FUNDAÇÃO DE ENSINO “EURÍPIDES SOARES DA ROCHA”
CENTRO UNIVERSITÁRIO “EURÍPIDES DE MARÍLIA” - UNIVEM
FACULDADE DE INFORMÁTICA

FERNANDA MARQUES NALIN

JLODI - UM SISTEMA DIDÁTICO BÁSICO DE LÓGICA DIGITAL PROGRAMADO EM JAVA USANDO PARPORT E FPGAs

MARÍLIA
2005

FERNANDA MARQUES NALIN

JLODI - UM SISTEMA DIDÁTICO BÁSICO DE LÓGICA DIGITAL PROGRAMADO EM JAVA USANDO PARPORT E FPGAs

Trabalho de Conclusão de Curso apresentado a Faculdade de Informática do Centro Universitário Eurípides de Marília, mantido pela Fundação de Ensino Eurípides Soares da Rocha, para obtenção do Título de Bacharel em Ciência da Computação.

Orientador:
Prof. Dr. Edward David Moreno Ordonez

MARÍLIA
2005

FERNANDA MARQUES NALIN

JLODI - UM SISTEMA DIDÁTICO BÁSICO DE LÓGICA DIGITAL PROGRAMADO EM JAVA USANDO PARPORT E FPGAs

Banca examinadora do Trabalho de Conclusão de Curso apresentado à faculdade de Informática da UNIVEM,/F.E.E.S.R., para obtenção do Título de Bacharel em Ciência da Computação. Área de Concentração:

Resultado: _____

ORIENTADOR: Prof. Dr. Edward David Moreno Ordonez

1º EXAMINADOR: Kalinka R. L. J. Castelo Branco _____

2º EXAMINADOR: Marco L. Mucheroni _____

Marília, 29 de novembro de 2005.

Dedico este trabalho, como maior prova de gratidão, aos meus pais queridos, que estiveram presentes em todos os momentos da minha vida e que sempre me incentivaram nos estudos, proporcionando essa minha grande realização pessoal.

Agradeço ao meu orientador por ter tido paciência perante minhas dificuldades, e por ter norteado meu caminho em busca da realização deste trabalho.
A minha irmã pelo apoio e compreensão.
A todos que de alguma forma colaboraram com este trabalho, seja com sugestões, esclarecimentos, incentivo ou ajuda direta como ocorreu em algumas implementações.

NALIN, Fernanda Marques. Um Sistema Didático Básico de Lógica Digital Programando em Java Usando Parport e FPGAs**:** 2005. Trabalho de Conclusão de Curso – Centro Universitário Eurípides de Marília, Fundação de Ensino Eurípides Soares da Rocha, Marília, 2005.

RESUMO

No desenvolvimento de trabalhos em hardware, existem algumas dificuldades na interpretação entre o hardware e a operação aritmética e lógica que ele executa. Para ajudar, desenvolvemos esse projeto, didático, para facilitar o desenvolvimento e entendimento dos trabalhos.

O projeto consiste em propor e desenvolver uma ferramenta em Java chamada de JLODI (Java Lógica Digital), com auxilio da interface Parport que permite interligar um hardware (FPGA) externo a um PC através da porta paralela. Ao mesmo tempo em que o FPGA é visto realmente, é visto também o resultado da operação na interface em Java, proporcionando melhor visualização e entendimento do trabalho em desenvolvimento. No projeto se apresenta um conjunto de operações lógicas e aritméticas que foram implementadas usando VHDL e FPGAs.

S U M Á R I O

1. Introdução

Todo o raciocínio lógico é baseado na tomada de uma decisão a partir do cumprimento de determinadas condições. Inicialmente têm-se os dados de entrada e uma condição (ou uma combinação de condições). Aplica-se a condição aos dados de entrada para decidir quais são os dados de saída. Em lógica digital trabalha-se apenas com variáveis cujos valores alternam exclusivamente entre dois estados e não admitem valores intermediários. Estes estados podem ser representados por "um" e "zero", "sim" e "não", "verdadeiro" e "falso" ou quaisquer outras grandezas cujo valor possa assumir apenas um dentre dois estados possíveis. Portanto, a lógica digital é a ferramenta ideal para trabalhar com grandezas cujos valores são expressos no sistema binário.

Em um computador, todas as operações são feitas a partir de tomadas de decisões que, por mais complexas que sejam, nada mais são que combinações das três operações lógicas correspondentes às condições acima descritas: NOT, AND e OR. Para tomadas de decisões mais complexas, tudo o que é preciso é combinar estas operações. E para isto é necessário um conjunto de ferramentas capaz de manejar variáveis lógicas (Piropo, 2004).

FPGAs (*Field Programmable Gate Array*): são circuitos programáveis compostos por um conjunto de células lógicas ou blocos lógicos alocados em forma de uma matriz. Em geral, a funcionalidade destes blocos assim como o seu roteamento são configuráveis por *software*. Dispositivos FPGA não possuem registradores, ULA, Unidade de Controle ou periféricos convencionais.

Um FPGA é constituído por milhares de portas lógicas desconexas, as quais serão "interligadas" para formar a lógica necessária. Permite desenvolver circuitos de lógica digital complexa. Dispositivos FPGA não possuem registradores, ULA, Unidade de Controle ou periféricos convencionais.

É possível projetar circuitos e sistemas digitais usando FPGAs. São usados em computadores para interfaces de memória, controladores DMA, controladores de memória cachê, co-processadores SSP (Standard Parallel Port), multimídia, gráficos, e em periféricos para controladores de disco, controladores de vídeo, FAX, máquinas de caixa, modens, cartões de aquisição de dados, terminais, impressoras, scanners e copiadoras, entre outros.

Sistemas Distribuídos (SDs) é a coleção de computadores independentes que são vistos pelo usuário do sistema como um único computador, ou seja, é a união das tecnologias de máquinas potentes conectadas através de redes de alta velocidade. O principal conceito de um Sistema Distribuído é permitir o compartilhamento de máquinas e informações de forma conveniente, ele permite compartilhar periféricos, por exemplo uma placa de FPGA (Tanenbaum, 2003).

Java atualmente é uma linguagem de programação de alto nível muito usada no desenvolvimento de ferramentas computacionais e sistemas distribuídos.

Os sistemas Java geralmente consistem em várias partes: um ambiente, a linguagem, a interface de programas aplicativos.

O processo de compilação de um programa de linguagem de alto nível em linguagem de máquina pode tomar uma quantidade considerável de tempo de computador. Java também lida com *applets* e aplicativos que podem se comunicar através de redes. Na comunidade Java foram desenvolvidos programas interpretadores para executar programas em linguagem de alto nível diretamente, sem a necessidade de compilar os programas para linguagem de máquina. Programas interpretados são populares em ambientes de desenvolvimento de programas em que os programas são recompilados freqüentemente enquanto novos recursos são adicionados e erros são corrigidos.

Uma vez que um programa é desenvolvido, uma versão compilada pode ser produzida para executar mais eficientemente, apesar de que os programas compilados executem mais rápidos que os programas interpretados. Os interpretadores têm desempenhado um papel

importante em ajudar Java a alcançar seu objetivo de portabilidade entre uma grande variedade de plataformas (Deitel, 2003).

Parport (*Linux Parallel Port):* é um pacote voltado para soluções em comunicações através da porta paralela de PCs com diversos outros dispositivos e equipamentos. Essa comunicação pode utilizar Java, entretanto, Parport emprega a linguagem C para a comunicação direta com a porta paralela. Não é necessário que o projetista programe em C, pois este aspecto já vem resolvido pelo próprio pacote Parport (Rossi, 2004).

1.1 JUSTIFICATIVA

Muitas ferramentas de simulação de circuitos e sistemas digitais permitem somente fazer a simulação lógica e/ou elétrica do projeto. O usuário fica sempre com a idéia que isso pode não acontecer na vida real (em circuitos e placas).

A ferramenta de circuitos programáveis para FPGAs da família Xilinx permite fazer simulações e protótipos dos FPGAs.

Não obstante, para ajudar pessoas com dificuldades de interpretação de operações em circuitos e sistemas digitais, ao invés de um usuário programar "olhando" diretamente para a placa de FPGA ele terá a interface da mesma direto na tela do computador, sendo mais fácil de visualizar possíveis erros, e a respectiva lógica do programa.

1.2. OBJETIVOS

O projeto consiste em melhorar a visualização da lógica digital, criando uma interface em Java que mostra a simulação de cálculos aritméticos e/ou lógicos no FPGA de forma mais clara, sendo mais fácil a interpretação do resultado de uma determinada operação, que é realizado

pelo circuito que foi mapeado no FPGA, isto é, o resultado é calculado pelo hardware que é enviado à interface JLODI através da porta paralela usando a ferramenta *parport.*

O FPGA será utilizado na parte externa do computador, que é ligado na porta paralela. O parport é um software que facilita a entrada e saída de dados pela porta paralela. A interface em Java importa o parport para que possa receber e enviar os dados da porta paralela, como mostra a figura 1.1.

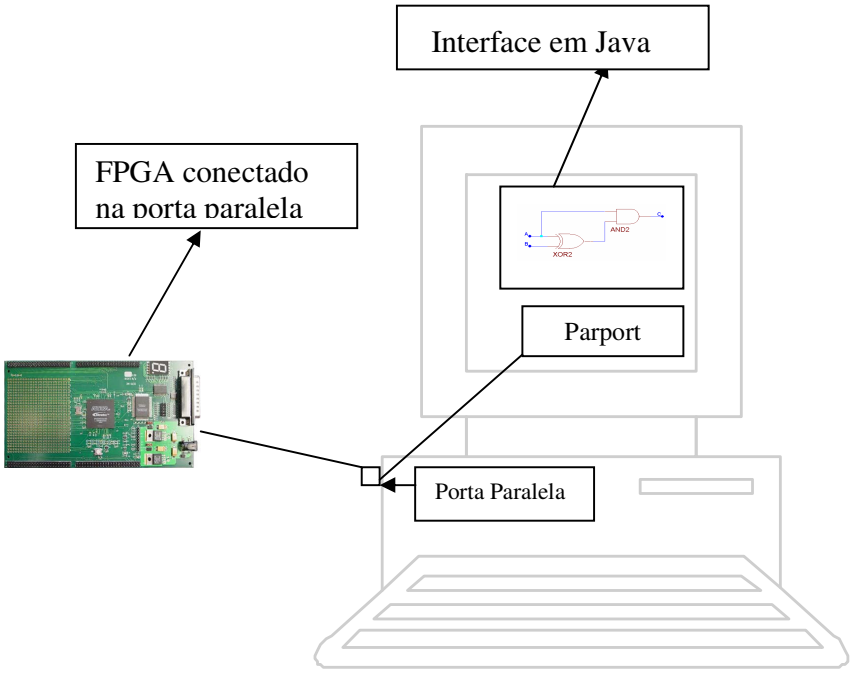

Figura 1.1 – Representação Física do projeto

1.3 ATIVIDADES REALIZADAS.

O trabalho foi realizado através das seguintes atividades:

1 - Conhecimento do assunto e entendimento da proposta;

2 - Levantamento do problema;

3 - Estudos de ferramentas que conectam FPGAs e PC, na porta paralela e/ou serial;

4 - Conhecer e instalar uma ferramenta de portas (Parport).

5 – Estudar o processo de desenvolvimento de interfaces em Delphi e Java.

