLART. Caso o usuário clique na aba *Lista Redlart* serão mostradas as informações dos parâmetros de monitoração de todos os experimentos que estão em execução. No entanto, se o usuário clicar na aba *Redlart1* serão apresentados os parâmetros de monitoração e controle do experimento executado na REDLART1. A Figura 4.7 mostra um exemplo de página que é apresentada ao usuário da aplicação *WebLab* para demonstrar o resultado de um experimento de captura de fluxo de sinal de áudio.

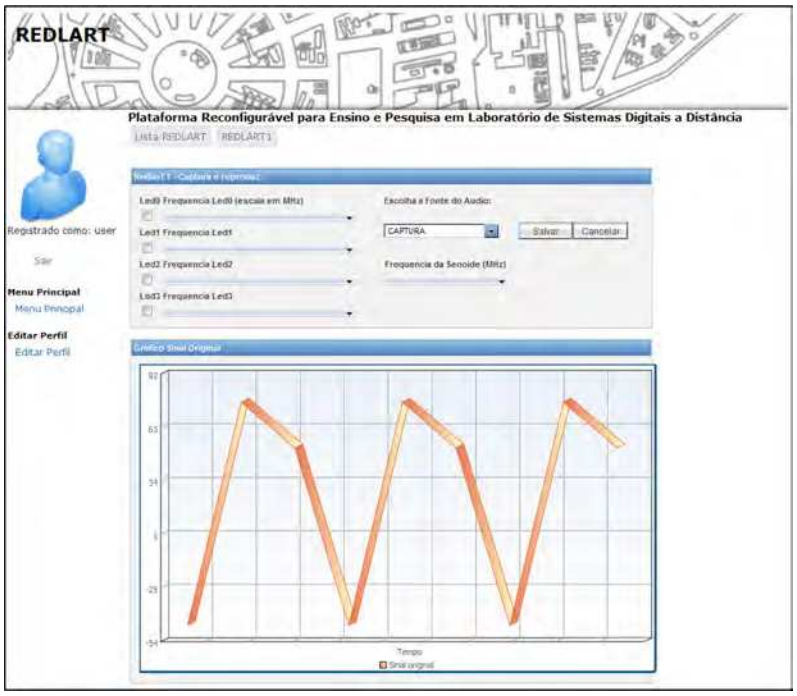

Fig. 4.7: Interface do experimento de captura do fluxo de sinais de áudio.

Essa página possibilita fazer o controle de alguns parâmetros daquele experimento. Por exemplo, é possível controlar a opção de acender um *led* com uma determina frequência ou apagá-lo. Além disso, o usuário poderá escolher qual é a fonte do fluxo de sinal de áudio. Nesse caso, o fluxo de sinais de áudio será obtido diretamente através do *codec* de áudio da plataforma REDLART. O usuário pode monitorar o fluxo de sinal de áudio capturado por meio do gráfico apresentado na Figura 4.7. Esse experimento será descrito em detalhes no Capítulo 6.

Portanto, a camada de apresentação é responsável por permitir a interação do usuário com o sistema. Caso ocorra uma alteração nessa camada, isso não irá interferir ou alterar o modo como os dados são buscados no banco de dados ou a lógica do sistema.

4.5 A Camada de Infraestrutura de Rede

A camada de infraestrutura de rede é responsável por estabelecer a troca de informações entre as camadas de *middleware* e de aplicação *WebLab*. Esse camada representa a infraestrutura de rede

que pode ser utilizada para estabelecer tal comunicação. Atualmente, além da rede mundial de computadores (Internet) é possível utilizar redes de computadores dedicadas ao ensino e pesquisa como, por exemplo, a rede do Projeto Kyatera (descrita na seção 2.3.1) e a rede do Projeto Giga (descrita na seção 2.3.2). Essas redes acadêmicas são mais robustas porque sua infraestrutura de rede pode prover extensa largura de banda, baixa variação do atraso e a taxa de perda de pacotes é pequena. Para representar essa diferença na infraestrutura, a camada de infraestrutura de rede é formada pelos sub-módulos Intranet, Internet e KyaTera. No entanto, é importante ressaltar que independente da rede de computadores utilizada, a comunicação acontece através dos protocolos de rede definidos no Modelo TCP/IP descrito na RFC-1180 [47].

Do ponto de vista da camada REDLART não é possível identificar se os pacotes vieram através da rede Internet, KyaTera ou Intranet. Quando o usuário faz uma requisição, essa requisição é enviada para a camada de *middleware*. A camada de *middleware* recebe o pedido, verifica e envia para a camada Redlart, através da camada de infraestrutura de rede, apenas a informação que é necessária para processar a requisição. Portanto, para a camada Redlart, é transparente a informação sobre qual sub-módulo de comunicação foi utilizado.

Capítulo 5

Plataforma REDLART

Este capítulo apresenta a especificação de requisitos, a modelagem e alguns cenários de utilização da plataforma REDLART. Também são apresentados o fluxo de projeto de desenvolvimento e as ferramentas utilizadas na implementação da plataforma REDLART.

5.1 Especificação de Requisitos

O desenvolvimento de projetos para dispositivos FPGA visando sistemas de comunicações digitais e processamento digital de sinais pode reutilizar uma arquitetura básica de *hardware* para acelerar o processo de desenvolvimento. Tal procedimento se torna ainda mais evidente ao se desenvolverem projetos de experimentos de laboratório tais como filtros e osciladores digitais, filtragem adaptativa, sistema de geração de sinais, transmissão e monitoração de pacotes sobre Ethernet, entre outros. De fato, constata-se não ser necessário especificar uma arquitetura de *software* e de *hardware* que seja totalmente nova para cada novo projeto de experimento a ser desenvolvido. A situação ideal é reutilizar uma infraestrutura básica de *hardware* e *software* disponível e a partir dessa infraestrutura desenvolver novos projetos de experimentos. Nesse contexto, a infraestrutura deve atender aos seguintes requisitos estruturais:

- disponibilizar interfaces de *hardware* para comunicação com dispositivos locais de aquisição e de visualização de dados;
- ser modular e flexível para facilitar a integração de novas funcionalidades tais como dispositivos externos ou outros módulos de função NPI (núcleos de propriedade intelectual - IP *Cores*[1]);

Além dos requisitos estruturais, considerando-se que as aplicações são experimentos de laboratório, pode-se definir também os seguintes requisitos funcionais:

- fornecer o acesso ao experimento em tempo real;
- prover o compartilhamento de recursos;

[1]bloco funcional existente que pode ser utilizado como solução para grandes sistemas. As três fontes principais destes núcleos são, em ordem, blocos criados em projetos anteriores, blocos fornecidos por fabricantes de FPGA e blocos fornecidos por desenvolvedores de NPIs [31], [34]

- permitir o controle e a monitoração do experimento;
- estar disponível para acesso remoto para que o usuário não esteja limitado a utilizar a infraestrutura básica apenas no local do laboratório.

Ao requisito de acesso remoto deve-se adicionar ainda que a solução emule o ambiente de laboratório para que o usuário possa vivenciar a mesma experiência que teria na prática. Entretanto, essa funcionalidade não está no escopo da plataforma de *hardware*, mas no escopo do *middleware* (plataforma de *software* cliente-servidor) entre o usuário e o experimento remoto. Para fornecer o acesso remoto em tempo real também é necessário que a infraestrutura de rede tenha baixa latência para viabilizar uma interação em tempo real com o experimento e tenha largura de banda suficiente para suportar os fluxos de dados requeridos. As redes de computadores de alta velocidade mencionadas na Seção 2.3 fornecem extensa largura de banda, baixo atraso e podem atender a esses requisitos de desempenho de rede.

Um dos fundamentos e requisitos deste trabalho é possibilitar a prototipagem rápida de novos experimentos sem exigir a montagem de uma equipe especializada nos aspectos técnicos de implementação em *hardware*, mantendo-se apenas a equipe especializada na funcionalidade que se deseja implementar. Detalhes sobre a implementação de *drivers* e interfaces de dispositivo, assim como mudança de domínio de *clock*, devem ser abstraídas pela plataforma. Nesse contexto, o desenvolvimento de um novo projeto de experimento deve ter uma abordagem sistêmica fundamentada nos conceitos de processamento digital de sinais.

Um exemplo de como a abstração dos detalhes de *hardware* é realizada pela plataforma pode ser percebida no processo de mudança de domínio de *clock*. Tipicamente, o sinal amostrado pode ser obtido através de um conversor AD (Analógico Digital) em um *clock* que pode ser diferente do utilizado no módulo onde será realizado o processamento. Após uma mudança de domínio de *clock*, o sinal é disponibilizado em rajada para o módulo de processamento. Nesse caso, é necessário então padronizar uma interface que inclua, além do sinal amostrado, uma sinalização de controle para indicar quais são as amostras válidas. O desenvolvimento do projeto deve então levar em consideração que o sinal a ser processado é de tempo real e disponibilizado em rajada.

Uma das aplicações da plataforma de *hardware* é suportar a colaboração no desenvolvimento de um experimento. Neste trabalho, o conceito de colaboração é definido pela possibilidade de desenvolvimento de partes do experimento em locais diferentes, integrados a partir de uma infraestrutura de rede. Esse requisito impacta diretamente a forma de aquisição dos sinais. Em vez dos sinais serem amostrados por um conversor, os mesmos devem ser transmitidos como fluxo de pacotes em tempo real via rede IP. Uma vez recebidas as amostras do sinal pela rede, as mesmas são disponibilizadas em rajada para o processamento, necessitando também de uma sinalização de controle para indicar as amostras válidas.

No contexto de um laboratório remoto, para desenvolver um experimento é necessário que a plataforma REDLART forneça acesso a componentes de *hardware* padrão como controladores, *drivers*, *codecs*, conversores, interface Ethernet, entre outros. A solução encontrada para permitir que o experimento possa ter acesso a essa infraestrutura de componentes é implementada em um módulo de

interface de dados e de comunicação responsável pela abstração e comunicação com esses dispositivos.

Em linhas gerais, a plataforma REDLART deve ser constituída por uma placa de circuito impresso contendo, obrigatoriamente, uma FPGA Xilinx e um PHY *Ethernet* 10 Mbps/100 Mbps. O *hardware* da plataforma pode incluir ainda dispositivos de aquisição e reprodução de dados como o *codec* de áudio AC97 e conversores AD e DA (Digital Analógico). A especificação dos conversores AD e DA depende do experimento a ser desenvolvido. A arquitetura da plataforma disponibiliza componentes funcionais para realizar a mudança de domínio de *clock* entre os conversores e o módulo de processamento. É importante destacar que devido à implementação atual da arquitetura, há uma restrição na taxa dos conversores em até 100 MHz. Os conversores AD e DA também podem ser externos à placa, conectados a partir de pinos de expansão.

Esses requisitos de *hardware* são atendidos na prova de conceito deste trabalho através dos kits de desenvolvimento XUP da Digilent com uma FPGA Virtex II Pro 2VP30 da Xilinx, um chip LM4550 da National (*codec* de áudio AC97), um *chip* PHY Ethernet 10/100 da Intel (modelo LXT972A) e pinos de I/O. Uma vantagem desse kit, além de possuir uma FPGA de alto desempenho (com 3 milhões de *gates* + 2 PowerPC + blocos de RAM + multiplicadores embarcados), é o seu custo reduzido de apenas U$ 299,00 (não inclusos os impostos) para o programa universitário da Xilinx (XUP - *Xilinx University Program*). Esse kit de desenvolvimento assim como um resumo dos principais recursos disponíveis são ilustrados no Apêndice A.2.

5.2 Modelagem da Plataforma

Para atender aos requisitos citados anteriormente, a arquitetura da plataforma foi estruturada em três módulos principais: Interfaces de Dados, Comunicação e Processamento. Uma visão geral da arquitetura da plataforma REDLART é apresentada na Figura 5.1.

O módulo de interface de dados é responsável por realizar toda a captura e reprodução local de sinais. Esse módulo também é responsável pela mudança de domínio de *clock* necessária para disponibilização do sinal em rajada para o módulo de processamento, como mencionado na seção anterior, e para a reprodução do sinal obtido pelo módulo de processamento. Portanto, o fluxo de dados entre os módulos de processamento e interface de dados é bidirecional. Toda abstração e comunicação com dispositivos de captura e reprodução é realizada por esse módulo.

O módulo de comunicação é responsável por gerenciar todo fluxo de pacotes em tempo real via rede IP e disponibilizar em rajada para o módulo de processamento. Esse módulo também é responsável por gerenciar a comunicação entre a plataforma de *hardware* (REDLART) e a plataforma de *software* cliente-servidor (*middleware*) para possibilitar a monitoração e o controle remoto do experimento. Nesse módulo são definidos os procedimentos e os protocolos para que tal comunicação aconteça.

De maneira geral, o módulo de comunicação envia pacotes de monitoração para o *middleware*

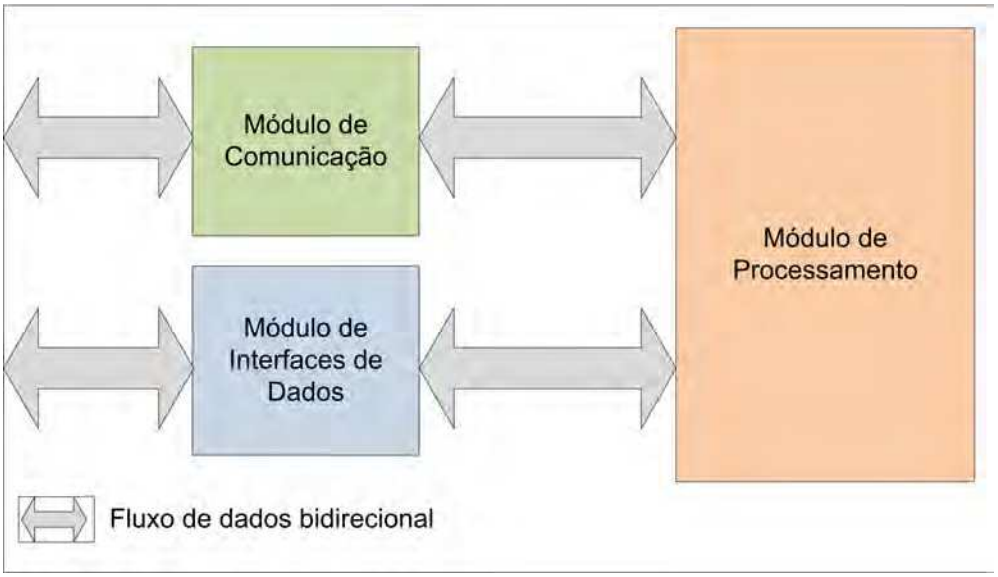

Fig. 5.1: Visão geral da plataforma REDLART.

para que o mesmo possa monitorar parâmetros de variação lenta do experimento desenvolvido no módulo de processamento, como, por exemplo, temperatura e pressão. O *middleware*, por sua vez, sinaliza a alteração de valor do parâmetro gerando pacotes de controle para o módulo comunicação. Para experimentos que geram fluxos de dados de tempo real com variação rápida, como áudio e sinais eletrônicos, o módulo comunicação gera pacotes de sinais para transporte do sinal entre plataformas remotas que implementam partes de um experimento, ou para visualização remota do sinal. O módulo de comunicação fornece suporte à troca de pacotes de monitoração, controle e fluxo de dados através do protocolo MCP (*Monitoring and Control Protocol*). O protocolo MCP foi desenvolvido para transportar e identificar os fluxos de dados possíveis da plataforma, ou seja, de controle, de monitoração e de sinal. Detalhes sobre a especificação desse protocolo são descritos na Seção 5.2.4.

O módulo de processamento permite que o usuário desenvolva novos experimentos utilizando uma solução própria ou incorpore novas funcionalidades através de NPIs existentes e os integrem à infraestrutura básica da plataforma. Através do módulo de interfaces de dados, o módulo de processamento têm disponíveis interfaces para comunicação com dispositivos locais de aquisição de sinais. Através do módulo de comunicação, o módulo de processamento pode se comunicar diretamente com o *middleware* e o experimento pode ser desenvolvido de maneira colaborativa. Portanto, a integração do experimento ao módulo de processamento permite que o mesmo possa usufruir de toda a infraestrutura de componentes de *software* e *hardware* que a plataforma REDLART oferece.

