FACULDADE DE INFORMÁTICA
PUCRS - Brazil
http://www.inf.pucrs.br

Ferramentas de Reconfiguração Parcial, Remota e Dinâmica de FPGAs Virtex

Leandro Heleno Möller, Fernando Gehm Moraes,
Ney Laert Vilar Calazans

TECHNICAL REPORT SERIES

Number 035
November, 2003

Contact:

moller@inf.pucrs.br

http://www.inf.pucrs.br/~moller

L. H. Möller is a research assistant at the GAPH Research Group in PUCRS/Brazil since 2000, where he received a federal undergraduate research grant from CNPq (Brazil) from 2000 to 2003. Mr. Möller holds a CS degree in Computer Science from PUCRS. Currently he receives a research assistant grant from CNPq (Brazil).

F. G. Moraes works at the PUCRS/Brazil since 1996. He is a professor since August 2003. His main research topics are digital systems design and fast prototyping, digital systems physical synthesis CAD, telecommunication applications, hardware-software codesign. He is a member of the Hardware Design Support Group (GAPH) at the PUCRS.

N. L. V. Calazans works at the PUCRS/Brazil since 1986. He is a professor since 1999. His main research topics are digital systems design and fast prototyping, hardware-software codesign, telecommunication applications. He is the head of the Hardware Design Support Group (GAPH) at the PUCRS.

Índice

Lista de Figuras

Lista de Figuras

1 Introdução

As ferramentas aqui documentadas têm por objetivo trabalhar com o arquivo final do projeto de hardware (bitstream) dos dispositivos Virtex da fabricante Xilinx [1]. O bitstream é o arquivo de configuração que é enviado para os dispositivos reconfiguráveis e que tem por finalidade programar o hardware de forma a implementar uma determinada funcionalidade. Existem dois formatos para este arquivo de configuração: ASCII (arquivos com extensão RBT) e binário (arquivos com extensão BIT). O formato ASCII é voltado para um melhor entendimento sobre o bitstream. O formato binário tem a vantagem de ser cerca de oito vezes menor que o bitstream em formato ASCII.

Um bitstream pode ser enviado para um dispositivo reconfigurável de diversas formas quando a plataforma de prototipação é externa ao computador. Atualmente as interfaces mais utilizadas do computador para fazer download de bitstreams são a paralela, a serial e a USB. Para interconectar o computador a partir destas interfaces à plataforma de prototipação são utilizados cabos Parallel Cable, MultiLinx e cabos paralelos. As interfaces da plataforma de prototipação são conhecidas como JTAG, SelectMAP e SlaveSerial. Cada uma destas interfaces e cabos possui características e velocidades diferentes de envio de dados que não serão aqui abordadas.

Com estas ferramentas é possível modificar propriedades dos bitstreams sem a necessidade de refazer todas as fases do fluxo de geração de um bitstream, que algumas vezes pode necessitar de um poder computacional muito alto, caras ferramentas de CAD, muito tempo de processamento ou até mesmo características de roteamento e temporização que podem ser difíceis de serem repetidas a cada síntese. A Figura 1 apresenta o fluxo padrão para geração de bitstreams.

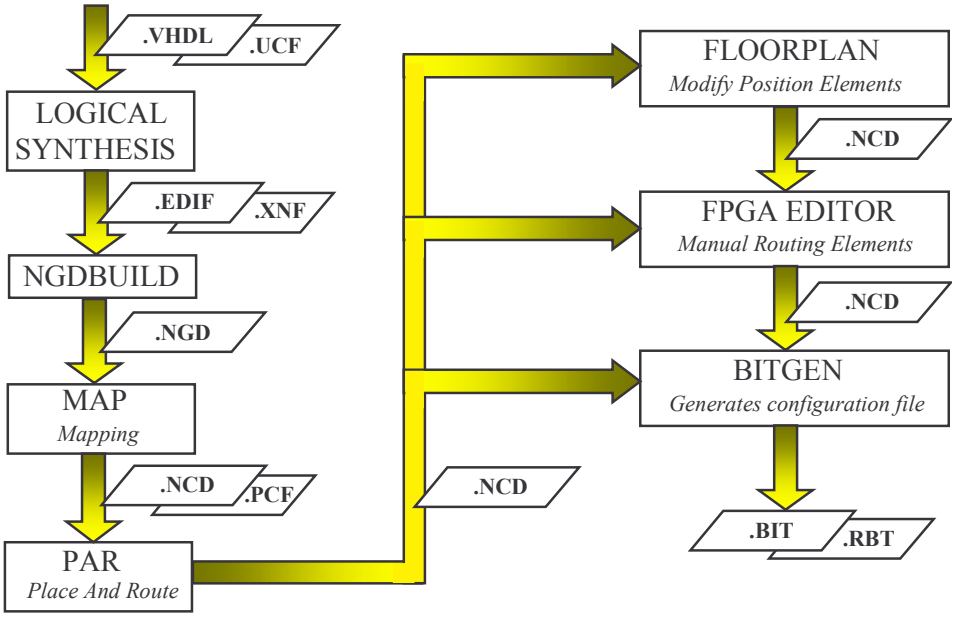

Figura 1 – Fluxo padrão para projetos de sistemas digitais de dispositivos Xilinx.

O que este documento propõe é que o fluxo apresentado na Figura 1 seja executa-

do uma única vez e o bitstream gerado seja enviado para o FPGA. As modificações ao projeto original devem ser realizadas a partir das ferramentas que serão apresentadas a seguir. Todas as ferramentas utilizam como entrada um bitstream e geram outro modificado, que deve ser novamente enviado para o FPGA, resultando em uma modificação do hardware em execução.

O principal motivo de Java como linguagem de programação se deve ao fato de já existir bibliotecas que mapeiem elementos do projeto de Hardware para o arquivo de configuração. Esta biblioteca que contém a implementação de leitura e modificação de dados é chamada de JBits, fabricada pela Xilinx [2].

Este relatório técnico está dividido como segue. A Seção 2 apresenta dicas importantes no processo de instalação de softwares pré-requisitos para as ferramentas que serão apresentadas no documento. A Seção 3 apresenta o pacote de ferramentas intitulada BITProgrammer, que permite o acesso a distância a diversos tipos de componentes internos do FPGA. Uma ferramenta chamada CircuitCustomizer que permite o usuário modificar parâmetros do FPGA a distância e sem conhecer detalhes da plataforma é mostrada na Seção 4. A Seção 5 apresenta a ferramenta BITAnalyzer que mostra o conteúdo do bitstream com o seu respectivo significado. A ferramenta RBTProgrammer é apresentada na Seção 6 e tem o objetivo de acessar LUTs de um bitstream em formato ASCII (RBT). Uma ferramenta chamada CoreUnifier que possibilita a realocação de núcleos de hardware é apresentada na Seção 7. A ferramenta BIT2RBT, apresentada na Seção 8, transforma arquivos do formato binário para o formato ASCII e a ferramenta RBT2BIT transforma arquivos no formato ASCII para o formato binário, sendo esta apresentada na Seção 9.

É importante observar que este documento está fortemente relacionado ao trabalho prático das dissertações de Daniel Gomes Mesquita [3] e José Carlos Sant´Anna Palma [4]. Este documento também é o relatório técnico para um conjunto de artigos publicados [5][6][7][8][9][10].

2 Requisitos de Software

A máquina com a plataforma de prototipação deverá ter instalado um software para realizar o download do bitstream, denominado Impact.

Outro requisito é a instalação do JDK para a execução das ferramentas feitas em Java. Recomenda-se a instalação da versão 1.2.2 do JDK, pois é a única que é compatível com os instaladores da versão 2 do JBits. Após a instalação bem sucedida do JBits é possível atualizar a versão do JDK. É importante ressaltar que as versões 2 do JBits são voltados para os dispositivos Virtex e a versão 3 para os dispositivos Virtex II.

Caso seja instalada a versão 2.4 do JBits ou anterior deve-se requisitar ao fabricante a atualização do arquivo Bitstream.class, pois ela gerá um bitstream inválido para computadores com data posterior ao ano 2000.

Para utilizar as ferramentas remotamente pelo navegador de internet é necessário um servidor de páginas, como, por exemplo, o Apache. As máquinas que acessarão remotamente a plataforma de prototipação deverão ter o Internet Explorer 5.5 ou versão superior instalado para rodar as Applets.

As ferramentas desenvolvidas estão disponibilizadas na área de distribuição de software de reconfiguração do GAPH em http://www.inf.pucrs.br/~gaph.

Nas próximas seções seguem algumas dicas de instalação dos softwares requisitos para as ferramentas que foram desenvolvidas.

2.1 Apache

É possível baixar no site http://www.apache.org o servidor Apache. Faça a instalação comum da versão baixada. Em um momento será requisitado o *Network Domain*, deixe-o com o nome que apareceu ou vazio. No campo *Server Name* coloque o IP da sua máquina (caso ela tenha um IP fixo) ou *localhost*. Para pegar o endereço IP em uma máquina Windows entre no DOS e digite *ipconfig*. Caso a máquina seja LINUX digite *ifconfig*. O endereço IP da máquina aparecerá onde diz *IP Address*. Anote este IP pois você precisará dele para acessar o servidor. No campo *Administrator´s Mail Address* coloque o endereço de mail da pessoa que é responsável pela máquina. Após isso continue com a instalação padrão.

2.2 JDK 1.2.2

No site http://java.sun.com/products/jdk/1.2/ escolha a versão que esteja de acordo com o sistema operacional da máquina. Siga com a instalação padrão. Não esqueça que se a versão do JBits a ser instalada é superior a 2.4 e inferior a 2.8 é obrigatório a instalação da versão 1.2.2 do JDK.

2.3 JBits

Se cadastre em http://www.xilinx.com e faça o download de uma versão do JBits. As versões 2 do JBits funcionam somente para o dispositivo Virtex e a versão 3 apenas para os dispositivos Virtex II.

