FILIPPO VALIANTE FILHO

FERRAMENTAS E METODOLOGIAS DE DESENVOLVIMENTO
PARA SISTEMAS PARCIALMENTE RECONFIGURÁVEIS

Dissertação apresentada à Escola
Politécnica da Universidade de São
Paulo para obtenção do título de
Mestre em Engenharia Elétrica

São Paulo
2008

FILIPPO VALIANTE FILHO

FERRAMENTAS E METODOLOGIAS DE DESENVOLVIMENTO
PARA SISTEMAS PARCIALMENTE RECONFIGURÁVEIS

Dissertação apresentada à Escola
Politécnica da Universidade de São
Paulo para obtenção do título de
Mestre em Engenharia Elétrica

Área de Concentração:
Microeletrônica

Orientador:
Prof. Dr. Edson Lemos Horta

São Paulo
2008

Este exemplar foi revisado e alterado em relação à versão original, sob responsabilidade única do autor e com a anuência de seu orientador.

São Paulo, 16 de junho de 2008.

Assinatura do autor _____

Assinatura do orientador _____

Valiante Filho, Filippo

Ferramentas e metodologias de desenvolvimento para sistemas parcialmente reconfiguráveis / F. Valiante Filho. -- São Paulo, 2008.

125 p.

Dissertação (Mestrado) - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Sistemas Eletrônicos.

1. Circuitos FPGA 2. Arquitetura reconfigurável 3. Computação reconfigurável 4. Circuitos integrados 5. Microeletrônica 6. CAD I. Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Sistemas Eletrônicos II. t.

A meu pai,
Filippo Valiante,
por há 20 anos ter me ensinado
os primeiros passos na Eletrônica.

AGRADECIMENTOS

Em primeiro lugar a Deus, porque até aqui Ele me ajudou.

Ao professor Edson Horta, pela orientação, ajuda, amizade, paciência, conselhos...

Ao professor Sergio Kofuji pela acolhida no PAD-LSI.

Aos professores Marius Strum e Maria Alice Ferreira da Poli-USP.

Ao professor Sérgio Miranda Paz da PUC-SP.

Aos meus alunos Thiago Rosa, Leonardo Machado e Daniel Silva pela ajuda com o PARBIT GUI.

Aos colegas professores e ex-professores da UNIb pelo apoio.

Aos amigos João, Marco e Rafael, companheiros de jornada na PUC e na Poli.

À Sônia Gomes pela intercessão, ajuda nos momentos difíceis e incentivo.

À Débora, meu amor, pela ajuda em todos os planos.

“Com Deus está a sabedoria e a força;
ele tem conselho e entendimento.”

Jó, cap. 12, verso 13.

RESUMO

Alguns tipos de FPGA (*Field Programmable Gate Array*) possuem a capacidade de serem reconfigurados parcialmente em tempo de execução formando um Sistema Parcialmente Reconfigurável (SPR), cuja utilização traz diversas vantagens dentre as quais a redução de custos. A maior utilização de SPRs enfrenta, como um dos fatores limitantes, a dificuldade de acesso e de utilização de ferramentas de desenvolvimento apropriadas. Este trabalho aborda os SPRs, suas aplicações e uma análise das ferramentas de desenvolvimento existentes. Posteriormente dedica-se ao aperfeiçoamento de uma dessas ferramentas, o PARBIT, com o desenvolvimento de uma interface gráfica de usuário (GUI, — *Graphical User Interface*) e a atualização de sua metodologia de desenvolvimento. As metodologias de projeto suportadas pelo fabricante do FPGA também são apresentadas. As metodologias são validadas através do projeto de um SPR.

Palavras-chave: FPGA. Sistema parcialmente reconfigurável. Lógica reconfigurável. Arquitetura reconfigurável. CAD para FPGAs.

ABSTRACT

Some types of FPGA (Field Programmable Gate Array) can be partially reconfigured during run-time forming a Partial Reconfigurable System (PRS). The use of PRSs brings several advantages like cost reduction. A larger use of PRSs faces a limiting factor: the difficult to access and use appropriate development tools. This work shows the PRSs, its applications and an analysis of the the existing development tools. Later, it dedicates to the improvement of one of these tools, the PARBIT, developing a graphical user interface (GUI) and updating its project methodology. The project methodologies supported by the manufacturer of the FPGA are also presented. The methodologies are validated through the design of a PRS.

Keywords: FPGA. Partial reconfigurable system. Reconfigurable logic. Reconfigurable architecture. CAD for FPGAs.

LISTA DE ILUSTRAÇÕES

Figura 1 - Reconfiguração em tempo de compilação (1).....	17
Figura 2 - Reconfiguração Dinâmica Global (1).....	17
Figura 3 - Reconfiguração Dinâmica Local.....	18
Figura 4 - Área reconfigurável definida no modo Slice (23).....	33
Figura 5 - Área reconfigurável original (esq.) e alvo (dir.) no modo Block (23).....	34
Figura 6 - Módulo reconfigurável original (esq.) e área destinada no componente alvo (dir.) no modo BIP (39).....	35
Figura 7 - Aparência de um software de desenvolvimento para FPGAs.....	43
Figura 8 - Exemplo de seleção gráfica de área.....	50
Figura 9 - Aspecto final da interface gráfica do PARBIT.....	53
Figura 10 - Fluxo de projeto com FPGAs Xilinx.....	57
Figura 11 - Visão geral e detalhada da busmacro criada.....	59
Figura 12 - Metodologia de projeto baseada em módulos.....	60
Figura 13 - Metodologia de projeto de SPRs na versão Partial Reconfiguration Early Access do ISE.....	62
Figura 14 - Fluxos possíveis para a geração dos bitstreams parciais.....	65
Figura 15 - Captura de tela da GUI do PARBIT.....	66
Figura 16 - Captura da tela de seleção de área nos modos Block e Target.....	67
Figura 17 - Fluxo de projeto simplificado para os MPRs adicionais.....	68
Figura 18 - Fluxo de projeto para apenas um MPR.....	69
Figura 19 - Diagrama básico do SPR com a distribuição temporal dos MPRs.....	72
Figura 20 - Resultado do MPR somador.....	75
Figura 21 - Resultado da parte estática.....	76
Figura 22: Resultado da configuração completa com o MPR somador.....	77
Figura 23 - Projeto roteado para a configuração inicial (MPR de captura).....	78
Figura 24 - Projeto roteado com o MPR somador.....	79
Figura 25 - MPR subtrator ladeado pelas conexões entre as busmacros e os pinos de E/S.....	80

LISTA DE TABELAS

Tabela 1.1 - Cronologia das ferramentas de desenvolvimento.....	31
Tabela 1.2 - Comparativo entre as ferramentas de desenvolvimento de SPRs.....	36
Tabela 2.1 - Opções disponíveis para a utilização do PARBIT (23)(39).....	41
Tabela 2.2 - Menu hierárquico da área de processos da interface do PARBIT.....	46
Tabela 2.3 - Dimensões dos FPGAs (51)(52)(53).....	49
Tabela 2.4 - Comparativo da utilização do PARBIT: linha de comando X GUI.....	54

LISTA DE ABREVIATURAS E SIGLAS

API	Application Programming Interface
ASIC	Application Specific Integrated Circuit
BCD	Binary Coded Decimal
BIP	Bitstream Intellectual Property
bit	binary digit
BITPOS	BITstream POSitioner
CAD	Computer-Aided Design
CLAY	Configurable Logic Array
CLB	Configurable Logic Block
CRC	Cyclic Redundancy Check
CTR	Compile-Time Reconfiguration
DHP	Dynamic Hardware Plugins
DLL	Dynamic Link Library
E/S	Entrada e Saída
EDIF	Electronic Data Interchange Format
FPGA	Field Programmable Gate Array
FPSLIC	Field Programmable System Level Integrated Circuits
GUI	Graphical User Interface
HDL	Hardware Description Language
HPC	High Performance Computing
IDE	Integrated Development Environment
JERC6K	Java Environment for Reconfigurable Computing
JTAG	Joint Test Action Group
LCD	Liquid Crystal Display

LUT	Look-up Table
MHz	Mega Hertz
MPR	Módulo Parcialmente Reconfigurável
NIDS	Network Intrusion Detection Systems
PARBIT	PARtial Bltfile Transformer
pBITPOS	partial BITstream POSitioner
PCI	Placa de Circuito Impresso
PLD	Programmable Logic Device
PR	Partial Reconfiguration
RAM	Random Access Memory
REPLICA	Relocation per online Configuration Alteration
SDR	Software Defined Radio
SEU	Single Event Upsets
SPR	Sistema Parcialmente Reconfigurável
TBUF	Tristate (3-state) Buffer
RTR	Run-Time Reconfiguration
ULA	Unidade Lógica e Aritmética
VHDL	Very High Speed Integrated Circuit Hardware Description Language
WURM	Wearable Unit with Reconfigurable Modules
XBI	Xilinx Bitstream Interface
XPART	Xilinx Partial Reconfiguration Toolkit

SUMÁRIO

INTRODUÇÃO.....	14
1 SISTEMAS PARCIALMENTE RECONFIGURÁVEIS.....	17
1.1 EVOLUÇÃO HISTÓRICA DA RECONFIGURAÇÃO PARCIAL.....	19
1.2 APLICAÇÕES DA RECONFIGURAÇÃO PARCIAL.....	20
1.3 FERRAMENTAS PARA DESENVOLVIMENTO DE SPRs.....	25
1.3.1 Histórico das ferramentas de desenvolvimento de SPRs.....	27
1.3.2 Xilinx ISE - Partial Reconfiguration Early Access.....	30
1.3.3 JBits.....	32
1.3.4 PARBIT.....	33
1.3.5 BITPOS.....	35
1.3.6 Comparativo entre as ferramentas de desenvolvimento de SPRs....	36
2 INTERFACE GRÁFICA DE USUÁRIO PARA O PARBIT.....	39
2.1 OPERAÇÃO DO PARBIT.....	39
2.2 ESPECIFICAÇÃO DA INTERFACE GRÁFICA.....	42
2.2.1 Definições da interface gráfica do PARBIT.....	44
2.2.1.1 Leitura do modelo do FPGA.....	48
2.2.1.2 Seleção das áreas dos componentes.....	49
2.2.1.3 Estimativa do tempo de reconfiguração.....	51
2.3 IMPLEMENTAÇÃO DA INTERFACE GRÁFICA.....	51
2.4 COMPARAÇÃO DO USO DO PARBIT COM O PARBIT GUI.....	53
3 METODOLOGIAS DE PROJETO DE SPRs.....	55
3.1 CONCEITOS BÁSICOS.....	56
3.2 METODOLOGIA DE PROJETO DE SPRs BASEADA EM MÓDULOS.....	59
3.3 METODOLOGIA UTILIZANDO A VERSÃO PARTIAL RECONFIGURATION EARLY ACCESS DO SOFTWARE ISE.....	62
3.4 METODOLOGIA DE PROJETO UTILIZANDO O PARBIT.....	64
4 IMPLEMENTAÇÃO DE SPR.....	72
4.1 IMPLEMENTAÇÃO COM A METODOLOGIA PARA A VERSÃO PARTIAL RECONFIGURATION EARLY ACCESS DO ISE.....	73

4.2 IMPLEMENTAÇÃO COM A METODOLOGIA DO PARBIT.....	77
5 CONCLUSÕES.....	81
5.1 TRABALHOS FUTUROS.....	83
REFERÊNCIAS.....	85
APÊNDICE A – ROTEIRO DE CRIAÇÃO DE UMA BUSMACRO.....	91
APÊNDICE B – ROTEIRO DE PROJETO DA METODOLOGIA PARA A VERSÃO PARTIAL RECONFIGURATION EARLY ACCES DO ISE.....	95
APÊNDICE C – ROTEIRO DE PROJETO UTILIZANDO A METODOLOGIA DO PARBIT.....	105

INTRODUÇÃO

O arranjo lógico programável em campo (FPGA, — *Field Programmable Gate Array*), é um tipo de dispositivo lógico programável (PLD, — *Programmable Logic Device*) que permite a configuração e reconfiguração de qualquer circuito digital. Os FPGAs têm sido cada vez mais utilizados em computação reconfigurável e no desenvolvimento de sistemas digitais em geral, substituindo inclusive os circuitos integrados de aplicação específica (ASIC, — *Application Specific Integrated Circuit*) convencionais.

Alguns modelos de FPGAs suportam não só uma reconfiguração completa, mas também a reconfiguração parcial, isto é, a alteração da programação de apenas uma área do componente, sem afetar o funcionamento dos circuitos programados nas demais áreas. Os sistemas que empregam essas características são chamados de “Sistemas Parcialmente Reconfiguráveis” (SPRs).

A utilização de SPRs traz vantagens como a possibilidade de se utilizar componentes menores e a redução da área e da complexidade da placa de circuito impresso (PCI); ou seja, benefícios técnicos e econômicos.

A maior utilização dos SPRs está limitada por dois fatores principais:

- tempo de reconfiguração da área do componente, que deve atender às exigências da aplicação;
- dificuldade de acesso à ferramentas de *software* para desenvolvimento de SPRs, que ainda se encontram em fase de desenvolvimento, ou apresentam falhas e restrições.

O primeiro fator limitante é condicionado aos requisitos de projeto da aplicação em desenvolvimento. Se uma aplicação demandar um tempo de reconfiguração menor do que o possível de ser obtido com determinado componente, pode-se tentar um outro componente; se isso não resolver será necessário alterar o projeto, ou mesmo desistir da utilização da reconfiguração parcial. Este fator limitante é inerente ao componente e, em primeira instância, melhorias nesse aspecto são restritas ao fabricante que pode lançar novos componentes que possuam menor tempo de

reconfiguração, ou melhor suporte à reconfiguração parcial que resulte em um menor tempo de reconfiguração.

O segundo fator limitante, o acesso à ferramentas de desenvolvimento adequadas, permite e carece da contribuição dos meios acadêmico e industrial e tem se mostrado de grande demanda e relevância, bem como uma área de pesquisa em expansão, motivando o foco do presente trabalho.

De modo abrangente, os objetivos do trabalho são:

- estudo das ferramentas de desenvolvimento para SPRs;
- aperfeiçoamento de uma ferramenta para projeto de SPRs;
- evolução da metodologia de projeto que utiliza a ferramenta aperfeiçoada;
- apresentação das metodologias de projeto de SPRs propostas pelo fabricante dos FPGAs;
- validação das metodologias apresentadas através da implementação de um SPR como estudo de caso.

Esta dissertação foi estruturada para apresentar a concretização desses objetivos de modo claro e encadeado. Desta forma, o primeiro capítulo apresenta uma introdução aos SPRs, sua contextualização histórica, uma revisão de suas aplicações e uma descrição e análise das ferramentas de desenvolvimento de SPRs existentes.

O segundo capítulo detalha o aperfeiçoamento da ferramenta de desenvolvimento PARBIT, com a especificação e o desenvolvimento de uma interface gráfica de usuário (GUI). O PARBIT é utilizado atualmente por pesquisadores em trabalhos acadêmicos, especialmente de pós-graduação, em todo o mundo. Os objetivos da interface gráfica são tornar o *software* mais amigável e de aprendizado mais fácil e rápido, possibilitando ampliar esse leque de usuários para estudantes de graduação e profissionais da indústria.

O terceiro capítulo detalha as metodologias de projeto de SPRs propostas pelo fabricante dos FPGAs e a metodologia utilizando o PARBIT com a interface gráfica desenvolvida. Alguns conceitos importantes para a compreensão das metodologias também são apresentados.

O quarto capítulo apresenta o projeto de um SPR de uma calculadora, implementado como estudo de caso para a validação da metodologia de projeto do

fabricante e do PARBIT.

Nas conclusões são resumidos os principais resultados obtidos, as contribuições da pesquisa e as propostas para trabalhos futuros.

1 SISTEMAS PARCIALMENTE RECONFIGURÁVEIS

O FPGA (*Field Programmable Gate Array*) é um tipo de dispositivo lógico programável que permite a configuração e reconfiguração de um sistema digital um número praticamente ilimitado de vezes. Segundo Hutchings (1), essa reconfiguração pode ser implementada de duas maneiras: reconfiguração em tempo de compilação (*Compile-Time Reconfiguration*, ou CTR), e reconfiguração em tempo de execução (*Run-Time Reconfiguration*, ou RTR).

Na reconfiguração em tempo de compilação, o FPGA tem a configuração carregada quando é inicializado e só pode ser reconfigurado em uma nova inicialização do sistema, conforme a Figura 1.

Figura 1 - Reconfiguração em tempo de compilação (1)

A reconfiguração em tempo de execução também é chamada de reconfiguração dinâmica; nela o FPGA troca de configuração sem interromper o funcionamento do sistema. Há dois tipos de reconfiguração dinâmica, a global e a local. Na reconfiguração dinâmica global todo o dispositivo é reprogramado de uma vez, conforme a Figura 2; nota-se que há uma divisão temporal do funcionamento do sistema, o dispositivo carrega e executa a configuração A e depois a configuração B, repetindo o ciclo até que seja reinicializado .

Figura 2 - Reconfiguração Dinâmica Global (1)

Na reconfiguração dinâmica local, apenas uma parte do dispositivo é reprogramada de cada vez, ou seja, apenas uma área física determinada; as outras áreas podem

ser mantidas fixas, ou reconfiguradas em diferentes momentos, conforme exemplificado na Figura 3, onde em um primeiro momento são configurados os circuitos A e B e, em um segundo momento, o circuito B dá lugar a dois outros circuitos, C e D, enquanto A é mantido inalterado (e possivelmente sem ter seu funcionamento interrompido), repetindo-se o ciclo até uma nova inicialização. É possível também reconfigurar mais de uma área do dispositivo. Em (2) essa técnica é apresentada com a analogia de um *hardware* virtual para o sistema, assim como a memória virtual está para um computador.

Figura 3 - Reconfiguração Dinâmica Local

Quando o sistema utiliza reconfiguração local é denominado Sistema Parcialmente Reconfigurável (SPR). Algumas das vantagens obtidas com a utilização de SPRs são (3)(4):

- possibilidade de efetuar atualizações parciais de *hardware* em campo, inclusive remotamente, quer sejam correções, quer sejam novas funcionalidades. Por exemplo: um roteador pode ter uma única porta modificada com um novo protocolo de comunicação, ou algoritmo de compactação, enquanto as demais portas continuam funcionando normalmente e sem interrupções.
- emprego de algoritmos adaptativos em *hardware*;
- redução do número de componentes empregados em um sistema;
- possibilidade de utilizar um FPGA de menor capacidade e, conseqüentemente, de menor custo;
- uso mais eficiente da área da placa de circuito impresso.

Ambas as referências citam ainda a redução de consumo de energia, porém em (5)

é mostrado que o consumo de energia durante a reconfiguração é bastante relevante o que leva a questionar essa vantagem. Dependendo das características do sistema e de quantas vezes ele é reconfigurado, o consumo pode não ser reduzido, ou ser até mesmo maior do que o de um sistema que não utilize reconfiguração parcial.

Os SPRs podem ser utilizados em: computação reconfigurável (2); sistemas tolerantes à falhas ou redundantes; equipamentos que implementem algoritmos em *hardware* (criptografia, p.ex.) ou reconhecimento de padrões (processamento de imagem, p.ex.) e equipamentos que necessitem de atualização remota.

Para se implementar um SPR é necessário o uso de ferramentas de projeto auxiliado por computador (CAD) mais complexas. Essas ferramentas devem ser capazes de gerar arquivos de configuração, chamados *bitstreams*, que contêm apenas os bits de controle para alterar a parte do dispositivo que será reconfigurada. A criação e a evolução dessas ferramentas foi, e ainda é, um dos principais fatores a permitir a criação e a sofisticação de aplicações utilizando SPRs.

O próximo item apresenta uma evolução histórica dos SPRs, suas principais aplicações são discutidas no item [1.2](#) e as principais ferramentas CAD para SPRs são apresentadas no item [1.3](#).

1.1 EVOLUÇÃO HISTÓRICA DA RECONFIGURAÇÃO PARCIAL

De acordo com Tommiska (6), as primeiras referências a sistemas que fossem capazes de reprogramar a si mesmos foram feitas por John von Neumann no final da década de 40 e começo da década de 50. A primeira sugestão de implementação de um componente com lógica reprogramável foi feita em 1967 por Sven Wahlstrom, que propôs incluir portas adicionais para personalizar um arranjo de circuitos integrados; porém, foi apenas no final da década de 80 e começo da década de 90 que surgiram as primeiras plataformas a utilizar substancialmente lógica reconfigurável.

O primeiro FPGA foi lançado pela Xilinx Inc. em 1985 (6), mas a reconfiguração parcial foi implementada em princípio pela Algotronix, com o CAL1024, criado em 1988 e lançado no mercado em 1989 com a criação da companhia (7). A National Semiconductors lançou em 1993 a linha CLAY (*Configurable Logic Array*) (8), que suportava reconfiguração parcial. Também em 1993 a Algotronix foi vendida para a Xilinx Inc. e sua tecnologia serviu de base para a família XC6200, lançada em 1995, com suporte à reconfiguração parcial e acesso interno à porta de reconfiguração, possibilitando a criação de sistemas auto-reconfiguráveis (7)(9)(10). Em 1998 a Xilinx lançou a família Virtex, com melhor suporte à reconfiguração parcial. As versões posteriores da família Virtex também suportam reconfiguração parcial, a saber: Virtex-E, lançada em 1999; Virtex-II, lançada em 2000; Virtex-II Pro, lançada em 2002; Virtex-4, lançada em 2004 e Virtex-5, lançada em 2006. A família Spartan-3 de FPGAs de menor custo, lançada pela Xilinx em 2004, também suporta a reconfiguração parcial (11).

O suporte à reconfiguração parcial é também oferecido pela Atmel, nos FPGAs das famílias AT40K e FPSLIC (*Field Programmable System Level Integrated Circuits*), sendo que esta última incorpora um microcontrolador AVR. A Atmel dá o nome de *cache logic* para sua tecnologia de reconfiguração parcial em tempo de execução, que foi lançada em 1994 (4). O foco da Atmel, no entanto, está em sistemas de pequeno porte, sendo que seus FPGAs têm no máximo 50.000 portas lógicas, enquanto que a Xilinx oferece opções com mais de um milhão de portas.

Neste item a evolução da reconfiguração parcial foi abordada considerando-se apenas os dispositivos que a suportam, porém sua utilização só foi possível com o desenvolvimento de ferramentas adequadas de projeto (abordadas no item [1.3](#)).

1.2 APLICAÇÕES DA RECONFIGURAÇÃO PARCIAL

Em (4) são sugeridos quatro grandes grupos gerais de aplicações para SPRs, aos quais pode-se acrescentar um quinto grupo com base nas listadas a seguir neste item. Esses grupos de aplicações são:

- aplicações onde consumo de energia e tamanho são críticos, tais como computadores portáteis, instrumentos alimentados por baterias, equipamentos de comunicação e equipamentos médicos portáteis;
- aplicações de computação intensiva com a implementação de algoritmos e filtros em *hardware*, tais como computação gráfica, processamento de imagem, compressão de dados, reconhecimento de fala e padrões;
- equipamentos reprogramáveis nas áreas de controle, instrumentação, testes, telecomunicações e computação para aplicações especiais;
- aceleração de aplicações na forma de co-processadores;
- aplicações tolerantes à falhas, onde problemas podem ser corrigidos através da reconfiguração parcial acionada remotamente, ou mesmo por algum tipo de auto-diagnóstico na própria aplicação.