5.2.1 Cenários de Utilização da Plataforma

Para ilustrar como os módulos de comunicação, de interfaces e de processamento são acionados em uma aplicação, dois cenários de uso são apresentados nesta seção. O primeiro cenário mostra um típico acesso remoto a um experimento desenvolvido na plataforma REDLART. Nesse cenário o experimento consiste basicamente na digitalização de um sinal fonte, no seu processamento e na

reprodução do sinal processado. Simultaneamente a essas fases, deve-se permitir que fluxos de sinais pré-especificados possam ser monitorados e que parâmetros do experimento possam ser controlados remotamente. No segundo cenário o experimento é dividido em duas partes. Cada parte é desenvolvida em um local distinto, porém o experimento é integrado através da infraestrutura de rede para operar como um único experimento.

A Figura 5.2 ilustra o primeiro cenário de utilização da plataforma. O experimento do primeiro cenário é iniciado pela aquisição de sinais realizada pelo módulo de interfaces de dados. Exemplos de fontes de sinais que podem ser citadas são sinais de áudio, de fala, de vídeo, de radiofrequência, de sensores diversos, entre outros. Após a aquisição, os fluxos de sinais são multiplexados e disponibilizados em rajada para o módulo de processamento. No módulo de processamento, os fluxos de sinais podem ser processados aplicando-se técnicas de processamento digital de sinais, tais como codificação e modulação. Após a fase de processamento, o fluxo de sinal é encaminhado, paralelamente, para o módulo de interfaces de dados e para o módulo de comunicação. O módulo de interfaces de dados pode realizar a reprodução e visualização local dos sinais. Por outro lado, o módulo de comunicação possibilita que o fluxo de sinal possa ser transmitido via pacotes IP para a aplicação cliente-servidor.

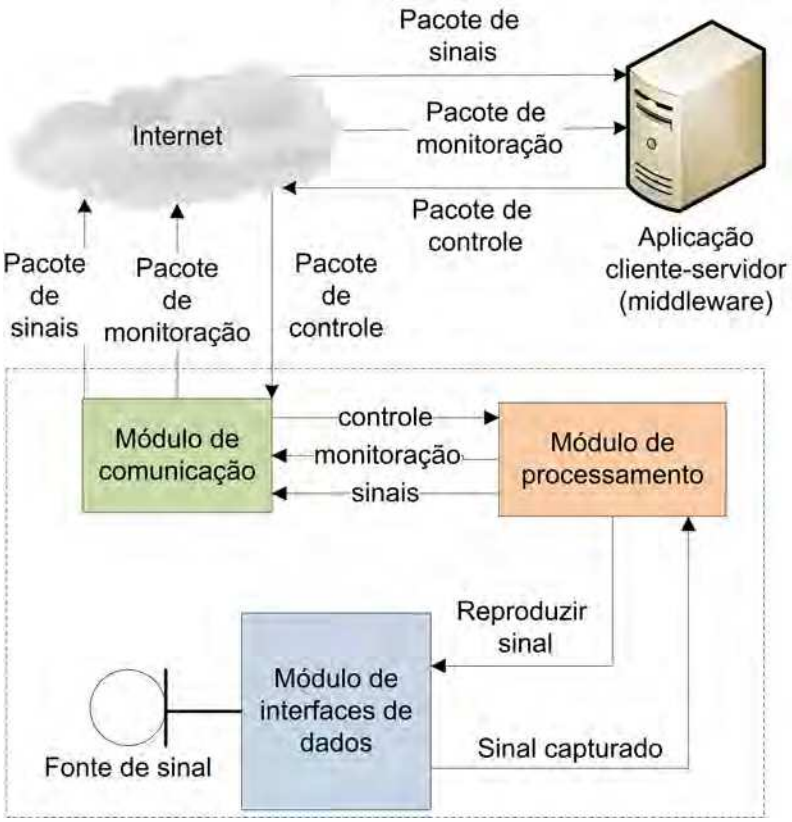

Fig. 5.2: Primeiro cenário de utilização da plataforma.

O módulo de comunicação também possibilita que a aplicação cliente-servidor possa controlar e monitorar o experimento através de pacotes IP. O pacote de monitoração é gerado com base nos valo-

res dos parâmetros de variação lenta do experimento, por exemplo, controle de volume, temperatura, taxa de codificação, tipo de modulação, coeficientes de filtros, entre outros. Na aplicação cliente-servidor os pacotes de fluxo de sinal são disponibilizados para a visualização e reprodução no cliente e os pacotes de monitoração são utilizados para indicar ao usuário os valores atuais dos parâmetros do experimento. O usuário pode realizar o controle dos parâmetros de variação lenta. Para atender a esse requisito, a aplicação cliente-servidor gera pacotes de controle que são recebidos pelo módulo de comunicação. No módulo de comunicação, os pacotes são desencapsulados, armazenados em uma memória RAM e disponibilizados para o módulo de processamento, onde tais parâmetros são efetivamente utilizados.

No segundo cenário ilustra-se a possibilidade de se executar partes do experimento em locais distintos. Ambas as partes são implementadas na plataforma REDLART e integradas através do módulo de comunicação e pela infraestrutura de rede para operar como um único experimento. A Figura 5.3 ilustra esse cenário. O fluxo de sinal é capturado de uma fonte geradora de sinal utilizando o módulo de interface de dados. O fluxo de sinal é disponibilizado para o módulo de processamento que apenas encaminha o sinal para o módulo de comunicação. O módulo de comunicação gera pacotes de sinais e envia esses pacotes, através da Internet, diretamente para a parte número dois do experimento. O módulo de comunicação também pode gerar pacotes de monitoração e receber pacotes de controle para monitorar e controlar, respectivamente, os parâmetros do experimento.

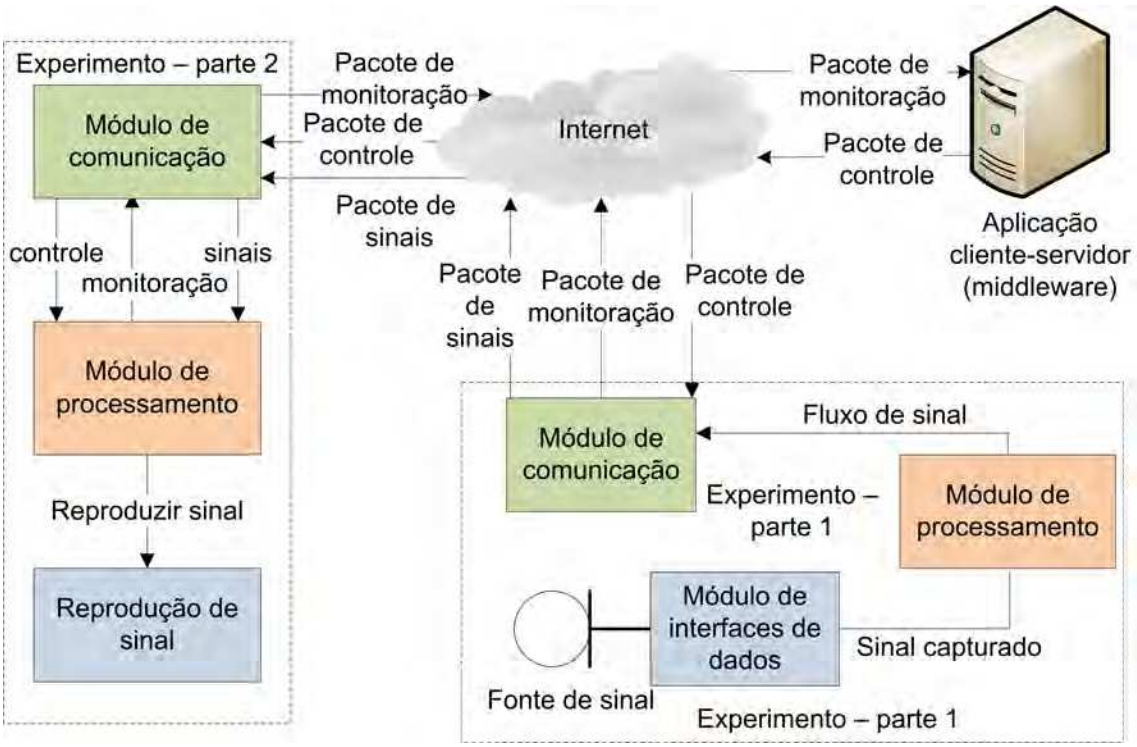

Fig. 5.3: Segundo cenário de utilização da plataforma.

Na parte número dois do experimento, os pacotes de sinais são recebidos, desencapsulados e o

fluxo de sinal obtido é entregue para o módulo de processamento. O módulo de processamento, por sua vez, realiza o processamento do fluxo de sinal em tempo real e encaminha o sinal processado para reprodução local através do módulo de interface de dados. É importante destacar que esse mesmo sinal poderia também ser transmitido pela rede através do módulo de comunicação para reprodução e visualização remota no local da primeira parte do experimento.

Na segunda parte do experimento, observe que o módulo de comunicação trabalha em paralelo com o módulo de interfaces de dados. Enquanto o módulo de interface de dados realiza a reprodução do sinal, o módulo de comunicação recebe pacotes de controle e de fluxo de sinal e também envia pacotes de monitoração. Como no primeiro cenário, o módulo de comunicação pode atualizar os parâmetros de controle escrevendo os dados recebidos pela rede em uma memória RAM. A monitoração é feita pela leitura dessa memória e pela transmissão dos dados pela rede. Os parâmetros do experimento estão disponíveis para o módulo de processamento através dessa memória RAM. O módulo do experimento também pode escrever na RAM para atualizar os parâmetros do experimento. Quando isso ocorre, pacotes de monitoração são gerados pelo módulo de comunicação para atualização de tais parâmetros na aplicação cliente-servidor. A aplicação cliente-servidor disponibiliza os dados para visualização e controle do experimento por parte do usuário.

5.2.2 Módulo de Interface de Dados

O módulo de interfaces de dados é responsável por fazer a captura e a reprodução de fluxo de dados e de sinais e também realizar a mudança de domínio do *clock* para que o sinal possa ser disponibilizado para o módulo de processamento. O sinal obtido pode ser então utilizado, através do módulo de processamento, em diversos experimentos de processamento digital de sinais em tempo real, como por exemplo, filtragem, equalização, extração de características, síntese, entre outros. No kit de desenvolvimento da Digilent, que foi utilizado na prova de conceito deste trabalho, estão disponíveis várias interfaces, tais como: interface para o *clock* do sistema, *clock* SATA (*Serial Advanced Technology Attachment*), *clock* do usuário, memória *flash* interna, memória *flash* externa, USB (*Universal Serial Bus*), *codec* de áudio, saída de vídeo, *leds*, *switches*, botões, portas RS-232[2] e PS/2, porta SATA e pinos de expansão. Esse kit não possui dispositivos AD e DA. Porém, os mesmos podem ser acoplados ao kit através dos pinos de expansão. A Figura 5.4 mostra o diagrama de blocos das interfaces de dados disponíveis no kit utilizado.

Das interfaces de dispositivos disponibilizados pelo kit da Digilent, foram implementadas na prova de conceito as interfaces de *codec* de áudio AC97, *leds*, botões e pinos de expansão. A interface de *clock* do sistema especifica que o ciclo de operação do sistema e dos módulos de comunicação e de processamento é de 100MHz. Entretanto, a interface do *clock* do sistema especifica que o ciclo de operação do módulo de interfaces de dados é de 24 MHz para possibilitar a captura e a reprodução do fluxo de sinal. A interface USB é usada para a programação ou configuração do *chip* FPGA quando o experimento desenvolvido no módulo de processamento está pronto para ser implementado no *chip*. Alguns experimentos podem permitir que o usuário interaja diretamente com a plataforma REDLART. Essa interação ocorre através de *leds* e botões que são utilizados como indicadores visu-

[2]é um padrão para troca serial de dados entre um terminal de dados e um comunicador de dados.

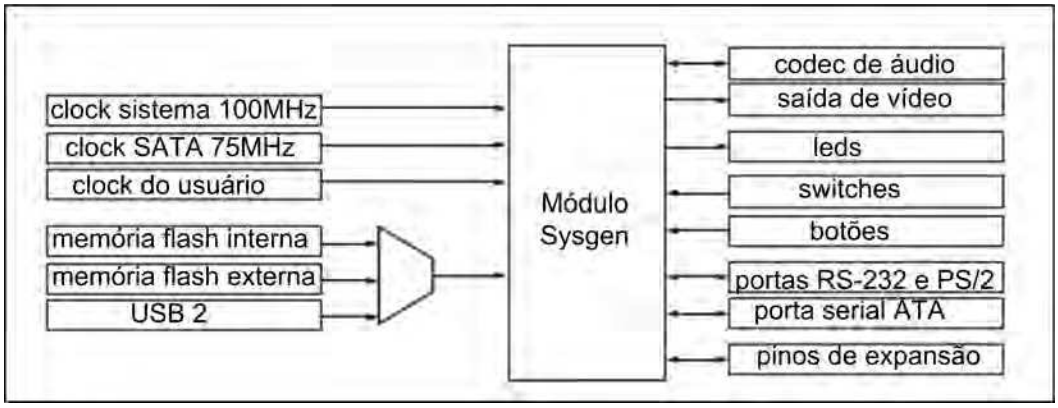

Fig. 5.4: Diagrama de bloco das interfaces de dados disponíveis.

ais para o usuário. Entretanto, a interface de comunicação Ethernet, que é implementada no módulo de comunicação, também pode ser utilizada para se implementar interfaces mais sofisticadas de usuário do que simples *leds* e botões.

Na prova de conceito implementou-se a interface do *codec* AC97 para ilustrar a reprodução e captura de sinais na plataforma. Procedimento semelhante ao da implementação da captura e reprodução de sinais de áudio no módulo de interfaces se aplica para os conversores de sinais AD e DA genéricos. O *codec* de áudio AC97 (*chip* LM4550 AC97 da *National Semiconductors*) realiza a conversão DA (amostragem) e AD (reprodução) em dois canais de áudio (esquerdo e direito) com amostragem de até 48 kHz e quantização de até 24 bits. Na prova de conceito a captura e reprodução foram implementadas para 48kHz e 16 bits em ambos os canais estéreo.

O *chip* AC97 é formado por um *codec* e um controlador de áudio. O *clock* entre o *codec* AC97 e o controlador AC97 é de 12,288 MHz. Esse *clock* é derivado, internamente dividido por dois, do sinal 24,576 MHz originário da entrada do gerador de *clock*. Entre o controlador AC97 e o módulo de processamento, o *clock* do sistema é de 24 MHz e a taxa de amostragem é de 48 KHz, o que resulta em um período de amostragem 500 vezes maior do que o período de *clock* de 24 MHz. O período de amostragem pode ser obtido através de um sinal de *clock enable*. A Figura 5.5 mostra em detalhes o módulo de interface de dados.

O controlador AC97 separa as funções analógicas e digitais do sistema de áudio. A interface do controlador AC97 permite obter o sinal de áudio digital. O módulo de interface de dados disponibiliza, através da interface do controlador AC97, o fluxo de sinais do áudio para o módulo de processamento. O fluxo de sinais contém informações sobre sinais de controle e sinais de dados. As informações sobre o *clock* e o pulso de sincronização são transmitidas para o módulo de processamento através dos sinais AC97_BIT_CLOCK e AC97_SYNCH, respectivamente. O controlador AC97 sinaliza os dados seriais de entrada através do AC97_SDATA_IN e os dados seriais de saída através do AC97_SDATA_OUT. Também é disponibilizado para o gerenciamento do controlador AC97 o sinal AUDIO_RESET_Z que é utilizado no módulo de interfaces de dados para reiniciar o *codec* AC97.

