RECONFIGURAÇÃO PARCIAL E DINÂMICA PARA NÚCLEOS DE PROPRIEDADE INTELECTUAL COM INTERFACES DE COMUNICAÇÃO PADRONIZADAS

Eduardo Wenzel Brião, Daniel Camozzato, Luis Henrique Leal Ries,
Fernando Gehm Moraes e Ney Laert Vilar Calazans

Faculdade de Informática – PPGCC – PUCRS
Av. Ipiranga – Prédio 16 – térreo – Caixa Postal 1429 – CEP 90619-900

{briao, camozzato, ries, moraes, calazans}@inf.pucrs.br

SUMMARY

This work presents the use of pre-designed and pre-verified modules of hardware on the development and implementation of computation systems in a unique integrated circuit (SoC – System on Chip). Such modules named intellectual property cores, IP cores or simply cores, they can be inserted or removed in the system in run-time execution through dynamic and partial reconfiguration. It is proposed a framework for reconfiguration through of adaptation of a digital project development flow of a FPGA manufacturer called Modular Design. As second contribution, a tool has been developed that automate the implementation process and generation of reconfigurable dynamic systems using the flow and proposed framework. Through of the case studies, it was possible to become feasible the partial reconfigurable in a high level of abstraction, without the knowledge of architectural details of reconfigurable device and demonstrate that reconfiguration can be feasible compared with computation function implemented in software approach.

RESUMO

Este trabalho aborda a utilização de módulos de hardware, pré-projetados e pré-validados no desenvolvimento e implementação de sistemas computacionais em um único circuito integrado (SoC - *System-on-Chip*). Tais módulos, denominados núcleos de propriedade intelectual, núcleos IP, ou simplesmente *cores*, podem ser inseridos e removidos no sistema em tempo de execução através da reconfiguração parcial e dinâmica. É proposta uma infra-estrutura para reconfiguração através da adaptação de um fluxo de desenvolvimento de projetos digitais de um fabricante de FPGAs chamado Projeto Modular. Como contribuição secudária, foi desenvolvida uma ferramenta que automatiza o processo de implementação e geração de sistemas digitais reconfiguráveis utilizando o fluxo e infra-estrutura proposta. Através dos estudos de caso, foi possível tornar factível a reconfiguração parcial em um nível de abstração alto, sem o conhecimento de especificidades arquiteturais do dispositivo reconfigurável e demonstrar que a reconfiguração pode ser viável comparada com funções computacionais implementados em software.

RECONFIGURAÇÃO PARCIAL E DINÂMICA PARA NÚCLEOS DE PROPRIEDADE INTELECTUAL COM INTERFACES DE COMUNICAÇÃO PADRONIZADAS

Eduardo Wenzel Brião, Daniel Camozzato, Luis Henrique Real Ries,
Fernando Gehm Moraes e Ney Laert Vilar Calazans

Faculdade de Informática – PPGCC – PUCRS
Av. Ipiranga – Prédio 16 – térreo – Caixa Postal 1429 – CEP 90619-900

{briao, camozzato, ries, moraes,calazans}@inf.pucrs.br

RESUMO

Este trabalho aborda a utilização de módulos de hardware pré-caracterizados, pré-projetados e pré-validados no desenvolvimento e implementação de sistemas computacionais integrados em um único circuito integrado (SoC - *System-on-Chip*). Tais módulos, denominados núcleos de propriedade intelectual, núcleos IP, ou simplesmente *cores*, podem ser inseridos e removidos no sistema em tempo de execução através da reconfiguração parcial e dinâmica. É proposta uma infra-estrutura para reconfiguração através da adaptação de um fluxo de desenvolvimento de projetos digitais de um fabricante de FPGAs chamado Projeto Modular. Como contribuição secundária, foi desenvolvida uma ferramenta para implementação e geração de sistemas digitais reconfiguráveis utilizando o fluxo e infra-estrutura proposta.

1. INTRODUÇÃO

No desenvolvimento de funções computacionais, deseja-se sempre máximo desempenho e menor custo. Porém esta tarefa é complexa. Para diminuir o custo, o mais importante atualmente, é diminuir o tempo de projeto, embora existam outras características que são relevantes. Esta redução pode ser atenuada através do reuso de núcleos de hardware. Para obtenção do reuso destes núcleos, estes devem ser mais genéricos possíveis. Porém a generalidade implica em perda de desempenho ao realizar tarefas específicas

Duas formas de se obter generalidade são através de sistemas *programáveis* e sistemas *reconfiguráveis*. Os primeiros possuem a propriedade de executar *software* (características dos GPPs - *General Purpose Processor*), e os últimos alcançam generalidade ao permitir que o *hardware* seja alterado de forma dinâmica *reconfigurabilidade*. Um sistema que não possua nenhuma destas propriedades pode apresentar o máximo desempenho na execução de uma tarefa como no caso de ASICs (*Application-Specific Integrated Circuit*).

