Centro Universitário Positivo - UnicenP
Núcleo de Ciências Exatas e Tecnológicas – NCET
Engenharia da Computação
Cristiano Saldanha Carneiro

SISTEMA DE CONFIGURAÇÃO E MONITORAÇÃO REMOTA DE LÓGICA PROGRAMÁVEL ATRAVÉS DE PROTOCOLO INTERNET

CURITIBA
2005

Centro Universitário Positivo - UnicenP
Núcleo de Ciências Exatas e Tecnológicas – NCET
Engenharia da Computação
Cristiano Saldanha Carneiro

SISTEMA DE CONFIGURAÇÃO E MONITORAÇÃO REMOTA DE LÓGICA PROGRAMÁVEL ATRAVÉS DE PROTOCOLO INTERNET

Monografia apresentada à disciplina de Projeto Final, como requisito parcial à conclusão do Curso de Engenharia da Computação. Orientador: Prof. Valfredo Pilla Jr.

CURITIBA
2005

TERMO DE APROVAÇÃO

Cristiano Saldanha Carneiro

Sistema de Configuração e Monitoração Remota de Lógica Programável Através de Protocolo Internet

Monográfica aprovada como requisito parcial à conclusão do curso de Engenharia da Computação do Centro Universitário Positivo, pela seguinte banca examinadora:

Prof. Valfredo Pilla Junior (Orientador)

Prof. José Carlos da Cunha

Prof. Mauricio Perretto

Curitiba, 12 de Dezembro de 2005

AGRADECIMENTOS

Agradeço acima de tudo a Deus, que me deu forças e por muitas vezes me ajudou nesta difícil caminhada que foi minha graduação, e por fim a elaboração deste.

Agradeço também a minha esposa Márcia, sem seu apoio e compreensão este desafio jamais poderia ter sido vencido, e a minha família que muito me apoiou nesta jornada.

Gostaria de agradecer ao meu orientador, Prof. Valfredo Pilla Jr., que desde o inicio do desafio que seria este projeto, acreditou em mim e na minha capacidade de concluí-lo, mesmo quando nem eu acreditava. Seu apoio e dedicação foram decisivos para a conclusão do projeto.

Por fim agradeço aos demais professores, laboratoristas e colegas que de alguma forma me ajudaram na conclusão deste, em especial ao Prof. Mauricio Perretto, pela paciência em me explicar as coisas mais simples que me faltavam conhecer.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

API – (*Application Program Interface*), conjunto de funções e bibliotecas utilizadas para a construção de programas aplicativos.

CI – Circuito Integrado.

DSP – (*Digital Signal Processor*) Processador digital de sinais.

FPGA – (*Field Programmable Gate Arrays*) Arranjos de portas programáveis em campo.

IP – (*Internet Protocol*) Protocolo Interredes.

ISO – (*International Organization for Standardization*) Organização Internacional para Padronização.

OSI – (*Open Systems Interconnection*) Interconexão de Sistemas Abertos.

PLD – (*Programmable Logic Device*) Dispositivo de lógica programável.

TCP – (*Transfer Control Protocol*) Protocolo de Controle de Transporte.

RESUMO

A tecnologia da Internet trouxe uma grande revolução para a telecomunicação. Da mesma forma a tecnologia de hardware reconfigurável deve mudar a forma como se projetam e implementam novos dispositivos embarcados, tornando-os mais dinâmicos e flexíveis.

Assim neste trabalho procura-se juntar essas duas "revoluções", a da comunicação e a do hardware, permitindo a reconfiguração de FPGA´s remotamente, e a interação com estas, através da leitura de sinais de saída e do envio de sinais de controle, procura-se demonstrar a viabilidade do uso de sistemas embarcados para a implementação de sistemas re-configuráveis remotamente.

O objetivo primário deste estudo foi o desenvolvimento de um sistema de reconfiguração remota microcontrolado, conectado à Internet através de uma interface Ethernet implementada no mesmo. Outro ponto almejado foi a construção de um protocolo que permitisse a configuração, o controle e monitoração remotos da FPGA.

Mesmo que não se tenha obtido o sistema completamente embarcado, proposto inicialmente, os resultados e conclusões obtidos são de extrema importância para estudos futuros e construção de um sistema completamente embarcado, ficou claro que este não é inviável ou impossível, e apenas requer um projeto de hardware mais aprimorado e dimensionado de forma mais especifica para a aplicação pretendida.

Palavras-chave: Hardware reconfigurável, Internet, FPGA, Protocolo, sistemas embarcados.

ABSTRACT

The technology of the InterNet brought a great revolution for the telecommunication, in the same way the technology of the reconfigurable hardware must change the form as if they project and they implement new embarked devices, becoming them more dynamic and flexible.

Thus in this work it is looked to join these two "revolutions", of the communication and of the hardware, allowing the reconfiguration of FPGA's remotely, and the interaction with these, through the reading of exit signals and of the sending of control signals, is looked to demonstrate the viability of the use of systems embarked for the implementation of systems reconfigured remotely.

The primary objective of this study was the development of a microcontrolled system of remote reconfiguration, connected to the InterNet through a Ethernet interface implemented in exactly. Another longed for point was the construction of a protocol that allowed the configuration, the remote control and monitoring of the FPGA.

Exactly that if it has not gotten the system completely embarked, considered initially, the gotten results and conclusions are of extreme importance for future studies and construction of a system completely embarked, was clearly that this is not impracticable or impossible, and only requires a project of the improved hardware more and dimensioned of form more it specifies for the intended application.

Keywords: The reconfigurable hardware, InterNet, FPGA, embarked Protocol, systems.

1 INTRODUÇÃO

Entre as principais soluções de comunicação e conectividade existentes, a Internet tem se fixado como solução amplamente difundida e de custo reduzido, tornando-se quase um padrão de comunicação. A implementação de Intranets, Extranets, redes de negócios, VPNs e outras redes de comunicação, baseadas na tecnologia Internet torna quase que obrigatória a inserção da mesma no ambiente corporativo [MOKARZEL, 2004].

Na atualidade a Internet vem sendo amplamente utilizada para interligar pessoas e sistemas, a fim de possibilitar a difusão de informação e dados. Porém outras aplicações vêm surgindo, aproveitando-se da facilidade e da abrangência da conectividade oferecida pela Internet. Uma delas é a utilização de servidores embarcados em dispositivos dedicados, enquanto a maior parte das aplicações de Internet roda em computadores com sistemas operacionais complexos e com finalidades e características diversificadas, os sistemas embarcados normalmente são especialistas e dispõe de recursos de hardware e software limitados [MOKARZEL, 2004].

Tendo em vista o grande crescimento que este tipo de aplicação vem apresentando e as inúmeras possibilidades de utilização da mesma, que vão desde a automação doméstica ao chão de fábrica de indústrias passando pela segurança pública, fica fácil observar que o domínio destas aplicações embarcadas conectadas à Internet, e das tecnologias envolvidas nas mesmas, apresenta-se como um diferencial profissional muito interessante e vantajoso.

De outro lado, os sistemas baseados em lógica reconfigurável surgem para mudar a forma como sistemas digitais são projetados e arquitetados. A própria idéia de um hardware que pode ser alterado e atualizado sem a necessidade de substituição física de componentes, por si só já desperta idéias e expectativas muito grandes por parte dos projetistas [RIBEIRO, 2002].

A junção do poder de conectividade remota da Internet com a possibilidade de reconfiguração e atualização do hardware dos sistemas baseados em lógica reconfigurável do tipo FPGA (*Field Programmable Gate Array*) [TOCCI,2003], trariam uma nova visão aos projetos de sistemas digitais, uma vez que permitiriam a reconfiguração de hardware em campo, através de uma simples conexão Internet, sem a

necessidade de substituição de componentes, e nem mesmo da presença física de algum técnico junto ao dispositivo.

Assim este projeto visa a construção de equipamento que possibilite esta conectividade, além da especificação de um protocolo de comunicação que permita a reconfiguração e a monitoração de FPGAs em campo.

2 REVISÃO TEÓRICA

Antes de adentrar a especificação técnica do projeto, se faz necessário revisar e apresentar alguns conceitos e princípios sobre os quais este projeto é construído. Porém, esta é apenas uma revisão superficial de tais conceitos, apresentando-os em um nível de abstração mais alto. Uma visão mais detalhada dos mesmos será vista mais à frente no próprio projeto.

Como este projeto se funda basicamente em duas grandes tecnologias, os padrões Internet e os dispositivos de lógica programável do tipo FPGA [TOCCI,2003], esta revisão teórica também se divide em duas partes.

Na primeira é apresentado um breve histórico da tecnologia Internet e tenta-se esclarecer as diferenças entre o que comumente se chama de Internet e a total abrangência do termo.

Na segunda objetiva-se a introdução ao conceito de dispositivos lógicos programáveis e à tecnologia de FPGA´s, apresentando-se uma visão geral de sua arquitetura e de suas aplicações.

2.1 Tecnologia Internet

Nos dias atuais, quando falamos em Internet pensa-se imediatamente na rede mundial de computadores, inclusive com a idéia errônea de que esta é uma única rede. Porém o termo Internet possui origens de significados mais abrangentes.

Para que se compreenda o potencial de interconexão a que este projeto se propõe, é necessário primeiro entender o real significado do termo Internet e por conseqüência toda sua aplicabilidade. [COMER,1998]

2.1.1 Histórico

Em meados de 1960 os primeiros terminais deixaram o centro de processamento de dados para serem instalados à “distância”, surgindo assim a necessidade de conectá-los ao computador principal, este era o inicio das tecnologias de teleprocessamento. Nesta mesma década o governo norte-americano através do Departamento de Defesa, deu inicio a suas pesquisas quanto a aplicabilidade das tecnologias de redes de

computadores, resultando com a primeira versão da rede ARPA em 1972, dando inicio à utilização da tecnologia de comutação de pacotes [CARVALHO, 1997].

Durante a década de 1970 muitas universidades dos EUA foram conectadas à rede ARPA, assim como outras instituições de pesquisa. Na década de 1980 houve um grande crescimento das redes locais de computadores, impulsionada pelas soluções de baixo custo baseadas em mini e microcomputadores. Neste contexto, cada fabricante de tecnologia de redes ditava as regras de comunicação em sua rede e criava seus próprios protocolos e suas próprias soluções de comunicação [CARVALHO, 1997].

Porém com isto surgiu um novo problema. Os usuários destas redes isoladas sentiam a necessidade de obter acesso às informações e recursos disponíveis em outras redes. Éra impossível se imaginar o estabelecimento de uma rede global única, baseada em uma única tecnologia de hardware, já que cada tecnologia existente possui características próprias que atendiam melhor a uma ou outra necessidade, fazendo com que cada grupo optasse por uma solução que atenda melhor ao seu perfil e que viesse de encontro aos seus anseios em particular [COMER, 1998].

Com intuito de interligar redes heterogêneas, baseadas em hardwares e sistemas operacionais diferentes, e espalhadas geograficamente, no inicio de década de 1980 teve inicio um projeto, utilizando-se como espinha dorsal o potencial instalado da rede ARPA, tinha início a Internet [CARVALHO, 1997].

2.1.2 Arquitetura Internet

Embora muita confusão se faça entre a arquitetura OSI e a arquitetura Internet, estas são arquiteturas distintas. A arquitetura OSI ainda é muito utilizada e citada quando se estudam as tecnologias de redes e as abordagens baseadas em camadas. Porém, esta parece ter sido abandonada, dando-se preferência à arquitetura Internet. A arquitetura OSI foi criada pelo ISO visando à padronização de uma arquitetura de redes baseada em sistemas abertos [COMER, 1998].

Quando finalmente a especificação do modelo OSI foi concluída e apresentada ao publico o modelo Internet já se encontrava em operação, o que talvez explique a não adoção do modelo OSI [CARVALHO, 1997]. Como o modelo Internet vem se apresentando flexível e robusto o suficiente para atender às demandas que surgiram nestes últimos 20 anos, com poucas e pequenas adaptações, possibilitando seu uso em situações nem mesmo previstas quando da sua elaboração, podemos crer que este se

firmou como sendo o modelo a ser utilizado ainda por um longo período [COMER, 1998].

2.1.3 Protocolo Internet

A Internet não é apenas uma rede, mas a interconexão de várias redes de computadores, as quais não precisam necessariamente ser baseadas na mesma tecnologia de hardware e sistema operacional.

Este feito deve-se ao Protocolo Internet (IP – *Internet Protocol*) o qual resolve o problema de interligação entre as redes, baseando-se em *gateways* para fazer a passagem dos pacotes de um meio de rede para outro. Por não definir protocolos ou padrões para as camadas de rede e de enlace, a arquitetura Internet é portável para qualquer plataforma, bastando que seus protocolos básicos sejam implementados na mesma [COMER, 1998].

2.1.4 Protocolo TCP

O Protocolo Internet (IP) se responsabiliza apenas pelo transporte dos dados, não dando nenhuma garantia da entrega dos mesmos, ou segurança quanto ao não comprometimento do conteúdo dos pacotes enviados. Para atender à esta necessidade é que existe o protocolo da camada de transporte conhecido como TCP (*Transfer Control Protocol*) [COMER, 1998].

O protocolo TCP foi criado para ser utilizado sobre o protocolo IP, pois seu sistema de *sockets*, baseia-se na combinação de portas e endereços IP. Algumas informações como o destinatário do pacote a ser expedido ou o remetente do pacote recebido são extraídas diretamente do cabeçalho do datagrama IP [MOKARZEL, 2004].

Embora a tecnologia IP se baseie em redes de comutação de pacotes, o protocolo TCP é orientado à conexão, ou seja, ele estabelece uma conexão com o outro computador e a mantém durante toda a transmissão dos dados, encerrando apenas ao final. Para isto o protocolo TCP cria circuitos virtuais de comunicação [COMER, 1998].

2.2 Dispositivos de lógica programável

Dispositivos de lógica programável ou simplesmente PLD´s (*Programmable Logic Devices*) ao contrário de computadores e DSP´s (*Digital Signal Processor*) não são programados com instruções a serem executadas em uma determinada ordem. Ao

invés disso este dispositivos tem seu hardware reconfigurado através da conexão ou desconexão eletrônica de partes do circuito [TOCCI, 2003].

Com estes dispositivos pode-se implementar os mesmos circuitos digitais, para os quais seriam necessários vários CI´s específicos e um considerável espaço físico, em um único chip [TOCCI, 2003].