A seguir são abordadas algumas aplicações comerciais ou de pesquisa que empregam reconfiguração dinâmica parcial, de modo a não esgotar todas as aplicações existentes, mas a oferecer um panorama abrangente das aplicações da reconfiguração parcial na indústria e na academia. É preciso ressaltar que, para algumas aplicações o uso de reconfiguração parcial não é mencionado explicitamente, porém pode ser deduzido a partir das informações disponíveis. Há também uma série de aplicações onde existe a reconfiguração dinâmica de determinados parâmetros utilizados no sistema e armazenados em uma memória interna no FPGA, mas sem a reconfiguração do circuito lógico em si, ou então que utilizem a reconfiguração global; esses casos, porém, não fazem parte do escopo deste trabalho.

A computação "vestível" tem como um dos maiores desafios de projeto combinar alto desempenho e baixo consumo de energia em uma mesma aplicação. O alto desempenho é necessário para tarefas como reconhecimento de voz e imagem e o baixo consumo é imperativo por se tratarem de dispositivos alimentados por baterias. O uso de um *hardware* reconfigurável atende plenamente este requisito e, para isso, os autores de (12) idealizaram e implementaram o protótipo de uma unidade vestível autônoma com módulos reconfiguráveis (WURM, — *Wearable Unit with Reconfigurable Modules*) onde podem ser implementadas diversas aplicações sob demanda.

A reconfiguração parcial também pode ser utilizada em sistemas de processamento de imagem e vídeo em tempo real, com aplicação em equipamentos médicos de diagnóstico por imagem, visualização científica e indústrias dos segmentos de semicondutores, aeroespacial e de defesa (13).

A própria Xilinx propõe em (14) a fabricação de equipamentos que possam ser atualizados, testados e até mesmo reparados remotamente através de uma rede, inclusive a Internet, ou rede satélite, o que permitiria a reconfiguração de uma aplicação rodando em um satélite, por exemplo. Em (15) a proposta de reconfiguração à distância através da Internet é proposta para aplicações automotivas envolvendo entretenimento (música, vídeo, jogos), navegação (sistemas baseados em GPS, celular e Internet), segurança (desde o chamado de socorro em um acidente à manutenção de distância segura do carro da frente) e auto-diagnóstico. A reconfiguração parcial à distância traria como benefício evitar a obsolescência de produtos, protocolos e aplicações, bem como a possibilidade de se agregar novas funcionalidades aos automóveis.

A aplicação de SPRs em automóveis, mais especificamente em sistemas para o auxílio do motorista como controle de distância do veículo à frente e do espaço dentro da faixa de rodagem, é também apresentada em (16). No caso, a reconfiguração parcial é utilizada para os algoritmos de processamento de imagem, que devem mudar conforme o ambiente em que o carro se encontra. Estes algoritmos são foco de pesquisa e precisarão, por mais alguns anos, serem constantemente atualizados.

Em (17) é mostrado um microprocessador superescalar cujas unidades de execução são configuradas dinamicamente de acordo com as instruções em execução. Até três unidades são configuradas simultaneamente, podendo ser ULAs de inteiros, multiplicadores/divisores de inteiros, unidades de carga e armazenamento, ULAs de ponto flutuante e multiplicadores/divisores de ponto-flutuante.

Em (18) é mostrada a aplicação de SPRs na área de instrumentação, com a criação de instrumentos de medição personalizados baseados em FPGAs e com suporte à reconfiguração dinâmica global e local de modo a obter performance e flexibilidade.

O uso de reconfiguração parcial de FPGAs em automação é explorado em (19), onde foi implementado um sistema controlador capaz de executar múltiplos

algoritmos de controle. Cada algoritmo é desenvolvido como se fosse um circuito dedicado e apenas um por vez é executado no componente. O módulo estático se encarrega, entre outras coisas, de supervisionar a reconfiguração carregando o algoritmo de controle necessário enquanto os demais ficam armazenados em uma memória.

As áreas de Telecomunicações e Comunicação de Dados são das que mais têm aproveitado a possibilidade da reconfiguração parcial com as mais diversas aplicações.

Plugins dinâmicos de *hardware* (DHP, — *Dynamic Hardware Plugins*) (20)(21) permitem que múltiplas aplicações sejam carregadas e removidas dinamicamente em um único *hardware* e sejam executadas em paralelo, possibilitando a implementação de roteadores, *firewalls* e outros equipamentos de rede programáveis e de alto desempenho. Como esses equipamentos possuem múltiplas portas e a reconfiguração é parcial, o funcionamento da rede não é interrompido uma vez que o equipamento inteiro não é desligado, mas apenas a porta onde o DHP será carregado ou removido. Esta idéia é similar ao conceito de DLLs (*Dynamic Link Libraries*) existente em programação, onde módulos de *software* podem ser anexados e descartados conforme demandado pela aplicação. Alguns dos DHPs propostos são para criptografia, vídeo-conferência e operações sobre *streaming* de áudio ou vídeo em alta velocidade.

O uso de coprocessadores para criptografia, de modo a obter maior desempenho e a possibilidade da troca ou atualização de algoritmos, ou mesmo da mudança de criptografia de chave pública (utilizada para sincronização dos equipamentos e estabelecimento de uma sessão segura para troca de chaves) para chave privada (utilizada para a comunicação efetiva dos dados) em uma mesma aplicação também são propostas em (6) e (22).

A utilização de SPRs na implementação de redes ativas é proposta por Horta em (23). Os nós “inteligentes” da rede seriam capazes de efetuar processamentos dedicados em *hardware* como codificação e decodificação de vídeo ou criptografia, e poderiam ser reconfigurados em tempo real, modificando as características da rede, porém sem interromper seu funcionamento.

Aplicação semelhante é apresentada em (24) com um sistema de distribuição de

conteúdo digital seguro. O gerenciamento de direitos autorais feito tradicionalmente em *software* pode ser suscetível a uma série de ataques, de modo que os autores construíram uma solução em *hardware* utilizando reconfiguração parcial onde, após a conexão ao servidor de conteúdo, o equipamento terminal do usuário é reconfigurado com algoritmos de criptografia e decodificação únicos para aquele par servidor-usuário.

Sistemas de Detecção de Intrusão em Redes (NIDS, — *Network Intrusion Detection Systems*) são usados para prevenir a invasão de redes de comunicação de dados. Esses sistemas podem ser implementados em *software*, ou em *hardware*, e funcionam inspecionando os pacotes recebidos em uma rede à procura de dados "indesejados". O maior fator limitante de um NIDS é a velocidade de reconhecimento das cadeias de dados. Em (25) é mostrada a implementação de um NIDS em um *hardware* baseado em FPGAs. Cada atualização dos padrões de dados a serem reconhecidos resulta em um novo módulo parcial para o reconhecimento de padrões, reconfigurado dinamicamente no componente, de modo a combinar a máxima eficiência do *hardware* com a facilidade de atualização do *software*. O desempenho chega a ser mais de 600 vezes melhor do que o de um NIDS implementado em *software*, quando se aumenta o tamanho da expressão procurada.

O emprego de reconfiguração dinâmica parcial em uma unidade de banda base de Bluetooth é proposto em (26). Os melhores resultados foram obtidos utilizando-se um módulo reconfigurável para a transmissão e outro para a recepção, ocupando a mesma área no *chip* e sendo reconfigurados de maneira seqüencial.

Em (27) é proposta a implementação de rádios definidos por *software* (SDR, — *Software Defined Radios*) em um único FPGA utilizando uma versão embarcada do sistema operacional Linux e reconfiguração parcial em *hardware*, de modo a obter o máximo de flexibilidade para atualização dos padrões existentes e a criação de novos padrões para codificação e decodificação, modulação e demodulação, etc.

O uso da reconfiguração dinâmica para sistemas tolerantes à falhas é proposto em (28). O método é baseado na definição de pequenas áreas de auto-teste que se movimentam através de reconfiguração parcial pelo componente. Essa movimentação é feita de forma a não interromper o funcionamento do sistema, assim, a área a ser analisada tem sua função copiada e transferida para outra área do componente antes de passar pela análise. Se uma falha for detectada, aquela

área é desativada e a função continua a operar na outra área do componente.

Outra utilização da tolerância à falhas se dá em aplicações espaciais que estão sujeitas à interferências causadas por radiação, em especial distúrbios do tipo “evento-único” (SEU, — *Single Event Upsets*) que podem alterar o estado lógico de um elemento de memória estática, seja um *latch*, *flip-flop*, ou célula RAM. Ou seja, pode afetar diretamente a configuração do FPGA fazendo com que ele funcione de maneira inesperada. Embora isso nem sempre aconteça, esses distúrbios precisam ser corrigidos. A possibilidade de ler e escrever a memória de configuração com o dispositivo funcionando permite implementar um sistema capaz de detectar e corrigir esses distúrbios sem interromper a operação do dispositivo e sem precisar reconfigurá-lo por inteiro. Um controlador compara os valores de CRC (*Cyclic Redundancy Check*) das colunas do FPGA em operação com os da memória de programação (os valores originais) e reconfigura aquela coluna caso haja alguma diferença (29).

Em (30) há o relato do lançamento de um satélite australiano com um computador reconfigurável a bordo implementado com FPGAs da Xilinx. Nesse projeto tolerante à falhas é possível a reconfiguração parcial não apenas para resolver problemas causados por um SEU, mas também modificações mais profundas como “contornar” um curto-circuito evitando a perda do satélite. Neste mesmo caso o FPGA é utilizado como componente principal de um computador de alta-performance (HPC, — *High Performance Computing*) capaz de se “especializar” em uma determinada função conforme a demanda. Algumas aplicações sugeridas são: o processamento de imagens para a detecção e monitoramento de desastres naturais como incêndios, por exemplo; provisão de serviços de banda larga via satélite para comunicação multimídia, acesso à Internet e serviços educacionais; sistema de navegação autônomo para o próprio satélite.

1.3 FERRAMENTAS PARA DESENVOLVIMENTO DE SPRs

A metodologia de projeto para SPRs é detalhada no capítulo [3](#), porém para um

entendimento do papel das ferramentas para desenvolvimento de SPRs é preciso ter-se claro que elas atuam principalmente em dois pontos:

- garantir que cada módulo (estático ou reconfigurável) ocupe uma área (física) específica do componente, o que é feito no processo de posicionamento e roteamento (*place & route*) obedecendo às limitações impostas pelo projetista (*constraints*);
- extrair os arquivos de configuração (*bitstreams*) inicial (parte estática e configuração inicial da parte reconfigurável), da parte estática e dos módulos reconfiguráveis.

A função de posicionamento e roteamento é, na maioria das vezes, realizada pelas próprias ferramentas de desenvolvimento dos fabricantes, que oferecem a possibilidade de definir a área do componente que cada módulo do projeto irá ocupar, da mesma forma que se pode definir o terminal a que cada sinal será ligado.

O processo de extração dos *bitstreams* concentra a maioria das ferramentas. É necessário obter-se um *bitstream* inicial de todo o FPGA, contendo a parte fixa e a primeira configuração da parte reconfigurável; depois é necessário obter-se o *bitstream* da parte estática e os *bitstreams* de cada um dos módulos reconfiguráveis. A obtenção desses arquivos de configuração dos módulos estático e parciais é feita através da extração das partes correspondentes do arquivo de configuração inicial. Outras funcionalidades podem ser agregadas como o reposicionamento dos módulos, ou a criação de módulos em um componente menor para a utilização em outro maior.

Em (31), os autores propõem a divisão dessas ferramentas que manipulam os arquivos de configuração em três grupos, o primeiro suportando todo o fluxo de projeto e os dois últimos com atuação na manipulação dos *bitstreams*:

- 1) ferramentas completas que cobrem todo o fluxo de projeto, desde a criação do projeto através de esquemático ou linguagem de descrição de *hardware* (HDL), até a geração dos *bitstreams*. Fazem parte deste grupo, principalmente, as próprias ferramentas dos fabricantes;
- 2) ferramentas que utilizam APIs (*Application Programming Interfaces*) para acessar recursos dos *bitstreams*;

- 3) ferramentas que manipulam diretamente o *bitstream* acessando as posições específicas no arquivo através da solução de equações.

Este último grupo possibilita o desenvolvimento de soluções embarcadas com um controlador de reconfiguração suportando a realocação dos módulos reconfiguráveis quando de seu carregamento. Desta forma é possível ter um sistema com espaços reservados para os módulos reconfiguráveis (*slots*), podendo-se (re)alocar o módulo reconfigurável no espaço que estiver disponível no momento da reconfiguração, alterando-se o *bitstream* em tempo de configuração.

No item [1.3.1](#) há um histórico abrangente do desenvolvimento dessas ferramentas, enquanto que nos itens subseqüentes são analisadas algumas das ferramentas existentes em cada grupo, a saber:

- 1) a versão *Partial Reconfiguration Early Access* do *software* ISE da Xilinx (item [1.3.2](#)), por cobrir todo o fluxo de projeto e abranger todas as famílias de FPGAs da Xilinx que suportam reconfiguração parcial. O *software* PlanAhead, que trabalha em conjunto com essa versão do ISE, também é analisado nesse item;
- 2) a ferramenta JBITS (item [1.3.3](#)), também da Xilinx, por ser a API base para a elaboração das outras ferramentas desse grupo;
- 3) as ferramentas PARBIT (item [1.3.4](#)) e BITPOS (item [1.3.5](#)), por serem as principais representantes desse grupo, visando as famílias Virtex-E e Virtex-II respectivamente.

Finalmente, no item [1.3.6](#), é apresentado um comparativo entre as principais ferramentas apresentadas.

1.3.1 Histórico das ferramentas de desenvolvimento de SPRs

De acordo com (32), as primeiras ferramentas de desenvolvimento de SPRs foram lançadas em 1997, para os componentes da família XC6200 da Xilinx. Essas ferramentas eram todas acadêmicas, a saber: JERC6K, RAGE e ConfDiff. Não havia

nenhuma ferramenta comercial disponível com suporte a reconfiguração parcial. Na Xilinx a JERC6K (*Java Environment for Reconfigurable Computing*) foi modificada para suportar a família XC4000 e renomeada para XBI (*Xilinx Bitstream Interface*).

Em 1999, a Xilinx Inc. lançou o JBits (33), baseada na XBI e incluindo suporte à família Virtex, mas não à Virtex-E. O JBits é um conjunto de classes Java que provê uma interface de programa (API) para acesso e manipulação dos *bitstreams* dos FPGAs, permitindo a manipulação individual de *look-up tables* (LUT), *flip-flops* e roteamento. A construção do JBits permite que ele também sirva de base para a criação de outras aplicações.

Em 2000 a Xilinx adicionou uma extensão ao JBits para suporte à reconfiguração parcial, chamada JRTR (32). No mesmo ano, foi lançada a JBitsDiff, uma ferramenta baseada no JBits capaz de extrair núcleos parciais de um *bitstream* total (34).

Em 2002 foram desenvolvidas três ferramentas para reconfiguração parcial de FPGAs: a JPG e a JBitsCopy, para a família Virtex, e a PARBIT, para a família Virtex-E.

A JPG (35) é baseada na API JBits e manipula os *bitstreams* parciais usando as ferramentas proprietárias da Xilinx, como o ISE Foundation, por exemplo. Esta ferramenta permite usar restrições extraídas das ferramentas de CAD existentes. A JPG trabalha com uma interface gráfica e é usada na fase de pré-síntese dos *bitstreams*. A JBitsCopy (36) também é baseada na API JBits e além de extrair os *bitstreams* parciais é capaz de realocá-los.

O PARBIT (*PARTial Bitfile Transformer*) (37)(23) é uma ferramenta capaz de gerar um *bitstream* parcial copiando apenas os bits correspondentes à área parcialmente reconfigurável do *bitstream* original e gerando cabeçalho e rodapé adequados para o novo *bitstream*. Na sua versão inicial o PARBIT possuía dois modos de operação: *slice*, onde são manipuladas colunas dos *chips*, e *block*, onde uma área retangular pode ser configurada dentro do *chip*.

Em 2003, a Xilinx apresentou o XPART (10) (*Xilinx Partial Reconfiguration Toolkit*, conjunto de ferramentas para reconfiguração parcial), derivado do JBits e basicamente com as mesmas funcionalidades, só que agora escrito em linguagem C. O XPART também provê suporte básico à realocação dos módulos reconfiguráveis e abre a possibilidade de ser implementado em um processador

embarcado. Os métodos fornecidos pelo XPART são muito semelhantes ao PARBIT, mas voltados à família Virtex-II.

Também em 2003 foi proposta uma ferramenta voltada à família Virtex chamada *core unifier* (38). A *core unifier* possui uma GUI e é baseada em equações. Seu funcionamento requer uma configuração estática inicial fixa com um controlador e módulos escravos. Esses módulos reconfiguráveis escravos são projetados normalmente utilizando as ferramentas do fabricante e os *bitstreams* parciais são gerados a partir da junção desses *bitstreams* com o inicial.

Em 2004 o PARBIT foi modificado para suportar um terceiro modo de operação, chamado de BIP (*Bitstream Intellectual Property*), que possibilita que um módulo gerado em um componente da família Virtex-E possa ser utilizado em outro, permitindo uma nova forma de geração de núcleos de propriedade intelectual, baseada em *bitstreams* (BIPs), e também a utilização de um dispositivo menor para o desenvolvimento e teste de um módulo, diminuindo significativamente o tempo de desenvolvimento (39).

Também em 2004 foi lançada a ferramenta BITPOS (31), bastante similar ao PARBIT, porém dedicada à família Virtex-II, sendo capaz de manipular não somente as colunas de blocos lógicos (CLBs, — *Configurable Logic Blocks*), como também as colunas de memória RAM e de multiplicadores existentes nessa família.

Em 2005 foi lançada a ferramenta REPLICA (40), que implementa em *hardware* um subconjunto do PARBIT. A REPLICA — *Relocation per online Configuration Alteration*, ou realocação por alteração da configuração em tempo de execução — foi desenvolvida sobre a plataforma RAPTOR 2000 (41), mas pode ser implementada em qualquer *hardware* baseado em FPGAs. O modelo de sistema adotado pelos autores da ferramenta impõe que os módulos ocupem colunas inteiras do componente, fazendo com que o reposicionamento dos módulos seja unidimensional. A REPLICA atua como um filtro para os *bitstreams* parciais, reposicionando-os dinamicamente durante o carregamento do *bitstream* para a coluna alvo indicada em um registrador de opções. Desta forma, ela não faz a extração do *bitstream* parcial, apenas posiciona dinamicamente um *bitstream* parcial na posição disponível no componente. Para minimizar o *overhead* do *bitstream* parcial, os autores implementaram em *hardware* um gerenciador de configuração.

Nesse mesmo ano a Xilinx lançou uma versão especial de sua ferramenta ISE com suporte à reconfiguração parcial, chamada *Partial Reconfiguration Early Access* (42) e lançou também uma ferramenta chamada PlanAhead (43) para simplificar as etapas de alocação e roteamento, bem como funcionar como um ambiente de desenvolvimento integrado (IDE, — *Integrated Development Environment*) para a reconfiguração parcial, trabalhando em conjunto com a versão mencionada do ISE.

Ainda em 2005, surge a PART-Y (44), um ambiente de desenvolvimento que automatiza a metodologia de projeto de sistemas parcialmente reconfiguráveis empregada pela Xilinx em (3) e que permite realizar a alocação dos módulos. Para as demais funções, no entanto, atua acionando ferramentas da própria Xilinx ou de terceiros.

E para os FPGAs da Atmel foi lançada a ferramenta FIGARO (45), que automatiza os processos de alocação e roteamento e de geração dos *bitstreams* total e parciais para as séries AT40K e AT94K. Essa ferramenta possui uma GUI desenvolvida com o objetivo de minimizar a intervenção do usuário.

Em 2006 surge uma evolução do BITPOS, o pBITPOS (46), que pode ser embarcado em um controlador de reconfiguração, realocando os módulos reconfiguráveis no momento da gravação, de modo similar ao REPLICA.

Um resumo deste histórico pode ser visto na cronologia da Tabela 1.1.

1.3.2 Xilinx ISE - Partial Reconfiguration Early Access

O *software* ISE da Xilinx é um ambiente de desenvolvimento de projetos integrado que suporta todos os FPGAs da Xilinx. Em 2005 foi lançada a versão *Partial Reconfiguration Early Access* (42)(47) que é um conjunto de ferramentas de implementação para reconfiguração parcial instalada sobre a versão normal do *software*, da mesma forma que uma atualização. Essa versão está disponível em uma página restrita a usuários cadastrados no *site* da Xilinx¹. Embora a ferramenta

1 Disponível no endereço <http://www.xilinx.com/support/prealounge/protected/index.htm>

seja essencialmente a mesma, há procedimentos especiais para a criação de SPRs, que são realizados em sua maior parte através de linha de comando, o que não ocorre com projetos convencionais, desenvolvidos totalmente dentro do ambiente gráfico do *software*. As famílias suportadas nesta versão são as Virtex-II, Virtex-II Pro, Virtex-4, Virtex-5 e Spartan-3.

Tabela 1.1 - Cronologia das ferramentas de desenvolvimento

Ano	Ferramenta	Família de FPGAs	Desenvolvedor
1997	JERC6K	XC6200	Terceiros
	RAGE	XC6200	Terceiros
	ConfDiff	XC6200	Terceiros
1999	JBits	Virtex	Xilinx
2000	JRTR	Virtex	Xilinx
	JBitsDiff	Virtex	Terceiros
2002	JPG	Virtex	Terceiros
	JBitsCopy	Virtex	Terceiros
	PARBIT	Virtex-E	Terceiros
2003	XPART	Virtex-II/II Pro	Xilinx
	core unifier	Virtex	Terceiros
2004	PARBIT 2.0	Virtex-E	Terceiros
	BITPOS	Virtex-II/II Pro	Terceiros
2005	REPLICA	Virtex-E	Terceiros
	PART-Y	Virtex-II (?)	Terceiros
	Partial Reconfiguration Early Access ISE	Virtex-II / II Pro / 4 / 5, Spartan-3	Xilinx
	PlanAhead		Xilinx
	FIGARO	Atmel	Atmel/Terceiros
2006	pBITPOS	Virtex-II/II Pro	Terceiros

O *software* PlanAhead (43)(48) atua sobreposto à versão *Partial Reconfiguration Early Access* do ISE, permitindo a implementação de um SPR a partir de um único ambiente de desenvolvimento. O PlanAhead, em si, traz funcionalidades avançadas

para a alocação de área, criação das *busmacros*, verificações de projeto e estimativa do tamanho dos *bitstreams*. Além disso, ele agrega um gerenciador do fluxo de projeto, permitindo que mesmo as funções realizadas pelo ISE sejam acionadas a partir dele de forma transparente. Isto possibilita que o projetista atue em um único ambiente de desenvolvimento e com interface gráfica todo o tempo.

1.3.3 JBits

O JBits (33) é um conjunto de classes Java que fornece uma API para acessar o *bitstream* dos FPGAs das famílias XC4000 e Virtex, sendo que apenas esta última oferece suporte à reconfiguração parcial. É possível controlar individualmente todos os recursos configuráveis do FPGA através do *software*, tais como *look-up tables*, roteamento e os *flip-flops*. Também é possível modificar circuitos existentes ou criar novos. E como pôde ser visto no item [1.3.1](#), um dos objetivos do JBits é servir de base para a construção de novas aplicações de uso geral ou especializadas utilizando sua API.

