MARIO ANDRÉS RAFFO JARA

**DESENVOLVIMENTO DE UM SISTEMA DINAMICAMENTE RECONFIGURÁVEL
BASEADO EM REDES INTRA-CHIP E FERRAMENTA PARA POSICIONAMENTO
DE MÓDULOS**

São Paulo

2010

MARIO ANDRÉS RAFFO JARA

**DESENVOLVIMENTO DE UM SISTEMA DINAMICAMENTE RECONFIGURÁVEL
BASEADO EM REDES INTRA-CHIP E FERRAMENTA PARA POSICIONAMENTO
DE MÓDULOS**

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. Wang Jiang Chau

São Paulo
2010

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, 05 de março de 2010.

Assinatura do autor _____

Assinatura do orientador _____

FICHA CATALOGRÁFICA

Raffo Jara, Mario Andrés

Desenvolvimento de um sistema dinamicamente reconfigurável baseado em redes intrachip e ferramenta para posicionamento de módulos / M.A. Raffo Jara. -- ed.rev. --São Paulo, 2010.

107 p.

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

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

DEDICATÓRIA

Dedico esta dissertação basicamente a duas pessoas:

À mulher mais maravilhosa e forte do mundo, María Emilia Jara Huamán de Raffo, minha mãe, quem apesar de ter ficado viúva exatamente duas semanas depois de minha partida do Peru ao Brasil, sempre me alentou para que eu fizesse o mestrado.

À memória do meu caro pai, Andrés Federico Raffo Hernández, quem me viu partir ao Brasil a procura do meu sonho, mas lastimosamente a vida não lhe deu tempo para me ver voltar triunfante. Sempre estará na minha mente e no meu coração.

AGRADECIMENTOS

Ao professor Dr. Wang Jiang Chau, pela oportunidade de ser o seu orientando, por todo o aprendizado neste tempo e pela sua sincera preocupação por minha pessoa.

Aos meus pais, por sempre terem lutado por mim. Todo o que eu sou e o que eu serei é por eles, são minha motivação.

À minha tia Margot e aos seus filhos, pelo carinho sincero e preocupação por mim.

À família Dulanto-Brescia, pela sincera preocupação e apoio à minha pessoa.

Ao Julio Saldaña, por ter me acolhido nos meus dois primeiros meses no São Paulo.

Aos professores da PUC do Peru (PUCP), Dr. Gustavo Kato, Dr. Carlos Silva e Eng. Andrés Flores, pelas suas recomendações para o ingresso na USP.

Aos amigos do Grupo de Microeletrônica da PUCP (GuE), especialmente ao meu amigo e colega da turma 2002-II, Jorge Lagos, que me incentivou a pertencer ao GuE. Veja Jorge, você traçou a linha da minha vida profissional.

Ao Dr. Marius Strum e aos demais colegas do GSEIS pelo aprendizado e vivência.

Aos amigos da USP, peruanos, brasileiros e colombianos, especialmente a Dante Coaquira, Joel Muñoz, Jonas Gomes Filho e Hugo Hernández.

A todos os meus familiares e aos amigos do Peru, da escola, do bairro, da PUCP (Ceprepuc, Turma 2002-II, GuE, Telecomunicações e Área Elétrica) que me deram suas amostras de carinho e solidariedade nos momentos difíceis. Nunca esquecerei os seus e-mails, mensagens pelo *msn* e suas chamadas por telefone.

Ao Peru por ser minha pátria e ao Brasil por ter me acolhido.

Ao MCT/CNPq, pelo apoio financeiro, para a realização deste trabalho

Só tenho para oferecer sangue, trabalho árduo, lágrimas e suor
(Winston Churchill)

RESUMO

Os sistemas dinamicamente reconfiguráveis (SDRs) são uma alternativa para o desenvolvimento de sistemas sobre silício baseados em circuitos programáveis (SoPC), cujo principal benefício é o bom aproveitamento da área do dispositivo. Sendo neles implementados circuitos que representam as tarefas que devem operar numa etapa específica do tempo de operação do sistema, permitem um menor consumo de área e de energia, parâmetros importantes nos sistemas portáteis. Isto tem gerado muito interesse no que se refere às metodologias de projeto utilizando FPGAs (*Field Programmable Gate Arrays*) dinamicamente reconfiguráveis (DRFPGAs) e à definição de um meio de comunicação estruturado para tratar da transferência de dados entre as partes reconfiguráveis e as fixas, mas estas tarefas, assim como a concretização de sua comunicação, seguem sendo ainda essencialmente manuais, devido à falta de metodologias de projeto e ferramentas de CAD que simplifiquem o projeto de SDRs. Este trabalho foca uma das limitações mais efetivas para a adoção da reconfiguração dinâmica: a falta de ferramentas de CAD que suportem o projeto de SDRs, inclusive os baseados em redes intra-chip (NoCs), em particular, no posicionamento dos módulos. Neste trabalho, uma arquitetura para SDRs baseado em NoCs é proposta e um algoritmo de posicionamento dos módulos de um SDR baseado em aspectos reais da família do DRFPGAs é desenvolvido, dentro de uma ferramenta denominada DynoPlace. Desenvolveu-se também um modelo de validação e simulação de SDRs, em tempo de operação, utilizando-se a técnica de chaveamento dinâmico de circuitos. Para o estudo do caso, de validação da arquitetura e metodologia, propõe-se uma aplicação teste baseada em computação de operações aritméticas. A metodologia de simulação permite determinar o tempo da reconfiguração e verificar o comportamento do SDR no momento da reconfiguração. A ferramenta DynoPlace permite gerar os arquivos de restrição de usuário (UCF) de posicionamento dos módulos do SDR no DRFPGA Virtex-4LX25. Este contém informações do posicionamento dos módulos do sistema, dos dispositivos usados para as entradas e saídas do sistema além do posicionamento dos *bus-macros*. Com os arquivos gerados pela metodologia e ferramenta DynoPlace, pode-se executar com sucesso

os *scripts* da metodologia Early Access da Xilinx para gerar o SDR de forma automática.

Palavras-chave: Reconfiguração dinâmica. Sistemas dinamicamente reconfiguráveis. DRFPGAs. Microeletrônica. Metodologias de Projeto. CAD para DRFPGAs. Chaveamento dinâmico de circuitos. Redes intra-chip.

ABSTRACT

Dynamically Reconfigurable Systems (DRSs) are an alternative for developing Systems on a Programmable Chip (SoPC), being the efficient use of device's area one of its main advantages. Circuits implemented as DRSs represent tasks which must be active in specific times into the system operation, allowing area and energy saving, which is an important goal for portable systems. This has generated interests on the design methodology using Dynamically Reconfigurable Field Programmable Gate Arrays (DRFPGAs) and on the definition of communication systems for handling data transfer between static and reconfigurable partitions. However, these tasks, as well as the communication structure, are still carried out manually due to lack of design methodologies and CAD tools applied to DRSs design. This work focuses on the one of main drawbacks to the adoption of dynamic reconfiguration methods: the absence of CAD tools which support DRS designs, specifically, in the module positioning task, included, for those based on Network-on-Chip (NoCs). In this work, an architecture for DRSs based on NoCs is presented and an algorithm for module positioning is developed in a tool called DynoPlace as well, based on real-specifications of DRFPGAs families. It is also developed a run-time simulation and validation model for DRSs, through a dynamic circuit switching technique. For the validation of architecture and methodology study case, an application test based on arithmetic operations has been proposed. The simulations methodology allows to determine the reconfiguration time and verify the DRS behavior at the moment of reconfiguration. The DynoPlace tool allows to generate User Constraint File (UCF) of DRS's modules positioning for the DRFPGA Virtex-4LX25. This file contains information of modules positioning in the system, of the devices used for inputs and outputs of the system, and the positioning of bus-macros. After the files generation by the methodology, and the DynoPlace tool, it is possible to successfully execute the Early Access *scripts* for generating the DRS automatically.

Keywords: Run time reconfiguration. Dynamically reconfigurable systems. DRFPGAs. Microelectronics. Design Methodology. CAD for DRFPGAs. Dynamic circuit switching. Network-on-chip.

SUMÁRIO

LISTA DE ILUSTRAÇÕES	iv
LISTA DE TABELAS	vi
LISTA DE ABREVIATURAS E SIGLAS	vii
1. INTRODUÇÃO	1
1.1. Contexto.....	1
1.2. Motivação.....	6
1.3. Justificativas.....	8
1.4. Objetivos	8
1.5. Organização do texto	10
2. FUNDAMENTOS TEÓRICOS.....	11
2.1. Arquiteturas de FPGAs com suporte da reconfiguração dinâmica.....	11
2.2. Sistemas dinamicamente reconfiguráveis (SDR).....	14
2.2.1. Metodologia Early Access	16
2.2.2. Metodologia para simulações de SDR.....	20
2.2.2.1. Chaves de Isolação.....	20
2.2.2.2. Computação do Tempo de Reconfiguração.....	22
2.3. Interfaces de comunicação.	23
2.3.1. Redes Intra-chip	24
2.3.2. Topologia.....	24
2.3.3. Algoritmos de roteamento.....	26
2.3.4. Mecanismo de Roteamento	27
2.3.5. Controle de Fluxo	28
2.3.6. Arbitragem	29
3. TRABALHOS RELACIONADOS	30
3.1. Tipos de interfaces de comunicação nos SDR	30
3.2. Propostas de interfaces de comunicação para SDR.....	32

3.2.1.	Proposta de Ahmadinia et al.....	32
3.2.2.	Proposta de Bobda et al.	34
3.2.3.	Proposta de Pionteck et al.....	36
3.2.4.	Proposta de Jovanovic et al.....	38
3.3.	Discussão das propostas de interfaces de comunicação revisadas.	41
3.4.	Propostas sobre posicionamento dos módulos nos DRFPGAs.	43
3.4.1.	Proposta de Teich et al.....	44
3.4.2.	Proposta de Ahmadinia et al, 2004.....	45
3.4.3.	Proposta de Tabero et al.	46
3.4.4.	Proposta de Ahmadinia et al, 2007.....	46
4.	ARQUITETURA E IMPLEMENTAÇÃO DE SDR COM NOC	47
4.1.	Arquitetura do SDR proposto.....	47
4.2.	A rede intra-chip Hermes	50
4.3.	PRMs no SDR com NoC.....	51
4.4.	Metodologia proposta para a simulação de SDR.....	53
4.5.	Implementação da metodologia de simulação do sistema usando DCS.....	55
5.	POSICIONAMENTO DE MÓDULOS	62
5.1.	Modificação da metodologia de projeto Early Access para SDRs.	62
5.2.	O problema do posicionamento	64
5.3.	Restrições de posicionamento	65
5.4.	Ferramenta DynoPlace	66
5.4.1.	A modelagem do sistema	67
5.4.2.	Algoritmo do DynoPlace	68
6.	RESULTADOS EXPERIMENTAIS DE POSICIONAMENTO	72
6.1.	Implementação da metodologia de posicionamento do DynoPlace.....	72
6.2.	Posicionamento dos módulos com a ferramenta DynoPlace.....	75
7.	CONCLUSÕES E TRABALHOS FUTUROS.....	85

7.1. Conclusões	85
7.2. Contribuições	87
7.3. Trabalhos futuros	88
8. REFERÊNCIAS BIBLIOGRÁFICAS	90
APENDICE A – DRFPGA DA XILINX	95
A.1. O dispositivo VirtexII	95
A.2. Dispositivo VirtexII-Pro	98
A.3. O dispositivo Virtex-4	98
APENDICE B – PARÂMETROS DE RECONFIGURAÇÃO DO DRFPGA VIRTEX-4 LX	101
APENDICE C – PROPOSTAS DE COMUNICAÇÃO DE SDR BASEADOS EM BARRAMENTOS	102
C.1. Proposta de Palma et al.	102
C.2. Proposta de Huebner et al.	104
C.3. Proposta de Walder et al.	106

LISTA DE ILUSTRAÇÕES

Figura 1 - Sistemas hardware no <i>FPGAs</i> comparado com <i>ASIC</i> (1).....	2
Figura 2 - Classificação dos <i>FPGAs</i> segundo sua capacidade de configuração (2). ..	3
Figura 3 – Reconfiguração dinâmica (2).	4
Figura 4 - Exemplo de um SDR (2).	4
Figura 5 - SDR implementado em um SoPC (6).	5
Figura 6 - Arquitetura do VirtexII (25).....	12
Figura 7 - Bloco lógico configurável (25).	12
Figura 8 - Configurações dos VirtexII (a) e Virtex-4 (b) (27).....	14
Figura 9 – Posicionamento do <i>bus-macro</i> (28).....	16
Figura 10 - <i>Bus-macro</i> : estreito (a) e amplo (b) (29).	17
Figura 11 - Três <i>bus-macros</i> aninhadas (29).	17
Figura 12 - Etapas da Metodologia Early Access.....	18
Figura 13 – Estrutura de simulação utilizando DCS (31).....	21
Figura 14 - Diagrama de blocos do exemplo de operação DCS.	22
Figura 15 - Topologias diretas: (a) Grelha 2D; (b) Toróide 2D; (c) Cubo 3D; (d) Cubo 4D; (e) Totalmente Conectada.	25
Figura 16 - Topologia indireta crossbar.....	25
Figura 17 - Diagrama de blocos do sistema RMBoc (37).	33
Figura 18 - Estrutura do crosspoint (37).....	34
Figura 19 - Interface de comunicação do sistema DyNoC (a). Estrutura do roteador (b) (29).	35
Figura 20 – Alocação dos components na DyNoC (29).	35
Figura 21 - Interface de comunicação do sistema CoNoChi (18).....	36
Figura 22 - Estrutura do nó de roteamento do sistema CoNoChi (18).	37
Figura 23 - Mapeamento dinâmico de módulos e do nodo de roteo (18).	37
Figura 24 - Estrutura interna de uma UC (19).	39
Figura 25 - Posições de UCs e de UCg (19).	40
Figura 26 - As três etapas para fornecer uma CuNoC (19).....	40
Figura 27 - Posicionamento temporal representado como módulos 3D (29).	44
Figura 28 - Sistema de múltiplas tarefas para um dispositivo reconfigurável (42).....	45
Figura 29 - Sistema dinamicamente reconfigurável.	48
Figura 30 - Metodologia da reconfiguração.....	51

Figura 31 - Comunicação entre roteador/PE (a); Comunicação entre roteador/BM/PE (b).....	52
Figura 32 - Metodologia de simulação usando DCS.	54
Figura 33 - Simulação do sistema dinamicamente reconfigurável no tempo de execução.....	57
Figura 34 - Simulação do SDR no tempo de execução detalhe da NoC.....	59
Figura 35 - Modificação na metodologia do projeto para SDR.....	63
Figura 36 - Configurações dos VirtexII (a) e Virtex-4 (b) (27).....	65
Figura 37 - Grafo G(I,IO,O,V,E).....	68
Figura 38 - Posicionamento de módulos no DRFPGA.	69
Figura 39 – Tarefas específicas do algoritmo do DynoPlace	69
Figura 40 - Etapas da ferramenta DynoPlace.	73
Figura 41 - Posicionamento dos módulos do SDR com o DynoPlace.....	76
Figura 42 - Roteamento do módulo reconfigurável PE4 no DRFPGA.....	77
Figura 43 - Roteamento do módulo reconfigurável PE5 no DRFPGA.....	77
Figura 44 - Roteamento do SDR completo.	78
Figura 45 - Posicionamento dos módulos de um SDR com o DynoPlace, ocupando a totalidade de CLBs do DRFPGA.	81
Figura 46 - Arquitetura do VirtexII (25).....	95
Figura 47 - Bloco lógico configurável (25).	96
Figura 48 - Slice do VirtexII (25).....	96
Figura 49 - Arquitetura do VirtexII-Pro (25).	98
Figura 50 - Diferenças entre os diversos dispositivos Virtex-4.....	99
Figura 51 - XtremeDSP slices (26).....	100
Figura 52 – Interface de comunicação proposta por Palma et al (13).....	102
Figura 53 – Diagrama de blocos do sistema de Huebner et al (14).	104
Figura 54 – Bus-Macro, entradas (a) e saídas (b) ao Bus-Com (14).....	105
Figura 55 - Diagrama de blocos do sistema de Walder et al (15).....	106
Figura 56 - Interface de comunicação do sistema de Walder et al (15).	107

LISTA DE TABELAS

Tabela 1 – Tabela de verdade da chave de isolamento	20
Tabela 2 – Propriedades arquiteturais de comunicação nos sistemas reconfiguráveis.	31
Tabela 3 - Parâmetros de desempenho da interface de comunicação nos sistemas reconfiguráveis.	31
Tabela 4 – Comparação segundo parâmetros de projeto.	41
Tabela 5 - Comparação segundo parâmetros de implementação.....	41
Tabela 6 - Comparação segundo propriedades arquiteturais dos sistemas.....	42
Tabela 7 - Definições da rede Hermes.....	50
Tabela 8 - Equivalência entre porto local do Roteador0201 e PE5.....	60
Tabela 9 – Comparação da quantidade dos CLBs do projeto pós-síntese individual respeito aos do DynoPlace.....	79
Tabela 10 - Comparação da quantidade dos CLBs do projeto pós-síntese individual em relação aos do DynoPlace ocupando a totalidade de CLBs do DRFPGA.....	82
Tabela 11 - Dados de configuração do Virtex-4	101

LISTA DE ABREVIATURAS E SIGLAS

ASIC	Application Specific Integrated Circuit
ATM	Asynchronous Transfer Mode
CAD	Computer-Aided (Assisted) Design
CLB	Configurable Logic Block
CoNoChi	Configurable Network on Chip
CuNoC	Communication unit Network on Chip
CTR	Compile-Time Reconfiguration
DCM	Digital Clock Manager
DCS	Dynamic Circuit Switching
DSP	Digital Signal Processing
DRFPGA	Dinamically Reconfigurable FPGA
DyNoC	Dynamic Network on Chip
EDA	Electronics Design Automation
EDK	Embedded Development Kit
EPUSP	Escola Politécnica da USP
E2PROM	Electrically Erasable Programmable Read-Only Memory
FIFO	First Input First Output
FLIT	Flow control digIT or Flow control unIT
FPGA	Field Programmable Gate Array
GAPH	Grupo de Apoio ao Projeto de Hardware
GPP	General-Purpose Processors
GRM	General Routing Matrix
GSEIS	Grupo de Projeto de Sistemas Eletrônicos Integrados e Software Aplicado
IC	Integrated Circuit
ICAP	Internal Configuration Access Port
IOB	Input Output Block
ISE	Integrated Software Enviromment
LME	Laboratório de Microeletrônica
LUT	Look-up Table
MHz	Mega Herz
NCD	Xilinx Native Circuit Description
NGD	Xilinx Netlist Generic Database
NoC	Network on Chip
PE	Processing Element
PLD	Programmable Logic Device
PRM	Partial Reconfigurable Module
PRR	Partial Reconfigurable Region
PUCRS	Pontifícia Universidade Católica de Rio Grande do Sul
RAM	Random Access Memory
RC	Reconfigurable Computing
RMB	Reconfigurable Multiple Bus
RMBnC	Reconfigurable Multiple Bus on Chip
ROM	Read Only Memory
RTL	Register Transfer Level
RTR	Run-Time Reconfiguration
SAF	Store And Fordward

SoC	System on Chip
SoPC	System on Programmable Chip
SDR	Sistema Dinamicamente Reconfigurável
SRAM	Static RAM
TBUF	Tristate (3-state) Buffer
TLM	Transaction Level Model
TTM	Time to Market
UC	Unidade de Comunicação
UCF	User Constraint File
UCg	Unidade de Comunicação gerenciadora de caminhos
USP	Universidade de São Paulo
VCT	Virtual Cut Through
VHDL	VHSIC Hardware Description Language
VHSIC	Very High Speed Integrated Circuit
VLSI	Very Large Scale Integration
VN	Von Neumann

1. INTRODUÇÃO

O propósito deste capítulo é apresentar os distintos aspectos que justificam a utilização dos sistemas dinamicamente reconfiguráveis e de estruturas especializadas de comunicação, em particular, redes intra-chip, para a troca de informações entre as partições fixas e reconfiguráveis. Também são detalhados a motivação, as justificativas e os objetivos do trabalho a ser realizado.

1.1.Contexto

Por muito tempo os sistemas computacionais baseados na arquitetura Von Neumann (VN)¹ e depois os processadores de uso geral (em inglês, *General Purpose Processors: GPP*) tiveram o domínio nos sistemas digitais, mas devido às limitações próprias de um sistema seqüencial nem sempre era possível alcançar os requisitos de desempenho, necessitando de aceleradores no *hardware*.

A combinação de processadores, contendo a parte em *software*, e co-processadores e periféricos, consistindo da parte *hardware* em um único circuito integrado (em inglês, *Integrated Circuit: IC*) gerou os chamados sistemas sobre silício (em inglês, *System on Chip: SoC*).

A implementação pela geração de todas as máscaras de um IC na solução conhecida como circuito integrado de aplicação específica (em inglês, *Application Specific Integrated Circuit: ASIC*), permite atingir os restritos requisitos de desempenho, porém apresenta custo excessivo, exceto para altíssimos volumes de produção. Por este fato, tem-se optado pela utilização de dispositivos programáveis como o arranjo lógico programável em campo (em inglês, *Field Programmable Gate Array: FPGA*)² que são menos custosos e permitem gerar o circuito final rapidamente. Desta forma, os FPGAs têm sido extensivamente usados em prototipagem rápida,

¹ A partir daqui os mnemônicos entre parênteses representam os nomes que lhe precedem.