2.2.1 FPGA´s

Um categoria especial de PLD´s, as FPGA´s são compostas por vários blocos lógicos que podem ser interconectados através de barramentos de linhas e colunas, possibilitando assim a implementação de uma grande variedade de projetos de sistemas digitais. As FPGA´s normalmente são dotadas de portas de I/O que podem ser utilizadas como portas de entrada, saída ou bidirecionais, através de pinos tristate. Podem ser construídas com tecnologias do tipo SRAM, EEPROM, Flash EEPROM e antifusível, entre outras [TOCCI, 2003].

As FPGA´s são um tipo bem avançado de PLD´s, cujo potencial de uso e a capacidade podem variar de dispositivos relativamente limitados à dispositivos com alto grau de granularidade.

3 ESPECIFICAÇÃO TÉCNICA

Aqui é apresentada uma visão geral do sistema desenvolvido, com uma descrição mais superficial de cada um dos componentes do mesmo e a sua respectiva função e importância no conjunto do sistema.

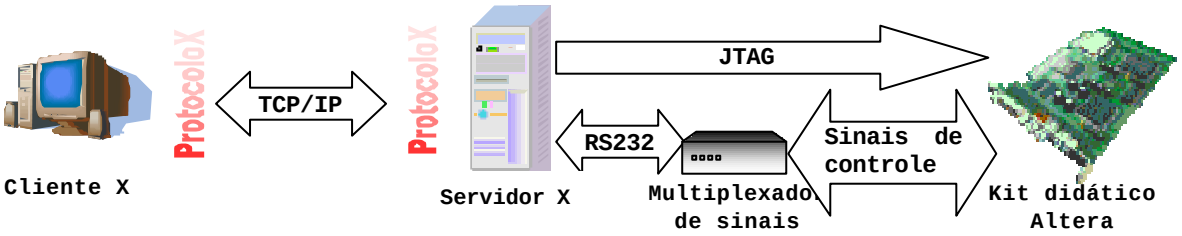


Figura 1 – Componentes do projeto

A

Figura 1 apresenta a estrutura básica do projeto e seus componentes em uma visão macro do mesmo. O sistema todo será composto de uma aplicação cliente, apresentada como “Cliente X”, o “ProtocoloX” responsável pela reconfiguração, controle e monitoração remota da FPGA, o “ServidorX”, responsável por receber os arquivos JAM e reconfigurar as PGA e uma multiplexador que é responsável pela interface entre a FPGA e a aplicação servidor.A seguir serão apresentados cada um destes componentes com suas respectivas características e uma descrição de suas funcionalidades.

3.1 JAM Byte Code e JAM Player

A linguagem Jam STAPL (*Jam Standard Teste and Programming Language*) foi criada pelos engenheiros da Altera e é suportada por um consórcio formado por fabricantes de PLD´s (*Programmable Logic Devices*), fabricantes de equipamentos de programação de PLD´s e fabricantes de equipamentos para teste de PLD´s. Esta linguagem foi adotada como linguagem padrão pelo JEDEC (*Joint Electron Device Engineering Council*) em 1999.

Os programas escritos nesta linguagem, após compilados geram um arquivo chamado JAM Byte Code, o qual é então interpretado por um programa chamado de

JAM Player, que é responsável por gerar os sinais para a interface J-TAG para a configuração ou teste do PLD, conforme ilustra a Figura 2.

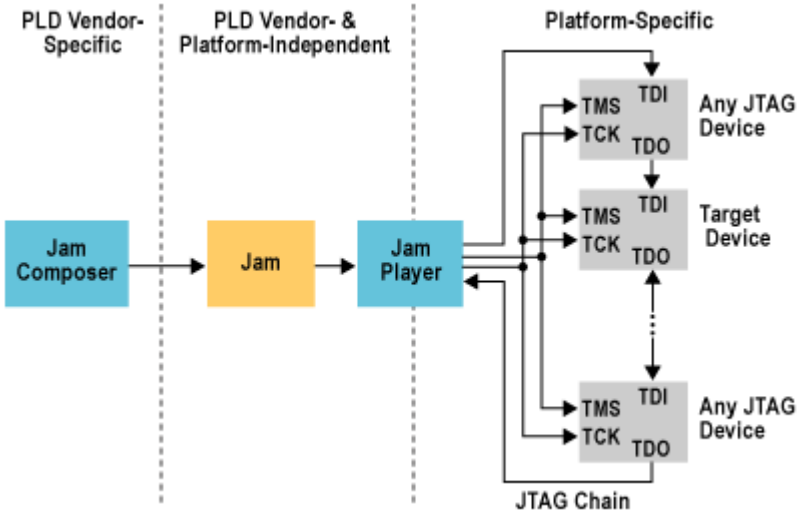


Figura 2 – Fluxo básico do Jam STAPL

3.2 Kit didático Altera

A Altera é uma das maiores fabricantes de FPGA´s, entre os quais os chips da família MAX 7000 e FLEX 10K, ambos presentes no kit didático utilizado no projeto.

O kit didático utilizado é o modelo *University Program UP2 Education Kit*, ele foi projetado para atender às necessidades dos professores de sistemas digitais das universidades com o que há de mais moderno em se tratando de dispositivos lógicos programáveis. O UP2 possui um chip Flex®10K e um MAX® 7000 e interface de comunicação JTAG, a qual é utilizada para a comunicação e reprogramação do kit [ALTERA, 2004].

3.3 Protocolo X

O protocolo descrito nos diagramas como protocolo X, por ainda não possuir um nome definido, é responsável pela comunicação entre o software de controle e monitoração e o hardware, ligado ao Kit Altera. Este protocolo será implementado sobre o protocolo TCP/IP conforme demonstra a Figura 3.

Figura 3 – Encapsulamento do protocolo X

O protocolo X é um protocolo orientado à conexão. Uma vez estabelecida uma conexão com um cliente, o protocolo torna o servidor dedicado à esta conexão, não podendo mais este estabelecer conexões com outros clientes enquanto esta conexão estiver ativa. Uma conexão poderá ser encerrada ou por requisição do cliente remoto ou por um longo período sem resposta por parte deste (time-out)

Os pacotes do Protocolo X estão divididos em 4 grupos, segundo suas funções.

3.3.1 Conexão

Os pacotes de conexão são utilizados para que seja estabelecida a conexão entre aplicação servidor e a aplicação cliente, são eles:

- requisição de início de conexão,
- resposta de início de conexão,
- confirmação de início de conexão,
- aviso de encerramento de conexão,
- verificação de conexão.

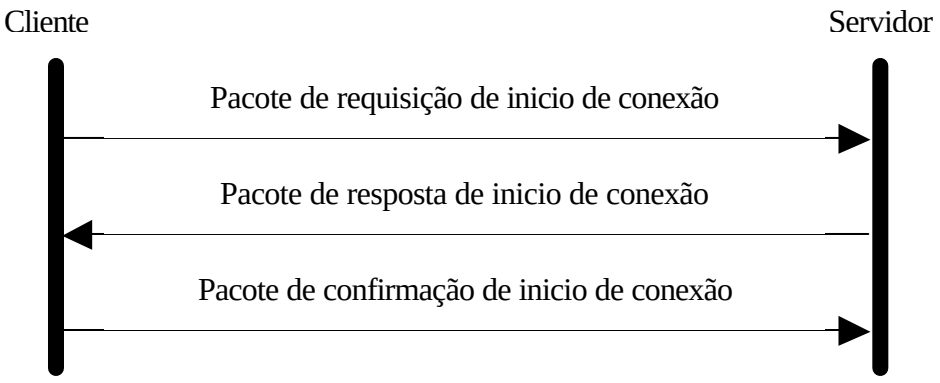


Figura 4 – Diagrama de seqüência do *hand-shake* de conexão o Protocolo X

A Figura 4 representa a seqüência utilizada para o HandShake para o estabelecimento de uma conexão do protocolo X.

O pacote de aviso de encerramento de conexão é enviado sempre que uma das partes decide, seja por excesso de tempo sem comunicação (*timeout*) ou por outra causa qualquer, para avisar a outra parte de que a conexão foi derrubada.

O pacote de verificação de conexão é utilizado para a verificação da permanência da conexão e também como resposta a um primeiro pacote de verificação de conexão, conforme apresentado na Figura 5.

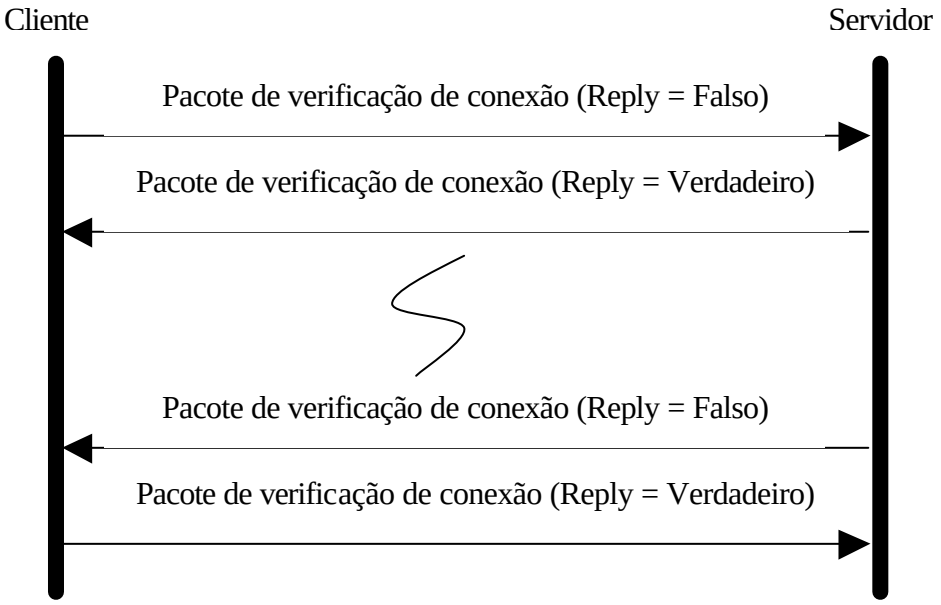


Figura 5 – Diagrama de seqüência da verificação de conexão do Protocolo X

3.3.2 Envio de arquivo

Os pacotes de envio de arquivo são utilizados para o envio dos arquivos JAM da aplicação cliente para aplicação servidor. Estes arquivos são posteriormente utilizados para a reconfiguração do Kit Altera. São os seguintes pacotes:

- requisição de início de envio de arquivo,
- autorização de início de envio de arquivo,
- início de envio de arquivo,
- envio de arquivo,
- fim de envio de arquivo.

A aplicação cliente solicita ao servidor o início do envio do arquivo JAM, uma vez autorizada por este ela indica o inicio do envio do arquivo e, passa a enviar diversos pacotes contendo o arquivo quebrado em pedaços. Ao final, a aplicação cliente indica o

final do envio do arquivo e informa ao servidor o número de pacotes enviados, para que este verifique a integridade do arquivo enviado. Este fluxo é ilustrado na Figura 6.

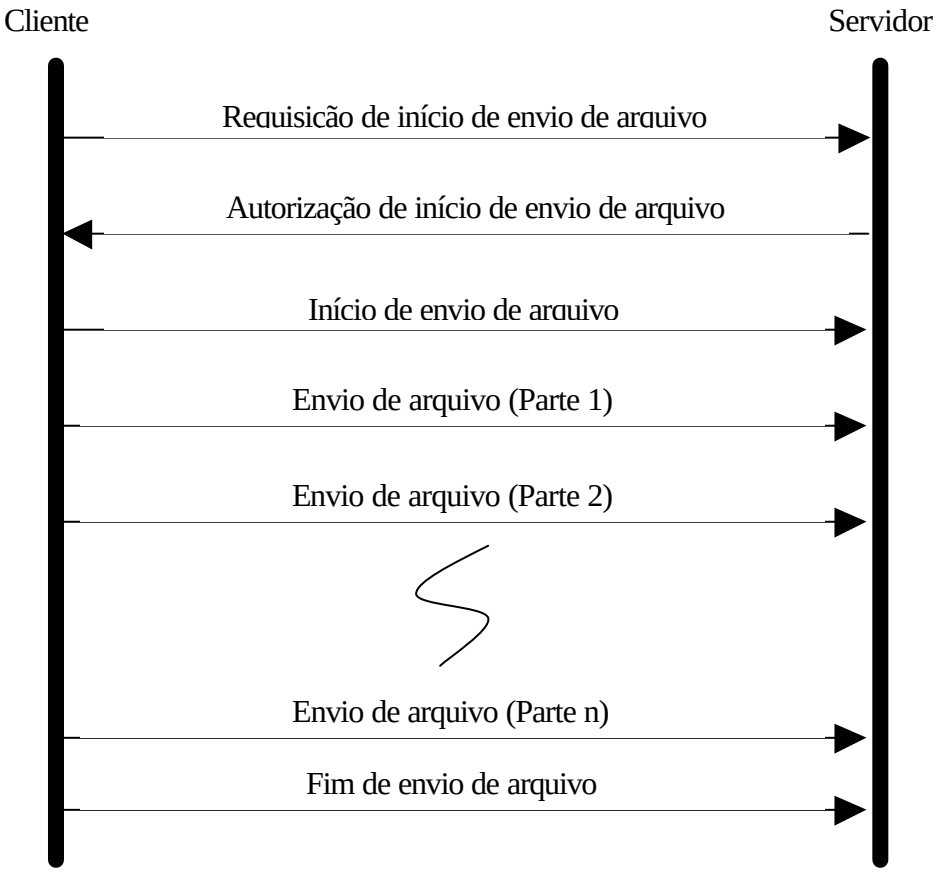


Figura 6 – Diagrama de seqüência do envio de arquivo JAM

3.3.3 Console remoto

O pacote de mensagem de console remoto é enviado pelo servidor para o cliente sempre que alguma alteração na saída do console ocorre, quando o cliente recebe o mesmo ele apenas ecoa esta alteração em seu console.

3.3.4 Controles do player

Os pacotes de controles do *player*, implementado na aplicação servidor, são enviados pelo cliente para que o servidor efetue alguma operação com o arquivo JAM recebido. Caso não tenha recebido nenhum arquivo JAM válido o servidor apenas ignora estes pacotes.