Um exemplo de dispositivos configuráveis/ reconfiguráveis são os FPGAs. FPGAs (*Field-Programmable Gate Arrays*) podem ter parte de seu hardware especializado para executar funções específicas (como em ASICs) e a configuração do mesmo pode ser modificada ao longo de um determinado tempo (características de GPPs).

O crescente avanço da tecnologia de implementação de circuitos integrados (CIs) viabiliza a construção de SoCs (*System-on-Chip*) [1]. SoCs podem ser desenvolvidos combinando módulos tais como FPGAs, ASICs e GPPs em um único CI. SoCs são compostos basicamente por núcleos de propriedade intelectual (também chamados de *cores* ou núcleos IP). Núcleos IP são módulos de hardware complexos pré-caracterizados e pré-validados [2]. Estes devem ser reaproveitáveis, tornando viável desenvolver SoCs em tempo reduzido, gerando produtos que podem levar menos tempo para chegar ao mercado.

Quando se emprega reconfiguração, a dificuldade de se obter o compromisso entre otimalidade e reutilizabilidade do núcleo IP quanto à diferentes critérios como potência, área, e desempenho pode ser reduzida, pois um núcleo IP, ao invés de ser parametrizável pode ser gerado em diversas versões. Isto garante a adaptabilidade do hardware ao cenário de uso, que pode assim, mesmo para uma aplicação específica, mudar dinamicamente.

De acordo com Paulson [4], uma das tendências da indústria de telefone celular é utilizar a reconfiguração em CIs adaptativos. Tradicionalmente, dispositivos precisam de um circuito integrado separado para trabalhar com cada padrão de comunicação e, variações do mesmo. Para prover a implementação destes padrões, é necessário adicionar circuitos para multiplexar/ demultiplexar

algumas funções destes padrões, o que ocupa maior espaço em silício, adicionando custos e aumentando a dissipação de potência. Com o uso de CIs adaptativos (reconfiguráveis), um software pode reconfigurá-los sob demanda.

Um fator que limita o reuso de núcleos é a falta de padronização das interfaces externas dos mesmos. Uma possível solução a este problema é a adoção de interfaces e protocolos padrão. Esta abordagem auxilia o projetista no desenvolvimento do seu núcleo, já que o mesmo concentra-se no projeto do IP, e não na forma como este interage com o restante do sistema e a qual meio de interconexão este será conectado. Dos padrões de comunicação existentes, OCP [3] têm a vantagem de ser um padrão aberto e por ser um padrão independente do sistema de interconexão (pode ser usado em diferentes sistemas de interconexões tais como barramentos ou redes).

Este documento está organizado da seguinte forma. A Seção II apresenta técnicas de reconfiguração parcial e dinâmica. A Seção III comenta sobre interfaces de comunicação padronizadas. A Seção IV descreve o fluxo do Projeto Modular, suas ferramentas, críticas e proposta de automação do fluxo. A Seção V apresenta estudos de caso de reconfiguração parcial utilizando Projeto Modular e OCP. Finalmente, a Seção VI conclui este documento.

2. RECONFIGURAÇÃO PARCIAL E DINÂMICA PARA DISPOSITIVOS XILINX VIRTEX

2.1 Reconfigurabilidade

Em um dispositivo ou sistema de hardware reconfigurável, uma *configuração* é um conjunto de bits que deve ser carregado em posições de uma *memória de controle* para determinar as funções e a estrutura de um hardware que se quer implementar.

Reconfiguração é o processo de alterar uma dada configuração de forma total ou parcial, mudando assim as funções desempenhadas pela estrutura do hardware. *Reconfiguração total* é uma configuração onde a memória de controle do dispositivo reconfigurável é inteiramente sobrescrita. *Reconfiguração parcial* é o processo de configuração onde a memória de controle do dispositivo é alterada apenas parcialmente.

A reconfigurabilidade pode contribuir para a economia de recursos: quando uma dada tarefa pode ser quebrada em várias fases, uma configuração diferente pode ser carregada para cada fase seqüencialmente operando de forma análoga à memória virtual em sistemas operacionais. Dessa forma, o tamanho do sistema pode ser menor que o necessário para implementar uma funcionalidade total, o que implica redução de custos e redução de área do dispositivo. Dentro deste contexto, um outro exemplo é o emprego da reconfigurabilidade em aplicações espaciais. Alterações indesejadas de funcionalidades de um determinado circuito no espaço provenientes de radiação são comuns e podem acarretar erros severos na funcionalidade de tais circuitos. Reconfigurabilidade pode ser usada para corrigir erros no circuito e torná-lo tolerante a falhas [5].

2.2 Classificações, Ferramentas e Técnicas de Reconfiguração Parcial e Dinâmica.

Estrin [6] desenvolveu um sistema reconfigurável chamado Sistema Computador Reestruturável (*Restructurable Computer System*) na década de 60. Este sistema baseia em um repertório de funções armazenadas em hardware. Se o processador utiliza uma determinada função, e esta está no repertório, então a função é executada em hardware, aumentando o desempenho desta em relação à mesma implementada no processador. Este foi um trabalho isolado. Cerca de vinte anos mais tarde, surgiram os primeiros dispositivos configuráveis comerciais.