² Os FPGAs foram introduzidos pela Xilinx em 1985. Modelo XC2064.

traduzindo-se na redução do tempo para o produto chegar ao mercado (em inglês, *time to market*: TTM). A Figura 1 ilustra o incremento no uso de sistemas de hardware baseados em FPGAs nos últimos anos, em detrimento dos ASICs.

Figura 1 - Sistemas hardware no FPGAs comparado com ASIC (1).

No presente trabalho os sistemas sobre silício feitos com circuitos programáveis ou configuráveis, doravante serão denominados pela sigla SoPC (em inglês: *System on Programmable Chip*).

Os FPGAs podem ser classificados (2) de acordo com a sua capacidade de configuração, conforme é mostrado na Figura 2 a começar pelo anel externo. Por definição, todos os FPGAs são programáveis ou configuráveis. Aqueles em que a programação pode ser feita apenas uma única vez, como na tecnologia *antifuse*, são os classificados especificamente como programáveis.

Os FPGAs baseados nas tecnologias E2PROM e FLASH são considerados reconfiguráveis, pois a programação pode ser alterada, assim como aqueles baseados na tecnologia SRAM, exceto que estes últimos são voláteis, perdendo a programação na falta de alimentação elétrica. O tempo de reconfiguração dos primeiros é aproximadamente três vezes mais lento do que os que se baseiam na tecnologia SRAM.

Figura 2 - Classificação dos *FPGAs* segundo sua capacidade de configuração (2).

A tecnologia SRAM permite a realização de uma configuração seletiva ou parcial, quando, de acordo com as necessidades do projetista, apenas partes do chip podem ser escolhidas para serem reconfiguradas; alguns modelos de *FPGAs* baseados nesta tecnologia como os da Xilinx e Atmel suportam uma forma especial de operação (3) na qual, a alteração dos circuitos de apenas determinadas áreas do componente pode ser feita, sem afetar o funcionamento dos circuitos programados nas demais áreas no que se conhece como reconfiguração parcial dinâmica. Estes dispositivos permitem a implementação explícita de sistemas que operam com reconfiguração dinâmica³. Segundo (4) outros termos para reconfiguração dinâmica são: reconfiguração em tempo de execução (em inglês, *run-time reconfiguration: RTR*), reconfiguração on-line (em inglês, *on-line reconfiguration*), *hardware* virtual (em inglês, *virtual hardware*) e arranjo lógico programável dinamicamente (DPGA), sendo o mais comum o uso do termo RTR. Um sistema que opere neste modo é definido neste trabalho doravante como sistema dinamicamente reconfigurável (SDR), enquanto, os *FPGAs* com reconfiguração parcial, que suportam a reconfiguração dinâmica serão denominados neste trabalho como *FPGAs* dinamicamente reconfiguráveis (DRFPGAs). A Figura 3 ilustra tal conceito, podendo-se notar que existe uma lógica ativa e uma parte da lógica inativa (que já foi ativa) cede o seu lugar para alocar uma nova lógica. As partes que nunca substituem outras ou nunca são substituídas são também chamadas de partes fixas; as outras são as partes reconfiguráveis.

³ Alguns dispositivos que suportam a reconfiguração dinâmica são os das famílias Virtex da Xilinx, AT6000 e AT40K da Atmel.

Figura 3 – Reconfiguração dinâmica (2).

A Figura 4 ilustra a idéia de que o conceito de SDR está ligado diretamente com o espaço temporal. No tempo inicial o sistema é configurado para começar sua operação (fase 1), onde se tem os módulos T_1 , T_2 , T_3 , T_4 , T_5 e T_6 . Depois de algum tempo de operação o sistema deve se reconfigurar para começar a fase 2 da operação, quando somente permanecem os módulos T_2 e T_6 , enquanto os restantes da primeira fase cedem o seu lugar aos módulos T_7 e T_8 . Em outro tempo depois, o sistema deve se reconfigurar para começar a fase 3, quando somente permanecem os módulos T_2 , T_6 e T_7 , o módulo T_8 cede o seu lugar para os módulos T_9 e T_{10} e também aloca-se o T_{11} . A figura também sugere que a duração do tempo de reconfiguração é proporcional à área a ser reconfigurada.

Figura 4 - Exemplo de um SDR (2).

Algumas vantagens do SDR são a redução de área e, conseqüentemente, consumo de energia. Como desvantagens (5), pode-se citar o potencial aumento no tempo de processamento, devido à interrupção da execução de uma parte do sistema durante a reconfiguração, além da necessidade de controle “externo” da reconfiguração, que pode ser feito por lógica ou por processador. O tempo que se precisa para que a nova reconfiguração seja lida do elemento de armazenamento e escrita na memória de configuração do FPGA e que esteja estabilizada a nova configuração do sistema é denominado latência de chaveamento.

Alguns exemplos de aplicações de SDR podem ser achados em controle de funções de sistemas de automóveis (6), nas aplicações militares como o RECOPS (7), nas aplicações de robótica (8), nos sistemas de comunicações como o RECATS (9), nas aplicações de processamento de sinais e vídeo (10), nas aplicações de precisão aritmética (11), em plataformas para o desenvolvimento de sistemas digitais (12), para citar só alguns. A Figura 5 ilustra como exemplo o SDR usado em sistemas de automóveis implementado em um SoPC.

Figura 5 - SDR implementado em um SoPC (6).

1.2.Motivação

A evolução dos circuitos programáveis como o FPGA propiciou a opção de ter os SoPCs, cuja vantagem é permitir a reconfiguração do *hardware* para que os dispositivos tenham o seu tempo de vida útil aumentado. O surgimento dos DRFPGAs permitiu o projeto de SDRs, que permite aproveitar melhor a área do dispositivo, além da redução do consumo da potência, o que é muito importante em aplicações portáteis. O aumento da complexidade dos sistemas desenvolvidos, junto com a exigência do mercado em diminuir o TTM, levou à necessidade de abordagens mais estruturadas que facilitassem a implementação de SDRs. É reconhecido que a falta de metodologias estruturadas e ferramentas é um dos grandes impeditivos para a expansão do uso de DRFPGAs em SDRs.

Nos últimos anos, uma das questões que mais tem gerado interesse no que se refere às metodologias de projeto utilizando DRFPGAs é a definição de um meio de comunicação estruturado para tratar da transferência de dados entre as partições de um SDR. A adoção de um modelo bem definido de estrutura de comunicação permite liberar o projetista de cuidar de detalhes de projeto de *bus-macros*⁴, podendo ele focar mais no projeto de lógicas adaptadoras (em inglês, *wrappers*) e da lógica de processamento, minimizando o esforço de projeto.

Algumas propostas foram apresentadas, tanto para barramentos (em inglês, *bus*) (13), (14), (15), (16) como para redes intra-chip (em inglês, *Network on Chip*: NoC) (17), (18), implementados sempre com a família Virtex II exceto em (19) onde um dispositivo Virtex 4 foi utilizado. No caso de barramentos, um dos problemas principais é que eles não permitem o paralelismo de tarefas o que faz com que a largura de banda seja compartilhada pelos módulos existentes (reduzindo o desempenho). As NoCs (20) são estruturas propostas mais recentemente baseadas na passagem de pacotes de informação entre os módulos comunicantes e têm gerado muito interesse nos últimos anos, devido ao seu potencial bom desempenho para sistemas com grande número de módulos. As NoCs suportam comunicações

⁴ Têm como finalidade garantir conexões corretas entre os módulos a serem reconfigurados com os fixos, estabelecendo canais fixos de roteamento

em paralelo além de ter uma boa escalabilidade, extensibilidade e modulabilidade tendo um potencial maior de crescimento no futuro.

Ao projetar uma interface de comunicação para os módulos inseridos dinamicamente em um SDR, deve-se definir um protocolo de comunicação que resolva problemas na transmissão de dados como a latência da reconfiguração dos módulos inseridos e o *wrapper* para adaptar a comunicação da interface com a do módulo. Os aspectos específicos de barramentos e NoCs, como, por exemplo, sobre o arbitrador da interface e algoritmos de roteamento, devem também ser considerados.

A principal vantagem do uso de NoCs para a conexão entre partições reconfiguráveis é que, por ser basicamente um sistema que se estende da mesma maneira em duas dimensões, ao contrário dos barramentos de uma dimensão, pode-se aproveitar a capacidade dos novos DRFPGAs para reconfigurar uma só parte específica (não a coluna inteira) o que diminui a latência do chaveamento de reconfiguração permitindo sua aplicação em sistemas cujas exigências de temporização sejam maiores. Tanto nos trabalhos referentes a barramentos como para NoCs, são reportados apenas os aspectos da arquitetura e implementação específica, com poucas indicações de desenvolvimento de suporte de CAD.

Apesar de ser possível encontrar na literatura técnica, diversas propostas apresentadas para a otimização do particionamento automático de circuitos em módulos reconfiguráveis e mesmo para o seu posicionamento nos DRFPGAs, não há indicações que incluam restrições para o posicionamento geométrico de blocos devido às famílias dos dispositivos ou dos *bus-macros*.

Estas tarefas, assim como a concretização de sua comunicação, seguem sendo ainda essencialmente manuais. Deve-se isto à falta de metodologias de projeto e ferramentas de CAD que simplifiquem o processo de projeto de SDRs, para que o projetista não sinta a sobrecarga de passar de um projeto plano (sem reconfiguração dinâmica) para um SDR. Este aspecto é mais evidente ainda em SDRs estruturados sobre NoCs, pelas restrições que surgem com a inclusão de estruturas fixas de comunicação.

1.3.Justificativas

O Grupo de Projeto de Sistemas Eletrônicos Integrados e Software Aplicado (GSEIS) do Laboratório de Microeletrônica (LME) da Universidade de São Paulo (USP) tem como seu interesse principal de investigação as metodologias de projeto de sistemas digitais. O grupo tem tido contribuição constante na área de reconfiguração dinâmica através do desenvolvimento de metodologias de uso geral e aplicadas a exemplos de comunicação de dados sem fio como o controlador de rede sem fio no protocolo Bluetooth (21), concebido dentro do projeto Brazil IP (22) e no desenvolvimento de uma máquina de vetores de suporte (em inglês, *Support Vector Machine: SVM*)⁵. A utilização de NoCs neste trabalho justifica-se pelo fato do GSEIS também manter um interesse em desenvolvimento neste tipo de estrutura de comunicação, explorando modelos de alto nível de abstração (23), da rede Hermes (24).

1.4.Objetivos

O objetivo geral do presente trabalho é contribuir para o desenvolvimento de metodologias e ferramentas para o projeto de sistemas dinamicamente reconfiguráveis. O presente trabalho irá focar em uma das limitações mais efetivas para a adoção da reconfiguração dinâmica: a falta de ferramentas de CAD que suportem o projeto de SDRs (inclusive os baseados em NoCs), em particular, no posicionamento dos módulos.

Este trabalho prevê o atendimento dos seguintes objetivos específicos:

- Modelar a arquitetura de um SDR baseado em NoC em uma camada de abstração de comunicação entre módulos fixos e reconfiguráveis,

⁵ Gomes Filho, Jonas. “Aplicação de Técnicas de Reconfiguração Dinâmica a Projeto de Máquina de Vetor Suporte (SVM)”, dissertação do Mestrado depositada na Escola Politécnica da Universidade de São Paulo.

usando um DRFPGA Virtex-4, compatível com a estruturação em 2D das NoCs.

- Estabelecer o modelo da comunicação entre os módulos reconfiguráveis e a estrutura de comunicação a partir de *bus-macros*.
- Implementar (em VHDL) a estrutura de comunicação entre módulos fixos e os reconfiguráveis e testá/validá-la com exemplos demonstrativos.
- Desenvolver uma metodologia automatizada de projeto de SDRs baseado em NoCs a partir de partições descritas em nível RTL.
- Desenvolver algoritmo e ferramenta de posicionamento dos módulos (fixo e reconfiguráveis) baseado em aspectos reais da família do DRFPGAs, para SDRs baseado em NoCs.
- Desenvolver um modelo de simulação de SDR baseado em NoC utilizando-se a técnica de chaveamento dinâmico de circuitos (do inglês, *dynamic circuit switching*: DCS) (3), e realizar simulações para validar o SDR.

1.5.Organização do texto

Esta dissertação está organizada da maneira descrita a seguir. No capítulo 1 a introdução, motivação, justificativa e objetivos desta dissertação foram apresentadas. No capítulo 2, alguns conhecimentos que consideramos importantes para o acompanhamento do restante da dissertação, como os fundamentos dos SDRs e as metodologias existentes para o seu projeto, além de um análise sob as interfaces de comunicação nos SoPC, são desenvolvidas. No capítulo 3 são apresentados os trabalhos publicados na literatura técnica recente sobre a descrição dos tipos de interfaces de comunicação e as características de cada uma, sendo que algumas estratégias de uso de NoCs em SDRs são discutidas e comparadas; além disso, alguns trabalhos sobre posicionamento dos módulos nos DRFPGAs são descritos. No capítulo 4, propomos um SDR com sua interface de comunicação baseada em redes intra-chip e apresentamos uma metodologia baseada nas DCS para fazer a simulação de um SDR no tempo total de execução. No capítulo 5, apresenta-se uma metodologia de particionamento, concretizada no desenvolvimento da ferramenta DynoPlace. No capítulo 6 resultados experimentais referentes à ferramenta DynoPlace são mostrados e, finalmente, no capítulo 7, conclusões e trabalhos futuros são listadas.

2. FUNDAMENTOS TEÓRICOS

O propósito deste capítulo é apresentar algumas arquiteturas de DRFPGAs, os fundamentos dos SDRs e as metodologias existentes. Na seção 2.1 se apresentam as arquiteturas de FPGAs com suporte da reconfiguração dinâmica. Na seção 2.2 se apresentam os SDR. Na seção 2.3 se faz um análise sob as interfaces de comunicação nos SoPC.

2.1.Arquiteturas de FPGAs com suporte da reconfiguração dinâmica

Os primeiros dispositivos a suportar a reconfiguração parcial foram fabricados pelas empresas National Semiconductors, Algotronix e Xilinx, gerando grande interesse no meio acadêmico, porém por não terem um adequado suporte de ferramentas de software, não obtiveram sucesso em nível comercial. Atualmente os líderes mundiais na fabricação e comercialização de DRFPGAS são as empresas Xilinx e Atmel, mas só a primeira disponibiliza dispositivos de grande porte, portanto o presente trabalho só abordará os DRFPGAs da Xilinx.

Dos dispositivos da Xilinx com capacidade para reconfiguração dinâmica e ainda com suporte, os mais antigos são os da família Virtex II (25). Apesar de diferenças substanciais em capacidade e recursos adicionais, as gerações posteriores apresentam arquitetura básica similar. Pode-se verificar pela Figura 6 que os dispositivos apresentam uma arquitetura simétrica, contendo os seguintes elementos principais:

Figura 6 - Arquitetura do VirtexII (25).

- **Blocos de entrada e saída (IOB):**
Os IOBs servem como interface entre os blocos lógico configuráveis (CLBs) e os pinos do componente que estão na periferia do circuito e incluem elementos de armazenamento, os quais podem ser configurados como *flip-flops* tipo D ou *latches*.
- **Bloco lógico configurável (CLB):**
Os CLBs são compostos de uma matriz de chaves (em inglês, *switch matrix*) para se ter acesso à linhas de roteamento de propósito geral (em inglês, *general-purpose routing*); estão divididos em quatro fatias (em inglês, *slices*), conforme mostrado na Figura 7.

Figura 7 - Bloco lógico configurável (25).

Cada fatia tem 2 blocos de memória para implementação de funções, conhecidos na literatura como *Look Up Tables* (LUTs), com 4 entradas e uma

saída que podem funcionar também como registradores de deslocamento de 16 bits ou como SelectRAM distribuída de 16 bits. Também tem lógica de *carry*, 2 registradores, multiplexadores e elementos de geração de lógica aritmética.

- Blocos de memória (RAM blocks):

Este dispositivo inclui elementos de memória de 18Kbits conhecidos como SelectRAM. Suportam configurações de porto simples e duplo. São programáveis para tamanhos de palavra de até 36 bits.

- Caminhos programáveis:

Adjacentes a cada CLB estão as matrizes de roteamento (GRM) que têm por objetivo proporcionar um caminho para que os CLBs acessem as linhas de roteamento de propósito geral (em inglês, *general-purpose routing*).

A família Virtex-II apresenta diferentes modos para a configuração dos dispositivos, como serial (Boundary Scan ou JTAG), SelectMap, mestre-serial (Master-Serial), escravo-serial (Slave-Serial) e finalmente ICAPS. A última forma, junto ao SelectMap, permite fazer um sistema auto reconfigurável⁶ no FPGA, já que provê meios de se fazer a leitura e escrita de bits de configuração (*bitstream*) no FPGA.

A diferença básica dos dispositivos da família VirtexII-Pro (25) comparado aos da VirtexII é a existência de núcleos de hardware do processador Power-PC⁷. Já a família Virtex-4 (26) é uma evolução dos dispositivos Virtex II (Virtex-4 LX/SX) e do VirtexII-Pro (Virtex-4 FX), com a introdução de lógica especial aritmética para alto desempenho, transceptores, etc.

⁶ A reconfiguração do FPGA pode ser feita do seu interior por intermédio de lógica específica, por um processador *hard core* como o Power-PC ou um *soft core* como o MicroBlaze.

⁷ Dos dispositivos VirtexII-Pro o único que não tem embutido nenhum Power-PC é o XC2VP2.

2.2. Sistemas dinamicamente reconfiguráveis (SDR).

Como explicado na seção anterior, os DRFPGAs da Xilinx estão organizados como uma matriz de CLBs e a reconfiguração de sua lógica para SDRs deve ser feita através de *bitstreams* inseridos pelo porto SelectMap ou ICAPs. Em reconfiguração parcial dinâmica, deve-se trocar somente uma parte do sistema em tempo de execução, enquanto o resto do sistema permanece sem alterações, havendo, então, as seguintes restrições:

- No caso dos DRFPGAs VirtexII e VirtexII-Pro, a menor unidade de reconfiguração é o quadro (*frame*) composto por uma coluna inteira de CLBs, conforme a Figura 8a, significando que a reconfiguração de um componente afeta todos os componentes que compartilham a mesma coluna, por este motivo são conhecidos como modelos de uma dimensão (1D).
- No caso do DRFPGA Virtex-4, o *frame* agora é e composto somente por 16 linhas de CLBs ao invés da coluna inteira, conforme a Figura 8b, por este motivo é conhecido como modelos de duas dimensões (2D).

Figura 8 - Configurações dos VirtexII (a) e Virtex-4 (b) (27).

Os DRFPGAs VirtexII, VirtexII-Pro e Virtex-4 têm a característica de serem *glitchless*, ou seja, não gera transitórios no momento da reconfiguração para os bits de reconfiguração, o que implica benefícios, já que nem todos os bits de reconfiguração do *frame* trocam de valor no momento que ela ocorrer.

Desde o lançamento dos primeiros DRFPGAs, duas metodologias para geração dos *bitstreams* parciais para projeto de SDRs foram apresentadas pela Xilinx: a do Modular Design Flow (28) e a do Early Access (27).

A metodologia Modular Design Flow é a mais antiga e estudada, utilizada para os dispositivos Virtex-II e VirtexII-Pro. Com ela, foi introduzido o conceito de *bus-macros*, que são lógicas adaptadoras de interface de módulos, reconfigurável-fixa ou reconfigurável-reconfigurável, objetivando isolá-los durante a reconfiguração. Canais fixos de roteamento são estabelecidos em linhas *tri-state* controlados por sinais *enable*. Um exemplo de utilização do *bus-macro* é ilustrado na Figura 9, conectando partições reconfiguráveis a outros módulos; é interessante notar que apenas conexões horizontais são possíveis por esta metodologia.

A metodologia modular permite dois estilos para reconfiguração dinâmica: a baseada em módulo e a baseada em diferença. Pelo primeiro, as configurações parciais da descrição dos módulos do sistema são inteiramente produzidos e carregados na DRFPGA, enquanto que, pelo segundo estilo, exploram-se as diferenças entre *bitstreams*- executado através da ferramenta FPGA Editor, pode-se obter um *bitstream* que só inclui a diferença entre partições reconfiguráveis que ocupem as mesmas fatias, o que se traduz em uma reconfiguração mais rápida.

Figura 9 – Posicionamento do *bus-macro* (28).