São dois os comandos possíveis de serem enviados pelo cliente. Um para a configuração do Kit com o arquivo enviado e outro para a extração de informações do arquivo enviado, o resultado de ambas pode ser visto na saída do console remoto.

3.3.5 Controles do monitor

Os pacotes de controle do monitor são utilizados tanto pelo aplicativo cliente como pelo aplicativo servidor para a transmissão dos estados dos sinais de controle do Kit, são eles:

- solicitação de estados dos pinos,
- envio de estados dos pinos,
- controle do monitor *online*.
-

O pacote de solicitação de estados dos pinos é enviado pelo cliente para solicitar ao servidor o estado atual dos sinais de controle do Kit, afim de atualizar seu status, quando este não esta operando no modo *online*.

O pacote de envio de estados dos pinos é enviado pelo servidor para o cliente para atualizar o mesmo com o estado atual dos sinais de controle, e é enviado pelo cliente para o servidor para informar este de uma interação feita com algum dos sinais de controle passiveis de serem alterados pelo usuário.

O pacote do controle do monitor *online* é enviado pelo cliente para o servidor para alternar o estado do monitor *online* entre ativado e desativado. Quando o monitor *online* esta ativado, sempre que há a alteração do valor de um sinal de controle, seja pelo lado cliente, seja pelo lado servidor, a mesma é enviada automaticamente para a outra parte da conexão.

3.4 Cliente X

O software desenvolvido para o monitoramento e envio do arquivo JAM para reprogramação do KIT implementa o protocolo X cliente, uma interface que permite o envio do arquivo JAM e um monitor que possibilita a visualização dos sinais de controle do Kit e o envio de sinais de controle para o mesmo. Também é possível utilizar-se de uma interface de simulação, a qual simula de forma gráfica o funcionamento do Kit Altera e permite a visualização do funcionamento do mesmo, assim como a interação com os controles do kit.

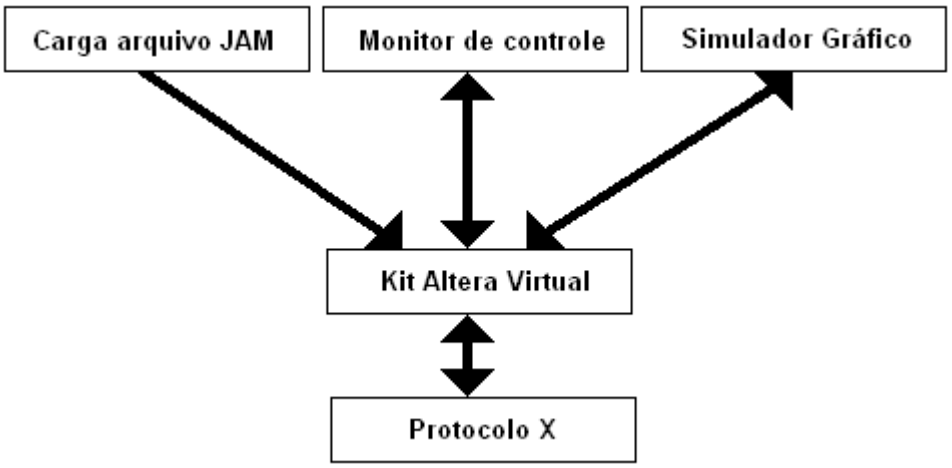


Figura 7 – Comunicação entre os componentes do software

A Figura 7 apresenta a comunicação entre os componentes do software, ressaltando o desenvolvimento modular do mesmo. Como o objetivo deste projeto é também o de fornecer API´s (*Application Program Interface*) e ferramentas que possibilitem o uso das tecnologias desenvolvidas no mesmo em projeto futuros, há a preocupação do desenvolvimento em separado de classes que implementem o Protocolo X. Uma representação virtual do Kit Altera que esta sendo controlado, o monitor de controle e o simulador gráfico. Permitindo assim que um outro projeto que faça um uso mais específico do Kit Altera, por exemplo, crie um simulador com a interface voltada para a sua aplicação, sem necessariamente ter de criar novamente a classe que representa o Kit Altera Virtual.

3.4.1 Estrutura do software

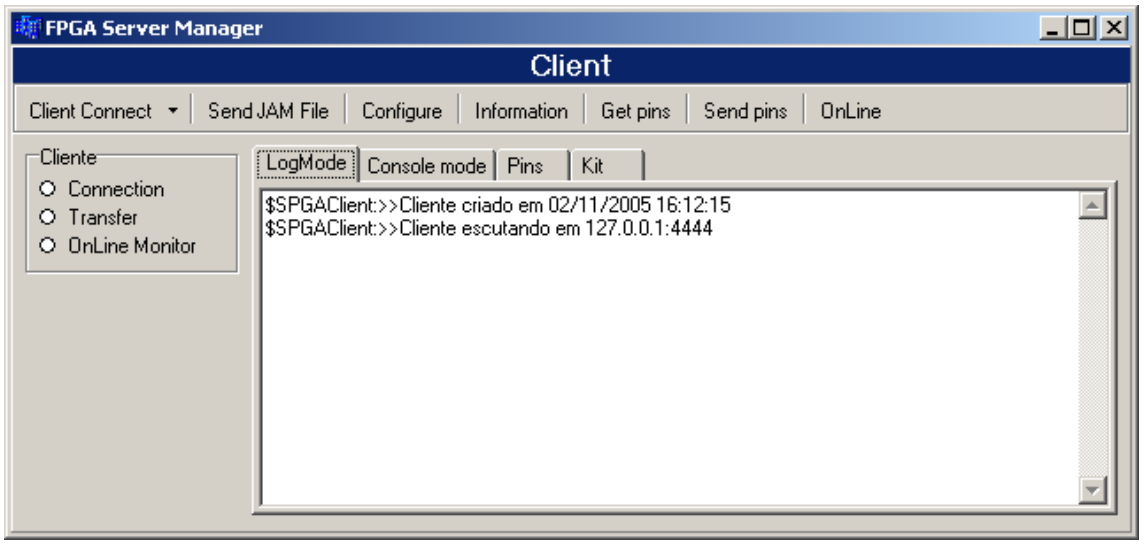


Figura 8 – Aplicação cliente em modo de histórico.

A aplicação cliente é composta de 4 telas principais, na primeira, mostrada na Figura 8, é apresentado o histórico da comunicação do aplicativo cliente com o aplicativo servidor, ali serão apresentadas mensagens referentes à comunicação entre ambos e à conexão.

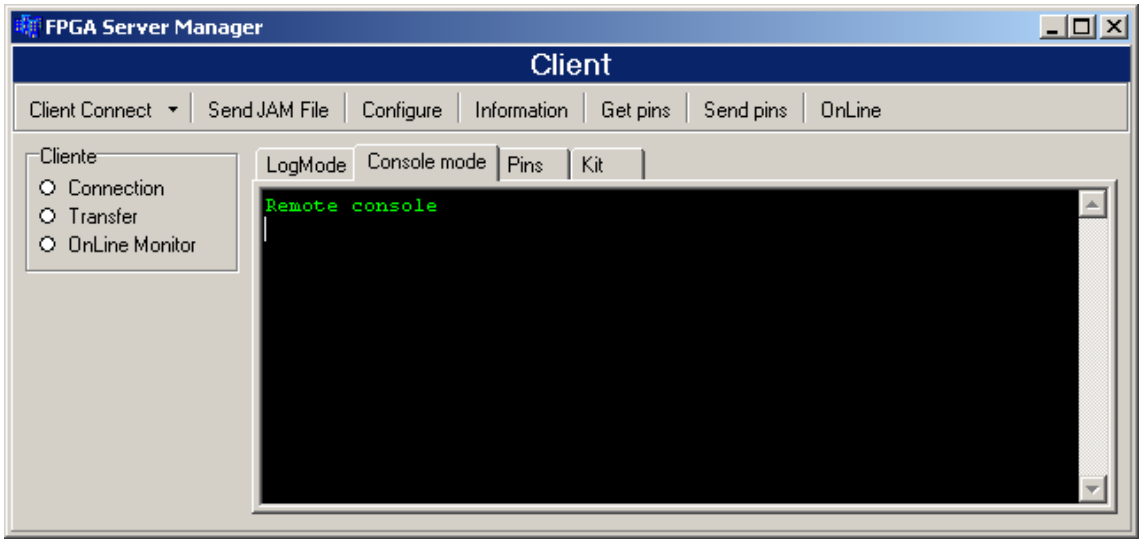


Figura 9 – Aplicação cliente em modo console

No modo console, apresentado na Figura 9,são visualizadas as saídas do JAM Byte Code Player da aplicação servidor, nesta tela podem ser visualizadas as saídas do comando de solicitação de informação do arquivo JAM enviado (Information) e do comando de configuração da FPGA (Configure).

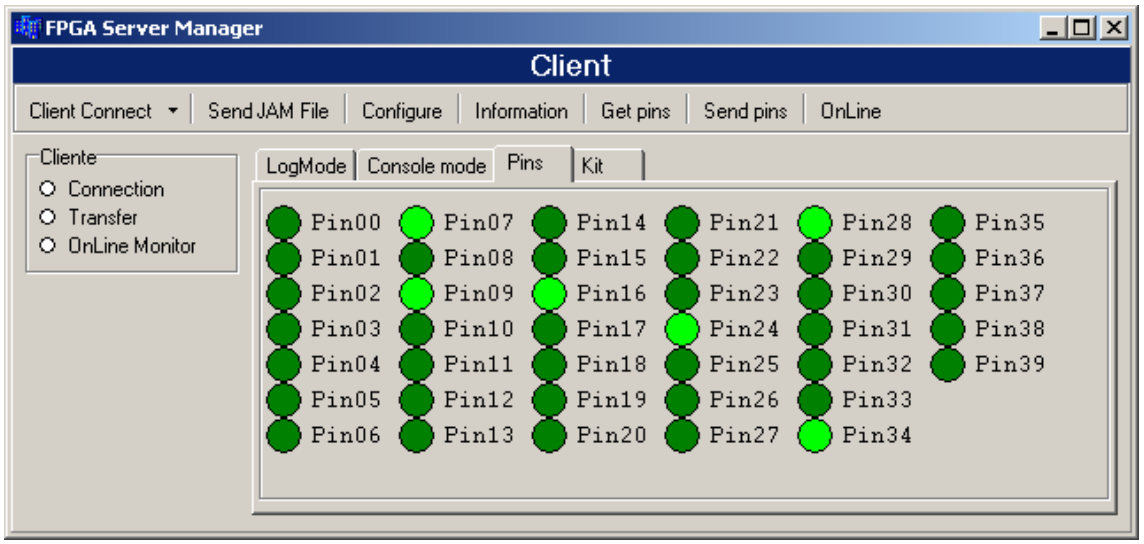


Figura 10 – Aplicação cliente no modo de sinais de controle

A Figura 10 apresenta o modo de sinais de controle, neste modo podem ser visualizados e alterados os estados dos pinos da FPGA, quando a aplicação cliente esta no modo OnLine qualquer alteração é automaticamente enviada ao Server, assim como todas as alterações do Server são enviadas automaticamente para a aplicação cliente.

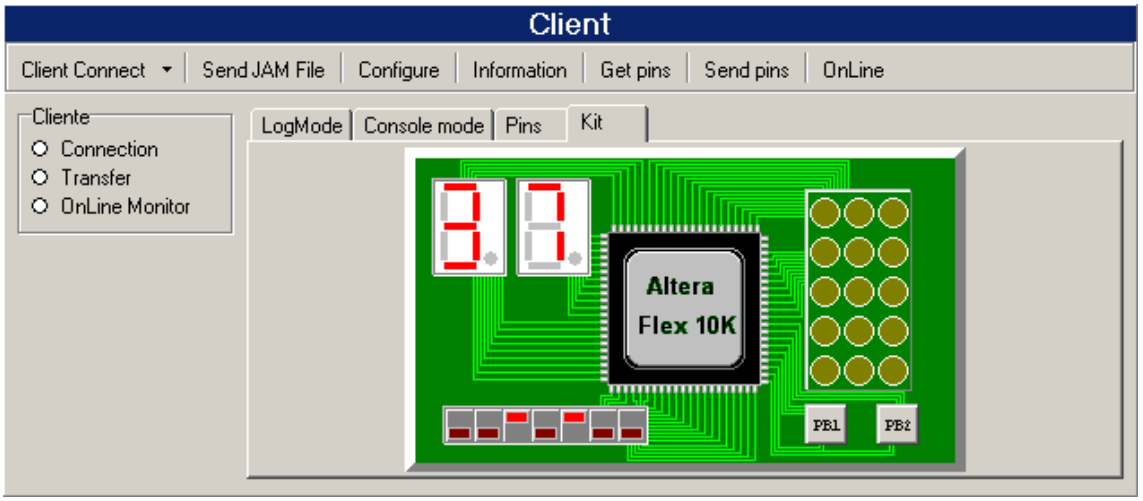


Figura 11 – Aplicação cliente no modo gráfico

No modo gráfico, apresentado na Figura 11, o usuário poderá ver uma representação gráfica do Kit Altera, com seus sinais de saída e ainda interagir com os DipSwitches e PushButtons de controle. No modo OnLine as interações com estes controles são enviadas automaticamente para o servidor, assim como as alterações percebidas pelo servidor são automaticamente enviadas à aplicação cliente.

3.5 Servidor X

O Servidor X é o software desenvolvido para a conexão com a aplicação cliente, ele é responsável por comunicar-se com o multiplexador para enviar e receber os valores dos sinais de controle do Kit Altera. Também no Servidor X esta implementado o JAM Player, responsável por executar o arquivo JAM enviado pela aplicação cliente, para a configuração da FPGA. Na aplicação servidor está disponível uma interface gráfica, a qual simula o funcionamento do Kit Altera e permite o acompanhamento do do mesmo.