2.4 Internet Explorer 5.5 ou superior

No endereço http://www.microsoft.com/downloads é possível baixar a versão mais atual do Microsoft Internet Explorer. Siga com a instalação padrão. A seguir é necessário alterar as permissões do browser fazendo o seguinte:

No Internet Explorer 5.5 clique em *Tools, Internet Options, Security, Custom Level* e role a tela até *Microsoft VM.* Clique em *Personalizar, Java Custom Settings, Edit Permissions e* Clique em *Enable* no *Run Unsigned Content*

3 BITProgrammer

O pacote de ferramentas desenvolvido com o nome de BITProgrammer acessa bitstreams a partir da máquina local ou remota. Estes bitstreams devem estar no formato binário e podem ser parciais ou totais. As ferramentas permitem visualizar valores do bitstream, procurar valores diferentes dos padrões, modificar valores internos das LUTs, executar o download remoto de um bitstream e localizar LUTs com uma lógica especifica.

Um arquivo que é comum a todas as ferramentas é o jt.cmd que contêm a especificação do dispositivo utilizado. Este arquivo é passado para o software responsável por enviar o bitstream para o FPGA (JTAG Programmer ou Impact). O comando *setMode* com parâmetro *–bscan* contido nesse arquivo significa que deve ser feito uma leitura em cadeia dos dispositivos existentes na placa de prototipação. SetCable –lpt1 significa que a porta LPT1 será utilizada para fazer o download. O comando *identify* é enviado para a plataforma de prototipação de forma a receber uma lista de dispositivos existentes na plataforma de prototipação. *AssignFile –p 1* significa que será enviado um arquivo de configuração para o dispositivo que foi detectado como número 1. *–file c:\arquivo.bit* contém o nome do arquivo de configuração que será enviado para o dispositivo 1. *Program –p 1* é o comando que inicia a programação do dispositivo 1 do FPGA. *Quit* sai do software que executa o download no dispositivo FPGA.

As seções seguintes apresentarão as ferramentas que fazem parte do pacote de ferramentas BITProgrammer.

3.1 BITProgrammerDownload

O BITProgrammerDownload é a versão local do BITProgrammer. Para que o mesmo possa ser executado, descompacte o arquivo *bitprogrammerdownload.zip* dentro do diretório onde foi instalado o JBits. Para rodar o programa execute *bitprogrammerdownload.bat.* Caso o JDK esteja funcionando corretamente será exibida uma tela semelhante a Figura 2. À esquerda da tela são mostrados os dados para ler ou escrever em qualquer LUT do dispositivo Virtex. Abaixo existem botões das ações que podem ser executadas sobre o bitstream. À direita é apresentada uma tela de status e resposta das operações executadas.

O primeiro passo para a execução do programa é abrir um bitstream (clicar em *Arquivo*, A*brir*, escolher arquivo, *OK*). Caso o bitstream seja válido as operações a serem executadas sobre ele são apresentadas na parte inferior da tela. Para ler o conteúdo de uma CLB selecione os seus parâmetros (linha, coluna, LUT e slice) e clique em *Ler CLB*. Para modificar o conteúdo de uma CLB selecione os seus parâmetros (linha, coluna, LUT, slice e os 16 bits a serem configurados) e clique em *Confirmar CLB*. A seguir é possível salvar o bitstream com um novo nome (clicar em *Arquivo*, *Salvar Como*, escolher arquivo, *OK*) ou salvar com o mesmo nome do arquivo aberto (clicar em *Arquivo*, *Salvar*).

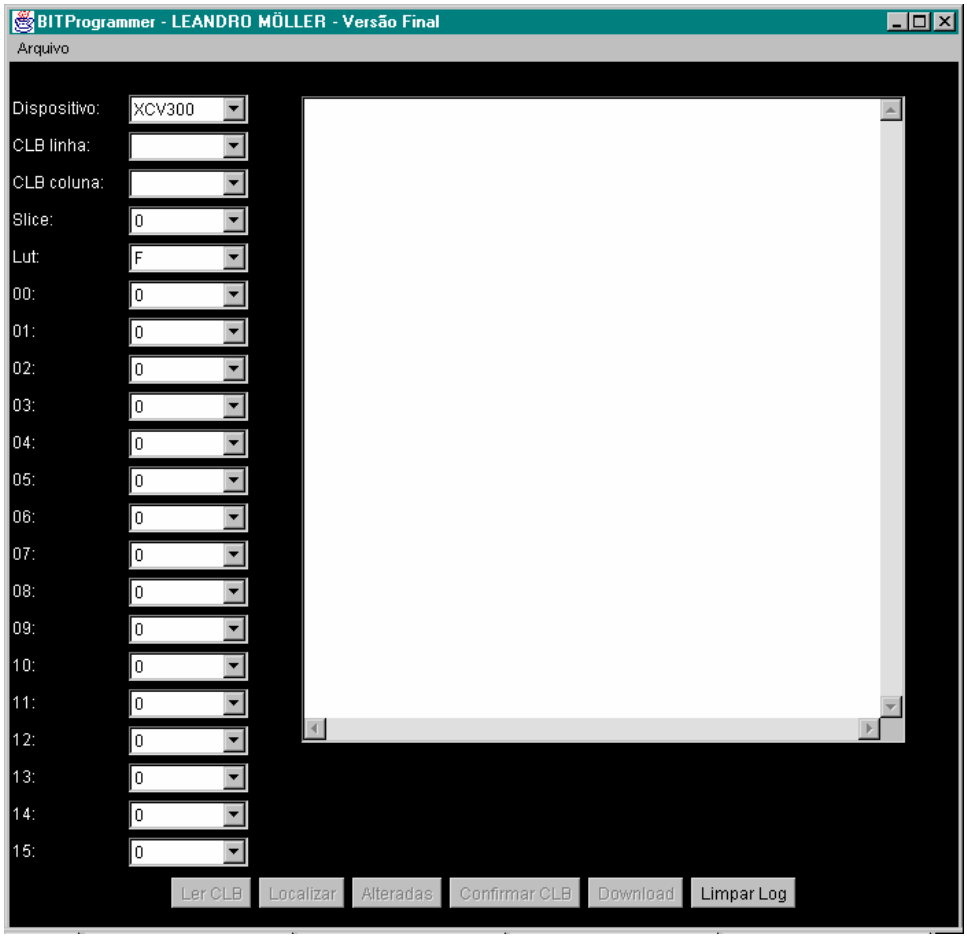

Figura 2 - Tela Inicial do BITProgrammer

A ferramenta também permite visualizar todas as LUTs com valores diferentes do padrão (clicar em *Alteradas*) ou até mesmo procurar uma configuração específica (selecionar os dezesseis bits de configuração da LUT e clicar em *Localizar*). Caso seja desejado também é possível fazer o download de um bitstream (clicar em *Download*) e limpar a tela que informa as operações realizadas (clicar em *Limpar Log*).

3.2 BITProgrammerServidor

O BITProgrammerServidor atende clientes como BITProgrammerCliente (seção 3.3), BITProgrammerApplet (seção 3.4) e CircuitCustomizer (seção 4). Ele recebe as requisições, executa as funções correspondentes e retorna o resultado ao cliente. Para que o mesmo possa ser executado, descompacte o arquivo *BITProgrammerServidor.zip* dentro do diretório onde foi instalado o JBits. Para rodar o programa com os valores iniciais execute *bitprogrammerservidor.bat.* Caso o JDK e o JBits esteja funcionando corretamente será exibida uma tela semelhante a apresentada na Figura 3.

```
C:\Program Files\JBits2.7>servidor

C:\Program Files\JBits2.7>java BITProgrammerServidor XCV300 5000 log.txt c:\prog
ra~1\jbits2.7\bits\ jt.cmd
Servidor iniciado em: Thu Nov 22 11:52:43 BRST 2001
```

Figura 3 - Exemplo de tela apresentada pelo BITProgrammerServidor

Para alterar os parâmetros aceitos por este servidor pode ser modificado o arquivo *bitprogrammerservidor.bat* para que o mesmo carregue corretamente as configurações de um determinado computador ou dispositivo. Inicialmente o BITProgrammerServidor está configurado com o seguinte conteúdo:

java BITProgrammerServidor XCV300 5000 log.txt c:\bits\ jt.cmd

Onde *java* é o programa que executa o BITProgrammerServidor. *BITProgrammerServidor* é o nome do aplicativo a ser executado. *XCV300* é o nome do dispositivo que será utilizado na abertura de bitstreams. *5000* é a porta de comunicação do servidor para atender os clientes. *log.txt* é o nome do arquivo que é criado após cada download, mostrando o sucesso ou fracasso do download do bitstream no dispositivo. *c:\bits* é o diretório onde os bitstreams estão armazenados. *jt.cmd* é o arquivo que contém o nome do arquivo que efetuará o download para a placa.

Após a inicialização do servidor pode-se minimizar a janela do servidor ou deixá-la aberta para acompanhar as transações dos clientes com o servidor.

3.3 BITProgrammerCliente

Este cliente envia requisições para o BITProgrammerServidor (seção 3.2) processar a partir de uma aplicação Java. Utilizando este cliente não é necessário que o Apache ou outro servidor de páginas esteja instalado na máquina que contém a placa de prototipação.

Para iniciar o BITProgrammerCliente em uma máquina qualquer que já tenha o JDK instalado, descompacte o *bitprogrammercliente.zip* e execute o *bitprogrammercliente.bat*. A seguir é necessário conectar o cliente com o servidor (clicar em *conexão*, *conectar*, informar o IP e a porta, *OK*). Caso o JDK esteja corretamente instalado será exibida uma janela semelhante a apresentada na Figura 4.