2.2.1. Metodologia *Early Access*

Em 2005 foi lançada a versão Partial Reconfiguration Early Access (27) que é a nova metodologia oficial da Xilinx para reconfiguração parcial dinâmica de dispositivos. As principais inovações desta metodologia em relação ao Module Design Flow são:

- É permitido que dispositivos, como o Virtex-4 ou lançados posteriormente, não utilizem a coluna inteira (aproveitem a estruturação por *frame*) para estabelecer áreas parcialmente reconfiguráveis.
- Sinais que atravessam as regiões reconfiguráveis, sempre que não interagem com sinais destas, não necessitam de conexão com *bus-macros*.
- O novo *bus-macro*⁸ fornecido pela Xilinx permite, além da transmissão horizontal de dados (da esquerda à direita e vice-versa) a transmissão de vertical (de cima para baixo e vice-versa). As conexões do *bus-macro* se dão agora pelos CLBs e não mais pelas linhas *tri-state*.
- O novo *bus-macro* (29) permite diferenciar entre os *bus-macros* estreitos e amplos, como ilustrado na Figura 10; esta classificação foca somente

⁸ Fornecido pelo fabricante Xilinx, os mesmos que agora se baseiam nos LUTs em lugar dos buffers tristates, já que estes últimos não são incluídos nos novos dispositivos como o Virtex-4.

naqueles CLBs que são interligadas pelos *bus-macros*. Os *bus-macros* estreitos são aqueles que interligam 2 CLBs que ficam juntos e os amplos são os que interligam 2 CLBs que ficam separados por 2 CLBs.

Figura 10 - *Bus-macro*: estreito (a) e amplo (b) (29).

- O novo *bus-macro* (29) também permite aproveitar os CLBs não utilizados entre o *bus-macros* amplos, para lógica ou para outros *bus-macros*, a Figura 11 ilustra a utilização do espaço aproveitado, neste caso, para aninhar *bus-macros*.

Figura 11 - Três *bus-macros* aninhadas (29).

A vantagem de se ter *bus-macros* amplos é que isto permite considerar também o caso no qual os módulos a se comunicarem não sejam adjacentes.

A metodologia Early Access apresenta um roteiro completo, que necessita ainda de grande esforço manual, para a geração dos bitstreams do SDR em projeto. Na Figura 12 as principais etapas da metodologia são ilustradas e, a seguir, breves explicações são dadas sobre elas.

Particionamento do projeto: Esta etapa o projeto do sistema digital é dividido manualmente num conjunto de módulos, sendo alguns do tipo dinâmico e os

restantes fixos ou do tipo estático. Como os módulos dinâmicos reconfiguráveis, denominados na metodologia como módulos parcialmente reconfiguráveis (em inglês, *Partial Reconfigurable Modules: PRMs*), só operam numa fase específica do tempo de operação, então são posicionados nas regiões parcialmente reconfiguráveis (em inglês, *Partial Reconfigurable Regions, PRRs*). O projeto do sistema deve incorporar junto a um arquivo-topo (TOP) as descrições dos elementos *bus-macros*, que compõem o conjunto BM.

Figura 12 - Etapas da Metodologia Early Access

Síntese individual dos módulos: Esta etapa todos os módulos, fixos e reconfiguráveis, devem ser sintetizados individualmente pela ferramenta ISE. De acordo com a metodologia, os módulos devem ser sintetizados como módulos e não como circuito, ou seja, com o cuidado de não estarem conectados aos blocos de IO.

Posicionamento manual dos módulos feito com o Floorplanner: Esta etapa, utiliza-se a ferramenta Floorplanner⁹ da Xilinx para fazer o posicionamento manual no DRFPGA de todos os módulos estáticos e de um módulo dinâmico para cada uma das PRRs. Além disso, posicionam-se os elementos de relógio (pino do relógio, DCM e BUFG) e os sinais de entrada e saída do projeto junto aos pinos do DRFPGA. Esta ferramenta permite gerar um arquivo de restrições de usuário que contém informações de posição das regiões onde foram posicionados os módulos, e também dos pinos de entrada, saída e de relógio.

Roteamento dos módulos estáticos e PRMs: Posicionamento e roteamento dos elementos posicionados são feitos nas regiões fixas e reconfiguráveis respectivamente com as funções dos *scripts* da Metodologia Early Access da Xilinx. Também, a informação do posicionamento do conjunto BM (do UCF) é usada para ao fazer o roteamento.

Geração dos *bitstreams*: Esta etapa, feita com os *scripts* da Metodologia Early Access, as partes estáticas são fundidas com cada um dos módulos a ser incluído inicialmente nos PRRs para gerar o *bitstream* de configuração inicial do DRFPGA. De forma similar, são criados *bitstreams* parciais de cada PRM que serão ativados no tempo alocado. O controle do disparo de cada *bitstream correspondente a um PRM* é de responsabilidade do projetista que deve criar o circuito associado.

⁹ A ferramenta Floorplanner é uma interface gráfica que auxilia no posicionamento dos módulos do sistema digital e inclui as informações da estrutura dos dispositivos da Xilinx.

2.2.2. Metodologia para simulações de SDR.

2.2.2.1. Chaves de Isolação

Para a realização de simulações em SDRs, utiliza-se a técnica chamada chaveamento dinâmico de circuitos (em inglês, *dynamic circuits switching*: DCS) (3), (30), (31). A técnica consiste em utilizar chaves de isolação para que as tarefas dinâmicas ativas sejam conectadas com os módulos estáticos (ou outros dinâmicos) do projeto e poder também excluir (da simulação) as tarefas que sejam inativas. O método também tem a capacidade de modelar o intervalo de reconfiguração o que é fundamental na simulação de um sistema com reconfiguração dinâmica.

A Figura 13 ilustra o esquema de um SDR, onde os módulos A e B são módulos estáticos; o sistema tem uma única região reconfigurável X, o qual possui N variações que são mutuamente exclusivas já que só uma delas pode estar ativa (enquanto as outras N-1 estão inativas) num dado tempo específico. As chaves permitem isolar do sistema os N-1 módulos inativos e permitem o funcionamento do módulo ativo, assim como garantem o funcionamento de cada uma delas e modelam os estados: inativo, ativo e de transição para sistemas com reconfiguração dinâmica.

Como pode se visto na figura, cada chave de isolação possui uma entrada, uma saída e uma variável de controle $Ctrl_X_i$ e o funcionamento é dada na Tabela 1.

Tabela 1 – Tabela de verdade da chave de isolação

Variável de Controle	Saída	Estado de conjunto
0	Entrada	Ativo
1	X	Transição
Z	Z	Inativo
X	Z	Inativo (adotado)

Figura 13 – Estrutura de simulação utilizando DCS (31).

A chave de isolamento, quando está ativa (variável de controle = 0), conecta a entrada à saída, enquanto que, se a variável de controle é 1, a saída tem o valor desconhecido X , próprio do estado de transição na reconfiguração. No estado inativo (variável de controle = Z), a saída do módulo estará em alta impedância para representar que o mesmo encontra-se desconectado do sistema. Os trabalhos das referências (31) e (32) adotaram um estado inativo que está presente quando a variável de controle tem o valor de X .

Ao instanciar as chaves de isolamento na realização da simulação do SDR, precisa-se incluir também uma unidade de controle para o funcionamento das mesmas como visto na Figura 13. Note-se que nesta unidade se terá como parâmetro o tempo de reconfiguração para os módulos reconfiguráveis durante os ciclos de transição.

A Figura 14 apresenta um exemplo simples de operação de um sistema com DCS, onde dois módulos reconfiguráveis representam duas operações diferentes de duas entradas para os dados e uma saída; desta forma, cada um dos módulos precisa de três chaves de isolamento.

Figura 14 - Diagrama de blocos do exemplo de operação DCS.

2.2.2.2. Computação do Tempo de Reconfiguração

O tempo de reconfiguração de um módulo é exatamente o tempo necessário para ingressar o correspondente *bitstream* de reconfiguração nos portos SelectMap ou ICAP de um DRFPGA, na frequência especificada pelo fabricante. De seu lado, o tamanho do *bitstream* depende do número de *frames* do dispositivo a serem reconfigurados. Os dados necessários para o cálculo do tempo de reconfiguração é específico da família de DRFPGA utilizada- no Apêndice B, se apresenta um exemplo de dados relevante para dispositivos Virtex-4LX.

Para o cálculo do tempo de reconfiguração, necessita-se saber basicamente qual é o tempo de carregamento do *bitstream*, o que depende de seu tamanho. A eq.(1), adaptada da referência (32) permite ao projetista obter o tempo de reconfiguração (em μs) para cada uma das PRMs nas PRRs. Deve-se tomar cuidado para que não se confunda o termo *frame*, usado na eq.(1), que refere-se a *frames* da memória de configuração e é diferente dos *frames* do DRFPGA ou de reconfiguração mencionados na seção 2.2. Os primeiros referem-se aos bits de

memória de configuração dos recursos da DRFPGA (que estão proximamente relacionados às palavras do *bitstream*) e os últimos é a menor área reconfigurável realizada pela reconfiguração dinâmica. A equação trata de um retângulo a ser reconfigurado, sendo necessária a determinação do tamanho do frame de *bitstream* (em bytes) correspondente à reconfiguração da largura de um frame de reconfiguração na altura do retângulo, o número de colunas de frames de reconfiguração, o número de bytes aceito pelo porto de configuração e a frequência de reconfiguração .

$$Tempo_{RECONFIG} = \frac{Bitstream}{Num_{BytesPortoConfig} * Freq_{RECONFIG}} \quad (1)$$

Deve-se notar que a correspondência entre o número de palavras do *bitstream* usadas na reconfiguração e o tamanho dos *frames* de reconfiguração é bem próxima, mas não direta; portanto, é sempre aconselhável ser mais conservador em relação ao tempo obtido, dando uma pequena margem de segurança.

2.3. Interfaces de comunicação.

Nos últimos anos tem havido um interesse crescente nas redes intra-chip, para o estabelecimento de comunicação em SoPCs, devido ao seu potencial bom desempenho em sistemas com grande número de módulos, podendo suportar comunicações em paralelo, além de apresentar boa escalabilidade, extensibilidade e modularidade. Nesta seção, alguns aspectos básicos de NoCs serão apresentados com o intuito de facilitar o entendimento do texto para o leitor pouco familiarizado com o assunto. Para apresentações mais completas sobre NoCs, os textos de (33), (34) são recomendados.

2.3.1. Redes Intra-chip

Os SoCs, desde o seu surgimento, têm suas interfaces de interconexão baseadas majoritariamente em barramentos, mas, devido a suas limitações como escalabilidade, reusabilidade e eficiência de consumo de potência (20), as implementações baseadas nas NoCs têm sido vistas como uma solução de um potencial maior de crescimento no futuro. As NoCs são estruturas de comunicação em SoCs baseadas em transmissão de dados em forma de pacotes entre os módulos do chip. Como vantagens principais, podem-se listar escalabilidade, modularidade e capacidade de conectar módulos em paralelo e o seu potencial bom desempenho para sistemas com grande número de módulos (20), (34).

Para o projeto de NoCs, algumas características ou parâmetros básicos existentes em todas elas (35), (23) devem ser considerados. São eles, a topologia, os algoritmos de roteamento, mecanismo de roteamento, controle de fluxo e arbitragem.

2.3.2. Topologia

A topologia estabelece a forma de interconexão entre os roteadores na rede; as topologias existentes pertencem a dois conjuntos: redes diretas e redes indiretas. As redes diretas são aquelas nas quais os roteadores estão conectados a um módulo do sistema através de ligações fixas, ou seja, ponto a ponto, o que é equivalente a dizer que a interconexão é estática. Desta forma, os pares roteador-módulo são definidos como nós. Pode-se citar como exemplos de topologias estáticas: grelha, toróide 2D, cubo 3D, cubo 4D e totalmente conectada, as que são ilustradas na Figura 15.

Figura 15 - Topologias diretas: (a) Grelha 2D; (b) Toróide 2D; (c) Cubo 3D; (d) Cubo 4D; (e) Totalmente Conectada.

Apesar de a topologia ideal ser totalmente conectada, uma vez que a distância entre os nós é minimizada, o alto grau de interconexão torna-a inviável para sistemas de maior envergadura, pela dificuldade no roteamento.

As redes indiretas são aquelas nas quais não é obrigatório o par fixo de roteador e módulo para os nós. Quando uma conexão entre dois módulos se faz necessária a rede adapta-se dinamicamente. Uma das redes indiretas mais destacadas é a matriz de chaveamento (em inglês, *crossbar*), na mesma configuração utilizada nas companhias telefônicas há décadas. Um exemplo de rede *crossbar* é dado na Figura 16.

Figura 16 - Topologia indireta crossbar.

2.3.3. Algoritmos de roteamento

O algoritmo elege o caminho a ser percorrido pelos pacotes na rede entre dois nós. Para garantir a transferência dos dados, deve-se impedir as ocorrências de *deadlock*, *livelock* e *starvation* (33) (29). O *deadlock* é uma situação onde o pacote está aguardando por um evento (solicitações de acesso a recursos de comunicação e armazenamento) que nunca poderá acontecer, devido à dependência circular dos recursos; um exemplo ocorre para um pacote que não pode ser transmitido antes de algum outro e este segundo não pode ser transmitido antes do primeiro. O *livelock* é uma configuração da rede onde os pacotes ficam circulando indefinidamente sem atingir os respectivos destinos. O fenômeno de *starvation* é uma situação onde o pacote está permanentemente bloqueado pelo tráfego intenso e os recursos requeridos por ele sempre são concedidos para outros pacotes.

Os algoritmos de roteamento são classificados segundo o local de implementação, na fonte ou no roteador. Quando implementado na fonte, é conhecido como roteamento fonte, ou seja, o elemento processante gerador do pacote determina o caminho a ser percorrido incluindo esta informação no pacote, o que aumenta o *overhead* do pacote. Quando implementado no roteador, é conhecido como roteamento distribuído, ou seja, cada roteador determina se o pacote deve ser enviado para o seu elemento processante local ou para outro roteador na sua vizinhança.

Os algoritmos também são classificados segundo a sua capacidade de se adaptarem às condições da rede no momento da transmissão. Os determinísticos são os que têm um caminho fixo definido entre a origem e o destino do pacote, sem importar-se com as condições momentâneas da rede. Os adaptativos são os que dependem das condições momentâneas da rede, sendo que, então, o caminho a ser percorrido por um pacote não se pode ser determinado de antemão. Esta característica faz com que possam ser utilizados para distribuir o tráfego de um modo mais homogêneo. Outro ponto a ser considerado, porém, é que este algoritmo tem uma maior complexidade comparada aos determinísticos, precisando, portanto,

de mais recursos de hardware. Um algoritmo adaptativo muito difundido é o XY por ser livre de *deadlock* (33).

2.3.4. Mecanismo de Roteamento

O mecanismo de roteamento define a forma de troca de dados entre os roteadores da rede. Os mecanismos existentes são os roteamentos de circuito e de pacote. O roteamento de pacotes se caracteriza por não ter um caminho determinado na transmissão de dados e tem os modos de roteamento: *store-and-forward* (SAF), *virtual-cut-through* (VCT), e *wormhole*.

No roteamento de pacotes, o cabeçalho de cada pacote informa a cada roteador qual é o seu destino. No caso do modo SAF, um pacote deve ser armazenado na memória (*buffer*) e a informação do destino deve ser analisada para determinar por qual canal de saída será enviado o pacote. Gera-se assim uma alta latência¹⁰ para a entrega de cada pacote, implicando que os *buffers* dos roteadores devem ser dimensionados para o tamanho máximo do pacote. O modo VCT objetiva aperfeiçoar o SAF, fazendo com que o roteador atual envie o pacote no momento que o próximo roteador garante que possa receber todo o pacote, diminuindo, assim, a latência; caso a rede esteja sobrecarregada, o funcionamento do VCT é similar ao SAF. O modo *wormhole* procura aperfeiçoar o VCT; baseando-se que, no caso de o próximo roteador não garantir o recebimento do pacote, a atual teria que armazená-lo, então, se o pacote é dividido em unidades menores denominadas *flits*, consegue-se diminuir o tamanho dos *buffers*. Uma desvantagem é que as informações sobre o direcionamento só estão no *flit* do cabeçalho então os demais *flits* que são elementos do pacote devem seguir o mesmo caminho. Se o *flit* do cabeçalho não puder avançar devido à contenção de recursos, todos os *flits* são bloqueados ao longo do caminho até que o caminho seja liberado.

¹⁰ A latência por pacote no SAF é igual à quantidade das chaves por onde o pacote deve passar por o tempo de transferência do pacote entre roteadores.

O roteamento de circuito¹¹ permite criar um caminho entre a fonte e o destino dos dados antes da transmissão dos mesmos começar. O procedimento começa com a transmissão de um cabeçalho que percorre a rede até atingir o destino e o caminho formado fica dedicado exclusivamente para a transmissão dos dados até que um finalizador desfaz o caminho. A desvantagem é o tempo que se requer para estabelecer um caminho entre a fonte e o destino, mas pode ser adequado quando o tempo de estabelecimento do caminho é mínimo em comparação com o tempo de transferência dos dados e quando se tem um fluxo contínuo de dados. Soares (35) concluiu que esta técnica é adequada para sistemas onde as comunicações entre os seus elementos processantes caracterizam-se por ser longas, mas com pouca frequência.

2.3.5. Controle de Fluxo

Quando os pacotes percorrem a rede, eles competem pelos canais e pelos *buffers* para atingir o endereço de destino. Quando ocorre uma condição em que um pacote não pode continuar percorrendo a rede (situação de colisão), o controle de fluxo deve ter uma política para tomar decisões de como solucionar este problema, entre os quais, o descarte do pacote, o bloqueio onde está o armazenamento temporário ou desvio para outro caminho. No caso das redes intra-chip o controle de fluxo se faz no nível de enlace, de tal forma que todos os portos do roteador têm um *buffer* para o armazenamento do pacote entrante. Caso o *buffer* do porto correspondente estiver cheio, o transmissor deve manter a informação armazenada no *buffer* local até que os recursos fiquem disponíveis para o término da transmissão. O controle de fluxo é o encarregado de bloquear a saída de dados do transmissor até que o receptor envie um sinal de controle ao transmissor para indicar se naquele momento pode-se ou não receber nova informação.

Um protocolo bastante utilizado para o controle do fluxo é o *handshake*; outros utilizados são os baseados em canais virtuais e os baseados em créditos. No

¹¹ Esta técnica é baseada nas redes de telefonia.

caso do *handshake*, o transmissor informa ao receptor a intenção de enviar um pacote e este último deve confirmar se o *buffer* tem espaço para armazenar a informação. No caso de protocolo baseado em créditos, o receptor informa ao transmissor o espaço disponível no *buffer* de entrada, o que é computado no transmissor como crédito e, segundo a quantidade de créditos disponíveis, ocorre ou não a transmissão de dados. No caso dos canais virtuais, o *buffer* físico é dividido logicamente em diversos canais que funcionam como pequenas filas, o que permite que transmitam os seus pacotes sem se importar com a situação das outras filas.

2.3.6. Arbitragem

A arbitragem determina por qual porta de saída será transmitido um pacote, isto é, utiliza um algoritmo para definir qual porta será servida em um determinado momento, tendo por objetivo resolver conflitos caso uma mesma saída seja solicitada simultaneamente. Uma má escolha do algoritmo pode produzir o fenômeno de *starvation*.

A arbitragem pode ser centralizada ou distribuída. No caso da centralizada um só módulo faz as funções de arbitragem e de algoritmo de roteamento. Na distribuída existe um módulo árbitro para cada saída do roteador. Algumas políticas de escalonamento são utilizadas para determinar qual entrada será servida (para ambos os tipos de arbitragem), sendo as mais comuns prioridade estática, prioridade dinâmica, escalonamento por *deadline*, *first-come-first-served* (FCFS), *least recently served* (LRS), *round-robin* (RR), entre outros.

3. TRABALHOS RELACIONADOS

O propósito deste capítulo é apresentar os trabalhos publicados na literatura técnica recente que trazem subsídios para discussão introduzida nesta dissertação. Na seção 3.1 faz-se a descrição dos tipos de interfaces de comunicação e as características de cada uma. Algumas estratégias de uso de NoCs em SDRs são revisadas na seção 3.2, enquanto que na seção 3.3, elas são discutidas e comparadas. Finalmente na seção 3.4 são apresentados alguns trabalhos sobre posicionamento dos módulos nos DRFPGAs.

3.1. Tipos de interfaces de comunicação nos SDR

Os projetos de sistemas reconfiguráveis desenvolvidos nos DRFPGAs têm adotado na sua maioria uma abordagem *ad-hoc*, construindo *bus-macros* específicos para a comunicação entre módulos reconfiguráveis e fixos (ou outros reconfiguráveis). Devido à complexidade e ao alto nível de detalhamento associado ao projeto de *bus-macros* específicos, soluções estruturadas têm sido propostas nos últimos anos; primeiramente, utilizaram-se os barramentos como interface de comunicação entre os módulos (13), (14), (15) e (16), porém, mais recentemente, têm surgido trabalhos adotando NoCs (17), (18) e (19).

Procuraremos fazer uma análise comparativa entre estas estratégias de interfaces (36) e, para isso, selecionamos algumas propriedades arquiteturais e parâmetros de desempenho que são apresentados nas Tabela 2 e 3 respectivamente. Em relação a barramentos e NoCs, várias diferenças se destacam e são reconhecidos na literatura técnica:

Tabela 2 – Propriedades arquiteturais de comunicação nos sistemas reconfiguráveis.

Parâmetro	Definição
Flexibilidade	A capacidade de a interface de comunicação suportar distintos padrões de comunicação, sem afetar o seu desempenho.
Escalabilidade	A capacidade de a interface de comunicação ser ampliada ou reduzida, sem afetar o seu desempenho no tempo de execução do sistema.
Extensibilidade	A capacidade de a interface de comunicação ser estendida e poder ser usada em sistemas que precisam de uma área maior de trabalho. Não é necessário manter o desempenho do sistema só a taxa de transferência no tempo de execução do sistema.
Modularidade	A propriedade de a interface de comunicação poder ser dividida, em submódulos, o que permite, em certa medida, a reusabilidade dos mesmos.