A aplicação servidor foi desenvolvida, a imagem da aplicação cliente, de forma modular, com o objetivo de fornecer API´s e ferramentas que possibilitem o uso das tecnologias desenvolvidas no mesmo em projeto futuros

3.5.1 Estrutura do software

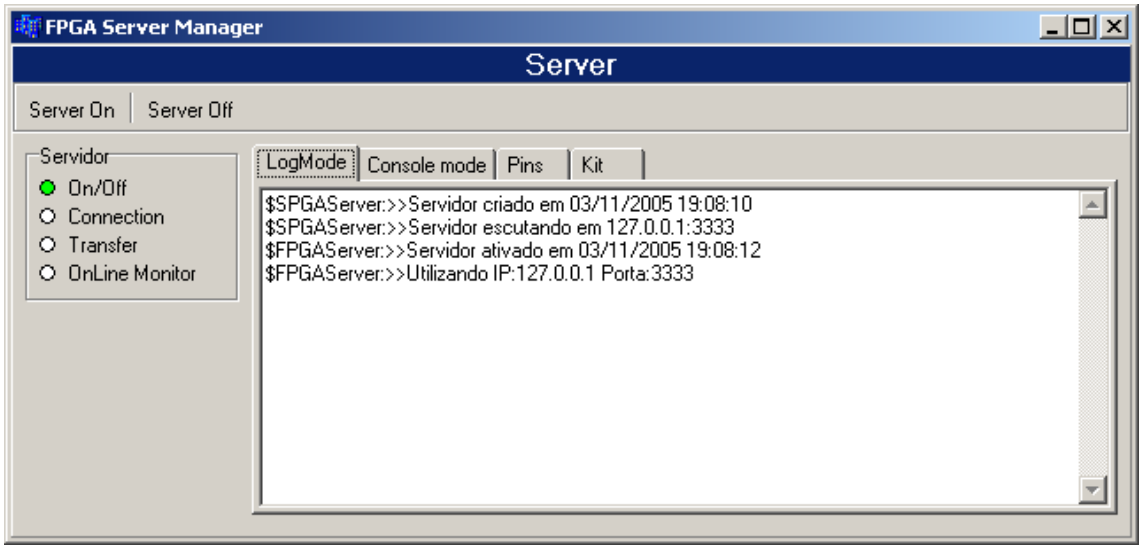


Figura 12 – Aplicação servidor em modo de histórico

A aplicação servidor é composta de 4 telas principais, na primeira é apresentado o histórico da comunicação com o aplicativo cliente, ali serão apresentadas mensagens referentes à comunicação entre ambos e à conexão, conforme mostra a Figura 12.

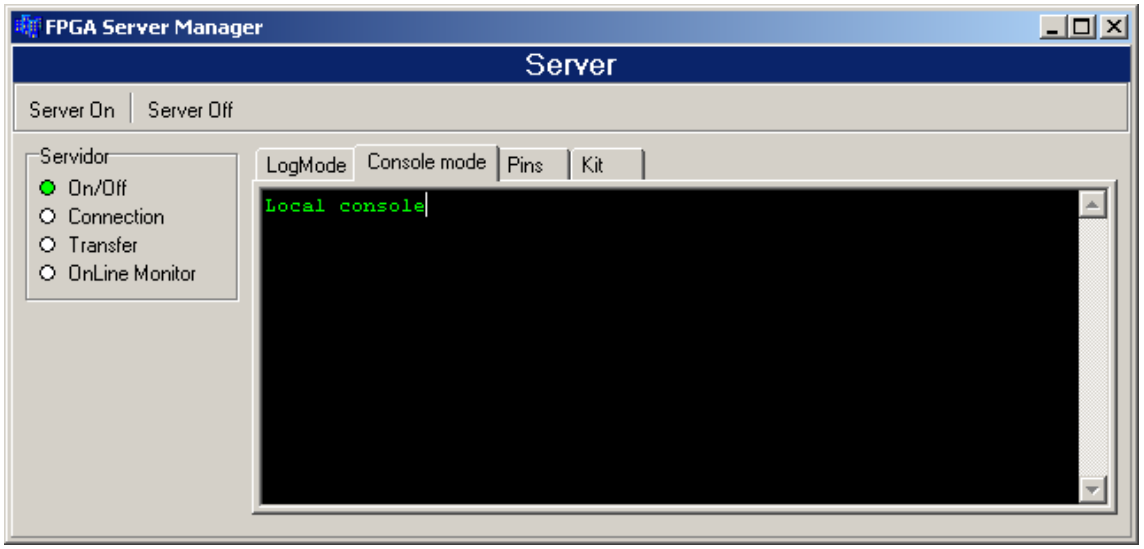


Figura 13 – Aplicação cliente em modo console

A Figura 13 apresenta o modo console, onde são visualizadas as saídas do JAM Byte Code Player, nesta tela podem ser visualizadas as saídas do comando de solicitação de informação do arquivo JAM enviado (Information) e do comando de configuração da FPGA (Configure) enviados pela aplicação cliente.

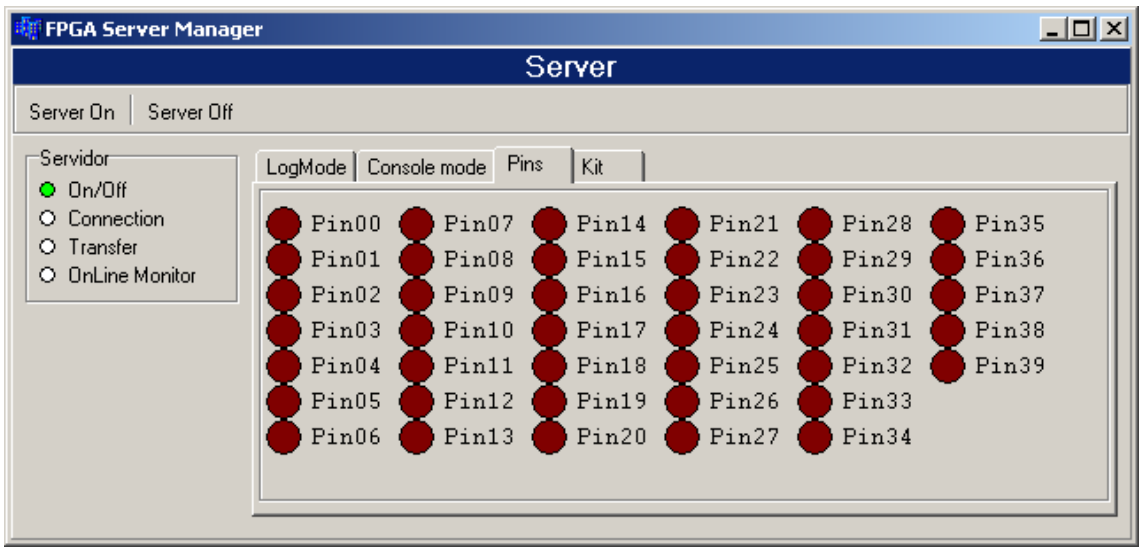


Figura 14 – Aplicação servidor no modo de sinais de controle

No modo de sinais de controle podem, mostrado na Figura 15, ser visualizados e alterados os estados dos pinos da FPGA, quando a aplicação servidor esta no modo OnLine qualquer alteração é automaticamente enviada pelo cliente, assim como todas as alterações lidas do Kit Altera são enviadas automaticamente para a aplicação cliente.

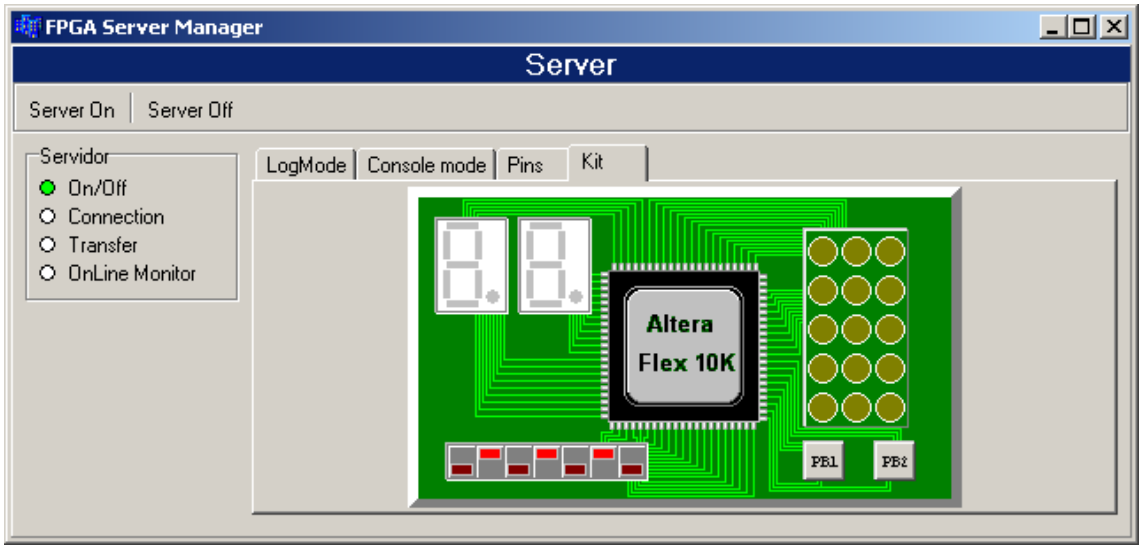


Figura 15 – Aplicação servidor no modo gráfico

No modo gráfico, apresentado na Figura 16, o usuário poderá ver uma representação do Kit Altera, com seus sinais de saída porém não é possível interagir com os DipSwitches e PushButtons de controle. No modo OnLine as interações com estes controles feitas na aplicação cliente são enviadas automaticamente para o servidor, assim como as alterações percebidas pelo servidor são automaticamente enviadas à aplicação cliente.

3.6 Multiplexador de sinais

Para que a aplicação servidor possa monitorar e interagir com os sinais de controle do Kit Altera, foi necessária a construção de um multiplexador de sinais que permitisse “quebrar” esta interação em vários “pedaços”. Para tanto foi utilizado o Kit Didático do microcontrolador 8051, utilizado pelos alunos do Unicenp, o Kit 8051 controla a comunicação serial com a aplicação servidor, e ainda se encarrega de “quebrar” os conjuntos de sinais de controle em vários pedaços de 1 Byte cada.

Para multiplexar os sinais enviados pelo 8051 de 1 barramento de 8 bits para 5 de 8 bits, permitindo assim o minitoramento/controle de 40 bits de controle, utiliza-se um circuito de multiplexação controlado através do barramento de endereços do Kit 8051.

3.7 Validação

A validação do projeto é feita em etapas distintas, seguindo uma ordem de implementação dos componentes do mesmo. Assim ao final da ultima etapa todo o sistema deverá estar operante.

3.7.1 Validação do Protocolo X

Por serem criadas em camadas, tanto a aplicação servidor quanto a aplicação cliente foram implementadas em paralelo, permitindo assim a construção e validação do Protocolo X de forma progressiva.

O Protocolo X é construído de forma única, criando uma API consolidada, a ser utilizada por ambas a aplicações.

São criadas e testadas primeiramente as funções de controle de conexão e desconexão, uma vez testadas estas, então é criada e validada a implementação do envio de arquivo JAM. Na seqüência os controles do player e do console remoto são implementados e testados, por fim a implementação do monitor OnLine é adicionada ao protocolo.

Por serem construídos independentes das interfaces gráficas, após a implementação da aplicação cliente e da aplicação servidor, são então construídas as interfaces dos mesmos e então testadas as interações entre estas e as aplicações.

3.7.2 Validação do Player JAM e da interface J-TAG

Para confirmar o funcionamento do Player JAM fornecido pela Altera, primeiramente um arquivo JAM, já previamente testado utilizando-se a ferramenta Quartus II da Altera, é enviado para o servidor e então executa-se a configuração do Kit para que seja verificado o funcionamento da mesma.

3.7.3 Validação do firmware do multiplexador

Para validar o controle do multiplexador, o Kit Altera foi configurado de forma a enviar continuamente valores fixos para as saídas e a alterar os mesmos de acordo com os sinais de entrada. De acordo com os sinais enviados pela aplicação servidor e dos resultados obtidos, pode-se validar o funcionamento do mesmo.

3.7.4 Validação final

Por fim um pequeno projeto de um contador simples, construído para o Kit Altera, é enviado para o servidor e então configurado no Kit, o funcionamento deste é monitorado através do simulador gráfico validando então as funções de monitoramento On-Line e de interação com o Kit Altera através do simulador.

3.8 Recursos necessários

Para o desenvolvimento deste projeto são necessários os seguintes recursos de materiais, equipamentos e softwares:

- Kit didático 8051.
- Kit didático Altera UP2.
- Rede padrão Ethernet.
- Compilador C para 8051.
- Borland C++ Builder.
- Altera Quartus II.
- Altera Player JAM Byte Code.
- Estação de trabalho Windows conectada à rede Ethernet.

3.9 Custos

Esta é uma avaliação preliminar dos custos para o desenvolvimento do projeto, sem considerar-se ainda os valores com materiais e componentes eletrônicos, uma vez que estes somente serão identificados e quantificados na fase de projeto. Assim o que se pode considerar é o tempo gasto com o desenvolvimento do mesmo e os recursos necessários.

Sendo este um projeto acadêmico e possuindo o UnicenP os equipamentos e ferramentas de desenvolvimento necessários ao projeto, o único custo então a ser considerado é o do tempo a ser despendido com o mesmo. Vale realçar porém que os custos com equipamentos e software são sim significativos, caso estes não estivessem a disposição. O que poderia elevar em muito o custo final deste projeto, fato que deve ser considerado quando ao planejamento do desenvolvimento de projeto similar, podendo inclusive em implicar na escolha de outras tecnologias ou equipamentos, como um outro Kit com FPGA´s Altera, por exemplo.

Para um melhor levantamento de custo, é necessário mensurarmos o tempo estimado de desenvolvimento de cada uma das etapas do projeto, conforme a Tabela 1.

Descrição	Tempo (dias)
Protocolo X	3
Interface cliente	1
Interface servidor	1
Multiplexador	3
Firmware do multiplexador	3

Tabela 1 – Tempos para desenvolvimento do projeto

Assim têm-se um tempo total previsto de 11 dias de trabalho para o desenvolvimento do projeto, a um custo estimado de R$ 20,00 a hora trabalhada e utilizando-se como parâmetro que 1 dia de trabalho equivale a 8 horas de trabalho chegamos a um custo de R$ 1.760,00 no que se refere à mão de obra.

3.10 Cronograma

Com base nas tarefas contabilizadas no levantamento de custos pode-se elaborar um cronograma para execução das tarefas conforme o apresentado na Tabela 2.

Descrição	Tempo	Início	Término
Protocolo X	3	10/09/2005	17/09/2005
Interface cliente	1	18/09/2005	18/09/2005
Interface servidor	1	24/09/2005	24/09/2005
Multiplexador	3	25/09/2005	02/10/2005
Firmware do multiplexador	3	08/10/2005	15/10/2005

Como algumas tarefas devem ser desenvolvidas nos laboratórios do UnicenP, este cronograma leva em conta os dias e horários disponíveis dos referentes laboratórios, desconsiderando assim a execução destas tarefas nos domingos.