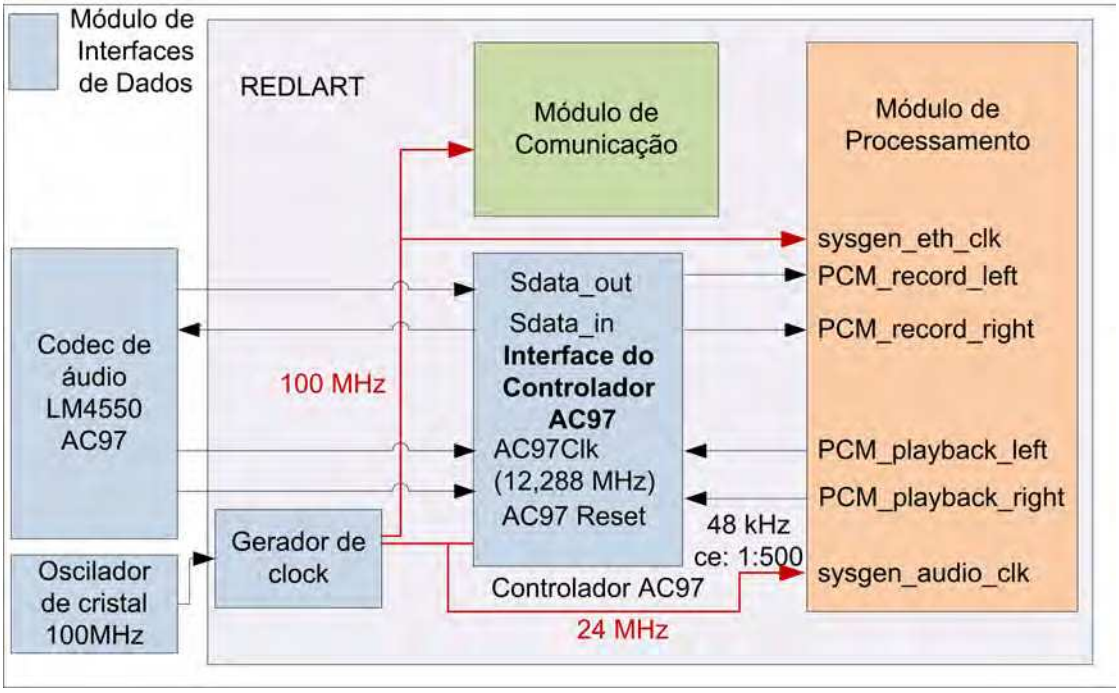

Fig. 5.5: Detalhamento do módulo de interfaces de dados.

5.2.3 Módulo de Comunicação

O módulo comunicação é responsável pela comunicação de pacotes, através de uma infraestrutura de rede, tais como Internet, KyaTera ou Intranet, entre a plataforma de *hardware* REDLART e o *middleware* da aplicação *WebLab*. O sinal a ser processado é obtido pelo desempacotamento dos pacotes recebidos na ordem em que chegam na FIFO (*First In First Out*) de recepção. O sinal é então disponibilizado em rajada para o módulo de processamento utilizando uma sinalização LocalLink. De forma semelhante, os sinais a serem transmitidos pela rede devem ser primeiramente empacotados segundo a pilha de protocolos MCP/UDP/IP/Ethernet. Para que o pacote possa ser transmitido ou recebido pela rede de computadores é necessário um adaptador de rede, um controlador do adaptador de rede e uma FIFO síncrona. Esse módulo opera com *clock* de 100 MHz. A Figura 5.6 mostra o módulo de comunicação em detalhes.

O adaptador de rede utilizado na prova de conceito é o *chip* LXT972A PHY (Intel) que é um transceptor de camada física Ethernet. Ele suporta operações bidirecionais a 10Mbps ou 100Mbps. O controlador do adaptador de rede é o Tri-Mode Ethernet MAC (*Media Access Controller*) da Xilinx. O núcleo do controlador do adaptador de rede suporta conexão unidirecional e bidirecional, com operações de 10 Mbps e 100 Mbps. O Tri-Mode Ethernet MAC foi projetado para o padrão IEEE 802.3 [48] e é responsável pelo gerenciamento de controle de fluxo dos pacotes da rede Ethernet. O MAC é responsável por enviar os pacotes para o adaptador de rede fazer a transmissão e a recepção. Os pacotes recebidos da Internet são armazenados em uma FIFO de recepção (FIFO RX) e ficam disponíveis para o módulo de processamento, através de uma interface de sinalização LocalLink [49]. De forma semelhante, os pacotes gerados pelo módulo de processamento são armazenados, através de

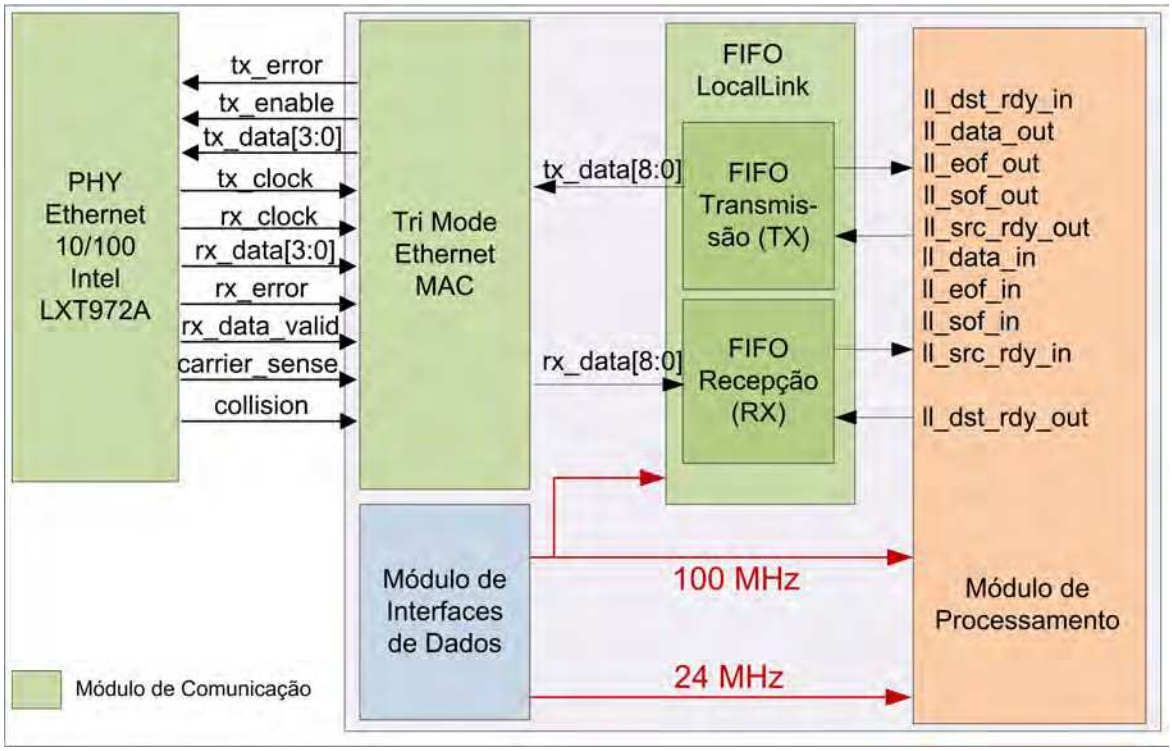

Fig. 5.6: Detalhamento do módulo de comunicação.

uma interface LocalLink, em uma FIFO de transmissão (FIFO TX), são lidos pelo controlador MAC e então transmitidos pela rede.

O módulo de comunicação disponibiliza, através da interface da LocalLink, o fluxo de sinais para o módulo de processamento. A interface LocalLink possui dois sinais de dados e oito sinais de controle. O módulo de processamento sinaliza que está pronto para receber dados da FIFO RX através do sinal de controle ll_dst_rdy_in. O sinal de dados é disponibilizado pela FIFO RX em ll_data_out, no qual os dados são de N bits onde N pode assumir um dos seguintes valores: 8, 16, 32, 64 ou 128. O ll_sof_out sinaliza o início da transferência do quadro no barramento ll_data_out. O sinal ll_eof_out representa o fim da transferência do quadro no barramento ll_data_out. No quadro sinalizado pelos sinais ll_sof_out e ll_eof_out, também é utilizado o sinal ll_src_rdy_out para sinalizar as amostras válidas.

Os dados são escritos na FIFO TX pelo módulo de processamento através do sinal ll_data_in, onde os dados são de P bits e P pode assumir um dos seguintes valores: 8, 16, 32, 64 ou 128. O quadro de dados a ser escrito na FIFO tem seu início e fim sinalizados pelos sinais ll_sof_in e ll_eof_in, respectivamente. O sinal ll_src_rdy_in indica as amostras válidas no quadro de dados a ser escrito na FIFO TX. Finalmente, o ll_dst_rdy_out sinaliza quando a FIFO síncrona está pronta para aceitar dados no barramento ll_data_in. A interface LocalLink da FIFO disponibiliza também outros sinais para o gerenciamento da FIFO que são utilizados no módulo de comunicação. Para obter mais detalhes sobre a interface LocalLink, veja o Apêndice A.1.

5.2.4 Pilha de Protocolos

Para fornecer suporte ao transporte de dados na comunicação por pacotes e possibilitar a distinção entre os tipos de pacotes de monitoração, controle e fluxo de sinais foi desenvolvido, neste trabalho, um protocolo chamado MCP (*Monitoring and Control Protocol*). O protocolo MCP foi projetado para ser transportado dentro do campo de carga útil do protocolo UDP (*User Datagram Protocol*) ou do protocolo TCP (*Transmission Control Protocol*). O protocolo MCP é formado por um cabeçalho de 16 bits que contém o campo tipo (indica o tipo de pacote) e um campo parâmetros/canais que é definido de acordo com o campo tipo. Além disso, possui o campo de carga útil que varia de 0 a 1314 bytes. O comprimento máximo do pacote MCP é de 1316 bytes e o comprimento mínimo é equivalente ao tamanho do cabeçalho. A Figura 5.7 mostra a estrutura do protocolo MCP.

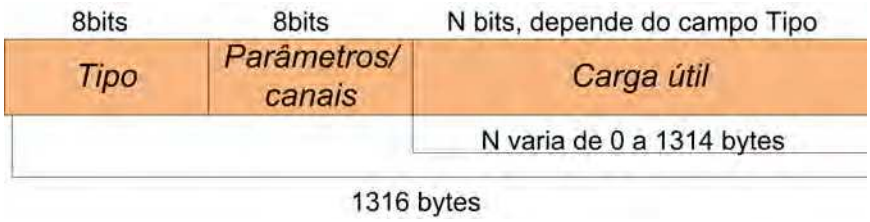

Fig. 5.7: Estrutura do protocolo MCP.

O cabeçalho do protocolo MCP identifica o tipo do pacote. Atualmente, o protocolo MCP define quatro tipos de pacotes distintos (controle, monitoração, fluxo de sinais e requisição), que são utilizados para o transporte de dados entre os equipamentos do laboratório remoto. Na rede, cada experimento é visto como uma aplicação distinta. Porém, não é necessário definir um tipo no protocolo MCP para cada experimento uma vez que o experimento já é identificado através do par endereço IP e porta UDP.

A monitoração do experimento é realizada a partir da plataforma de *software* cliente-servidor (*middleware*). A monitoração é realizada através da leitura e abertura dos pacotes do tipo monitoração gerados pela plataforma REDLART. O pacote de monitoração contém a marcação 0 no campo tipo e o campo parâmetros/canais indica quantos parâmetros de monitoração serão definidos no campo de carga útil. Em cada pacote de monitoração, o campo parâmetros/canais define que a carga útil pode conter até 219 parâmetros do experimento. A carga útil do pacote é composta por uma sequência de pares de endereço e valor que especificam qual é o parâmetro do experimento e o valor do mesmo. O campo endereço é representado por 16 bits e o campo valor por 32 bits. A Figura 5.8 mostra como o pacote de monitoração está estruturado.

O controle do experimento, assim como a monitoração, é realizada a partir da plataforma de *software* cliente-servidor (*middleware*). O controle é feito pelo empacotamento de comandos e geração de pacotes do tipo controle para a plataforma REDLART. A estrutura do pacote de controle é semelhante a estrutura do pacote de monitoração. Para indicar que o pacote é de controle, o *middleware* faz a marcação do campo tipo para 0. O campo parâmetros/canais indica quantos parâmetros do experimento estão na carga útil do pacote. Desta forma, o conteúdo da carga útil deve indicar os endereços

Fig. 5.8: Estrutura do pacote de monitoração definido no protocolo MCP.

dos parâmetros a serem modificados com seus respectivos valores. Convencionou-se que pacotes de controle também são definidos pelo valor 0 no campo tipo. Não há confusão em definir tanto os pacotes de monitoração quanto os de controle como sendo do tipo 0. A distinção entre os dois tipos se faz simplesmente pela plataforma que recebe o pacote. Se um pacote do tipo 0 é recebido pelo middleware, o mesmo sempre o interpretará como sendo de monitoração. Por outro lado, um pacote tipo 0 recebido pela REDLART sempre é interpretado como sendo de controle.

O protocolo MCP também pode ser utilizado para gerar pacotes do tipo fluxos de sinais (*streaming*) para interconectar várias plataformas na realização de um experimento distribuído ou para geração, visualização e reprodução remota de fluxos de sinais de tempo real. É importante salientar que os fluxos de sinais multiplexados em um mesmo pacote devem estar na mesma taxa de amostragem. Neste caso, o campo tipo é marcado para 1 e o campo parâmetros/canais indica o número de fluxos de sinais multiplexados na comunicação. Futuramente, comunicações com múltiplas taxas serão distinguidas através de portas UDPs distintas. Atualmente a carga útil do pacote está organizada para transportar amostras de sinais de 16 bits em múltiplos canais. A Figura 5.9 ilustra a organização do pacote para 3 sinais que são transportados pelo protocolo MCP.

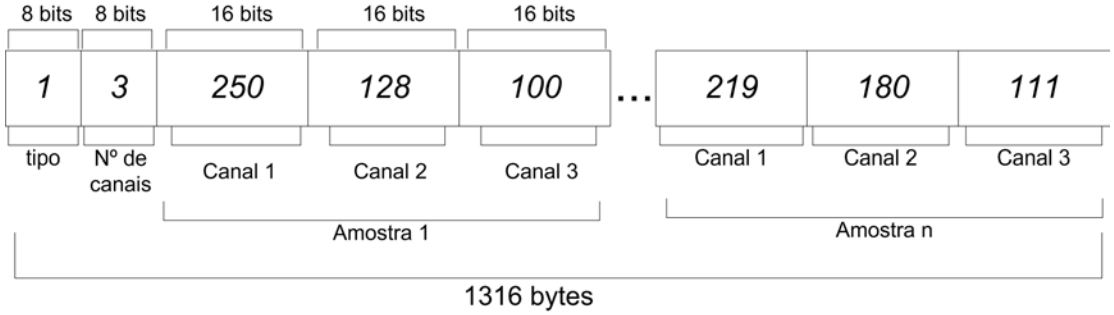

Fig. 5.9: Estrutura do pacote de sinais definido no protocolo MCP.

A plataforma de *software* cliente-servidor pode enviar também pacotes de requisição para a plataforma de *hardware* REDLART. A estrutura do pacote de requisição contém apenas a definição do campo tipo. Para indicar que o pacote é de requisição, o *middleware* faz a marcação do campo tipo para 2. O *middleware* envia pacotes de requisição para solicitar à plataforma REDLART que envie

pacotes de monitoração nos quais devem constar os valores atualizados dos parâmetros do experimento.

Atualmente o protocolo MCP é inserido no campo de carga útil do protocolo UDP. O protocolo UDP foi escolhido porque a parte pesada do tráfego é gerada por pacotes de fluxo de sinais que, por serem de tempo real, não há necessidade de serem orientados a conexão. De fato, neste cenário, confirmações e pedidos de retransmissão podem ser pior do que perdas de pacotes, uma vez que os atrasos podem inviabilizar a aplicação. Além disso, o experimento de laboratório remoto, em princípio, será executado utilizando a infraestrutura da rede KyaTera. Resumindo, neste trabalho, toda a comunicação da aplicação *WebLab* utiliza a pilha de protocolos MCP/UDP/IP/Ethernet conforme mostrado na Figura 5.10.