Tabela 3 - Parâmetros de desempenho da interface de comunicação nos sistemas reconfiguráveis.

Parâmetro	Definição
Latência	O número de ciclos de relógio em que o dado atrasa ao passar por um elemento da rede.
Largura de banda (em inglês, <i>bandwidth</i>)	A quantidade de dados que o caminho da rede pode transportar fisicamente por unidade de tempo.
Taxa de transferência (em inglês, <i>throughput</i>)	A quantidade de dados transportada de maneira efetiva pelo caminho da rede por unidade de tempo.
Paralelismo	A quantidade de dados independentes que podem ser transportados simultaneamente em uma rede.

- **Barramento**

Apresenta vantagens como flexibilidade, extensibilidade e a facilidade para ser projetado. Uma desvantagem está na sua baixa escalabilidade, já que a largura da banda é compartilhada por todos os módulos do sistema. Outra desvantagem consiste nos longos caminhos de comunicação que, no caso do FPGA, produzem problemas para o roteamento interno, além de um consumo maior de potência. Na presença de muitos módulos, ocorre um aumento na latência das respostas do árbitro do barramento às solicitações recebidas.

- **NoC**

A maior vantagem do NoC refere-se à sua capacidade de suportar comunicações em paralelo, além de ter uma boa escalabilidade em comparação com os barramentos, devido ao suporte a uma maior largura de banda. A modularidade das NoCs permite o reuso de seus elementos de projeto. O uso de roteamento local nas NoCs, devido a sua estrutura dividida em segmentos, permite ter um consumo menor de potência e uma maior frequência de relógio em comparação com barramentos. A grande

desvantagem das NoC é o excessivo uso de área, uma vez que utiliza muitos roteadores.

3.2.Propostas de interfaces de comunicação para SDR

Nesta seção, são apresentadas as propostas de interfaces de comunicação para SDR que consideramos mais relevantes. São detalhadas a estrutura usada, além da data de publicação do trabalho correspondente e o dispositivo usado, já que os avanços da tecnologia dos DRFPGA são importantes na definição da modelagem de uma interface de comunicação.

Apesar de Palma et al (13), Huebner et al (14) e Walder et al (15), com trabalhos focando em barramentos, serem os precursores do uso de meios de comunicação estruturados em SDRs, as estratégias apresentadas por seus autores trazem, quando comparados a NoCs, as desvantagens listadas na seção 3.1, particularmente, quando o número de módulos processantes é grande. Trabalhos mais recentes têm focado na utilização de NoCs como meio de comunicação em SDRs, estratégia que seguimos nesta dissertação. Por esta razão, nesta seção nos deteremos apenas nos trabalhos encontrados na literatura técnica sobre SDRs baseados em NoCs e não nos barramentos, cujo detalhamento será deixado no Apêndice C.

3.2.1.Proposta de Ahmadinia et al.

O sistema proposto por Ahmadinia, o RMBoC (Reconfigurable Multiple Bus on Chip) (37) é uma proposta de 2005, validado em um DRFPGA VirtexII 6000, para ter uma interface de comunicação entre módulos reconfiguráveis, denominados elementos processantes (em inglês, *processing elements*, PEs). A Figura 17 ilustra o RMBoC, onde a interface de comunicação fica na parte superior do dispositivo e é

composta de módulos denominados *crosspoints*. A faixa azul representa os *bus-macros* que isolam os elementos a serem reconfigurados. Os PEs têm suas áreas fixas, o que pode gerar um gasto desnecessário de recursos, uma vez que a área é definida de forma a acomodar a maior lógica processante. Esta estratégia vem do fato de que DRFPGAs de modelo 1D (definido na seção 2.1) são utilizados e faixas verticais inteiras devem ser reconfiguradas.

A idéia para a realização dos *crosspoints* nesta proposta foi de se modificar a estrutura de comunicação RMB (Reconfigurable Multiple Bus) (38) para ser utilizado como uma NoC, ou seja, passar informação em forma de pacotes.

Figura 17 - Diagrama de blocos do sistema RMBBoC (37).

A Figura 18 ilustra a estrutura do *crosspoint* que é composta pelas portas à esquerda e à direita para a transmissão horizontal dos dados e pela entrada local para que o módulo se conecte com o barramento horizontal. Seguindo a denominação proposta pelos autores de (38), estes portos representam os pontos de acesso do *crosspoint*. Para cuidar do encaminhamento dos pacotes, o *crosspoint* contém também um controlador e FIFOs, estes para armazenarem dados de transmissão.

Figura 18 - Estrutura do crosspoint (37).

Como os *crosspoints* armazenam dados de transmissão, é necessário que se garanta que os valores não sejam apagados na reconfiguração (desvantagem de se usar o antigo DRFPGA de modelo 1D). Para isto, Block-Rams que estão distribuídos em seis regiões do dispositivo são utilizados para guardar os dados, significando que não se pode ter mais de 6 PEs no RMBoC.

Outro problema nesta estratégia é a deterioração do desempenho da interface de comunicação pela paralisação da transmissão de dados entre PEs distantes durante a latência de chaveamento. Por exemplo, no caso de uma transmissão de dados do PE3 ao PE1 e, no entanto, desejar-se a reconfiguração de PE2, então o *crosspoint* 2 terá que ser reconfigurado, interrompendo a comunicação.

3.2.2. Proposta de Bobda et al.

O sistema de Bobda (17) é o DyNoC (Dynamic Network on Chip), uma proposta de 2004, validado em dois DRFPGA: VirtexII 1000 e VirtexII 6000. A interface de comunicação do sistema é ilustrada na Figura 19a e consiste em ter um sistema baseado em uma NoC, rede grelha. A estrutura de comunicação consiste dos seguintes elementos: nós terminais, ou seja, de processamento (o PE da Figura 19a), e nós de roteamento ou roteadores. O roteador tem um controlador, 6 FIFOs, trata dados de 32 bits e endereçamento de 4 bits, além de possibilitar 5 conexões, sendo: uma para o PE e as outras quatro, para a comunicação com os outros roteadores. Tais detalhes do roteador são ilustrados na Figura 19b.

Figura 19 - Interface de comunicação do sistema DyNoC (a). Estrutura do roteador (b) (29).

Como ilustrado na Figura 20, a idéia é alocar os módulos reconfiguráveis (os PEs, neste caso C_1 , C_2 , C_3 e C_4) de tal forma a estarem sempre rodeados por roteadores. Por exemplo, antes de alocar C_1 se desativaria os 2 roteadores localizados na área a ser ocupada por ele. O componente envia um sinal de desativação aos roteadores vizinhos para indicar que não enviem pacotes na sua direção. A comunicação do componente é por intermédio do roteador que se aloca no canto superior direito do mesmo.

Figura 20 – Alocação dos components na DyNoC (29).

A comunicação se faz por meio do modelo de envio de pacotes (em inglês, *packet-based*) e o algoritmo de roteamento S-XY (Surrounding XY) pertence aos conjuntos dos algoritmos adaptativos.

Um empecilho prático, similar ao da seção 3.2.1 (Ahmadinia), foi a configuração do sistema ser de tipo 2D e a implementação ser em um DRFPGA do tipo 1D. Isto resulta em deterioração do desempenho, pois, ao se fazer a reconfiguração de um dos módulos, os que estejam na mesma coluna (acima ou abaixo) também devem ser reconfigurados, significando a paralisação de alguma comunicação existente.

3.2.3. Proposta de Pionteck et al.

O sistema de Pionteck, o CoNoChi (Configurable Network on Chip) (18), é uma proposta de 2006, validado em um DRFPGA VirtexII-Pro1000. A interface de comunicação consiste em ter um sistema baseado em uma NoC onde não somente se faz uma reconfiguração dos módulos (tal qual no caso da DyNoC), como também, da própria NoC.

A Figura 21 ilustra a interface de comunicação do sistema CoNoChi, limitada a uma grelha de 5x5, e se percebe que a mesma tem a mínima quantidade necessária de nós de roteamento para que os módulos correspondentes possam se comunicar.

Figura 21 - Interface de comunicação do sistema CoNoChi (18).

O detalhe de um nó de roteamento é ilustrado na Figura 22, onde se observa uma unidade de controle de nome NoC/Switch Control e blocos de entrada contendo FIFOs, uma unidade de controle e um registrador. Para a saída de dados, utilizam-se multiplexadores.

Figura 22 - Estrutura do nó de roteamento do sistema CoNoChi (18).

A necessidade de se reconfigurar também a NoC pode ser ilustrada na Figura 23. Pode-se observar que, no caso de se trocar o módulo M_3 por outros menores (M_4 e M_5), é preciso adicionar outro nó de roteamento para que o M_4 tenha acesso à rede.

Figura 23 - Mapeamento dinâmico de módulos e do nodo de roteamento (18).

Uma vantagem alegada do CoNoChi em relação ao DyNoC é, dado o uso de elementos de NoC reconfiguráveis, necessita-se de uma quantidade mínima de elementos de roteamento, levando a uma redução da área e da latência da rede.

O algoritmo de roteamento usado é determinístico, então, antes de se fazer a reconfiguração dinâmica, é necessário atualizar as tabelas de roteamento nos roteadores. No caso do exemplo da Figura 23, a atualização seria feita nos roteadores 1 e 2, para os dados serem redirecionados pelos roteadores 3, 4 e 5, e desabilitar o enlace entre os roteadores 1 e 2. Depois da reconfiguração, deve-se iniciar o roteador 6 e estabelecer o enlace entre os roteadores 1, 6 e 2.

A arquitetura foi implementada como um sistema 2D em um DRFPGA VirtexII-Pro (dispositivo 1D)¹². Um problema derivado disto é que os nós de roteamento que estejam na mesma coluna de um módulo a ser reconfigurado não podem acessar as suas tabelas de roteamento caso uma reconfiguração esteja ocorrendo em uma de suas colunas de CLBs, uma vez que os Distributed RAM não podem ser acessados no momento da reconfiguração. A solução para contornar o problema foi usar a técnica de roteamento *virtual-cut-through* (VCT) ao invés da técnica *wormhole*, o que leva a ter maiores elementos de armazenamento nos roteadores. Como consequência, o sistema apresenta alguns problemas para estabelecer comunicação entre os módulos durante a reconfiguração deteriorando o desempenho da interface de comunicação.

3.2.4. Proposta de Jovanovic et al.

O sistema de Jovanovic, o CuNoC (Communication unit Network on Chip) (40), é uma proposta de 2007 validada em um DRFPGA VirtexII. O sistema foi projetado novamente em 2008 (19) em um DRFPGA Virtex-4. Como para o CoNoChi, o CuNoC permite a reconfiguração da NoC e dos módulos de processamento.

A NoC é composta de unidades de comunicação (UC) que são nós de roteamento de 4 conexões (norte, sul, oeste e leste). A Figura 24 ilustra a estrutura de uma UC; ela não tem *buffers* nas entradas e saídas, tem a unidade de

¹² Na seguinte referência eletrônica 39. DynaCORE. Disponível em <http://www12.informatik.uni-erlangen.de/sprrr/colloquium06/DynaCore_Muenchen.pdf>. Acesso em: 21 de Fevereiro de 2009 é mencionado que o CoNoChi já foi implementado em um dispositivo Virtex-4 FX, entretanto não foi encontrado nenhum artigo que documente isto.

roteamento e arbitragem, que segundo suas políticas, alocam os dados entrantes nos *buffers* internos sendo também a que indica a saída correspondente. Outros componentes da UC são multiplexadores e demultiplexadores.

Figura 24 - Estrutura interna de uma UC (19).

O algoritmo de roteamento é o S-XY e o modo de roteamento usado é o *store-and-forward* (SAF). Como a UC não tem *buffers* nas entradas e saídas, para se evitar congestionamentos e gargalos, utiliza-se uma política de prioridade para a direita (como na intersecção de duas ruas), onde o pacote entrante em uma direção e que não tem outros pacotes entrando à sua direita, tem a prioridade. No caso em que os pacotes chegam ao mesmo tempo pelos 4 portos, a prioridade é a estabelecida pelo projetista.

No caso de gargalo, as UC estão ocupadas; então, para solucionar o problema, aproveita-se o fato que sempre uma UC está rodeada pelas unidades de comunicação gerenciadora de caminho (UCg). O algoritmo descrito em (19) propõe basicamente trocar o estado das UCgs a uma UC “livre” para permitir o transporte dos pacotes. A Figura 25 ilustra a alocação das UCgs em relação às UCs.

Figura 25 - Posições de UCs e de UCg (19).

Para inserir um módulo reconfigurável, deve-se primeiramente ter uma estimativa da área do módulo relativo à da UC. No caso de ser menor ou igual à da UC, simplesmente o módulo entra no lugar da UC. No caso de ser maior, o controlador de reconfiguração procura um conjunto de UCs maior ou igual à área do módulo para que possa ser inserido. Também o controlador de reconfiguração informa a todas as UCs e UCgs quais serão eliminadas, aguardando o tempo necessário para que estas liberem os seus *buffers* internos e troquem o seu estado para a condição de módulo. A Figura 26 ilustra as três etapas para a formação de uma NoC.

Figura 26 - As três etapas para fornecer uma CuNoC (19).

Os resultados de taxa de transferência e de frequência de operação mostrados em (19), para dados de 8 até 512 bits, são muito superiores no caso do DRFPGA Virtex-4 que no Virtex-II, o que é consequência de o primeiro ser um modelo 2D. Também, como decorrência da implementação, o sistema não apresentou problemas para estabelecer comunicação entre os módulos durante a reconfiguração.

3.3. Discussão das propostas de interfaces de comunicação revisadas.

Para os sistemas propostos na seção 3.2, incluindo-se os SDRs baseados em barramento, foram realizados uma análise e um estudo comparativo com foco em suas propriedades arquiteturais, definidas na Tabela 2. Para organizar este trabalho, os seus parâmetros de projeto e de implementação foram sistematizados, sendo os diferentes atributos anotados nas Tabela 4 e Tabela 5, respectivamente. Os valores das tabelas foram preenchidos a partir de informações dos artigos correspondentes e, infelizmente, a Tabela 5 ficou incompleta, pois, nem todos os dados são fornecidos.

Tabela 4 – Comparação segundo parâmetros de projeto.

Proposta	Tipo	Topologia	Tamanho do módulo	Tipo de Comunicação	Largura em bit	Camadas de protocolo
Palma	Bus	Arranjo 1D	Fixo	Comutação de circuitos	1	1
Huebner	Bus	Arranjo 1D	Fixo	Comutação de circuitos	8	1
Walder	Bus	Arranjo 1D	Variável só na largura	Comutação de circuitos	8	1
RMBoc	Bus	Arranjo 1D	Fixo	Comutação de circuitos	1 – 32	1
DyNoc	NoC	Arranjo 2D	Variável (2D)	Pacotes	8 – 32	1
CoNoChi	NoC	Arranjo 2D	Variável (2D)	Pacotes	8 – 32	3
CuNoC	NoC	Arranjo 2D	Variável (2D)	Pacotes	8 – 512	1

Tabela 5 - Comparação segundo parâmetros de implementação.

Proposta	Latência (ciclos)	Frequência (MHz)	Área (Slices)	Dispositivo
Huebner (4 mód ent 32 bits, saída 16 bits)	1	66	296	VirtexII
RMBoc (32 bits)	Fase inicial: 8 Estabelecer a conexão:1	94	5084	VirtexII
DyNoc (roteador, 32 bits)	1	75	1480	VirtexII
CoNoChi (roteador, 32 bits, grelha de 5x5)	5	73	1640	VirtexII-Pro
CuNoC (roteador, 32 bits, grelha de 4x4)	23	549,3	2088	Virtex-4

A Tabela 6 apresenta um resumo da análise feita segundo as propriedades arquiteturais: flexibilidade, escalabilidade, extensibilidade e modularidade.

O CoNoChi tem alta flexibilidade devido às suas tabelas de roteamento distribuído e a sua característica de redirecionamento de pacotes. O RMBoC também é flexível já que permite uma largura de banda adaptativa com número flexível de conexões entre os módulos. Os trabalhos de Palma, Huebner e Walder não permitem larguras de banda adaptativas sendo pouco flexíveis, o mesmo se verificando para o DyNoc e CuNoC, já que o seu algoritmo de roteamento S-XY não suporta largura de banda variável. A escalabilidade segundo a Tabela 2 considera o tempo de execução, então, dessa forma, o desempenho da comunicação se deteriora com a latência de chaveamento. Nesta premissa, somente a proposta de Huebner e o CuNoc tem alta escalabilidade. Os CoNoChi e DyNoC têm escalabilidade média, já que são medianamente afetados pela latência de chaveamento por serem sistemas 2D, implementados em um dispositivo 1D. Os trabalhos de Palma, Walder e o RMBoC paralisam suas comunicações durante o tempo da latência de chaveamento, com baixa escalabilidade. Com respeito à extensibilidade, como característica arquitetural, pode-se dizer que as NoC são extensíveis, uma vez que novos componentes podem ser alocados nas bordas, em qualquer direção. No caso dos barramentos, esse conceito limita-se a uma dimensão, devido à sua própria estrutura, sendo assim, as NoC têm melhor extensibilidade que os barramentos.

Tabela 6 - Comparação segundo propriedades arquiteturais dos sistemas.

Proposta	Flexibilidade	Escalabilidade	Extensibilidade	Modularidade
Palma	baixa	baixa	baixa	baixa
Huebner	baixa	alta	media	baixa
Walder	baixa	alta	media	media
RMBoC	alta	baixa	baixa	baixa
DyNoC	baixa	baixa	media	alta
CoNoChi	alta	media	media	alta
CuNoC	baixa	alta	alta	alta

Com uma análise mais detalhada das interfaces segundo os dispositivos e as ferramentas que foram utilizadas nas implementações, conclui-se o seguinte:

- A proposta de Palma tem obstáculos para ser estendida pelo fato de haver uma limitação de *buffers tri-state* no dispositivo, usados nos *bus-macros*.

- As propostas de Huebner e de Walder podem ser estendidas só na dimensão horizontal, segundo a largura do DRFPGA.
- A proposta do RMBoC só pode ser estendida até 6 módulos devido à limitação nos número dos Block-Rams no dispositivo.
- Nas propostas do DyNoC e CoNoChi, a limitação é para serem estendidas verticalmente, devido à implementação em dispositivos 1D.
- O CuNoC não tem limitações para ser estendida, seja horizontal ou verticalmente, salvo pelas dimensões do DRFPGA, já que foi implementado em um dispositivo 2D.

A modularidade, no caso da reconfiguração, pode-se limitar à granularidade dos elementos (módulos) que podem ser trocados, sendo que a maior modularidade ocorre nos sistemas com NoCs. No caso do DyNoc, CoNoChi e CuNoc, os módulos são retangulares de tamanho variável, enquanto que nos baseados em barramentos, só o trabalho de Walder permite que os módulos tenham largura variável (em múltiplos da largura dos módulos de tarefas nulas); todos os outros têm baixa modularidade por definirem uma área fixa para alocar os módulos reconfiguráveis.

3.4. Propostas sobre posicionamento dos módulos nos DRFPGAs.

Nesta seção, são apresentadas algumas propostas que consideramos relevantes para o posicionamento dos módulos que compõem SDR reportadas nos últimos anos. A pesquisa bibliográfica mostra a existência de posicionamento temporal tanto em tempo de compilação (41) como de execução (no modo *online* (42)). No primeiro caso, a sequência de posicionamentos é decidida antes de se colocar o sistema em operação, enquanto que, no segundo caso, o posicionamento de módulos reconfiguráveis é decidido durante a execução das tarefas da aplicação. Os trabalhos apresentados focam no aspecto algorítmico, buscando soluções

otimizadas, porém tendo como alvo SDRs genéricos; apesar de estes algoritmos serem genéricos e poderem ser utilizados também para SDRs baseados em NoCs, a não observância de especificidades destes sistemas, como a adjacência entre NoCs e roteadores, pode trazer resultados sub-ótimos. Aspectos de tráfegos dentro de NoCs podem também ser incorporados, introduzindo uma nova dimensão que aumentaria consideravelmente a complexidade do problema tratado.

3.4.1. Proposta de Teich et al.

A proposta de Teich et al (41) é do ano 1999 sendo a primeira de que se tem referência na área de posicionamento de módulos no DRFPGA. O posicionamento do módulo é feito espacial e temporalmente, portanto, depende das dimensões x, y, além do tempo de execução da mesma, t, como apresentado na Figura 27

Figura 27 - Posicionamento temporal representado como módulos 3D (29).

O posicionamento temporal é feito em tempo de compilação, onde a sequência de posicionamentos é decidida antes de se colocar o sistema em operação. Desta forma, esta proposta não se adapta a sistemas em tempo real, foco verificado em vários trabalhos posteriores.

3.4.2. Proposta de Ahmadinia et al, 2004

Um dos primeiros trabalhos de posicionamento em tempo de execução foi apresentado por Ahmadinia et al (42), onde propõem o posicionamento de PEs retangulares por um sistema dividido em três partes, tal como é ilustrado na Figura 28: uma base de dados dos módulos reconfiguráveis, um gerenciador de solicitações de posicionamento e o posicionador dos módulos no DRFPGA. Quando uma tarefa e o módulo correspondente solicitam o posicionamento, o algoritmo analisa *on-line* o espaço ocupado pelos módulos posicionados naquele instante no DRFPGA, para determinar onde posicionar o novo componente (nas áreas livres). O posicionador, então, informa ao gerenciador se o posicionamento do PE foi atingido ou não. O critério para a escolha do local de posicionamento é a otimização da comunicação entre dois componentes.