6 - Apresentar proposta inicial para banca

7 - Teste inicial básico: Java – Parport – FPGA

8 - Melhorar a interface usando Java

9 - Mais testes de circuitos lógicos

10 - Apresentação do relatório final

1.4. TRABALHOS CORRELATOS

Uma breve descrição de trabalhos correlatos pesquisados com importância na área de ferramentas de desenvolvimento de hardware é feito a seguir:

O "*Xilinx Ise Project Manager"* é um ambiente integrado de projetos de circuitos digitais. Inclui, entre outras ferramentas de projeto um Editor de Esquemas, um Simulador Lógico (externo) *Modelsim* e programas para mapeamento automático do circuito digital em dispositivos lógicos programáveis da família Xilinx (Albuquerque, 2001).

Um outro ambiente é o software Max-Plus II da Altera, que é um ambiente desenvolvido para a prática de tecnologia de lógica programável. Nele é possível criar, editar e simular esquemas com circuitos lógicos e programar componentes. A interface com o usuário é de fácil aprendizado (Montebeller, 2005).

O objetivo é basicamente o custo, o qual tem a ver com tempo, confiabilidade e precisão, mas os simuladores eletrônicos também podem ser usados como ferramentas auxiliares e junto ao professor e aluno no estudo da eletrônica. Para ensinar a um aluno como ligar um instrumento (amperímetro ou voltímetro), devemos ir ao laboratório e na bancada montar um circuito para efetuar as medidas. Ao invés disso, pode-se ligar o computador e usar um simulador com o qual pode-se mostrar ao aluno como fazer sem risco algum (queimar o equipamento, curto circuito, etc) e depois de ter treinado no simulador o aluno tendo ganhado confiança poderá usar os instrumentos com segurança (Montebeller, 2005).

Ao invés de fazer todo esse processo de simulação este projeto propõe uma ferramenta onde o usuário digite os operadores e selecione uma operação aritmética e/as lógicas e “pede” ao sistema para executar. A operação desejada será reeditada e executada por um hardware real (FPGA), que enviará o resultado ao PC e a ferramenta JLODI fará toda a parte de comunicação e avaliação do processo todo.

1.5 ORGANIZAÇÃO DO TRABALHO

O relatório final desta monografia está composto por 5 capítulos. O primeiro apresenta os assuntos a serem abordados, a detecção de um problema, o objetivo do projeto, e as atividades realizadas para o desenvolvimento.

No Capítulo 2 descreve-se as ferramentas usadas no projeto e a comunicação entre elas, contendo um exemplo.

No Capítulo 3 é definida a interface indicando qual ferramenta foi usada no projeto de alguns circuitos básicos de lógica digital implementados em VHDL e FPGAs. Também aborda a ferramenta didática, que de inicio fez uso do Delphi e depois usou-se a linguagem Java.

Finalmente o Capítulo 4 é descrita a conclusão e depois se apresentam as referências bibliográficas.

2. FERRAMENTAS PARA COMUNICAÇÃO DAS PORTAS COM FPGA

2.1 FPGA (*Field Programmable Gate Array*)

FPGAs são circuitos programáveis compostos por um conjunto de células lógicas ou blocos lógicos alocados em forma de uma matriz. Em algumas arquiteturas, os blocos lógicos possuem recursos seqüenciais tais como flip-flops e/ou registradores. Cada fabricante nomeia seu bloco lógico, podendo haver mais de um nome para um mesmo fabricante.

Em 1985 surgiu o primeiro FPGA disponível comercialmente. Foi desenvolvido pela companhia Xilinx, desde então vários dispositivos têm sido desenvolvidos por diversos fabricantes, trazendo um baixo custo desses dispositivos programáveis de alta capacidade.

Os FPGAs além de proporcionar um ambiente de trabalho simplificado e de baixo custo, possibilita operar com um número ilimitado de circuitos através da configuração do próprio dispositivo. Os blocos e seu roteamento são configuráveis via software (ORDONEZ, 2003).

2.2 Arquitetura de um FPGA

Existem várias arquiteturas de FPGAs comercialmente disponíveis, por aproximadamente 10 fabricantes. Três aspectos principais definem a arquitetura de um FPGA:

- tipo de tecnologia de programação;
- arquitetura das células; e
- estrutura de roteamento.

Um FPGA é composto por CLBs, IOBs e SBs; conforme se observa na figura 2.1:

- CLBs (*Configurable Block Logic*): Blocos lógicos configuráveis.

- IOBs (*In/Out Blocks*):Blocos de entrada e saída.

- SBs(*Switch Box*): caixa de conexão.

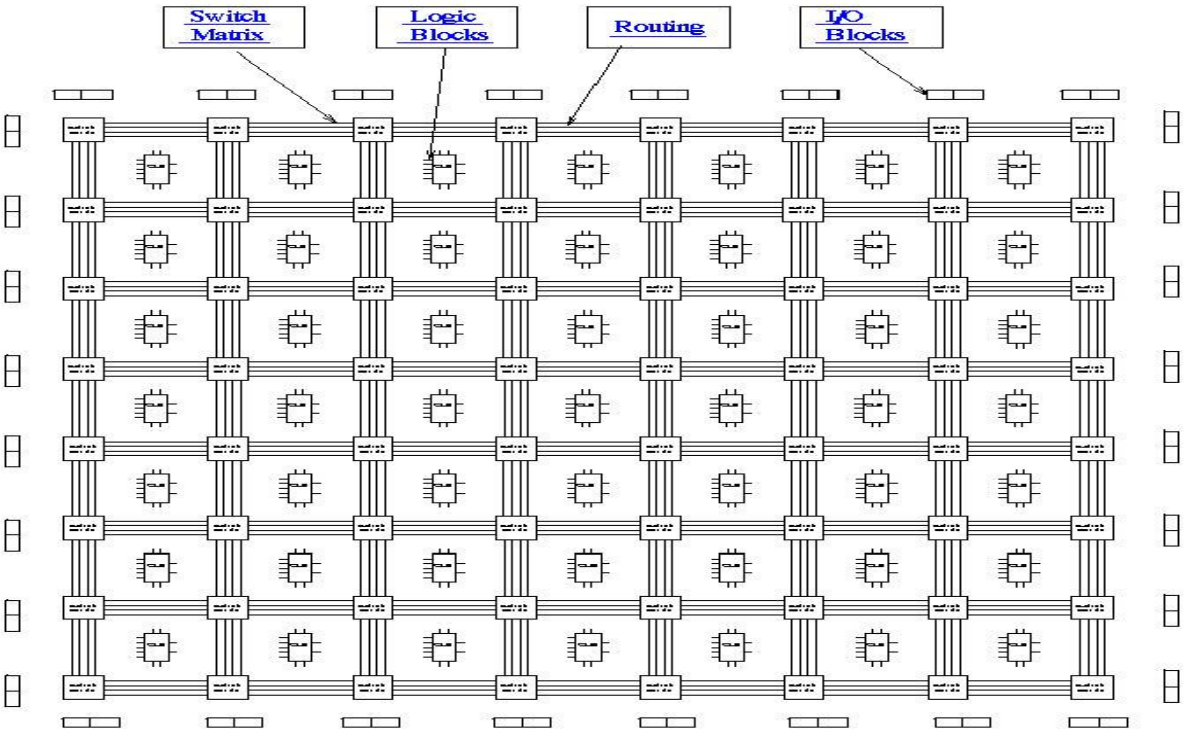

Figura 2.1 - Estrutura de uma FPGA (SILVA, 2003).

CLBs são unidades lógicas das FPGAs. Um exemplo de número de CLBs em uma FPGA é o da Xilinx XC4000 com 100 CLBs. Eles possuem 3 geradores de função combinacional, 2 flip-flops tipo D, lógica de controle, lógica aritmética dedicada. Cada bloco pode ser configurado como um somador de 2 bits. Na figura 2.2 está a representação de um CLB da família XC3000.

A quantidade de portas lógicas disponíveis num FPGA tem aumentado muito nos últimos anos, assim facilitando a implementação de arquiteturas cada vez mais complexas.

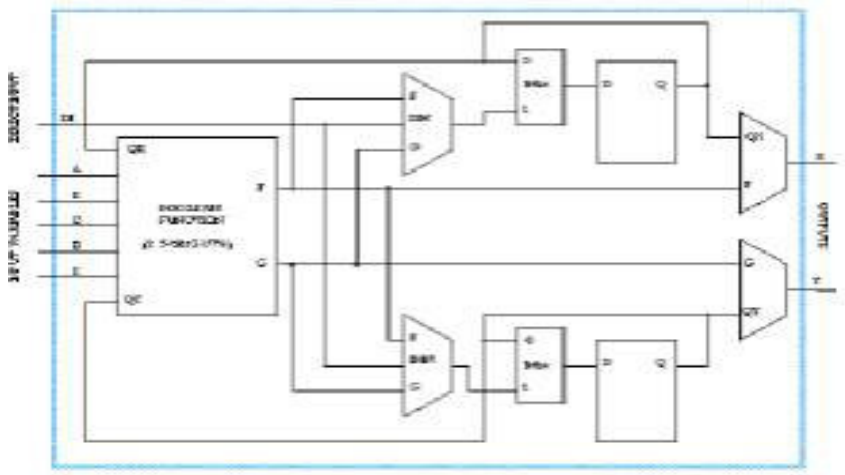

Figura 2.2 - Representação de um CLB da família XC3000 (SILVA, 2003).

IOBs fazem a interface de uma FPGA com o resto do sistema. Por exemplo, para uma FPGA da família XC4003 da Xilinx existem 80 IOBs e eles se localizam na periferia do *chip*. SBs têm a finalidade de fazer a interconexão entre os CLBs através dos canais de roteamento.

A interconexão entre os blocos é feita através de uma rede de duas camadas de metal.As conexões físicas entre os fios são feitas ora com transistores de passagem controlados por bits de memória (PIP) ora com chaves de interconexão (Switch Matrix).

As conexões podem ser realizadas por conexões globais, conexões diretas, linhas longas ou matrizes de conexão (Switch Matrix), conforme é mostrado na Figura 2.3.

Os blocos lógicos dos FPGAs variam muito de tamanho e capacidade de implementação lógica. A construção dos blocos lógicos pode ser tão simples como um transistor ou tão complexo como um microprocessador. Geralmente são capazes de implantar uma ampla variedade de funções lógicas combinacionais e seqüenciais.