Entre o JBits e o *bitstream* propriamente dito há uma “*bit-level interface*”, que é o nível responsável por determinar a localização de cada um dos recursos configuráveis do componente no *bitstream* e ligar ou desligar o bit correspondente. Deste modo, o projetista manipula apenas uma camada de abstração superior, sem necessitar de conhecimento do *bitstream*.

O JBits também possui uma biblioteca de núcleos que são uma coleção de classes Java que podem ser utilizadas pelo projetista. Esses núcleos incluem contadores, circuitos aritméticos, etc.

A principal limitação do JBits, apontada por seus autores, é a necessidade do conhecimento detalhado da arquitetura interna do FPGA por parte dos usuários.

1.3.4 PARBIT

A ferramenta PARBIT (*PARTial Bitfile Transformer*) manipula os arquivos de configuração dos FPGAs da família Virtex-E para a geração dos *bitstreams* parciais, que são usados para a reconfiguração parcial. O PARBIT lê o *bitstream* original, relativo a todo o componente, e copia para o *bitstream* parcial apenas os *frames* relativos à área parcialmente reconfigurável, definida pelo projetista; esse *bitstream* parcial tem um tamanho proporcional à área reconfigurável. As áreas reconfiguráveis e os outros parâmetros são armazenados em um arquivo de opções.

O PARBIT é implementado em linguagem C, sendo possível a sua utilização em ambientes Unix/Linux ou Windows (mediante o ambiente Cygwin), e está atualmente na versão 2.0. A partir dessa versão, a ferramenta passou a suportar três modos de operação (37)(23)(39):

- Slice: Neste modo uma área definida por uma ou mais colunas do *bitstream* original é copiada para um *bitstream* parcial, conforme pode ser visto na Figura 4.

Figura 4 - Área reconfigurável definida no modo Slice (23)

- **Block:** Neste modo o projetista define uma área retangular do componente que contém o *bitstream* original. Essa área é definida por linhas e colunas iniciais e finais. A área do *bitstream* parcial no componente “alvo” terá o mesmo tamanho, mas a linha e a coluna finais podem estar em qualquer posição do componente, desde que comporte a área do *bitstream* selecionado. Este modo é ilustrado na Figura 5.

Figura 5 - Área reconfigurável original (esq.) e alvo (dir.) no modo Block (23)

- **BIP:** Este modo é similar ao *block*, porém os componentes original e alvo podem ser de tamanhos diferentes, desde que da mesma família. Isso permite que componentes menores e mais baratos possam ser utilizados para o teste e desenvolvimento de módulos reconfiguráveis e também a comercialização de propriedade intelectual baseada em *bitstreams*. Este modo só está disponível na versão 2.0 e pode ser visualizado na Figura 6.

Para melhor atender às diversas necessidades dos projetistas de SPRs foi desenvolvida neste trabalho uma interface gráfica para o PARBIT, conforme previsto no trabalho original (23).

Figura 6 - Módulo reconfigurável original (esq.) e área destinada no componente alvo (dir.) no modo BIP (39)

1.3.5 BITPOS

O BITPOS (*BITstream POSitioner*), é bastante similar ao PARBIT em todos os aspectos. O princípio de funcionamento é o mesmo, a interface também é por linha de comando e são requeridos dois *bitstreams*, um com a configuração original, chamado de *core repository*, e outro com o alvo, chamado de *target*; há também um arquivo de descrição que guarda as informações da posição da área reconfigurável nos *bitstreams* original e alvo (31).

Há dois modos de funcionamento: *simple* e *merge*. No modo *simple* a área reconfigurável deve envolver colunas inteiras, exatamente como no modo *slice* do PARBIT. No modo *merge* é possível trabalhar com áreas retangulares, como no modo *block* do PARBIT, porém a realocação se dá na horizontal, ou seja, não é possível deslocar verticalmente a área reconfigurável fazendo-a iniciar em outra linha, apenas fazê-la iniciar em outra coluna. A estrutura proposta pelos autores para as aplicações é baseada em *slots*. O BITPOS é capaz de manipular não apenas as colunas de CLBs, como também as colunas de memória RAM e de multiplicadores existentes nos componentes da família Virtex-II.

O pBITPOS (*partial BITstream POSitioner*) lançado posteriormente, tem capacidade de realocar dinamicamente os *bitstreams* para um *slot* livre no momento da gravação, podendo inclusive ser integrado em um sistema de controle de reconfiguração que seja executado “embarcado” (46). Neste caso a funcionalidade é a mesma do REPLICA, porém para a família Virtex-II.

1.3.6 Comparativo entre as ferramentas de desenvolvimento de SPRs

O comparativo apresentado na Tabela 1.2, é inspirado no existente em (31), mas apresentando aqui com diversas modificações, segundo (49). A ferramenta comparada no grupo 1 foi atualizada para a nova metodologia e ferramenta da Xilinx descrita em (42). No grupo 2, as ferramentas construídas com base na API JBits foram agrupadas juntamente com a mesma, substituindo a JPG, e o ano de lançamento considerado foi o da inclusão do suporte à reconfiguração parcial, isto é, o lançamento do JRTR. O XPART também foi incluído no grupo 2. Por fim, a *core unifier* foi considerada apenas no grupo 3, como parece ter sido a intenção principal de seus autores. A comparação aqui também corrige as famílias de FPGAs suportadas por cada ferramenta e mostra características mais pontuais.

Tabela 1.2 - Comparativo entre as ferramentas de desenvolvimento de SPRs

Grupo	1) fluxo total do projeto	2) Acesso ao bitstream via APIs		3) Acesso direto ao bitstream (baseados em equações)		
Ferramenta	ISE PR	Derivados do JBits	XPART	PARBIT	Core unifier	BITPOS
Tipo do Desenvolvedor	Xilinx	Xilinx e terceiros	Xilinx	Tecelios	Tecelios	Tecelios
Famílias de FPGAs	Virtex II/Pro, Virtex-4, Virtex-5 e Spartan-3	Virtex	Virtex-II/Pro	Virtex-E	Virtex	Virtex-II/Pro
Interface com usuário	GUI e linha de comando. Apenas GUI quando usando o PlanAhead	Java API, GUI em alguns derivados	API em C	GUI ou linha de comando.	GUI	Linha de comando
Realocação dos módulos reconfiguráveis	Não	Não	Sim	Sim	Não	Sim
Possibilidade de realocação do módulo reconfigurável para um FPGA diferente do original	Não	Não	Não	Sim	Não	Não
Ferramenta de realocação derivada que possa ser embarcada em um controlador de reconfiguração	Não	Não	Pode ser embarcado	REPLICA	Não	pBITPOS
Manipulação das colunas BlockRAM e de multiplicadores	Sim	Não	Não	Não	Não	Sim
Possibilidade de leitura de configurações a partir dos arquivos da Xilinx	Não-aplicável	Sim (alguns derivados)	?	Não	Não	Não
Ano de lançamento	2005	2000 (JRTR)	2003	2002	2003	2004

Uma análise comparativa é relativa, já que as ferramentas oferecem suporte à famílias específicas de FPGAs e também possuem funcionalidades específicas. É

perfeitamente normal que o projetista tenha como objetivo apenas uma família de FPGAs, seja pelo conjunto de desenvolvimento disponível, seja por restrições de orçamento; nesta situação a ferramenta a ser escolhida deve ser a que suporte a família de componentes objetivada. Por outro lado, o projetista pode necessitar de funcionalidades específicas, disponíveis em apenas uma ou outra ferramenta; neste caso o componente escolhido deve ser o suportado pela ferramenta em questão. A situação ideal, e na verdade a única possível, é que a escolha da família do FPGA combine com as funcionalidades demandadas da ferramenta. Portanto é importante estabelecer as restrições e discutir as vantagens de cada ferramenta para que o projetista afortunado com a possibilidade de escolha, possa fazê-la da melhor forma.

Acerca da família de FPGAs a ser utilizada, no caso da Virtex-E, a escolha deve ser pela ferramenta PARBIT e, no caso das famílias Virtex-4, Virtex-5, ou Spartan-3, a ferramenta escolhida deve ser a versão *Partial Reconfiguration Early Access* do ISE. Apenas as famílias Virtex, Virtex-II e Virtex-II Pro oferecem reais possibilidades de escolha da ferramenta de desenvolvimento de SPRs a ser utilizada.

Uma possibilidade para a realocação dos módulos reconfiguráveis é o uso de uma solução embarcada em um controlador de reconfiguração implementado em *hardware*. Em um sistema que possua espaços dedicados para os módulos reconfiguráveis (*slots*), um único *bitstream* do módulo reconfigurável seria realocado dinamicamente para um *slot* qualquer disponível no momento de seu carregamento. Se este tipo de solução é necessário, ou mesmo considerado, devem ser escolhidas as famílias Virtex-E, utilizando a ferramenta REPLICA, ou Virtex-II/Pro, utilizando as ferramentas pBITPOS ou XPART.

Se for necessário utilizar BIPs a escolha deve recair sobre a família Virtex-E com a ferramenta PARBIT. Deve-se lembrar que as BIPs possibilitam a realocação do módulo parcialmente reconfigurável não apenas para uma posição diferente, mas também para um modelo de componente diferente desde que da mesma família.

As ferramentas dos grupos 2 e 3 ainda dependem das ferramentas do fabricante para gerar os *bitstreams*, mesmo que essas ferramentas não suportem reconfiguração parcial. A partir desse ponto a geração do *bitstream* parcial é autônoma e independe das ferramentas utilizadas nas etapas anteriores do projeto.

Muito embora as ferramentas do grupo 2 possibilitem a execução de grande parte do

fluxo do projeto, na prática as ferramentas são mais restritas, realizando apenas a extração dos *bitstreams* parciais e, eventualmente, o posicionamento dos módulos fixos e reconfiguráveis, ou a leitura direta do arquivo de restrições gerado nas etapas anteriores do projeto.

As ferramentas do grupo 3 trazem outras vantagens tais como:

- extração rápida do *bitstream*;
- pouco poder computacional requerido para processar os *bitstreams*;
- não é necessário executar todo o fluxo de projeto novamente para realocar um *bitstream* (mudar a posição do módulo reconfigurável), como ocorre nas ferramentas do grupo 1.

A facilidade de uso e aprendizado é outro fator importante a ser considerado e também impacta a velocidade de execução do projeto. Esse fator está relacionado com a presença de uma GUI, o que acontece com a PlanAhead, a JPG (derivada do JBits), a *core unifier* e a PARBIT, cuja GUI foi desenvolvida neste trabalho.

2 INTERFACE GRÁFICA DE USUÁRIO PARA O PARBIT

O PARBIT é utilizado atualmente por pesquisadores em trabalhos acadêmicos, especialmente de pós-graduação, em todo o mundo. Esses usuários utilizam regularmente ferramentas de desenvolvimento para FPGAs, seja do próprio fabricante dos dispositivos, seja de terceiros. Essas ferramentas possuem interfaces gráficas de usuário (GUI), enquanto a maioria das ferramentas que manipulam os *bitstreams*, vistas no item [1.3](#), possuem interface do usuário baseadas em linha de comando.

Segundo Foley (50), os objetivos no projeto de uma GUI são: aumentar as velocidades de aprendizagem e de utilização; reduzir a quantidade de erros cometidos pelo usuário; reduzir a possibilidade do usuário cometer erros; ser fácil relembrar sua utilização (após um período sem utilizar a interface) e, finalmente; deve ser atrativa para os potenciais usuários e, eventualmente, compradores.

Para ampliar o leque de usuários do PARBIT, alcançando estudantes de graduação e profissionais projetistas que lidam com FPGAs, o desenvolvimento de uma GUI é de suma importância, além de já haver sido previsto em (23).

Deve-se ressaltar que a GUI proposta é autônoma, executando o PARBIT em modo de linha de comando de forma transparente ao usuário; porém, os usuários que assim o desejarem poderão continuar a utilizar o PARBIT da forma anterior.

Os itens subseqüentes detalham a operação do PARBIT ([2.1](#)), a especificação da GUI ([2.2](#)), sua implementação ([2.3](#)) e uma comparação com o modo de operação antigo, baseado em linha de comando ([2.4](#)).

2.1 OPERAÇÃO DO PARBIT

Atualmente o PARBIT é operado através de linha de comando com a seguinte

sintaxe (23)(39):

parbit opção original parcial [alvo]

Onde:

- original: arquivo do *bitstream* original que contém a área reconfigurável a ser extraída pelo PARBIT;
- parcial: arquivo do *bitstream* parcial gerado pelo PARBIT;
- alvo: arquivo do *bitstream* do componente alvo, necessário para os modos Block e BIP
- opção: Arquivo texto puro, com extensão “opt” contendo todas as opções a serem utilizadas pela ferramenta. Cada linha traz uma opção utilizando o formato “opção:valor”. As opções disponíveis são exibidas na Tabela 2.1.

Um exemplo da utilização do PARBIT seria o comando:

parbit par-block.opt altled.bit alt44-46.bit targ.bit

Onde o conteúdo do arquivo de opções “par-block.opt” seria o seguinte:

```
FPGA:XCV50E
StartColumn:12
EndColumn:14
StartRow:5
EndRow:25
TargetRow:4
TargetColumn:44
Port:SelectMAP
ShutDown:No
Verbose:3
Header:No
```

Os três modos de operação apresentados, bem como todas as opções de configuração, devem ser contemplados na interface gráfica do PARBIT, automatizando e simplificando o processo.

Tabela 2.1 - Opções disponíveis para a utilização do PARBIT (23)(39)

Opção	Valores	Significado
FPGA	XCV50E, XCV100E, XCV200E, XCV300E, XCV400E, XCV405E, XCV600E, XCV812E, XCV1000E, XCV1600E, XCV2000E, XCV2600E, XCV3200E	Modelo do componente
StartColumn	1 ao número máximo de colunas do FPGA escolhido	Coluna inicial da área reconfigurável
EndColumn	1 ao número máximo de colunas do FPGA escolhido	Coluna final da área reconfigurável
StartRow	1 ao número máximo de linhas do FPGA escolhido	Linha inicial da área reconfigurável nos modos Block ou BIP
EndRow	1 ao número máximo de linhas do FPGA escolhido	Linha final da área reconfigurável nos modos Block ou BIP
TargetRow	1 ao número máximo de linhas do FPGA escolhido	Linha da nova posição da área reconfigurável nos modos Block ou BIP
TargetColumn	1 ao número máximo de colunas do FPGA escolhido	Coluna da nova posição da área reconfigurável nos modos Block ou BIP
Port	JTAG SelectMAP (padrão)	Porta de programação para a reconfiguração do FPGAs
Shutdown	No (padrão) Yes (com desligamento)	Define se a reconfiguração será dinâmica (sem desligar) ou se o componente será desligado e reiniciado durante o processo
Side	Right Left	Define uma área reconfigurável igual a metade esquerda ou direita do FPGA calculando os valores de StartColumn e StartRow automaticamente. Válida apenas no modo Slice.
Verbose	0 a 4	Define a quantidade de informação exibida na tela durante a geração do <i>bitstream</i>
Header	Yes (padrão) No	Define se o <i>bitstream</i> parcial será gerado com o cabeçalho original da Xilinx, mais o rodapé do PARBIT (ou apenas o <i>bitstream</i>).

2.2 ESPECIFICAÇÃO DA INTERFACE GRÁFICA

Sobre o desenvolvimento de GUIs, Foley (50) alerta sobre diversos princípios que devem ser levados em conta no projeto, para que atinjam os objetivos mencionados na introdução deste capítulo. Esses princípios implicam no desenvolvimento de uma interface cuja organização seja clara e onde os comandos estejam ordenados de forma lógica e intuitiva, de modo a ser agradável e diminuir o trabalho de memorização necessário por parte do usuário.

A possibilidade do usuário cometer erros também deve ser minimizada no projeto de interfaces; assim, uma opção que não possa ser selecionada em um determinado contexto deve permanecer desabilitada, por exemplo. Os testes de consistência devem ser implementados não apenas neste sentido, mas também no de evitar que se passe algum parâmetro incorreto para a linha de comando, resultando em um erro.

Finalmente Foley (50) discute que o conhecimento dos padrões já estabelecidos em *softwares* similares e a experiência prévia do usuário com esses padrões devem ser levados em consideração; ele acrescenta também que deve ser mantida uma padronização dentro da própria interface. Esse princípio é chamado de consistência.

Considerando-se os pontos abordados e o padrão adotado em praticamente todas as ferramentas de desenvolvimento para FPGAs existentes, a interface gráfica do PARBIT deve adotar o mesmo padrão no que tange a leiaute, estrutura de menus, janelas de diálogo, etc. O leiaute de um desses programas, o Xilinx ISE, pode ser visto na Figura 7.

A área dos fontes mostra os arquivos fontes do projeto que estão, ou podem ser manipulados.

A área de processos exibe em forma de um menu hierárquico em árvore todos os processos, ou tarefas, que são realizados pelo *software*, ordenados pela seqüência em que aparecem no desenvolvimento de um projeto. Cada processo é identificado pelo nome e um ícone, além de uma segunda representação simbólica indicando que aquela etapa foi cumprida (✓), ou apresentou problemas (*), ou ainda possa estar desatualizada (?). Sempre que pertinente, isto é, se não houver dependência

de algum processo anterior, o processo é disparado com um clique duplo sobre seu ícone ou nome. De acordo com o contexto, são executados os processos anteriores àquele ponto também. Assim, ao clicar-se no processo de configuração do FPGA, todos os anteriores são realizados e, ao final, abre-se o *software* de gravação. Por outro lado, ao clicar-se no ícone do relatório de uma operação, abre-se o relatório na área de trabalho.

Figura 7 - Aparência de um *software* de desenvolvimento para FPGAs

A área de trabalho exibe informações pertinentes a cada processo do *software*, seja um diálogo de configuração, seja a exibição de um relatório.

A área de mensagens retorna as mensagens geradas no console de comando.

As barras de menus e ferramentas não foram nomeadas na figura, mas apresentam uma estrutura padrão em relação a outros programas executados em ambiente gráfico.

Por abarcarem uma metodologia de projeto bastante complexa, essas ferramentas de desenvolvimento possuem uma série de assistentes (*wizards*) para agilizar as

configurações. Em um primeiro momento, a GUI para o PARBIT não irá incorporar outros pontos dessa metodologia, não tendo necessidade de assistentes ou inclusão de outras áreas específicas na janela principal. Também não contará com uma área para os arquivos fontes, já que apenas um arquivo será manipulado por vez.

A linguagem escolhida para a implementação da GUI é a Java, pelos seguintes motivos:

- é uma linguagem multi-plataforma e os *softwares* de desenvolvimento comerciais para FPGAs estão disponíveis para os sistemas operacionais Windows, Linux e, eventualmente, outros sistemas variantes do Unix; portanto, é importante que o PARBIT possa ser executado no mesmo ambiente em que se está desenvolvendo o restante do projeto;
- os compiladores e ferramentas de desenvolvimento necessários são gratuitos;
- relativa facilidade para o desenvolvimento da interface e o acionamento de um programa em linha de comando;
- indicação de diversos profissionais de desenvolvimento de *software* e pesquisadores com os quais conversamos sobre o projeto.

2.2.1 Definições da interface gráfica do PARBIT

A janela principal do PARBIT deve ter aspecto similar ao exposto na Figura 7. As áreas de mensagens e dos fontes, no entanto, não são necessárias, apenas as de processos e de trabalho.

A barra de menus contém duas opções, com as respectivas ações e dependências:

- *File:*
 - *New* – abre um novo arquivo de opções (extensão “opt”) em branco; o usuário deve escolher o nome do arquivo e a pasta onde será salvo. Este é o ponto de partida para a utilização da GUI em um novo projeto.

- *Open* – carrega todas as opções a partir de um arquivo da GUI já existente.
- *Close* – fecha o arquivo de opções em uso.
- *Exit* – sai do programa.
- *Help*:
 - *Content* – abre o índice da ajuda da GUI.
 - *About* – exhibe os créditos do programa.

O arquivo de opções deve ser salvo a cada mudança de janela onde haja a configuração de algum de seus parâmetros, bem como antes de efetuar o fechamento do arquivo (opção “*close*” do menu).

Além do arquivo de opções, deve ser gravado um novo arquivo contendo informações adicionais específicas da GUI (extensão “*btp*”).

A área de processos contém um menu hierárquico, cujos itens e respectivas ações e dependências são expostos na Tabela 2.2. A hierarquia é representada na tabela através do espaçamento do texto e, na interface, como um menu em árvore. Todas as configurações e respostas ao usuário são feitas na área de trabalho.

O conteúdo das janelas da área de trabalho variam de acordo com o processo, ou tarefa, selecionado no menu hierárquico da área de processos. Esse conteúdo, bem como as ações relacionadas, são detalhados a seguir para cada processo:

- *Mode and File Selection*:
 - *Operation Mode* – caixas de seleção exclusiva (tipo botão de rádio) para a seleção do modo de funcionamento: *Block*, *Slice* (padrão), *BIP*. Armazena a seleção para a exibição dos parâmetros corretos nos itens seguintes.
 - *File Selection*
 - *Original* – formulário de seleção de arquivo para digitação do nome e caminho de arquivo do *bitstream* original (todos os modos). Armazena a opção para o parâmetro “original” do comando *parbit*.
 - *Partial* – formulário de seleção de arquivo para digitação do nome e caminho de arquivo do *bitstream* parcial (todos os modos). Armazena a opção para o parâmetro “parcial” do comando *parbit*.

- *Target* – formulário de seleção de arquivo para digitação do nome e caminho de arquivo do *bitstream* alvo (habilitado apenas nos modos *Block* e BIP). Armazena a opção para o parâmetro “alvo” do comando parbit.

Tabela 2.2 - Menu hierárquico da área de processos da interface do PARBIT

Item	Ação / Dependência
<i>General Configuration</i>	Expandir o menu, exibindo os subitens abaixo.
<i>Mode and File Selection</i>	Permite a seleção do modo de funcionamento do PARBIT (<i>Slice</i> , <i>Block</i> , BIP). Permite a seleção dos arquivos de <i>bitstreams</i> conforme o modo de funcionamento e também do arquivo de opções.
<i>Configuration Options</i>	Permite a seleção das opções (Vide Tabela 2.1)
<i>Area Configuration</i>	As opções deste menu são habilitadas conforme o modo de funcionamento selecionado anteriormente: - <i>Block</i> e BIP: <i>reconfigurable</i> e <i>target</i> ; - <i>Slice</i> : <i>reconfigurable</i> .
<i>Reconfigurable Area</i>	Permite a seleção gráfica da área reconfigurável original. Exibe uma previsão do tempo de reconfiguração.
<i>Target Area</i>	Permite a seleção gráfica da área reconfigurável do componente alvo. Exibe uma previsão do tempo de reconfiguração.
<i>Bitstream Generation</i>	Opções para finalizar o processo iniciando o comando parbit.
<i>Generate Bitstreams</i>	Este item permite iniciar o processo de geração dos <i>bitstreams</i> acionando o comando parbit. Visualiza as informações geradas pelo comando parbit.
<i>Report</i>	Visualiza o relatório gerado pelo comando parbit.
<i>Options File</i>	Visualiza o arquivo de opções.