Figura 28 - Sistema de múltiplas tarefas para um dispositivo reconfigurável (42).

Neste trabalho, assim como nos outros encontrados na literatura, o foco é na eficiência algorítmica e não é dada atenção para elementos específicos da arquitetura da família do DRFPGA a ser utilizada, tornando necessária a inclusão destes dados posteriormente através de novas adaptações no fluxo de projeto.

3.4.3. Proposta de Tabero et al.

Uma evolução deste trabalho sobre particionamento foi publicado por Tabero et al (43), onde apresentaram uma estratégia similar, porém introduzindo um gerenciador de área disponível no dispositivo baseado em uma estrutura de vértices. O gerenciador determina as regiões disponíveis para o posicionamento de módulos correspondentes a tarefas que aguardam na fila de execução. Ao invés de se armazenar áreas, uma lista de vértices é usada e atualizada cada vez que módulos entram ou saem do DRFPGA. O uso da lista de vértices em conjunto com o algoritmo *Best Fit* (BF) fragmentado como algoritmo de gestão da área, permite atingir resultados otimizados. Assim como o trabalho de Ahmadinia et al, não é dada atenção para elementos específicos da arquitetura da família do DRFPGA a ser utilizada, tornando necessária a inclusão destes dados posteriormente através de novas adaptações no fluxo de projeto.

3.4.4. Proposta de Ahmadinia et al, 2007

Em 2007, Ahmadinia et al (44) apresentaram mais um trabalho sobre o assunto, nas mesmas bases que o trabalho anterior, com as seguintes inovações: inclusão de método para controle do espaço livre por meio do método “plane sweep” (da geometria computacional); minimização do custo de comunicação (análise do roteamento) por meio da redução da distância de Manhattan (e não euclidiano).

Segundo os resultados apresentados pelos autores, melhor desempenho (maior rapidez e melhores soluções) é obtido com a nova proposta, porém, novamente ainda não é dada atenção para elementos específicos da arquitetura da família do DRFPGA a ser utilizada.

4. ARQUITETURA E IMPLEMENTAÇÃO DE SDR COM NOC

O propósito deste capítulo é apresentar uma proposta de SDR, com a sua interface de comunicação baseada em redes intra-chip, além da metodologia a ser empregada para a sua simulação. Na seção 4.1 é apresentada a arquitetura do SDR para a qual desenvolveremos uma metodologia de projeto e ferramenta de apoio ao posicionamento. Na seção 4.2 a rede intra-chip Hermes será introduzida enquanto na seção 4.3, é apresentada uma explicação sobre os PRMs no SDR com rede intra-chip. Na seção 4.4 são dadas as explicações sobre a metodologia proposta de simulação no tempo total de execução, baseada na estratégia DCS. Na seção 4.5 a implementação da metodologia de simulação é apresentada, assim como os resultados.

4.1.Arquitetura do SDR proposto.

Como discutido na seção 2.1 os DRFGPAs Virtex-4 são denominados 2D devido à sua capacidade de reconfiguração baseada em unidades denominadas *frames*, fazendo com que os dispositivos Virtex-4 sejam compatíveis com as estruturas de NoC. Neste trabalho, desenvolvemos a arquitetura de um SDR, mostrada na Figura 29, cujo elemento central é uma NoC Hermes (24) que, dada a sua relevância, será descrita em seção posterior à parte. Ligados à NoC, temos elementos de processamento (PEs), um árbitro geral e outros circuitos auxiliares como multiplexadores, memórias, registradores, contadores, comparadores e uma porta lógica OR.

No exemplo da Figura 29, a rede Hermes é de 3x3 conectada a 9 PEs físicos, isto é, 9 PEs podem estar ativos ao mesmo tempo. Só alguns elementos do conjunto de PEs são associados a PRMs, enquanto os demais são módulos estáticos (fixos). De qualquer maneira, os PRMs comunicam-se com os não reconfiguráveis por meio

da NoC. Os PEs podem ser módulos de operações aritméticas, operações lógicas ou de memórias. São conectados ao respectivo roteador através de *wrappers*¹³.

Figura 29 - Sistema dinamicamente reconfigurável.

A arquitetura apresentada na Figura 29 prevê que uma determinada aplicação que é distribuída entre os PEs (na realidade, nos módulos, fixos e PRMs) seja executada corretamente através de um controlador geral (FSM árbitro) que controla toda a operação do sistema. O comportamento do sistema é definido por um número t de passos (seqüenciais), sendo cada um dos passos associado à ativação de um subconjunto de PEs, cujos dados de saída correspondentes são enviados a outros PEs. Para controlar esta sequência de passos, um módulo de contador de programa (PCcounter) indica o passo atual por meio de um sinal que controla os módulos ROMsel, ROMrptas e nove módulos ROMdest. As ROMs e outros módulos associados funcionam da seguinte maneira na execução de um passo:

- O módulo ROMsel tem t endereços para armazenamento dos passos, sendo acessados pelo Pcounter. À sua saída, o ROMsel envia um sinal de início para cada um dos elementos do subconjunto de PEs que devem operar no passo atual.

¹³ A principal função dos *wrappers* é prover uma interface para adaptar a transmissão e recepção de/para PEs pela rede.

- Os n^2 módulos ROMdest¹⁴, um para cada PE (a ser ativado), tem a função de indicar ao PE o destino do resultado de suas operações. Dentro destas ROMs, no mesmo endereço acessado pelo Pcounter, fica armazenado o destino das suas operações, ou seja, cada PE ativo recebe o endereço de destino e o adiciona ao cabeçalho de seu pacote.
- O módulo ROMrptas contém o valores esperados (conhecidos) dos sinais de pronto (*ready*) dos PEs, isto é, o sinal que indica o fim do processo para o passo atual indicado pelo Pcounter. Os valores serão 1 para os PEs que realizarem alguma operação no presente passo e 0 para os outros. Objetiva a sincronização para a definição do encerramento do passo em conjunto com o registrador Reg.
- O registrador Reg tem como objetivo armazenar os valores do sinal de pronto do conjunto dos PEs que são gerados à medida que estes terminem a sua operação no presente passo, portanto com valor 1.
- O módulo Comp compara o valor das saídas do módulo Reg com valor das saídas do módulo ROMrptas e, caso sejam iguais, indica ao controlador geral que o passo atual foi concluído.

Exemplificando o funcionamento desta arquitetura, vamos supor um caso em que a aplicação tenha sido distribuída em 100 passos. Vamos assumir que no primeiro passo, {PE0, PE1, PE6, PE7} devam realizar o processamento, sendo que os três primeiros devem encerrar o processamento neste mesmo passo e enviar os resultados obtidos a {PE2, PE3, PE0}, ou seja PE7 inicia o processamento neste passo, mas o terminará em um outro passo. Então, com Pcounter=1, no endereço '1' de ROMsel teríamos '110000110', indicando os PEs a serem ativados. Ao mesmo tempo, nos endereços '1' das ROMs de destino teríamos: ROMdest0=2 (ou binário 010), ROMdest1=3, ROMdest6=0, e todos os outros ROMdestj=X. Estes endereços são capturados pelos PEs 0, 1 e 6 e adicionados aos seus cabeçalhos dos pacotes de saída. O valor em ROMrptas seria '110000100', indicando os PEs para os quais deve-se esperar a finalização de processamento. Supondo PE1 termine o processamento primeiro e a seguir, PE6 e PE0, então o valor de Reg

¹⁴ Para a arquitetura proposta n é igual a 3, devido a que a NoC é 3x3.

mudará da seguinte forma: → '000000000' → '010000000' → '010000100' → '110000100'. Com este último, reconhece-se que o passo 1 tenha se encerrado.

No momento que se conclui o presente passo, o controlador geral cuida do apagamento do conteúdo do registrador Reg, além de incrementar o Pcounter para passar à execução do passo seguinte. Pelo exemplo dado, quando ao final do passo 1, Reg='110000100', o contador muda para Pcounter=2 e Reg é limpo. Caso todas as tarefas sejam finalizadas no exemplo, Pcounter=100, o sistema estaciona no estado de "fim", aguardando por um sinal de reinício do sistema.

4.2.A rede intra-chip Hermes

A rede Hermes (24) é uma rede intra-chip desenvolvida pelo grupo de pesquisa e desenvolvimento GAPH da PUCRS para SoPC, que será utilizada na arquitetura do SDR proposta na seção 4.1. As descrições em VHDL-RTL utilizadas são as que estão disponíveis no endereço eletrônico da referência (45). A rede Hermes é uma rede intra-chip padrão, sendo as suas principais características relacionadas na Tabela 7. Mais detalhes do funcionamento da rede podem ser obtidos junto à distribuição dos códigos-fonte.

Tabela 7 - Definições da rede Hermes.

Rede Intra-chip Hermes				
Topologia	Algoritmo de roteamento	Roteamento	Controle de Fluxo	Arbitragem
Grelha	Distribuído, adaptativo XY	Pacotes tipo Wormhole	Handshake	Round-Robin

4.3. PRMs no SDR com NoC

Não fosse pelo fato de que PRMs compartilhem um mesmo PE, a reconfiguração dinâmica não se faria perceber na arquitetura da Figura 29. Diferentemente das soluções apresentadas na seção 3.4, propusemos uma arquitetura simplificada de SDR com NoC com o intuito de podermos focar na exploração do problema de posicionamento de módulos. Pela estratégia adotada nesta dissertação, qualquer PE físico da arquitetura pode representar o local de processamento de diferentes PRMs que se substituem no tempo e, necessariamente, um PRM que tenha sido substituído só deve retornar à mesma posição. O escalonamento das tarefas (dos PRMs) dentro do sistema é feito em tempo de compilação e o seu comportamento depende do controlador da reconfiguração dinâmica, o qual é ilustrado na Figura 30

Figura 30 - Metodologia da reconfiguração.

- **Primeira etapa:** O árbitro geral solicita a reconfiguração de uma das PRRs ao controlador de reconfiguração.
- **Segunda etapa:** O controlador de reconfiguração tem a informação dos endereços da memória onde residem os *bitstreams* de cada um dos PRMs correspondentes por uma PRR e solicita a leitura da memória ao controlador da mesma.
- **Terceira etapa:** O controlador de reconfiguração indica ao controlador do ICAPS que a reconfiguração será feita e fica aguardando confirmação de fim da mesma.
- **Quarta etapa:** O Controlador do ICAPS indica ao controlador de reconfiguração o fim da reconfiguração.
- **Quinta etapa:** O controlador de reconfiguração indica ao árbitro geral o fim da reconfiguração (*bitstream*).

O tempo de reconfiguração depende do *bitstream* e do dispositivo utilizado.

Como dito anteriormente, a Metodologia Early Access precisa garantir o correto funcionamento do sistema no momento da reconfiguração de uma PRR. Por isso, deve-se inserir os *bus-macros* na quantidade apropriada para seus sinais externos, os quais interatuam com o roteador que é uma parte fixa do sistema. Desta forma, uma comunicação PE/roteador normal, Figura 31a, pode ser modificada, com a introdução de *bus-macros* que mantêm a comunicação dos mesmos, permitir o isolamento do PRM no momento da reconfiguração, como mostrado na Figura 31b, então, obtendo-se o grupo PRM/BM/roteador.

Figura 31 - Comunicação entre roteador/PE (a); Comunicação entre roteador/BM/PE (b).

4.4. Metodologia proposta para a simulação de SDR.

Na seção 2.2.2 foram apresentadas as modificações necessárias a uma descrição de um SDR para se poder simular o seu comportamento no tempo de execução. Nesta seção, dada a arquitetura do SDR baseado em NoC apresentada neste capítulo, apresentaremos a metodologia proposta para a sua simulação ilustrada na Figura 32. Pode-se verificar que ela é uma variante da metodologia tradicional de projeto de SDRs mostrada na seção 2.2.1.

Particionamento do Projeto: Esta etapa é igual à ilustrada na Figura 12 da seção 2.2.1.

Síntese individual dos módulos PRM: Também igual à ilustrada na Figura 12 da seção 2.2.1. Obtêm-se parâmetros importantes como a quantidade de *slices*, *frames*, *bitstreams* e o tempo de reconfiguração dos PRMs de interesse.

Modificação do árbitro geral para permitir DCS: O árbitro geral deve ser modificado na forma explicada na seção 2.2.2.1.

Controlador de configuração de PRM com chaves de isolamento: Como explicado na seção 2.2.2, o controlador de configuração deve ser projetado para controlar as chaves de isolamento, portanto deve ter como parâmetros os tempos de reconfiguração de cada um dos PRMs, obtidos na segunda etapa. As chaves de isolamento estão prontas e parametrizadas em arquivos disponíveis na biblioteca criada com suas definições.

Simulação do comportamento do sistema: A simulação do sistema modificado é feita por meio dos arquivos de teste para se poder verificar o comportamento do sistema. Se a simulação não corresponder ao comportamento esperado, então os módulos devem ser corrigidos e o processo deve ser executado novamente até que se atinja o comportamento esperado.

Figura 32 - Metodologia de simulação usando DCS.

4.5.Implementação da metodologia de simulação do sistema usando DCS.

O sistema mostrado na Figura 29 foi modificado segundo as indicações da seção 2.2.2 para a inclusão das chaves de isolamento e de seu controlador, para possibilitar a simulação RTL do comportamento da reconfiguração dinâmica.

O circuito-exemplo utilizado foi uma aplicação na qual, PE4 e PE5 são unidades que realizam operações aritméticas e lógicas, respectivamente, os outros elementos do conjunto PE são memórias. Todos os elementos do conjunto PE têm *wrappers* internos.

Para o exemplo, o sistema segue em oito passos:

1. Os dados armazenados em PE0 e PE8 são processados por PE5 e armazenados em PE3.
2. Os dados armazenados em PE2 e PE6 são processados por PE4 e armazenados em PE7.
3. Os dados armazenados em PE7 são transmitidos a PE2 para serem armazenados por ele.
4. Os dados armazenados em PE3 são transmitidos a PE8 para serem armazenados por ele.
5. Os dados armazenados em PE1 e PE2 são processados por PE5 e armazenados em PE6.
6. Os dados armazenados em PE7 e PE2 são processados por PE4 e armazenados em PE3.
7. Os dados armazenados em PE3 são transmitidos a PE2 para serem armazenados por ele.
8. Os dados armazenados em PE6 são transmitidos a PE1 para serem armazenados por ele.

O DRFPGA-alvo do projeto foi o Virtex-4LX25, com o qual foi observada uma ocupação de 45% do dispositivo Virtex-4LX25, após a síntese. A quantidade total de CLBs foi 1213 e a frequência máxima de operação é de 96 MHz.

Após a etapa da metodologia de simulação, explicada na seção 4.3, já se tem os dados para se poder calcular o tempo de reconfiguração dos elementos processantes PE4 e PE5. Para isto, utiliza-se a eq.(1), da seção 2.2.2.2 e os dados do DRFPGA alvo no Apêndice B; a seguir, é apresentado o cálculo para um dispositivo XC4VLX-25 contendo 96 linhas e 28 colunas de CLBs. Assume-se aqui que, tanto PE4 como PE5, ocupam uma altura de 16 linhas de CLBs e uma largura de 3 CLBs, padrão que será apresentado na Tabela 9 da seção 6.2.

Ilustrando o cálculo, primeiro procuramos obter a quantidade de *frames* de memória de configuração necessária para se reconfigurar uma coluna de CLBs, dado por $Frame_{COLUMN}$ na eq.(2), abaixo.

$$Frame_{COLUMN} = \frac{Frames_{RECONFIG}}{NUM_{COLUMN}} = \frac{5928}{28} = 212 \quad (2)$$

A seguir, procura-se obter a quantidade de *frames* de memória de configuração por retângulo na reconfiguração de uma PRR (PE4 ou PE5). Para isto usa-se a altura da PRR, $NUM_{ALTURA-CLBS-PRR}$, de 16 CLBs e a largura, $NUM_{LARGURA-CLBS-PRR}$, de 3 CLBs, e calcula-se $Frame_{PRR}$.

$$\begin{aligned} Frame_{PRR} &= \frac{Frames_{COLUMN}}{ALTURA_{COLUMN-CLBS}} * NUM_{ALTURA-CLBS-PRR} * NUM_{LARGURA-CLBS-PRR} \\ &= \frac{212}{96} * 16 * 3 = 106 \end{aligned} \quad (3)$$

O terceiro passo é calcular o tamanho do *bitstream* em kbytes para reconfigurar a coluna de PRR, dada pela eq.(4), a seguir.

$$\begin{aligned} Bistream_{FRAME-PRR} &= Frame_{PRR} * Palavras_{FRAME} * Bytes_{PALAVRAS} = 106 * 41 * 4 \\ &= 17384 \text{ Bytes} \end{aligned} \quad (4)$$

Adicionando o *overhead* de configuração, temos:

$$\begin{aligned} Bitstream &= Bistream_{FRAME-PRR} + Palavras_{OVHD} * Bytes_{PALAVRAS} \\ &= 17384 + 1312 * 4 = 22638 \text{ Bytes} \end{aligned} \quad (5)$$

Finalmente, compatibilizando os valores obtidos com a eq.(1),

$$Tempo_{RECONFIG} = \frac{22638Bytes}{4Bytes * 100MHz} = 57\mu seg \quad (6)$$

O resultado indica que o tempo de reconfiguração para o PE4 e PE5 será de aproximadamente 57 μ seg.

Com os tempos de reconfiguração computados, podem-se inserir os valores em forma de ciclos de relógio de processamento do sistema e realizar a simulação. Um pedaço dos resultados da simulação é apresentado na Figura 33, onde pode-se verificar o correto comportamento do sistema nos momentos destacados.

Figura 33 - Simulação do sistema dinamicamente reconfigurável no tempo de execução.

A carta de tempos ilustra os sinais relacionados com o PE 5 e o controlador de configuração da PRR correspondente nas etapas a, b, c e d, podendo-se verificar os diversos estados da chave de isolamento:

- a) Nesta etapa, o PRM 5a está em estado de transição e o PRM 5b está no estado inativo. O sinal starta = '1' e startb = '0' indica que o PRM5a deve ativar-se e o PRM5b deve permanecer inativo (desligado

da lógica). Durante esta etapa o sinal de reconfiguração tem o valor '1' indicando que se está produzindo uma reconfiguração. O sinal de controle das chaves relacionadas com o PRM5a está em '1', ou seja, as saídas do módulo têm valor X (desconhecido), próprio da reconfiguração. O sinal de controle das chaves relacionadas com o PRM5b está em 'Z', ou seja, representa que as entradas e a saída deste módulo estão desligadas. A saída do PE5 é a referente ao PRM5a.

b) Nesta etapa, o PRM 5a está em estado ativo e o PRM 5b está no estado inativo. O sinal $starta = '1'$ e $startb = '0'$ indica que o PRM5a deve ficar ativo e o PRM5b deve permanecer desligado da lógica. Durante esta etapa, o sinal de reconfiguração tem o valor '0' indicando que não se está produzindo nenhuma reconfiguração, já que na etapa anterior aquilo foi feito, ou seja, agora o PRM5a está interagindo com a lógica do sistema enquanto PRM5b fica desligado. O sinal de controle das chaves relacionadas com o PRM5a está em '0', ou seja, as entradas e saídas do PRM5a estão conectadas com o SDR. O sinal de controle das chaves relacionadas com o PRM5b está em 'Z', ou seja, representa que as entradas e a saída deste módulo estão desligadas. A saída do PE5 é a referente ao PRM5a.

c) Nesta etapa, o PRM 5a está em estado inativo e o PRM 5b está no estado de transição. O sinal $starta = '0'$ e $startb = '1'$ indica que o PRM5b deve ativar-se e o PRM5a deve permanecer inativo (desligado da lógica). Durante esta etapa, o sinal de reconfiguração tem o valor '1' indicando que se está produzindo uma reconfiguração. O sinal de controle das chaves relacionadas com o PRM5b está em '1' ou seja, as saídas do módulo têm valor X (desconhecido), próprio da reconfiguração. O sinal de controle das chaves relacionadas com o PRM5a está em 'Z', ou seja, representa que as entradas e a saída deste módulo estão desligadas. A saída do PE5 é a referente ao PRM5b.

d) Nesta etapa, o PRM 5a está em estado inativo e o PRM 5b está no estado ativo. O sinal $starta = '0'$ e $startb = '1'$ indica que o PRM5b deve ficar ativo e o PRM5a deve permanecer desligado da lógica. Durante esta etapa, o sinal de reconfiguração tem o valor '0' indicando que não se está produzindo nenhuma reconfiguração, já que tal foi realizado na etapa anterior, ou seja, agora o PRM5b está interagindo com a lógica do sistema enquanto PRM5a fica desligado. O sinal de controle das chaves relacionadas com o PRM5a está em '0', ou seja, as entradas e saídas do PRM5b estão conectadas com o SDR. O sinal de controle das chaves relacionadas com o PRM5a está em 'Z', ou seja, representa que as entradas e a saída deste módulo estão desligadas. A saída do PE5 é a referente ao PRM5b.