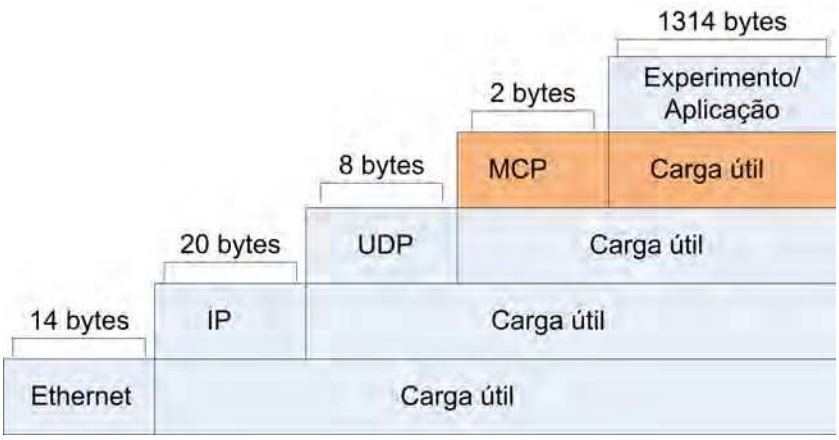

Fig. 5.10: Pilha de protocolos utilizada para fazer a comunicação ente a plataforma REDLART e o *middleware*.

No protocolo MCP os tipos de pacotes pré-definidos, tais como o pacote de controle e o pacote de monitoração, têm a restrição de transmitir até 219 parâmetros do experimento por pacote. Essa restrição se deve ao tamanho do pacote MCP que, por sua vez, está restrito ao tamanho do protocolo Ethernet. O tamanho máximo do pacote Ethernet, de maneira geral, está definido em 1500 bytes. Na implementação desse protocolo na plataforma REDLART foi especificado que o tamanho máximo é de 1358 bytes. Desses 1358 bytes, 42 bytes são de cabeçalho e 1316 bytes são de carga útil. Os 42 bytes do cabeçalho estão estruturados da seguinte maneira: 8 bytes do cabeçalho UDP, 20 bytes do cabeçalho IP e 14 bytes do cabeçalho Ethernet. Os 1316 bytes são utilizados pelo protocolo MCP, conforme mostrado anteriormente.

O protocolo UDP possui a característica de ser não-orientado a conexão, consequentemente não há garantia de entrega de pacotes. O protocolo UDP está definido na RFC 768 [50]. Ele permite que a aplicação de origem gere um pacote encapsulado no protocolo IP para ser enviado para a aplicação de destino. Em aplicações com comunicação bidirecional de fluxo de dados em tempo real como fluxo de sinais de áudio e vídeo, por exemplo, é mais importante ter baixo atraso do que a garantia de entrega. O protocolo IP (*Internet Protocol*) está definido na RFC 791 [51] e trabalha com o conceito

de datagramas, ou seja, pacotes de dados são transmitidos na rede através de um serviço sem conexão. O protocolo IP, por sua vez, é encapsulado no campo de carga útil do protocolo Ethernet. O protocolo Ethernet [48] fornece uma interface para a camada de rede, acessa o meio físico e faz o controle de transmissão de dados.

Para permitir que a plataforma REDLART seja automaticamente identificada pela rede, também é necessário utilizar a pilha de protocolos ARP/Ethernet. O protocolo ARP (*Address Resolution Protocol*) está definido na RFC 826 [52] e é utilizado para encontrar o endereço físico da camada de enlace de rede (Ethernet, por exemplo) a partir do endereço da camada de rede (endereço IP). Esse protocolo é gerado pela plataforma REDLART para indicar aos diversos componentes de rede, principalmente a roteadores e *switches*, qual o endereço físico (endereço MAC) correspondente ao endereço IP atribuído à plataforma.

5.2.5 Detalhamento da Plataforma

Os módulos de interfaces de dados e de comunicação disponibilizam uma infraestrutura de controladores de dispositivos (*drivers*) que propiciam a prototipagem rápida de novos experimentos. O experimento deverá ser desenvolvido no módulo de processamento, a partir do qual se pode ter acesso a todos os recursos disponibilizados pelos módulos de interface de dados e de comunicação. Para compreender como esse módulos se comunicam e como as informações contidas nos módulos de interfaces de dados e de comunicação são disponibilizadas para o módulo de processamento, é importante entender como está estruturada a arquitetura da plataforma REDLART.

A Figura 5.11 mostra uma visão detalhada dos módulos de interface de dados e de comunicação. O módulo de interface de dados está estruturado nos sub-módulos de Captura *Streaming* (áudio), FIFO PCM IN, FIFO PCM OUT e Reproduz *Streaming* (áudio). O módulo de comunicação, por sua vez, é descrito pelos sub-módulos FIFO Desempacotador, Desempacotador, Controle de escrita, RAM, Controle de leitura, Empacotador e FIFO Empacotador.

O módulo de interfaces de dados é detalhado nessa arquitetura considerando que o dispositivo de captura (conversão analógico para digital) e de reprodução (conversão digital para analógico) utilizado é o *codec* AC97 de áudio. O procedimento básico realizado por este módulo é baseado em FIFOs assíncronas, para mudança de domínio de *clock* e disponibilização do sinal em rajada. Tal procedimento se aplica a qualquer tipo de conversão AD e DA, além da ilustrada na Figura 5.11 para o *codec* AC97.

A arquitetura da plataforma REDLART está organizada em múltiplos domínios de *clock*. O domínio principal, onde é realizado o processamento, executa a um *clock* de 100 MHz. Apesar do *clock* ser baixo, é importante destacar que o processamento pode ser realizado em paralelo. Isso pode aumentar significativamente a capacidade de processamento do experimento. Devido a especificidades dos sinais a serem processados, cada qual é tipicamente capturado/reproduzido a uma taxa que irá definir o domínio de *clock* do sub-módulo específico de captura/reprodução. Para o controlador do *codec* de áudio AC97 esse domínio de *clock* deve ser de 24 MHz. A taxa de sinal é de apenas 48 kHz, porém o fator multiplicativo de 500 é necessário porque a interface com o *codec* é binária e

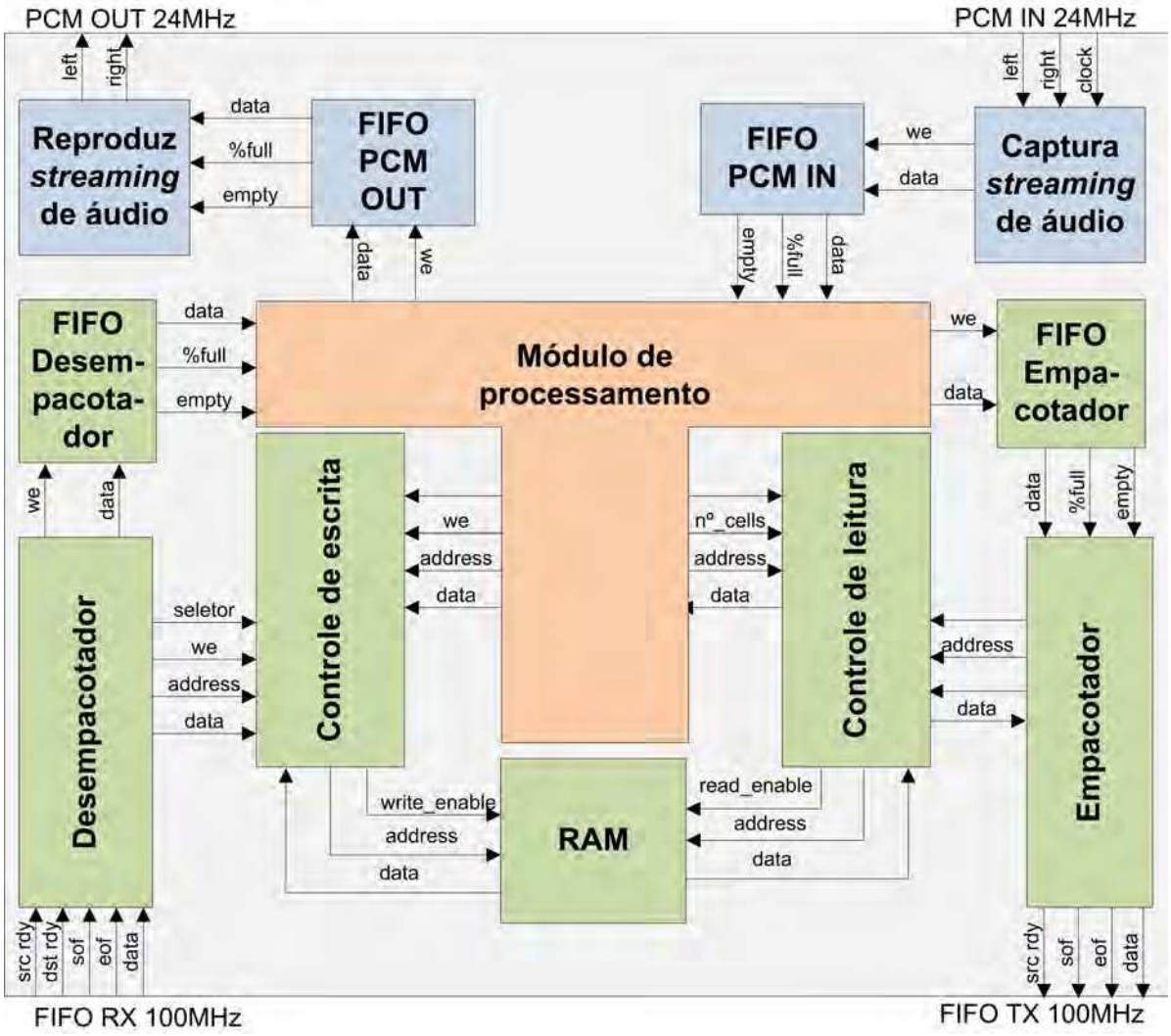

Fig. 5.11: Visão detalhada da plataforma REDLART.

serial. O módulo de interfaces de dados gerencia esses domínios utilizando FIFOs assíncronas que possibilitam a transferência de dados entre múltiplos domínios de *clock*.

O sub-módulo captura *streaming* de áudio obtém o fluxo de sinal de áudio do *codec* AC97 através dos canais de áudio esquerdo e direito com taxa de amostragem de 48kHz e quantização de 16 bits em ambos canais. O fluxo de sinal capturado é armazenado na FIFO PCM IN. A FIFO PCM IN é uma FIFO assíncrona composta por duas interfaces, uma específica para a escrita e outra específica para a leitura de fluxo de sinais. A interface de escrita da FIFO PCM IN armazena o fluxo de sinais obtido do sub-módulo captura *streaming* de áudio a um *clock* de 24 MHz. A interface de leitura dessa FIFO PCM IN disponibiliza para o módulo de processamento o sinal que será lido em rajada sempre que uma determinada ocupação da FIFO for alcançada.

De maneira análoga, a FIFO PCM OUT é assíncrona e também possui uma interface para escrita

e outra interface para a leitura de fluxo de sinais. O módulo de processamento escreve na FIFO PCM OUT o sinal a ser reproduzido a um *clock* de 100 MHz que pode ser bem maior que o *clock* de leitura do sub-módulo de reprodução. Neste caso, há uma sinalização de controle, disponibilizada pela própria FIFO, que indica para o módulo de processamento que a FIFO está cheia e não pode receber mais dados. Quando a ocupação baixar de um determinado nível, a FIFO sinaliza que está pronta para receber mais dados. De forma análoga, na captura, onde ocorre o processo inverso, há uma sinalização que indica para o módulo de processamento quando a FIFO PCM IN está vazia. Portanto, todo o processamento do experimento é realizado em rajada e a taxa média dos fluxos de sinal estão limitadas pelos dispositivos de captura/reprodução através de sinalizações de controle que indicam quais são as amostras válidas na leitura e quando se pode encaminhar as amostras para reprodução.

Como já foi mencionado anteriormente, o módulo de comunicação é responsável por enviar pacotes de monitoração/sinais e receber pacotes de controle/sinais. O sub-módulo empacotador é responsável por enviar pacotes de monitoração (parâmetros do experimento) a partir de parâmetros armazenados em uma RAM e enviar pacotes de sinais (fluxos de sinais de tempo real) a partir de amostras do sinal armazenadas na FIFO empacotador. Cada pacote de monitoração é marcado com o tipo 0 no protocolo MCP e pode conter até 219 parâmetros do experimento. Caso o experimento possua mais de 219 parâmetros, poderão ser utilizados quantos pacotes forem necessários para varrer todos os parâmetros disponíveis. Os parâmetros do experimento são armazenados pelo par endereço (indica a posição na memória) e valor (o valor armazenado na posição de memória correspondente). O campo de endereço aceita palavras de 16 bits e o de valor palavras de 32 bits. Há dois cenários onde acontece o envio de pacotes de monitoração. No primeiro, o envio ocorre quando o módulo de processamento indica que o empacotador deve enviar pacotes a cada 2 segundos (este período pode ser configurado). No segundo cenário, o empacotador é programado para enviar pacotes de monitoração apenas quando houver uma requisição (pacote MCP com o campo tipo marcado para 2) por parte do middleware. A definição de como deve proceder o envio de pacotes de monitoração é feita pelo módulo de processamento através de um registrador específico.

Os pacotes de sinais são enviados com a marcação do tipo 1 no protocolo MCP. Para esse tipo de pacote, há um campo específico no protocolo MCP que indica a quantidade de canais de áudio utilizada. Por exemplo, um fluxo de áudio do módulo de interface de dados é formado por dois canais de 16 bits. No módulo captura *streaming* de áudio, o fluxo de sinal é obtido através da FIFO PCM IN e encaminhado, através de um controle de fluxo apropriado, para o módulo de processamento. No módulo de processamento, algum processamento pode ser feito nos sinais, tais como, síntese, reverberação, aplicação de filtros digitais, e depois, o sinal resultante pode ser distribuído na rede através da FIFO Empacotador. Quando houver dados disponíveis e suficientes para gerar pacotes, o empacotador retira os dados da FIFO empacotador, monta o pacote e o transmite para outro nó do experimento.

Para o empacotador enviar pacotes de monitoração, é necessário o acesso aos dados dos parâmetros do experimento que estão armazenados na RAM. O controle de acesso a leitura na RAM é feito pelo módulo controle de leitura. Os módulos aptos à leitura da RAM são os sub-módulos de processamento e empacotador. Existe uma janela de tempo definida no sub-módulo de controle de leitura que faz o controle de acesso à RAM. Em linhas gerais, a cada 1 segundo é feita a troca de per-

missão de acesso a RAM. A janela de tempo de 1 segundo possui no *clock* de operação 100 milhões de ciclos. Destes, a geração de pacotes de monitoração não deve gastar mais do que 10000 ciclos, o que corresponde a apenas 0,1 milissegundo. Isso significa que na maior parte do tempo a RAM estará disponível para o módulo de processamento.

O sub-módulo RAM é composto de uma *Dual Port* RAM com profundidade da memória (número de endereços) que pode variar até 64k. O tamanho das palavras armazenadas é de 32 bits. Esse sub-módulo é responsável por armazenar os parâmetros do experimento. A RAM possui duas interfaces distintas, uma para escrita e outra para a leitura. Essas interfaces são gerenciáveis e permitem definir qual a ordem de execução das operações de escrita e de leitura. Na prova de conceito, a ordem de execução dessas operações foi definida como leitura antes de escrita. Os módulos aptos a fazerem a leitura e a escrita são o controle de leitura e o controle de escrita, respectivamente.

O sub-módulo desempacotador recebe os pacotes da interface LocalLink da FIFO Ethernet RX. Ao receber um pacote, o desempacotador verifica no cabeçalho UDP se a porta de destino especificada é compatível com a porta designada para o experimento. Caso seja diferente, o desempacotador descarta o pacote. Em seguida, verifica o cabeçalho MCP. Caso o protocolo MCP não seja identificado, o pacote também é descartado. O primeiro campo do cabeçalho MCP indica o tipo de pacote. Se o pacote for do tipo sinal, é feita a concatenação dos *bytes* da carga útil de maneira que formem palavras de 16 *bits* por amostra e por canal. Veja mais detalhes na Seção 5.2.4. Essas palavras são armazenadas na FIFO Desempacotador e ficam disponíveis para o módulo de processamento. No módulo de processamento, os dados devem ser tratados de acordo com a configuração do experimento. Por exemplo, o fluxo de sinal pode ser proveniente de um processamento prévio, feito por outro nó do experimento, devendo então ser processado no nó atual e, talvez, redistribuído na rede. Outra possibilidade é realizar também a reprodução/visualização local do sinal.