- *Configuration Options*:
 - FPGA – lista de seleção de FPGA, com os parâmetros exibidos na Tabela 2.1. Grava a opção selecionada no arquivo de opções utilizando o formato FPGA:seleção. Por exemplo: “FPGA:XCV50E”. Este parâmetro é preenchido automaticamente a partir da leitura do *bitstream* original, conforme descrito adiante.

- *Port* – lista de seleção para a porta de configuração com as opções JTAG (padrão) e SelectMAP. Grava a opção correspondente no arquivo de opções com a sintaxe Port:opção. Por exemplo: “Port:JTAG”.
- *Shutdown* – caixa de seleção exclusiva (botão de rádio) para a seleção entre *Yes* ou *No* (padrão), correspondendo a com e sem desligamento, respectivamente. A opção correspondente é a Shutdown:opção.
- *Side* – esta opção só deve ser habilitada se o modo selecionado for Slice. Duas caixas de verificação para as opções Left e Right. Caso as duas opções sejam acionadas deve haver uma mensagem de erro informando para que seja uma, outra, ou nenhuma opção.
- *Verbose* – lista de seleção com indicações relativas à quantidade de informação a ser gerada para o relatório. Os rótulos são *none*, *less*, *medium*, *high* e *all* (padrão) e a linha a ser gravada no arquivo de opções é verbose:opção, onde opção, nesse caso são os algarismos de 0 a 4 (padrão).
- *Header* – caixas de seleção exclusiva (botões de rádio) para seleção entre *Yes* (padrão) ou *No*, gravando Header:Yes ou Header:No, respectivamente no arquivo de opções.
- *Area Configuration:*
 - *Reconfigurable Area* – permite a seleção gráfica da área reconfigurável, conforme descrito adiante. As opções relacionadas são StartColumn, EndColumn, StartRow e EndRow. Exibe uma estimativa do tempo de reconfiguração e carrega automaticamente o modelo do FPGA a partir da leitura do *bitstream*.
 - *Target Area* – permite a seleção gráfica da área reconfigurável no componente alvo, conforme descrito adiante. As opções relacionadas são TargetRow e TargetColumn. Exibe uma estimativa do tempo de reconfiguração e carrega automaticamente o modelo do FPGA a partir da leitura do *bitstream*.
- *Bitstream Generation:*
 - *Generate Bitstreams* – aciona o comando parbit com as opções definidas

anteriormente, no diretório onde foi salvo o arquivo de opções, para que os *bitstreams* parciais gerados sejam gravados no mesmo local. A saída do comando é exibida na tela e direcionada para um arquivo texto de relatório, também salvo na pasta.

- *Report* – exibe o arquivo texto do relatório resultante do processo anterior.
- *Options File* – exibe o conteúdo do arquivo de opções (opt) atual.

2.2.1.1 Leitura do modelo do FPGA

O formato de arquivo do *bitstream* inclui alguns caracteres especiais, seguidos do nome do arquivo que o originou (extensão “.ncd”), um espaço e o caractere “b”; aparecendo na sequência o modelo do FPGA e depois a data, como se segue:

```
#0#0#0#0 #a #calctop_routed.ncd b #v300efg256 c #2007/11/19
```

Para que se possa identificar o modelo do FPGA, a cadeia de caracteres “.ncd b ” deve ser identificada, para se garantir a leitura da posição correta e, a partir dela, isolar-se o texto delimitado pelos caracteres “v” e “e” seguintes, obtendo-se o modelo do FPGA. Para o parágrafo acima, lê-se “v300e”, que corresponde ao modelo XCV300E e assim sucessivamente.

A informação extraída do *bitstream* original deve ser gravada na opção FPGA e levada para a janela *Reconfigurable Area*, para que o FPGA seja desenhado corretamente. Para o *bitstream* alvo o modelo deve ser carregado na janela *Target Area* para que o FPGA seja desenhado corretamente.

O modelo do FPGA pode continuar sendo preenchido manualmente, por exemplo, para permitir a estimativa do tempo de reconfiguração em função da área reconfigurável.

2.2.1.2 Seleção das áreas dos componentes

As opções *Reconfigurable Area* e *Target Area* demandam a seleção de áreas do componente com base em colunas e linhas. A quantidade de linhas e colunas é dependente do componente selecionado (opção FPGA) e pode ser vista na Tabela 2.3. Essa informação será usada para desenhar na tela uma matriz com a quantidade de linhas e colunas correspondente ao modelo de FPGA selecionado. A área dentro dessa matriz poderá ser selecionada com a digitação dos valores em caixas de texto correspondentes às coordenadas. As colunas de memória RAM devem ser representadas através de um pequeno espaçamento, para orientação visual do projetista. A quantidade de bits por CLB serve de referência para o cálculo do tempo de reconfiguração.

Tabela 2.3 - Dimensões dos FPGAs (51)(52)(53)

Modelo do FPGA	Linhas de CLB	Colunas de CLB	Posição das colunas de memória RAM	Bits por CLB
XCV50E	16	24	0, 6, 18, 24	1.377
XCV100E	20	30	0, 12, 18, 30	1.243
XCV200E	28	42	0, 12, 30, 42	1.098
XCV300E	32	48	0, 12, 36, 48	1.106
XCV400E	40	60	0, 12, 48, 60	1.035
XCV405E	40	60	0, 4, 8, 12, 16, 20, 24, 36, 40, 44, 48, 52, 56, 60	1.125
XCV600E	48	72	0, 12, 24, 48, 60, 72	1.038
XCV812E	56	84	0, 4, 8, 12, 16, 20, 24, 28, 32, 36, 48, 52, 56, 60, 64, 68, 72, 76, 80, 84	1.085
XCV1000E	64	96	0, 12, 24, 72, 84, 96	993
XCV1600E	72	108	0, 12, 24, 36, 72, 84, 96, 108	976
XCV2000E	80	120	0, 12, 24, 36, 84, 96, 108, 120	975
XCV2600E	92	138	0, 12, 24, 36, 96, 108, 120, 138	947
XCV3200E	104	156	0, 12, 24, 36, 108, 120, 138, 156	941

Esse processo gráfico de seleção de área do componente, no caso utilizando-se de um dispositivo apontador, é empregado pela Xilinx, empresa fabricante dos FPGAs,

em seu *software* de desenvolvimento ISE, e mostra-se bastante eficiente. A Figura 8 mostra uma tela do *software* ISE com quatro áreas definidas, no caso do PARBIT é feita a seleção de apenas uma área.

Figura 8 - Exemplo de seleção gráfica de área

Para o processo *Reconfigurable Area*, ao selecionar-se os valores nas caixas StartColumn, EndColumn, StartRow e EndRow (estas duas últimas não habilitadas no modo Slice), a área correspondente deve ficar selecionada no gráfico. Ao concluir-se a seleção, há uma opção de confirmação para que as coordenadas sejam salvas nos parâmetros correspondentes do arquivo de opções.

Para o processo *Target Area* o tamanho da área é obrigatoriamente igual a do processo anterior, podendo ter apenas sua posição no componente modificada. As caixas de texto e, igualmente, os parâmetros salvos no arquivo de opções são os seguintes: TargetColumn e TargetRow.

Em ambos os processos são implementados testes de consistência para assegurar que a área selecionada esteja dentro das dimensões do *chip*.

2.2.1.3 Estimativa do tempo de reconfiguração

As janelas de seleção exibem uma estimativa do tempo de reconfiguração, servindo de orientação ao projetista. A fórmula utilizada para o cálculo é a seguinte:

$$\text{Tempo de reconfiguração} = \frac{A_{rec} \times Q_{bits} \times 1.000}{F_{conf}}$$

Onde:

- Tempo de reconfiguração é dado em milissegundos;
- A_{rec} é a área reconfigurável, calculada por
 $(EndColumn - StartColumn) \times (EndRow - StartRow)$;
- Q_{bits} é a quantidade de bits por CLB, dada na Tabela 2.3, calculada a partir de (52) e (53), tomando-se a quantidade de bits total do *chip*, dividindo-se pela quantidade de CLBs;
- F_{conf} é a frequência de configuração, que varia conforme a porta de programação utilizada e indicada pelo parâmetro *Port*, sendo 4 MHz para JTAG e 50 MHz para SelectMAP.

O resultado obtido é aproximado, uma vez que não considera a real quantidade de *frames* de configuração e nem o cabeçalho do *bitstream*, portanto, se o valor obtido estiver muito próximo às restrições temporais do sistema, o projetista deve fazer contas mais precisas com base no tamanho do *bitstream* gerado.

2.3 IMPLEMENTAÇÃO DA INTERFACE GRÁFICA

A implementação da interface gráfica foi realizada como um programa de estágio acadêmico interno na Universidade Ibirapuera, em moldes similares ao de uma iniciação científica. A codificação foi realizada pelos alunos Thiago Albuquerque Rosa, Leonardo Machado e Daniel Costa Lemes da Silva, do 3º ano do curso de

Ciência da Computação, sob minha orientação e supervisão diretas.

A metodologia utilizada foi a seguinte:

- estudo básico sobre FPGAs e SPRs para familiarização com o tema;
- estudo da especificação da GUI;
- estudo da linguagem Java e da implementação de GUIs;
- seleção de uma ferramenta de desenvolvimento Java adequada;
- implementação da interface (codificação);
- teste da interface;
- realização das correções necessárias;
- teste e validação final;
- elaboração de relatório de estágio.

Foram testadas as ferramentas Eclipse, Abeille e NetBeans. Esta última foi a escolhida. A NetBeans² é da própria Sun Microsystems, criadora da linguagem da Java.

O processo de implementação durou um ano, durante o qual a especificação foi melhorada e, em alguns pontos, modificada ou corrigida. Também foram levantadas novas funcionalidades que poderão ser implementadas em futuras versões.

A Figura 9 mostra o aspecto final da interface, onde é possível visualizar a área de processos à esquerda e a janela de seleção da área reconfigurável à direita.

Os requisitos de sistema para a versão final em ambiente Windows são:

- Java Run Time Enviroment 1.6;
- Cygwin 1.5.x;
- Resolução mínima de 800 x 600.

A versão final, batizada de PARBIT GUI, está disponível na página institucional do PARBIT³.

2 Disponível no endereço <http://www.netbeans.org/>

3 Disponível no endereço <http://www.arl.wustl.edu/projects/fpx/parbit/>

Figura 9 - Aspecto final da interface gráfica do PARBIT

2.4 COMPARAÇÃO DO USO DO PARBIT COM O PARBIT GUI

O PARBIT pode continuar a ser utilizado normalmente através da linha de comando, nesse caso o projetista deve obedecer aos passos contantes na Tabela 2.4, que são contrapostos ao que ocorre quando o PARBIT GUI é utilizado.

Tabela 2.4 - Comparativo da utilização do PARBIT: linha de comando X GUI

Linha de Comando	GUI
Copiar os arquivos dos <i>bitstreams</i> para um diretório de trabalho ou tomar nota do caminho completo deles.	É utilizada uma caixa de seleção de arquivos facilitando a localização dos mesmos e preenchendo o caminho correta e automaticamente.
Criar um arquivo de opções em formato texto puro.	*
Editar o arquivo de opções verificando quais as opções que devem ser incluídas e quais os valores adequados.	Todas as opções e os valores já são fornecidos para a escolha do projetista.
Caso o projetista não se recorde, o modelo do FPGA precisa ser verificado nos arquivos do projeto e, caso utilize o modo BIP, as dimensões do FPGA alvo devem ser anotadas.	Os modelos dos FPGAs são identificados automaticamente a partir do <i>bitstream</i> completo, inclusive no modo BIP.
No posicionamento o projetista deve se assegurar de que as áreas estejam definidas dentro dos limites do FPGA escolhido.	A GUI verifica se as áreas estão dentro dos limites do FPGA escolhido.
Verificar a sintaxe do arquivo de opções e salvá-lo.	*
Digitar o comando “parbit” incluindo os caminhos completos para os <i>bitstreams</i> envolvidos e o arquivo de opções.	Clicar no botão “ <i>generate bitstream</i> ”.
Para poder rever o relatório posteriormente, o projetista deve desviar a saída do comando parbit para um arquivo texto, ou copiar as informações exibidas no interpretador de comandos.	O relatório é salvo automaticamente e pode ser reaberto a qualquer instante.

* Não há intervenção do projetista, já que o arquivo de opções é manipulado de forma transparente.

Nota-se que o uso da GUI traz maior praticidade para o projetista.

3 METODOLOGIAS DE PROJETO DE SPRs

Um SPR possui uma parte estática composta por um ou mais módulos fixos e ao menos uma área reconfigurável onde são programados os módulos parcialmente reconfiguráveis (MPRs). Cada área reconfigurável pode assumir diversas configurações ao longo do funcionamento do sistema, cada qual correspondendo a um MPR. A configuração inicial do SPR a ser gravada no FPGA é composta pela parte estática (módulos fixos) mais a configuração do primeiro MPR (ou do primeiro conjunto de MPRs). Para que o FPGA possa ser programado são necessários o *bitstream* para essa configuração inicial e *bitstreams* parciais adicionais para cada MPR.

Conceitualmente há duas diferenças fundamentais na criação de SPRs utilizando FPGAs em relação a criação de projetos convencionais: a necessidade dos módulos fixos e parcialmente reconfiguráveis ocuparem áreas definidas no componente e a necessidade de se obter os arquivos de configuração (*bitstreams*) correspondentes para que possam ser gravados. Essas diferenças acarretam em mudanças na metodologia de projeto (ou fluxo de projeto) a ser adotada.

Em 2002 foi apresentada a primeira metodologia para projeto de SPRs, baseada em módulos, proposta pela Xilinx na primeira versão de (3). Essa metodologia foi modificada em 2005 para integração com a versão *Partial Reconfiguration Early Access* do *software* ISE (42).

Em conjunto com a metodologia baseada em módulos foi apresentada uma metodologia baseada em diferenças (3) que abrange apenas mudanças muito pequenas no componente, feitas com o próprio editor de FPGA, tais como a função lógica de uma LUT ou o conteúdo de um bloco de RAM. O *bitstream* parcial é gerado baseado na diferença entre os dois projetos. Como essa metodologia não se aplica a SPRs mais complexos, ela não será abordada neste trabalho.

Outras metodologias foram introduzidas juntamente com as ferramentas apresentadas no item [1.3](#). Basicamente, cada nova ferramenta gerou uma nova metodologia adaptando a metodologia original da Xilinx. Essas adaptações concentram-se essencialmente no processo de obtenção dos *bitstreams*. As

ferramentas que manipulam diretamente o *bitstream* possuem metodologias bastante semelhantes, o que ocorre de igual modo entre as ferramentas que se utilizam das APIs JBits e XPART.

Neste capítulo são descritas três metodologias:

- metodologia baseada em módulos (item [3.2](#)), que pode ser aplicada para todas as famílias de FPGAs da Xilinx que suportam reconfiguração parcial;
- metodologia para a versão *Partial Reconfiguration Early Access* do ISE (item [3.3](#)), que pode ser aplicada para as famílias Virtex-II, Virtex-II Pro, Virtex-4, Virtex-5 e Spartan-3;
- metodologia utilizando a ferramenta PARBIT com a interface gráfica implementada (item [3.4](#)), que pode ser aplicada para a família Virtex-E.

Para uma melhor compreensão das metodologias, alguns conceitos básicos são apresentados no próximo item.

3.1 CONCEITOS BÁSICOS

O projeto de qualquer sistema baseado em FPGAs inicia-se com a descrição do circuito, seja através do desenho do esquema elétrico em um CAD, seja através da descrição do mesmo utilizando uma HDL. No caso de um SPR o projeto deve ser dividido em módulos, que podem ser fixos (parte estática) ou reconfiguráveis (MPRs). Os circuitos de cada um desses módulos são descritos individualmente e há também a descrição do projeto principal, que contém todos os módulos de uma configuração do SPR interconectado (em geral a configuração inicial).

Após a etapa da descrição dos circuitos, eles são sintetizados, ou seja, traduzidos para uma lista de ligações (*netlist*). Deste ponto até que se tenha os arquivos de configuração que serão gravados no FPGA há a implementação do projeto, denominação utilizada pelos principais fabricantes para os processos que ocorrem após a descrição e síntese do(s) circuito(s) até a geração dos arquivos de configuração (54). No caso da Xilinx, esse fluxo pode ser observado na Figura 10.

Figura 10 - Fluxo de projeto com FPGAs Xilinx

Os passos principais da implementação de FPGAs da Xilinx são os seguintes (54):

- Tradução (*Translate*) – converte a lista de ligações do projeto, gerada após a síntese em um formato da Xilinx que contém uma descrição lógica dos elementos do projeto. Os diversos arquivos sintetizados são combinados em um único arquivo juntamente com as macros de *hardware* existentes e eventuais circuitos adquiridos de terceiros como propriedades intelectuais em formato EDIF (*Electronic Data Interchange Format*);
- Mapeamento (*Mapping*) – nesta fase o arquivo resultante da tradução é mapeado para componentes internos de um FPGA específico, tais como células lógicas ou de entradas e saídas (E/S);
- Posicionamento e Roteamento (*Place and Route*) – os componentes do arquivo mapeado são fisicamente posicionados no FPGA e é feito o

roteamento (conexão entre eles conforme a lista de ligações).

Na etapa de geração do *bitstream* é gerado o arquivo de configuração que será programado no FPGA.

No caso dos SPRs, as etapas de tradução, mapeamento e posicionamento e roteamento da implementação são as mesmas, podendo variar as configurações ou ferramentas de *software* utilizadas. A geração do *bitstream* é uma etapa mais complexa no caso dos SPRs e por isso é tratada separadamente.

Outro ponto importante no projeto de SPRs é que para que a comunicação entre os módulos fixos e os módulos reconfiguráveis de um SPR seja bem-sucedida, é essencial que esta comunicação seja feita utilizando-se os mesmos sinais, trafegando nas mesmas vias e ocupando as mesmas posições durante as diversas configurações do sistema. Por outro lado, se as linhas roteadas cruzarem os limites dos módulos há a possibilidade de mal-funcionamento, ou mesmo dano, do FPGA durante a reconfiguração (55). As *busmacros* são linhas de comunicação fixas posicionadas nos limites entre os módulos reconfiguráveis e fixos e solucionam ambos os problemas. Elas estão disponíveis como arquivos de macros de *hardware* e podem ser posicionadas na quantidade necessária no FPGA.

As *busmacros* apresentadas neste trabalho foram criadas usando apenas elementos lógicos (CLBs) ao invés de *buffers* de três estados (TBUFs). Cada *busmacro* suporta 8 sinais e ocupa uma área de 4 CLBs (2 x 2). Há uma versão para comunicação da esquerda para a direita e outra da direita para a esquerda, ambas assíncronas. O processo de criação da *busmacro* é detalhado no APÊNDICE A. Versões síncronas, ou mais longas, podem ser criadas com base no processo descrito.

A *busmacro* criada, na versão para comunicação da esquerda para a direita, pode ser vista na Figura 11 em uma visão geral no FPGA e em uma visão detalhada de uma de suas linhas (quatro sinais).

Figura 11 - Visão geral e detalhada da busmacro criada.

3.2 METODOLOGIA DE PROJETO DE SPRs BASEADA EM MÓDULOS

A metodologia de projeto de SPRs baseada em módulos é proposta pela Xilinx em (3) e detalhada em (56) e (57). Ela é baseada na metodologia de projeto modular, apresentada em (54), que permite dividir um projeto em blocos, ou módulos, de modo que o desenvolvimento de cada módulo seja feito de maneira autônoma. A metodologia baseada em módulos impõe como restrição que o MPR ocupe colunas inteiras do FPGA.

O fluxograma da Figura 12 mostra o fluxo de projeto de um SPR segundo essa metodologia, detalhado a seguir conforme (3), (56) e (57).

O projeto se inicia com a descrição e síntese do código do projeto principal em uma HDL. O projeto principal (*top-level*) contém a estrutura geral do projeto e a definição dos módulos na forma de caixas-pretas auto-contidas. As portas de comunicação de cada módulo são definidas como sendo de entrada ou saída e as *busmacros* são instanciadas para a comunicação entre os módulos. O projeto principal é sintetizado e depois cada módulo é descrito utilizando-se uma HDL, ou esquemático, e sintetizado individualmente, porém configurado como projeto principal e sem a opção de adição de *buffers* de E/S, já que esses só devem ser utilizados no projeto principal, onde estão definidos os pinos de E/S.

No posicionamento e definição de restrições são definidas a área e a localização de cada módulo no FPGA, os pinos de E/S e a posição das *busmacros* para a interconexão entre os módulos.

Figura 12 - Metodologia de projeto baseada em módulos

Cada módulo da configuração inicial é então implementado individualmente. Após isso, o projeto principal é implementado correspondendo à configuração inicial completa do SPR. Esta etapa é repetida para cada configuração do SPR, isto é, para cada combinação de MPR com a parte estática do projeto.

Antes da geração do *bitstream* o projeto deve passar por uma inspeção visual utilizando-se o editor de FPGA para garantir que cada módulo tenha se mantido dentro das áreas previamente definidas sem que nenhum sinal tenha sido roteado ultrapassando os limites do módulo. Se isso ocorrer, esses sinais podem ser roteados nova e individualmente no próprio editor de FPGA de modo a se adequar às restrições. Se esse processo não for bem-sucedido devem ser feitas mudanças ou nas restrições de área, ou no posicionamento das *busmacros*, ou no posicionamento dos pinos de E/S (quando isso for possível); após as mudanças deve-se implementar os módulos e o projeto principal novamente, fazendo-se uma nova inspeção e repetindo-se o processo até que todas as restrições sejam atendidas.

A geração do *bitstream* total, correspondente à configuração inicial do projeto, e dos *bitstreams* parciais de cada MPR é feita utilizando-se opções especiais da ferramenta original da Xilinx, a bitgen.

O FPGA pode finalmente ser programado com o *bitstream* da configuração inicial e reconfigurado parcialmente conforme necessário.

A vantagem da metodologia baseada em módulos é servir para todas as famílias de FPGAs da Xilinx que suportam reconfiguração parcial: Virtex, Virtex-E, Virtex-II, Virtex-II Pro, Virtex-4, Virtex-5 e Spartan-3.

As desvantagens são:

- os MPRs devem ocupar colunas inteiras do *chip*;
- não há suporte para realocação dos MPRs, ou seja, os *bitstreams* gerados nessa metodologia servem apenas para a posição original do módulo definida na etapa correspondente;
- são gerados *bitstreams* apenas para o mesmo modelo de FPGA.

3.3 METODOLOGIA UTILIZANDO A VERSÃO PARTIAL RECONFIGURATION EARLY ACCESS DO SOFTWARE ISE

A metodologia de projeto de SPRs baseada em módulos foi modificada pela Xilinx e chamada inicialmente de *Merge Partial Reconfiguration*, apresentada em (58) e (59), porém utilizando ferramentas restritas à própria Xilinx. Em seguida foi lançada a versão *Partial Reconfiguration Early Access* do software ISE e a metodologia correspondente, também chamada de *Early Access Partial Reconfiguration Flow* ou *PR Flow*, apresentada em (42), (47) e (60). A mesma metodologia é apresentada para o software PlanAhead em (42) e (48), com a diferença de que nesse caso sua execução é mais automatizada. O fluxograma da metodologia pode ser visto na Figura 13 e é explicado a seguir.