Para o mesmo intervalo da simulação apresentado na Figura 33, o comportamento do sistema e da NoC é mostrado na Figura 34, incluindo-se os sinais do PE5 (módulos PRM5a e PRM5b), além do porto local roteador 0201 (aquele que interatua com PE5).

Figura 34 - Simulação do SDR no tempo de execução detalhe da NoC.

Antes de descrever o que acontece com os elementos da NoC é necessário fazer uma equivalência entre as entradas e saídas do porto local do roteador 0201 em relação ao PE5, o que está detalhado na Tabela 8. A aparente troca dos nomes deve-se ao fato de que, no momento que o roteador transmite dados para o PE, este último tem a referência de receptor com seus portos representando sinais no sentido oposto.

Tabela 8 - Equivalência entre porto local do Roteador0201 e PE5

Portos locais do roteador 0201	Portos do PE5
txlocal	rx
ack_txlocal	ack_rx
rxlocal	tx
ack_rxlocal	ack_tx
data_inlocal	data_out
data_outlocal	data_in

Na carta de tempos pode-se verificar os diversos valores dos sinais dos portos locais do roteador0201 e dos portos de entrada e saída do PE:

- a) Nesta etapa, o PRM 5a está em estado de transição e o PRM 5b está no estado inativo. O sinal starta = '1' e startb = '0' indica que o PRM5a deve ativar-se e o PRM5b deve permanecer inativo (desligado da lógica). As entradas e saídas do PE5 são as referentes ao PRM5a, mas esta situação se evita no SDR com o posicionamento dos BM portanto, não tem relevância o que se mostra neste intervalo.
- b) Nesta etapa o PRM 5a está em estado ativo e o PRM 5b está no estado inativo. O sinal starta = '1' e startb = '0' indica que o PRM5a deve ficar ativo e o PRM5b deve permanecer desligado da lógica. As entradas e saídas do PE5 são as referentes ao PRM5a, portanto, ao se observar a Figura 34 os sinais do roteador txlocal, ack_txlocal, rxlocal, ack_rxlocal, data_inlocal, data_outlocal coincidem com os sinais do PRM5a rx,ack_rx, tx, ack_tx, data_out, data_in.
- c) Nesta etapa o PRM 5a está em estado inativo e o PRM 5b está no estado de transição. O sinal starta = '0' e startb = '1' indica que o PRM5b deve ativar-se e o PRM5a deve permanecer inativo (desligado da

lógica), as entradas e saídas do PE5 são as referentes ao PRM5b, mas esta situação se evita no SDR com o posicionamento dos BM portanto, não tem relevância o que se mostra neste intervalo.

d) Nesta etapa, o PRM 5b está em estado ativo e o PRM 5a está no estado inativo. O sinal $starta = '0'$ e $startb = '1'$ indica que o PRM5b deve ficar ativo e o PRM5a deve permanecer desligado da lógica. As entradas e saídas do PE5 são as referentes ao PRM5b, portanto, ao se observar a Figura 34 os sinais do roteador $txlocal$, $ack_txlocal$, $rxlocal$, $ack_rxlocal$, $data_inlocal$, $data_outlocal$ coincidem com os sinais do PRM5b rx , ack_rx , tx , ack_tx , $data_out$, $data_in$.

5. POSICIONAMENTO DE MÓDULOS

O propósito deste capítulo é apresentar o desenvolvimento da ferramenta de posicionamento DynoPlace, assim como aspectos da metodologia de projeto de SDRs que levaram a ele. Na seção 5.1 se estabelece a modificação da metodologia do projeto de SDR objetivando a automatização da tarefa de posicionamento. O problema do posicionamento é declarado nas seções 5.2 e 5.3, enquanto a ferramenta de posicionamento DynoPlace é apresentada na seção 5.4.

5.1. Modificação da metodologia de projeto Early Access para SDRs.

Na seção 2.2.2, foram apresentadas as etapas do projeto dentro da metodologia Early Access. Foi mencionado ali que muitas delas necessitavam de intervenção manual por parte do projetista; é um fato de que para que o projeto de SDRs seja aceito de forma generalizada falta suporte adequado que resulte em pouco custo adicional na comparação a projetos planos.

Neste trabalho, foi realizada uma análise do projeto baseado no Early Access para se determinar os pontos onde seria possível se fazer uma contribuição em termos de metodologia e ferramentas. Foi possível perceber que a tarefa de posicionamento feita manualmente com a ferramenta Floorplanner é aquela que mais dificuldades causa por necessitar cuidados por parte do projetista para não alocar um módulo em uma posição errada e por deixar nas mãos do projetista a decisão de posicionamento. Se por um lado, um projetista experiente pode realizar um excelente trabalho de posicionamento, apesar de necessitar muitas vezes de tempo excessivo, o mesmo não seria comum para projetistas médios ou então quando grande número de módulos reconfiguráveis está previsto.

É proposta deste trabalho então, como mostrada na Figura 35, modificar a quinta etapa, realizando-a a partir de uma ferramenta automática de posicionamento.

Baseado em biblioteca de DRFPGAs e restrições definidas pelo projetista, o arquivo final contendo todo o posicionamento do circuito é gerado. Desta forma, pode-se realizar o conjunto de tarefas executando-se os *scripts* da Xilinx (27).

Figura 35 - Modificação na metodologia do projeto para SDR.

5.2.O problema do posicionamento

A quinta tarefa da Figura 35, tanto na Early Access como na nova metodologia é responsável pela geração automática do arquivo UCF que contém o posicionamento de todos os módulos do SDR. A grande diferença, na nova metodologia, é a captura de informações de cada módulo e a definição de suas posições de alocação por uma ferramenta de auxílio.

O problema do posicionamento que se pretende resolver aqui pode ser definido da seguinte forma:

- Deseja-se dado um módulo da descrição RTL do SDR, definir-lhe uma posição dentro da planta-baixa de um DRFPGA, de tal forma que fique minimizado o consumo de área (ou algum outro parâmetro de medida) e que sejam atendidas todas as restrições envolvidas.

Para a resolução dos problemas são dados:

- a) Descrição estrutural do sistema através de conjunto de módulos fixos e reconfiguráveis a serem posicionados, descritos em linguagem de descrição de hardware.
- b) Biblioteca de DRFPGAs, contendo informações de quantidade e posição de CLBs e de posicionamento de recursos (linhas de *clock*, lógicas DSP, BlockRams, etc.)
- c) Banco de restrições, a ser descrito na próxima seção.

Como saída após a resolução do problema, temos:

- Arquivo (UCF) contendo as informações de posicionamento de cada módulo, além de outros dados de posicionamento de linhas de *clock* e demais recursos. Este arquivo é lido pela ferramenta de posicionamento físico e roteamento da Xilinx.

5.3. Restrições de posicionamento

Segundo a referência sobre Early Access (27), para diferentes famílias de dispositivos, uma série de restrições específicas devem ser observadas, além de outras que devem ser incluídas diretamente no arquivo UCF. As restrições que se apresentam nesta seção se referem à família de dispositivos Virtex-4, mas também podem ser estendidas para os Virtex-5, entretanto, com outros parâmetros numéricos.

A primeira restrição se refere ao tamanho do *frame* de reconfiguração, já mencionado na seção 2.2, que é de 16 CLBs de altura por 1 CLB de largura (ao invés de uma coluna inteira, como no Virtex II), como pode ser visto na Figura 36¹⁵. Desta forma, a altura dos módulos será de 16 CLBs enquanto a largura será de uma quantidade a determinar segundo critérios apresentados mais adiante. Resumindo, os módulos devem ser padronizados na altura e em uma posição inicial fixa.

Figura 36 - Configurações dos VirtexII (a) e Virtex-4 (b) (27).

Outro aspecto interessante refere-se à quantidade de *bus-macros* que podem ser alocados por quantidade de CLBs. O número possível de fios é 8 bits por CLB, mas infelizmente não se pode alocar os *bus-macros* em todos os CLBs de um *frame* de reconfiguração. Atendendo as restrições da tecnologia, a metodologia, impõe a separação entre os CLBs usados para as conexões dos *bus-macros* em 4 filas de

¹⁵ Para facilitar a leitura a Figura 8 é replicada aqui.

CLBs. Em outras palavras, só 4 CLBs de um frame podem ser utilizados para *bus-macros*, fornecendo 32 bit de fios ao invés de 128 bits(16×8), que seria o caso ideal.

Dadas estas restrições, ao analisarmos o número de fios num PRM, somos capazes de calcular o número requerido de CLBs para as conexões, assim como o número real de CLBs para a implementação da PRM.

5.4.Ferramenta DynoPlace

O problema de posicionamento descrito na seção 5.3 está dentro da classe dos NP-completo, de alta complexidade. Algoritmos exatos para solucioná-lo exigirão um tempo de processamento com dependência exponencial ao número de elementos a serem posicionados, portanto, heurísticas são necessárias ao se tratar casos com grande número de elementos.

Um possível atalho que pode aliviar um pouco as dificuldades acima é simplificar o próprio problema de posicionamento. Isto pode ser feito pela inclusão de algumas restrições como a da altura igual de 16 CLBs para todos os módulos, apresentada na seção 5.3. Com altura fixa, fica estabelecido de acordo com o DRFPGA utilizado, o número de fileiras que se pode ter. Por exemplo, para o dispositivo Virtex-4LX25, o número de filas é 6, uma vez que o número total de CLBs na vertical é 96.

O novo problema passa a ser o seguinte: dado um DRFPGA com número fixo de filas (de 16 CLBs), como alocar um conjunto de módulos de larguras diferentes neste dispositivo, de tal forma que a fileira mais longa seja minimizada. Este problema é semelhante ao do conhecido escalonamento de processamentos em processadores com minimização de *makespan* (momento em que termina a execução da última tarefa) (46).

Para se resolver este novo problema, a ferramenta DynoPlace foi desenvolvida, em uma versão que, apesar de poder ser de uso genérico em diversas

famílias de DRFPGAs, por ser baseada em bibliotecas de componentes e restrições, apresenta uma implementação restrita ao posicionamento dos módulos que compõem o sistema digital no DRFPGA Virtex-4LX25.

A ferramenta é baseada em regras, ou seja, para cada um dos elementos do conjunto de módulos a serem posicionados, as suas restrições são analisadas para se decidir onde exatamente deve ser sua posição no DRFPGA.

5.4.1.A modelagem do sistema

O sistema dinamicamente reconfigurável pode ser representado por um conjunto M de n módulos, elementos da união dos conjuntos S e D . Os elementos do conjunto S são t módulos estáticos e os elementos do conjunto D são f módulos dinâmicos, portanto:

$$n = t + f \quad (6)$$

$$M = \{m_1, \dots, m_n\} = S \cup D = \{s_1, \dots, s_t\} \cup \{d_1, \dots, d_f\} \quad (7)$$

No exemplo usado na seção 4.1, para a arquitetura apresentada na Figura 29 dois elementos para cada um dos PEs (PE4 e PE5) são elementos do conjunto D , enquanto todos os outros módulos são elementos do conjunto S .

Para a estruturação do procedimento de posicionamento, a dependência entre os módulos será acompanhada. Pode-se representar o sistema como um conjunto de elementos relacionados como mostrado na Figura 37. Os módulos $v_i \in V$ são elementos, cujos portos são somente sinais internos do sistema (não são entradas nem saídas do sistema), enquanto os módulos $w_i \in W$, são todos os outros, ou seja, módulos do sistema com portos de entrada ($i_i \in I$), de saída ($o_i \in O$) ou ambos tipos ($io_i \in IO$). As dependências (conexões) entre os módulos são representados pelos elementos $e_i \in E$.

Figura 37 - Grafo $G(I, IO, O, V, E)$

No exemplo usado na seção 4.1, a arquitetura contém uma série de elementos que são do tipo v . Podem ser representados da seguinte forma: $V \in (PE \cup R \cup OE)$. PE é o conjunto de elementos processantes da rede (que podem ser estáticos ou PRMs a serem reconfigurados nas PRRs) e estão conectados aos roteadores. R é o conjunto dos roteadores da rede e OE é o conjunto de outros elementos, módulos que não têm entradas nem saídas do sistema e tampouco são elementos processantes ou roteadores. Na arquitetura definida para o SDR são as RAMs, muxes, etc.

5.4.2. Algoritmo do DynoPlace

O algoritmo de posicionamento em si é pertencente ao grupo do tipo míope (*greedy*). Para resolver o problema descrito no início desta seção, se propõe usar como base o algoritmo *list scheduling*, bastante conhecido da área de metodologia de projeto de CIs. O algoritmo é mesclado a decisões regidas por biblioteca de restrições do DRFPGA.

O algoritmo recebe os dados da quantidade de CLBs de cada um dos módulos do conjunto M e suas dependências (conexões) com outros módulos. O algoritmo de *list scheduling* adota o escalonamento em que todos os módulos, da

mesma altura, são posicionados seqüencialmente nas filas de baixo para cima e da esquerda para direita. O grupo (um módulo ou um conjunto de módulos relacionados que serão posicionados como um só) são sempre posicionados na posição livre mais a esquerda e a prioridade é de baixo para cima, caso se tenha mais de uma posição disponível. A sequência é definida de acordo com o tipo dos módulos e com o critério de prioridade descrito nesta seção.

Inicialmente o posicionamento começa de baixo para cima com os elementos alocados no ponto zero (canto inferior esquerdo). A Figura 38a ilustra a condição onde só se posicionam os módulos na esquerda para completar a primeira coluna, enquanto a Figura 38b mostra a etapa seguinte, quando a coluna foi completada. Finalmente a Figura 38c mostra a etapa seguinte onde um módulo é posicionado na fila onde se tem a posição livre mais a esquerda. A Figura 39 mostra as tarefas específicas do algoritmo do DynoPlace, que serão explicadas a seguir.

Figura 38 - Posicionamento de módulos no DRFPGA.

Figura 39 – Tarefas específicas do algoritmo do DynoPlace

Preprocessamento: Esta tarefa consiste em computar as dimensões dos módulos e registrar uma série de regras para o posterior processamento. As dimensões (altura e largura) de todos os módulos são definidas nesta etapa e, para normalizá-las, todos os módulos terão a mesma altura, ou seja, um frame do Virtex-4 (16 filas de CLBs). A largura de cada módulo será calculada fazendo a divisão da sua quantidade total de CLBs entre 16, como descrito na seção 2.2.

A largura dos módulos que são elementos dos conjuntos I, IO e O são calculadas do mesmo modo, mas para prevenir possíveis erros com o posicionamento dos *bus-macros*, devem ter, no mínimo, largura de 6 CLBs se estiverem conectados com algum PRR. Se um elemento do conjunto OE estiver entre o módulo que pertence aos conjuntos I/IO/O e uma PRR, então a largura para o módulo de I/IO/O deve ser, no mínimo, de 3 CLBs. As entradas serão posicionadas máxima e preferencialmente junto ao canto esquerdo do DRFPGA para que estejam perto dos IOBs do dispositivo.

Posicionamento de PRR e módulos adjacentes: Esta é a primeira etapa para o posicionamento de módulos com maiores restrições. Neste momento, só alguns PEs, os que são PRMs, serão posicionados nas PRRs. Neste caso, começando-se pelos mais largos, cada elemento composto por uma PRR e seus módulos dependentes devem ser posicionados numa mesma fila (enquanto couber) e o posicionamento dos componentes sempre segue a sequência horizontal:

I/O/IO → PRR (para os correspondentes PRMs) → R

ou

I/O/IO → OE → PRR (para os correspondentes PRMs) → R

Após o posicionamento do grupo indicado, se procede a posicionar os *bus-macro* no canto esquerdo e no canto direito da PRR.

Posicionamento de demais módulos fixos I, IO, O e adjacentes: Os módulos a serem posicionados pelo próximo critério de prioridade são os que têm entradas ou saídas do sistema. Na sequência, todos os outros PEs estáticos com seus módulos dependentes devem ser posicionados numa mesma fila (enquanto couber) e o posicionamento dos componentes sempre segue a sequência horizontal

I/IO/O → PEs estáticos → R

ou

I/IO/O → OE → PEs estáticos → R

Posicionamento de módulos fixos V e adjacentes: Esta etapa os módulos internos são posicionados um a um, começando-se pelos maiores.

Posicionamento dos bus-macros: As mesmas restrições para o posicionamento dos *bus-macros* apresentadas na referência (27) são adotadas nesta ferramenta. Por simplicidade, nesta versão do DynoPlace, os *bus-macros* não estão previstos para comunicação vertical.

6. RESULTADOS EXPERIMENTAIS DE POSICIONAMENTO

Nesta seção são apresentados os resultados experimentais relacionados à implementação da metodologia proposta no presente trabalho para o posicionamento dos módulos no DRFPGA (ferramenta DynoPlace), segundo os algoritmos e definições apresentadas na seção 5.

6.1.Implementação da metodologia de posicionamento do DynoPlace.

Para o desenvolvimento da ferramenta DynoPlace, decidiu-se pela utilização da linguagem C++. A idéia de se utilizar tal linguagem deve-se ao fato de ser orientado a objetos, já que no futuro (não foi implementado no presente trabalho) poderia-se definir a classe DRFGPA Virtex-4 para depois ter subclasses: LX, FX e SX, as quais devido ao conceito de herança podem usar os parâmetros gerais dela. Depois para cada subclasse se podem definir subclasses com os dispositivos correspondentes como o LX15, LX25, etc. Assim, finalmente seriam definidos todos os FPGAs de interesse da Xilinx. Para o desenvolvimento do programa, foram consideradas as etapas apresentadas na Figura 40, que são explicadas a seguir.

Restrições do dispositivo: Esta etapa é onde o programa carrega os parâmetros do dispositivo Virtex-4LX25 como: quantidade de CLBs por fila e por coluna, localização e nome dos elementos RAMB_16, FIFO_16 e DSP_16, das entradas do relógio e os buffer globais de relógio, além dos IOBs. Estes dados são muito importantes já que a tarefa da ferramenta é posicionar os módulos que compõem o SDR no DRFPGA e todos os bloqueios e afastamentos devem ser computados.

Figura 40 - Etapas da ferramenta DynoPlace.

Geração de um grupo (módulos fixos e o PRM com que interatuam) com restrições de largura para os módulos: Esta etapa, o programa já carregou os arquivos de texto que contêm a seguinte informação:

- Nome e quantidade de CLBs dos módulos fixos.
- Nome e quantidade de CLBs dos módulos reconfiguráveis.
- Nome dos módulos relacionados com cada PRR.

O programa foi implementado para calcular a largura de cada módulo segundo a restrição de altura de 16 CLBs e de largura para módulos relacionados com o PRM segundo o explicado na 5.4.2. Nesta etapa, a ocupação real (número de frames) é obtida.

Análise da adequação do projeto ao Virtex-4LX25: Com os dados da largura, calculados na etapa anterior, computa-se a quantidade real de CLBs que o projeto precisa ter a sua disposição no dispositivo. Se a quantidade for menor ou igual à quantidade de CLBs do dispositivo, então a ferramenta segue para as etapas seguintes, caso contrário, uma mensagem de insuficiência é escrita na tela, junto a informações relevantes para o projetista como a largura e quantidade real de CLBs de cada um dos módulos do SDR, além da quantidade total de CLBs do projeto.

Posicionamento dos grupos segundo algoritmo do DynoPlace: Com a altura fixa e a largura real calculada na etapa anterior, procede-se ao posicionamento, ou seja do cálculo da posição absoluta, do primeiro grupo, a partir da posição 0,0 (canto inferior esquerdo) para a direita. Os outros grupos seguirão a mesma lógica e as posições serão computadas para colocação acima dos outros.

Posicionamento dos outros módulos fixos segundo algoritmo do DynoPlace: Igual que a etapa anterior, como explicado na seção 5.4.2.

Posicionamento das entradas e saídas e dos BM: Nesta etapa o programa já carregou os arquivos de texto com as seguintes informações:

- Nome e sentido dos BMs de cada uma das PRRs.
- Nome das entradas e saídas da entidade principal do SDR.

Agora se procede a posicioná-los no dispositivo posicionando as entradas e saídas o mais perto possível dos módulos implicados com elas. No caso dos BM se posicionam respeito dos cantos de cada um dos PRRs segundo as restrições da metodologia Early Access (27).

Geração dos três UCFs: São gerados 3 arquivos UCFs. O primeiro contém o posicionamento de todos os elementos que compõem o SDR no FPGA, necessários

na etapa inicial dos *scripts* da Xilinx. O segundo e o terceiro são os que têm o posicionamento dos *bus-macros*, além de outras linhas adicionais; há apenas uma diferença no *modo* entre eles, caso o arquivo UCF esteja na pasta base ou no top, *closed* e *reconfig*, respectivamente. Estes são necessários nas diferentes etapas posteriores nos *scripts* da Xilinx na metodologia Early Access.

6.2. Posicionamento dos módulos com a ferramenta DynoPlace

A ferramenta DynoPlace gera 3 arquivos UCF, como foi explicado na seção 6.1, o primeiro é utilizado com a ferramenta de Floorplanner, do qual gera-se o posicionamento dos módulos no FPGA e a sua informação gráfica como mostrada na Figura 41. Para os principais módulos há um número identificador, cuja associação a módulo é dada segundo a Tabela 9. Pode-se observar que todos os elementos do conjunto de módulos têm a mesma altura de 16 linhas de CLBs, e que a largura de cada um depende dos critérios já descritos do algoritmo do DynoPlace. Os grupos gerados pelo DynoPlace explicados na seção 6.1 estão compostos por:

- Grupo1: FSMarbitro - flipflopesp_4 – PE4 - Roteador 0101, representados pelos identificadores: 10 – 27 – 41 - 5 respectivamente.
- Grupo2: FSMarbitro - flipflopesp_5 – PE5 - Roteador 0201, representados pelos identificadores: 10 - 28 - 42 - 6 respectivamente.