O primeiro passo ao carregar o programa é listar os bitstream que estão no servidor (*arquivo*, *listar*). A seguir abra um dos arquivos listados (clicar em *Arquivo*, *Abrir*, informe o nome do bitstream a ser aberto, *OK*).

Após isto o funcionamento da versão cliente é análoga a versão local, apresentada na seção 3.1.

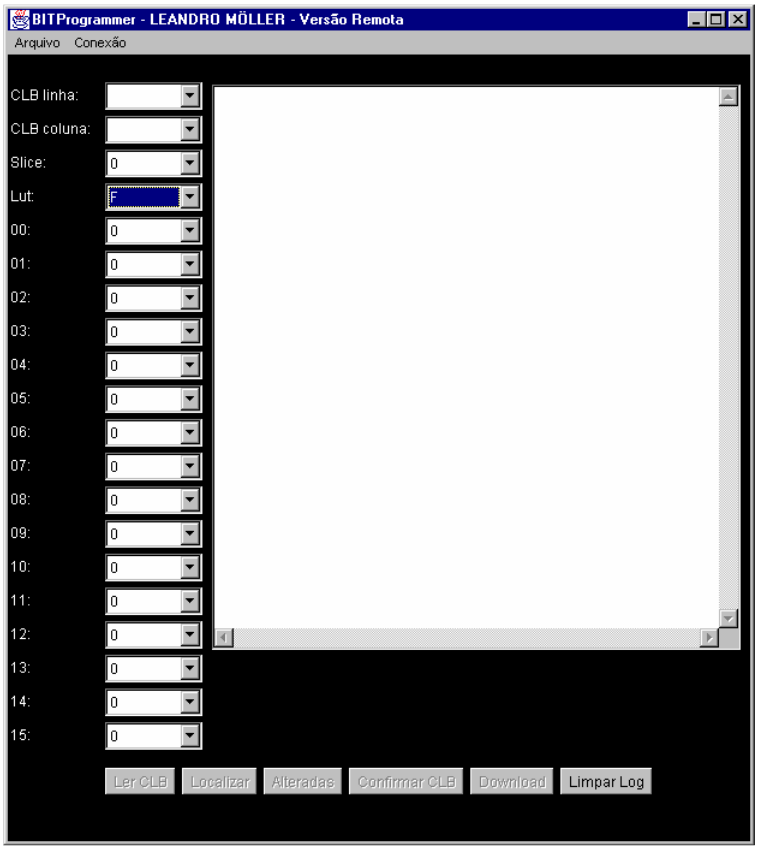

Figura 4 - Tela inicial do BITProgrammerCliente

3.4 BITProgrammerApplet

Este é um cliente que executa as mesmas funções que o BITProgrammerCliente (seção 3.3) com a vantagem de acessar o servidor a partir do navegador de internet, não sendo necessário instalar o JDK na máquina cliente.

Para utilizar esta ferramenta é obrigatório a instalação de um servidor de páginas de Internet no servidor como relatado na Seção 2. Também é necessário que seja descompactada o *bitprogrammerapplet.zip* em um uma pasta pública do servidor de páginas de Internet.

Para acessar o BITProgrammerApplet em um browser de uma máquina cliente digite na barra de endereços o IP da máquina servidor, o caminho onde o *bitprogrammerapplet*.zip foi descompactado e *applet.htm* que é a página que se comunica com o servidor. Caso o servidor de página de Internet esteja funcionando corretamente, o browser carregará uma página de Internet conforme a apresentada na Figura 5.

A seguir é necessário conectar o BITProgrammerApplet ao BITProgrammerServidor (preencher o IP da máquina servidor no campo IP, a porta que o BITProgrammerServidor está utilizando e clicar em conectar).

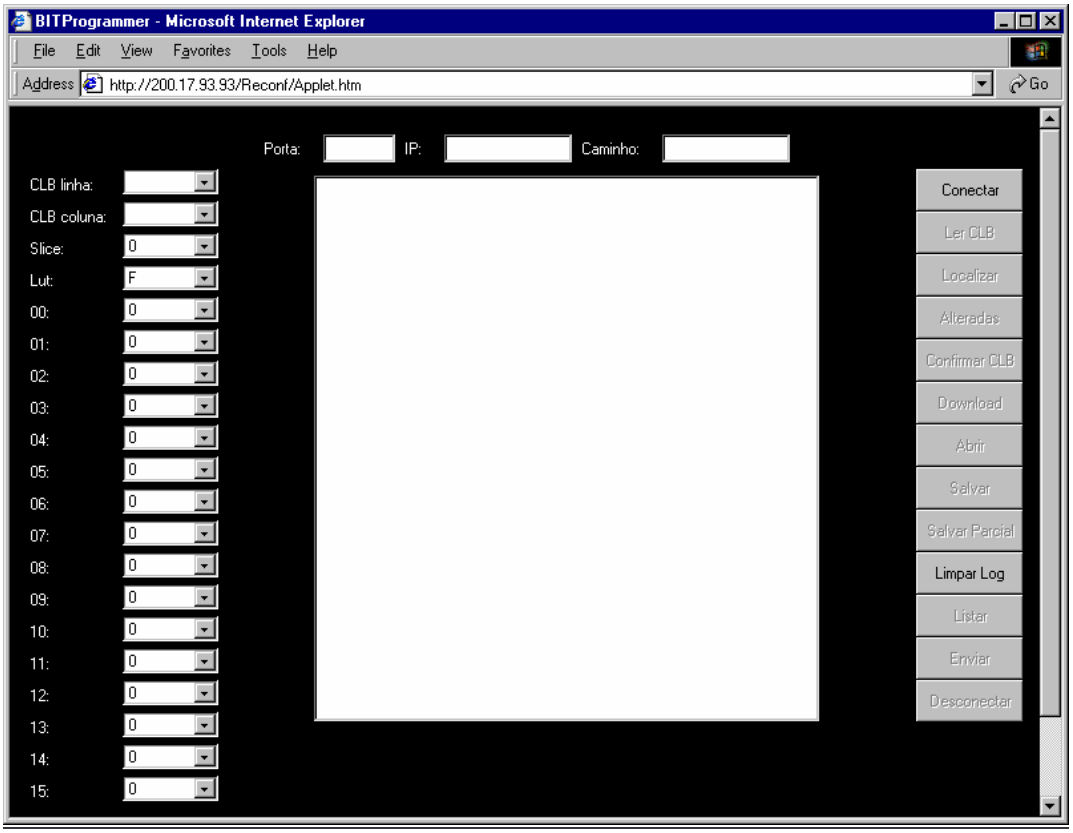

Figura 5 - Exemplo de tela inicial apresentado pelo BITProgrammerApplet

As funcionalidades do BITProgrammerApplet são as mesmas do BITProgrammerDownload (seção 3.1) e do BITProgrammerCliente (seção 3.3). A única diferença é na interface gráfica da ferramenta que apresenta os botões com as funcionalidades à direita da janela.

4 CircuitCustomizer

Esta ferramenta é um cliente adicional aos apresentados nas seções 3.1, 3.3 e 3.4. A sua principal vantagem é ser mais alto nível que os clientes anteriores, permitindo ao projetista do circuito especificar a localização de sinais de um projeto de hardware para a posterior visualização e alteração dos valores a partir de um HTML. Desta forma o usuário final não precisa conhecer LUTs, CLBs ou posições onde os parâmetros foram fixados para alterar parâmetros do sistema.

Para utilizar esta ferramenta é obrigatório a instalação de um servidor de páginas de Internet no servidor como relatado na Seção 2. Também é necessário que seja descompactado o *circuitcustomizer.zip* em um uma pasta pública do servidor de páginas de Internet.

Para acessar o CircuitCustomizer em um browser de uma máquina cliente digite na barra de endereços o IP da máquina servidor, o caminho onde o CircuitCustomizer foi descompactado e o nome da página que se comunica com o servidor. Caso o servidor de página de Internet e o BITProgrammerServidor estejam funcionando corretamente, o browser carregará uma página de Internet conforme a apresentada na Figura 6.

Existem três atores envolvidos nesta ferramenta: o desenvolvedor do software, o projetista do circuito e o usuário do circuito.

O **desenvolvedor do software** implementa uma camada de software escondendo os detalhes da arquitetura do FPGA. Esta camada de software é implementada como uma applet que se comunica com o servidor. O servidor utiliza as classes do JBits para acessar e modificar informações contidas no bitstream. Essa applet é a mesma para todos os circuitos a serem customizados.

O **projetista do circuito** utiliza tags para customizar a HTML de acordo com o circuito implementado. Um exemplo desta descrição HTML pode ser visto na Figura 7 e é melhor analisado na seção 4.1.

O **usuário do circuito** recebe a HTML para acessar e modificar os sinais do projeto. A página HTML da descrição apresentada na Figura 7 pode ser visualizada na Figura 6.