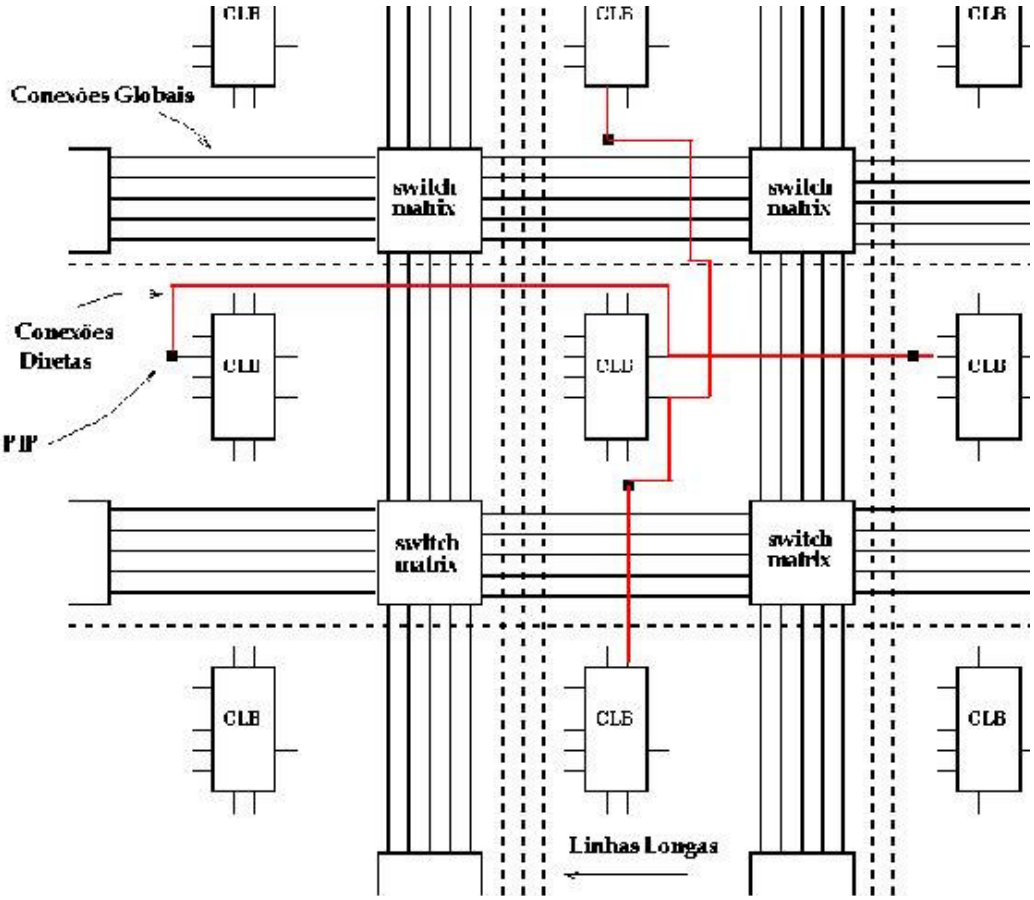

Figura 2.3 - Possíveis conexões internas de um FPGA [SILVA, 2003].

Os blocos lógicos dos FPGAs comerciais são baseados em um ou mais dos seguintes componentes:

- pares de transistores;
- portas básicas do tipo NAND ou XOR de duas entradas;
- multiplexadores;
- Look Up Tables (LUTs);
- estruturas AND e OR de múltiplas entradas; e
- *flip-flops* associados com diversos tipos de portas lógicas.

A fim de classificar os FPGAs quanto à capacidade lógica dos blocos lógicos, pode-se dividi-los em três categorias de granularidade: fina, média e grossa. A primeira categoria designa

os blocos simples e pequenos, a segunda os blocos mais complexos e a terceira, sistemas completos em um único bloco lógico (RIBEIRO, 2002).

Arquitetura de Roteamento: a arquitetura de roteamento de um FPGA é a maneira pela qual os comutadores programáveis e segmentos de trilhas são posicionados para permitir a interconexão dos blocos lógicos (SALCIC, 2000).

As arquiteturas de roteamento podem ser descritas a partir de um modelo geral, conforme a figura 2.4. São necessários alguns conceitos para um melhor entendimento desse modelo:

- pinos: são as entradas e saídas dos blocos lógicos, é válido ressaltar que estes pinos são internos aos FPGAs (conforme indicado na figura 2.4) e não devem ser confundidos com os pinos externos do encapsulamento que são ligados aos blocos de I/O;

- conexão: é a ligação elétrica entre um pino e um segmento de trilha, as conexões são realizadas pelo bloco de conexão;

- bloco de conexão: é o dispositivo utilizado para conectar eletricamente um pino e um segmento de trilha, estes dispositivos utilizam alguma tecnologia de programação.

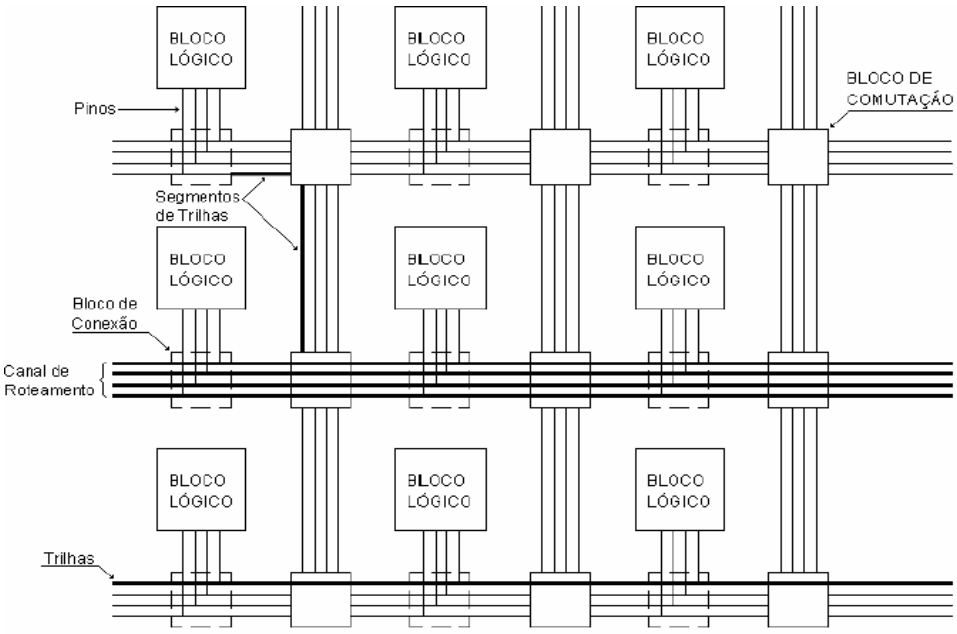

Figura 2.4 – Arquitetura de Roteamento de um FPGA [Ribeiro, 2002].

- segmentos de trilha: são os fios entre dois blocos de comutação;

- trilhas: é uma seqüência de um ou mais segmentos de trilha em uma direção, estendendo-se por todo o comprimento de um canal de roteamento, pode ser composta de segmentos de tamanhos variados;

- bloco de comutação (*switch*): é o dispositivo utilizado para conectar eletricamente dois segmentos de trilha, estes dispositivos também utilizam alguma tecnologia de programação;

- canal de roteamento: é a área entre duas linhas ou colunas de blocos lógicos, sendo que um canal é formado por várias trilhas paralelas.

O modelo da figura 2.4 contém duas estruturas básicas. A primeira é o bloco de conexões que aparece em todas as arquiteturas. O bloco de conexão permite a conectividade das entradas e saídas (pinos) de um bloco lógico com os segmentos de trilhas nos canais. A segunda estrutura é o bloco de comutação que permite a conexão entre os segmentos de trilhas horizontais e verticais. Em algumas arquiteturas, o bloco de comutação e o bloco de conexões são distintos; em outras, estão combinados numa mesma estrutura. Nem todas as arquiteturas seguem esse modelo, pois há arquiteturas que apresentam uma estrutura hierárquica e outras que possuem somente canais horizontais.

Segundo Ribeiro (2002), uma importante questão a ser considerada é se a arquitetura de roteamento permite que se alcance um roteamento completo e, ao mesmo tempo, uma alta densidade lógica. Sabe-se que usando um grande número de comutadores programáveis torna-se fácil alcançar um roteamento completo, mas esses comutadores consomem área, que é desejável minimizar. Algumas pesquisas estabelecem uma relação entre a flexibilidade da arquitetura de roteamento, a capacidade de roteamento e uso eficiente da área.

2.3 Programação de um FPGA

O FPGA pode ter seu comportamento descrito através de diagramas esquemáticos, em linguagem de *Hardware* (HDL – *Hardware Description Language*) e/ou diagrama de fluxo, com a máquina de estados. Essa linguagem é própria para modelar a estrutura e/ou comportamento de um hardware.

Algumas ferramentas para essa programação, como por exemplo a Xilinx, possibilitam a implementação do projeto em alto nível físico, assim o circuito programável assume o comportamento descrito no projeto.

Os FPGAs da Xilinx: a Xilinx, fundada em 1984, se tornou a primeira fabricante de FPGAs do mercado, lançando em 1985 a família XC2000. A cada ano ela oferece novas gerações, introduzindo inovações tecnológicas que aumentam a capacidade lógica e a velocidade dos dispositivos, que já ultrapassam o patamar de 10 milhões de portas lógicas (RIBEIRO, 2002).

A estrutura de roteamento, os blocos lógicos e a forma de programação variam. Dependendo da aplicação, uma arquitetura de FPGA pode ter características mais desejáveis para a aplicação em questão.

A estrutura básica das principais famílias de FPGAs da Xilinx é baseada em *array* e é conhecida como LCA (Logic Cell Array). Cada *chip* inclui um arranjo 2-D de blocos lógicos que podem ser interconectados através de canais horizontais e verticais (ARAGÃO, 1998).

Neste trabalho de TCC descreve-se somente a família usada no projeto e por sinal a mais popular, a XC4000, que não possui capacidade de reconfiguração dinâmica (ARAGÃO, 1998). A figura 2.5 mostra o bloco lógico da família XC4000.

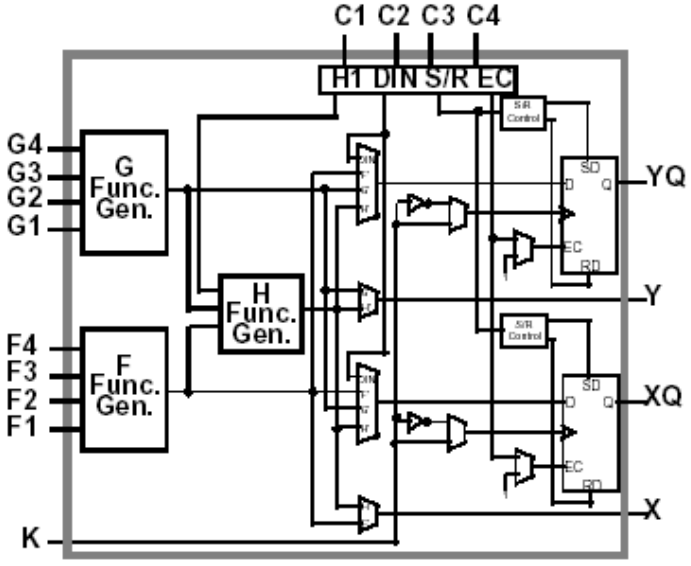

Figura 2.5 - Bloco Lógico da família XC4000 [Aragão, 1998].

O bloco XC4000 da Xilinx incorpora diversas características com o objetivo de prover dispositivos de alta capacidade que suportem a integração de um sistema inteiro. Os *chip* XC4000 têm características orientadas a sistemas, como por exemplo, circuitos que permitem eficientemente a realização de operações aritméticas. Também, cada *chip* XC4000 inclui um plano de portas AND de várias entradas na periferia do arranjo de blocos lógicos, para facilitar a implementação de circuitos como decodificadores.

Os recursos de roteamento da série XC4000 são arranjados em canais verticais e horizontais. Cada canal contém uma quantidade de segmentos de trilhas que se estendem por apenas um CLB, segmentos duplos que atravessam dois CLBs e segmentos longos que atravessam a largura ou altura inteira do *chip*. A vantagem de segmentos maiores é que oferece um caminho de menor resistência em série ao sinal que passa. Além disso, utiliza menos comutadores e células programáveis melhorando a densidade do FPGA.