Note-se que nos grupos1 e 2, repete-se o módulo FSMarbitro, sendo isto devido ao fato de que os sinais dos módulos flipflopesp_4 e flipflopesp_5 também interatuam com ele.

Depois de posicionar os 2 grupos, procede-se a posicionar o maior módulo dos restantes, na posição mais livre da esquerda, que é feito, segundo a Tabela 9, com o Roteador 0102 identificado pelo identificador 8; depois dele se posiciona o módulo Roteador 0001 e continua-se sucessivamente com o algoritmo.

Pode-se observar que as duas regiões PRR estão localizadas na parte inferior do DRFPGA, com o identificador 41 para o PRR1 (PE4) e o identificador 42 para o

PRR2 (PE5). Observa-se também que não foi necessário para tal SDR utilizar o total da área disponível do dispositivo.

Figura 41 - Posicionamento dos módulos do SDR com o DynoPlace.

Na Figura 42 e Figura 43 é ilustrado o roteamento dos módulos reconfiguráveis PE4 e PE5, respectivamente, visualizado com a ferramenta FPGAEitor da Xilinx. Na Figura 44 o roteamento do SDR total é ilustrado, incluindo-se os PRRs, onde os PE4 (retângulo cor amarelo) e PE5 (retângulo cor azul) são posicionados, além dos *bus-macros* que estão dentro de retângulos vermelhos.

Figura 42 - Roteamento do módulo reconfigurável PE4 no DRFPGA.

Figura 43 - Roteamento do módulo reconfigurável PE5 no DRFPGA.

Figura 44 - Roteamento do SDR completo.

A quantidade total de CLBs do sistema computados após a síntese individual dos blocos foi de 1238, mas devido ao fato de usarmos *frames* inteiros para os diversos módulos, como explicado na seção 5.4.2, a quantidade real de CLBs utilizados foi aumentado para 1856, equivalente a um incremento do 49,92% em relação à quantidade original. Este aumento é detalhado na Tabela 9, onde na coluna 1 se tem o identificador do módulo, na coluna 2 o nome dos módulos, na coluna 3 mostra-se a quantidade de CLBs original por módulo, na coluna 4 a altura em termos de CLBs do módulo obtidos na ferramenta DynoPlace, na coluna 5 tem-se a largura em termos de CLBs do módulo obtidos na ferramenta DynoPlace, na coluna 6 a quantidade de CLBs por módulo obtida pelo DynoPlace e finalmente na coluna 7 o incremento percentual da quantidade total de CLBs, por módulo, dos dados originais com relação aos obtidos pela ferramenta.

Tabela 9 – Comparação da quantidade dos CLBs do projeto pós-síntese individual respeito aos do DynoPlace.

Ident	Módulos	CLBs orig	Altura DynoPlace (CLBs)	Largura DynoPlace (CLBs)	CLBs DynoPlace	% incremento CLBs
1	Roteador 0000	86	16	6	96	11,63
2	Roteador 0100	125	16	8	128	2,4
3	Roteador 0200	95	16	6	96	1,06
4	Roteador 0001	124	16	8	128	3,23
5	Roteador 0101	160	16	10	160	0
6	Roteador 0201	137	16	9	144	5,11
7	Roteador 0002	91	16	6	96	5,5
8	Roteador 0102	137	16	9	144	5,11
9	Roteador 0202	104	16	7	112	7,7
10	FSMarbitro	6	32	3	96	1500
11	pCounter	1	16	1	16	1500
11	memromsel	1	16	1	16	1500
12	memromrpta	1	16	1	16	1500
13	memromdirdest_0	1	16	1	16	1500
14	memromdirdest_1	1	16	1	16	1500
15	memromdirdest_2	1	16	1	16	1500
16	memromdirdest_3	1	16	1	16	1500
17	memromdirdest_4	1	16	1	16	1500
18	memromdirdest_5	1	16	1	16	1500
19	memromdirdest_6	1	16	1	16	1500
20	memromdirdest_7	1	16	1	16	1500
21	memromdirdest_8	1	16	1	16	1500
22	modcontresreg	1	16	1	16	1500
23	flipflopesp_0	1	16	1	16	1500
24	flipflopesp_1	1	16	1	16	1500
25	flipflopesp_2	1	16	1	16	1500
26	flipflopesp_3	1	16	1	16	1500
27	flipflopesp_4	1	16	3	48	4700
28	flipflopesp_5	1	16	3	48	4700
29	flipflopesp_6	1	16	1	16	1500
30	flipflopesp_7	1	16	1	16	1500
31	flipflopesp_8	1	16	1	16	1500
32	comparador	1	16	1	16	1500
33	mux2a1gen	2	16	1	16	700
34	PE0	15	16	1	16	6,67
35	PE1	15	16	1	16	6,67
36	PE2	15	16	1	16	6,67
37	PE3	15	16	1	16	6,67
38	PE6	15	16	1	16	6,67
39	PE7	15	16	1	16	6,67
40	PE8	15	16	1	16	6,67
41	PE4	24	16	3	48	140
42	PE5	19	16	3	48	152,64
	TOTAL	1238			1856	49,92

Para se testar algumas limitações da ferramenta DynoPlace, foi proposto o projeto de um SDR, cuja quantidade original de CLBs pós-síntese individual foi de 2011, mas com o incremento verificado com a ferramenta, o valor final foi de 2688 (capacidade total do DRFPGA), ou seja, o incremento é de 33,67%. O posicionamento obtido para tal SDR é apresentado na Figura 45. Este incremento é detalhado na Tabela 10 a qual segue a estrutura da Tabela 9.

Pode-se observar, como no exemplo anterior, que todos os elementos do conjunto de módulos têm a mesma altura de 16 linhas de CLBs, e que a largura de cada um depende dos critérios do algoritmo do DynoPlace. Os grupos gerados pelo DynoPlace estão compostos de maneira similar, assim como os posicionamentos posteriores. Pode-se observar que algumas diferenças ocorreram nos módulos menores.

O projeto não apresentou dificuldade de ser finalizado com os *scripts* da metodologia Early Access da Xilinx a partir dos arquivos UCF obtidos pelo DynoPlace. O autor acredita que tal fato sempre se verificará desde que o projeto caiba no DRFPGA.

Figura 45 - Posicionamento dos módulos de um SDR com o DynoPlace, ocupando a totalidade de CLBs do DRFPGA.

Tabela 10 - Comparação da quantidade dos CLBs do projeto pós-síntese individual em relação aos do DynoPlace ocupando a totalidade de CLBs do DRFPGA.

Ident	Módulos	CLBs orig	Altura DynoPlace	Largura DynoPlace	CLBs DynoPlace	% incremento CLBs
1	Roteador 0000	120	16	8	128	6,67
2	Roteador 0100	175	16	11	176	0,58
3	Roteador 0200	133	16	9	144	8,28
4	Roteador 0001	174	16	11	176	1,15
5	Roteador 0101	224	16	14	244	0
6	Roteador 0201	191	16	12	192	0,53
7	Roteador 0002	127	16	8	128	0,79
8	Roteador 0102	192	16	12	192	0
9	Roteador 0202	146	16	10	160	9,59
10	FSMarbitro	23	32	3	96	178,27
11	pCounter	5	16	1	16	220
11	memromsel	1	16	1	16	1500
12	memromrpta	1	16	1	16	1500
13	memromdirdest_0	1	16	1	16	1500
14	memromdirdest_1	1	16	1	16	1500
15	memromdirdest_2	1	16	1	16	1500
16	memromdirdest_3	1	16	1	16	1500
17	memromdirdest_4	1	16	1	16	1500
18	memromdirdest_5	1	16	1	16	1500
19	memromdirdest_6	1	16	1	16	1500
20	memromdirdest_7	1	16	1	16	1500
21	memromdirdest_8	1	16	1	16	1500
22	modcontresreg	1	16	1	16	1500
23	flipflopesp_0	1	16	1	16	1500
24	flipflopesp_1	1	16	1	16	1500
25	flipflopesp_2	1	16	1	16	1500
26	flipflopesp_3	1	16	3	48	4700
27	flipflopesp_4	1	16	3	48	4700
28	flipflopesp_5	1	16	1	16	1500
29	flipflopesp_6	1	16	1	16	1500
30	flipflopesp_7	1	16	1	16	1500
31	flipflopesp_8	1	16	1	16	1500
32	comparador	1	16	1	16	1500
33	mux2a1gen	2	16	1	16	700
34	PE0	50	16	4	64	28
35	PE1	50	16	4	64	28
36	PE2	50	16	4	64	28
37	PE3	50	16	4	64	28
38	PE6	50	16	4	64	28
39	PE7	50	16	4	64	28
40	PE8	50	16	4	64	28
41	PE4	72	16	6	96	33,34
42	PE5	55	16	5	80	45,46
	TOTAL	2011			2688	33,67

Com os resultados da Tabela 9 e da Tabela 10, comprova-se existir uma ineficiência, em termos de quantidade de CLBs, dentro desta primeira versão do DynoPlace, devido à restrição de que todos os módulos têm uma altura fixa de 16 CLBs.

Voltando ao exemplo da Figura 41, para a obtenção da frequência máxima de operação do sistema deve-se analisar o projeto plano, já que os *scripts* da metodologia Early Access da Xilinx não proporcionam nos seus relatórios esta informação. Como o SDR tem 2 PRMs por PRR, então deve-se fazer 4 projetos planos para considerar todas as possibilidades:

- Arquitetura proposta com PE4a e PE5a
- Arquitetura proposta com PE4a e PE5b
- Arquitetura proposta com PE4b e PE5a
- Arquitetura proposta com PE4b e PE5b

Após a síntese dos 4 projetos planos, foi possível obter a medida de frequência máxima de operação no valor de 96MHz e a quantidade de CLBs obtida neste caso foi 1170 (valor menor que o 1238 apresentados na Tabela 9 devido a otimizações da lógica), porém sem significado prático frente ao resultado do DynoPlace. Pode-se considerar o valor da frequência obtido com a ferramenta ISE um valor aproximado, já que ela faz um posicionamento de modo automático distinto ao gerado pelo DynoPlace.

Caso se fizesse a implementação do projeto plano do sistema, a quantidade de PEs seria 11 ao invés de 9 e seria necessária a mudança da rede Hermes de 3x3 para 4x4, portanto, com 5 roteadores que só estariam fazendo tarefa de transmissão de dados de/para outros roteadores na rede, porém não conectados a um PE. Isto poderia implicar um aumento de 1278 CLBs (incremento em 109%), sem considerar a possível existência de 5 PEs adicionais de lógica simples.

Comparado a este resultados, a utilização da reconfiguração dinâmica permitiu obter uma redução de aproximadamente a metade da quantidade de CLBs,

o que significa permitir utilizar o DRFPGA Virtex-4LX25 ao invés do Virtex-4LX40 (4608 CLBs).

7. CONCLUSÕES E TRABALHOS FUTUROS

Os sistemas dinamicamente reconfiguráveis e suas aplicações em SoPC são ainda timidamente utilizados em produtos comerciais devido à falta de ferramentas que ajudem o projetista a automatizar a metodologia do Early Access. Faltam também metodologias para simular sistemas com tais características no tempo total de execução que permitam verificar o desempenho esperado do sistema. O presente trabalho procurou contribuir para suprir esta demanda com o aporte de duas metodologias, uma sobre a simulação de SDR e a outra sobre o posicionamento do SDR no DRFPGA.

As duas metodologias propostas no presente trabalho foram submetidas e aceitas para apresentação oral e publicação nos anais, nos congressos internacionais IBERCHIP 2010 e SPL 2010, respectivamente, com os nomes de “A Simulation Methodology for a NoC-Based Dynamically Reconfigurable System” e de “A Placement Tool for a Noc-Based Dynamically Reconfigurable System” respectivamente.

7.1.Conclusões

Após o desenvolvimento deste trabalho, podemos retirar as seguintes conclusões:

- Ferramentas de posicionamento que permitam aos projetistas desenvolver sistemas dinamicamente reconfiguráveis sem se ter conhecimento prévio das restrições dos dispositivos e da metodologia Early Access são uma ajuda fundamental dentro do fluxo de projeto. É possível, facilmente, se adaptar algoritmos para SDRs com NoC.

- A Ferramenta DynoPlace, que na sua primeira versão automatiza o posicionamento do conjunto dos módulos que pertencem ao SDR no DRFPGA Virtex-4LX25, mostrando bons resultados, porém indicando ainda uma série de ineficiências.
- O uso do algoritmo do DynoPlace pode incrementar consideravelmente a quantidade dos CLBs do projeto SDR (quase 50% segundo um dos exemplos), o que se deve ao fato de fixar a altura de todos os módulos em 16 CLBs, sendo vários dos apresentados na Tabela 9 de só 1 CLB de área.
- A ferramenta DynoPlace precisa carregar uns arquivos de texto, explicado na seção 6.1, que são criados pelo usuário, já que não se tem uma etapa prévia de pré-posicionamento, ou seja, há uma falta de automatização nesta tarefa, relativamente simples.
- O uso da reconfiguração dinâmica permitiu, na análise do autor, sobre a arquitetura do caso de estudo, uma redução de CLBs quase de aproximadamente 41% comparada ao caso de projeto plano, permitindo o uso do DRFPGA Virtex-4LX25 ao invés do Virtex-4LX40. Como referência, os preços dos mesmos são 330,00 e 416,25 dólares americanos, respectivamente¹⁶, permitindo uma economia de 26%.
- A metodologia de simulação por DCS se mostra bastante apropriada e direta para a validação de SDRs baseadas em NoCs. Infelizmente, também vários passos necessitam de intervenção manual, trazendo certa ineficiência ao processo.

¹⁶ Informação obtida o dia 25 de janeiro dos 2010 na página do fabricante: <http://www.xilinx.com>

7.2.Contribuições

Segundo a visão do autor, este trabalho contribui nos seguintes aspectos:

- Desenvolvimento de uma metodologia para posicionamento dos módulos que compõem os sistemas dinamicamente reconfiguráveis nos DRFPGAs cuja interface de comunicação baseia-se em redes intra-chip, o desenvolvimento da ferramenta própria chamada DynoPlace, que considera as restrições próprias do dispositivo além do posicionamento dos *bus-macros*. Com o uso da mesma se automatiza as etapas da metodologia Early Access (27) e permite aos projetistas projetar sistemas dinamicamente reconfiguráveis sem ter conhecimento prévio das restrições dos dispositivos e desta.
- Desenvolvimento de uma metodologia de simulação de sistemas dinamicamente reconfiguráveis, cuja interface de comunicação se baseia em redes intra-chip, no tempo de execução, baseados na técnica de chaveamento dinâmico de circuitos (DCS). Isto permite ao projetista testar o SDR antes da implementação do mesmo no dispositivo, podendo apresentar o tempo de reconfiguração dos PRMs.

7.3.Trabalhos futuros

Entre as atividades previstas para o futuro imediato e médio prazo, citam-se:

Atividades relacionadas com o sistema dinamicamente reconfigurável:

- Adaptar a rede intra-chip Hermes para permitir que o meio de comunicação seja reconfigurável ou utilizar outras que sejam como o DyNoC (17) (47), o CoNoChi (18) e a CuNoC (19) ou fazer uma própria.
- Analisar a potência para o SoPC desenvolvido, já que é um parâmetro muito importante para dispositivos portáteis e segundo a referência (48) durante o processo de reconfiguração o consumo de potência é elevado.

Atividades relacionadas com a ferramenta DynoPlace:

- Generalização da base de dados dos dispositivos do DynoPlace para trabalhar com todos os dispositivos das sub famílias LX, FX e SX para os DRFPGAs Virtex-4 e Virtex-5.
- Modificação da metodologia do DynoPlace para que possa automatizar as etapas da ferramenta PlanAhead da Xilinx além do Early Access.
- Modificar o algoritmo do DynoPlace para que os módulos a serem posicionados não tenham que ter uma dimensão específica, como o apresentado na referência (43), só os PRMs terão uma altura que será múltipla de 16CLBs.

- Modificar o algoritmo do DynoPlace para que o posicionamento dos módulos seja feito utilizando algoritmos que permitam diminuir os custos das interconexões (comunicação), como os apresentados nas referências (42) e (44).
- Modificar o algoritmo do DynoPlace para o uso de *bus-macros* que permitam a comunicação vertical entre módulos reconfiguráveis e fixos ou reconfiguráveis.
- Permitir que a ferramenta escolha o dispositivo certo (quantidade suficiente de CLBs, de multiplicadores, RAMB16, FIFO16 e DSP48) das subfamílias LX, FX ou SX com a qual o projetista deseja trabalhar para o Virtex-4 ou Virtex-5.

Atividades relacionadas com a metodologia proposta para a simulação de sistemas dinamicamente reconfiguráveis usando DCS:

- Desenvolver um conjunto de módulos de controladores de configuração e outro para as chaves de isolamento que sejam descritos de um modo geral para que o projetista só tenha que fixar os parâmetros genéricos para sua aplicação.
- Desenvolver uma ferramenta baseada na metodologia proposta na seção 4.4 para a simulação dos SDR no tempo de execução. A ferramenta deve também encarregar-se de posicionar as chaves de isolamento com os seus respectivos controladores de configuração no arquivo TOP do sistema. Isto permitirá validar o SDR antes da implementação do mesmo no dispositivo.

8. REFERÊNCIAS BIBLIOGRÁFICAS

1. HENKEL, J.; PARAMESWARAN, S. (Ed.). **Basics of Reconfigurable Computing**. Dordrecht: Springer, 2007. 525 p.
2. LYSAGHT, P.; DUNLOP, J. **Dynamic reconfiguration of field programmable gate arrays**. In: The 1993 International Workshop on Field-Programmable Logic and Applications, 1993, pp 82-94.
3. LYSAGHT, P.; STOCKWOOD, J. A Simulation Tool for Dynamically Reconfigurable Field Programmable Gate Arrays. In: **IEEE Transactions on Very Large Scale Integration (VLSI) Systems**, v. 4, n. 3, p. 381-390, 1996.
4. ZHANG, X.; NG, K. W. **A review of high-level synthesis for dynamically reconfigurable FPGAs**. In: Microprocessors and Microsystems, 24. 2000, pp 199-211.
5. SANCHEZ, E. et al. Static and dynamic configurable systems. In: **IEEE Transactions on computers**, v. n. 6, p. 556-564, 1999.
6. BECKER, J. et al. Dynamic partial FPGA exploitation. In: **Proceedings of the IEEE**, v. n. 2, p. 438-452, 2007.
7. MANET, P. et al. **RECOPS: Reconfiguring Programmable Devices for Military Hardware**. In: Design, automation and test in Europe, 2007, Nice, pp 994 - 999.
8. UPEGUI, A. et al. **An FPGA dynamically reconfigurable framework for modular robotics**. In: Workshop Proceedings of the 18th International Conference on Architecture of Computing Systems (ARCS 05). 2005, Innsbruck, pp 1-7.
9. HORTA, E. **Comutador ATM reconfiguravel dinamicamente a traves de hardware**. 2002. 94 p. Tese (Doutorado) - Escola Politécnica, Universidade de São Paulo, São Paulo, 2002.
10. MAJER, M. et al. The erlangen slot machine: a dynamically reconfigurable FPGA-based computer. In: **The Journal of VLSI Signal Processing**, v. 47, n. 1, p. 15-31, 2007.

11. VERA, G. A. **A dynamic arithmetic architecture: precision, power and performance considerations.** 2008. 117 p. Tese (Doctorado) -, Universidade de New Mexico, Albuquerque, 2008.

12. SEIXAS, J. L. **Aquarius - Uma plataforma para desenvolvimento de sistemas digitais dinamicamente reconfiguráveis.** 2007. 168 p. Dissertação (Mestrado) - Centro de Informática, Universidade Federal de Pernambuco, Recife, 2007.

13. PALMA, J. C. et al. **Core communication interface for FPGAs.** In: 15th Symposium on Integrated Circuits and Systems Design, 2002., 2002, pp 183-188.

14. HUEBNER, M.; BECKER, T.; BECKER, J. **Real-time LUT-based network topologies for dynamic and partial FPGA self-reconfiguration.** In: 17th Symposium on Integrated Circuits and Systems Design, 2004. SBCCI 2004., 2004, Pernambuco, pp 28-32.

15. WALDER, H.; PLATZNER, M. **A runtime environment for reconfigurable hardware operating systems.** In: 14th International Conference on Field Programmable Logic and Applications (FPL), 2004, pp 831-835.

16. AHMADINIA, A. et al. **A practical approach for circuit routing on dynamic reconfigurable devices.** In: The 16th IEEE International Workshop on Rapid System Prototyping (RSP), 2005, Montréal, pp 84- 90.

17. BOBDA, C. et al. **A Dynamic NoC Approach for Communication in Reconfigurable Devices.** In: 14th International Conference on Field Programmable Logic and Applications (FPL), 2004, Leuven, pp 1032-1036.