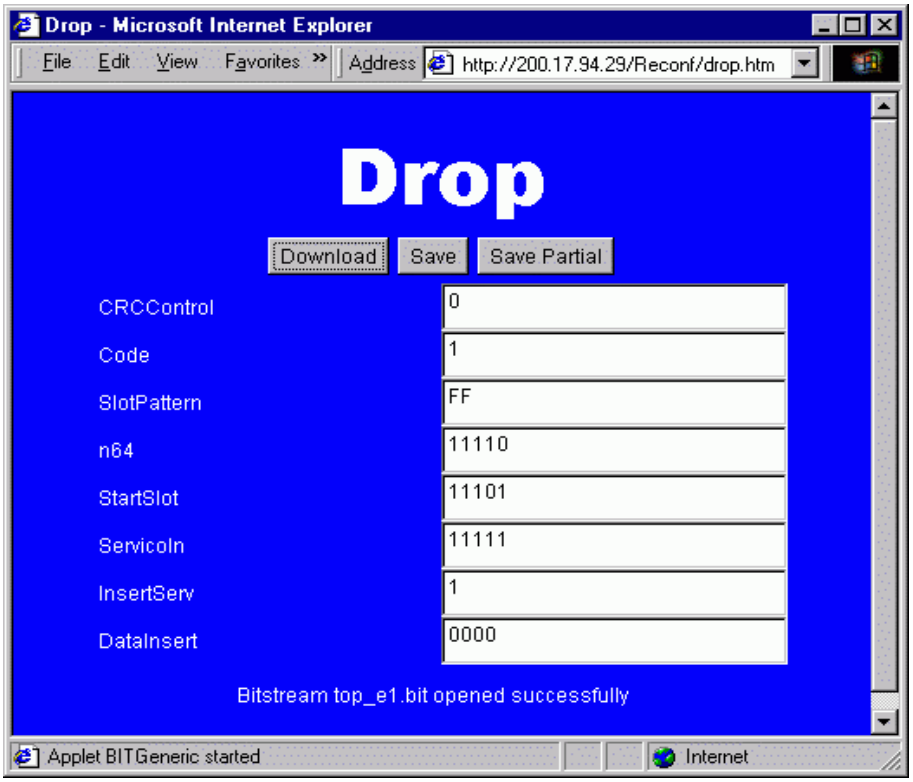

Figura 6 – Exemplo de página de Internet que apresenta os sinais de um projeto.

```
1.  <html>
2.  <head> <title> Drop </title>        </head>
3.  <body bgcolor=blue>
4.  <center>
5.  <font color=white size=7 face="Arial Black"> Drop </font>
6.  <APPLET code="CircuitCustomizer.class" width=400 height=300>
7.  <PARAM name="path"          value="top_e1.bit">
8.  <PARAM name="ip"            value="200.17.93.93">
9.  <PARAM name="port"          value="5000">
10. <PARAM name="nbsignals"     value="8">
11. <PARAM name="l[1]"          value="CRCControl  bin, 32, 37, G, 0, 0, 0">
12. <PARAM name="l[2]"          value="Code        bin, 32, 37, G, 0, 1, 1">
13. <PARAM name="l[3]"          value="SlotPattern hex, 32, 37, G, 0, 2, 9">
14. <PARAM name="l[4]"          value="n64         bin, 31, 37, G, 0, 4, 0">
15. <PARAM name="l[5]"          value="StartSlot   bin, 31, 37, G, 0, 9, 5">
16. <PARAM name="l[6]"          value="ServicoIn   bin, 31, 37, G, 0,14,10">
17. <PARAM name="l[7]"          value="InsertServ  bin, 31, 37, G, 0,15,15">
18. <PARAM name="l[8]"          value="DataInsert  hex, 28, 37, G, 0, 0,15">
19. </APPLET>
20. </center>
21. </body>
22. </html>
```

Figura 7 - Descrição em código HTML da página de Internet apresentada na Figura 6.

4.1 Projetista do Circuito

É tarefa do projetista do circuito gerar uma descrição HTML para cada bitstream gerado. O bitstream deve ser colocado no repositório de bitstreams do servidor e a descrição HTML que foi gerada para ele deve ser colocada em uma pasta pública do servidor de páginas de Internet do servidor.

As linhas de 1 a 5 da Figura 7 representam o código HTML para apresentação da página. A linha 6 contém a chamada do CircuitCustomizer para carga em um browser com tela de tamanho 400 por 300 pixels. Na linha 7 é passado para o parâmetro *path* o nome do bitstream que deve ser carregado, no caso *top_e1.bit*, que deve estar no diretório que armazena os bitstreams no servidor. Na linha 8 é passado o IP do servidor para o parâmetro *ip*. Na linha 9 é passado para o parâmetro *port* a porta que foi aberta no servidor para atender os clientes. A linha 10 especifica quantos sinais devem ser apresentados na página de Internet. Nesse caso especificou-se para o parâmetro *nbsignals* que 8 sinais sejam apresentados. As linhas de 11 a 18 contêm os dados de cada um dos 8 sinais, que serão apresentados cada um em uma linha da página HTML. Cada sinal deve informar pelo parâmetro *l* em qual linha será apresentado da página e as seguintes informações: nome do sinal, formato de apresentação do valor (bin, hex ou dec), linha da LUT que armazena o valor, coluna da LUT que armazena o valor, LUT F ou G, slice 0 ou 1, bit inicial e bit final que faz parte deste sinal.

Existem dois comandos opcionais que podem ser passados para o parâmetro *l* em um dado sinal. O primeiro deles é o *negado*, que converte tudo que é 0 para 1 e viceversa. O segundo é o *invertesubstring* que espelha os bits de cada um dos nibbles a serem apresentados na página de internet. A Figura 8 ilustra melhor a operação efetuada por estes dois comandos em um byte.

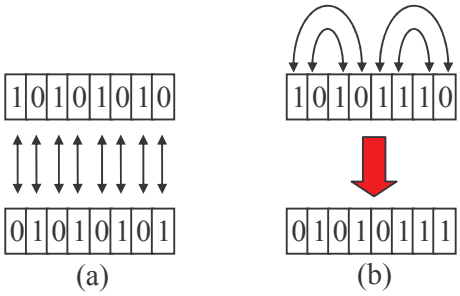

Figura 8 – Comandos opcionais para apresentação dos sinais no HTML; (a) negado; (b) invertesubstring.

4.2 Usuário do circuito

Ao carregar a página todos os sinais especificados pelo projetista serão mostrados na tela, conforme apresentado na Figura 6. A seguir pode-se alterar os valores das caixas de texto por novos valores. Para salvar estas novas alterações no arquivo de configurações basta clicar em *Salvar*. Também é possível executar um download para a placa de prototipação clicando em *Download*.

5 BITAnalyzer

Esta ferramenta abre arquivos de configuração do tipo RBT totais ou parciais. O arquivo de configuração é mostrado linha a linha com seu respectivo propósito, valor em binário, valor em hexadecimal e endereço. Neste programa é possível escolher uma parte de um arquivo de configuração total e gerar um arquivo de configuração parcial partindo de um protocolo de geração de arquivos parciais. Pode-se localizar um ou mais bits das LUTs do bitstream e alterar seus respectivos valores.

Para utilizar esta ferramenta é necessário apenas que o JDK esteja instalado na máquina. Descompacte o *bitanalyzer.zip* e execute o arquivo *bitanalyzer.bat*. Caso o JDK esteja funcionando corretamente, será a exibida a tela mostrada na Figura 9.

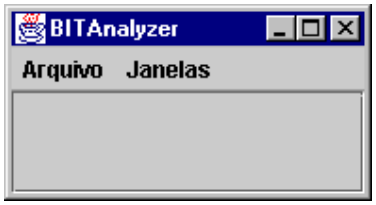

Figura 9 - Tela inicial do BITAnalyzer

O primeiro passo é abrir um bitstream (clicar em *Arquivo*, *Abrir*, selecionar o arquivo a ser aberto). A seguir todo o bitstream será exibido e comentado, e a tela que será mostrada é semelhante a Figura 10.

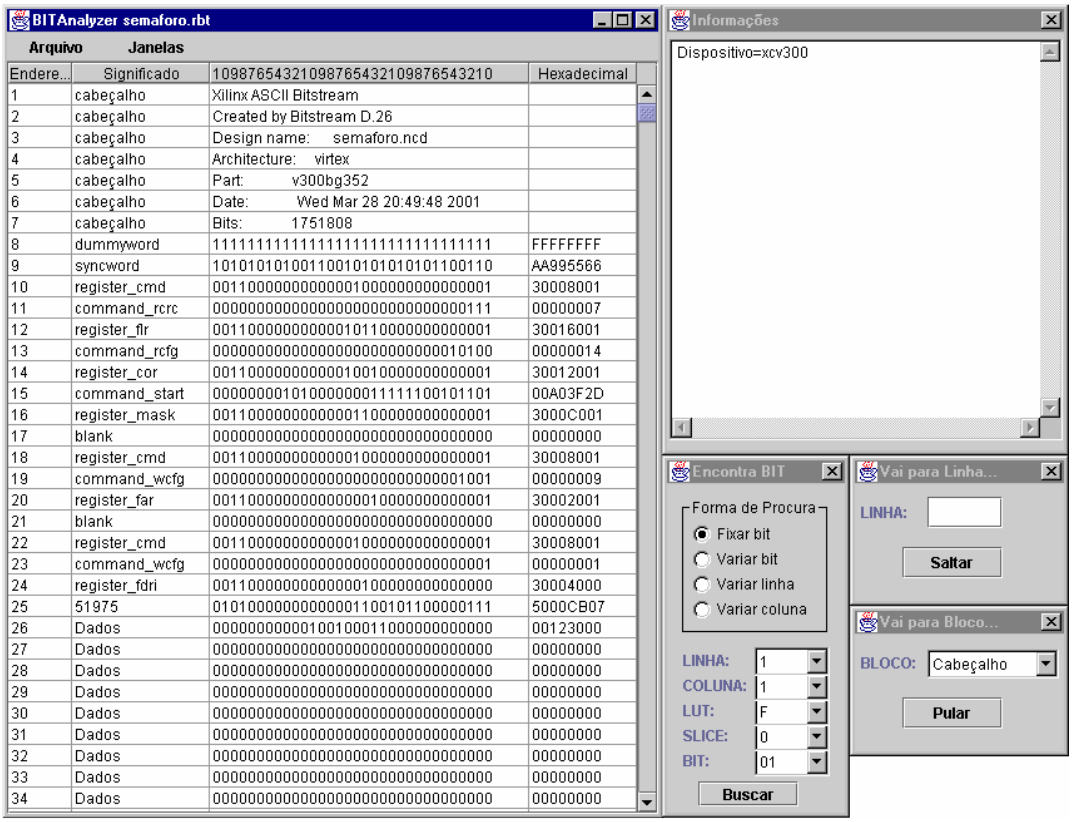

Figura 10 – Tela do BITAnalyzer com um bitstream carregado.