Figura 13 - Metodologia de projeto de SPRs na versão *Partial Reconfiguration Early Access* do ISE

O projeto se inicia na fase de planejamento e síntese com a divisão do projeto em uma parte estática e uma ou mais partes reconfiguráveis. A comunicação entre os módulos deve ser definida e as *busmacros* necessárias instanciadas. Nessa etapa todos os arquivos HDL são inseridos no projeto e sintetizados: o do projeto principal (*top-level*); o da parte estática (todos os módulos que a compõem) e o de cada um dos MPRs. O HDL do projeto principal contém a estrutura geral do projeto e a descrição dos módulos na forma de caixas-pretas, inclusive das *busmacros*.

No posicionamento e definição de restrições são definidas a posição e a área de cada módulo no FPGA, os pinos de E/S e a posição das *busmacros* para a interconexão entre os módulos.

O projeto principal é então traduzido, juntando-se as *busmacros* ao arquivo HDL. O resultado desta etapa é utilizado tanto na implementação da parte estática como na de cada um dos MPRs.

A parte estática do projeto, isto é, todos os módulos fixos, é implementada, seguida pela implementação dos MPRs, etapa que é repetida tantas vezes quantos forem os MPRs existentes no sistema.

A geração do *bitstream* total inicial e dos *bitstreams* parciais de cada módulo reconfigurável é feita através de um processo de fusão (*merge*) entre a parte estática e as configurações dos MPRs, que é repetido para quantas forem as configurações do SPR. Neste ponto é possível fazer uma inspeção visual do projeto utilizando-se o FPGA editor, porém aqui, os sinais na parte estática do projeto podem passar pela área de um MPR sem a utilização de uma *busmacro*, isto porque durante uma reconfiguração os bits a serem gravados que forem idênticos aos já existentes não são modificados, de modo que os sinais da parte estática não são afetados. Esta característica, porém, não é válida para os FPGAs da família Spartan-3.

Com os *bitstreams* gerados, o FPGA pode finalmente ser programado.

As vantagens desta metodologia são:

- a área reconfigurável não precisa ocupar toda a coluna do componente, podendo ser qualquer área retangular;
- os sinais na parte estática do projeto podem passar pela área de um MPR

sem a utilização de uma *busmacro*;

Essas vantagens, porém, não são válidas para a família Spartan-3, uma vez que nesses dispositivos todos os bits de configuração da coluna sendo programada são levados a “0” para depois assumirem o valor presente no *bitstream*, o que é conhecido como *glitch*. Nota-se que isso inviabiliza as duas vantagens, já que um módulo fixo posicionado na mesma coluna de um MPR seria apagado e reconfigurado juntamente com este e um sinal que cruzasse o MPR seria desligado e religado durante o processo de reconfiguração. Os FPGAs das diversas famílias Virtex não possuem esse problema.

As desvantagens são:

- não há suporte para realocação dos MPRs, ou seja, os *bitstreams* gerados nessa metodologia servem apenas para a posição original do módulo definida na etapa correspondente;
- são gerados *bitstreams* apenas para o mesmo modelo de FPGA.
- não há suporte para as famílias Virtex e Virtex-E.

3.4 METODOLOGIA DE PROJETO UTILIZANDO O PARBIT

O PARBIT é utilizado para geração dos *bitstreams* parciais e sua metodologia de projeto é apresentada em (37), (21) e (39), sendo detalhada e modificada aqui em função da GUI desenvolvida.

Utilizando-se o PARBIT, o primeiro *bitstream* parcial pode ser extraído diretamente do *bitstream* total da configuração inicial. Para a obtenção dos demais *bitstreams* parciais há dois métodos, conforme ilustrado na Figura 14. O primeiro utiliza um projeto contendo apenas o MPR e atende aos requisitos de projeto de BIPs. Este método pode ser usado sozinho quando projetando apenas o núcleo BIP. O segundo método usa uma versão completa do projeto com os módulos fixos e o novo MPR, do mesmo modo que para a configuração inicial. Este método permite a simulação e teste de cada MPR ou, em outras palavras, de cada configuração do

SPR.

O projeto da configuração inicial deve seguir a metodologia baseada em módulos apresentada no item [3.2](#) e na Figura 12.

O primeiro passo é a descrição e síntese de um projeto principal contendo todos os módulos do sistema na forma de “caixas-pretas”, incluindo as *busmacros*. Depois disso cada módulo é descrito e sintetizado individualmente como projetos principais e sem *buffers* de E/S. Durante o posicionamento cada módulo é alocado em uma área específica do FPGA e os pinos de E/S são definidos. Em seguida as *busmacros* são posicionadas durante a definição das restrições. Elas devem ser posicionadas uma coluna antes do limite com o MPR. Se elas forem posicionadas em um local não permitido os passos seguintes falharão. O posicionamento de cada *busmacro* inclui linhas para sua declaração, posicionamento e fixação no arquivo de restrições. A fixação deve conferir com a base da área da *busmacro*. O projeto principal é então combinado com as macros de *hardware* em um único arquivo durante a tradução.

Figura 14 - Fluxos possíveis para a geração dos *bitstreams* parciais.

Após isso, cada módulo (parciais e fixos) da configuração inicial é implementado individualmente, seguindo-se da implementação da configuração inicial completa, possibilitando a geração do *bitstream* correspondente. Antes da geração do *bitstream* o projeto deve ser inspecionado visualmente utilizando-se o editor de FPGA para garantir que cada módulo esteja confinado à área para ele reservada. Se

necessário, os sinais que tenham ultrapassado os limites dos módulos podem ser roteados individualmente no próprio editor de FPGA. Se esse processo não for bem-sucedido devem ser feitas mudanças ou nas restrições de área, ou no posicionamento das *busmacros*, ou no posicionamento dos pinos de E/S (quando isso for possível). Nesses casos, após as mudanças deve-se implementar os módulos e o projeto principal novamente e proceder uma nova inspeção.

O *bitstream* total para a configuração inicial é gerado utilizando-se o bitgen, a ferramenta original da Xilinx. Mais detalhes acerca desses passos iniciais podem ser obtidos em (3), (21) e (39).

A geração do *bitstream* parcial é feita usando a GUI do PARBIT. A Figura 15 mostra o fluxo de projeto que está disponível na seqüência correta no menu em árvore, como na área de processos do ISE Project Navigator da Xilinx. Cada passo do fluxo deve ser seguido fazendo-se as configurações necessárias nas janelas correspondentes, que se abrem no lado direito da tela, e salvando-se as modificações. As opções mais comuns foram pré-definidas como padrão.

Figura 15 - Captura de tela da GUI do PARBIT

O fluxo de projeto para a geração de cada *bitstream* parcial envolve três passos:

configuração geral (*general configuration*); configuração de área (*area configuration*) e geração do *bitstream* (*bitstream generation*), onde além da geração do *bitstream* é possível ver o relatório.

A configuração geral permite a seleção do modo de operação, dos arquivos dos *bitstreams* e das opções de configuração para a geração dos *bitstreams* parciais. As opções de configuração abrangem o modelo do FPGA, que é reconhecido automaticamente pela interface a partir do *bitstream* da configuração inicial, porta de configuração, desligamento do dispositivo após configuração, seleção de lado para o modo de operação *slice*, quantidade de informação fornecida no relatório e inclusão de cabeçalho no *bitstream* gerado.

A configuração de área permite a configuração e visualização da área do MPR para os FPGAs original e alvo. Essa última opção só está disponível nos modos de operação *block* e *target* e é mostrada na Figura 16. Para o modo de operação *slice* está disponível apenas a seleção de colunas. As janelas de seleção de área também mostram o modelo e as dimensões de cada FPGA, bem como uma estimativa para o tempo de reconfiguração de acordo com a área e a porta de configuração selecionadas.

Figura 16 - Captura da tela de seleção de área nos modos Block e Target

A geração do *bitstream* é a fase final onde o *bitstream* parcial é gerado e o relatório é exibido. O arquivo de opções utilizado também pode ser visualizado.

Neste ponto tem-se um *bitstream* total correspondente à configuração inicial e um *bitstream* parcial correspondente ao primeiro módulo reconfigurável. Como mencionado anteriormente, há dois caminhos possíveis para a obtenção dos outros *bitstreams* parciais, como se segue.

Os outros módulos reconfiguráveis podem ser gerados dentro do SPR de modo a serem testados e simulados. Nesta situação todo o fluxo para a configuração inicial deve ser seguido novamente, porém é possível simplificar o processo usando os mesmos nomes de entidades e de arquivos para os módulos reconfiguráveis, mas deixando os arquivos em diretórios diferentes (60). Procedendo desta maneira é possível descrever, sintetizar e implementar novamente apenas o módulo reconfigurável e continuar com o restante do fluxo, como mostra a Figura 17.

Figura 17 - Fluxo de projeto simplificado para os MPRs adicionais

Após cumprir as etapas de implementação da configuração do SPR e da inspeção visual, o *bitstream* parcial pode ser gerado usando a interface gráfica do PARBIT seguindo os mesmos passos descritos anteriormente.

Se não for necessário efetuar nenhuma simulação, ou quando projetando um núcleo BIP, o fluxo de projeto é similar ao utilizado para a configuração inicial, mas o mais importante, é totalmente independente do mesmo. Um projeto principal é criado para o MPR contendo apenas sua “caixa-preta” e o instanciamento das *busmacros*. Os sinais que vêm do e vão para o MPR devem ser conectados diretamente aos pinos de E/S. O fluxo de projeto a ser seguido pode ser visto na Figura 18. Ele é o mesmo fluxo da configuração inicial adaptado para o novo projeto principal com seu único MPR. No caso de um núcleo BIP, o mesmo pode ser gerado em um modelo de FPGA diferente (maior ou menor) do utilizado no projeto alvo, isto é, no projeto onde o BIP será inserido.

Figura 18 - Fluxo de projeto para apenas um MPR

Como há apenas um módulo, o fluxo é completado muito mais rápido do que para a configuração inicial ou outra configuração completa do SPR. Na etapa de posicionamento pode-se aproveitar o mesmo arquivo de restrições da configuração inicial eliminando-se os módulos não utilizados. Um *bitstream* total é então gerado contendo o MPR e as linhas roteadas entre as *busmacros* e os pinos de E/S. Caso o projeto utilize o mesmo modelo de FPGA, o *bitstream* parcial é gerado utilizando-se a interface gráfica do PARBIT como detalhado anteriormente.

Caso o projeto utilize um modelo de FPGA diferente do utilizado no projeto alvo, há alguns cuidados especiais na geração do *bitstream*. O autor do BIP deve enviar o *bitstream* da configuração completa do MPR para o projetista que utilizará o BIP. Esse projetista usará o modo BIP do PARBIT GUI para gerar o *bitstream* parcial do BIP para o projeto alvo. Para isso, o projetista deve gerar um *bitstream* de seu projeto que será utilizado como alvo no PARBIT GUI. Esse *bitstream* alvo deve conter as *busmacros* e a área reservada para o BIP. Essa área deve estar preenchida com uma lógica fictícia ou com um outro módulo que eventualmente seja utilizado na mesma área; isto é necessário para assegurar o roteamento dos sinais de *clock* para o módulo. Após selecionar adequadamente a área no FPGA alvo o processo de geração do *bitstream* é análogo ao descrito anteriormente.

Finalmente, o FPGA pode ser programado com o *bitstream* da configuração inicial sucedido pelas reconfigurações parciais.

A metodologia utilizando o PARBIT apresenta as seguintes vantagens em relação às metodologias anteriores:

- permite que os *bitstreams* parciais sejam gerados de maneira independente do *software* do fabricante do FPGA, no caso, sem o ISE. Isto permite que seja utilizado um microcomputador que nem sequer possua o ISE instalado e, eventualmente, que tenha configurações mais modestas do que as requeridas para executá-lo. Permite também que em um *bitstream* parcial seja gerado para diversas posições diferentes, em um sistema que utilize *slots* por exemplo, a partir de um mesmo *bitstream* completo;
- a obtenção do *bitstream* parcial depende apenas de outro *bitstream* correspondente a uma configuração completa do sistema. Isto, aliado ao descrito no item anterior, permite a distribuição de BIPs;
- permite não apenas a realocação, mas também a criação de um *bitstream* parcial para outro modelo de FPGA dentro da família Virtex-E, o que é utilizado tanto na distribuição de BIPs, como no teste de módulos individuais

de um projeto maior em um FPGA menor;

- em relação à metodologia baseada em módulos permite a realocação do MPR;
- com a GUI implementada todo o processo de extração dos *bitstreams* é feito em ambiente gráfico, com menor intervenção por parte do usuário e minimização dos erros graças aos testes de consistência incluídos;
- a GUI fornece uma estimativa do tempo de reconfiguração, permitindo que seja utilizada pelo projetista ainda na etapa de posicionamento para que se defina uma área que atenda às restrições de temporização do projeto.

Como desvantagens em relação às metodologias anteriores têm-se que:

- o PARBIT está limitado à família Virtex-E;
- as etapas iniciais do projeto do SPR permanecem dependentes das ferramentas do fabricante.

4 IMPLEMENTAÇÃO DE SPR

Para a validação das metodologias expostas no capítulo anterior foi implementado o projeto de um SPR completo. O projeto escolhido é uma versão modificada de um antigo projeto de referência divulgado pela Xilinx na versão 4.2 da ferramenta ISE. Trata-se de uma calculadora com dois módulos fixos, um acionador de mostrador de cristal líquido (LCD) e um módulo de botões, com um MPR central. Esse MPR alterna entre um módulo de captura, que é a configuração inicial, um módulo somador e outro subtrator, de acordo com o comando do usuário através dos botões. Outros MPRs poderiam ser projetados, tais como multiplicadores e divisores. A comunicação do MPR com os módulos fixos é feita de modo serial, assim há a presença de vários conversores série-paralelo e paralelo-série no projeto; isto foi feito para se diminuir a quantidade de *busmacros* no projeto. A estrutura do SPR pode ser vista na Figura 19.

Figura 19 - Diagrama básico do SPR com a distribuição temporal dos MPRs.

O SPR foi implementado utilizando duas metodologias:

- para a versão *Partial Reconfiguration Early Access* do *software* ISE;
- utilizando o PARBIT com a interface gráfica implementada.

O projeto não foi implementado com a metodologia baseada em módulos, já que ela praticamente não é mais utilizada, visto que serve de base para a metodologia do PARBIT e que a Xilinx a substituiu pela versão *Partial Reconfiguration Early Access*.

A descrição do projeto principal foi atualizada e recebeu a descrição das *busmacros*. Com exceção dos nomes das *busmacros*, foram utilizados os mesmos arquivos VHDL do projeto principal em ambas as implementações. Os arquivos de descrição dos módulos, com exceção de algumas modificações pontuais necessárias para adaptação à família de FPGA escolhida, também foram os mesmos.

Os resultados da implementação são expostos a seguir e o detalhamento pode ser encontrado nos apêndices.

4.1 IMPLEMENTAÇÃO COM A METODOLOGIA PARA A VERSÃO PARTIAL RECONFIGURATION EARLY ACCESS DO ISE

O modelo de FPGA escolhido para a implementação do projeto foi o XC3S200, da família Spartan-3. Essa família foi escolhida por ser de baixo custo quando comparada com as demais. Ela possui as limitações de não suportar MPRs que ocupem áreas retangulares, o que não afeta o SPR proposto, e de não permitir que os sinais da parte estática passem pela área reconfigurável. A metodologia apresentada no item [3.3](#) foi seguida passo-a-passo, conforme detalhado no APÊNDICE B.

Utilizando-se o *software* ISE foram feitas as modificações necessárias na descrição do projeto principal, inclusive a inserção das *busmacros* e sua síntese, procedimento seguido para todos os demais módulos, tanto fixos como reconfiguráveis.

O posicionamento dos módulos foi feito utilizando-se o *floorplanner* do ISE e o arquivo de restrições foi editado manualmente, acrescentando-se as informações relativas à reconfiguração parcial e ao posicionamento das *busmacros*.

O projeto principal foi traduzido para incorporar as *busmacros* e a parte estática implementada (módulos acionador de LCD e de botões). O roteamento da parte estática invadiu a área do MPR e foi corrigido manualmente no editor do FPGA. Após isso os MPRs de captura, somador e subtrator foram implementados.

Finalmente os *bitstreams* são gerados a partir de um processo de fusão para cada MPR, correspondendo a cada configuração do SPR e, portanto, realizado três vezes neste exemplo. Esse processo utiliza as ferramentas *pr_verifydesign* e *pr_assemble*. Em cada processo são gerados o *bitstream* parcial do MPR e o *bitstream* apenas da parte estática, o que é feito pelo *pr_verifydesign*, e também o *bitstream* da configuração completa com a parte estática mais o MPR, gerado pelo *pr_assemble*. Cada versão pode ser visualizada no editor de FPGA. Tomados os resultados para o MPR somador, a Figura 20 mostra o roteamento apenas do MPR, a Figura 21 mostra a parte estática e a Figura 22 mostra todo o SPR utilizando o MPR somador.

Figura 20 - Resultado do MPR somador

Figura 21 - Resultado da parte estática

Figura 22: Resultado da configuração completa com o MPR somador

4.2 IMPLEMENTAÇÃO COM A METODOLOGIA DO PARBIT

O modelo de FPGA escolhido para a implementação do projeto foi o XCV300E, da família Virtex-E. A metodologia apresentada no item [3.4](#) foi seguida passo-a-passo, conforme detalhado no APÊNDICE C.

A configuração inicial utiliza o MPR de captura. O projeto principal foi descrito, incluindo a inserção das *busmacros*, com o auxílio do *software* ISE, onde também é feita a síntese. O procedimento é repetido para os demais módulos da configuração

inicial. De fato, os MPRs que não fazem parte da configuração inicial também podem ser sintetizados nesta etapa, o que elimina a necessidade de executar novamente o ISE mais adiante.

Os módulos foram posicionados através do *floorplanner* do ISE e o arquivo de restrições editado manualmente, com o acréscimo das informações referentes à reconfiguração parcial e às posições das *busmacros*.

Os módulos fixos e o MPR de captura foram implementados individualmente, seguidos da implementação da configuração inicial do SPR. O resultado dessa etapa pode ser inspecionado visualmente através do editor de FPGA, conforme mostra a Figura 23. O único sinal que cruza os módulos é o de *clock*, o que é permitido, uma vez que o roteamento de *clock* é independente da configuração dos CLBs.

Figura 23 - Projeto roteado para a configuração inicial (MPR de captura).

O *bitstream* completo da configuração inicial é gerado utilizando a ferramenta bitgen que faz parte do ISE.

O PARBIT GUI foi usado conforme descrito anteriormente para extrair o *bitstream* parcial do MPR de captura.

O *bitstream* do MPR somador foi extraído utilizando-se o fluxo simplificado exibido na Figura 17. O diretório dos arquivos do módulo de captura foi copiado e o arquivo do somador renomeado para captura, de modo a acelerar o processo de implementação. O resultado pode ser visto na Figura 24.

Figura 24 - Projeto roteado com o MPR somador

O *bitstream* completo desta configuração foi extraído com o bitgen e o parcial utilizando-se o PARBIT GUI como descrito anteriormente. No PARBIT GUI foi necessário apenas abrir uma cópia do arquivo do projeto utilizado para o MPR de captura, mudar a seleção de arquivos e gerar o *bitstream* parcial do MPR somador.

Para o MPR subtrator foi utilizado o fluxo de projeto para apenas um MPR, descrito na Figura 18, para poder exemplificá-lo. Neste caso o subtrator é tratado como se fosse um núcleo BIP. Um novo projeto principal foi descrito contendo apenas as *busmacros*, o módulo subtrator e os sinais a ele relacionados. Esse projeto principal foi sintetizado, bem como o módulo subtrator. Na etapa de posicionamento e definição de restrições os sinais das *busmacros* foram ligados diretamente a pinos de E/S e foram utilizadas as mesmas posições para as *busmacros* e para o MPR. O

módulo foi implementado, seguido da implementação da configuração completa. A inspeção visual desta etapa no editor de FPGA mostrou que algumas linhas foram roteadas invadindo a área do MPR, o que não é permitido. Essas linhas foram desroteadas e roteadas novamente para que o projeto atendesse à restrição de área. O resultado final pode ser visto na Figura 25.

Figura 25 - MPR subtrator ladeado pelas conexões entre as *busmacros* e os pinos de E/S

O PARBIT GUI foi utilizado novamente para a geração do *bitstream* parcial deste último módulo.

Neste ponto, têm-se todos os *bitstreams* que compõem o SPR.

5 CONCLUSÕES

Neste trabalho foi realizado um estudo sobre sistemas parcialmente reconfiguráveis, compreendendo sua evolução histórica, um panorama das principais aplicações e uma análise das ferramentas disponíveis para a manipulação dos *bitstreams*, necessária à utilização em SPRs. As ferramentas foram classificadas e descritas as principais. Foi feita uma análise comparativa de modo a fornecer aos projetistas um guia para a escolha da ferramenta de desenvolvimento apropriada para o projeto de um SPR. Essa comparação também mostrou que há uma carência de ferramentas alternativas ao ISE versão *Partial Reconfigurable Early Access* que suportem as famílias Virtex-4, Virtex-5 e a família Spartan-3 de FPGAs de baixo custo; especialmente ferramentas baseadas em equações. Suporte a realocação de módulos e interfaces gráficas para as ferramentas também são itens que precisam ser mais disseminados.

A descrição dessas ferramentas e a análise comparativa produzidas foram apresentados no artigo “*Development Tools for Partial Reconfigurable Systems*” (49), na SPL2008 4th *Southern Conference on Programmable Logic*⁴, promovida pela *Universidad Autónoma de Madrid* e pela IEEE Circuits and Systems Society (CAS).

Das ferramentas apresentadas, foi dada especial atenção ao PARBIT, para o qual foi especificada e implementada uma interface gráfica de usuário (GUI) utilizando linguagem Java. O PARBIT GUI possibilita uma utilização mais intuitiva e amigável do *software*, facilitando o aprendizado e diminuindo o trabalho do projetista. Estas vantagens devem contribuir para o aumento da base de usuários, além de auxiliar os atuais. Uma vez que foi desenvolvida de modo independente do programa original, o PARBIT GUI possibilita a incorporação de novos recursos que venham a ser desenvolvidos para o PARBIT, bem como a incorporação de outras etapas da metodologia de projeto não contemplados atualmente.

As metodologias de projeto para SPRs propostas pela Xilinx foram apresentadas em sua versão original (baseada em módulos) e atual (*Partial Reconfiguration Early*

4 Informações sobre o evento disponíveis em <http://www.splconf.org>.

Access). A metodologia de projeto de SPRs utilizando o PARBIT foi atualizada e consolidada. As vantagens e desvantagens de cada metodologia também foram apresentadas.

Para a viabilização da metodologia utilizando o PARBIT foram criadas *busmacros* para a família de FPGAs Virtex-E, baseadas em blocos lógicos ao invés de *buffers tri-state* e compatíveis com as versões mais recentes das ferramentas da Xilinx.

A metodologia mais recente da Xilinx e a metodologia do PARBIT GUI foram validadas com a implementação de um SPR de uma calculadora, com dois módulos fixos e um módulo parcialmente reconfigurável que assume três diferentes configurações. No caso do PARBIT GUI a nova metodologia mostrou melhoras em relação à anterior diminuindo significativamente o trabalho manual necessário.