Um importante ponto a ser notado é que os sinais têm que passar por comutadores para alcançar um CLB, e o total de comutadores que são atravessados depende do conjunto particular de segmentos utilizados. Assim, o desempenho de um circuito implementado depende em parte de como os segmentos de trilha são alocados aos sinais individuais pelas ferramentas EDA (ARAGÃO, 1998).

2.4 Ferramentas Para Comunicação Via Portas do PC

O programa *Userport*, permite o uso do software de programação do µDX - PG - em ambiente *Windows XP*. O sistema operacional *Windows XP* bloqueia o acesso direto à porta paralela do computador. O programa *Userport* desbloqueia este acesso, permitindo a comunicação DXNET. O usuário pode habilitar o endereço I/O que precisa utilizar, no caso da porta paralela (DEXTER, 2005).

Aciona (Placa para Acionamento e Controle de Máquinas através da Porta Paralela do PC). Para que o software de controle da Porta Paralela execute em plataforma, *Windows 2000 e XP* é necessário instalar alguns arquivos que serão enviados juntamente com a placa de

acionamento. A interface permite o controle pela porta paralela de máquinas com interfaces em Delphi, Visual Basic (VB), C e também através do PowerPoint que, juntamente com o software ***Aciona*** desenvolvido pela Níquel, possibilita a automação de máquinas, robôs ou qualquer outro dispositivo que desejar a partir da interface de controle via porta paralela

A seguir descreve-se os passos necessários para criar uma apresentação que irá comandar o acionamento de até 8 relês através da interface de controle.

1. Abrir o Power Point e crie uma apresentação, inserindo os textos e imagens que desejar.

2. Envie um botão de ação conforme indicado na figura 2.7. Este botão acionará o aplicativo que controlará a **interface.**

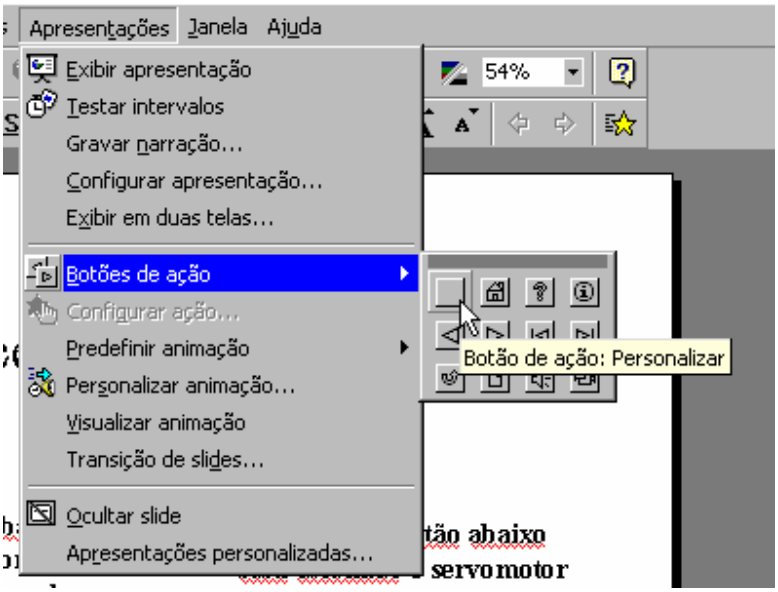

Figura 2.7 – Configurar Botões

3. Posicione e dimensione o botão criado na posição desejada e clique com o botão direito, sobre o botão, abrindo a caixa *Configurar Ação* onde deve ser ativada a opção Executar programa. Ver na figura 2.8. Passe o parâmetro c:\Aciona 1 2 50 .

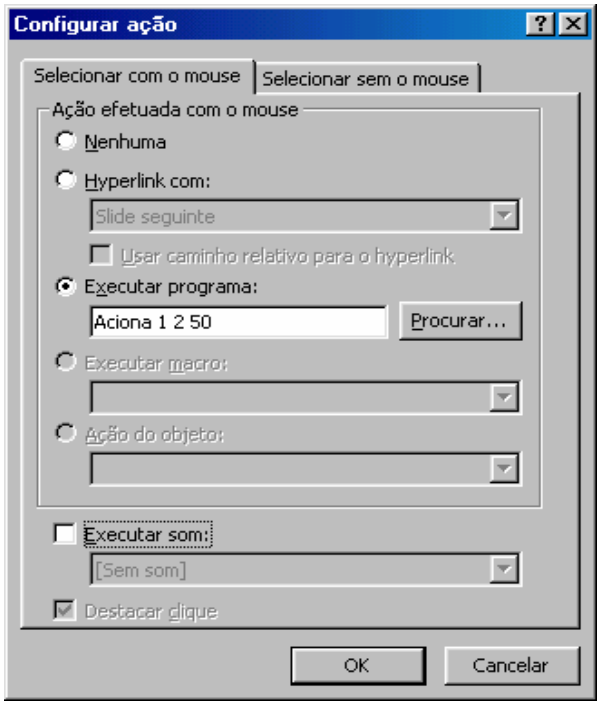

Figura 2.8 – Configuração da ação

Caso esteja utilizando o Office 2000, entrar no menu ferramentas do PowerPoint na opção Macro - Segurança. Mudar o nível de segurança para "baixo" senão o sistema sempre perguntará se deseja mesmo executar o arquivo (pois ele "acha" que pode ser vírus) a cada vez que se deseja clicar em um botão de ação.

Executar a apresentação e clicar obre o botão criado. A saída 2 deverá ser ligada por 5 segundos e então desligada.

Quanto à linha de comando digitada, seus parâmetros têm o seguinte significado:

Aciona A B C

- parâmetro A igual à 1 significa liga e igual à 0 significa desliga.
- parâmetro B é o número da saída podendo ser de 0 á 7
- parâmetro C é o tempo de acionamento em décimos de segundo:

Ex: C= 3 - Tempo = 0,3 segundos

C= 50 - Tempo = 5 segundos

C= 800 - Tempo = 80 segundos

Obs: tempo máximo digitado = 999.
tempo = 0 significa simplesmente ligar ou desligar a saída.

Selecionar a função para cada botão criado e sua apresentação estará pronta para comandar seus dispositivos. A apresentação do *PowerPoint* e o programa *Aciona* devem estar no mesmo diretório. Este aplicativo executa somente em *Windows 95/98* ou anterior.

Neste TCC, a ferramenta de comunicação utilizada é o *Parport* que será explicada na seção seguinte. Ainda assim, é bom salientar que a ferramenta *Userport* e *Aciona* podem ser úteis em alguns sistemas onde o *Parport* não funciona (por problemas de configuração do sistema operacional *Windows*) sendo necessário o uso do *Userport* que libera a porta paralela para que o *Parport* utilizar.

2.5 Ferramenta Parport

Linux Parallel Port (Parport) é um pacote voltado para soluções em comunicações através da porta paralela de PC com diversos outros dispositivos e equipamentos. Essa comunicação utiliza Java, entretanto, *Parport* emprega a linguagem C para a comunicação direta com a porta paralela. Não é necessário que o projetista programe em C, pois este aspecto já vem resolvido pelo próprio pacote. O Pacote pode ser instalado tanto em sistemas operacionais *Windows* quanto *Linux* (ROSSI, 2004).

2.5.1 Funcionamento do Parport

O *Parport* faz a comunicação entre dois computadores que não possuem placas de rede, que desejam compartilhar arquivos ou até mesmo conexão com a Internet. Por isso, utilizar a porta paralela seria a melhor opção. A taxa de transferência é razoável (cerca de 150kbps no modo normal e pode chegar entre 2 ou 6Mbps usando EPP/ECP, enquanto um modem atinge 56kbps).

2.5.2 Instalação

a) Descarregar o arquivo *parport-win32.zip,* este arquivo pode ser obtido em http://www.geocities.com/Juanga69/parport/

b) Descompactar o arquivo num diretório do seu PC, por exemplo: C:\j2sdk1.4.0\parport. Dentro da pasta criada são disponibilizados o arquivo *parport.dll*, os códigos fonte em C, alguns exemplos e um tutorial básico;

c) Copiar o arquivo *parport.dll* em C:\j2sdk1.4.0\bin;

d) O caminho adequado deve ser criado, incluindo no arquivo *autoexec*. A variável de ambiente *classpath* (*Windows*): set CLASSPATH=c:\j2sdk1.4.0;%CLASSPATH%;

e) Reinicie o computador. O sistema ficará pronto para executar programas que utilizam a porta paralela do PC, para operações de escrita e leitura;

e) Caso o sistema operacional usado seja *Windows® NT/2000/XP*, o aplicativo *Userport* deve ser instalado a fim de liberar a porta paralela para usuários.

2.5.3 Userport

Este projeto foi feito em um computador que possui o *Windows XP* como Sistema Operacional, sendo necessário então o uso do *Userport*.

Por motivos de segurança, no modo usuário os sistemas operacionais *Windows NT, 2000 e XP* restringem o acesso às portas de I/O de um PC. Em 2001, foi criado por Thomas Franzon um aplicativo de fácil utilização chamado *Userport* que possibilita o acesso a essas portas. Assim, o usuário pode habilitar o endereço I/O que precisa utilizar. No caso deste projeto a porta paralela. Os endereços dos registradores para LPT1 é 0x37A.

A seguir uma descrição dos passos da instalação do *Userport*:

a) Descarregar o arquivo *Userport*.zip do site: http://www.geocities.com/Juanga69/Parport;

b) Descompactar o arquivo num diretório do PC, por exemplo: C:\j2sdk1.4.0*userport*;

c) Copiar o arquivo *Userport.sys* em ...\System32\Drivers;

d) Executar o programa e configurá-lo;

Após o aplicativo ter sido corretamente instalado é suficiente executar o arquivo executável e configurar o endereço 0x37A da seguinte forma:

a) Na janela esquerda digitar o valor 0x378-0x37A e pressionar o botão *"Add"*. Posteriormente, remover os endereços *"default":* 200-37F, 3BC-3BF e 3E8-3FF, usando *"Remove";*

b) Na janela direita digitar o valor 0x378-0x37A e pressionar o botão *"Add"*, logo depois, remover os outros endereços;

c) Pressionar o botão *"Start"*.

2.6 Comunicação com a Porta Paralela

A porta paralela é uma interface de comunicação entre o computador e um periférico. Quando a IBM criou seu primeiro PC (Personal Computer) ou Computador Pessoal, a idéia era conectar a essa Porta uma impressora, mas atualmente, são vários os periféricos que utilizam-se desta Porta para enviar e receber dados para o computador (exemplos: Scanners, Câmeras de vídeo, Unidade de disco removível e outros) (MESSIAS, 2005).

2.6.1 Funcionamento da Porta Paralela

A Porta Paralela é composta por 3 diferentes seções :

- Data Lines ou Linhas de Dados

- Control Lines ou Linhas de Controle

- Status Lines ou Linhas de Estado

Existem 8 linhas de dados, que são sempre saídas, para enviar dados através da Porta. Em aplicações simples, um usuário se concentrará principalmente nestas linhas, que estão no endereço **888**. As linhas de controle são outras 4 saídas, cuja função é controlar a impressora, que é o dispositivo normalmente ligado na porta paralela. As linhas de status são linhas de entrada de dados externos adquiridos por sensores. Há 5 delas no PC, cuja função é receber informações da impressora, tais como erro, sem papel e porta ocupada [Luís, 2005].