Como pode ser observado na Figura 10, existem três janelas para auxiliar na navegação do bitstream. A janela *encontra BIT* permite encontrar um determinado BIT de uma LUT (selecionar opção fixar bit), todos os bits de uma LUT (selecionar variar bit), todos os bits de uma determinada posição de bit de uma mesma linha (selecionar variar linha) e todos os bits de uma determinada posição de bit de uma mesma coluna (selecionar variar coluna). A janela *vai para linha* permite pular diretamente para uma determinada linha (digitar uma linha do bitstream e clicar em *saltar*). A janela *vai para bloco* permite ir diretamente para a primeira linha de um bloco (selecionar um bloco do bitstream e clicar em *pular*).

Quando qualquer uma das funções de navegação, explicada no parágrafo anterior, for utilizada, a janela principal selecionará a linha do bitstream que contém o primeiro bit da resposta. A janela *informações* fará um relatório com os seguintes dados: o número das linhas que contêm os bits solicitados e qual a posição destes bits nas linhas. Com essas informações o usuário pode alterar os bits das LUTs de forma a gerar um novo bitstream (total ou parcial).

Aconselha-se que antes de salvar o bitstream o usuário clique em outra linha da tabela, pois na versão atual do JDK (j2sdk1.4.0) só é possível ler um valor que foi modificado de uma célula depois que o usuário selecionar outra célula da tabela. Após isso é possível salvar as modificações em um bitstream total (clicar em *Arquivo*, *Salvar*) ou visualizar o bitstream parcial a ser gerado (clicar em *Janelas*, *bitstream parcial*) e salvar o bitstream parcial (clicar em *Arquivo* da janela do bitstream parcial e *Salvar parcial*).

Outra função que pode ser explicitamente chamada da ferramenta é a de calcular o CRC do bitstream (clicar em *Arquivo*, *Calcular CRC*) que verifica e informa qual foi o CRC que foi gerado para o bitstream que está aberto.

5.1 Protocolo de geração de bitstreams parciais

Como este programa tem o principal objetivo de estudar arquivos de configuração, não foi adotado um protocolo fixo para a geração de bitstreams parciais. Por este motivo o bitstream parcial é gerado segundo o protocolo definido no arquivo *prototipoparcial.txt*. Neste arquivo são colocados três tipos de informações: blocos que devem ser copiados do bitstream total, registradores e valores para os registradores. Um exemplo de *prototipoparcial.txt* e os possíveis parâmetros para a geração do arquivo parcial estão descritos, respectivamente, na Figura 11 e Tabela 1.

```
CABEÇALHO
dummyword
syncword
CMD
RCRC
COR
shutdown on
CMD
start
CRC
dado
CMD
RCRC
CMD
AGHIGH
COR
shutdown off
FAR
01 02 F 0 14
CMD
wcfg
FDRI
21
DADOS
CRC
dado
CMD
lfrm
FDRI
21
PREENCHIMENTO1
CMD
start
CTL
persist on
CRC
dado
PREENCHIMENTO2
```

Figura 11 – Exemplo de prototipoparcial.txt

BLOCOS	REGISTRADORES	VALORES	VALORES	VALORES
CABEÇALHO	FLR	Dummyword	Shutdown on (COR)	Done_cycle (COR)
COMANDOS 1	COR	Syncword	Shutdown off (COR)	Lck_cycle (COR)
DADOS	LOUT	Blank	Done_pipe yes (COR)	Gts_cycle (COR)
COMANDOS 2	STAT	Switch (CMD)	Done_pipe no (COR)	Gwe_cycle (COR)
RAM 0	MASK	Aghigh (CMD)	Drive_done open (COR)	Gsr_cycle (COR)
COMANDOS 3	CTL	Rcrc (CMD)	Drive_done high (COR)	Dado (CRC)
RAM 1	CMD	Rcap (CMD)	Single on (COR)	Persist on (CTL)
CRC 1	FDRO	Start (CMD)	Single off (COR)	Persist off (CTL)
COMANDOS 4	FDRI	Rcfg (CMD)	Oscfsel (COR)	Gts_usr_b on (CTL)
PREENCHIMENTO 1	FAR	Lfrm (CMD)	Ssclksrc cclk (COR)	Gts_usr_b off (CTL)
COMANDOS 5	CRC	Wcfg (CMD)	Ssclksrc userclk (COR)	Sbits enable (CTL)
CRC 2			Ssclksrc jtagclk (COR)	Sbits disable (CTL)
PREENCHIMENTO 2			Lock_wait (COR)	Sbits crc (CTL)

Tabela 1 - Lista de parâmetros válidos para o prototipoparcial.txt

Explicações sobre os registradores e os valores dos registradores podem ser encontradas no Application Note 151 da Xilinx [11].

5.2 Bitstream Parcial

Esta seção apresenta um exemplo de um bitstream parcial gerado a partir do protocolo de geração de parciais apresentado na Figura 11.

- Cabeçalho (Figura 12)

Algumas informações do bitstream parcial são cópias do bitstream total. São elas as linhas: 1, 2 e 4. As linhas 1 e 2 representam o tipo de bitstream e por quem foi gerado. O *design name* (linha 3) contém o nome do arquivo do projeto que gerou este bitstream. No momento em que o bitstream é salvo, esta linha é modificada para o nome do arquivo salvo. A linha 4 apresenta a arquitetura para o qual este bitstream foi gerado, neste caso é uma arquitetura tipo Virtex. O *part* (linha 5) indica qual é o dispositivo reconfigurável que está sendo utilizado. A data (linha 6) é atualizada para o momento em que a janela do bitstream parcial é aberta e ao salvar o arquivo ela é novamente atualizada. O número de bits (linha 7) é referente ao número de bits de dados que este bitstream parcial contém.

Endere...	Significado	10987654321098765432109876543210	Hexadecimal
1	cabeçalho	Xilinx ASCII Bitstream	
2	cabeçalho	Created by Bitstream D.26	
3	cabeçalho	Design name: semaforo.ncd	
4	cabeçalho	Architecture: virtex	
5	cabeçalho	Part: v300bg352	
6	cabeçalho	Date: Sun Sep 02 20:50:17 2001	
7	cabeçalho	Bits: 2859	

Figura 12 - Linhas de cabeçalho do bitstream

- Dummyword e Syncword (Figura 13)

As *dummywords* e *syncwords* são palavras que são enviadas pelo software no início da configuração do FPGA com o propósito de preparar e sincronizar o hardware. Após a recepção destas palavras o hardware está pronto para receber apenas comandos válidos.

Endere...	Significado	10987654321098765432109876543210	Hexadecimal
8	dummyword	11111111111111111111111111111111	FFFFFFFF
9	syncword	10101010100110010101010101100110	AA995566

Figura 13 - Linhas de sincronização do bitstream

- Informações de configuração de um bitstream parcial (Figura 14)

As linhas 10 a 23 indicam que está sendo enviado um bitstream parcial e em que posição do FPGA os dados que estão sendo enviados devem ser configurados.

A linha 10 e 11 significam que o contador do CRC deve ser inicializado. As linha 12 e 13 informam que o processo de configuração do FPGA está iniciando. As linhas 14 e 15 significam que na próxima verificação de CRC com sucesso a seqüência de inicialização será iniciada. A linha 16 fará um cálculo de CRC que comparará a linha 17 com a resposta deste cálculo. Se a resposta do cálculo for diferente do valor da linha 17 o registrador de estado ficará habilitado com erro de CRC. As linhas 18 e 19 indicam que o valor do CRC será zerado, significando que a partir daqui iniciará a seqüência de con-

figuração do FPGA. As linhas 20 e 21 colocam os tri-states em alta impedância no momento da reconfiguração dinâmica. As linhas 22 e 23 indicam que o processo de inicialização da configuração terminou. As linhas 24 e 25 fazem a configuração a partir da linha 1 coluna 2 *LUT* f *slice* 0 bit 14. Aplicando as equações de localização de um bit [10] obtém-se que este bit está no *major address* 46 e *minor address* 46. Essa informação do *major address* é convertida para binário e inserida nos bits de 24 a 17, de forma análoga, o *minor address* é convertido para binário e inserido nos bits de 16 a 9 da linha seguinte ao FAR. O comando de WCFG referente às linhas 26 e 27 significam que será feita uma escrita no dispositivo. A linha 28 (FDRI) indicará quantas palavras a partir da posição armazenada em FAR serão reconfiguradas. Os bits de 10 a 0 mostram o número de palavras a serem escritas, que serão sempre múltiplas ao número de palavras de um quadro (*frame*) do dispositivo.

Endere...	Significado	10987654321098765432109876543210	Hexadecimal
10	cmd	00110000000000001000000000000001	30008001
11	rcrc	00000000000000000000000000000111	00000007
12	cor	00110000000000010010000000000001	30012001
13	shutdown on	01010000000000001100101100000111	5000CB07
14	cmd	00110000000000001000000000000001	30008001
15	start	00000000000000000000000000000101	00000005
16	crc	00110000000000000000000000000001	30000001
17	dado	00000000000000000111111001111010	00007E7A
18	cmd	00110000000000001000000000000001	30008001
19	rcrc	00000000000000000000000000000111	00000007
20	cmd	00110000000000001000000000000001	30008001
21	aghigh	00000000000000000000000000001000	00000008
22	cor	00110000000000010010000000000001	30012001
23	shutdown off	01010000000000001001011000000111	50004B07
24	far	00110000000000000010000000000001	30002001
25	01 02 f 0 14	00000000010111000101110000000000	005C5C00
26	cmd	00110000000000001000000000000001	30008001
27	wcfg	00000000000000000000000000000001	00000001
28	fdri	00110000000000000100000000010101	30004015

Figura 14 – Linhas de configuração do dispositivo

- Dados que serão reconfigurados

A Figura 15 apresenta as 21 palavras que foram copiadas do bitstream total para o parcial.