O projeto dos SPRs e das *busmacros* foram detalhados passo-a-passo nos apêndices, na forma de roteiros de projeto.

As metodologias apresentadas e o PARBIT GUI serão aplicados no próprio programa de pós-graduação em engenharia elétrica da escola, na disciplina PSI5003 – Projeto de Sistemas Reconfiguráveis.

Este trabalho também deve contribuir de forma geral para o ensino da reconfiguração parcial em cursos de graduação e pós-graduação e, o mais importante, sua aplicação na indústria.

De maneira pontual, as principais contribuições deste trabalho são:

- análise das ferramentas de desenvolvimento para SPRs existentes, com os resultados publicados em artigo;
- elaboração do PARBIT GUI, interface gráfica de usuário para o PARBIT GUI;
- descrição das metodologias de projeto de SPRs propostas pela Xilinx, empresa fabricante dos FPGAs;
- aperfeiçoamento da metodologia de projeto de SPRs do PARBIT utilizando o PARBIT GUI;
- criação de *busmacros* baseadas em lógica para os FPGAs Virtex-E;
- validação das metodologias através do projeto de um SPR exemplo;
- descrição detalhada do projeto das *busmacros* e do SPR exemplo;

- utilização desta dissertação com finalidades didáticas na disciplina PSI-5003, em cursos de graduação e pós-graduação e para profissionais já formados, disseminando o uso de reconfiguração parcial.

5.1 TRABALHOS FUTUROS

O leque de trabalhos que se abre a partir deste abrange três aspectos:

- modificações no PARBIT com as conseqüentes modificações no PARBIT GUI para suporte a novos componentes ou recursos na extração dos *bitstreams*;
- melhorias no PARBIT GUI com novas funcionalidades e automação de mais etapas das metodologias de desenvolvimento;
- exploração de pontos mais específicos no projeto de SPRs partindo-se da aplicação das metodologias e ferramentas aqui apresentadas.

O PARBIT e conseqüentemente o PARBIT GUI podem ser modificados para suportar as famílias Virtex, Virtex-II, Virtex-4, Virtex-5 e Spartan-3. Este trabalho requer parceria com a Xilinx, empresa fabricante dos FPGAs, da mesma forma que já ocorreu para a versão atual. Outra modificação que pode ser feita é o suporte à manipulação das colunas de memória RAM (BRAM) dos FPGAs da família Virtex-E, à semelhança do que foi desenvolvido no BITPOS, para a família Virtex-II (46).

As funcionalidades do PARBIT GUI podem ser expandidas incorporando a seleção da área reconfigurável a partir da leitura do arquivo de restrições do usuário (“ucf”). De modo similar, a GUI poderia ser utilizada para fazer as modificações necessárias à reconfiguração parcial no mesmo arquivo, com a inserção das *busmacros* e a modificação dos parâmetros necessários. Outros pontos da metodologia de projeto poderiam ser automatizados, como o acionamento dos comandos a partir da interface gráfica, de forma transparente ao usuário, e com a geração de um relatório de projeto. Isso permitiria também a automação do projeto para outras famílias além da Virtex-E, acionando a versão *Partial Reconfiguration Early Access* do ISE. Outras pequenas modificações podem ser incorporadas à interface do PARBIT GUI para

melhorar seu uso.

Finalmente, uma vez que as ferramentas e metodologias foram aqui exploradas, é possível a exploração de outras questões relacionadas à reconfiguração parcial, tais como o gerenciamento da reconfiguração, o emprego de diversos MPRs simultâneos em um sistema e a implementação prática de um SPR para uma aplicação real.

REFERÊNCIAS

1. HUTCHINGS, B. L.; WIRTHLIN, M.J. Implementation approaches for reconfigurable logic applications. In: FIELD-PROGRAMMABLE LOGIC AND APPLICATIONS (FPL), 5., 1995, Oxford. **Proceedings**. Berlin: Springer-Verlag, 1995. p. 419-428.
2. COMPTON, K.; HAUCK, S. Reconfigurable computing: a survey of systems and software. **ACM Computing Surveys**, v. 34, n. 2, p. 171-210, 2002.
3. XILINX INC. **Two flows for partial reconfiguration: module based or difference based**. XAPP290 (v1.2). 2004.
4. ATMEL CORPORATION. **Implementing cache logic with FPGAs**. Application Note. 1999.
5. BECKER, J.; HUEBNER, M.; ULLMANN, M. Power estimation and power measurement of Xilinx Virtex FPGAs: trade-offs and limitations. In: SYMPOSIUM ON INTEGRATED CIRCUITS AND SYSTEMS DESIGN (SBCCI), 16., 2003, São Paulo. **Proceedings**. IEEE Computer Society, 2003. p. 283-288.
6. TOMMISKA, M. **Applications of reprogrammability in algorithm acceleration**. Tese (doutorado) - Department of Electrical and Communications Engineering, Helsinki University of Technology, Espoo, 2005.
7. ALGOTRONIX LTD. **Building on experience**. 2003. Disponível em <<http://www.algotronix.com/company/Background.html>>. Acesso em: jun.2005.
8. NATIONAL SEMICONDUCTOR. **Versatile and efficient cell-to-local bus interface in a configurable logic array**. United States Patent 5298805. 12.mar. 1994.
9. XILINX INC. Introducing the XC6200 FPGA architecture. **XCell Journal**, n.18, p. 22-23, 1995.
10. BLODGET, B.; JAMES-ROXBY, P.; KELLER, E.; McMILLAN, S.; SUNDARARAJAN, P. A self-reconfiguring platform. In: FIELD-PROGRAMMABLE LOGIC AND APPLICATIONS (FPL), 13., 2003, Lisboa. **Proceedings**. Berlin: Springer, 2003. p. 565-574.
11. TSENG, C. W. **Spartan-3 advanced configuration architecture**. XAPP452 (v1.0). Xilinx Inc. 3.dez.2004.
12. PLESSL, C.; ENZLER, R.; WALDER, H.; BEUTEL, J.; PLATZNER, M.; THIELE, L.; TRÖSTER, G. The case for reconfigurable hardware in wearable computing.

Personal and Ubiquitous Computing, Londres, v. 7, n. 5, p. 299-308, 2003.

13. SNOWDEN, T. VisiCom uses Xilinx FPGAs for a image processing reconfigurable module. **XCell Journal**, n. 36, p.51-53, 2000.
14. WESTFELDT, W. Silicon Xpresso and Internet reconfigurable logic. **XCell Journal**, n. 32, p. 10-12, 1999.
15. PARNELL, K. Reconfigurable vehicles are just around the corner. **XCell Journal**, n. 41, 2001.
16. CLAUS, C.; ZEPPENFELD, J.; MÜLLER, F.; STECHELE, W. Using partial-run-time reconfigurable hardware to accelerate video processing in driver assistance system. In: DESIGN, AUTOMATION AND TEST IN EUROPE CONFERENCE AND EXPOSITION (DATE), 2007, Nice. **Proceedings**. ACM, 2007. p. 498-503.
17. NIYONKURU, A.; ZEIDLER, H. C. Designing a runtime reconfigurable processor for general purpose applications. In: INTERNATIONAL PARALLEL AND DISTRIBUTED PROCESSING SYMPOSIUM (IPDPS), 18., 2004, Santa Fe. **Proceedings**. IEEE Computer Society, 2004. CD-ROM.
18. VAN DEN BRANDEN, G.; BRAECKMAN, G.; TOUHAFI, A.; DIRKX, E. A dynamically reconfigurable virtual instrument. In: INTERNATIONAL SYMPOSIUM ON HIGH-PERFORMANCE COMPUTER ARCHITECTURE, 12., 2006, Austin. **Proceedings**. 2006. Disponível em: <<http://www.cc.gatech.edu/wisa06/>>. Acesso em: 29.fev.2008.
19. DANNE, K.; BOBDA, C.; KALTE, H. Run-time exchange of mechatronic controllers using partial hardware reconfiguration. In: FIELD-PROGRAMMABLE LOGIC AND APPLICATIONS (FPL), 13., 2003, Lisboa. **Proceedings**. Berlin: Springer, 2003. p. 272-281.
20. TAYLOR, D. E.; TURNER, J. S.; LOCKWOOD, J. W.; HORTA, E. L. Dynamic hardware plugins (DHP): exploiting reconfigurable hardware for high-performance programmable routers. **Computer Networks**, v. 38, n. 3, p. 295-310, 2002.
21. HORTA, E.L.; LOCKWOOD, J.W.; TAYLOR, D.E.; PARLOUR, D. Dynamic hardware plugins in an FPGA with partial run-time reconfiguration. In: DESIGN AUTOMATION CONFERENCE (DAC), 39., 2002, New Orleans. **Proceedings**. 2002.. p. 343-348.
22. LAGGER, A.; UPEGUI, A.; SANCHEZ, E.; GONZALEZ, I. Self-reconfigurable pervasive platform for cryptographic application. In: FIELD-PROGRAMMABLE LOGIC AND APPLICATIONS (FPL), 16., 2006, Madrid. **Proceedings**. IEEE, 2006. p. 1-4.
23. HORTA, E. L. **Comutador ATM reconfigurável dinamicamente através de**

- hardware**. 2002. 117 p. Tese (doutorado) - Escola Politécnica, Universidade de São Paulo. São Paulo, 2002.
24. HORI, Y.; YOKOYAMA, H.; TODA, K. Secure content distribution system based on run-time partial hardware reconfiguration. In: FIELD-PROGRAMMABLE LOGIC AND APPLICATIONS (FPL), 16., 2006, Madrid. **Proceedings**. IEEE, 2006. p. 1-4.
 25. HUTCHINGS, B. L.; FRANKLIN, R.; CARVER, D. Assisting network intrusion detection with reconfigurable hardware. In: IEEE SYMPOSIUM ON FIELD-PROGRAMMABLE CUSTOM COMPUTING MACHINES (FCCM), 10., 2002, Napa. **Proceedings**. IEEE Computer Society, 2002. p. 111-120.
 26. ESQUIAGOLA, J.; OZARI, G.; TERUYA, M.; STRUM, M.; CHAU, W. A dynamically reconfigurable bluetooth base band unit. In: INTERNATIONAL CONFERENCE ON FIELD PROGRAMMABLE LOGIC AND APPLICATIONS (FPL), 15., 2005, Tampere. **Proceedings**. IEEE, 2005. p. 148-152.
 27. RYSER, P. **Software defined radio with reconfigurable hardware and software**. 2005. In: Embedded Systems Conference, 2005, San Francisco. Disponível em: http://www.xilinx.com/products/design_resources/proc_central/resource/ETP-442Paper.pdf. Acesso em: 2.mar.2008.
 28. EMMERT, J.; STROUD, C.; SKAGGS, B.; ABRAMOVICI, M. Dynamic fault tolerance in FPGAs via partial reconfiguration. In: IEEE SYMPOSIUM ON FIELD-PROGRAMMABLE CUSTOM COMPUTING MACHINES (FCCM), 8., 2000, Napa. **Proceedings**. IEEE Computer Society, 2000. p. 165-174.
 29. CARMICHAEL, C.; CAFFREY, M.; SALAZAR, A. **Correcting single-event upsets through Virtex partial configuration**. XAPP216 (v1.0). Xilinx Inc. 1.jun.2000.
 30. BERGSMAN, P. Xilinx FPGA Blasted into Orbit. **XCell Journal**, n. 46, 2003.
 31. KRASTEVA, Y.E.; JIMENO, A. B.; DE LA TORRE, E.; RIESGO, T. Straight method for reallocation of complex cores by dynamic reconfiguration in Virtex II FPGAs. In: IEEE INTERNATIONAL WORKSHOP ON RAPID SYSTEM PROTOTYPING (RSP), 16., 2005, Montreal. **Proceedings**. IEEE Computer Society, 2005. p. 165-174.
 32. MCMILLAN, S.; GUCCIONE, S. A. **Partial run-time reconfiguration using JRTR**. In: INTERNATIONAL CONFERENCE ON FIELD PROGRAMMABLE LOGIC AND APPLICATIONS (FPL), 10., 2000, Villach. **Proceedings**. Springer, 2000. p. 352-360.
 33. GUCCIONE, S.; LEVI, D.; SUNDARARAJAN, P. **JBits: Java based interface for reconfigurable computing**. In: ANNUAL MILITARY AND AEROSPACE

APPLICATIONS OF PROGRAMMABLE DEVICES AND TECHNOLOGIES CONFERENCE (MAPLD), 2., 1999, Laurel. Disponível em: <<http://www.ittc.ku.edu/~dandrews/690/files/JBitsMAPPLD.pdf>>. Acesso em: 2.mar.2008.

34. JAMES-ROXBY, P.; GUCCIONE, S. A. Automated extraction of run-time parameterisable cores from programmable device configurations. In: IEEE SYMPOSIUM ON FIELD-PROGRAMMABLE CUSTOM COMPUTING MACHINES (FCCM), 8., 2000, Napa. **Proceedings**. IEEE Computer Society, 2000. p. 153-164.
35. RAGHAVAN, A. K.; SUTTON, P. JPG – A partial bitstream generation tool to support partial reconfiguration in Virtex FPGAs. In: INTERNATIONAL PARALLEL AND DISTRIBUTED PROCESSING SYMPOSIUM (IPDPS), 16., 2002, Fort Lauderdale. **Proceedings**. IEEE Computer Society, 2002. CD-ROM.
36. DYER, M.; PLESSL, C.; PLATZNER, M. Partially reconfigurable cores for Xilinx Virtex. In: INTERNATIONAL CONFERENCE ON FIELD PROGRAMMABLE LOGIC AND APPLICATIONS (FPL), 12., 2002, Montpellier. **Proceedings**. Springer, 2002. p. 292-301.
37. HORTA, E. L.; LOCKWOOD, J. W. **PARBIT: A tool to transform bitfiles to implement partial reconfiguration of field programmable gate arrays (FPGAs)**. Washington University, 2001. 37 p. (Relatório Técnico WUCS-01-13).
38. MESQUITA, D.; MORAES, F.; PALMA, J.; MÖLLER, L.; CALAZANS, N. **Remote and partial reconfiguration of FPGAs: tools and trends**. In: INTERNATIONAL PARALLEL AND DISTRIBUTED PROCESSING SYMPOSIUM (IPDPS), 17., 2003, Nice. **Proceedings**. IEEE Computer Society, 2003. CD-ROM.
39. HORTA, E. L.; LOCKWOOD, J. W. **Automated method to generate bitstream intellectual property cores for Virtex FPGAs**. In: INTERNATIONAL CONFERENCE ON FIELD PROGRAMMABLE LOGIC AND APPLICATIONS (FPL), 14., 2004, Leuven. **Proceedings**. Springer, 2004. p. 975-979.
40. KALTE, H.; LEE, G.; PORRMANN, M.; RÜCKERT, U. **REPLICA: a bitstream manipulation filter for module relocation in partial reconfigurable systems**. In: INTERNATIONAL PARALLEL AND DISTRIBUTED PROCESSING SYMPOSIUM (IPDPS), 19., 2005, Denver. **Proceedings**. IEEE Computer Society, 2005. CD-ROM.
41. KALTE, H.; PORRMANN, M.; RÜCKERT, U. A prototyping platform for dynamically reconfigurable system on chip designs. In: IEEE WORKSHOP HETEROGENEOUS RECONFIGURABLE SYSTEMS ON CHIP (SoC), 2002, Hamburg. **Proceedings**. Disponível em: <<http://www.raptor2000.de/en/sct/extern/raptor2000/publications/>>. Acesso em: 2.mar.2008.

42. LYSAGHT, P. BLODGET, B.; MASON, J.; YOUNG, J.; BRIDGFORD, B. Enhanced architectures, design methodologies and CAD tools for dynamic reconfiguration of Xilinx FPGAs. In: INTERNATIONAL CONFERENCE ON FIELD PROGRAMMABLE LOGIC AND APPLICATIONS (FPL), 16., 2006, Madrid. **Proceedings**. IEEE, 2006. p. 1-6.
43. DORAIRAJ, N.; SHIFLET, E.; GOOSMAN, M. PlanAhead software as a platform for partial reconfiguration. **Xcell Journal**, n. 55, p. 68-71, 2005.
44. DITTMANN, F.; RETTBERG, A.; SCHULTE, F. A Y-Chart based tool for reconfigurable system design. In: INTERNATIONAL CONFERENCE ON ARCHITECTURE OF COMPUTING SYSTEMS (ARCS), 18., 2005, Innsbruck. **Proceedings**. VDE Verlag, 2005. p. 67-73.
45. NASI, K.; KAROUBALIS, T.; DANEK, M.; POHL, Z. FIGARO - An automatic tool flow for designs with dynamic reconfiguration. In: INTERNATIONAL CONFERENCE ON FIELD PROGRAMMABLE LOGIC AND APPLICATIONS (FPL), 15., 2005, Tampere. **Proceedings**. IEEE, 2005. p. 590-593.
46. KRASTEVA, Y. E.; DE LA TORRE, E.; RIESGO, T.; JOLY, D. VIRTEX II FPGA bitstream manipulation: application to reconfiguration control systems. In: INTERNATIONAL CONFERENCE ON FIELD PROGRAMMABLE LOGIC AND APPLICATIONS (FPL), 16., 2006, Madrid. **Proceedings**. IEEE, 2006. p. 1-4.
47. XILINX INC. **Early access partial reconfiguration user guide for ISE 8.1.01i**. UG208 (v1.1). 2006.
48. JACKSON, B. **Partial reconfiguration design with PlanAhead 9.2**. Version 1.1. Xilinx Inc. 2007.
49. VALIANTE FILHO, F.; HORTA, E. L. Development tools for partial reconfigurable systems. In: SOUTHERN CONFERENCE ON PROGRAMMABLE LOGIC (SPL), 4., 2008, Bariloche. **Proceedings**.
50. FOLEY, J.D.; VAN DAM, A.; FEINER, S.K.; HUGHES, J. F. **Computer graphics principles and practice – second edition in C**. Addison-Wesley, 1996. p. 391-470.
51. XILINX INC. **Virtex series configuration architecture**. XAPP151 (v1.7). 2004.
52. XILINX INC. **Virtex™-E 1.8 V field programmable gate arrays**. DS-022. 2002.
53. XILINX INC. **Virtex™-E 1.8 V extended memory field programmable gate arrays**. DS-025. 2002.
54. XILINX INC. **Development system reference guide**. 2005.

55. HUEBNER, M.; BECKER, T.; BECKER, J. Real-time LUT-based network topologies for dynamic and partial FPGA self-reconfiguration. In: SYMPOSIUM ON INTEGRATED CIRCUITS AND SYSTEM DESIGN (SBCCI), 17., 2004, Ipojuca. **Proceedings**. New York: ACM, 2004. p. 28-32.
56. BOBDA, C.; AHMADINIA, A.; RAJESHAM, K.; MAJER, M. Partial configuration design and implementation challenges on Xilinx Virtex FPGAs. In: INTERNATIONAL CONFERENCE ON ARCHITECTURE OF COMPUTING SYSTEMS (ARCS), 18., 2005, Innsbruck. **Proceedings**. VDE Verlag, 2005. p. 61-66.
57. BLODGET, B.; BOBDA, C.; HUEBNER, M. NIYONKURU, A. Partial and dynamically reconfiguration of Xilinx Virtex-II FPGAs. In: INTERNATIONAL CONFERENCE ON FIELD PROGRAMMABLE LOGIC AND APPLICATIONS (FPL), 14., 2004, Leuven. **Proceedings**. Springer, 2004. p. 801-810.
58. SEDCOLE, P.; BLODGET, B.; ANDERSON, J.; LYSAGHT, P.; BECKER, T. Modular partial reconfiguration in Virtex FPGAs. In: INTERNATIONAL CONFERENCE ON FIELD PROGRAMMABLE LOGIC AND APPLICATIONS (FPL), 15., 2005, Tampere. **Proceedings**. IEEE, 2005. p. 211-216.
59. SEDCOLE, P.; BLODGET, B.; BECKER, T.; ANDERSON, J.; LYSAGHT, P. Modular dynamic reconfiguration in Virtex FPGAs. **IEE Proceedings - Computers and Digital Techniques**, v. 153, issue 3, p. 157-164, 2.maio.2006.
60. BRIDGFORD, B. **Using partial reconfiguration to time-share device resources in Virtex-II and Virtex-II Pro**. XAPP255. Xilinx Inc. 25.maio.2005.

APÊNDICE A – ROTEIRO DE CRIAÇÃO DE UMA BUSMACRO

As *busmacros* são criadas utilizando-se o editor de FPGA, integrante do *software* ISE da Xilinx. A seguir é descrita passo-a-passo a criação de uma *busmacro* para a família Virtex-E de FPGAs, do tipo assíncrona, curta e da esquerda para a direita “L2R”.

- 1) Iniciar o editor de FPGA (FPGA Editor).
- 2) Criar um novo arquivo para a *busmacro* selecionando “file – new” no menu. Na janela que se abre deve-se marcar “hard macro”, fornecer um nome para o arquivo (BM_Virtex-E_L2R, por exemplo), escolher um componente qualquer da família de FPGAs para o qual será utilizada a *busmacro* (XCV50E-6-CS144, por exemplo) e confirmar a criação do arquivo pressionando “OK”.
- 3) Selecionar um conjunto de 8 CLBs (uma área de 2 colunas e 2 linhas) e pressionar o botão “add”. Vide Fig. A1.

Fig. A1 - Seleção dos CLBs

- 4) Selecionar um pino de saída (finais X ou Y) de um CLB à esquerda e um pino de entrada do CLB correspondente à direita (finais “Gx” ou “Fx”, no caso “G4” e “F4”) utilizando o *mouse* e a tecla “Ctrl”. Pressionar o botão “add” e uma nova rede (*net*) é inserida. O nome da rede pode ser editado. Confirmar pressionando o botão “Ok”.
- 5) Repetir o passo 4 até a adição de todas as redes. A Fig. A2 mostra a metade superior da *busmacro* após o processo.

Fig. A2 - Metade superior da *busmacro* após a adição das redes

- 6) Selecionar um CLB para edição das funções com um duplo clique sobre ele e ativar o modo de edição pressionando o botão . Marcar as opções LUT nas caixas que permitem a seleção de função. Marcar as opções “F” e “G” nos multiplexadores conectados às saídas e ativar os *buffers* de saída correspondentes.
- 7) Habilitar a edição das funções com um clique no ícone . Para os CLBs de entrada da *busmacro* (lado esquerdo), preencher as equações de “F” e “G” com “A1”. Para os CLBs de saída da *busmacro* (lado direito), preencher as equações de “F” e “G” com a entrada onde está conectada a rede, “A4” no caso. Salvar e sair com um clique sobre o ícone . A Fig. A3 mostra o aspecto do CLB editado.
- 8) Repetir os passos 6 e 7 até que todos os CLBs tenham sido editados.
- 9) Para um CLB de entrada, selecionar um pino de entrada que está conectado à LUT (finais F1 e G1, no caso) e depois as opções “edit – add hard macro external pin” no menu. Na janela que se abre digitar o nome do pino (input0 no caso) em “external name”. Executar o mesmo procedimento para o outro pino do CLB, conectado à outra LUT, mudando o nome do pino (input1, etc.).

Fig. A3 - CLB editado.