18. PIONTECK, T.; KOCH, R.; ALBRECHT, C. **Applying partial reconfiguration to networks on chips.** In: 16th International Conference on Field Programmable Logic and Applications (FPL), 2006, pp 155-160.

19. JOVANIVIC, S. et al. CuNoC: A dynamic scalable communication struture for dynamically reconfigurable FPGAs. In: **Microprocessors and Microsystems**, v. 33, n. 1, p. 24-36, 2008.

20. BENINI, L.; DE MICHELI, G. Net works on chips: A New SoC Paradigm. **Computer**, v. 35, n. 1, p. 70-78, 2002.

21. The official Bluetooth wireless info site. **Bluetooth**. Disponível em <<http://www.bluetooth.com/>>. Acesso em: 19 de janeiro 2009

22. BRAZIL-IP NETWORK. **BRAZIL-IP**. Disponível em <<http://www.brazilip.org.br/>>. Acesso em: 19 de janeiro 2009

23. FLÓREZ, M. J. S. **Estimativa de desempenho de uma NoC a partir de seu modelo em SystemC - TLM**. 2006. 172 p. Dissertação (Mestrado) - Escola Politécnica, Universidade de São Paulo, São Paulo, 2006.

24. MORAES, F. et al. HERMES: an infrastructure for low area overhead packet-switching networks on chip. In: **INTEGRATION, the VLSI journal**, v. 38, n. p. 69-73, 2004.

25. XILINX; **Virtex-II Pro and Virtex-II Pro X Platform FPGAs: Complete Data Sheet**. DS083, v. 4.7, November 5, 2007

26. XILINX; **Virtex-4 Family Overview** DS112, v. 3.0, September 28, 2007

27. XILINX; **Early Access Partial Reconfiguration User Guide**. UG208, v. 1.2, September 9, 2008

28. XILINX; **Two flows for partial reconfiguration: module based or difference based**. XAPP290, v. 1.2, September 9, 2004

29. BOBDA, C. **Introduction to Reconfigurable Computing**. Dordrecht: Springer, 2007. 359 p.

30. ROBINSON, D. Methods of exploiting simulation technology for simulating the timing of dynamically reconfigurable logic. In: **IEE Proceedings Computers and Digital Techniques**, v. 147, n. 3, p. 175-180, 2000.

31. BRUNELLI, L. **Abordagem para Redução de Complexidade de RNA usando Reconfiguração Dinâmica**. 2005. 169 p. Tese (Doutorado) - Centro de Ciências e Tecnologia, Universidade Federal de Campina Grande, Campina Grande, 2005.

32. KOJIMA, L. **Metodologia de projeto de sistemas dinamicamente reconfiguráveis**. 2007. p. Dissertação (Mestrado) - Escola Politécnica, Universidade de São Paulo, São Paulo, 2007.

33. DUATO, J.; YALAMANCHILI, S.; NI, L. **Interconnection Networks**. San Francisco: Elsevier, 2003. 600 p.
34. JANTSCH, A.; TENHUNEN, H. **Networks on Chip**. Dordrecht: Kluwer Academic Publishers, 2004. p.
35. SOARES, R. **Projeto e implementação de uma plataforma MP-SoC usando systemC**. 2006. 111 p. Dissertação (Mestrado) - Informática e Matemática Aplicada, Universidade Federal do Rio Grande do Norte, Natal, 2006.
36. PIONTECK, T. et al. **Communication Architectures for Dynamically Reconfigurable FPGA Designs**. In: IEEE International Parallel and Distributed Processing Symposium, 2007. IPDPS 2007., 2007, pp 1-8.
37. AHMADINIA, A. et al. **Design and implementation of reconfigurable multiple bus on chip (RMBoc)**. Erlangen: Universidade de Erlangen-Nuremberg, 8 de dezembro de 2004. (Co-Design-Report, 11-2004).
38. ELGINDY, H. A. et al. **RMB -A reconfigurable multiple bus network**. In: Second International Symposium on High-Performance Computer Architecture, 1996, pp 108-117.
39. DynaCORE. Disponível em <http://www12.informatik.uni-erlangen.de/spprr/colloquium06/DynaCore_Muenchen.pdf>. Acesso em: 21 de Fevereiro de 2009
40. JOVANOVIC, S. et al. **CuNoc: A scalable dynamic NoC for dynamically reconfigurable FPGAs**. In: 17th International Conference on Field Programmable Logic and Applications (FPL), 2007, pp 753-756.
41. TEICH, J.; FEKETE, S.; JOERG., S. Optimization of Dynamic Hardware Reconfigurations. In: **The Journal of Supercomputing**, v. 19, n. 1, p. 57-75, 2001.
42. AHMADINIA, A. et al. **A New Approach for On-line Placement on Reconfigurable Devices**. In: 18th International Parallel and Distributed Processing Symposium, 2004, pp 134-140.
43. TABERO, J. et al. **A low Fragmentation Heuristic for Task Placement in 2D RTR HW Management**. In: 14th International Conference on Field Programmable Logic and Applications (FPL), 2004, Leuven, pp 241-250.

44. AHMADINIA, A. et al. Optimal Free-Space Management and Routing-Conscious Dynamic Placement for Reconfigurable Devices. In: **IEEE Transactions on Computers**, v. 56, n. 5, p. 673-680, 2007.

45. HERMES. Disponível em
<<http://www.inf.pucrs.br/~gaph/Projects/Hermes/Hermes.html>>. Acesso em: 1 de março de 2009

46. RÚBIA, M. O.; SARAJANE, M. P. **Aplicação de Técnicas Evolutivas para Minimização do Makespan em Processadores Paralelos Idênticos**. In: I Workshop de Computação e Aplicações, 2006, Campo Grande, pp 5 - 10.

47. BOBDA, C.; AHMADINIA, A. Dynamic interconnection of reconfigurable modules on reconfigurable devices. **IEEE Design & Test of Computers**, v. 22, n. 5, p. 443-451, 2005.

48. BECKER, J.; HUEBNER, M.; ULLMANN, M. **Power Estimation and Power Measurement of Xilinx Virtex FPGAs: Trade-Offs and Limitations**. In: Brazilian Symposium on Integrated Circuits and Systems Design (SBCCI), 2003, pp 283 - 288.

49. XILINX; **Virtex-II Platform FPGAs: Complete Data Sheet**. DS03,1, v. 3.5, November 5, 2007

50. XILINX; **Virtex-II Platform FPGA User Guide**. UG002, v. 2.2, November 5, 2007

51. DESCHAMPS, J. P.; BIOUL, G. J. A.; SUTTER, G. D. **Synthesis of Arithmetic Circuits**. Wiley-Interscience, 2006. 556 p.

52. XILINX, I. **Virtex-4 Configuration User Guide**. . UG071, v. 1.11, June 9, 2009

53. XILINX; **Virtex-4 FPGA Data Sheet: DC and Switching Characteristics**. DS302, v. 3.3, June 6, 2008

54. PALMA, J. C. **Métodos de distribuição e conexão de IP cores para dispositivos programáveis FPGA**. 2001. 92 p. Dissertação (Mestrado) - Faculdade de Informática, Pontifícia Universidade Católica do Rio Grande do Sul, Porto Alegre, 2001.

APENDICE A – DRFPGA DA XILINX

Os primeiros dispositivos a suportarem a reconfiguração parcial foram fabricados pelas empresas National Semiconductors, Algotronix e Xilinx, gerando grande interesse no meio acadêmico, porém por não terem um adequado suporte de ferramentas de software, não obtiveram sucesso em nível comercial. Atualmente os líderes mundiais na fabricação e comercialização de DRFPGAS são as empresas Xilinx e Atmel, mas só a primeira disponibiliza dispositivos de grande porte, portanto o presente trabalho só abordará os DRFPGAs da Xilinx.

A.1.O dispositivo VirtexII

O DRFPGA VirtexII (49), (50), (51), apresentado na Figura 46 baseia-se em uma arquitetura simétrica regular que tem os seguintes elementos:

Figura 46 - Arquitetura do VirtexII (25).

- Blocos de entrada e saída (IOBs):

Os IOBs servem como interface entre os blocos lógico configuráveis (CLBs) e os pinos do componente que estão na periferia do circuito e incluem elementos de armazenamento, os quais podem ser configurados como *flip-flops* tipo D ou *latches*.

- Bloco lógico configurável (CLB):

Os CLBs são compostos por uma matriz de chaves (em inglês *switch matrix*) para se ter acesso às linhas de roteamento de propósito geral (em inglês, *general-purpose routing*); estão divididos em quatro fatias (em inglês, *slices*), conforme mostrado na Figura 47.

Figura 47 - Bloco lógico configurável (25).

Cada fatia tem 2 blocos de memória para implementação de funções, conhecidos na literatura como *Look Up Tables* (LUTs), com 4 entradas e uma saída que podem funcionar também como registradores de deslocamento de 16 bits ou como SelectRAM distribuída de 16 bits. Também tem lógica de *carry*, 2 registradores, multiplexadores e elementos de geração de lógica aritmética. Figura 48, ilustra os componentes de uma fatia.

Figura 48 - Slice do VirtexII (25).

- Blocos de memória (RAM blocks):

Este dispositivo inclui elementos de memória de 18kbits conhecidos como SelectRAM. Suportam configurações de porta simples e dupla. São programáveis para tamanhos de palavra de até 36 bits.

- Multiplicadores:

O dispositivo tem multiplicadores embutidos de (18x18) bits otimizados em relação à lógica das fatias, permitindo trabalhar com altas velocidades e baixo consumo de potência; isto permite um melhor desempenho para implementações de arquiteturas para processamento digital de sinais (DSP). Os multiplicadores podem atuar de modo independente ou associados a blocos SelectRAM.

- Recursos de relógio:

O dispositivo tem elementos gerenciadores digitais de sinal de relógio (DCM), tendo por finalidade eliminar os retardos de distribuição de relógio (*clock skew*), gerar várias frequências diferentes a partir do relógio de entrada, além de produzir sinais com deslocamentos em fase de 90, 180, 270 graus.

- Caminhos programáveis:

Adjacentes a cada CLB estão as matrizes de roteamento (GRM), que têm por objetivo proporcionar um caminho para que os CLBs acessem as linhas de roteamento de propósito geral (*general-purpose routing*).

A família Virtex-II apresenta diferentes modos para a configuração dos dispositivos, como serial (*Boundary Scan* ou JTAG), SelectMap, mestre-serial (*Master-Serial*), escravo-serial (*Slave-Serial*) e finalmente ICAP. A última forma, junto ao SelectMap, permite fazer um sistema auto reconfigurável¹⁷ no FPGA, já que provê meios de se fazer a leitura e escrita de bits de configuração (*bitstream*) no FPGA.

¹⁷ A reconfiguração do FPGA pode ser feita do seu interior por intermédio de lógica específica, por um processador *hard core* como o Power-PC ou um *soft core* como o MicroBlaze.

A.2. Dispositivo VirtexII-Pro

A diferença principal¹⁸ do dispositivo VirtexII-Pro (25) comparado ao VirtexII é a existência de núcleos de hardware do processador Power-PC¹⁹, conforme ilustrado na Figura 49.

Figura 49 - Arquitetura do VirtexII-Pro (25).

A.3. O dispositivo Virtex-4

Os dispositivos Virtex-4 (26) foram projetados para atender três classes de aplicação:

- Virtex-4 LX:
Projetados para alto desempenho em aplicações baseadas em lógica.
- Virtex-4SX:
Projetados para alto desempenho em aplicações DSP.

¹⁸ Outra diferença é que o VirtexII-Pro possui transceptores de dados seriais de alta velocidade (Gb/s) RocketIO MGT.

¹⁹ Dos dispositivos VirtexII-Pro o único que não tem embutido nenhum Power-PC é o XC2VP2.

- Virtex-4FX:

Projetados para alto desempenho em aplicações que incluem plataformas embutidas e conexões seriais de alta velocidade.

A descrição anterior sugere que os Virtex-4 LX/SX são evoluções da família VirtexII enquanto os Virtex-4 FX são uma evolução do VirtexII-Pro. Um resumo das características é apresentado na Figura 50.

Figura 50 - Diferenças entre os diversos dispositivos Virtex-4.

Os dispositivos Virtex-4 contêm as características apresentadas na seção 2.1, adicionados das seguintes funções:

- XtremeDSP Slices:

São elementos especialmente feitos para alcançar um alto desempenho nas operações aritméticas. Como ilustra a Figura 51 estes elementos têm: registradores de entrada e saída, multiplicadores de (18x18) bits em complemento de 2 com sinal, somador / subtrator de três entradas, multiplexadores. Ideais para projeto dos filtros em aplicações de DSP.

Figura 51 - XtremeDSP slices (26).

Além dos elementos anteriores os dispositivos Virtex-4 FX suportam:

- **RocketIO Multi-Gigabit Transceiver (MGT):**
São módulos transceptores de dados seriais de alta velocidade, acima de 6.5 Gb/s por canal.
- **PowerPC 405 RISC Core:**
Núcleo de processador PowerPC 405, RISC (frequência de 450 MHz), embutido.
- **Tri-Mode Ethernet Media Access Controller:**
O módulo opera a 10, 100 e 1000 Mb/s segundo IEEE 802.3.

APENDICE B – PARÂMETROS DE RECONFIGURAÇÃO DO DRFPGA VIRTEX-4 LX

Os dados apresentados na Tabela 11 foram obtidos da referência (52) e são parâmetros importantes para se obter o tempo de reconfiguração para o SDR a ser implementado no Virtex-4LX. Deve-se lembrar para evitar confusões, que *frames* aqui referem-se a *frames* da memória de configuração e não devem ser confundidos com os *frames* de reconfiguração mencionados na seção 2.2. Os primeiros referem-se aos bits de memória de configuração dos recursos da DRFPGA (que estão proximamente relacionados às palavras do *bitstream*) e os últimos é a menor área reconfigurável realizada pela reconfiguração dinâmica. Desta forma, os dados de reconfiguração no *bitstream* são compostos por dois componentes: os referentes aos *frames* usados na reconfiguração e as palavras de overhead de configuração usadas para controle. Cada *frame* de memória é formado por 41 palavras, como mostra a tabela e, cada palavra é formado de 4bytes.

Para completar as informações da família Virtex-4LX, os portos para a reconfiguração são o SelectMap e o ICAP, ambos de 4 Bytes. A frequência máxima para a entrada do bitstream (a reconfiguração mais rápida) é de 100 MHz (53).

Tabela 11 - Dados de configuração do Virtex-4

Dispositivo	Frames não usados para reconfiguração	Frames usados para reconfiguração	Frames por dispositivos	Palavras por Frame	Total de palavras de configuração do dispositivo	Overhead de Configuração (palavras)
XC4VLX15	140	3600	3740	41	147600	1312
XC4VLX25	234	5928	6162	41	243048	1312
XC4VLX40	376	9312	9688	41	381792	1312
XC4VLX60	536	13472	14008	41	552352	1312
XC4VLX80	710	17720	18430	41	726520	1312
XC4VLX100	948	23376	24324	41	958416	1312
XC4VLX160	1260	30720	31980	41	1259520	1312
XC4VLX200	1620	39120	40740	41	1603920	1312

APENDICE C – PROPOSTAS DE COMUNICAÇÃO DE SDR BASEADOS EM BARRAMENTOS

C.1.Proposta de Palma et al.

O sistema apresentado por Palma e outros (13), (54), de 2001, propunha uma interface de comunicação entre módulos fixos e reconfiguráveis. A Figura 52 apresenta o barramento para a comunicação, o árbitro do mesmo, o módulo mestre e até 3 módulos escravos. O sistema foi implementado na plataforma VW300 que tem um DRFPGA Virtex 300.

Figura 52 – Interface de comunicação proposta por Palma et al (13).

As partes fixas são somente o módulo mestre, o árbitro e o barramento. As partes reconfiguráveis são os módulos escravos. Para poder inserir um módulo dentro do módulo escravo se projetou uma lógica adaptadora.

Nesta proposta os módulos escravos têm suas áreas fixas, o que é um gasto desnecessário de recursos, já que nem sempre os módulos a serem realocados precisam toda essa área sendo sua modularidade baixa. Esta desvantagem é devida a que o antigo DRFPGA de modelo 1D tem a restrição da reconfiguração pela coluna inteira.

Para isolar as partes reconfiguráveis das fixas, os autores utilizaram os “virtual pins” que correspondem basicamente a um *bus-macro* baseadas nos *tri-states* buffers (TBUF) (ilustrado na Figura 52 na parte de conexão dos módulos escravos com as linhas de dados e de controle). Devido à restrição do número de TBUF (limitação dos FPGAs Virtex) e com o problema da ferramenta de *floorplanning* (naquela época) para fixar determinadas linhas que têm TBUF, os autores tiveram que limitar o número de TBUF utilizados e limitaram as linhas de dados (devido a que também se compartilham com as partes reconfiguráveis) então se utilizou no barramento a comunicação serial, em detrimento do desempenho da interface de comunicação o mesmo que é uma desvantagem com respeito a outras propostas. Outra desvantagem é sua baixa escalabilidade, já que o sistema não permite muitos módulos escravos por a limitação dos TBUF.

O módulo mestre faz as solicitações de comunicação ao árbitro do barramento, para o envio serial de operação e de dados (40 bits: 8 de endereço e 32 de dados) ao módulo escravo correspondente, assim como também faz a recepção dos resultado enquanto os módulos escravos ao atenderem as requisições do módulo mestre, respondem pelo barramento, solicitando o acesso ao árbitro de barramento. Portanto, sendo um meio compartilhado, o árbitro é o responsável para dar a todos os módulos o acesso ao barramento e evitar colisões.

Para o teste do sistema, utilizou-se um exemplo com módulos reconfiguráveis escravos de realização de operações aritméticas, porém com somente um módulo escravo por vez, o exemplo não foi testado propriamente como um SDR devido aos problemas já mencionados. Desta forma, no teste, o sistema teve que ficar suspenso até que o usuário fizesse a reconfiguração de modo manual do mesmo.

C.2.Proposta de Huebner et al.

O sistema de Huebner (6), (14) é uma proposta de 2004 para ter uma interface de comunicação entre módulos fixos e reconfiguráveis. A Figura 53 ilustra um SoPC autorreconfigurável formado pelo processador MicroBlaze (*softcore*), árbitro do barramento (*Arbiter*), barramento para a comunicação denominado pelos autores Bus-Macro²⁰, 4 módulos reconfiguráveis, uma memória Flash contendo os *bitstreams* da reconfiguração, uma unidade de descompactação e a interface ICAPS para acessar a memória de configuração do FPGA. O meio de comunicação Bus-Macro transporta 8 bits de dados e tem uma interface que permite isolar o módulo a ser reconfigurado. O sistema foi validado no DRFPGA VirtexII 3000.

Figura 53 – Diagrama de blocos do sistema de Huebner et al (14).

Nesta proposta os módulos reconfiguráveis têm suas áreas fixas, o que é um gasto desnecessário de recursos, já que nem sempre os módulos a serem realocados precisam toda essa área sendo sua modularidade baixa. Esta desvantagem é devida às restrições para reconfiguração que tem o antigo DRFPGA de modelo 1D.

Cada módulo reconfigurável tem associada uma interface chamada BusCom com a qual estabelece a comunicação com outros módulos, conectando-se diretamente ao barramento.

²⁰ Uma confusão comum é pensar que o termo Bus-Macro do Huebner é o mesmo do termo *bus-macro* da Xilinx.

Outro aspecto interessante deste trabalho foi que, na estrutura interna do Bus-Macro, que está ilustrada na Figura 54, se estabeleceu uma maneira de se isolar os módulos a serem reconfigurados, fazendo uso dos LUTs ao invés dos TBUF por problemas de roteamento da ferramenta de ISE da Xilinx, segundo os autores. A saber, atualmente, os componentes *bus-macro* fornecidos pela Xilinx são baseados nos LUTs ao invés dos TBUF daquela época.

Figura 54 – Bus-Macro, entradas (a) e saídas (b) ao Bus-Com (14).

O sistema não apresenta problema algum para estabelecer comunicação entre os módulos no instante de fazer a reconfiguração.

C.3.Proposta de Walder et al.

O sistema de Walder (15) é uma proposta de 2004, validado em um DRFPGA VirtexII 3000-4. A Figura 55 ilustra o sistema, onde o barramento fica na parte superior do dispositivo e a parte fixa consiste de 2 módulos que situam-se nos extremos do DRFPGA.

Figura 55 - Diagrama de blocos do sistema de Walder et al (15).

Ao iniciar o funcionamento, o sistema é composto somente dos módulos fixos e no lugar dos reconfiguráveis, tem-se módulos de tarefas nulas de largura $w=1$; estes módulos têm somente a função de estabelecer a estrutura de comunicação. Na parte da direita da Figura 55 está indicado que a largura dos módulos a serem inseridos (T_1 e T_2) pode ser variável, em múltiplos de w o que faz com que a sua modularidade seja superior e não desperdiça muita área para o uso dos módulos reconfiguráveis ao invés das duas propostas anteriores apesar de ser validado em um antigo DRFPGA de modelo 1D.

A Figura 56 ilustra com mais detalhes a interface de comunicação do sistema, que consiste de 3 barramentos: de dados para esquerda, de dados para direita e de controle; os barramentos de dados para esquerda e para direita estão divididos em dois grupos, uma vez que utilizam *bus-macros* baseados em TBUF, que eram só de