Outro tipo de pacote que pode ser tratado pelo desempacotador é o de controle. Esse pacote indica que houve alterações nos parâmetros do experimento por parte do usuário e, portanto, é necessário atualizar os dados dos parâmetros disponíveis na RAM. Para o desempacotador poder escrever na RAM é necessário ter a permissão de escrita fornecida pelo sub-módulo controle de escrita. O controle de escrita gerencia o acesso de escrita na RAM. Ele fornece prioridade de escrita para o Desempacotador. Podem escrever na RAM o módulo de processamento e o módulo de comunicação, representado pelo sub-módulo desempacotador, de maneira não simultânea. Quando chega um pacote de controle, automaticamente a permissão de escrita passa para o Desempacotador. A janela de tempo de permissão de escrita, disponível para o módulo de processamento é adaptativa e depende da frequência de chegada dos pacotes de controle.

O módulo de processamento deve ser projetado pelo aluno ou pesquisador. Isso demonstra a flexibilidade da plataforma REDLART. O desenvolvedor pode utilizar todos os módulos disponíveis da plataforma, beneficiando-se da infraestrutura que a mesma oferece. O desenvolvedor não precisa se preocupar com os detalhes de implementação de *drivers*, de gerenciamento de *clock* ou de gerenciamento de memória. O desenvolvedor precisa conhecer apenas as interfaces disponibilizadas por cada módulo. O trabalho do pesquisador ou do aluno é simplificado, porque é possível desenvolver um novo experimento apenas substituindo o módulo de processamento da plataforma REDLART. Além

disso, o desenvolvimento do módulo de processamento pode ser implementado a partir da ferramenta *System Generator* da Xilinx, integrada ao ambiente Matlab/Simulink da MathWorks, o que acelera o processo de desenvolvimento.

5.3 Implementação

Após concluir as fases de especificação e modelagem, a fase seguinte consiste no desenvolvimento da plataforma REDLART e implementação no *chip* FPGA. O fluxo de desenvolvimento utilizado nesse trabalho segue a forma de desenvolvimento e implementação descrito na Seção 3.2. O fluxo de desenvolvimento é constituído pelas etapas de modelagem do sistema, especificação, verificação, implementação e depuração do sistema. Esse padrão de desenvolvimento é aplicado através da ferramenta *Project Navigator* - ISE (*Integrated Software Environment*) da Xilinx. Essa ferramenta é utilizada durante todas as fases do fluxo de desenvolvimento [53]. A versão 7.1i foi empregada para o desenvolvimento da plataforma. Além dessa ferramenta foram utilizadas outras ferramentas, tais como: *System Generator* versão 7.0 da Xilinx [9] utilizada durante a fase de modelagem do sistema, *Impact* versão 7.1i da Xilinx [54] usada na fase de implementação, *ChipScope* versão 7.1i da Xilinx [55] usada na fase de depuração da plataforma e o ambiente de desenvolvimento *Matlab/Simulink* versão 7.0 da MathWorks [4] integrado ao *System Generator*.

O *System Generator* é uma ferramenta de linguagem de mais alto nível que oferece um ambiente de desenvolvimento para modelagem de sistemas. Ele possibilita ao desenvolvedor uma visão sistêmica e conceitual do projeto que se pretende desenvolver. A partir do projeto modelado, é possível gerar código HDL eficiente para implementação em *hardware*. O *System Generator* não visa substituir o VHDL ou Verilog, mas ele torna possível focar outras dificuldades de projeto que não a linguagem, obtendo-se um ganho de produtividade bastante significativo [9], [34].

A ferramenta *Impact* permite aos desenvolvedores executar facilmente a programação e a configuração de dispositivos FPGAs através de uma interface gráfica de usuário [54]. A complexidade dos projetos envolvendo FPGAs tem aumentado consideravelmente. A ferramenta *ChipScope* é utilizada para reduzir a dificuldade em se fazer a depuração de um projeto de *hardware*. Essa ferramenta minimiza a quantidade de tempo necessária para se fazer a depuração de tal projeto, através de um conjunto de ferramentas que possibilitam visualizar os sinais envolvidos, através de uma interface gráfica [55].

Na implementação da plataforma REDLART, as fases de modelagem do sistema e de especificação foram realizadas utilizando subsistemas desenvolvidos em *System Generator* e VHDL conforme mostra a Figura 5.12. A integração desses subsistemas também foi realizada em VHDL, utilizando o ambiente ISE. Os principais módulos (interface de dados, comunicação e processamento) foram implementados em *System Generator*, que é representado pelo componente processamento da Figura 5.12. A Figura 5.13 mostra, no esquemático do modelo, como esses componentes foram desenvolvidos no *System Generator*. Os controladores de dispositivos utilizados pelos módulos de interfaces de dados e de comunicação foram implementados em VHDL nos demais componentes dessa mesma figura.

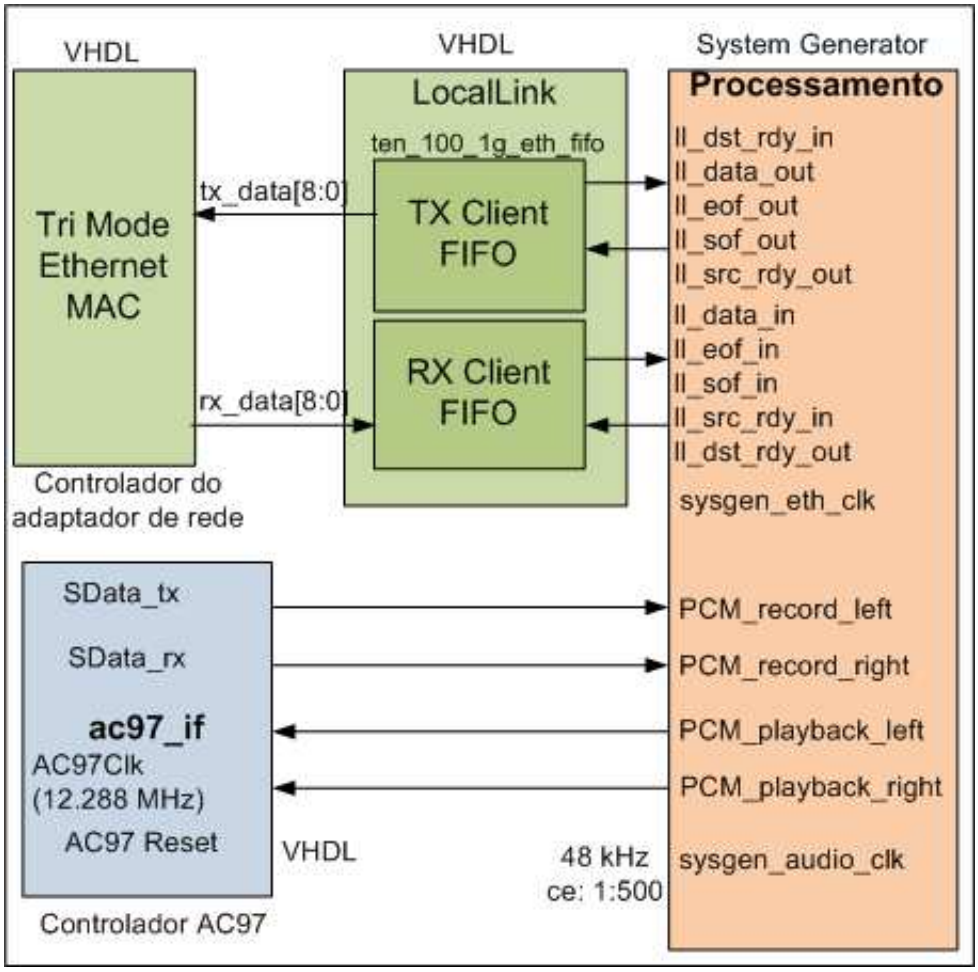

Fig. 5.12: Composição do componente VHDL de mais alta hierarquia utilizado para integrar netlists de componentes desenvolvidos em *System Generator* e códigos VHDL para controladores de dispositivos.

A integração do componente processamento no esquemático do projeto VHDL é feita a partir do *netlist* que o próprio *System Generator* gera automaticamente. A fase de geração do binário a partir dos *netlists* dos subsistemas (tradução, mapeamento, roteamento) é então acionada no próprio ambiente ISE.

Como mencionado, ao término das fases de modelagem de sistema e de especificação, o *netlist* e os códigos HDLs gerados foram integrados na ferramenta ISE para prosseguir o fluxo de projeto de desenvolvimento. A fase de verificação permitiu checar se as funcionalidades modeladas estavam funcionando de maneira apropriada. A fase de implementação permitiu descrever o projeto utilizando os componentes definidos no kit de desenvolvimento. Além disso, foi possível verificar se o roteamento físico da interconexão entre os blocos lógicos não estava inconsistente. A fase de *download* consiste em transferir o arquivo *bitstream* gerado para o *chip* FPGA. Essa fase utiliza a ferramenta

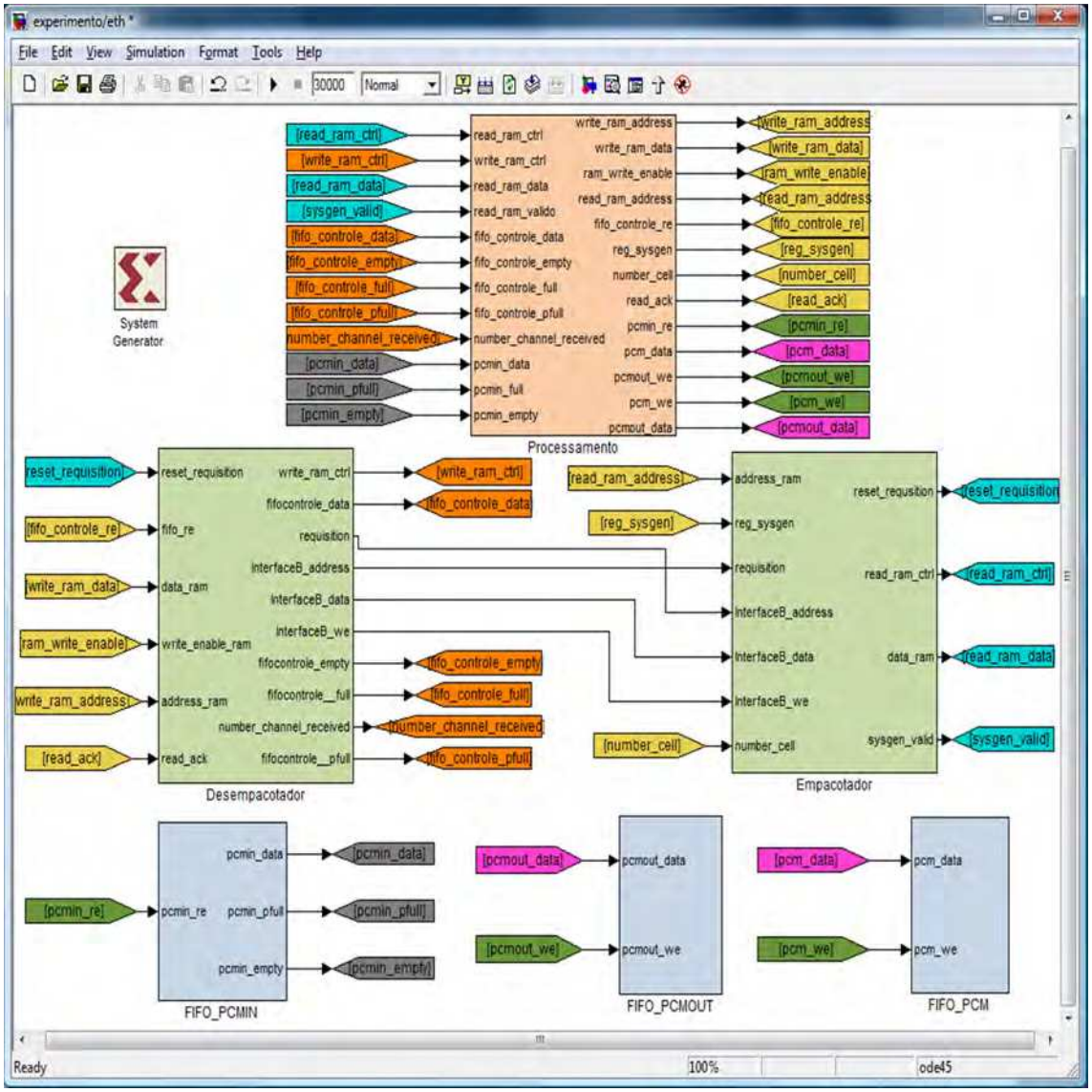

Fig. 5.13: Top-level da plataforma REDLART utilizando o *System Generator*.

Impact para executar tal operação. Após o *download* da plataforma REDLART para o *chip* FPGA é possível fazer a depuração da plataforma utilizando a ferramenta *ChipScope*.

Capítulo 6

Validação da Plataforma

Este capítulo descreve os testes realizados para validar a plataforma REDLART. O objetivo é analisar a viabilidade de se padronizar uma plataforma experimental em *hardware* reconfigurável para que se possam desenvolver experimentos acessíveis de um local remoto e do próprio local do laboratório. Embora o processamento realizado nos testes seja simples, os experimentos foram definidos para ilustrar as funcionalidades da plataforma no acesso remoto e na realização de um experimento com suas partes distribuídas em uma rede.

6.1 Configuração Sistêmica

A Figura 6.1 mostra a configuração sistêmica básica utilizada nas demonstrações. O objetivo é abordar questões sobre configuração, modo de acesso, comunicação e funcionalidades dos experimentos realizados. Como mostrado nessa figura, os principais elementos sistêmicos são a aplicação *WebLab*, o *middleware*, a infraestrutura de rede, o servidor de configuração e a plataforma REDLART.

A aplicação *WebLab* é responsável por estabelecer um canal de comunicação entre o experimento e o *middleware* e prover ao usuário uma interface gráfica amigável para que o mesmo possa interagir com o sistema. Para estabelecer esse canal de comunicação, a aplicação *WebLab* faz um pedido de conexão ao *middleware*. Esse pedido de conexão pode ser realizado quando o usuário acessa a aplicação *WebLab* e faz a solicitação de autenticação na aplicação. O pedido de autenticação é encaminhado para o *middleware* (servidor). O servidor tem a opção de rejeitar ou aceitar o pedido. Se o pedido for aceito, a conexão é estabelecida através do protocolo TCP e continua estável até acontecer um pedido de fechamento de conexão ou ocorrer congestionamento na rede. Após o estabelecimento da conexão inicia-se a troca de informações e dados entre o cliente e o servidor. Essa troca de informações e de dados é realizada através do módulo comunicação TCP/IP.