Endere...	Significado	10987654321098765432109876543210	Hexadecimal
29	dados	00000010111011000000000010111011	02EC00BB
30	dados	00000000001011101100000000001011	002EC00B
31	dados	10110000000000101110000000000000	B002E000
32	dados	01000000000000000000000000000000	40000000
33	dados	00000000000000000000000000000000	00000000
34	dados	00001000000001000000110000000000	08040C00
35	dados	10110011000000000010110011000000	B3002CC0
36	dados	00001011001100000000001011001100	0B3002CC
37	dados	00000000101100110000000000101100	00B3002C
38	dados	11000000000010110011000000000010	C00B3002
39	dados	11001100000000001011001100000000	CC00B300
40	dados	00101100110000000000101100110000	2CC00B30
41	dados	00000010110011000000000010110011	02CC00B3
42	dados	00000000001011001100000000001011	002CC00B
43	dados	00110000000000101100110000000000	3002CC00
44	dados	10110011000000000010110011000000	B3002CC0
45	dados	00001011001100000000001011001100	0B3002CC
46	dados	00000000101100110000000000101100	00B3002C
47	dados	11000000000010110011000000000010	C00B3002
48	dados	11001100000000001011001100000000	CC00B300
49	dados	00101100110000000000101100110000	2CC00B30

Figura 15 - Linhas de dados a serem passados para o dispositivo

- Informações de finalização de um bitstream parcial

Nas linhas 50 e 51 outro cálculo de CRC é realizado para verificar a existência de erros na criação e/ou envio do *bitstream* ao FPGA. A seguir vem um comando (LFRM) que indica o último *frame* do *bitstream*. O FDRI a seguir escreverá um quadro inteiro que será útil apenas para uma leitura da placa (neste caso não utilizado). Logo após vem um comando de START para finalizar a seqüência de configuração. As linhas 78 e 79 indicam que o restante da configuração permanecerá no dispositivo. Em seguida é feito mais um cálculo de CRC para verificar se não houve erro na seqüência de finalização. E finalmente quatro linhas de preenchimento que fazem parte do protocolo de geração do *bitstream*.

Endere...	Significado	10987654321098765432109876543210	Hexadecimal
50	crc	00110000000000000000000000000001	30000001
51	dado	00000000000000000101000010010010	00005092
52	cmd	00110000000000001000000000000001	30008001
53	lfrm	00000000000000000000000000000011	00000003
54	fdri	00110000000000000100000000010101	30004015
55	preenchimento1	00000000000000000000000000000000	00000000
56	preenchimento1	00000000000000000000000000000000	00000000
57	preenchimento1	00000000000000000000000000000000	00000000
58	preenchimento1	00000000000000000000000000000000	00000000
59	preenchimento1	00000000000000000000000000000000	00000000
60	preenchimento1	00000000000000000000000000000000	00000000
61	preenchimento1	00000000000000000000000000000000	00000000
62	preenchimento1	00000000000000000000000000000000	00000000
63	preenchimento1	00000000000000000000000000000000	00000000
64	preenchimento1	00000000000000000000000000000000	00000000
65	preenchimento1	00000000000000000000000000000000	00000000
66	preenchimento1	00000000000000000000000000000000	00000000
67	preenchimento1	00000000000000000000000000000000	00000000
68	preenchimento1	00000000000000000000000000000000	00000000
69	preenchimento1	00000000000000000000000000000000	00000000
70	preenchimento1	00000000000000000000000000000000	00000000
71	preenchimento1	00000000000000000000000000000000	00000000
72	preenchimento1	00000000000000000000000000000000	00000000
73	preenchimento1	00000000000000000000000000000000	00000000
74	preenchimento1	00000000000000000000000000000000	00000000
75	preenchimento1	00000000000000000000000000000000	00000000
76	cmd	00110000000000001000000000000001	30008001
77	start	00000000000000000000000000000101	00000005
78	ctl	00110000000000001010000000000001	3000A001
79	persist on	00000000000000000000000001000000	00000040
80	crc	00110000000000000000000000000001	30000001
81	dado	00000000000000001110110010100001	0000ECA1
82	preenchimento2	00000000000000000000000000000000	00000000
83	preenchimento2	00000000000000000000000000000000	00000000
84	preenchimento2	00000000000000000000000000000000	00000000
85	preenchimento2	00000000000000000000000000000000	00000000

Figura 16 - Linhas de finalização e preenchimento do bitstream

6 RBTProgrammer

O RBTProgrammer têm muitas das características do pacote de ferramentas BIT-Programmer (seção 3). Esta é uma ferramenta que funciona apenas na máquina local e permite abrir um bitstream no formato RBT. A grande vantagem desta ferramenta sobre o BITProgrammer é que está não utiliza o JBits. Com ela é possível exibir as LUTs com valores diferentes do padrão e visualizar/modificar valores das LUTs. A própria ferramenta tem um atalho para realizar o download com o Hardware Debugger ou Impact.

Para utilizar esta ferramenta é necessário apenas que o JDK esteja instalado na máquina. Descompacte o *rbtprogrammer.zip* e execute o arquivo *rbtprogrammer.bat*. Caso o JDK esteja funcionando corretamente, será a exibida a tela mostrada na Figura 17.

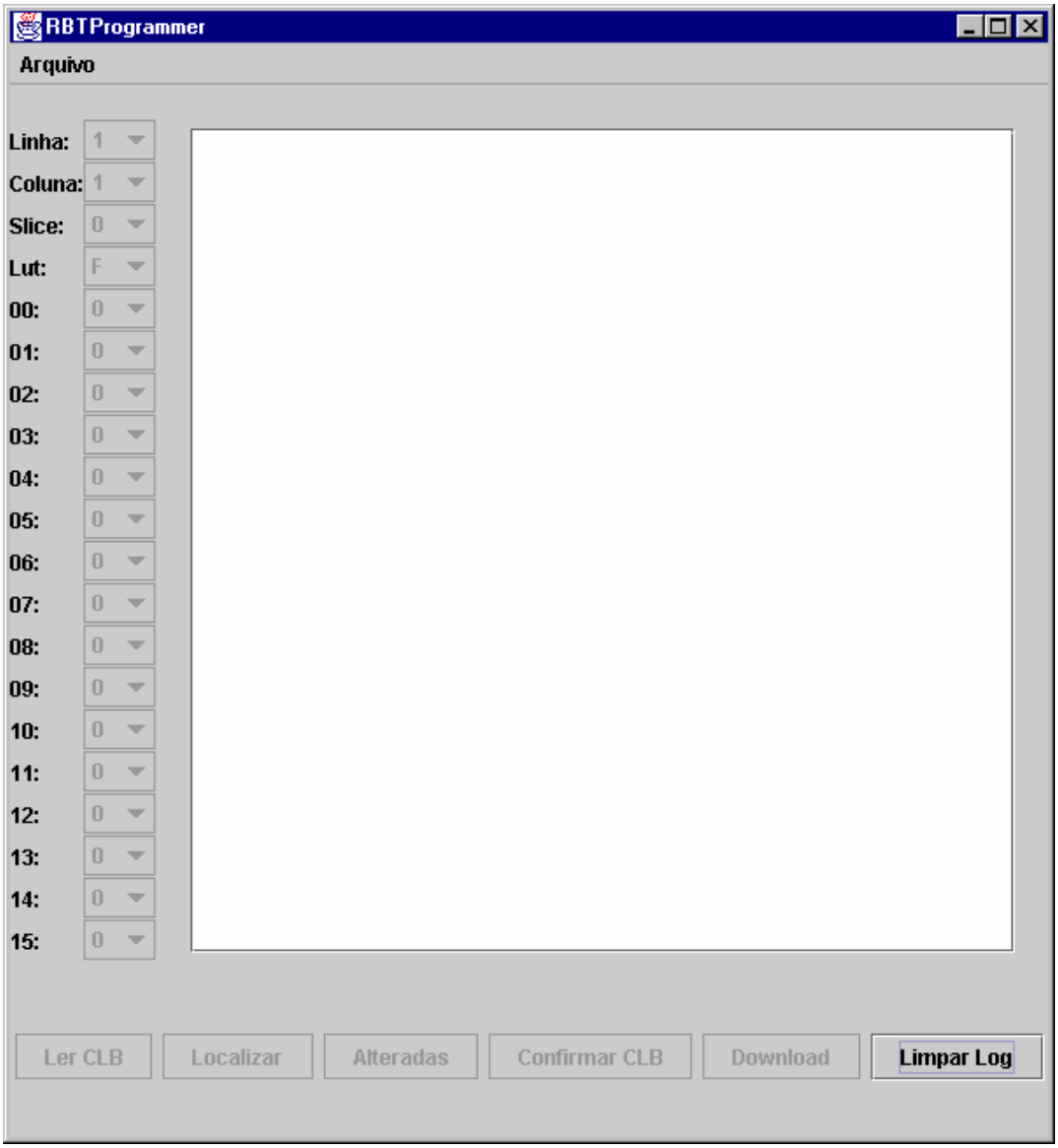

Figura 17 - Tela inicial do RBTProgrammer

O primeiro passo é abrir um bitstream (clicar em *Arquivo*, *Abrir*, selecionar o arquivo a ser aberto, *OK*). Para ler o conteúdo de uma CLB selecione os seus parâmetros (linha, coluna, LUT e slice) e clique em *Ler CLB*. Para modificar o conteúdo de uma CLB selecione os seus parâmetros (linha, coluna, LUT, slice e os 16 bits a serem configurados) e clique em *Confirmar CLB*. A seguir é possível salvar o bitstream (clicar em *Arquivo*, *Salvar Como*, escolher arquivo, *OK*).