4 PROJETO

4.1 Software

Nesta seção serão descritos os detalhes da construção e do funcionamento do software desenvolvido no projeto.

4.1.1 Protocolo X

Conforme dito anteriormente, o Protocolo X possui 5 grupos de pacotes, assim a estrutura básica de um pacote do Protocolo X segue o apresentado na Figura 16.

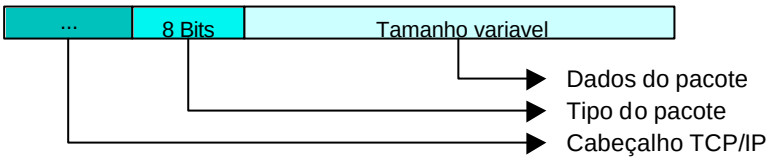


Figura 16 – Estrutura básica de um Pacote X

4.1.1.1 Conexão

São 5 os pacotes utilizados para o controle de conexão do Protocolo X, conforme apresentado abaixo.

O pacote de requisição de início de conexão, apresentado na Figura 17, é enviado pela aplicação cliente para o servidor, solicitando ao mesmo uma nova conexão. Caso o servidor esteja esperando por novas conexões, modo *Listen*, ele envia um novo pacote do tipo resposta de inicio de conexão ao cliente, indicando que foi aceita a solicitação. Caso o servidor não esteja em modo *Listen*, ele responde ao cliente com um pacote de resposta de inicio de conexão, porém indicando como recusada a solicitação. A Figura 18 ilustra o processamento do pacote de requisição de início de conexão.

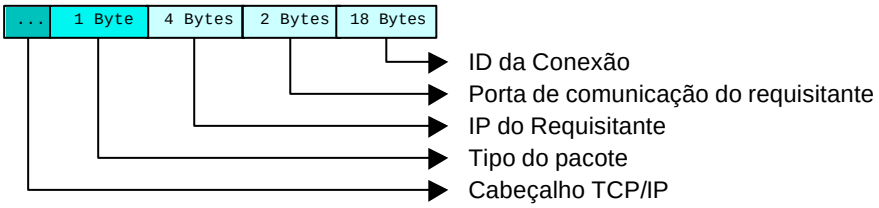


Figura 17 – Requisição de inicio de conexão

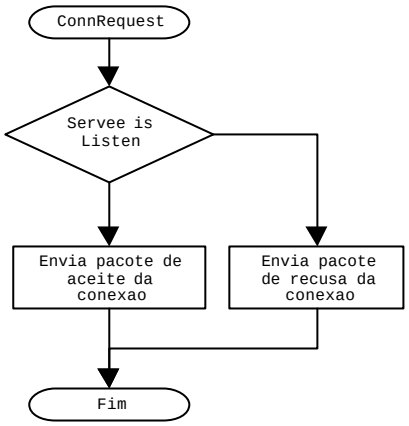


Figura 18 – Processamento do pacote de início de requisição

O pacote de resposta de início de conexão, mostrado na Figura 20, enviado pelo servidor para o cliente, indica se a requisição de conexão efetuada pelo cliente foi aceita ou não. Ao recebê-la, a aplicação cliente verifica se estava aguardando uma resposta de conexão e caso esteja, se a resposta recebida é referente à conexão que tentou iniciar anteriormente. Caso as condições anteriores sejam atendidas, a aplicação cliente envia ao servidor um pacote informando a este o início da conexão. Caso contrário ele ignora o pacote recebido e o descarta. A Figura 20 ilustra o processamento deste pacote.

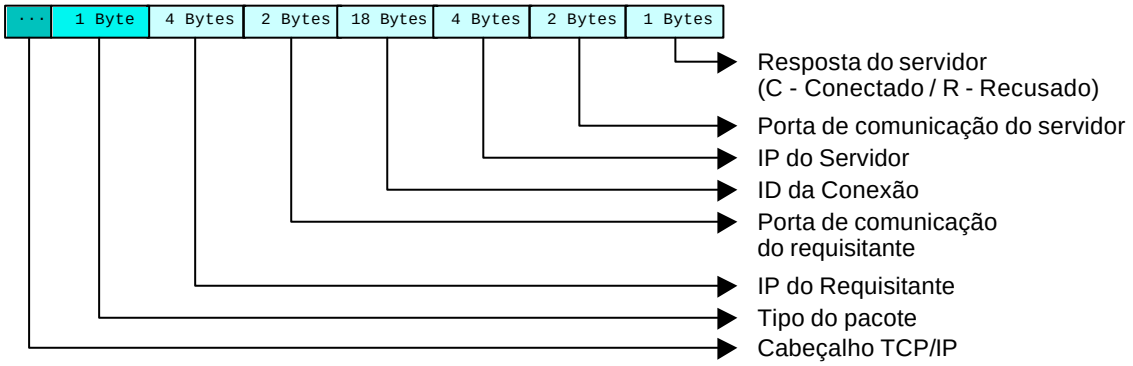


Figura 19 – Resposta de início de conexão

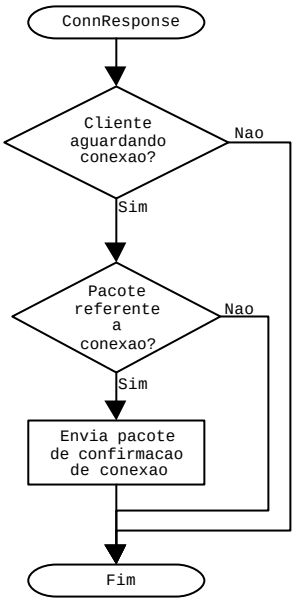


Figura 20 – Processamento do pacote de resposta de início de conexão

O pacote de confirmação de início de conexão, apresentado na Figura 21, é enviado pelo cliente para o servidor, informando este de que a conexão está estabelecida. O servidor ao receber este tipo de pacote verifica se estava aguardando uma confirmação de conexão, e se a confirmação recebida é referente à conexão iniciada anteriormente. Caso sejam atendidas ambas as condições anteriores o servidor passa ao estado de conectado. A Figura 22 ilustra o processamento deste pacote.

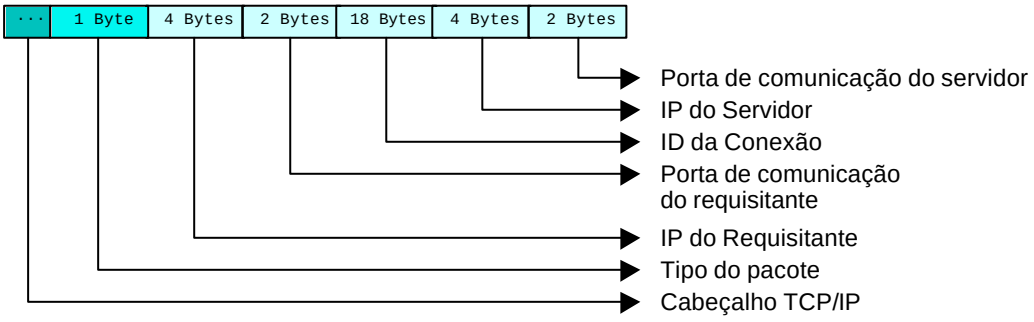


Figura 21 - Confirmação de início de conexão

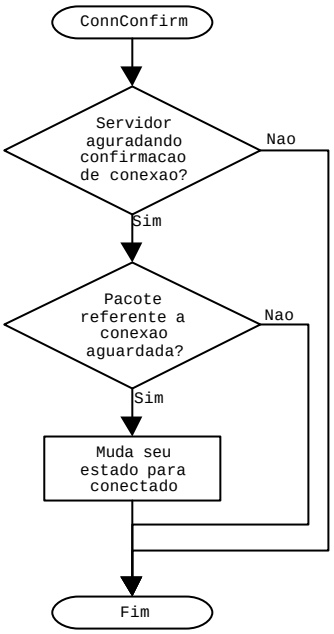


Figura 22 – Processamento da confirmação do inicio de conexão

As duas causas possíveis para o encerramento de uma conexão: por solicitação do cliente ou caso qualquer uma das partes detecte que a outra ficou tempo demais sem responder às suas requisições. Quando uma destas duas ocorre, a parte que tomou a iniciativa de derrubar a conexão apenas informa isto a outra, através de uma mensagem de aviso de encerramento da conexão, cujo pacote é apresentado na Figura 23. Ao receber este tipo de pacote, tanto a aplicação cliente como a aplicação servidor alteram seus estados e dão por encerrada a conexão, conforme ilustra a Figura 24.

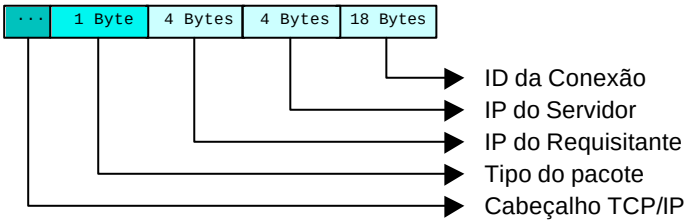


Figura 23 - Aviso de encerramento de conexão

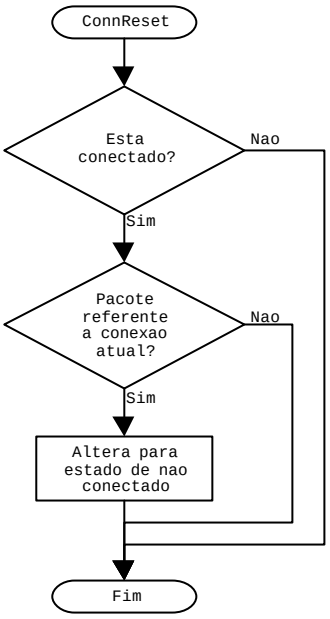


Figura 24 – Processamento do pacote de encerramento de conexão

Sempre que uma das partes detectar que desde o último envio de pacote se passou metade do tempo máximo para que a conexão caia, esta envia a outra um pacote para verificação da conexão, mostrado na Figura 25. A outra parte ao receber este tipo de pacote responde à solicitante, reenviando o mesmo pacote, porém marcando este como sendo uma resposta, conforme ilustra a Figura 26.

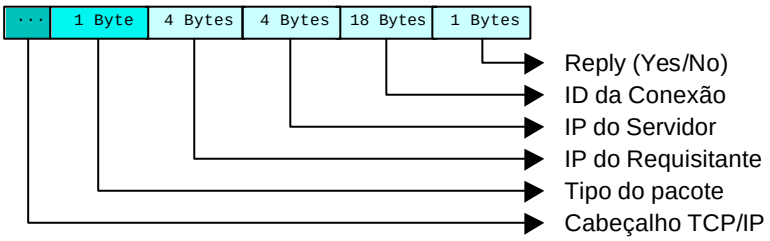


Figura 25 - Verificação de conexão

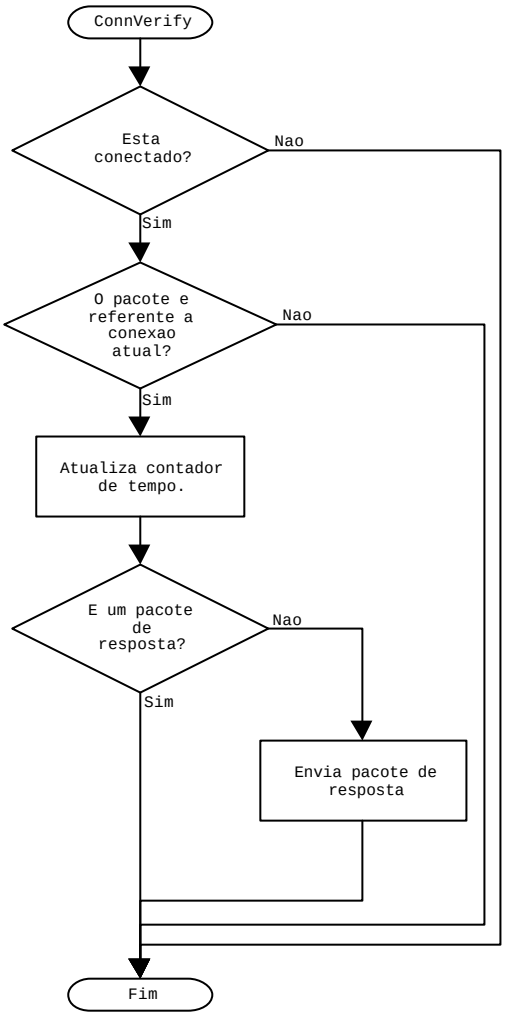


Figura 26 – Processamento do pacote de verificação de conexão

4.1.1.2 Pacotes de envio de arquivo

Abaixo são descritos os pacotes utilizados para a transmissão do arquivo contendo o JAM Byte Code, enviado do aplicativo cliente para o servidor.

O pacote de requisição de envio de arquivo, mostrado na Figura 27, é enviado pelo cliente para o servidor, solicitando a este autorização para o início do envio do arquivo. Caso o servidor esteja conectado com o cliente solicitante e não esteja recebendo ainda um arquivo previamente enviado, este responde a solicitação com um pacote de autorização de envio de arquivo, conforme ilustra a Figura 28.

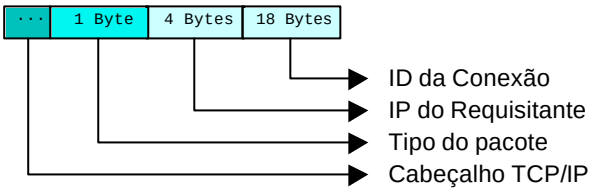


Figura 27 - Requisição de inicio de envio de arquivo

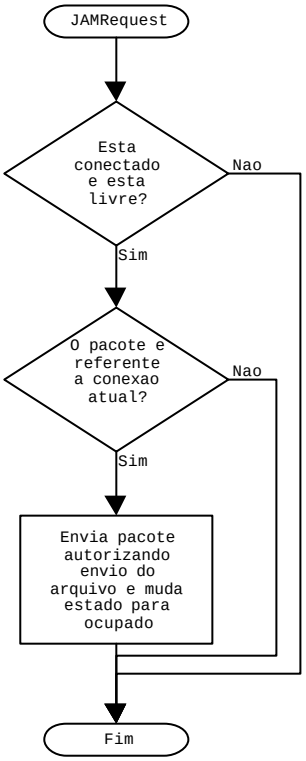


Figura 28 – Processamento da requisição de inicio de envio de arquivo

O servidor envia ao cliente um pacote de autorização de início de envio de arquivo, apresentado na Figura 29, caso este possa iniciar o envio do arquivo. Ao receber este tipo de pacote o cliente envia um pacote informando ao servidor que iniciará o envio do arquivo e na seqüência envia os pacotes com os pedaços do arquivo, conforme ilustra a Figura 30.