Após o estabelecimento da conexão entre o *middleware* e a aplicação *WebLab*, inicia-se o processo de troca de informações entre ambos. O servidor encaminha o fluxo de dados contendo informações sobre os parâmetros do experimento para que a aplicação *WebLab* possa fazer o tratamento de dados e a apresentação dos mesmos ao usuário. O módulo tratamento de dados é responsável por fazer a codificação e a decodificação dos dados e dos comandos gerados ou recebidos pelo experimento. Também

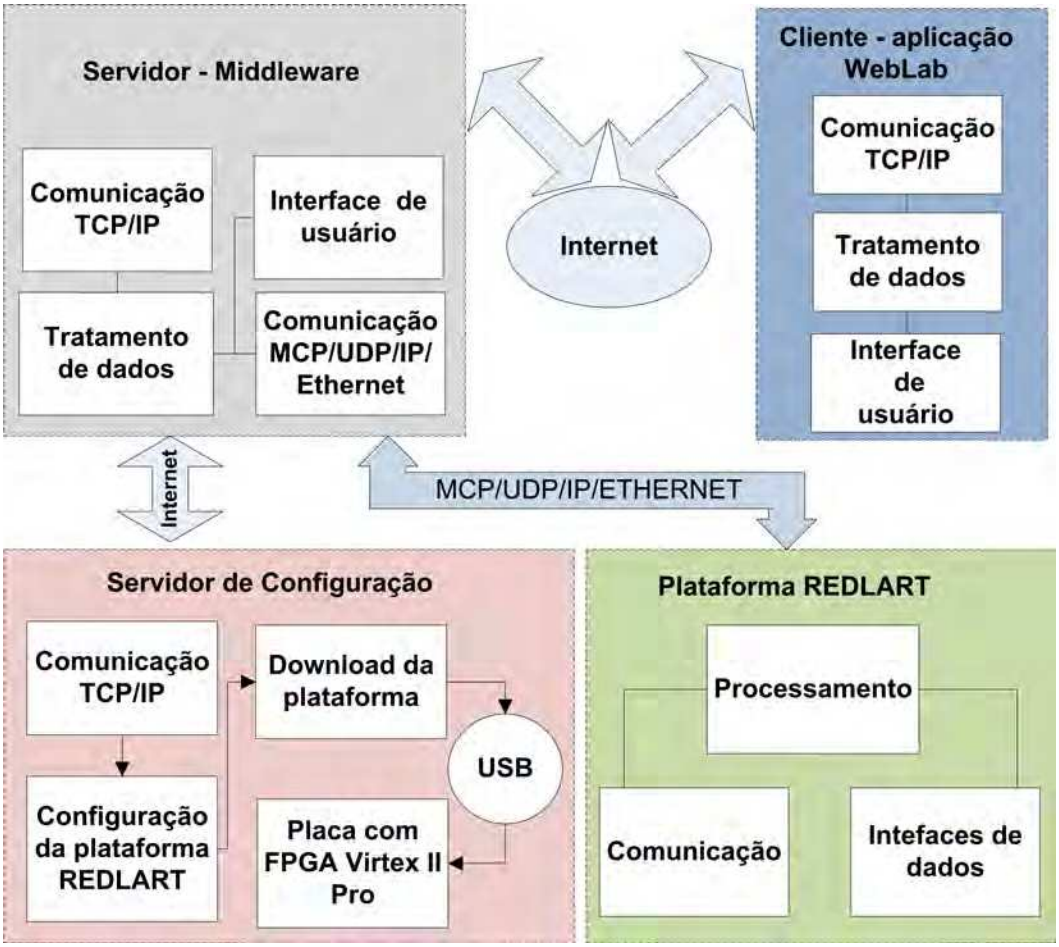

Fig. 6.1: Configuração sistêmica básica utilizada nas demonstrações.

é responsável por selecionar quais dados serão encaminhados para a plataforma REDLART e quais devem ser enviados para a interface do usuário. O módulo interface de usuário permite o acompanhamento das operações que estão ocorrendo no experimento, assim como facilita a visualização, permite o controle dos parâmetros e possibilita ao usuário fazer a análise dos resultados obtidos.

Quando o usuário solicita a alteração de parâmetros, alteração do modo de processamento do experimento, ou quando é necessário enviar fluxo de sinais para outro experimento, a aplicação *WebLab* encaminha essa requisição para o servidor. O servidor é responsável por estabelecer a comunicação com a plataforma REDLART via pilha de protocolo especificada na plataforma REDLART. O módulo comunicação UDP/MCP/IP/Ethernet estabelece tal comunicação. Esse módulo transporta informações relacionadas a comandos enviados pelo cliente, fluxos de sinais destinados à plataforma e dados retornados pela plataforma. Quando essas informações chegam à plataforma, elas são disponibilizadas para o módulo de processamento. O módulo de processamento tem a liberdade de decidir entre executar o processamento, encaminhar os dados recebidos para os módulos de comunicação ou de interfaces de dados, conforme definição do experimento.

O sistema possibilita que o usuário administrador carregue um novo experimento na plataforma de *hardware* (REDLART). Essas operações estão disponíveis no servidor de configuração para que o usuário administrador possa executá-las. As tarefas do servidor de configuração podem ser atribuídas também ao servidor *middleware*, mas não necessariamente o *middleware* que é responsável por carregar o experimento no *hardware*. Separar tal função do *middleware* possibilita utilizar um único servidor para gerenciar vários experimentos. Desta forma, o servidor de configuração está tipicamente em um local diferente de onde se encontra o *middleware*. Quando um novo experimento é inserido ou atualizado, o *middleware* sinaliza, através do módulo comunicação TCP/IP, ao servidor de configuração que o experimento deve ser configurado.

No módulo configuração da plataforma REDLART é definido um identificador para o experimento (endereço IP e porta UDP), a periodicidade de envio de pacotes de monitoração, os parâmetros do experimento que podem ser controlados, entre outros. Após essas configurações deve-se integrar o experimento à plataforma, recompilar e fazer o *download* para o *chip* FPGA. O *download* é feito através do programa Impact da Xilinx e descarregado na FPGA com o auxílio do cabo USB (*Universal Serial Bus*). O módulo *download* da plataforma é responsável por cumprir essa função. O módulo placa com FPGA Virtex II Pro refere-se à placa de *hardware* com o *chip* FPGA da Xilinx que executa o experimento propriamente dito. A plataforma REDLART será executada nesse *hardware* e fará uso dos módulos de interfaces de dados, de comunicação e de processamento, vistos anteriormente. Na plataforma de *hardware* não há limite de acessos simultâneos de usuário, porque a plataforma utiliza *chip* FPGA cuja característica é o paralelismo. Os módulos da plataforam REDLART foram projetados para explorar essa característica. No entanto, há limitação de acessos simultâneos proporcionada pela plataforma de *software* cliente-servidor. Atualmente, caso dois usuários acessem simultaneamente a plataforma de *software* cliente-servidor, um dos usuários poderá realizar o controle do experimento e o outro usuário poderá apenas monitorar o experimento.

6.2 Teste de Controle e Monitoração

O primeiro teste teve por base experimentos de processamento de sinais realizados no curso IE344B [3]. Esses experimentos consistem basicamente na captura de um sinal de áudio, no processamento, e na reprodução do sinal resultante. O processamento realizado consiste em filtros passa-baixa e passa-alta e filtro para gerar o efeito de reverberação. Nos testes também são realizados controles de *leds* e de volume de áudio. O objetivo desse experimento foi verificar o funcionamento dos módulos de interfaces de dados, de processamento, de comunicação e também testar o comportamento da plataforma. Na aplicação *WebLab*, o usuário pode definir o tipo de processamento e realizar os mesmos controles dos experimentos originais.

O experimento foi desenvolvido como mostrado na Figura 6.2 e funcionou da seguinte maneira. Um microfone pode ser conectado à plataforma através da entrada *mic in*. Simultaneamente, uma fonte de áudio estéreo, como um MP3 *player*, também pode ser conectada na entrada *line in*. O experimento foi realizado a partir de uma única plataforma denominada plataforma REDLART1. Essa plataforma foi responsável por realizar a captura fazendo a sobreposição de sinais de áudio das entradas *line in* e *mic in*, respectivamente. Os sinais resultantes da captura dos canais esquerdo e direito,

multiplexados, foram encaminhados para o módulo de processamento.

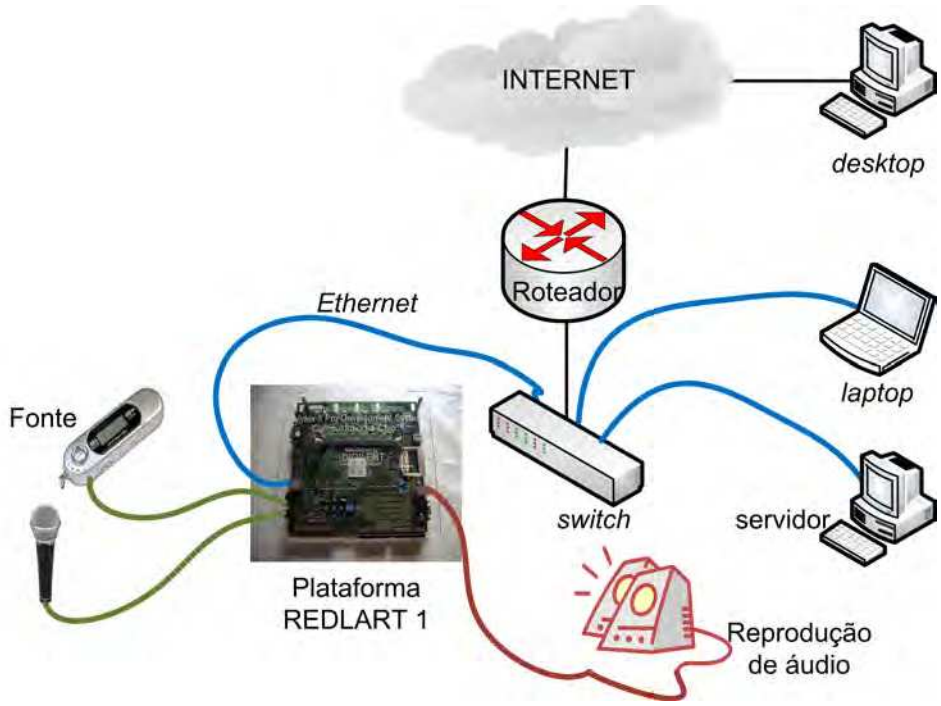

Fig. 6.2: Experimento acessado remotamente via rede Intranet.

O módulo de processamento recebeu o fluxo de sinais de áudio e o reproduziu de acordo com o tipo de processamento que o usuário escolheu. Paralelamente aos processos de captura, processamento e reprodução do áudio, o usuário pode controlar o acendimento de *leds*. Por exemplo, se o usuário acessasse a página *Web* do experimento e apertasse cinco vezes o botão de acionamento, o *led* zero da plataforma de *hardware* era acendido. Se ele apertasse o botão esquerdo quinze vezes, o *led* número um ficaria aceso e caso ele apertasse o botão *reset* todos os contadores de botões voltavam para o estado inicial e o *led* número dois acenderia. O número de vezes que um botão é pressionado é armazenado em uma RAM e é utilizado como parâmetros do experimento, assim como o volume do áudio e o tipo de processamento do áudio. Portanto, um usuário pode monitorar tais parâmetros e controlá-los remotamente.

O laboratório remoto foi acessado através da Intranet da rede da FEEC e o resultado foi satisfatório. Os fluxos de sinais de áudio foram capturados e reproduzidos a uma taxa média de 48 kHz, com *clock* de 24 MHz, e processados em um domínio de *clock* de 100 MHz. A plataforma foi configurada para enviar pacotes de monitoração a cada dois segundos para o servidor. Isso permitiu testar o módulo de comunicação. Praticamente, a variação no atraso de transmissão e recepção de pacotes permaneceu estável e não se perceberam atrasos significativos. Para verificar o envio e a recepção de pacotes, tanto na plataforma quanto no servidor, foi utilizado o analisador de protocolos *Wireshark* [56]. Constatou-se que o módulo de comunicação, especificamente o sub-módulo empacotador, funcionou apropriadamente, enviando pacotes de monitoração precisamente a cada dois segundos. O

comportamento da rede foi bastante estável e não houve perda de pacotes. Entretanto, nesta fase não se testou o desempenho do experimento sob condições adversas de tráfego. O objetivo foi verificar a funcionalidade dos módulos.

Nos testes realizados, verificou-se o pleno comportamento funcional da plataforma assim como o desempenho satisfatório sob condições normais de tráfego (sem congestionamento).

6.3 Teste de Distribuição

No segundo teste, manteve-se a especificação do experimento anterior, mas, no entanto, o arranjo do experimento e a infraestrutura da rede de computadores utilizada foi alterada. Nesse experimento, assim como no experimento anterior, o usuário pode escolher entre tipos de processamento (reverberação, filtro passa-baixas e filtro passa-altas). O usuário também pode controlar o acionamento de *leds* e o volume do áudio. Entretanto, o experimento foi configurado para utilizar duas plataformas REDLART e a infraestrutura de rede do projeto KyaTera. A Figura 6.3 mostra uma visão geral do experimento no cenário em que o mesmo é acessado através da rede KyaTera. Essa demonstração foi realizada no Workshop Kyatera 2008 [57].

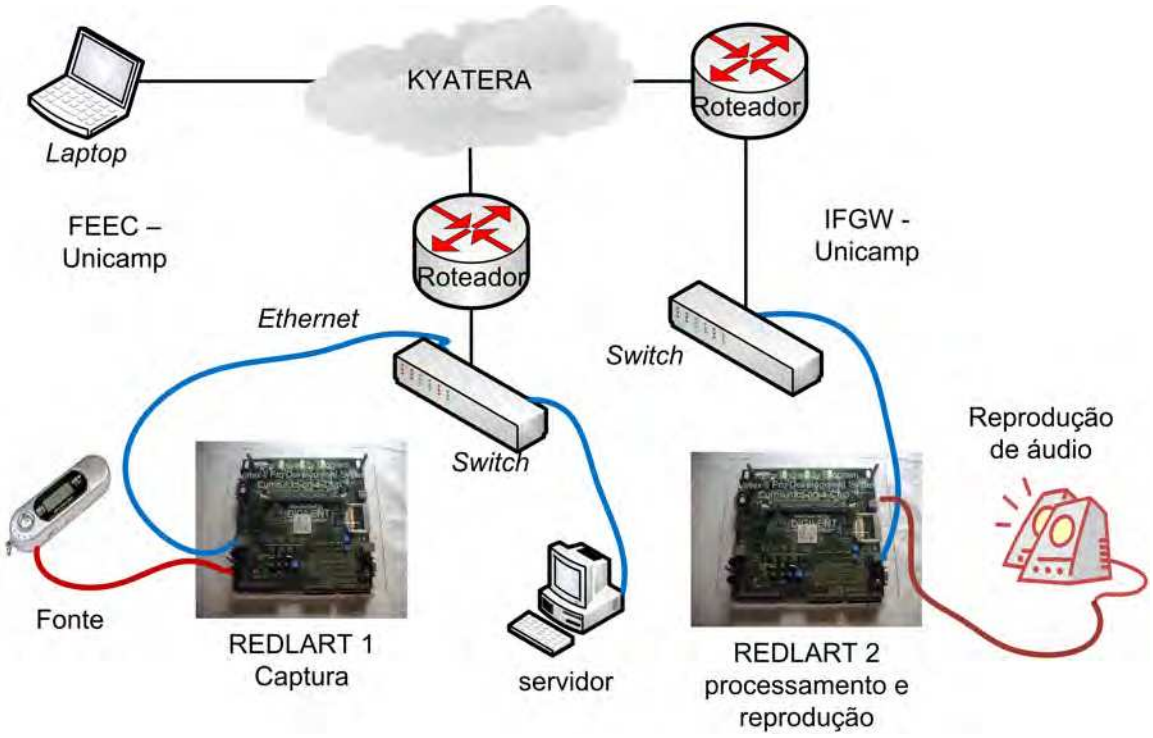

Fig. 6.3: Experimento acessado remotamente via rede KyaTera.

O experimento é constituído de dois nós. Um deles, denominado de REDLART1, captura o fluxo de sinal de áudio da entrada *line in* e o disponibiliza em rajada para o módulo de processamento. Uma

fonte de áudio estéreo, como um MP3 *player*, foi conectada na entrada *line in* na plataforma REDLART1. Os sinais resultantes dos canais esquerdo e direito podem então ser reproduzidos localmente no nó REDLART1. O usuário tem a opção de reproduzir remotamente o sinal resultante oriundo diretamente da fonte. Isso é possível porque o sinal resultante é enviado pela Internet em tempo real como um fluxo de sinais sobre pacotes MCP/UDP/IP/Ethernet.