A ferramenta também permite visualizar todas as LUTs com valores diferentes do padrão (clicar em *Alteradas*) ou até mesmo procurar uma configuração específica (selecionar os dezesseis bits de configuração da LUT e clicar em *Localizar*). Caso seja desejado também é possível fazer o download de um bitstream (clicar em *Download*) e limpar a tela que informa as operações realizadas (clicar em *Limpar Log*).

7 CoreUnifier

O objetivo do CoreUnifier é abrir múltiplos arquivos de configuração e gerar um novo bitstream a partir da seleção de partes de outros bitstreams (*cores*). O CoreUnifier é mais visual que as ferramentas anteriores e facilita a visualização dos valores das LUTs, automaticamente indicando as LUTs com valores diferentes do padrão. Ela é uma ferramenta que se aproxima muito de uma outra pré-existente chamada BoardScope da Xilinx. Entretanto, a grande vantagem é que está não utiliza o JBits e permite abrir arquivos no formato RBT. Ela, como algumas ferramentas anteriores, realiza o download utilizando o Hardware Debugger ou o Impact.

Para utilizar esta ferramenta é necessário apenas que o JDK esteja instalado na máquina. Descompacte o *coreunifier.zip* e execute o arquivo *coreunifier.bat*. Caso o JDK esteja funcionando corretamente, será a exibida a tela mostrada na Figura 18.

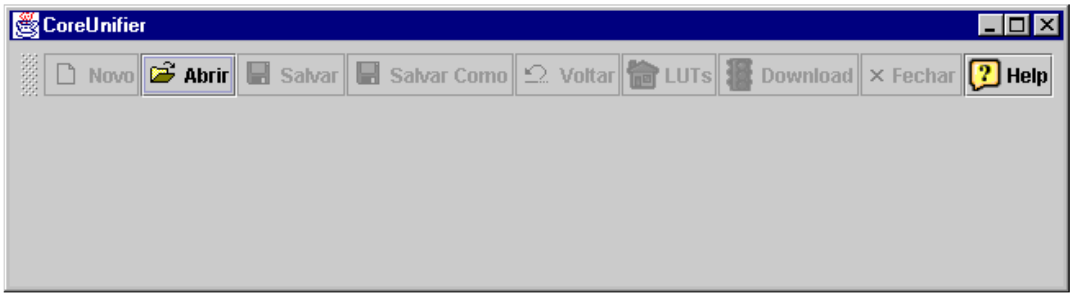

Figura 18 - Tela inicial do CoreUnifier

O primeiro passo é abrir o bitstream que será o base, ou seja, o arquivo que receberá a lógica de outros bitstreams (clicar em *Abrir*, selecionar o arquivo a ser aberto, *OK*). A seguir a tela anterior será atualizada para uma semelhante a mostrada na Figura 19.

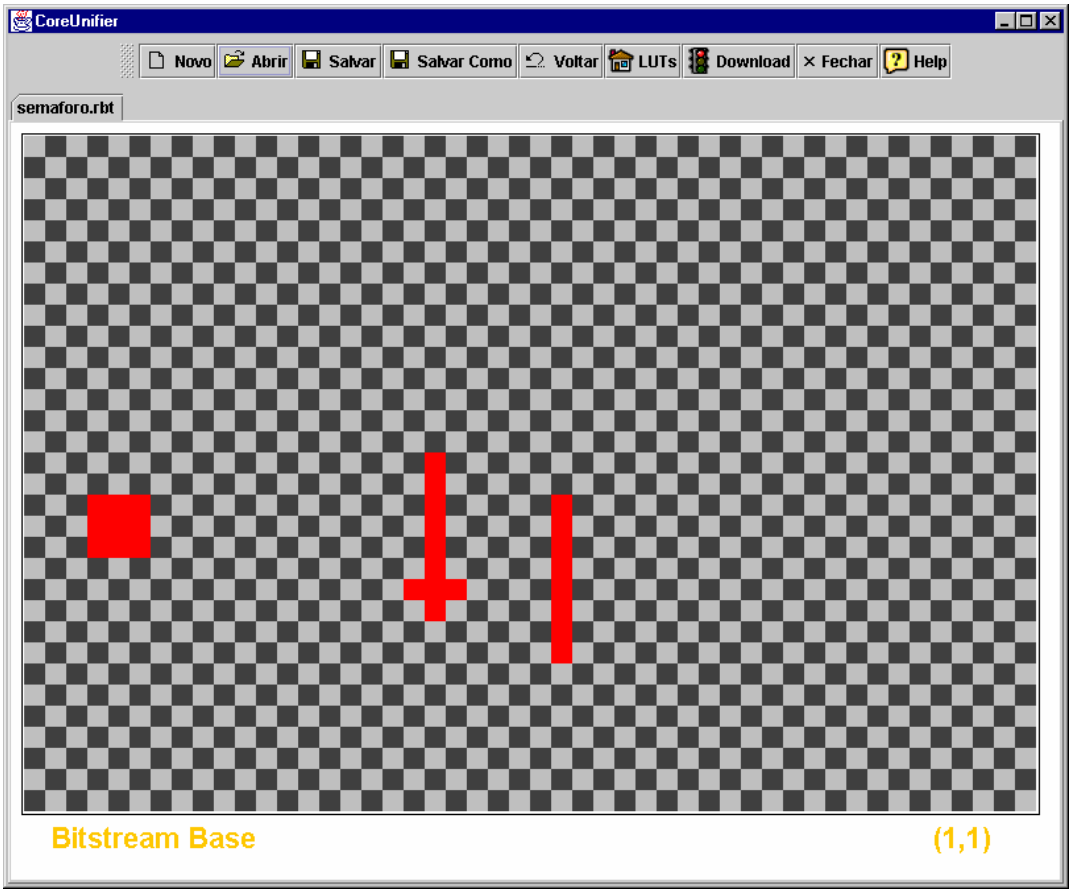

Figura 19 - Exemplo de bitstream base

Como o bitstream que foi carregado é destinado ao dispositivo XCV300 foram apresentadas 32 linhas e 48 colunas de quadrados cinzas. Cada quadrado representa uma CLB do dispositivo. Quadrados em cinza claro e cinza escuro estão com os valores padrão em suas LUTs, que são todos os valores em 1. Quadrados em vermelho significam que os valores das LUTs foram alterados.

Para visualizar os bits das LUT nesta ferramenta basta apertar no botão LUTs. Uma janela semelhante a Figura 20 mostrará os valores das LUTs dos CLBs que estão sendo apontados pelo mouse.

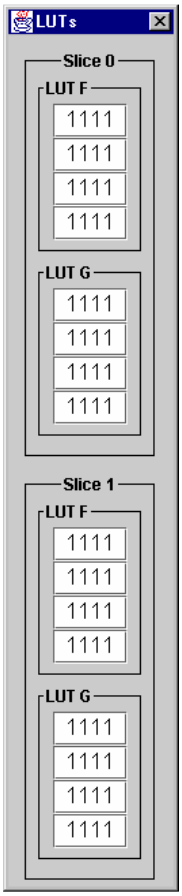

Figura 20 - Exemplo de visualização dos valores das LUTs de uma CLB

Para abrir outros bitstreams basta executar o mesmo processo de abertura de bitstreams citado anteriormente. Feito isto é possível alternar de um bitstream para o outro a partir das abas que contêm os nomes dos arquivos. Para copiar uma lógica do bitstream *semaforo2.rbt* para o bitstream base *semaforo.rbt* basta selecionar a área desejada mantendo pressionado o botão esquerdo do mouse. A Figura 21 mostra como fica a tela da ferramenta após a seleção do *core*.

Neste momento se você retornar para o bitstream base, clicando na aba mais a esquerda, verá que o *core* foi inserido exatamente como selecionado. Cada bitstream aberto gerará um *core* de cor diferenciada no bitstream base para melhor visualização dos *cores* inseridos. A Figura 22 mostra como fica o bitstream base.