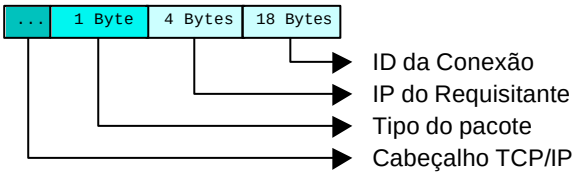


Figura 29 – Autorização de inicio de envio de arquivo

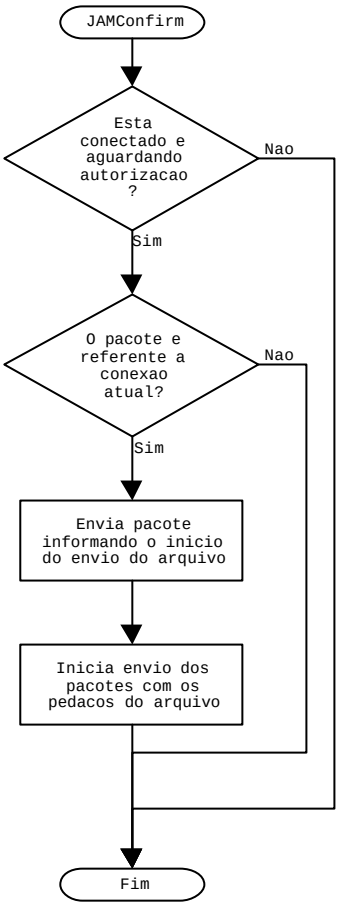


Figura 30 – Processamento do pacote de autorização de inicio de envio de arquivo

Ao receber a autorização para enviar o arquivo, antes de enviar o primeiro pacote referente ao mesmo, a aplicação cliente envia ao servidor um pacote de início de envio de arquivo, mostrado na Figura 31. Ao processar este pacote o servidor sabe que a partir deste pacote começarão a ser enviados os pacotes referentes ao arquivo JAM, conforme mostra a Figura 32.

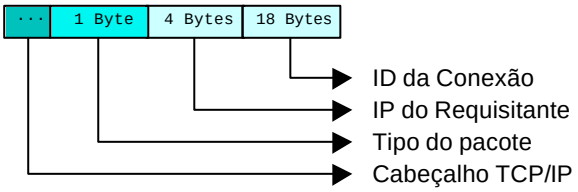


Figura 31 – Inicio do envio de arquivo

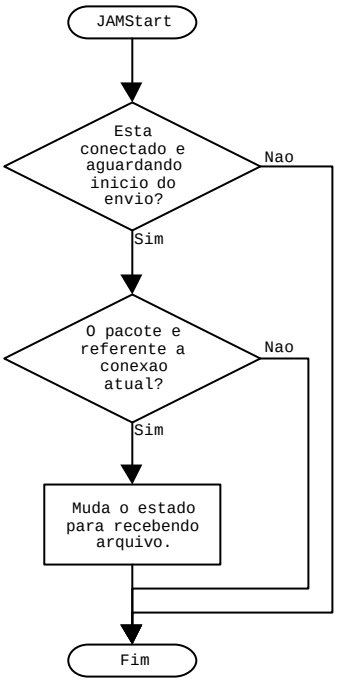


Figura 32 – Processamento do pacote de inicio de envio de arquivo

Para enviar um arquivo JAM ao servidor, o cliente quebra o mesmo em vários pedaços de 255 bytes, ficando apenas o último pedaço com um tamanho menor. O cliente numera de forma seqüencial cada um dos pacotes enviados ao servidor, cuja estrutura é apresentada na Figura 33, permitindo que este verifique se esta recebendo os pacotes esperados. O servidor, por sua vez, ao receber um pacote de envio de arquivo verifica o seqüencial do pacote e caso este confira com o seqüencial esperado pelo servidor então adiciona o mesmo ao final do arquivo, caso se perca um pacote o servidor passará a recusar todos os demais pacotes e por fim recusará o arquivo JAM, este processo é ilustrado na Figura 34.

Nesta primeira versão do protocolo não há um controle para o reenvio de pacotes perdidos, porém a estrutura do protocolo permite que este controle seja implementado no futuro sem grandes alterações.

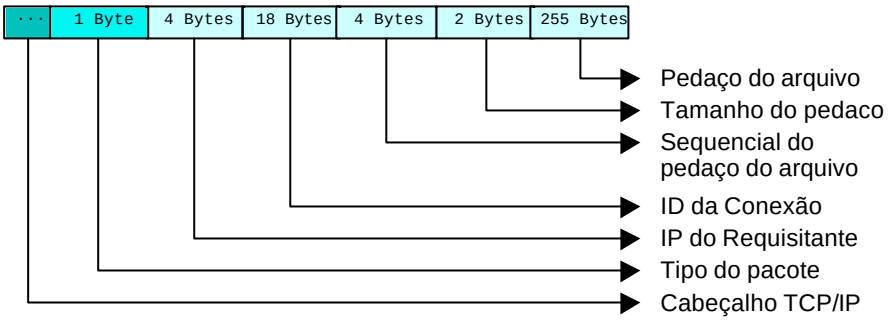


Figura 33 - Envio de arquivo

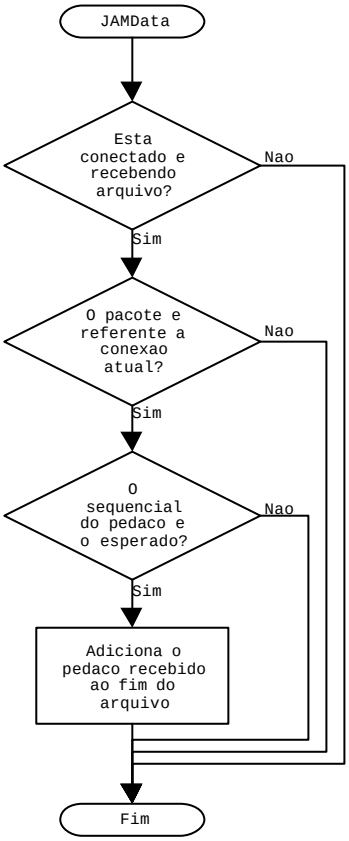


Figura 34 – Processamento de pacote de envio de arquivo

Após enviar o último pedaço do arquivo JAM o cliente envia ao servidor um pacote informando ao mesmo o fim da transmissão do arquivo, incluindo neste pacote o número de seqüência do último pacote enviado, a estrutura do pacote é apresentada na Figura 35. Ao receber este pacote o servidor verifica se a seqüência do último pacote recebido confere com a seqüência do último pacote enviado pelo cliente. Caso confira o servidor marca o arquivo como válido, caso contrário o servidor descarta o arquivo recebido, conforme apresentado na Figura 36.

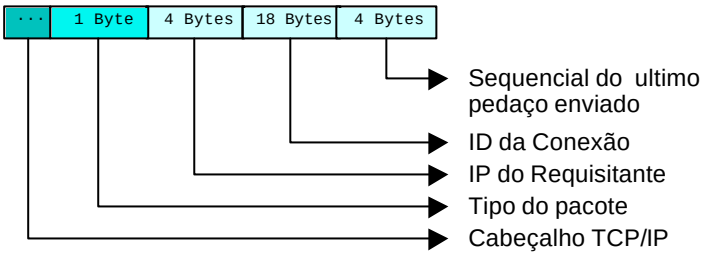


Figura 35 - Fim de envio de arquivo

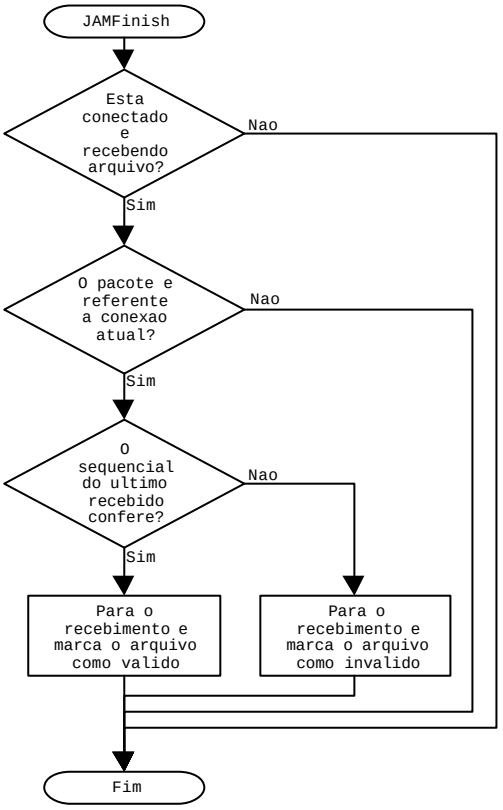


Figura 36 – Processamento do pacote de fim de envio de arquivo

4.1.1.3 Pacotes do console remoto

O console remoto é um retransmissor das saídas do JAM Player embutido na aplicação servidor. Assim há apenas um pacote responsável por enviar essas saídas do servidor para o cliente, mostrado na Figura 37.

Ao receber um pacote deste tipo a aplicação cliente apenas verifica se o pacote é referente à conexão atual e caso seja apresenta o conteúdo do mesmo para a “tela” do console remoto, conforme mostra a Figura 38.

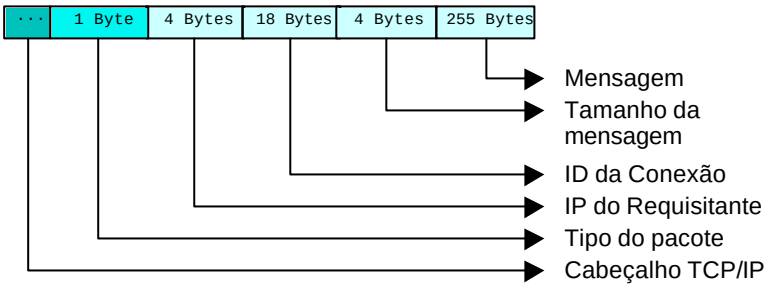


Figura 37 - Mensagem de console remoto

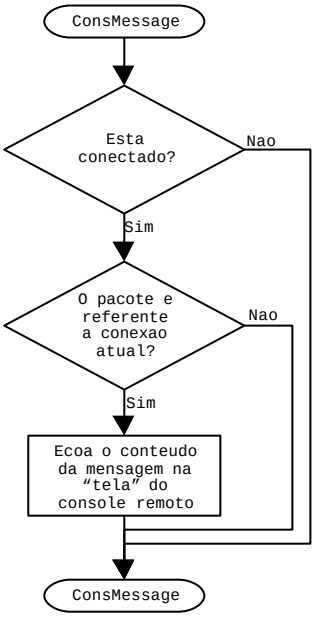


Figura 38 – Processamento de pacote de mensagem de console remoto

4.1.1.4 Pacotes de controle do player

O JAM Player embutido na aplicação servidor pode ser utilizado para duas tarefas. A primeira é configurar a FPGA conectada ao servidor utilizando-se do arquivo JAM Byte Code enviado pela aplicação cliente. A segunda é extrair as informações contidas no arquivo JAM Byte Code. Para que o servidor execute estas ações são utilizados dois pacotes com comandos para o JAM Player, o resultado de ambos os comandos é retornado à aplicação cliente através do console remoto.

Ao receber um comando de configuração, Figura 39, o servidor verifica se possui um arquivo JAM válido, caso possua executa a configuração da FPGA usando o mesmo, caso contrário apenas descarta o pacote, conforme apresenta a Figura 40.

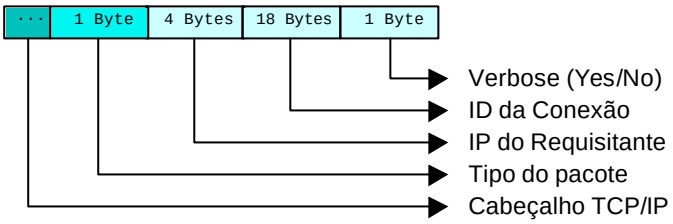


Figura 39 - Comando de configuração

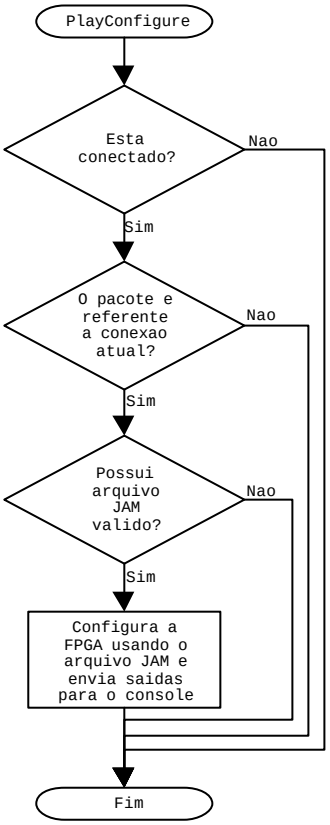


Figura 40 – Processamento do comando de configuração

Ao receber um pacote do tipo de solicitação de informação do arquivo JAM, Figura 41, o servidor verifica se está conectado e se possui um arquivo JAM válido. Caso atenda a estas condições então o servidor executa o JAM Player solicitando a este as informações referentes ao arquivo JAM recebido e envio as saídas resultantes para o console remoto do cliente, conforme apresenta a Figura 42.