Cada seção é acessada pelo seu endereço próprio, e atuará independente do resto, quase como se fossem independentes.

Os respectivos endereços são mostrados a seguir :

Tabela das seções dos tipos de linhas da Porta Paralela (LUÍS, 2005).

Porta	**Função**	**Address (Decimal)**	**Address (Hex)**
Data Lines	Ativar Cargas Externas	888	378h
Control Lines	--	890	37Ah
Status Lines	Ler Sensores externos	889	379h

2.6.2 Endereços da Porta Paralela

Para enviar ou receber dados da porta paralela precisa-se saber o endereço base. Este endereço é geralmente indicado pela LPT1 ou LPT2. Veja detalhes na tabela a seguir:

Tabela de endereço LPT1 e LPT2 (LUÍS, 2005).

Nome da Porta no SO	Endereço
LPT1	378 hexadecimal / 888 decimal
LPT2	278 hexadecimal / 632 decimal

2.6.3 Pinagem da porta Paralela

Para enviar o arquivo BIT (.bit) que contém o sistema digital no FPGA através da porta paralela, é necessário então conhecer em detalhes os pinos de entrada-saída e controle da porta paralela.

A figura 2.9 mostra a pinagem da porta paralela DB-25.

A pinagem no conector DB25 é dividida em três grupo, são eles:

1. Pinos de Dados (Data Register)
2. Pinos de Controle (Control Register)
3. Pinos de Status (Status Register)

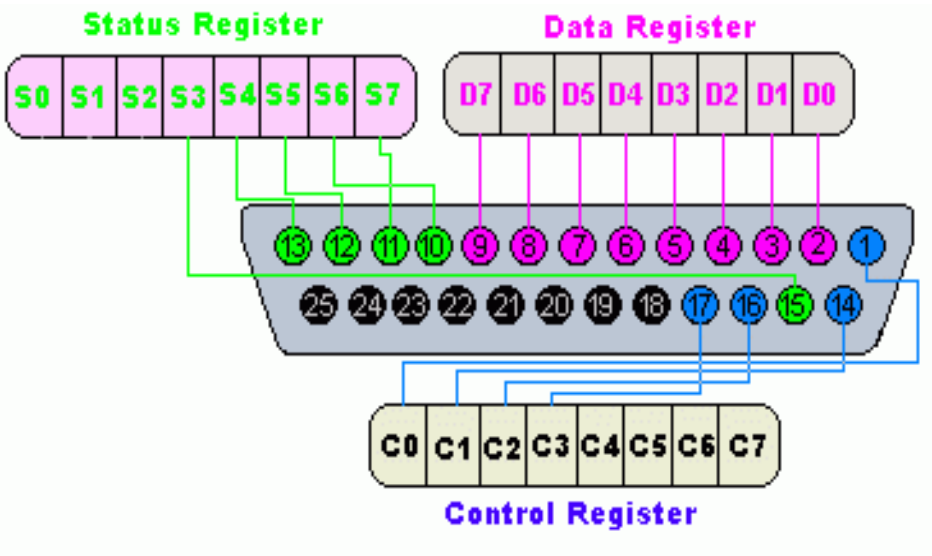

Figura 2.9 - porta paralela DB-25. (CAVALCANTI, 2004).

Interessante também que D0 a D7 representam 8 bits (1 byte). Então o valor de saída pode variar entre 00000000 (0 decimal) e 11111111(255 decimal) (CAVALCANTI, 2004).

2.6.4 Ferramenta Xsload

O XSLOAD é uma ferramenta da empresa Xess que faz parte do Kit XSTools e tem a função de descarregar o arquivo descrito em VHDL e gerado pelas portas lógicas no Xilinx (arquivo **.bit**) no FPGA.

Quando se faz um circuito digital e se mapeia ele em um FPGA é necessário descrever os pinos de entrada e saída que esse circuito terá.

Importante salienta que os pinos de entrada e saída do FPGA dêem ter uma relação com os pinos da porta paralela, para que o descarregar o arquivo BIT aconteça com sucesso.

A ferramenta XLOAD pode ser obtida no site www.xess.com/ho07000.html

3. INTERFACE DE COMUNICAÇÃO PARALELA

3.1 Interface de Simuladores

Em uma interface de simuladores, por exemplo, o Xilinx, é possível ver: as portas lógicas na forma de esquemático, o código em VHDL e o diagrama de tempo. Neste caso, a idéia que o usuário tem é que aquela simulação tem resultados provenientes do próprio micro. Sendo que na verdade é apenas uma simulação de como seria feito em algum outro circuito. Caso o usuário não entenda a ferramenta, não saberá interpretar o resultado da simulação.

Na figura 3.1 se visualiza a forma esquemática da porta lógica AND na interface da ferramenta Xilinx.

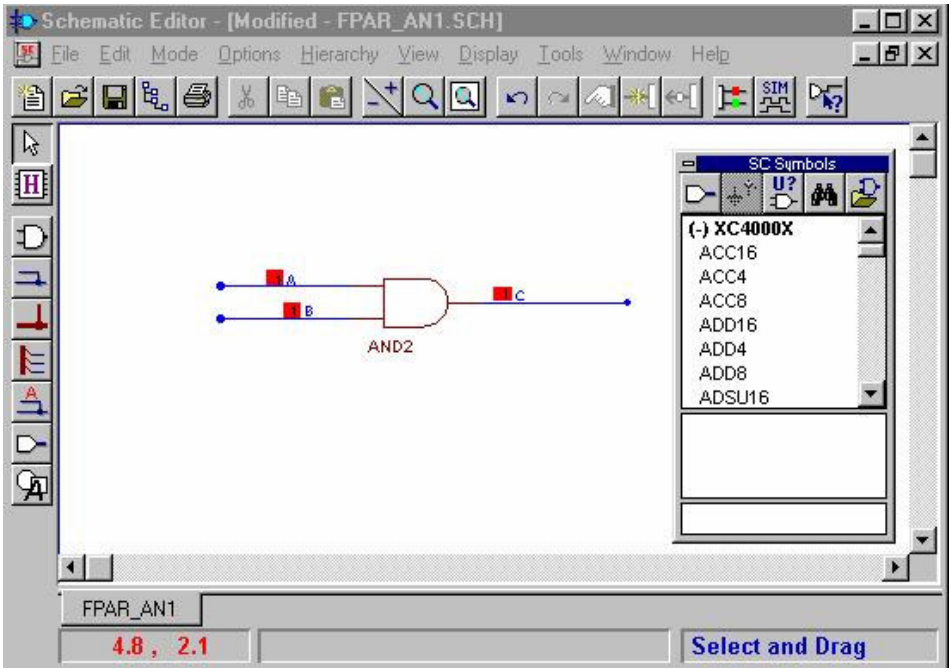

Figura 3.1 – Porta Lógica And em Esquemático na Interface do Xilinx.

A figura 3.2 permite visualizar como seria programada, em VHDL, a porta lógica AND.

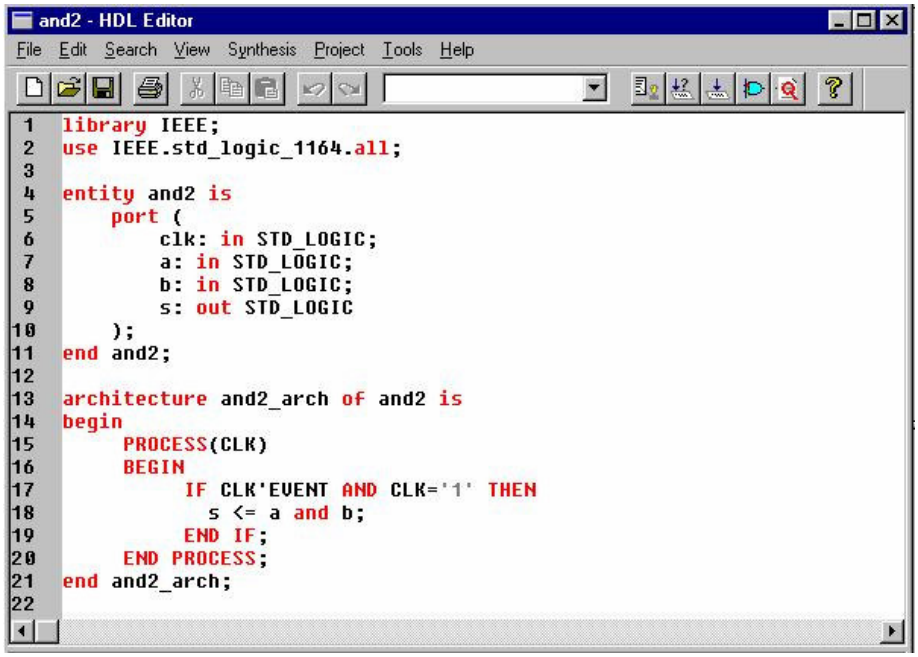

Figura 3.2 – Porta Lógica And em VHDL

A figura 3.3 mostra o diagrama de tempo dado pela ferramenta de simulação, que gera o resultado em binário o qual precisa ser interpretado pelo usuário.

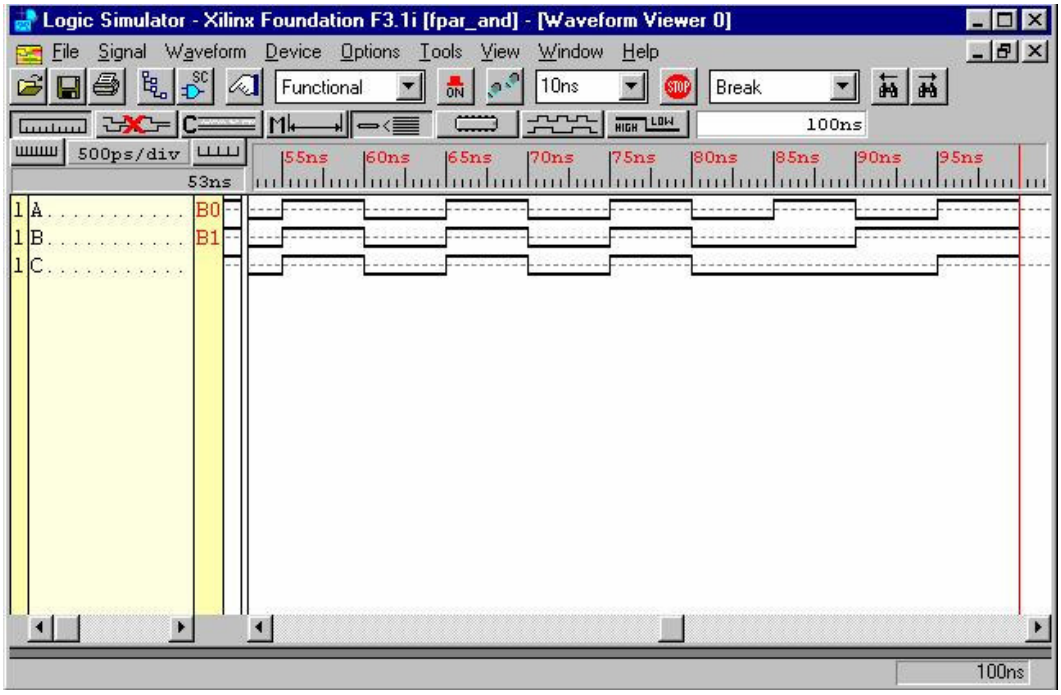

Figura 3.3 – Diagrama de Tempo da Porta Lógica And.