O módulo de processamento da REDLART1 apenas encaminha o sinal para reprodução local, através do módulo interface de dados e para o nó REDLART2, através do módulo de comunicação e da rede KyaTera. A plataforma REDLART2 recebe os pacotes de sinal enviados pela plataforma REDLART1, desempacota-os e verifica se os pacotes recebidos são de controle ou de fluxo de sinais. O usuário pode escolher reproduzir o áudio processado ou reproduzi-lo na forma original sem processamento. Quando o usuário seleciona o tipo de processamento, o parâmetro do experimento que faz referência ao tipo de reprodução é alterado na memória. Cada parâmetro está associado ao par endereço e valor na memória, portanto cada parâmetro de controle do experimento é mapeado para uma posição na memória. O módulo de processamento faz a leitura desse parâmetro e caso a escolha do usuário seja a reverberação, o módulo de processamento realiza a reverberação do fluxo de sinais e encaminha o sinal resultante para o módulo de interfaces de dados executar a reprodução. Paralelamente aos processos de captura, transmissão, recepção, processamento e reprodução do áudio, o usuário pode controlar o acendimento de *leds* e o volume do áudio.

Para cada plataforma foi atribuído um endereço IP estático, uma porta UDP e um endereço MAC para que de fato pudessem participar da rede como um nó que envia e recebe pacotes de dados. No entanto, percebeu-se que o roteador periodicamente envia pacotes do protocolo ARP. O protocolo ARP realiza o mapeamento dinâmico entre endereços IP e endereços físicos. O roteador executa essa operação para certificar que os nós que pertencem a sua rede estão ativos e para manter atualizada a sua tabela ARP. Em geral, quando um nó recebe uma requisição do protocolo ARP, esse nó responde à requisição enviando um pacote ARP em *broadcast* onde constam os seus endereços IP e MAC. No entanto, inicialmente a plataforma não fornecia suporte ao protocolo ARP. Percebeu-se que os pacotes de fluxo de sinais e monitoração/controle destinados à plataforma não eram entregues pelo roteador. Isso porque no roteador não constava em sua tabela ARP a rota que o pacote deveria percorrer para chegar ao nó destinatário. Portanto, fez-se necessário implementar o protocolo ARP na plataforma REDLART. O protocolo ARP foi desenvolvido no módulo de comunicação e está integrado ao submódulo empacotador.

No primeiro teste, todos os atores da aplicação *WebLab* (usuário cliente, servidor do *middleware*, servidor de configuração e plataforma REDLART) estavam na mesma rede, sob o mesmo roteador. O servidor de configuração foi omitido na Figura 6.3 para não sobrecarregar a figura. No segundo teste, o experimento foi dividido em duas partes que são executadas em duas redes distintas, interconectadas pela rede KyaTera. A plataforma REDLART1 permaneceu na Faculdade de Engenharia Elétrica e Computação e a plataforma REDLART2 foi colocada no Instituto de Física, ambos da Unicamp. Como mostra a Figura 6.3 as plataformas foram conectadas através da rede KyaTera. Como visto nas seções anteriores, a rede KyaTera é uma rede que leva a fibra óptica até o laboratório (*fiber-to-the-lab*), portanto fez-se necessário usar uma classe especial de *switches* para que houvesse a conversão de sinal óptico em sinal elétrico e vice-versa.

O resultado do acesso ao experimento remoto foi satisfatório. Não ocorreu atraso significativo e a variação no atraso foi praticamente imperceptível, possibilitando que a plataforma operasse em tempo real. Na demonstração realizada no Workshop KyaTera [56] foram mantidas as mesmas configurações da demonstração anterior. O nó REDLART1 foi configurado para enviar pacotes de monitoração a cada dois segundos. Isso foi confirmado utilizando o *Wireshark* [56] para verificar o envio e a recepção de pacotes na plataforma e no servidor.

O módulo de comunicação, especificamente o sub-módulo desempacotador da REDLART2 trabalhou conforme projetado na recepção de pacotes. Não houve congestionamento na rede, portanto não aconteceu descarte de pacotes pela FIFO RX. Os pacotes foram reproduzidos a uma taxa média de 48 kHz, com *clock* de 24 MHz, e processados em um domínio de *clock* de 100 MHz. Os testes preliminares realizados demonstraram o funcionamento adequado da plataforma. Por exemplo, nos primeiros testes não se perceberam atrasos ou perdas de comandos no controle do experimento, o que é um bom indicativo de que a plataforma pode se tornar viável para realização de experimentos distribuídos e de tempo real.

6.4 Teste de Expansão

No teste de escala, analisa-se o funcionamento da plataforma e da aplicação *WebLab* quando se adiciona um terceiro nó na rede. Essa é uma aplicação onde se consegue aumentar a capacidade de processamento e possibilitar maior colaboração, tendo em vista que o experimento é dividido em três partes como mostra a Figura 6.4. Para executar o processamento, o usuário pode escolher entre aplicar filtro passa-baixa, filtro passa-alta ou reverberação. O usuário pode escolher também entre fazer a captura de sinal através do *codec* AC97 ou gerar ruído branco para ser utilizado como fonte de sinal.

O experimento é acessado através de uma aplicação *Web* (cliente) e do *middleware* (servidor). Para que o usuário tenha acesso ao experimento é necessário que o mesmo autentique-se no servidor. Após ser validado, o usuário pode ter acesso ao experimento. O experimento foi distribuído em três partes: captura ou geração do sinal fonte, processamento e reprodução do sinal resultante. No nó REDLART1 era feita a escolha da fonte de sinal. De acordo com a escolha do usuário, a plataforma REDLART1 gerava e transmitia os pacotes de fluxos de sinais para a plataforma REDLART2. A plataforma REDLART2, por sua vez, realizava o processamento do fluxo de sinais recebido da plataforma REDLART1, de acordo com o modo de processamento escolhido pelo usuário. Após o processamento, a plataforma REDLART2 encaminhava os pacotes de fluxo de sinais contendo os sinais resultantes para a plataforma REDLART3. A reprodução do fluxo de sinais será realizada na plataforma REDLART3 através do módulo de interfaces de dados.

O laboratório remoto foi acessado através da rede Internet convencional e o resultado foi satisfatório. Não ocorreu atraso significativo no envio e recebimento de pacotes. Na subrede KyaTera de 1Gbps e na rede da Unicamp de 100Mbps os atrasos e a variação de atrasos foram imperceptíveis. Nessa demonstração, mantiveram-se as mesmas configurações da demonstração anterior. Para cada plataforma REDLART foi atribuído um endereço IP estático, uma porta UDP para identificar

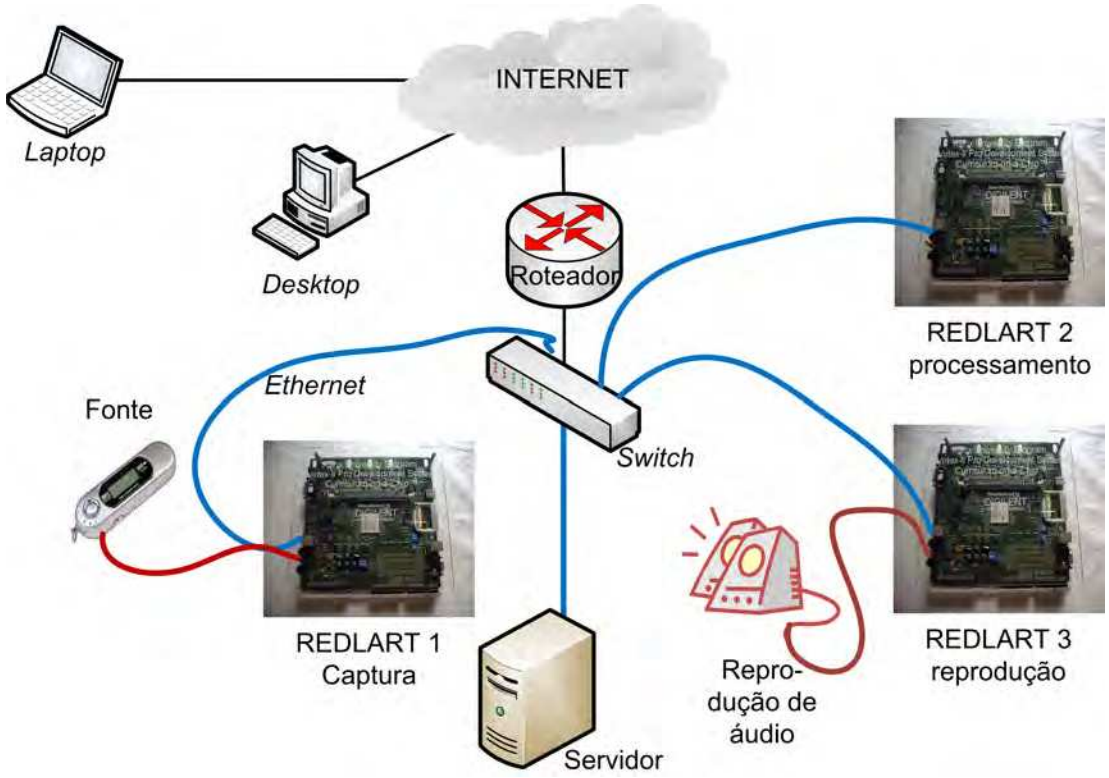

Fig. 6.4: Experimento acessado remotamente via rede Internet.

o experimento e um endereço físico MAC para identificar cada plataforma REDLART como um nó da rede. Os nós REDLART1, REDLART2 e REDLART3 foram configurados para enviar pacotes de monitoração a cada dois segundos. Também foi utilizado o analisador de pacotes *Wireshark* para verificar o envio e a recepção de pacotes nas três plataformas e no servidor. Através do *Wireshark* constatou-se que os pacotes de dados foram gerados conforme o previsto.

Portanto, foi possível comprovar o funcionamento dos módulos da plataforma REDLART e verificar a viabilidade da solução proposta para possibilitar aumentar a capacidade de processamento e de colaboração a partir da distribuição do experimento em rede. Também foi testada a integração do experimento no *middleware*. Vale ressaltar que os experimentos realizados são bastante objetivos e demonstram a viabilidade de uma plataforma em *hardware* reconfigurável integrada a uma plataforma de *software* que possibilite ao usuário desenvolver o seu próprio experimento e que tal experimento possa ser distribuído em nós de rede e disponibilizado para o acesso remoto.

Capítulo 7

Considerações Finais e Conclusão

A proposta deste trabalho foi desenvolver uma plataforma experimental em *hardware* reconfigurável para a prática de laboratório a distância com aplicações em ensino e pesquisa colaborativa. Apresentou-se o conceito de laboratório a distância, os requisitos de rede necessários para suportar essa aplicação e os trabalhos relacionados. A plataforma experimental proposta é baseada em *hardware* reconfigurável. Para compreender o funcionamento dos dispositivos semicondutores do tipo FPGA foram apresentados os conceitos e o fluxo de desenvolvimento do projeto para esse tipo de dispositivo. Também foi proposta uma arquitetura de sistema para desenvolver aplicações de laboratório a distância, visando a possibilidade de desenvolver experimentos remotos sem que o desenvolvedor se preocupe em realizar o gerenciamento de recursos e de usuários, facilitando, assim, a interação do usuário com a plataforma REDLART e permitindo o acesso remoto ao experimento através de uma interface acessível.

O intuito do laboratório a distância é idealmente alcançar um número muito maior de pessoas e oferecer educação de qualidade a um custo bem menor em infraestrutura e recursos humanos. Além disso, se bem projetado, o ensino a distância pode incentivar o aluno a ter uma participação bem mais ativa no processo de ensino e aprendizagem do que no curso presencial. No contexto de uma arquitetura de sistema que suporte a prática de laboratório a distância e a pesquisa colaborativa, foi proposta neste trabalho uma plataforma de *hardware* integrada a uma plataforma de *software* (servidor ou *middleware* da aplicação) que possibilitasse o desenvolvimento rápido de novos experimentos no escopo de aplicações *WebLabs*.

A plataforma de *hardware* é denominada neste trabalho de REDLART. A proposta dessa plataforma é possibilitar a criação de experimentos de sistemas digitais a partir de um conjunto de funcionalidades que isolam completamente a complexidade do *hardware* utilizado. Este trabalho apresenta a modelagem da plataforma REDLART, a forma de desenvolvimento e a implementação utilizada. A arquitetura da plataforma proposta possibilita disponibilizar o desenvolvimento e execução de experimentos remotos e distribuídos com controle e acesso pela *Web*.

Para verificar a viabilidade de desenvolvimento da arquitetura proposta foram apresentados três testes de validação que representam provas de conceitos de tal viabilidade. Também foram apresentados os resultados iniciais de experimentos de laboratório remoto desenvolvido utilizando a plataforma

REDLART. Os testes realizados até agora indicam a viabilidade de uma plataforma de *hardware* para o desenvolvimento de experimentos distribuídos na rede e em tempo real, seguindo o conceito de laboratório remoto pela *Web*. Avaliações subjetivas dos testes realizados constataram a funcionalidade, confiabilidade e flexibilidade da plataforma no contexto de provas de conceito.

Trabalhos futuros envolverão o desenvolvimento de novos *WebLabs* que serão incorporados a plataforma. O objetivo é que esses *WebLabs* sejam avaliados quanto ao desempenho, escalabilidade e usabilidade por grupos de alunos e de pesquisadores como usuários do sistema. As informações obtidas servirão como referência para o desenvolvimento de futuras versões da plataforma e dos experimentos.

Referências Bibliográficas

[1] M. G. Moore and G. Kearsley. *Educação a Distância - Uma visão Integrada*. Cengage Learning, 2008.

[2] M. Casini, D. Prattichizzo, and A. Vicino. *The Automatic Control Telelab: A User-Friendly Interface for Distance Learning*. Number 46. IEEE Transactions on Education, 2003.

[3] F. C. M. A. Cardoso e D. S. Arantes. *Curso IE344 B - Simulação, Co-simulação e Prototipagem de Sistemas de Comunicações Digitais*. `http://www.decom.fee.unicamp.br/~cardoso/ie344b.html`, 2007. acessado em 22/08/2008.

[4] Mathworks. *Matlab - Simulink*. `http://www.mathworks.com/products/simulink/`. acessado em 27/07/2008.

[5] L. M. Jiménez, R. Puerto, O. R. C. Fernández, and R. Ñeco. *RECOLAB: Laboratorio remoto de control utilizando matlab y simulink*, volume 2. Revista Iberoamericana de Automática e Informática Industrial, Abril 2005. 64-72.

[6] J. P. Oliver and F. Haim. *Lab at Home: Hardware Kits for a Digital Design Lab*, volume 52. IEEE TRANSACTIONS ON EDUCATION, 2008. 46-51.

[7] National Instruments. *LabVIEW*. `http://www.ni.com/labview/`. acessado em 14/11/2009.

[8] A. R. Marchezan, M. T. Chella, and E. C. Ferreira. *Laboratório Remoto Aplicado ao Ensino de Engenharia Eletrônica*. I Workshop de Ciência da Computação e Sistemas da Informação da Região Sul - WORKCOMPSUL, 2004.

[9] Xilinx Inc. *System Generator for DSP*. `http://www.xilinx.com/support/sw_manuals/sysgen_ug.pdf`. acessado em 23/08/2008.

[10] J. Huan and A. Ganz. *A New Model for Remote Laboratory Education Based on Next Generation Interactive Technologies*. ASEE New England Regional Conference, 2003.

[11] A. Najhlsk, Z. Nedic, and J. Machotkd. *Remote Laboratories Versus Virtual and Real Laboratories*. 33rd Annual Frontiers in Education (FIE'03), 2003.

[12] A. Bagnasco and A. M. Scapolla. *A Grid of Remote Laboratory for Teaching Electronics*. 2nd International LeGE-WG Workshop on e-Learning and Grid Technologies, 2003.

[13] M. Stegawski and R. Schaumann. *A New Virtual Instrumentation-Based Experimenting Environment for Undegraduate Laboratories with Application in Research and Manufacturing*. Number 47. IEEE Transaction on Instrumentation and Measurement, 1998.

[14] T. Schafer, J. M. Seigneur, and A. Donelly. *PEARL: A Generic Architecture for Live Experiments in a Remote Lab*. International Conference on Simulation and Multimedia in Engineering Education, 2003.

[15] B. Teitelman and T. Hanss. *Qos Requirements for Internet2*. Internet2 QoS Work Group Draft, 1998.