- 10) Repetir o passo 9 para todos os CLBs de entrada.
- 11) Para um CLB de saída, seleccionar o pino de saída (finais Y e X, no caso), tomando o cuidado de verificar na outra ponta da rede nele conectado qual o nome dado ao pino de entrada. Seleccionar as opções “edit – add hard macro external pin” no menu. Na janela que se abre digitar em “external name” o nome do pino de saída correspondente ao de entrada (output0 no caso). Executar o mesmo procedimento para o outro pino do CLB, conectado à outra LUT, mudando o nome do pino (output1, etc.).
- 12) Repetir o passo 11 para todos os CLBs de saída.
- 13) Executar o DRC (Design Rule Check) para verificar possíveis erros, através do menu “tools – DRC – run”.
- 14) Salvar e sair do editor de FPGA (opção “file – save” e “file – exit” do menu).

Para que a *busmacro* possa ser utilizada é necessário criar sua descrição em HDL, que será inserida no arquivo do projeto principal do SPR. A descrição da *busmacro* criada em VHDL é a seguinte:

```
component BM_Virtex-E_L2R is
  port (
    input0 : in std_logic;
    input1 : in std_logic;
    input2 : in std_logic;
    input3 : in std_logic;
    input4 : in std_logic;
    input5 : in std_logic;
    input6 : in std_logic;
```

```

        input7 : in std_logic;
        output0 : out std_logic
        output1 : out std_logic;
        output2 : out std_logic;
        output3 : out std_logic;
        output4 : out std_logic;
        output5 : out std_logic;
        output6 : out std_logic;
        output7 : out std_logic
    );
end component;

```

Essa descrição do componente *busmacro* deve ser inserida dentro da arquitetura da entidade, no VHDL do projeto principal. Uma descrição para cada tipo de *busmacro* utilizada (esquerda para direita e direita para esquerda, por exemplo).

Em um SPR podem ser utilizadas diversas *busmacros*, de acordo com a necessidade de interconexão dos módulos. Cada uma dessas *busmacros* necessita ter todas as suas conexões declaradas, isto é, tanto os sinais como os pinos deixados em aberto, conforme o exemplo:

```

BM_add_to_pbt : BM_Virtex-E_L2R

    port map (
        input0  => Sync_S2P_sig,
        output0 => Sync_S2P_PB_thruBM,
        input1  => LED_RM_sig,
        output1 => LED_RM_sig_thruBM,
        input2  => '1',
        output2 => open,
        input3  => '1',
        output3 => open,
        input4  => '1',
        output4 => open,
        input5  => '1',
        output5 => open,
        input6  => '1',
        output6 => open,
        input7  => '1',
        output7 => open
    );

```

Posteriormente, essas *busmacros* devem ser posicionadas no FPGA editando-se manualmente o arquivo de restrições do projeto (“ucf”). A *busmacro* criada deve ser declarada, posicionada e travada em um CLB uma coluna antes do limite entre os módulos, como se segue:

```

INST "BM_add_to_pbt" AREA_GROUP = "AG_BM_add_to_pbt" ;
AREA_GROUP "AG_BM_add_to_pbt" RANGE = CLB_R6C36:CLB_R7C37 ;
INST "BM_add_to_pbt" LOC = "CLB_R6C36.S1" ;

```

APÊNDICE B – ROTEIRO DE PROJETO DA METODOLOGIA PARA A VERSÃO PARTIAL RECONFIGURATION EARLY ACCES DO ISE

Este roteiro demonstra o projeto de um SPR de acordo com a metodologia para a versão *partial reconfiguration early access* do *software* ISE, tomando como exemplo o SPR de uma calculadora para a família Spartan-3 de FPGAs de baixo custo. Essa calculadora possui dois módulos fixos, um acionador de LCD e um módulo de botões, com um MPR central que possui três configurações: um módulo de captura (inicial), um somador e um subtrator. Divulgado inicialmente pela Xilinx como um projeto de referência na versão 4.2 do ISE, os arquivos HDL de descrição dos módulos foram modificados para compatibilização com a atual versão do ISE, ou com o FPGA utilizado.

A implementação do SPR da calculadora começa com a criação de uma estrutura de diretórios adequada, conforme se segue:

Os arquivos das *busmacros* e das descrições dos circuitos (se houverem) devem ser copiados para os respectivos diretórios.

A partir deste ponto pode-se acompanhar passo-a-passo a metodologia, conforme ilustra a Fig. B1.

Fig. B1 - Metodologia de projeto com a indicação dos itens correspondentes

B1 PLANEJAMENTO E SÍNTESE

O *software* ISE deve ser aberto e um novo projeto criado, no diretório indicado. Deve ser selecionada a família Spartan-3 e o dispositivo XC3S200. As outras configurações podem ser as padrões, notando-se que o tipo de arquivo do projeto principal (*top-level*) deve ser HDL:

- menu “file – new project”.

O arquivo do projeto principal (calctop.vhd) deve ser inserido no circuito:

- menu “project – add source”.

Este arquivo não deve conter lógica, apenas uma descrição do projeto, com o instanciamento de cada módulo na forma de “caixas-pretas”, inclusive as *busmacros*, a definição dos sinais entre os módulos e dos sinais e pinos de E/S. No

caso, o arquivo foi modificado a partir do projeto inicial, com algumas modificações para compatibilização do código e com o acréscimo das *busmacros*.

Devem ser declarados todos os tipos de *busmacros* utilizados como componentes, conforme o exemplo:

```
component busmacro_xc3s_l2r_async_narrow is
  port (
    input0 : in std_logic;
    input1 : in std_logic;
    input2 : in std_logic;
    input3 : in std_logic;
    input4 : in std_logic;
    input5 : in std_logic;
    input6 : in std_logic;
    input7 : in std_logic;
    output0 : out std_logic;
    output1 : out std_logic;
    output2 : out std_logic;
    output3 : out std_logic;
    output4 : out std_logic;
    output5 : out std_logic;
    output6 : out std_logic;
    output7 : out std_logic
  );
end component;
```

Devem ser descritas todas as *busmacros* utilizadas no projeto, conforme o exemplo:

```
BM_pbt_to_add: busmacro_xc3s_r2l_async_narrow

port map (
  input0  => MiddlePB_Rising,
  output0 => MiddlePB_Rising_thruBM,
  input1  => LeftPB_Rising,
  output1 => LeftPB_Rising_thruBM,
  input2  => Reset_reconfig_module,
  output2 => Reset_reconfig_module_from_RM,
  input3  => DownPB_Rising,
  output3 => DownPB_Rising_thruBM,
  input4  => RightPB_Rising,
  output4 => RightPB_Rising_thruBM,
  input5  => TopPB_Rising,
  output5 => TopPB_Rising_thruBM,
  input6  => '1',
  output6 => open,
  input7  => '1',
  output7 => open
);
```

Após as modificações necessárias o arquivo deve ser definido como projeto-principal (*top-level*) e sintetizado:

- seleccionar o arquivo na janela sources e utilizar o menu “source – set as top module”;

- duplo clique em “Synthesize – XST” na janela de processos;
 - Toda síntese gera um arquivo com mesmo nome e extensão “.ngc” (calctop.ngc) no diretório do projeto.

Se a síntese não for bem-sucedida deve-se investigar os erros e saná-los, repetindo a síntese em seguida. É comum que haja advertências (*warnings*) resultantes do projeto. Elas devem ser analisadas pelo projetista para avaliar se são todas normais ou se há algum problema que deva ser corrigido. Após a síntese, o arquivo do projeto principal deve ser removido do projeto:

- selecionar o arquivo na janela “sources” e pressionar a tecla “delete”.

Os próximos passos serão repetidos para cada um dos módulos componentes do SPR.

Deve-se inserir no projeto o arquivo do módulo acionador de LCD (lcd_driver.vhd) e defini-lo como projeto principal:

- menu “project – add source”;
- selecionar o arquivo na janela sources e utilizar o menu “source – set as top module”.

Antes da síntese, deve-se remover a opção para adição de *buffers* de E/S, já que essas conexões foram declaradas no arquivo do projeto-principal e aqui há apenas o conteúdo da “caixa-preta” lá declarada:

- selecionar a opção “Synthesize – XST” na janela de processos e o menu “source – properties”. Selecionar “Xilinx Specific Options” e desmarcar a caixa “Add I/O Buffers”.

Sintetizar o arquivo do módulo:

- duplo clique em “Synthesize – XST” na janela de processos.

Se houver algum erro, ele deve ser corrigido e o processo repetido. Do contrário, o arquivo deve ser removido do projeto:

- selecionar o arquivo na janela “sources” e pressionar a tecla “delete”.

Esses passos devem ser repetidos para os módulos de botões (pushbutton.vhd), captura (capture.vhd), somador (adder.vhd) e subtrator (subtractor.vhd). Estes dois últimos módulos possuem um sub-módulo conversor de binário para decimal, cujo

arquivo de descrição (binary_to_bcd.vhd) também deve ser inserido no projeto antes da síntese, ficando abaixo dos módulos subtrator e somador.

Com todos os módulos sintetizados o ISE pode ser fechado.

B2 POSICIONAMENTO E DEFINIÇÃO DE RESTRIÇÕES

Para esta etapa, deve-se copiar o arquivo “ngc” do projeto principal (calctop.ngc) para o diretório “top” e fazer uma tradução inicial do arquivo:

- comando “ngdbuild -modular initial calctop.ngc”;
 - este comando gera um arquivo com mesmo nome e extensão “ngd” (calctop.ngd).

Para o posicionamento dos módulos é utilizado o floorplanner:

- comando “floorplanner”;
- abrir o arquivo “\top\calctop.ngd” gerado previamente.

Cada módulo deve ter sua área atribuída no componente, a saber o módulo acionador do LCD no lado esquerdo, o MPR de captura no centro e o módulo de botões no lado direito. O posicionamento é feito da seguinte forma:

- selecionar o módulo correspondente na janela “design hierarchy” e o menu “floorplan – assign area constraint”;
- marcar a área com o *mouse* no desenho do FPGA.

Os pinos de E/S apresentados em “calctop Primitives” também devem ser atribuídos a um pino de E/S do FPGA arrastando-se o símbolo da janela de hierarquia para um pino do desenho do FPGA. As primitivas que ficam no lado esquerdo são “LCD_Databus[7:0]”, “E”, “RW”, “RS”, “Reset_lcd_driver”. As primitivas que ficam no lado direito são “TopPushButton”, “RightPushButton”, “MiddlePushButton”, “LeftPushButton”, “DownPushButton”, “LED_A_out”, “LED_B_out”, “LED_C_out”, “LED_D_out”, “Reset_pushbutton” e “Reset_reconfig_module”. A primitiva “U1” é atribuída a um “BUFGMUX” (BUFGMUX5) e a “Clock_in” ao pino correspondente (A9). O floorplanner deve ser fechado, confirmando-se o salvamento dos arquivos.

O floorplanner gera um arquivo de restrições de usuário, com extensão “ucf”

(calctop.ucf). Este arquivo deve ser modificado manualmente:

- abrir o arquivo “calctop.ucf” em um editor de texto puro;
- modificar as linhas referentes aos módulos da parte fixa, alterando os nomes dos “area_group” para “AG_base1” e “AG_base2”:

```
INST "DisplayLCD" AREA_GROUP = "AG_base1" ;
AREA_GROUP "AG_base1" RANGE = SLICE_X0Y47:SLICE_X5Y0 ;
AREA_GROUP "AG_base1" RANGE = RAMB16_X0Y0:RAMB16_X0Y5 ;
AREA_GROUP "AG_base1" RANGE = MULT18X18_X0Y0:MULT18X18_X0Y5 ;

[...]

INST "Switch_LEDs" AREA_GROUP = "AG_base2" ;
AREA_GROUP "AG_base2" RANGE = SLICE_X34Y47:SLICE_X39Y0 ;
AREA_GROUP "AG_base2" RANGE = RAMB16_X1Y0:RAMB16_X1Y5 ;
AREA_GROUP "AG_base2" RANGE = MULT18X18_X1Y0:MULT18X18_X1Y5 ;
```

- modificar as linhas referentes ao MPR alterando o “area_group” e inserindo a declaração de que o módulo é reconfigurável:

```
INST "Reconfigurable_Module" AREA_GROUP = "AG_PRM" ;
AREA_GROUP "AG_PRM" RANGE = SLICE_X6Y47:SLICE_X33Y0 ;
AREA_GROUP "AG_PRM" MODE = RECONFIG ;
```

- acrescentar as *busmacros*:

```
INST "BM_lcd_to_add" LOC = "SLICE_X4Y30" ;
INST "BM_add_to_lcd" LOC = "SLICE_X4Y10" ;
INST "BM_add_to_pbt" LOC = "SLICE_X32Y30" ;
INST "BM_pbt_to_add" LOC = "SLICE_X32Y10" ;
```

Obs.: As *busmacros* devem ser posicionadas duas colunas antes do limite da área do módulo.

- salvar e fechar o arquivo.

B3 TRADUÇÃO DO PROJETO PRINCIPAL

O projeto principal é traduzido para incorporar as *busmacros*:

- comando “ngdbuild -p xc3s200ft256 -sd ../busmacros/NMC/ -uc calctop.ucf -modular initial calctop.ngc calctop.ngd”;
- o arquivo “calctop.ngd” é sobrescrito.

B4 IMPLEMENTAÇÃO DA PARTE ESTÁTICA

Copiar os arquivos “ngc” dos módulos acionador de LCD e de botões (ISE\lcd_driver.ngc e ISE\pushbutton.ngc) para o diretório “base”. Neste diretório proceder a implementação da parte estática:

- tradução com o comando “ngdbuild -p xc3s200ft256 -sd ../busmacros/NMC/-uc ../top/calctop.ucf -modular initial ../top/calctop.ngc calctop.ngd”;
 - é gerado um arquivo com extensão “ngd” (calctop.ngd);
- mapeamento através do comando “map calctop.ngd”;
 - são gerados arquivos com as extensões “pcf” e “ncd” e ngm”, além de um arquivo de relatório do mapeamento, com a extensão “mrp”;
- posicionamento e roteamento utilizando “par -uc ../top/calctop.ucf -w calctop.ncd calctop_base_routed.ncd”;
 - é gerado um novo arquivo ncd com o sufixo “routed” (calctop_routed.ncd).

Ao verificar o arquivo gerado no editor de FPGA, nota-se vários sinais que traspassam os limites do MPR, o que não é permitido na família Spartan-3. Nos FPGAs da família Virtex não haveria este problema. Para solucioná-lo deve-se:

- desrotear os sinais que invadiram a área do MPR, selecionando-os e clicando no botão “unroute”;
- acessar a configuração de roteamento no menu “tools – route – auto route setup” e mudar a opção selecionada em “auto route selection”, no caso, de “resource driven” para “delay driven”;
- selecionar cada linha não roteada e clicar em “autoroute” até rotear todas as linhas;
- salvar o arquivo.

B5 IMPLEMENTAÇÃO DOS MPRs

Este procedimento será repetido para cada um dos MPRs.

Copiar o arquivo “ngc” do módulo de captura (ISE\capture.ngc) para o subdiretório correspondente em “reconfigmodules” (reconfigmodules\capture). Neste subdiretório proceder a implementação do módulo de captura:

- cópia do arquivo “base\static.used” e renomeação para “arcs.exclude” utilizando “copy ../../base/static.used arcs.exclude”;
- tradução com o comando “ngdbuild -p xc3s200ft256 -sd ../../busmacros/NMC/ -uc ../../top/calctop.ucf -modular module -active capture ../../top/calctop.ngc”;
 - é gerado um arquivo com extensão “ngd” (calctop.ngd);
- mapeamento do módulo através do comando “map calctop.ngd”;
 - são gerados arquivos com as extensões “pcf”, “ncd” e “ngm”, além de um arquivo de relatório do mapeamento, com a extensão “mrp”;
- posicionamento e roteamento do módulo utilizando “par -ol high -w calctop.ncd capture_routed.ncd”;
 - é gerado um novo arquivo “ncd” com o sufixo “routed” (calctop_routed.ncd). Este arquivo pode ser visualizado no editor de FPGA;

Esses passos são repetidos para os módulos somador (adder) e subtrator (subtractor), cada módulo em seu respectivo subdiretório. É importante observar que o comando “ngdbuild” não é alterado, porém, no comando “par”, deve-se utilizar o nome do MPR em questão (adder_routed.ncd e subtractor_routed.ncd).

B6 GERAÇÃO DOS BITSTREAMS

Este procedimento será repetido para cada configuração do SPR, no caso, para cada um dos MPRs.

Copiar o arquivo roteado do módulo de captura (reconfigmodules\capture\capture_routed.ncd) para o subdiretório correspondente em “merges” (merges\capture).

Os *bitstreams* serão gerados a partir de um processo de fusão:

- cópia do arquivo roteado da parte estática com o comando “copy

..\.\base\calctop_base_routed.ncd”;

- geração dos *bitstreams* parciais e da parte estática através do comando “pr_verifydesign calctop_base_routed.ncd capture_routed.ncd”;
- geração do *bitstream* da configuração completa do SPR executando “pr_assemble calctop_base_routed.ncd capture_routed.ncd”.

Os comandos “pr_verifydesign” e “pr_assemble” geram, além dos *bitstreams*, os correspondentes arquivos “ncd” que podem ser visualizados no editor de FPGA e um arquivo “calctop_base_routed.summary” onde há informações sobre a geração dos *bitstreams* e onde são relatados eventuais erros.

A Fig. B2 mostra o resultado do roteamento do módulo de captura e a Fig.B3 mostra o da parte estática, ambos gerados pelo “pr_verifydesign”. A Fig.B4 mostra o resultado do roteamento de todo o SPR, gerado pelo comando “pr_assemble”.

Este procedimento deve ser repetido para os módulos somador e subtrator, de modo a se obter todos os *bitstreams* do SPR.

Esses *bitstreams* podem então ser gravados no FPGA utilizando-se a ferramenta iMPACT, do ISE.

Fig. B2 - Resultado do roteamento do módulo de captura

Fig.B3 - Resultado do roteamento da parte estática

Fig.B4 - Resultado do roteamento para a configuração completa do SPR

APÊNDICE C – ROTEIRO DE PROJETO UTILIZANDO A METODOLOGIA DO PARBIT

Este roteiro demonstra o projeto de um SPR de acordo com a metodologia do PARBIT, tomando como exemplo o SPR de uma calculadora para a família Virtex-E de FPGAs. Essa calculadora possui dois módulos fixos, um acionador de LCD e um módulo de botões, com um MPR central que possui três configurações: um módulo de captura (inicial), um somador e um subtrator. Divulgado inicialmente pela Xilinx como um projeto de referência na versão 4.2 do ISE, os arquivos HDL de descrição dos módulos foram modificados para compatibilização com a atual versão do ISE, ou com o FPGA utilizado.

A implementação do SPR da calculadora começa com a criação de uma estrutura de diretórios adequada, conforme se segue:

Os arquivos das *busmacros* e das descrições dos circuitos (se houverem) devem ser copiados para os respectivos diretórios.

A partir deste ponto pode-se acompanhar passo-a-passo a metodologia, conforme exibido na Fig. C1.

Fig. C1 - Metodologia de projeto com a indicação dos itens correspondentes

C1 CONFIGURAÇÃO INICIAL — MÓDULOS FIXOS E MPR DE CAPTURA

C1.1 Descrição e síntese do projeto principal

O *software* ISE deve ser aberto e um novo projeto criado, no diretório indicado. Deve ser selecionada a família Virtex-E e o dispositivo XCV300E. As outras configurações podem ser as padrões, notando-se que o tipo de arquivo do projeto principal (*top-level*) deve ser HDL:

- menu “file – new project”.

O arquivo do projeto principal (calctop.vhd) deve ser inserido no circuito:

- menu “project – add source”.

Este arquivo não deve conter lógica, apenas uma descrição do projeto, com o instanciamento de cada módulo na forma de “caixas-pretas”, inclusive as *busmacros*, a definição dos sinais entre os módulos e dos sinais e pinos de E/S. No caso, o arquivo foi modificado a partir do projeto inicial, com algumas modificações para compatibilização do código e acréscimo das *busmacros*.

Devem ser declarados todos os tipos de *busmacros* utilizados como componentes, conforme o exemplo:

```
component BM_VirtexE_RL is
  port (
    input1 : in std_logic;
    input2 : in std_logic;
    input3 : in std_logic;
    input4 : in std_logic;
    input5 : in std_logic;
    input6 : in std_logic;
    input7 : in std_logic;
    input8 : in std_logic;
    output1 : out std_logic;
    output2 : out std_logic;
    output3 : out std_logic;
    output4 : out std_logic;
    output5 : out std_logic;
    output6 : out std_logic;
    output7 : out std_logic;
    output8 : out std_logic
  );
end component;
```

Devem ser descritas todas as *busmacros* utilizadas no projeto, conforme o exemplo:

```
BM_add_to_lcd: BM_VirtexE_RL
  port map (
    input1 => Address_int_from_RM,
    output1 => Address_from_RM_thruBM,
    input2 => Sync_S2P_sig,
    output2 => Sync_S2P_lcd_thruBM,
    input3 => Sync_P2S_sig,
    output3 => Sync_P2S_lcd_thruBM,
    input4 => LCD_data_from_RM,
    output4 => LCD_data_from_RM_thruBM,
    input5 => WriteEnable_from_RM,
    output5 => WriteEnable_from_RM_thruBM,
    input6 => '1',
    output6 => open,
    input7 => '1',
    output7 => open,
    input8 => '1',
    output8 => open
  );
```

Após as modificações necessárias o arquivo deve ser definido como projeto-principal

(*top-level*) e sintetizado:

- seleccionar o arquivo na janela sources e utilizar o menu “source – set as top module”;
- duplo clique em “Synthesize – XST” na janela de processos;
 - Toda síntese gera um arquivo com mesmo nome e extensão “.ngc” (calctop.ngc) no diretório do projeto.

Se a síntese não for bem-sucedida deve-se investigar os erros e saná-los, repetindo a síntese em seguida. É comum que haja advertências (*warnings*) resultantes do projeto. Elas devem ser analisadas pelo projetista para avaliar se são todas aceitáveis ou se há algum problema que deva ser corrigido. Após a síntese, o arquivo do projeto principal deve ser removido do projeto:

- seleccionar o arquivo na janela “sources” e pressionar a tecla “delete”.

C1.2 Descrição e síntese dos módulos

As próximas etapas serão repetidas para cada um dos módulos componentes do SPR.

Deve-se inserir no projeto o arquivo do módulo acionador de LCD (lcd_driver.vhd) e defini-lo como projeto principal:

- menu “project – add source”;
- seleccionar o arquivo na janela sources e utilizar o menu “source – set as top module”.

Antes da síntese deve-se remover a opção para adição de *buffers* de E/S, já que essas conexões foram declaradas no arquivo do projeto-principal e aqui há apenas o conteúdo da “caixa-preta” lá declarada:

- seleccionar a opção “Synthesize – XST” na janela de processos e o menu “source – properties”. Seleccionar “Xilinx Specific Options” e desmarcar a caixa “Add I/O Buffers”.

Sintetizar o arquivo do módulo:

- duplo clique em “Synthesize – XST” na janela de processos.

Se houver algum erro, ele deve ser corrigido e o processo repetido. Do contrário, o arquivo deve ser removido do projeto:

- selecionar o arquivo na janela “sources” e pressionar a tecla “delete”.

Esses passos devem ser repetidos para os módulos de botões (pushbutton.vhd) e captura (capture.vhd).