Nesta figura é possível notar que não é um resultado impresso do tipo binário 0 ou 1, de forma a exigir que o usuário saiba interpretar o diagrama.

3.2 Interface Didática do Projeto em Java

A primeira interface para este projeto foi desenvolvida na ferramenta Delphi, como mostra a figura 3.4. Essa interface não pode ser concluída, pois o *parport* é um software direcionado para a linguagem JAVA, na instalação o arquivo *parport.dll* é colocado na pasta j2sdk1.4.0, especifica do JAVA, no caso o delphi não contém essa pasta além dissu alguns comandos na implementação do *parport* na interface em Delphi não foram encontrados.

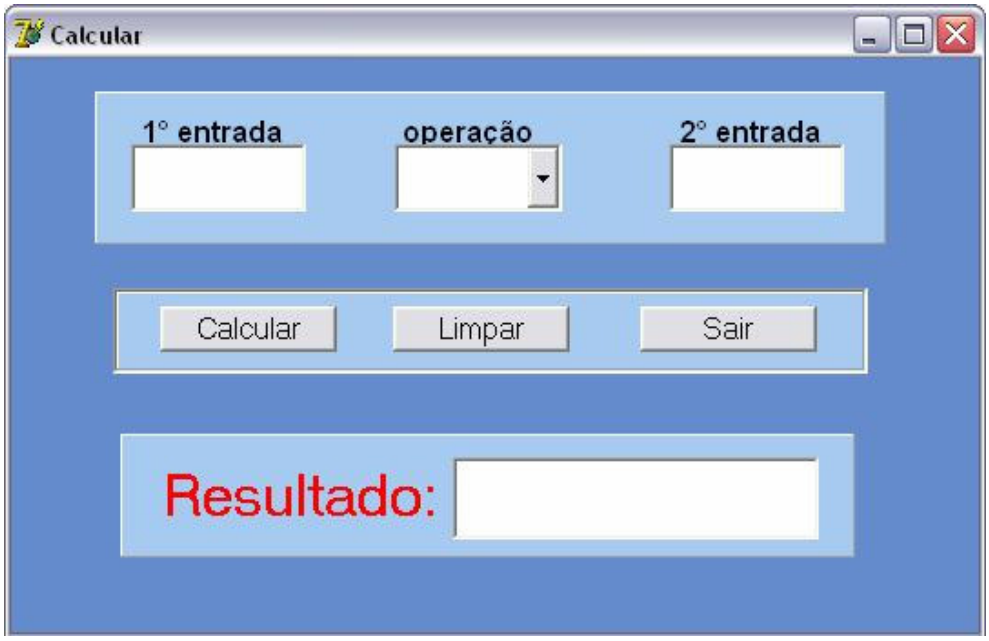

Figura 3.4 – Interface Didática em Delphi

Por esse motivo, neste projeto decidiu-se usar a lingaguem JAVA. Assim, a interface, desenvolvida em JAVA, com a ferramenta NetBeans. Esta interface mostra o cálculo feito na

placa de FPGA em binário através de portas lógicas, e oferece ao usuário a possibilidade de definir os valores das entradas e escolher a operação lógica desejada.

É necessário que o Pacote *Parport* seja importado no código, quando o usuário solicita a interface para calcular, as entradas são enviadas à porta paralela através do pacote *parport* (veja a implementação do *parport* no Apêndice A).

No momento em que o usuário seleciona a porta lógica que deseja usar, o arquivo BIT desse evento é descarregado através do XSLOAD na placa de FPGA. No código foi implementado uma estrutura switch(), assim que a opção da porta lógica é definida pelo usuário, no codigo a opção entra no case da estrutura, onde está implementado o caminho do arquivo BIT para ser descarregado na placa, como mostra o Apêndice B.

O resultado retorna para o código na mesma estrutura que é enviada nas entradas, em seguida é impresso na interface. No Apêndice C mostra-se a implementação completa da interface. Na figura 3.5 mostra a estrutura física dessa implementação

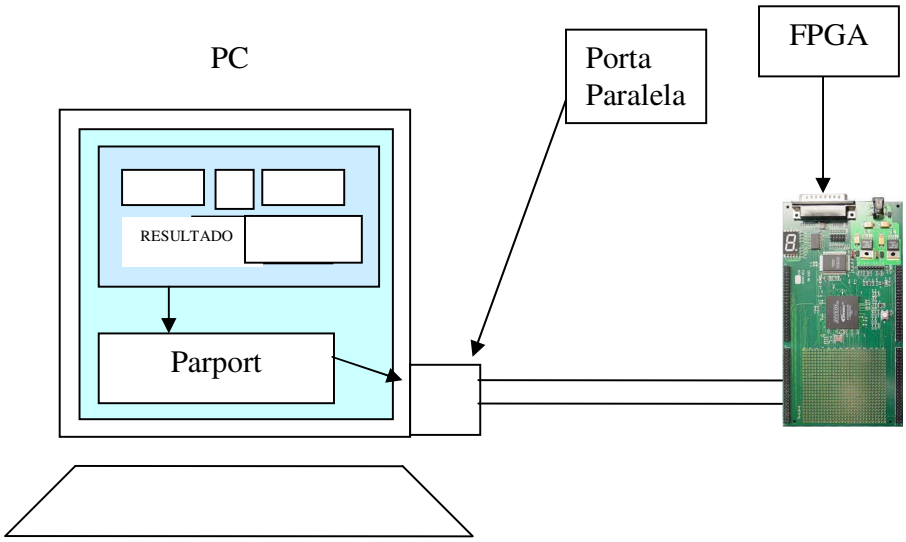

Figura 3.5 – Ligações do projeto: PC – Porta Paralela – FPGA

3.2.1 Primeira versão em Java

Por ser uma ferramenta didática, a interface desse projeto é simples e de fácil entendimento.

É necessário que se defina os valores de entrada e a operação que se deseja fazer, depois basta solicitar que seja calculado.

A figura 3.6 apresenta a interface onde percebe-se que há 2 (duas) entradas, onde o usuário digita os valores desejados e seleciona a operação. Para obter o resultado basta clicar no botão *Calcular* e no campo *Resultado* aparecerá o valor exato da operação. Consta também, o botão *Limpar* para que se possam deixar todos os campos da operação em branco como de início, e o botão *Sair* para fechar a tela de execução. Interessante perceber que o resultado que aparece nesta interface é obtido pelo circuito que está implementado no FPGA. Isto é, o resultado é calculado por um circuito real (no FPGA), não sendo simulado. Para obter este resultado vindo da placa FPGA que está conectada na porta paralela do PC foi necessárioa a utilização do *parport* que facilita a entrada e saída de dados da porta paralela.

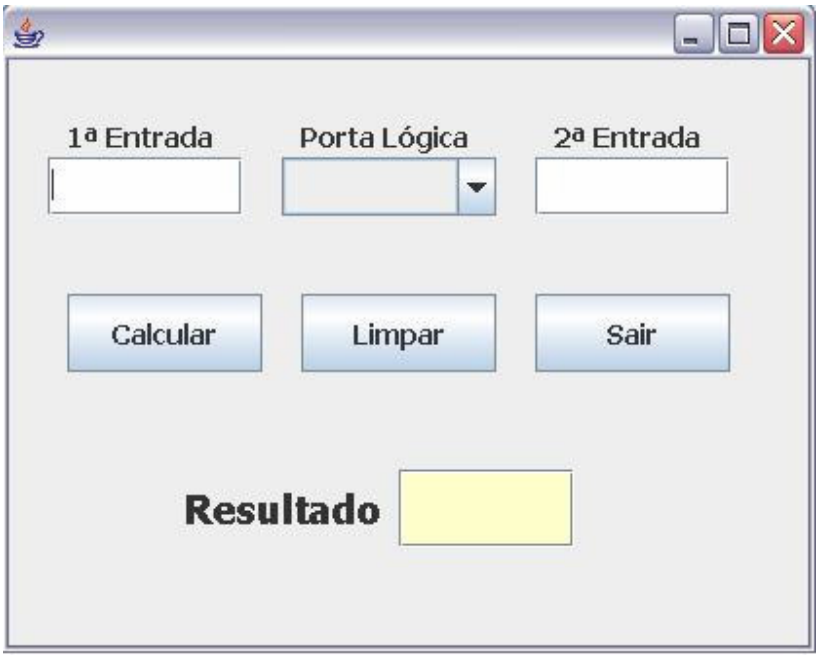

Figura 3.6 – Interface em JAVA da Comunicação PC – Porta Paralela - FPGA

O usuário poderá ver então o resultado tanto na placa quanto nesta interface, sendo mais bem entendido e talvez incentivando mais os alunos a participarem em projetos com hardware. Importante salientar que o resultado que aparece no campo resultado é aquele obtido em uma placa real.

Na figura 3.7 pode-se ver o menu aberto com as opções de portas lógicas, que o usuário poderá escolher para fazer o cálculo das entradas. Logo que o usuário seleciona a porta lógica desejada, o arquivo BIT será descarregado na placa de FPGA através do XSLOAD, para isso foi necessário que no código fosse escrito o caminho completo de onde se encontra o arquivo BIT, por exemplo: *c:\\xstools\\xsload -p 1 -b XS40-010XL -fpga c:\\load\\par_or.bit.*

Figura 3.7 – Interface mostrando Menu

Caso o usuário selecione a opção em branco do menu Porta Lógica, ele mostra um aviso como mostra a figura 3.8.

Figura 3.8 – Mensagem de escolha da Porta Lógica

E na figura 3.9 mostra a operação, primeiro o usuário define as 2 (duas) entradas e a porta lógica, então clica no botão calcular para executar, nesse momento o código lê as entradas e envia através do *parport* até a placa de FPGA, a porta lógica já está descarregada na placa, faz-se o cálculo no circuito que foi mapeado no FPGA e o programa chama o resultado através do *parport* e o imprime na interface.

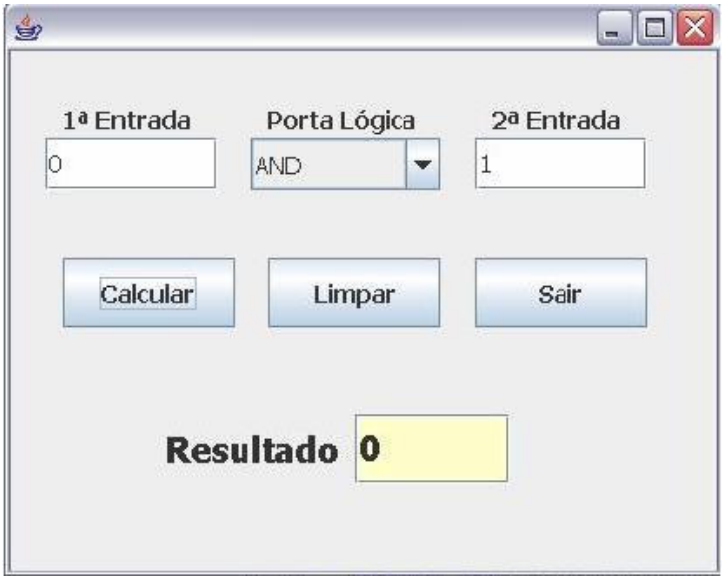

Figura 3.9 – resultado da operação

O botão *limpar* serve para limpar o campo resultado, veja na figura seguinte:

Figura 3.10 – Botão que Limpa o Resultado

3.2.2 Os Arquivos .BIT

Os arquivos BIT foram gerados na ferramenta *Xilinx* na linguagem VHDL (ORDONEZ, 2003). Na programação foi necessário declarar duas variáveis de entrada, que será as entradas que o usuário digitará na interface para serem calculadas, uma variável que receberá o resultado da operação (saída) e o clock, então cada código de cada porta lógica é simulado para se ter a certeza de que os arquivos BIT não possuíra nenhum erro.