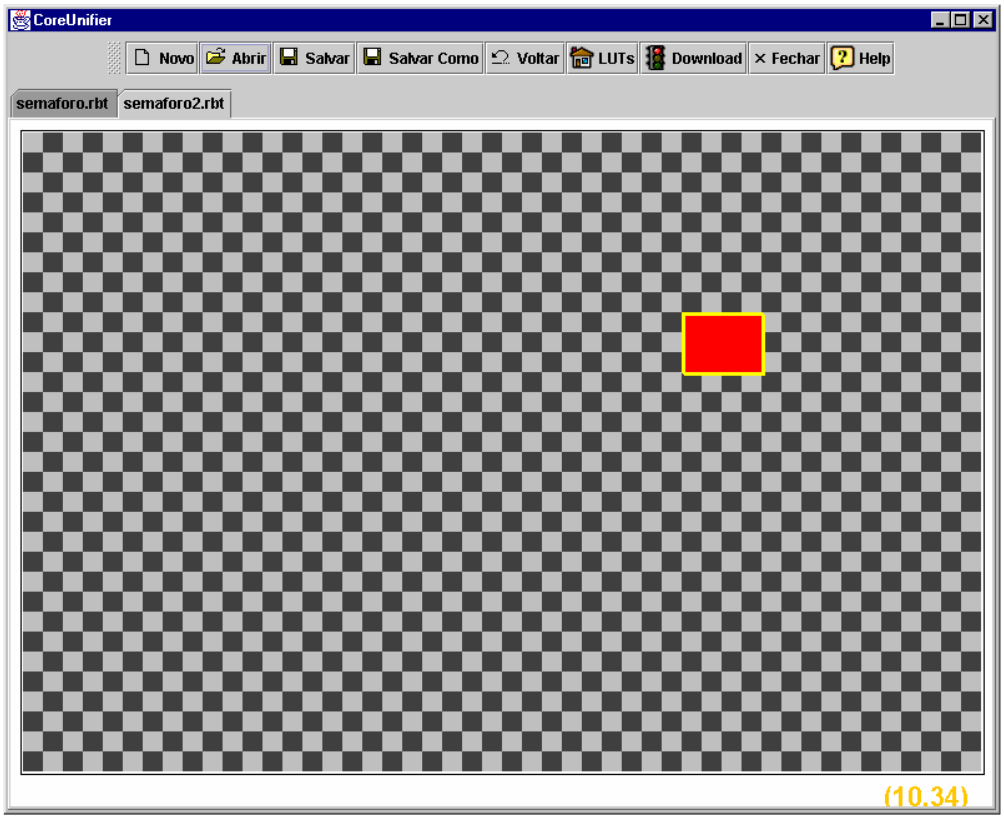

Figura 21 - Exemplo de seleção de um *core*

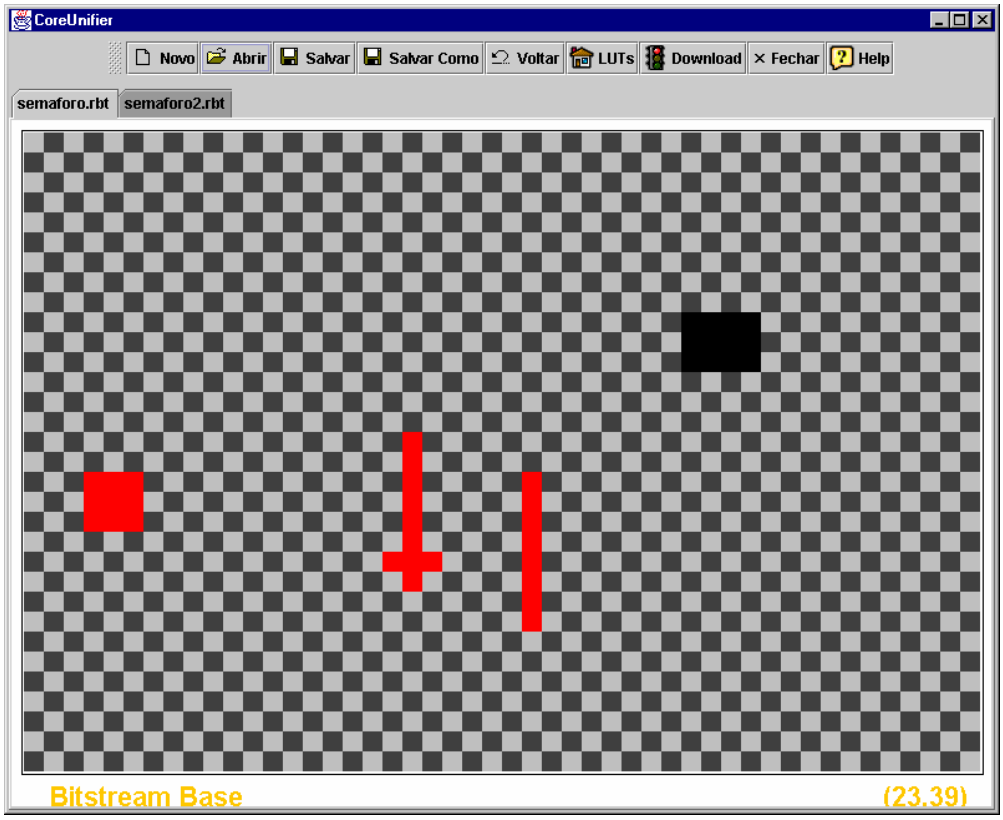

Figura 22 - Exemplo de inserção de um *core* no bitstream base

Caso você tenha selecionado a área errada pode-se apertar o botão de *Voltar* para cancelar a última ação. A partir disso é possível salvar o arquivo base com um nome novo automático clicando em salvar (o bitstream será salvo com o nome antigo mais "_s" no final do mesmo) ou é possível salvar com um nome a ser escolhido (clicar em *Salvar Como* e escolher o nome do arquivo e o diretório onde este será salvo).

Caso seja desejado também é possível fazer o download de um bitstream (clicar em *Download*) e para fechar um único bitstream clique em *Fechar*. Para iniciar um novo projeto clique em *Novo*. A Tabela 2 apresenta as teclas de atalho da ferramenta.

CTRL + A, O	Abre um novo bitstream
CTRL + S	Salva o bitstream base
CTRL + Z	Volta ao estado anterior
CTRL + L	Ativa/Desativa a descrição das LUTs
CTRL + D	Executa o download do bitstream base
CTRL + N	Cria um novo projeto
CTRL + F4	Fecha o bitstream atual
F1	Ajuda

Tabela 2 – Teclas de atalho da ferramenta CoreUnifier

8 BIT2RBT

Esta é uma ferramenta que transforma bitstreams do formato binário para o formato ASCII. Ela é uma ferramenta sem interface gráfica e funciona em qualquer plataforma que rode programas em Java.

Para gerar o bitstream *semaforo.rbt* a partir do *semaforo.bit* basta digitar:

java bit2rbt semaforo

9 RBT2BIT

Esta é uma ferramenta que transforma bitstreams do formato ASCII para o formato binário. Ela é uma ferramenta sem interface gráfica e funciona em qualquer plataforma que rode programas em Java.

Para gerar o bitstream *semaforo.bit* a partir do *semaforo.rbt* basta digitar:

java rbt2bit semáforo

10 CONCLUSÃO

As ferramentas de reconfiguração parcial, remota e dinâmica apresentadas atendem, atualmente, funcionalidades básicas do meio acadêmico. Elas podem ser facilmente portadas para equipamentos eletrônicos e eletrodomésticos disponíveis no mercado, provendo características e facilidades até hoje apenas sonhadas:

(1) fabricantes de equipamentos eletrônicos podem fazer atualizações do hardware do cliente sem a intervenção do usuário;

(2) usuários de equipamentos eletrônicos podem ter acesso ao seu produto a partir da Internet;

(3) modificações no hardware do produto podem ser efetuadas sem interromper a operabilidade do equipamento;

(4) defeitos de hardware podem ser avaliados e consertados sem a necessidade de levar o produto em assistências técnicas;

Sem dúvida a facilidade (2) é uma das características mais almejadas pelos usuários, permitindo comodidade a clientes residenciais e segurança a empresas. Para maiores exemplos de utilidades das ferramentas de reconfiguração parcial, remota e dinâmica consulte [12].

As ferramentas apresentadas são compatíveis com as famílias de dispositivos Virtex e VirtexE, pois as equações de endereçamento e bibliotecas de acesso aos elementos internos destas famílias foram disponibilizadas pelo fabricante. Um trabalho futuro é portar estas ferramentas para os dispositivos Virtex II, já que o fabricante lançou recentemente as bibliotecas (JBits 3) de acesso a esta família de FPGAs. É importante ressaltar que o JBits 3 não é mais compatível com os dispositivos Virtex e VirtexE, sendo necessário, portanto, manter o JBits 2 para acesso a estes dispositivos. Outra observação importante sobre o JBits 3 é que ele ainda não é compatível com a família de FPGAs Virtex II Pro da Xilinx.

11 REFERÊNCIAS BIBLIOGRÁFICAS

[1] Xilinx. **Virtex – Field Programmable Gate Arrays**. Apr. 2001. Available at: http://www.xilinx.com

[2] Xilinx. **The JBits 3.0 SDK for Virtex-II**. Nov. 2003. Available at: http://www.xilinx.com/labs/projects/jbits/

[3] Mesquita, D. **Contribuições para Reconfiguração Parcial, Remota e Dinâmica de FPGAs**. 2002. (Dissertação de Mestrado - Pontifícia Universidade Católica do Rio Grande do Sul - PPGCC - FACIN - Porto Alegre, RS, Brasil).

[4] Palma, J. **Métodos para desenvolvimento e distribuição de IP-cores**. 2002. (Dissertação de Mestrado - Pontifícia Universidade Católica do Rio Grande do Sul - PPGCC - FACIN - Porto Alegre, RS, Brasil).

[5] Mesquita, D.; Moraes, F.; Palma, J.; Möller, L.; Calazans, N. **Reconfiguração Parcial e Remota de Cores FPGAs**. In: 7th Workshop Iberchip, Mar, 2001.

[6] Mesquita, D.; Moraes, F.; Palma, J.; Möller, L.; Calazans, N. **Reconfiguração Parcial e Remota de Dispositivos FPGA da Família Virtex**. In: Seminário de Computação Reconfigurável, Aug. 2001.

[7] Möller, L.; Mesquita, D.; Moraes, F. **Tool-Set for Remote and Partial Reconfiguration**. In: XVII South Symposium on Microelectronics, Jun. 2002.

[8] Palma, J.; Mello, A.; Möller, L.; Moraes, F.; Calazans, N. **Core Communication Interface for FPGAs**. In: 15th Symposium on Integrated Circuits and Systems Design, (SBCCI´02). IEEE Computer Society Press, Sep. 2002.

[9] Moraes, F.; Mesquita, D.; Palma, J; Möller, L.; Calazans, N. **Development of a Tool-Set for Remote and Partial Reconfiguration of FPGAs**. In: Design, Automation and Test in Europe (DATE'03), Mar. 2003.

[10] Mesquita, D.; Moraes, F.; Palma, J.; Möller, L.; Calazans, N. **Remote and Partial Reconfiguration of FPGAs: tools and trends**. In: 10th Reconfigurable Architectures Workshop (RAW'03), Apr. 2003.

[11] Xilinx. **Virtex Series Configuration Architecture User Guide**. Nov. 2001. Available at: http://www.xilinx.com/xapp/xapp151.pdf

[12] Amory, A.; Junior, J. **Sistema Integrado e Multiplataforma Para Controle Remoto De Residências**. 2000. (Trabalho de Conclusão de Graduação em Ciência da Computação - Pontifícia Universidade Católica do Rio Grande do Sul - FACIN - Porto Alegre, RS, Brasil).