[16] *Projeto Kyatera - FAPESP*. `http://kyatera.incubadora.fapesp.br/portal`, 2001. acessado em 22/07/2008.

[17] *Rede experimental do Projeto Kyatera - FAPESP*. `http://kyatera.incubadora.fapesp.br/portal/projeto-kyatera/rede-kyatera/plataforma-optica/redes-experimentais/redes-experimentais`, 2005. acessado em 22/10/2008.

[18] *Projeto GIGA*. `http://www.rnp.br/pd/giga/`, 2002. acessado em 22/07/2008.

[19] U.S. Advanced Networking Consortium. *Network Internet2*. `http://www.internet2.edu`. acessado em 22/07/2008.

[20] *IEEE standard codes, formats, protocols, and common commands. For use with ANSI/IEEE Std 488.1-1987*. IEEE standard digital interface for programmable instrumentation, 1988.

[21] H. Shen, Z. Xu, B. Dalager, V. Kristiansen, Ø. Strøm, M. S. Shur, T. A. Fjeldly, J. Lü, and T. Ytterdal. *Conducting Laboratory Experiments over the Internet*, volume 42. IEEE Transaction on Education, 1999.

[22] T. Schafer, J. M. Seigneur, and A. Donelly. *PEARL: A Generic Architecture for Live Experiments in a Remote Lab*. International Conference on Simulation and Multimedia in Engineering Education, 117-122, 2003.

[23] *ILabs - Internet Access to real labs*. `http://icampus.mit.edu/iLabs/default.aspx`. acessado em 24/07/2008.

[24] *ISILab - Internet Shared Instrumentation Laboratory*. `http://isilab-esng.dibe.unige.it/`. acessado em 22/08/2008.

[25] *LaDABio - Laboratório para o Desenvolvimento e automação de bioprocesso*. `http://kyatera.incubadora.fapesp.br/portal/pesquisa/laboratorios/ladabio`. acessado em 25/07/2008.

[26] *ComLab - Laboratório KyaTera de Comunicações Digitais*. Faculdade de Engenharia Elétrica e Computação (FEEC). Departamento de Comunicações (Decom).

[27] A. A. Cruz. *Projeto e Implementação de um Framework para WebLabs baseado em Ajax e Padrões de Projeto*. Universidade Estadual de Campinas, 2007.

[28] F. Nebeker. *Signal Processing - The Emergence of a Discipline: 1948 to 1998*. New Jersey: IEEE History Center, 1988.

[29] A. Senda, M. S. S. Melo, M. T. M. Silva, M. Eisencraft, and M. A. A. Melo. *Aplicações de Pro-cessamento Digital de Sinais em Engenharia Elétrica*. XXXIII Congresso Brasileiro de Ensino de Engenharia, 2005.

[30] E. Boruchovitch and J. A. Bzuneck. *A Motivação do Aluno - Contribuições da Psicologia Contemporânea*. Editora Vozes, Petrópolis - RJ, 4ª edição edition, 2009.

[31] C. M. Maxfield. *The Design Warrior's guide to FPGAs - Devices, Tools and Flows*. Elsevier Science and Technology Books, 2004.

[32] A. S. Oliveira and F. S. Andrade. *Sistemas Embarcados - Hardware e Firmware na Prática*. Editora Érica Ltda, São Paulo - SP, 2006. 1ª Edição.

[33] Xilinx Inc. *Programmable Logic Design - Quick Start Handbook*. `http://www.xilinx.com/publications/products/cpld/logic_handbook.pdf`. acessado em 10/09/2008.

[34] F. L. Garcia. *Implementação de um Codificador LDPC para um Sistema de TV Digital usando Ferramentas de Prototipagem Rápida*. Universidade Estadual de Campinas, 2006.

[35] F. C. M. A. Cardoso e D. S Arantes. *Introdução à FPGA e ao Fluxo de Projeto*. `http://www.decom.fee.unicamp.br/~cardoso/ie344b.html`. acessado em 15/09/2009.

[36] IEEE Std 1364-2005. *IEEE Standard for Verilog Hardware Description Language*. 2006.

[37] IEEE Std 1076-2008. *IEEE Standard VHDL Language Reference Manual*. 2009.

[38] V. A. Pedroni. *Circuit Design with VHDL*. The MIT Press, 2004.

[39] ISO/IEC 42010 IEEE Std 1471-2000. *Systems and Software Engineering - Recommended Practice for Architectural Description of Software-Intensive Systems*. Edition 1 (technically identical to ANSI/IEEE Std 1471-2000), 2007.

[40] J. A. Zachman. *Zachman International Enterprise Architecture*. `http://www.zachmaninternational.com/index.php`. acessado em 16/10/2009.

[41] D. Whitehurst. *The AppFuse Primer*. Part of the Matt Raible Series. SourceBeat, LLC, 2007.

[42] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns CD - Elements of Reusable Object-Oriented Software*. Addison-Wesley professional computing series. Addison Wesley Longman, 1998.

[43] *PostgreSQL*. `http://www.postgresql.org`. acessado em 29/09/2009.

[44] J. C. Mitchell. *Concepts in Programming Languages*. Cambridge University Press, 1st edition edition, 2002.

[45] *Hibernate*. `https://www.hibernate.org`. acessado em 29/09/2009.

[46] *JavaServer Faces Technology*. `http://java.sun.com/javaee/javaserverfaces/`. acessado em 13/10/2009.

[47] Network Working Group. *RFC 1180 - A TCP/IP Tutorial*. Internet Engineering Task Force (IETF), 1991. `http://tools.ietf.org/html/rfc1180`, acessado em 11/10/2009.

[48] Technical Committee on Computer Communications of the IEEE Computer Society. *IEEE standards for local area networks: supplements to carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications*. IEEE Computer Society, 1987.

[49] Xilinx Inc. *Parameterizable LocalLink FIFO*. `http://www.xilinx.com/support/documentation/application_notes/xapp691.pdf`. acessado em 23/11/2009.

[50] J. Postel. *RFC 768 - User Datagram Protocol*. Internet Engineering Task Force (IETF), 1980. `http://tools.ietf.org/html/rfc768`, acessado em 26/11/2009.

[51] Information Sciences Institute University of Southern California. *RFC 791 - Internet Protocol*. Internet Engineering Task Force (IETF), 1981. `http://www.ietf.org/rfc/rfc0791.txt`, acessado em 23/07/2008.

[52] Network Working Group. *RFC 826 - An Ethernet Address Resolution Protocol or Converting Network Protocol Addresses*. Internet Engineering Task Force (IETF), 1982. `http://www.ietf.org/rfc/rfc0826.txt`, acessado em 21/11/2009.

[53] Xilinx Inc. *ISE Quick Start Tutorial*. `http://www.xilinx.com/itp/xilinx7/books/docs/qst/qst.pdf`. acessado em 23/11/2009.

[54] Xilinx Inc. *Impact*. `http://www.xilinx.com/products/design_tools/logic_design/design_entry/impact.htm`. acessado em 23/11/2009.

[55] Xilinx Inc. *ChipScope*. `http://www.xilinx.com/support/documentation/sw_manuals/chipscope_pro_sw_cores_10_1_ug029.pdf`. acessado em 23/11/2009.

[56] Wireshark Foundation. *Wireshark*. `http://www.wireshark.org/`. acessado em 23/11/2009.

[57] Fapesp projeto Tidia-KyaTera. *Workshop KyaTera 2008*. `http://kyatera.incubadora.fapesp.br/portal/eventos/workshop-kyatera-2008`, 2008. acessado em 23/11/2008.

Apêndice A

Apêndice

A.1 Kit de Desenvolvimento e Interface LocalLink

O kit de desenvolvimento utilizado para a implementação da plataforma REDLART e a documentação da interface LocalLink são apresentados nas próximas seções.

A.1.1 Especificação da interface LocalLink

A Figura A.1 mostra a especificação da FIFO interface LocalLink.

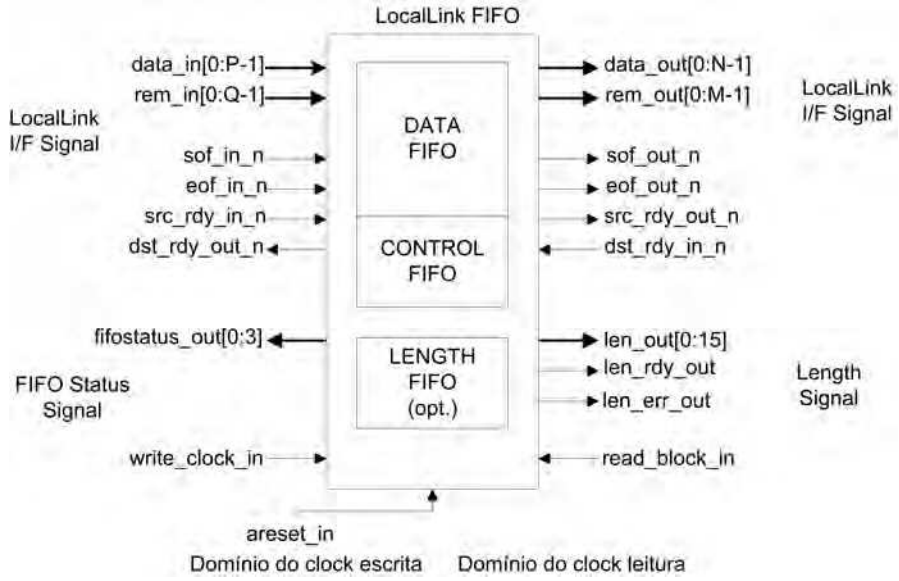

Fig. A.1: Interface LocalLink FIFO.

A Tabela A.1 mostra a especificação dos sinais definidos na interface da FIFO LocalLink [49]. Esses sinais são responsáveis por fazer o controle e o gerenciamento das FIFOs de transmissão e recepção.

Tab. A.1: Especificação da Interface LocalLink FIFO.

Sinal	Escopo	Direção	Descrição
areset_in	Upstream/Downstream	Input	Entrada *reset* assíncrona.
write_clock_in	Upstream	Input	*clock* de entrada para porta *write* da aplicação usuário *downstream*.
read_clock_in	Downstream	Input	*clock* de entrada para a porta *read* da aplicação usuário *downstream*.
data_in[0:P-1]	Upstream	Input	Dados de entrada. Os dados são de P bits. P = 8,16,32,64,128.
rem_in[0:Q-1]	Upstream	Input	Indica o número de bytes válidos no data_in quando o eof_in é declarado. Se P=8, Q deve ser; senão Q=log2(P/8).
sof_in_n	Upstream	Input	Indica o início da transferência de um *frame* do barramento data_in.
eof_in_n	Upstream	Input	Indica o fim da transferência de um *frame* do barramento data_in.
src_rdy_in_ n	Upstream	Input	Indica que o data_in é válido durante o ciclo atual.
dst_rdy_out_n	Upstream	Output	Indica que o LocalLink FIFO está pronto para aceitar os dados apresentados a ele através do barramento data_in no ciclo atual.
data_out[0:N-1]	Downstream	Output	Dados de saída. Os dados são de N bits. N = 8,16,32,64,128.
rem_out[0:M-1]	Downstream	Output	Indica o número de bytes válidos no data_out quando o eof_out_n é declarado. Se N=8, M deve ser 1; senão M=log2(N/8).
sof_out_n	Downstream	Output	Indica o início da transferência do *frame* no barramento data_out.
eof_out_n	Downstream	Output	Indica o fim da transferência do *frame* no barramento data_out.

Tab. A.1 Continuação

Sinal	Escopo	Direção	Descrição
src_rdy_out_n	Downstream	Output	Indica que o data_out é válido durante o ciclo atual.
dst_rdy_in_n	Downstream	Input	Indica que a aplicação usuário *downstream* está pronto para aceitar dados apresentados no barramento data_out do ciclo atual.
fifo status_out[0:3]		Output	Para indicar quando a FIFO está metade cheia, 3/4 cheia e assim por diante.
len_out[0:15]		Output	Tamanho do *frame* em bytes.
len_rdy_out		Output	Um pulso de sinal *active high* indica uma nova saída de tamanho do *frame* é válido no barramento len_out.
len_err_out		Output	Um sinal *active high indica* um *overflow* no tamanho da FIFO em uma implementação do bloco selectRAM.

A.1.2 Kit de Desenvolvimento

O kit de desenvolvimento XUP da Digilent foi utilizado na prova de conceito deste trabalho. A Figura A.2 ilustra esse kit. A Tabela A.2 mostra as principais características de recursos e periféricos contidos nesse kit de desenvolvimento.

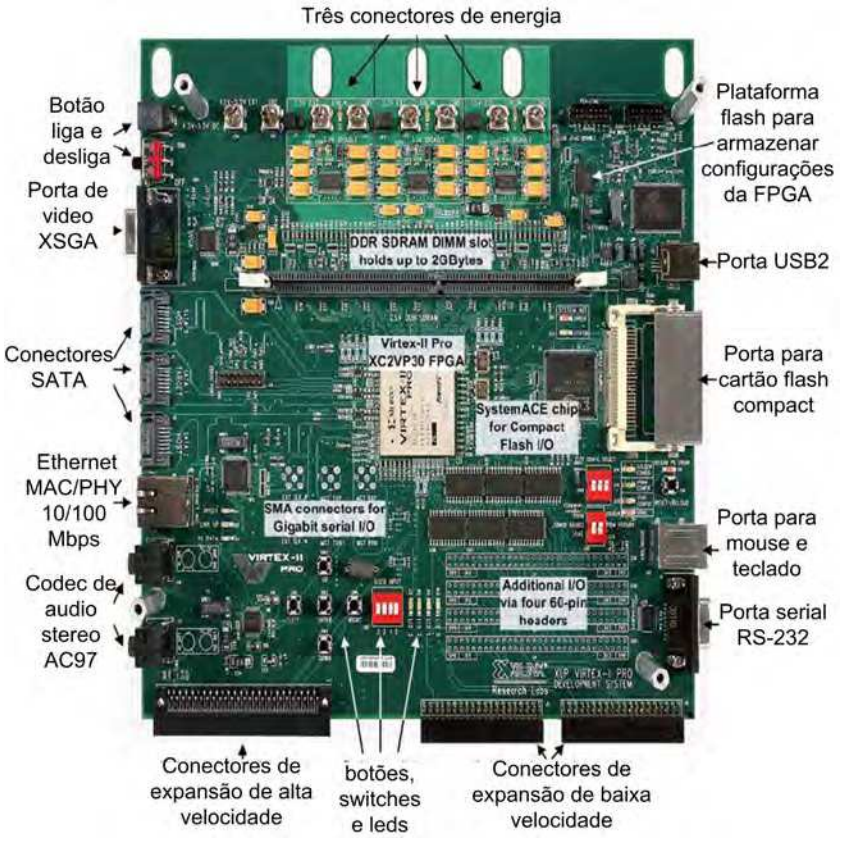

Fig. A.2: Kit de desenvolvimento XUP Virtex II Pro da Digilent.

Tab. A.2: *Principais características do kit de desenvolvimento XUP Virtex II Pro da Digilent.*

Características
- FPGA Virtex II Pro com PowerPC.
- Memória DDR SDRAM expansível até 2 GBytes.
- Controlador System ACE e conector CompactFlash tipo II.
- Porta USB.
- Dispositivo Phy Ethernet 10/100 Mbits/s.
- Porta serial RS-232 DB9.
- Porta serial PS-2.
- 4 *leds* conectados aos pinos de I/O da Virtex II Pro.
- 4 *switches* conectados aos pinos de I/O da Virtex II Pro.
- 5 botões conectados aos pinos de I/O da Virtex II Pro.
- 6 conectores de expansão interligados aos pinos de I/O da Virtex II Pro.
- *Codec* de áudio AC97 com amplificador de áudio, *line out* e saída para microfone e fone.
- *Line in* e entrada para microfone.
- Saída XSGA.
- 3 portas serial ATA, sendo 2 portas *host* e 1 porta *target*.
- *Clock* do sistema de 100MHz, *clock* SATA 75MHz.
- Previsão para definição de *clock* do usuário.
- Circuito de reinicialização do PowerPC.