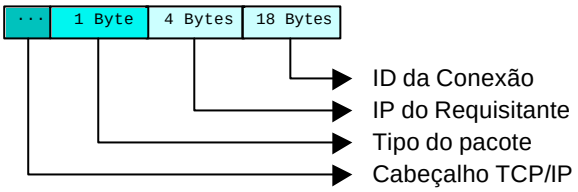


Figura 41 - Comando de informação

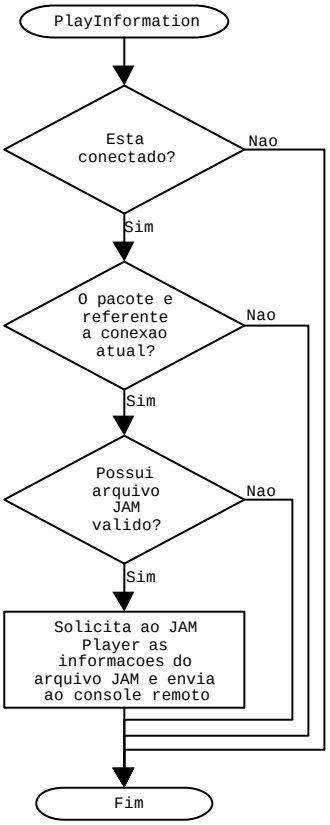


Figura 42 – Processamento do comando de informação

4.1.1.5 Pacotes de controle do monitor

Os pacotes de controle do monitor são utilizados para o envio dos estados dos sinais de saída do Kit Altera, enviados pelo servidor para o cliente, e para o envio das alterações dos sinais de entrada do Kit Altera, pelo cliente. Também há um pacote utilizado para ativação e desativação o monitor on-line.

Ao receber um pacote de solicitação de estados de sinais de controle, Figura 43, o servidor, caso esteja conectado ao cliente solicitante, responde com um pacote de envio de estado dos sinais, conforme mostra a Figura 44.

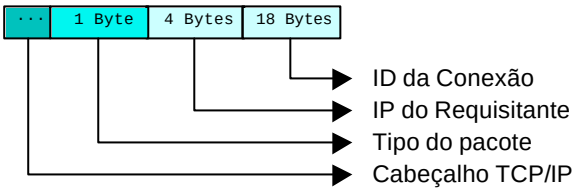


Figura 43 - Solicitação de estados dos sinais de controle

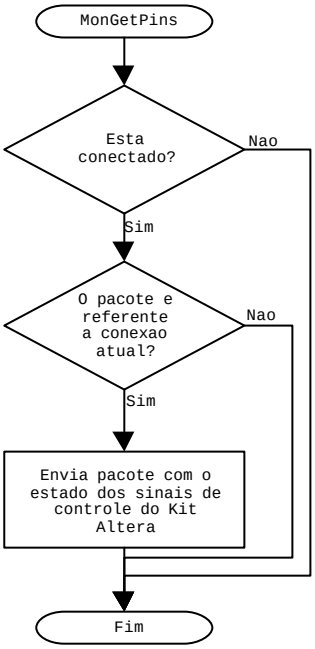


Figura 44 – Processamento do pacote de solicitação de estados dos pinos

Sempre que solicitado, através de um pacote de solicitação de estados dos pinos, o servidor envia ao cliente um pacote com o estado dos mesmos, Figura 45. Este envio também ocorre por parte do servidor, quando este se encontra no modo de monitor on-line e percebe alguma alteração em um dos sinais de controle do Kit Altera. Já, no lado do cliente, o envio dos estados dos sinais de controle ocorre sempre que solicitado pelo usuário da aplicação ou quando o cliente encontra-se no modo monitor on-line e um dos sinais tem seu estado alterado pelo usuário, conforme ilustra a Figura 46.

Ao receber este tipo de pacote o servidor envia ao multiplexador, através da interface serial, os novos valores dos sinais de controle, afim de que este atualize as entradas do Kit Altera. Já no lado cliente, sempre que um pacote deste tipo é recebido, o mesmo é processado para que sejam atualizadas as saídas tanto no monitor dos sinais quanto na interface gráfica do Kit.

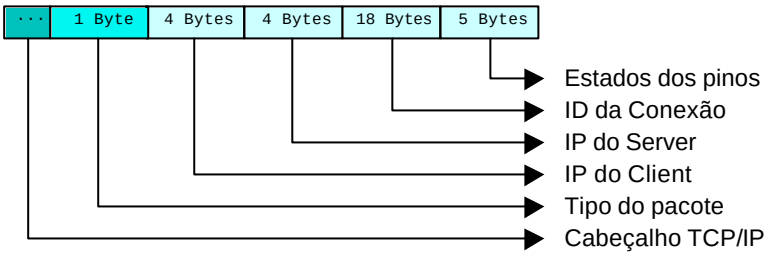


Figura 45 - Envio de estados dos sinais de controle

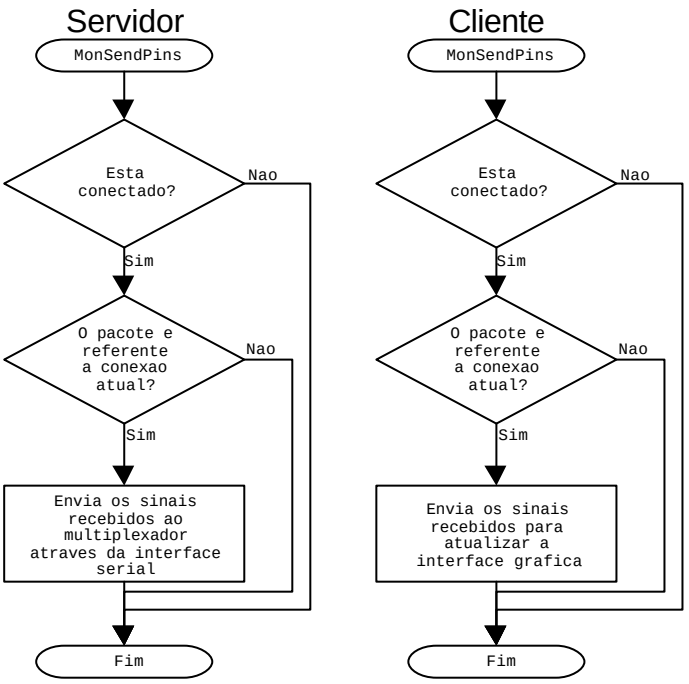


Figura 46 – Processamento do pacote de envio de estados dos sinais de controle

Ao ser ativado ou desativado pelo usuário o modo monitor on-line, o aplicativo cliente envia ao servidor um pacote, Figura 47, informando o mesmo da ação executada e, muda seu estado. Quando recebe este pacote o servidor muda o estado do monitor on-line para o estado solicitado pelo cliente. Quando acionado o monitor on-line, tanto o servidor quanto o cliente enviam automaticamente qualquer alteração dos sinais de controle. O processamento do pacote é apresentado na Figura 48.

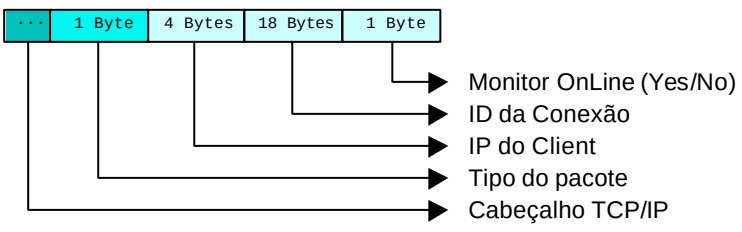


Figura 47 - Controle do monitor on-line

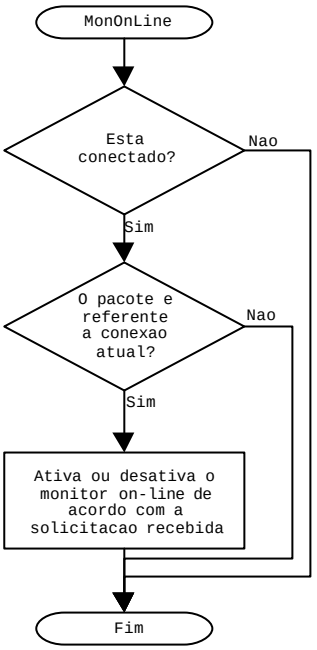


Figura 48 – Processamento do pacote de controle do monitor on-line.

4.1.2 Firmware

O multiplexador dos sinais de controle é composto por um Kit Didático do microcontrolador 8051 e um circuito multiplexador, controlado através do barramento de endereços do Kit 8051, que por sua vez controla também a comunicação serial com o microcomputador onde é executado o aplicativo servidor. Assim o firmware desenvolvido neste projeto, é executado no Kit 8051 e é responsável por controlar o circuito multiplexador e efetuar a comunicação serial com o PC.

O protocolo de comunicação criado sobre o RS232 é bem simples, são possíveis 3 tipos de "pacotes", 2 do PC para o firmware e 1 do firmware para o PC. A estrutura básica dos pacote é apresentada na Figura 49.

A comunicação se inicia com o envio do caractere correspondente ao código 2 da tabela ASCII, este caractere indica o inicio da transmissão do pacote, após ele um byte indicando o tipo do pacote. De acordo com cada tipo de pacote seguem-se "*n*" bytes referentes aos dados enviados, não há nenhum indicador de fim do pacote, uma vez que cada tipo de pacote tem um tamanho fixo pré-definido.

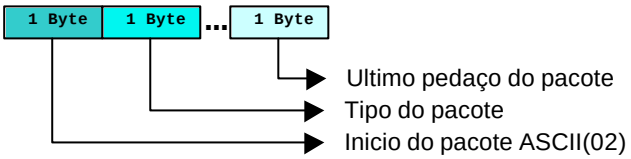


Figura 49 – Estrutura geral dos pacotes da comunicação serial

O byte de tipo de pacote e os bytes de estado dos pinos, possuem nos 6 primeiros bits o valor a ser transmitido e nos 2 últimos o controle de paridade do próprio byte, conforme indicado na Figura 50.

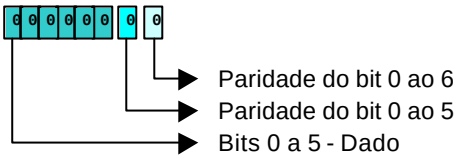


Figura 50 – Estrutura do byte de tipo de pacote

4.1.2.1 Solicitação de estados

Em intervalos constantes de tempo o servidor requisita ao firmware o estado dos sinais de controle do Kit Altera, enviando a este um pacote de solicitação de estados, cuja estrutura é apresentada na Figura 51.

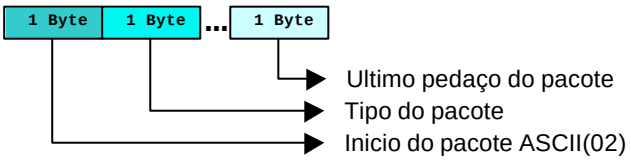


Figura 51 – Estrutura do pacote de solicitação de estados

Sempre que recebe uma solicitação de estados através da interface serial, o firmware lê 3 endereços da memória externa, cada um correspondente a um dos 3 *latchs* de leitura do circuito multiplexador. Após isto o firmware envia os valores dos 3 bytes lidos ao PC através da interface serial, utilizando-se de 4 palavras de 1 byte cada, conforme apresentado na Figura 52.

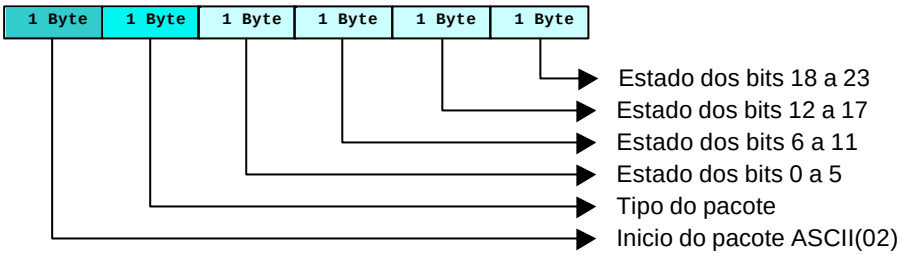


Figura 52 – Estrutura do pacote de envio de estados

4.1.2.2 Atualização de sinais de controle

O servidor também envia ao firmware um comando de atualização dos sinais de controle. Ao receber este tipo de pacote o firmware "grava" em 2 endereços de memória externa, cada um correspondente a um dos 2 *latchs* de escrita do circuito do multiplexador, os valores recebidos. O pacote de envio dos sinais de controle tem a estrutura apresentada na Figura 53.

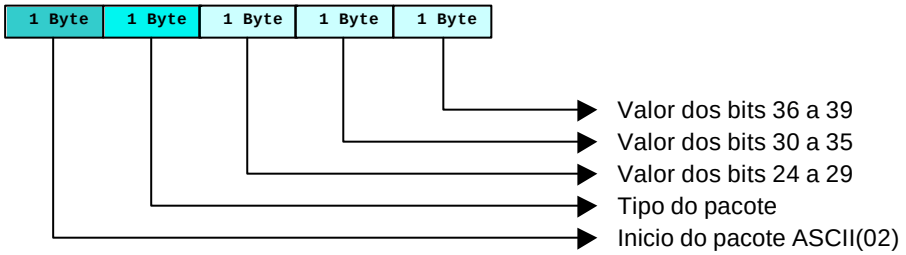


Figura 53 – Estrutura do pacote de atualização dos sinais

No caso do pacote de atualização de sinais os bits 4 e 5, do ultimo byte do pacote, possuem sempre o valor 0 (zero), por não possuírem pinos correspondentes no multiplexador.

4.2 Hardware

Um circuito multiplexador é necessário, para a utilização de 40 sinais de controle, dentre os muitos disponíveis nas expansões do Kit Altera. Este circuito é composto de 5 *latchs*, 3 para a leitura dos sinais do Kit e, 2 para envio de sinais para o Kit. Desta forma ficam disponíveis 24 sinais de saída e 16 sinais de entrada para a construção dos projetos que utilizarem o Kit Altera através deste sistema.

Para habilitar ou desabilitar os latchs utiliza-se o barramento de endereços do Kit 8051, os endereços 0x8000 a 0x8002 referem-se aos *latchs* de leitura, e os endereços

0x8003 e 0x8004 aos *latchs* de escrita. Para simplificar o circuito do multiplexador não foi implementado um decodificador de endereço absoluto, assim o endereçamento dos *latchs*, na pratica, segue o apresentado na Tabela 2.