Antes que o arquivo BIT seja gerado a partir do código de VHDL, é preciso que seja declarado os pinos do FPGA, ou seja, por qual pino será entrado os valores e por qual pino saíra o resultado. No caso os pinos declarados foram o P46 e P47 para as variáveis de entradae o P70 para a saída do resultado e o P13 que é o pino do clock.

3.3 Pinagem com os FPGAs e Porta Paralela

Inicialmente, nesta primeira versão da ferramenta, os operadores são portas lógicas programadas em VHDL na ferramenta Xilinx. Neste projeto foi necessário gerar o arquivo com extensivo BIT, para poder ser descarregado em uma placa de FPGA através da ferramenta XSLOAD.

O arquivo BIT é um arquivo que contém a função da porta lógica, e a pinagem adequada do FPGA de entrada e saída de dados. É necessário que se defina os pinos, pois cada um possui uma função, como pode ser visto na figura 3.11.

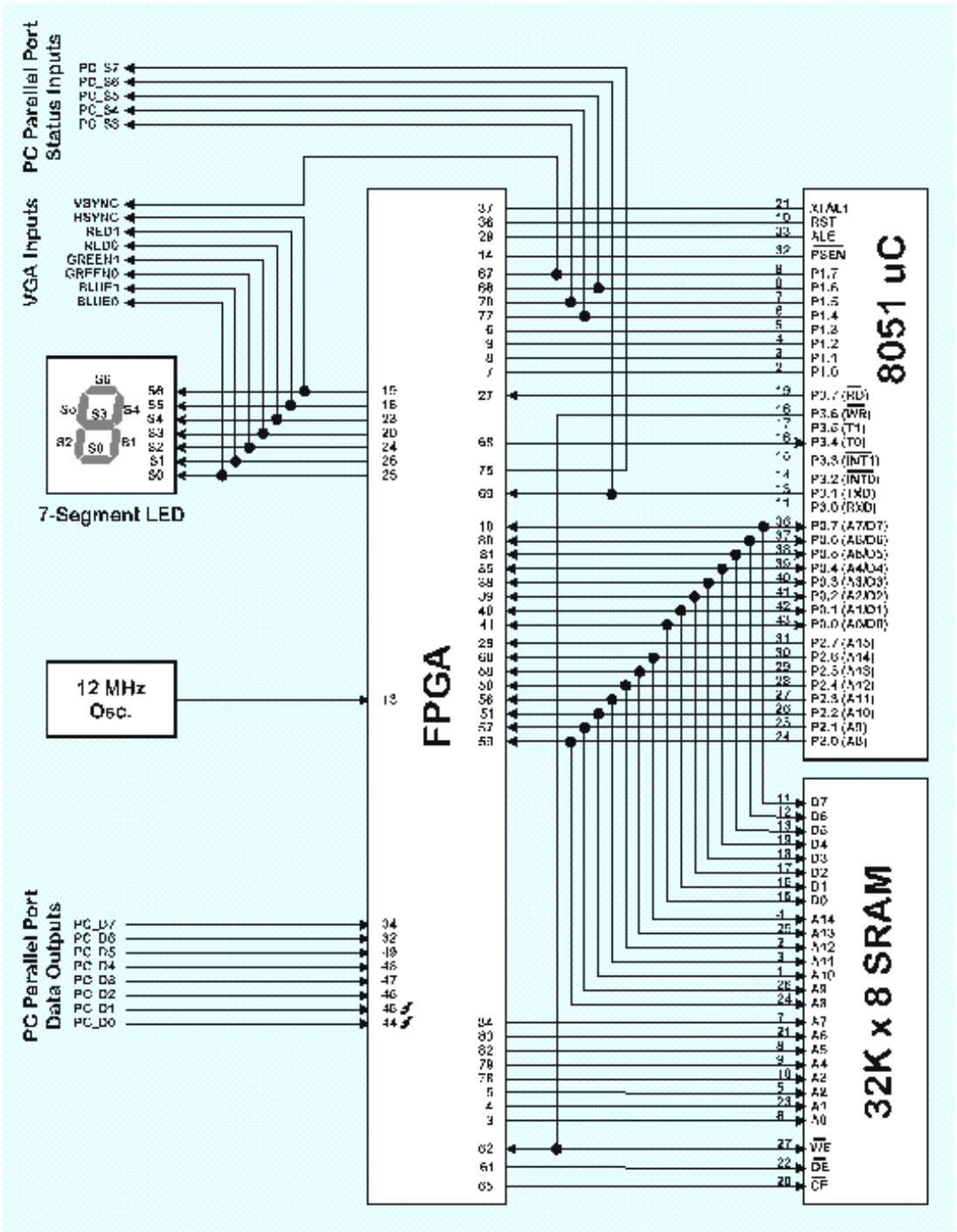

Figura .11 – Ligação dos pinos do FPGA com a Porta Paralela [Xess].

Os pinos do FPGA possuem uma relação direta com os pinos da porta paralela, ou seja, quando declarado o pino do FPGA, automaticamente já é definido os pinos da porta paralela que será usado. A seguir pode-se ver a tabela de funções de cada pino (ver tabela 3.1) de um FPGA XC4010, conforme descrito em [Xess].

XS40 Pin	Connects to...	Description
25	S0, BLUE0	Esses pinos controlam os segmentos individuais do display do LED (S0-S6 e DP). Eles também controlam a cor e os sinais de sincronismo horizontais para o monitor VGA.
26	S1, BLUE1	
24	S2, GREEN0	
20	S3, GREEN1	
23	S4, RED0	
18	S5, RED1	
19	S6, HSYNCB	
13	CLK	Uma entrada controlada pelo oscilador de 12 MHz.
44	PC_D0	Esses pinos são controlados pelos pinos de saída de dados da porta paralela do PC. Sinais de Clock podem apenas ser confiávelmente aplicados pelos pinos 44 e 45 desde que estes têm circuitos de atraso adicionais. Os pinos 32 e 34 são sinais de modo para o FPGA então você deve ajustar seu design para contar da maneira que a "Foundation tools" gerencia esses pinos.
45	PC_D1	
46	PC_D2	
47	PC_D3	
48	PC_D4	
49	PC_D5	
32	PC_D6	
34	PC_D7	
37	XTAL1	Pino que controla o clock de entrada de uC
36	RST	Pino que controla a entrada de reset de uC
29	ALEB	Pino que monitora o endereço de chave ligado de uC
14	PSENB	Pino que monitora programa de armazenagem ligado de uC
7	P1.0	Esses pinos conectam aos pinos da Porta 1do uC. alguns dos pinos também são conectados aos pinos de entrada de status da porta paralela do PC. o Pino 67 controla o sinal de sincronismo vertical para o monitor VGA.
8	P1.1	
9	P1.2	
6	P1.3	
77	P1.4, PC_S4	
70	P1.5, PC_S3	
66	P1.6, PC_S5	
67	P1.7, VSYNCB	

69	P3.1(TXD), PC_S6	Esses pinos se conectam a alguns dos pinos da Porta 3 do uC. O uC tem funções especializadas para cada porta dos pinos indicada em parênteses. O pino 62 se conecta ao pino de escrita de dados do uC e o pino da RAM de "write-enable". O pino 69 se conecta ao pino de status de entrada da porta paralela do PC.
68	P3.4(T0)	
62	P3.6(WRB), WEB	
27	P3.7(RDB)	
41	P0.0(AD0), D0	Esses pinos se conectam a Porta 0 do uC o qual é também uma porta multiplexada de endereço/dados. Esses pinos também se conectam aos pinos de dados da RAM.
40	P0.1(AD1), D1	
39	P0.2(AD2), D2	
38	P0.3(AD3), D3	
35	P0.4(AD4), D4	
81	P0.5(AD5), D5	
80	P0.6(AD6), D6	
10	P0.7(AD7), D7	
59	P2.0(A8), A8	Esses pinos se conectam a Porta 2 do uC o qual também sai para o byte de endereço superior. Esses pinos também são conectados ao 7 bit de endereço superior da RAM.
57	P2.0(A9), A9	
51	P2.0(A10), A10	
56	P2.0(A11), A11	
50	P2.0(A12), A12	
58	P2.0(A13), A13	
60	P2.0(A14), A14	
28	P2.0(A15)	
3	A0	Esses pinos controlam o 8 bit de endereço inferior da RAM.
4	A1	
5	A2	
78	A3	
79	A4	
82	A5	
83	A6	
84	A7	
61	OEB	Pino que controla a saída ligada da RAM.
65	CEB	Pino que controla o Chip ligado da RAM.
75	PC_S7	Pino que controla um pino de status de entrada da porta paralela do PC.

Tabela 3.1 – Referencia de pinagem [Xess].

3.4 Operadores

Neste projeto foram usados alguns operadores Lógicos como por exemplo: AND, NOT, NAND, OR, NOR, XOR. Foi realizado testes com esses operadores, em seguida se mostram as telas de execução desses testes.

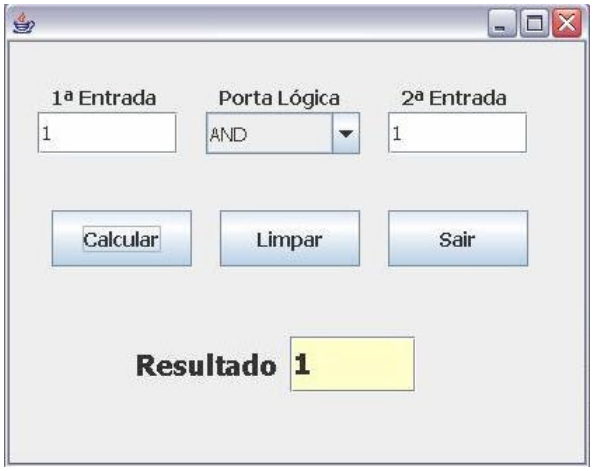

Figura 3.12 – Exemplo do Teste da Porta Lógica And

Tabela 3.2 – Tabela verdade do funcionamento de uma porta And

1° Entrada	2° Entrada	Resultado
0	**0**	**0**
0	**1**	**0**
1	**0**	**0**
1	**1**	**1**

Figura 3.13 – Teste da Porta Lógica Nand

Tabela 3.3 – Tabela verdade do funcionamento de uma porta And

1° Entrada	2° Entrada	Resultado
0	**0**	**1**
0	**1**	**1**
1	**0**	**1**
1	**1**	**0**