Com todos os módulos sintetizados o ISE pode ser fechado.

C1.3 Posicionamento e definição de restrições

Para esta etapa, deve-se copiar o arquivo “ngc” do projeto principal (calctop.ngc) para o diretório “top\initial” e fazer uma tradução inicial do arquivo:

- comando “ngdbuild -modular initial calctop.ngc”;
 - este comando gera um arquivo com mesmo nome e extensão “ngd” (calctop.ngd).

Para o posicionamento dos módulos é utilizado o “floorplanner”:

- comando “floorplanner”;
- abrir o arquivo “\top\initial\calctop.ngd” gerado previamente.

Cada módulo deve ter sua área atribuída no componente, a saber o módulo acionador do LCD no lado esquerdo, o MPR de captura no centro e o módulo de botões no lado direito. O posicionamento é feito da seguinte forma:

- selecionar o módulo correspondente na janela “design hierarchy” e o menu “floorplan – assign area constraint”;
- marcar a área com o *mouse* no desenho do FPGA.

Os pinos de E/S apresentados em “calctop primitives” também devem ser atribuídos a um pino de E/S do FPGA arrastando-se o símbolo da janela de hierarquia para um pino do desenho do FPGA. As primitivas que ficam no lado esquerdo são “LCD_Databus[7:0]”, “E”, “RW”, “RS”, “Reset_lcd_driver”. As primitivas que ficam no lado direito são “TopPushButton”, “RightPushButton”, “MiddlePushButton”, “LeftPushButton”, “DownPushButton”, “LED_A_out”, “LED_B_out”, “LED_C_out”,

“LED_D_out”, “Reset_pushbutton” e “Reset_reconfig_module”. As primitivas “U1” e “Clock_in” não são adicionadas. O *software* as atribui automaticamente ao *clock* global do FPGA. O floorplanner deve ser fechado, confirmando-se o salvamento dos arquivos.

O floorplanner gera um arquivo de restrições de usuário, com extensão “ucf” (calctop.ucf). As *busmacros* devem ser posicionadas manualmente neste arquivo:

- abrir o arquivo “calctop.ucf” em um editor de texto puro;
- acrescentar as seguintes linhas ao arquivo:

```
# BUS MACROS
INST "BM_lcd_to_add" AREA_GROUP = "AG_BM_lcd_to_add" ;
AREA_GROUP "AG_BM_lcd_to_add" RANGE = CLB_R6C12:CLB_R7C13 ;
INST "BM_lcd_to_add" LOC = "CLB_R6C12.S1" ;
INST "BM_add_to_lcd" AREA_GROUP = "AG_BM_add_to_lcd" ;
AREA_GROUP "AG_BM_add_to_lcd" RANGE = CLB_R30C12:CLB_R31C13 ;
INST "BM_add_to_lcd" LOC = "CLB_R30C12.S1" ;
INST "BM_add_to_pbt" AREA_GROUP = "AG_BM_add_to_pbt" ;
AREA_GROUP "AG_BM_add_to_pbt" RANGE = CLB_R6C36:CLB_R7C37 ;
INST "BM_add_to_pbt" LOC = "CLB_R6C36.S1" ;
INST "BM_pbt_to_add" AREA_GROUP = "AG_BM_pbt_to_add" ;
AREA_GROUP "AG_BM_pbt_to_add" RANGE = CLB_R30C36:CLB_R31C37 ;
INST "BM_pbt_to_add" LOC = "CLB_R30C36.S1" ;
```

A *busmacro* criada para a família Virtex-E se inicia uma coluna antes do limite da área do módulo.

Feitas as restrições a etapa é finalizada com uma nova tradução do projeto principal, para se incorporar as *busmacros*:

- comando “ngdbuild -p xcv300efg256 -sd ..\..\busmacros\NMC calctop.ucf -modular initial calctop.ngc calctop.ngd”;
- o arquivo “calctop.ngd” é sobrescrito.

C1.4 Implementação dos módulos

Copiar o arquivo “ngc” do módulo de captura (SPR\capture.ngc) para o subdiretório correspondente em “modules” (modules\capture). Neste subdiretório proceder a implementação do módulo de captura:

- tradução com o comando “ngdbuild -p xcv300efg256 -sd ..\..\busmacros\NMC -uc ..\..\top\initial\calctop.ucf -modular module -active capture

- ..\\..\\top\\initial\\calctop.ngc”;
 - é gerado um arquivo com extensão “ngd” (calctop.ngd);
- mapeamento do módulo através do comando “map -ol high calctop.ngd”;
 - são gerados arquivos com as extensões “pcf” e “ncd” e ngm”, além de um arquivo de relatório do mapeamento, com a extensão “mrp”;
- posicionamento e roteamento do módulo utilizando “par -ol high -uc ..\\..\\top\\initial\\calctop.ucf -w calctop.ncd calctop_routed.ncd”;
 - é gerado um novo arquivo ncd com o sufixo “routed” (calctop_routed.ncd). Este arquivo pode ser visualizado no editor de FPGA;
- a implementação do módulo é concluída com sua publicação através do comando “pimcreate ..\\..\\pim -ncd calctop_routed.ncd”;
 - os arquivos necessários para a implementação final são publicados em “\\pim\\capture”.

Esses passos são repetidos para os módulos acionador de LCD (lcd_driver) e de botões (pushbutton). Cada módulo em seu respectivo diretório, tomando-se ainda o cuidado de alterar o nome do módulo ativo no comando “ngdbuild”.

C1.5 Implementação da configuração completa do SPR

Para a implementação da configuração inicial do SPR (todos os módulos estáticos mais o MPR de captura), devem ser copiados os arquivos “\\top\\initial\\calctop.ucf” e “\\SPR\\calctop.ngc” para o diretório “\\top\\final”.

- o projeto é traduzido usando-se o comando “ngdbuild -p xcv300efg256 -sd ..\\..\\busmacros\\NMC -uc calctop.ucf -modular assemble -pimpath ..\\..\\pim calctop.ngc”;
- o mapeamento é realizado com “map -ol high calctop.ngd”;
- o posicionamento e roteamento dá-se com o comando “par -ol high -w calctop.ncd calctop_routed.ncd”;

C1.6 Inspeção visual

O resultado do roteamento pode ser inspecionado visualmente no editor de FPGA (Fig. C2) com o comando:

- “fpga_editor calctop_routed.ncd”.

Fig. C2 - Resultado do roteamento da configuração inicial

Caso necessário, deve-se corrigir o roteamento manualmente.

C1.7 Geração dos bitstreams

O *bitstream* total da configuração inicial é gerado com o comando:

- “bitgen -g ActiveReconfig:yes -g Persist:yes calctop_routed.ncd calctop_routed.bit”

O *bitstream* parcial do módulo de captura é gerado com o PARBIT, utilizando sua interface gráfica como se segue:

- o PARBIT GUI é iniciado executando-se o arquivo “PARBIT.jar”;
- selecionar a opção “file – new” do menu e criar o arquivo “capture.opt” em um

diretório de preferência (“\top\final”, por exemplo);

- na janela “mode and file selection” em “general configuration”, selecionar o modo de operação “slice”, o arquivo “\top\final\calctop_routed.bit” em “original”, o arquivo “\top\final\capture.bit” em “partial” e salvar as opções (Fig. C3);

Fig. C3 - Janela de seleção de arquivos e modo de operação

- pode-se mudar alguma das opções na janela “configuration options”, se desejado;
- em “area configuration”, selecionar “reconfigurable area”. Esta janela deve ser preenchida conforme a restrição de área feita na etapa de posicionamento e definição de restrições. Caso não se tenha anotado os valores, basta se verificar no arquivo “ucf” as restrições de área para o MPR, no caso entre as colunas 13 e 36:

```
AREA_GROUP "AG_Reconfigurable_Module" RANGE = CLB_R1C13:CLB_R32C36 ;
```

- clicar em “preview”, a área é visualizada, bem como uma estimativa para o tempo de reconfiguração. Se estiver correto clicar em “save” (Fig. C4);

Fig. C4 - Janela de seleção de área

- em “bitstream generation” e “generate bitstream”, clicar no botão “generate bitstream”;
 - o *bitstream* parcial é gerado no arquivo “capture.bit”. O relatório é exibido na tela e pode ser visto a qualquer tempo na opção “report” do menu.

Com isso temos o *bitstream* completo da configuração inicial e o *bitstream* parcial do primeiro MPR.

C2 CONFIGURAÇÃO COM MPR SOMADOR

Para obtermos o *bitstream* parcial do MPR somador será seguido o fluxo mostrado na parte central da Fig. C1, que permite a simulação e teste do SPR.

C2.1 Descrição e síntese do MPR somador

O *software* ISE é novamente aberto e carregado o mesmo projeto do SPR. O arquivo de descrição do MPR somador (adder.vhd) será inserido no projeto e sintetizado. Este módulo (e o subtrator) possuem um sub-módulo de conversão

binário-BCD (binary_to_bcd.vhd) que também deve ser inserido.

- inserir os arquivos “adder.vhd” e “binary_to_bcd.vhd” através da opção “project – add source” do menu;
- seleccionar o módulo “adder” na janela sources e utilizar o menu “source – set as top module”;
- seleccionar a opção “Synthesize – XST” na janela de processos e o menu “source – properties”. Seleccionar “Xilinx Specific Options” e desmarcar a caixa “Add I/O Buffers”;
- duplo clique em “Synthesize – XST” na janela de processos.

C2.2 Implementação do MPR somador

Copiar o arquivo “ngc” do módulo somador (SPR\adder.ngc) para o subdirectório correspondente em “modules” (modules\adder). O módulo presente no SPR é o “capture”. Para maior agilidade na implementação, ao invés de repetir todo o fluxo, deve-se simplesmente renomear o arquivo e proceder a implementação do módulo somador:

- renomear o arquivo “modules\adder\adder.ngc” para “modules\adder\capture.ngc”;
- tradução com o comando “ngdbuild -p xcv300efg256 -sd ..\..\busmacros\NMC -uc ..\..\top\initial\calctop.ucf -modular module -active capture ..\..\top\initial\calctop.ngc”;
- mapeamento do módulo através do comando “map -ol high calctop.ngd”;
- posicionamento e roteamento do módulo utilizando “par -ol high -uc ..\..\top\initial\calctop.ucf -w calctop.ncd calctop_routed.ncd”;

A publicação do módulo com o comando “pimcreate” irá sobrescrever os arquivos anteriores do módulo de captura, por isso pode-se renomear o directório anterior, por exemplo, para “\pim\capture-initial”.

- a implementação do módulo é concluída com sua publicação através do comando “pimcreate ..\..\pim -ncd calctop_routed.ncd”;

- os arquivos necessários para a implementação final são publicados em “\pim\capture”, mas correspondem ao módulo somador.

Se esse mecanismo de renomeação não fosse utilizado, seria necessário repetir todo o fluxo desde a etapa de descrição e síntese do projeto principal, uma vez que o nome do módulo reconfigurável ativo é definido no HDL do projeto principal.

C2.3 Implementação da configuração completa do SPR

Para a implementação da configuração completa do SPR com o MPR somador devem ser copiados os arquivos “\top\initial\calctop.ucf” e “\SPR\calctop.ngc” para um novo diretório, no caso “\top\final-adder”.

- o projeto é traduzido usando-se o comando “ngdbuild -p xcv300efg256 -sd ../../busmacros\NMC -uc calctop.ucf -modular assemble -pimpath ../../pim calctop.ngc”;
- a montagem é realizada com “map -ol high calctop.ngd”;
- o posicionamento e roteamento dá-se com o comando “par -ol high -w calctop.ncd calctop_routed.ncd”;

C2.4 Inspeção visual

O resultado do roteamento pode ser inspecionado visualmente no editor de FPGA com o comando:

- “fpga_editor calctop_routed.ncd”.

Caso necessário, deve-se corrigir o roteamento manualmente.

C2.5 Geração dos bitstreams

O *bitstream* total para esta configuração com o MPR somador é gerado com o comando:

- “bitgen -g ActiveReconfig:yes -g Persist:yes calctop_routed.ncd

calctop_routed.bit”.

O *bitstream* parcial do módulo de captura é gerado com o PARBIT, utilizando sua interface gráfica. Antes disso, para simplificar o processo pode-se aproveitar uma cópia do arquivo da GUI utilizado para a configuração anterior.

- copiar o arquivo “\top\final\capture.btp” para o diretório “\top\final-adder” e renomeá-lo para “adder.btp”;
- iniciar o PARBIT GUI executando o arquivo “PARBIT.jar”;
- selecionar a opção “file – open” do menu e abrir o arquivo “\top\final-adder\adder.btp”;
- na janela “mode and file selection” em “general configuration” os arquivos devem ser mudados para “top\final-adder\calctop_routed.bit” em “original” e “top\final\adder.bit” em “partial”, salvando-se a nova configuração;
- em “bitstream generation” e “generate bitstream”, clicar no botão “generate bitstream”;
 - o *bitstream* parcial é gerado no arquivo “adder.bit”. O relatório é exibido na tela e pode ser visto a qualquer tempo na opção “report” do menu.

Com isso temos o *bitstream* parcial do MPR somador.

C3 MPR SUBTRATOR

Para a geração do *bitstream* parcial do MPR subtrator será seguido o procedimento para a geração de um BIP, conforme ilustrado na coluna direita da Fig. C1. Esse fluxo tem apenas o MPR subtrator e um novo projeto principal.

C3.1 Descrição e síntese do projeto principal

Deve-se criar um novo diretório, com a mesma estrutura do utilizado para a configuração inicial, porém apenas com o módulo “subtractor”. O *software* ISE deve ser aberto e um novo projeto criado, no diretório correspondente, da mesma forma

que indicado anteriormente.

O arquivo do projeto principal (*calctop.vhd*) deve ser inserido no circuito:

- menu “project – add source”.

Novamente, este arquivo não deve conter lógica, apenas uma descrição do projeto, com o instanciamento do MPR na forma de uma “caixa-preta”, as *busmacros* e a definição dos sinais entre o MPR e os pinos de E/S. No caso, o arquivo pode ser modificado a partir do utilizado no projeto inicial, eliminando tudo o que se refira à parte estática do projeto e substituindo o nome do módulo “capture” por “subtractor”. Cabe lembrar que esse arquivo já contém as *busmacros*.

O arquivo deve ser definido como projeto-principal (*top-level*), sintetizado e depois removido:

- selecionar o arquivo na janela sources e utilizar o menu “source – set as top module”;
- duplo clique em “Synthesize – XST” na janela de processos;
- selecionar o arquivo na janela “sources” e pressionar a tecla “delete”.

C3.2 Descrição e síntese do MPR subtrator

O arquivo de descrição do MPR subtrator (*subtractor.vhd*) deve ser inserido no projeto, juntamente com o sub-módulo de conversão binário-BCD (*binary_to_bcd.vhd*), e depois sintetizado:

- inserir os arquivos “subtractor.vhd” e “binary_to_bcd.vhd” através da opção “project – add source” do menu;
- selecionar o módulo “subtractor” na janela sources e utilizar o menu “source – set as top module”;
- selecionar a opção “Synthesize – XST” na janela de processos e o menu “source – properties”. Selecionar “Xilinx Specific Options” e desmarcar a caixa “Add I/O Buffers”;
- duplo clique em “Synthesize – XST” na janela de processos.

Os arquivos podem ser removidos do projeto e o ISE fechado.

C3.3 Posicionamento e definição de restrições

Para esta etapa, deve-se copiar o arquivo “ngc” do projeto principal (calctop.ngc) para o diretório “top\initial” e fazer uma tradução inicial do arquivo:

- comando “ngdbuild -modular initial calctop.ngc”;

Para o posicionamento dos módulos é utilizado o floorplanner:

- comando “floorplanner”;
- abrir o arquivo “\top\initial\calctop.ngd” gerado previamente.

Atribuir a área do MPR subtrator na mesma posição central que foi utilizada quando da configuração inicial. Os pinos de E/S apresentados em “calctop Primitives” devem ser atribuídos no FPGA. As primitivas que ficam no lado esquerdo são “Calc_data_from_memory”, “Sync_S2P_lcd_thruBM”, “Address_from_RM_thruBM”, “WriteEnable_from_RM_thruBM”, “LCD_data_from_RM_thruBM” e “Sync_P2S_lcd_thruBM”. As primitivas que ficam no lado direito são “Reset_reconfig_module”, “Sync_S2P_PB_thruBM”, “LED_RM_sig_thruBM”, “MiddlePB_Rising”, “LeftPB_Rising”, “DownPB_Rising”, “RightPB_Rising” e “TopPB_Rising”. As primitivas “U1” e “Clock_in” não são adicionadas. O *software* as atribui automaticamente ao *clock* global do FPGA. O floorplanner deve ser fechado, confirmando-se o salvamento dos arquivos.

O floorplanner gera um arquivo de restrições de usuário (calctop.ucf) no qual devem ser posicionadas as *busmacros*. Como elas têm que ficar na mesma posição da configuração inicial, bastar editar o arquivo inserindo as mesmas linhas que foram acrescentadas ao arquivo “ucf” da configuração inicial do SPR.

Esta etapa é finalizada com uma nova tradução do projeto principal, para se incorporar as *busmacros*:

- comando “ngdbuild -p xcv300efg256 -sd ..\..\busmacros\NMC calctop.ucf -modular initial calctop.ngc calctop.ngd”;
- o arquivo “calctop.ngd” é sobrescrito.

C3.4 Implementação do MPR subtrator

Copiar o arquivo “ngc” do módulo subtrator (ISE\subtractor.ngc) para o subdiretório correspondente em “modules” (modules\subtractor). Neste subdiretório proceder a implementação do MPR:

- tradução com o comando “ngdbuild -p xcv300e-6fg256 -sd ..\..\busmacros\NMC -uc ..\..\top\initial\calctop.ucf -modular module -active subtractor ..\..\top\initial\calctop.ngc”;
- mapeamento do módulo através do comando “map -ol high calctop.ngd”;
- posicionamento e roteamento do módulo utilizando “par -ol high -uc ..\..\top\initial\calctop.ucf -w calctop.ncd calctop_routed.ncd”;
- a implementação do módulo é concluída com sua publicação através do comando “pimcreate ..\..\pim -ncd calctop_routed.ncd”;
 - os arquivos necessários para a implementação final são publicados em “\pim\subtractor”.

C3.5 Implementação da configuração completa do SPR

Para a implementação da configuração completa do SPR devem ser copiados os arquivos “\top\initial\calctop.ucf” e “\ISE\calctop.ngc” para o diretório “\top\final”.

- o projeto é traduzido usando-se o comando “ngdbuild -p xcv300efg256 -sd ..\..\busmacros\NMC -uc calctop.ucf -modular assemble -pimpath ..\..\pim calctop.ngc”;
- a montagem é realizada com “map -ol high calctop.ngd”;
- o posicionamento e roteamento dá-se com o comando “par -ol high -w calctop.ncd calctop_routed.ncd”;

C3.6 Inspeção visual

O resultado do roteamento pode ser inspecionado visualmente no editor de FPGA com o comando (Fig. C5):

- “fpga_editor calctop_routed.ncd”.

Fig. C5 - Aspecto inicial do roteamento do MPR subtrator e projeto principal

Neste caso, nota-se uma série de sinais que traspassaram os limites do MPR (Fig. C5), o que não é permitido. Para solucionar este problema deve-se proceder o seguinte:

- desrotear os sinais que invadiram a área do MPR, selecionando-os e clicando no botão “unroute”;
- acessar a configuração de roteamento no menu “tools – route – auto route setup” e mudar a opção selecionada em “auto route selection”, no caso, de “resource driven” para “delay driven”;
- selecionar cada linha não roteada e clicar em “autoroute” até rotear todas as linhas (Fig. C6);

Fig. C6 - Aspecto final do roteamento após as correções

- salvar o arquivo.

Pode ser que o roteamento que esteja em lugar impróprio não seja corrigido na primeira tentativa como descrito, neste caso há algumas estratégias que podem ser utilizadas:

- alterar novamente a configuração em “auto route setup” para a linha específica;
- desrotear todo o projeto e alterar em “auto route setup” a configuração do “auto route design” e rotear todo o projeto novamente;
- desrotear todo o projeto e rotear individualmente as linhas que invadiram a área do MPR, depois fazer o roteamento do restante do projeto;
- remover componentes que estejam dificultando o roteamento em “unplace” e reinseri-los de modo automático (“auto place”) ou manual (“manual place”) e rotear novamente as linhas afetadas;
- proceder um roteamento manual das linhas que estão com problema.

C3.7 Geração dos bitstreams

O *bitstream* total da configuração é gerado com o comando:

- “bitgen -g ActiveReconfig:yes -g Persist:yes calctop_routed.ncd calctop_routed.bit”

O *bitstream* parcial do módulo de captura é gerado usando o PARBIT GUI, aproveitando-se novamente uma cópia do arquivo da GUI utilizado nas configurações anteriores:

- copiar o arquivo “capture.btp” ou “adder.btp” utilizado nas configurações anteriores para o diretório “\top\final” e renomeá-lo para “subtractor.btp”;
- iniciar o PARBIT GUI executando o arquivo “PARBIT.jar”;
- selecionar a opção “file – open” do menu e abrir o arquivo “\top\final-adder\subtractor.btp”;
- na janela “mode and file selection” em “general configuration” os arquivos devem ser mudados para, dentro do diretório do projeto atual, “\top\final\calctop_routed.bit” em “original” e “\top\final\subtractor.bit” em “partial”, salvando-se a nova configuração;
- em “bitstream generation” e “generate bitstream”, clicar no botão “generate bitstream”;
 - o *bitstream* parcial é gerado no arquivo “subtractor.bit”. O relatório é exibido na tela e pode ser visto a qualquer tempo na opção “report” do menu.

Com isso temos o *bitstream* parcial do último MPR.

Os *bitstreams* do SPR da calculadora podem então ser gravados no FPGA utilizando-se a ferramenta iMPACT, integrante do ISE da Xilinx.

Caso se estivesse projetando um BIP utilizando um modelo de FPGA diferente do utilizado no projeto alvo, o procedimento de geração do *bitstream* parcial seria o seguinte:

- o autor do BIP envia o *bitstream* da configuração completa do MPR, gerado com o comando “bitgen”, para o projetista que utilizará o BIP;

- o projetista que utilizará o BIP gera um *bitstream* de seu projeto que será utilizado como alvo no PARBIT GUI. Esse *bitstream* alvo deve conter as *busmacros* e a área reservada no projeto para o BIP. Essa área deve estar preenchida com uma lógica fictícia como, por exemplo, um registrador de deslocamento, ou mesmo com um outro MPR que venha a ser utilizado na mesma área. O preenchimento da área destinada ao BIP com alguma lógica é necessário para assegurar o roteamento do *clock*;
- na geração do *bitstream* parcial utilizando o PARBIT GUI, devem ser fornecidos o nome do *bitstream* do BIP fornecido por seu autor (“original”), o nome do *bitstream* parcial a ser gerado (“partial”) e o nome do *bitstream* alvo (“target”) criado pelo projetista que utilizará o BIP conforme descrito no passo anterior;
- em “reconfigurable area” deve ser selecionada a área do MPR do BIP conforme fornecido por seu autor;
- em “target area” deve ser fornecida a posição onde de fato ficará o BIP no projeto;
- os demais passos da geração do *bitstream* parcial são idênticos aos descritos anteriormente.