Bit de endereçamento																***Latch***
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**	
1	X	X	X	X	X	X	X	X	X	X	X	X	0	0	0	**00**
1	X	X	X	X	X	X	X	X	X	X	X	X	0	0	1	**01**
1	X	X	X	X	X	X	X	X	X	X	X	X	0	1	0	**02**
1	X	X	X	X	X	X	X	X	X	X	X	X	0	1	1	**03**
1	X	X	X	X	X	X	X	X	X	X	X	X	1	0	0	**04**

Tabela 2 – Endereçamento dos latchs

As portas de entrada dos 3 latchs de leitura estão conectadas aos pinos referentes aos sinais selecionados para serem utilizados como sinais de saída do Kit Altera, o sinal de controle de habilitação da saída destes *latchs* é controlado pelo circuito decodificador. Para garantir a sincronização com o Kit Altera, o controle de escrita dos *latchs* está ligado ao sinal de *clock* do Kit Altera.

As portas de entrada dos 2 *latchs* de escrita estão conectadas aos sinais selecionados para serem utilizados como sinais de entrada do Kit Altera, a habilitação da saída destes *latchs* foi conectada diretamente ao sinal de terra, e o controle de escrita ao decodificador de endereços, habilitando assim a escrita nos mesmos através do endereçamento.

5 RESULTADOS E DISCUSSÃO

5.1 Protocolo X

A seguir serão apresentados os resultados obtidos nos testes do Protocolo X e discutidos alguns aspectos destes resultados.

5.1.1 Controle de conexão

Para os teste do controle de conexão foram simuladas situações de queda da conexão física entre as partes em qualquer um dos passos da conexão e após o estabelecimento da mesma, a fim de verificar se a aplicação cliente ou a aplicação servidor ficariam presas em algum dos estados intermediários do processo de conexão, ou se responderiam a contento no que tange ao controle de *time-out*.

Nas duas situações o controle de conexão do Protocolo X respondeu conforme esperado, derrubando as conexões em caso de *time-out*, e retornando as aplicações ao estado inicial.

5.1.2 Transferência de arquivo

Nos demais pacotes o Protocolo X cria uma conexão TCP/IP, envia o pacote desejado e derruba a conexão criada. Porém para o envio dos arquivos JAM esta tática apresentou problemas, uma vez que arquivos pequenos geram em media 170 pacotes de envio, o que resultava na criação de 170 conexões TCP/IP.

Assim para o envio dos arquivos JAM foi necessária a implementação de outra abordagem, uma fila de pacotes é criada, com todos os pacotes a serem enviados, começando pelo pacote de inicio de transmissão e sendo finalizada pelo pacote de fim de transmissão.

Quando a construção da fila esta concluída, uma única conexão TCP/IP é criada e todos os pacotes da fila são transmitidos de forma seqüencial. Esta abordagem resolveu alguns problemas de transferência de arquivo e ainda melhorou em muito a velocidade da mesma.

Como teste um arquivo JAM foi criado no software Quartus II e enviado do aplicativo cliente ao aplicativo servidor diversas vezes, afim de validar a confiabilidade da transmissão efetuada.

5.1.3 Controle do player

Para testar os controles do JAM Player embutido na aplicação servidor, um arquivo JAM foi criado no software Quartus II da Altera, configurando o Kit Altera para apresentar um contador de 0 a 9 em um dos displays de 7 segmentos do Kit. Este arquivo foi utilizado para configurar o Kit através do Quartus II, afim de confirmar a validade do mesmo, após isto o arquivo foi levado ao micro onde rodava a aplicação cliente e através desta transferido para o servidor, então o comando de configuração foi enviado ao servidor, o qual, utilizando-se do JAM Player, reconfigurou com sucesso o Kit Altera ligado ao PC.

5.1.4 Console remoto

Durante a implementação do console remoto notou-se a perda de vários pacotes do mesmo, o que resultava em perda de informações do console. Para solucionar o problema foi implementada na API do Protocolo X uma fila de pacotes, assim sempre que um pacote é recebido, ao invés de processá-lo de imediato, a API apenas adiciona o mesmo à fila, e um processo paralelo, implementado através de *Thread*, vai retirando os pacotes da fila e processando os mesmos. Com esta melhoria os pacotes referentes ao console remoto não foram mais perdidos e o mesmo apresentou o comportamento esperado com ótimos resultados e ganho de desempenho.

5.1.5 Monitor on-line

Para os testes do monitor on-line, o mesmo foi ativado através da aplicação cliente e então através de interação com os sinais de controle na mesma verificou-se a resposta a estas interações na interface gráfica da aplicação servidor, demonstrando que esta apresentava o que acontecia no cliente. Para o teste do caminho do servidor para o cliente, foram alterados alguns valores dos sinais diretamente na interface da aplicação servidor e observados os “ecos” destas alterações na aplicação cliente.

5.2 Firmware

Para a validação do firmware do Kit 8051, uma variação do contador utilizado nos testes do Protocolo X foi criada. Nesta variação, o contador é reiniciado quando um dos botões é pressionado, alem de permitir a escolha do display a ser utilizado, através das chaves do tipo DipSwitch.

5.3 Circuito multiplexador

Para os testes do circuito multiplexador foi criado no Quartus II um projeto configurando Kit Altera para enviar continuamente valores fixos para os sinais de saida. Um firmware que lê os latchs do multiplexador e envia os valores dos mesmos para a serial foi então gravado no Kit 8051, assim monitorando os valores recebidos na serial e comparando-os com os valores fixos das saídas do Kit Altera pode-se validar a função de leitura do circuito multiplexador.

Para validar a função de escrita do circuito, um outro projeto criando uma variação do primeiro, ecoando na saída um valor “espelhado” do valor recebido na entrada, permitiu a validação da função de escrita do circuito multiplexador.

6 CONCLUSÕES

6.1 O que não funcionou

Antes de discursar sobre as conclusões e resultados obtidos no decorrer deste projeto, é importante ressaltar que muito se esconde no que não funciona, e muita informação se gera com base no que não funciona.

Em se tratando de pesquisa acadêmica, o produto mais importante desta não é o resultado final obtido pela mesma, mas sim o aprendizado adquirido durante o processo de desenvolvimento desta [SANTOS, 1999].

Dito isto, parece adequado olhar a proposta pretendida quando do inicio dos trabalhos deste projeto. Pretendia-se a construção de um sistema microcontrolado, que incorporasse as funções de servidor para o Protocolo X, dotado de interfaces Ethernet e J-TAG. Pretendia-se ainda construir este sistema com base no Kit Didático do microcontrolador 8051, utilizado pelos alunos do Unicenp, de forma a criar ao fim do projeto uma interface Ethernet e uma API de implementação do protocolo TCP/IP, que pudessem ser utilizados pelos demais alunos do curso de engenharia da computação em projetos futuros.

Já na implementação da API do protocolo TCP/IP apareceram os primeiros problemas ocasionados pela pequena quantidade de memória disponível no Kit 8051, apenas 32 KBytes. Mais adiante, nos primeiros testes do JAM Player para o 8051 novamente a pequena quantidade de memória mostrou-se um grande limitador, uma vez que os projetos mais simples para o chip Flex 10K geravam byte codes de aproximadamente 29 KBytes.

Na tentativa de solucionar o problema de armazenagem do arquivo JAM, foi projetada e construída uma expansão de memória para o Kit 8051, afim de armazenar nesta o arquivo JAM a ser utilizado. Para tanto alterações tiveram de ser feitas no Player disponibilizado pela Altera, o qual utilizava-se de ponteiros para acessar diretamente na memória pedaços do byte code.

Mesmo com a expansão, sendo o byte code um programa compilado que será executado pelo player, este requer uma quantidade de memória para suas variáveis. No caso dos byte codes gerados pelo Quartus para o Flex 10K estes exigiam muita memória, o que inviabilizou o uso do player no Kit 8051.

Uma vez que não era possível executar o byte code no Kit 8051 foi necessário partir para uma abordagem alternativa, optando-se então pelo uso do PC para a execução do mesmo. Construiu-se então a idéia de criar um aplicativo servidor, que fosse responsável pela comunicação com o aplicativo cliente e pela reconfiguração do Kit Altera.

Porém não foi abandonada por completo a idéia inicial de, no futuro, se construir um dispositivo microcontrolado totalmente autônomo, independente do uso de PC. Mas fica claro que este necessita de hardware projetado para tanto, com grande capacidade de memória, utilizando-se de um microcontrolador baseado em 8051, mas com mais recursos do que o utilizado no Kit.

Uma boa abordagem para o problema de endereçar grandes quantidades de memória parece ser o uso de memórias seriais, onde maiores volumes de memória não acarretam em aumento da largura do barramento. Um microcontrolador com mais de uma interface serial, seria nesse caso uma opção interessante.

Também será necessária a revisão e talvez reescrita de boa parte do player do 8051, uma vez que o mesmo utiliza-se massivamente do endereçamento absoluto de memória para implementar algumas de suas rotinas.

6.2 O que funcionou

Mas há que se olhar também aos resultados obtidos, e ao produto final deste projeto. A possibilidade comprovada de se configurar, monitorar e controlar remotamente um hardware, utilizando-se de tecnologia Internet, abre um leque enorme de possibilidades a serem exploradas.

A primeira, proposta inicialmente neste projeto, permite que estudantes testem seus projetos em hardware de verdade, e não apenas em simuladores, sem que estes necessitem estar nas instalações da instituição de ensino.

Mas são muitas e variadas as outras possibilidades a serem exploradas, tais como a atualização de centrais telefônicas globalmente distribuídas, sem a necessidade de deslocamento de técnicos, ou o monitoramento e correção de estações radio base situadas em locais de difícil acesso, de forma facilitada e a um custo reduzido, entre muitas que se pode vislumbrar.

7 REFERÊNCIAS BIBLIOGRÁFICAS

MOKARZEL, M.P.; CARNEIRO, K.P.M.C. *Internet Embedded: TCP/IP para Microcontroladores*. 1.ed. São Paulo, Érica, 2004.

RIBEIRO, A.A.L. *Reconfigurabilidade dinâmica e remota de FPGAs*. São Paulo, USP – São Carlos, 2002.

CARVALHO, T.C.M.B.,org. *Arquitetura de redes de computadores OSI e TCP/IP*. 2. ed. São Paulo, Makron Books, 1997.

COMER, D. *Interligação em rede com TCP/IP*. 3. ed. Rio de Janeiro, Campus, 1998.

TOCCI, R.J. *Sistemas digitais: princípios e aplicações*. 8. ed. São Paulo, Prentice Hall, 2003.

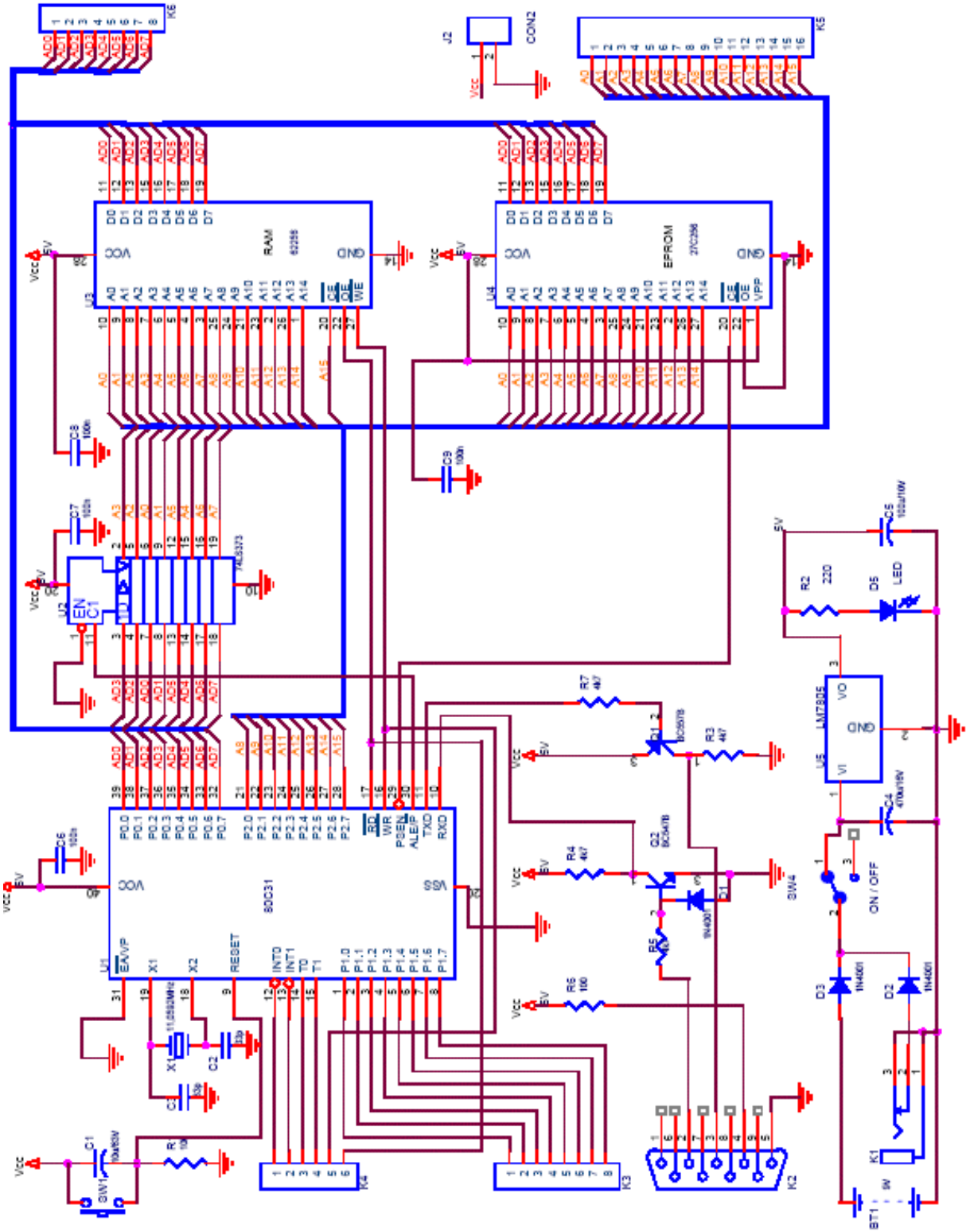
ALTERA CORPORATION. University Program UP2 Education Kit: User Guide. San Jose, 2004.

MICRON TECHNOLOGY INC. 4, 8 MEG x 32 DRAM SIMMs. Boise, 1998.

SANTOS, A. R. *Metodologia científica: a construção do conhecimento*. 2. ed. Rio de Janeiro, DP&A, 1999.

ANEXO I – ESQUEMAS ELÉTRICOS

Kit 8051



Circuito multiplexador