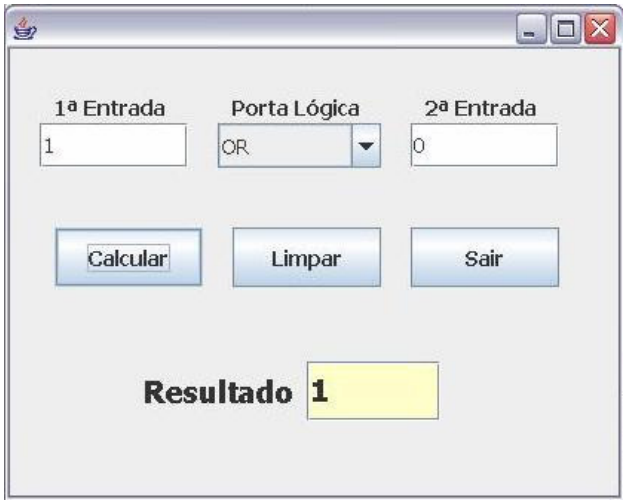

Figura 3.14 – Teste da Porta Lógica Or

Tabela 3.4 – Tabela verdade do funcionamento de uma porta Or

1° Entrada	2° Entrada	Resultado
0	**0**	**0**
0	**1**	**1**
1	**0**	**1**
1	**1**	**1**

Figura 3.15 – Teste da Porta Lógica Nor

Tabela 3.5 – Tabela verdade do funcionamento de uma porta Nor

1° Entrada	2° Entrada	Resultado
0	**0**	**1**
0	**1**	**0**
1	**0**	**0**
1	**1**	**0**

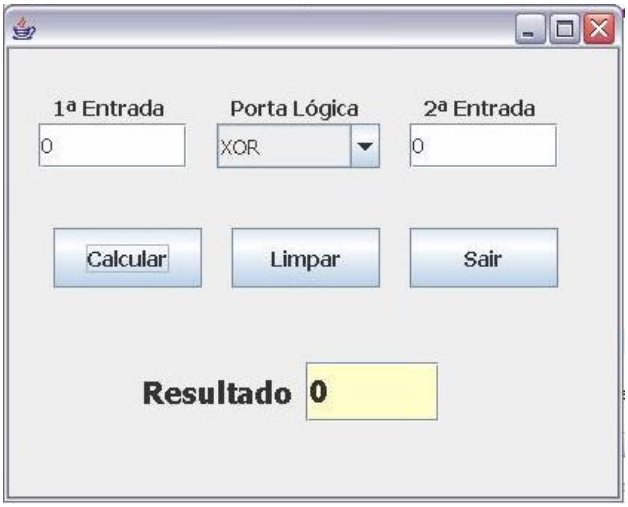

Figura 3.16 – Teste da Porta Lógica Xor

Tabela 3.6 – Tabela verdade do funcionamento de uma porta Xor

1° Entrada	2° Entrada	Resultado
0	0	0
0	1	1
1	0	1
1	1	0

3.5 Como Instalar a Ferramenta JLODI

- É necessário que seja instalado o NetBeans, de preferência a versão 5.0 beta, assim não será necessário que nada seja alterado;

- a pasta load, que contém os arquivos BIT deve ser colocada no C: ;

- se o Sistema Operacional for *Windows NT, 2000 e XP* o userport deverá ser instalado, como é explicado Na sesão 2.5.3;

- o parport já faz parte do pacote JLODI, ou seja, já estará configurado;

- o JLODI ainda não está disponível no mercado.

3.6 Como inserir novos circuitos

Caso o usuário queira adicionar outros arquivos BIT na interface, basta criar uma novoa estrutura case na estrutura switch... case..., mudando apenas o nome do arquivo que se encontra desta forma c:\\load\\par_or.bit. No caso o nome do arquivo bit é par_or.bit, é só atualizar este nome.

4. CONCLUSÕES GERAIS

4.1 Conclusão

Neste projeto de TCC (trabalho de Conclusão de Curso), se propôs e foi apresentado o desenvolvimento de uma interface na linguagem JAVA, usando o NetBeans para a implementação, onde foi utilizando ferramentas para hardware como o XSLOAD que descarrega o arquivo BIT na placa de FPGA externa e ferramentas novas para porta paralela como o Parport e o Userport, onde a operação da interface foi executada em um hardware externo (FPGA). O arquivo BIT foi gerado na ferramenta Xilinx, usando-se a linguagem VHDL.

Para que a interface pudesse ler e enviar dados pela porta paralela, foi necessário o uso do pacote Parport que facilitou esse processo. Foi necessário usar o Userport para poder liberar os recursos do Parport com a porta paralela, pois o computador utilizado possui a Sistema Operacional Windows XP.

Realizaram-se testes lógicos com portas lógicas e a ferramenta funcionando corretamente. Pode-se notar então, o uso de ferramentas novas que futuramente terá grande ênfase e ajudarão aos projetistas.

A contribuição mais importante deste projeto é a forma de como se pode criar uma interface didática para uso de hardwares externos, e a comunicação leitura/escrita entre um PC e um FPGA, com ênfase de envio de dados do FPGA ao PC através da porta paralela.

4.2 Dificuldades

Durante o desenvolvimento desse projeto, foi estudado todo o conteúdo, sendo ferramentas, tutoriais, teorias ou qualquer tipo de informação relacionada na ferramenta. Desde o momento em que foi gerado os arquivos BIT ate a resolução da operação mostrada na interface, a cada passo, foi um novo estudo, um novo conhecimento.

Antes de ser gerado o arquivo BIT, teve que ser estudo a pinagem do FPGA, pois por qual pino as informações percorreriam?

A maior dificuldade do projeto se deve a utilização do *parport* e do *userport,* tais ferramentas não possuem muitas informações, de forma que pudesse esclarecer de que as duas devem ser usadas em conjunto em alguns Sistemas Operacionais.

Pela interface em Delphi não ter sido concluída, foi-se necessário o aprendizado da linguagem Java, a qual se adapta melhor a ferramenta *parport*.

Vários testes foram aplicados de forma a esclarecer como os dados da interface seria enviados a interface e de qual forma o resultado retorna, e de como os arquivos BIT seria descarregados na placa de FPGA.

4.3 Sugestões de Trabalhos Futuros

Seria interessante a continuação do desenvolvimento dessa interface em sistemas distribuídos, onde o usuário pode executar operações no hardware de forma remota. Podemos citar um exemplo, de que alguma faculdade ou instituição que não possui condições, ou que não possui placas de FPGA, os projetistas poderiam desenvolver projetos, executar cálculos utilizando a placa de uma instituição que se encontra em outra cidade e estado, aumentando a praticidade e diminuindo custos.

Além disso, poderia-se também pensar em adicionar mais funcionalidades e operações na interface, que permitam a visualização de outros circuitos e sistemas digitais. Outras formas de se mostrar o resultado, como em decimal, hexadecimal, octal poderá ser desenvolvido também.

A possibilidade de ser usado outros tipos de hardware e não exclusivamente o FPGA, deve ser cogitada, ampliando as opções de hardware e suas funcionalidades.

REFERÊNCIAS OU BIBLIOGRAFIA

[ALBUQUERQUE, 2001] – Albuquerque, R. *"Simulação de Circuitos no Computador– MultiSIM 2001"*. Disponível em: http://www.eletronica24h.com.br/artigos/Instrumentos/MultSIM2001.html . Acesso em: 05 de mai 2005

[ARAGÃO, 1998] Aragão, S., O., C., A. *"Uma Arquitetura Sistólica para Solução de Sistemas Lineares Implementada com circuitos FPGAs"*. Disponível em: http://www.picomponentes.com.br/files_pi/dissert_aragao.pdf . Acesso em: 06 mai 2005. Dissertação de Mestrado.

[CAVALCANTI, 2004] Cavalcanti, E. *"Controle de dispositivos externos através da porta paralela utilizando C#"*. Disponível em: http://www.linhadecodigo.com.br/artigos.asp?id_ac=254&pag=1 . Acesso em: 10 jun 2005.

[DEITEL, 2003] – Deitel, H., M.;Deitel, P.,J. "*Java: Como Programar*". 4.ed. Bookman

[DEXTER, 2005] *"DEXTER Indústria e Comércio de Equipamentos Eletrônicos Ltda"*. Disponível em: http://www.dexter.ind.br/toppage1.htm . Acesso em: 05 nov 2005.

[ENGO, 1997] Engo, F. *"Como Programar em Delphi 3"* . Makron Books

[LUÍS,2005] Luís,A. *"Projeto SANKA"*. Disponivel em: http://www.portaparalela.hpg.ig.com.br/ . Acesso em: 10 ago 2005.

[MESSIAS, 2005] Messias, R., A. *"Porta Paralela"*. Disponível em: http://www.rogercom.com/. Acesso em: 01 nov 2005.

[MONTEBELLER, 2005] – Montebeller, S., J. *"Dispositivos de Lógica Programável – ALTERA – MAX-PLUS II"*. Disponível em: http://www.facens.br/alunos/material/SidneyI402TNI1/sidney_Introducao_MAX_Plus_II.pdf . Acesso em: 04 mai 2005

[NSE, 2005] "Níquel Soluções Eletronicas". Disponível em: http://www.nse.com.br/nse.php . Acesso em: 10 jun 2005.

[ORDONEZ, 2003] Ordonez, M., D., E.; Pereira, D., F.; Penteado, G., C.; Pericini, A., R. *"Projeto, Desempenho e Aplicações de Sistemas Digitais em Circuitos Programáveis (FPGAs)"* Bless.

[PAIVA] Paiva, C.; Moreno, E.; Torres,J.; Carvalho, M. *"Computação Reconfigurável: conceitos, tendências e aplicações"*. Disponível em: http://www.ppgee.pucminas.br/gsdc/papers/martins_eri02.pdf . Acesso em: 10 set 2005

[SILVA, 2003] Silva, E., R. *"Sistemas Adaptativos Implementados em FPGA"*. Disponível em: http://www.inf.ufrgs.br/procpar/disc/cmp135/trabs/rafael/T1/FPGA.htm . Acesso em : 06 mai 2005

[PEREIRA, 2004] – Pereira, F., D. *"Prototipação Rápida de Sistemas Digitais usando VHDL e FPGAs"*. Curso de Extensão VHDL, UNIVEM- Centro Universitário Eurípides de Marília.

[PIROPO, 2004] – Piropo, B. "*Arquitetura de Computadores"*. Disponível em: http://www.bpiropo.com.br/arqcom1.htm . Acesso em: 05 mai. 2005

[RIBEIRO, 2002] Ribeiro, L., A., A. *"Reconfigurabilidade dinâmica e remota de FPGAs"*. Disponível em: http://www.picomponentes.com.br/files_pi/diss_ribeiro.pdf . Acesso em 10 mai 2005. Dissertação de Mestrado.

[ROSSI, 2004] – Rossi, S., R. *"Implementação de um nó IEEE 1451, baseado em ferramentas abertas e padronizadas, para aplicações em ambientes de instrumentação distribuída"*. Tese *de Doutorado,* Departamento de Engenharia Elétrica, Unesp Ilha Solteira, 2004.

[SALCIC, 2000] Salcic, Z.; Smailagic, A.; "*Digital Systems Design and Prototyping: Using Field Programmable Logic and Hardware Description Languages"*;
Kluwer Academic Publishers, Second Edition, 621 p., 2000;

[STALLINGS, 2004] Stallings, W. "Arquitetura e Organização de Computadores". 5.ed. Pearson.

[TANENBAUM, 2003] – Tanenbaum, A., S. *"Redes de Computadores"*. 4. ed. Campus 2003

[Xess] XESS Corp. Disponivel em: http://www.xess.com/prgmdl40.html . Acesso em: 20 out 2005.