Sistemas digitais reconfiguráveis podem ser classificados, de acordo com o número de dispositivos que se compõe, em duas grandes classes: sistemas reconfiguráveis em nível de circuito integrado e sistemas reconfiguráveis em nível de placa.

2.2.1 Reconfiguração em Nível de Circuito Integrado

Este tipo de reconfiguração parcial acontece sobre áreas de silício de um CI. A unidade fundamental para a reconfiguração parcial neste nível pode ser constituída por LUTs. FPGAs são por excelência sistemas deste tipo. Entre estes, pode-se citar as famílias ATMEL AT40K [7] e Xilinx Virtex [8].

Os FPGAs da família AT40K da ATMEL foram projetados para suportar reconfiguração parcial e dinâmica. Porém esta família de FPGAs da ATMEL suportam um máximo de 50 mil portas lógicas para implementação projetos reconfiguráveis, ou seja, o tamanho do FPGA é pequeno, comparado com o estado da arte em FPGAs de alta densidade, que excedem este valor em pelo menos 2 ordens de grandeza.

Mais recentemente, a *Xilinx* desenvolveu os FPGAs das famílias *Virtex* e *Virtex* II. A memória de configuração destas famílias pode ser vista como uma matriz bidimensional de bits. Estes bits são agrupados em quadros verticais de 1 bit de largura, e se estendem do topo à base do dispositivo. Um quadro é a unidade mínima de configuração, ou seja, é a menor porção de memória de configuração que pode ser lida ou escrita no FPGA. Quadros são lidos e escritos seqüencialmente, com endereços crescentes para cada operação. Como os

quadros podem ser lidos e escritos individualmente, é possível reconfigurar parcialmente esses dispositivos através da modificação desses quadros no dispositivo. Os elementos configuráveis são CLBs (*Configurable Logic Block*), BRAMs, IOBs e roteamento.

2.2.2 *Reconfiguração em Nível de Placa*

Este tipo de reconfiguração é realizada em sistemas implementados sobre uma ou mais placas de circuito impresso contendo vários componentes como processadores, memórias, UARTs, barramentos e FPGAs ou outros dispositivos programáveis.

Algumas propostas de arquiteturas reconfiguráveis que apresentam reconfiguração parcial em nível de placa são: GARP [9] , Splash 2 [10] e CHESS [11].

No escopo deste trabalho aborda-se apenas a reconfiguração em nível de circuito integrado usando como base FPGA Virtex II da Xilinx [12].

2.3 Técnicas para Reconfiguração Parcial

Nas Seções seguintes, serão apresentadas técnicas e ferramentas desenvolvidas na Academia e no setor industrial.

2.3.1 Small Bit Manipulations

Este método, descrito por Lim em [13], consiste em gerar arquivos de configuração (*bitstream*) parciais usando a diferença de um bitstream total e modificações feitas num arquivo que posteriormente é lido pela ferramenta geradora de bitstreams. A ferramenta de edição do layout do FPGA (FPGA *Editor*) permite a modificação das funções contidas em uma ou mais LUTs. Isto é útil para aplicações reconfiguráveis que necessitam de pequenas alterações pontuais.

2.3.2 Conjunto de Classes JBits

Jbits [14] é um conjunto de classes Java que fornecem uma *Application Peripheral Interface* (API) que permite manipular o arquivo de configuração das famílias de FPGAs Virtex e Virtex-E da Xilinx. Esta interface opera tanto em arquivos de configuração gerados pelas ferramentas de projeto da Xilinx quanto em arquivos de configuração lidos do hardware (*Readback*). Para que este conjunto de classes possa dar suporte à reconfiguração parcial e dinâmica, a API deve ser capaz de manipular informações físicas detalhadas a respeito da arquitetura do hardware, bem como elementos arquiteturais reconfiguráveis do dispositivo a ser configurado.

A limitação desta API é que todos os recursos e manipulação destes devem ser explicitados no código-fonte. Outra limitação é que o usuário deve estar bem familiarizado com a arquitetura-alvo, pois sem o conhecimento devido dos detalhes arquiteturais do dispositivo, pode-se danificar o FPGA.

2.3.3 Interface Padrão de Comunicação

Palma [16] propôs uma interface para interconectar núcleos IP reconfiguráveis possibilitando o desenvolvimento de sistemas dinamicamente reconfiguráveis. A porção de hardware fixo no FPGA, denominado *controlador*, é um barramento responsável pela comunicação com o mundo externo (pinos de entrada/saída do dispositivo) e pela comunicação com os núcleos IP da aplicação. O dispositivo reconfigurável deve ser inicialmente carregado apenas com o controlador. Os núcleos são carregados sob demanda.

A comunicação entre o controlador e os demais núcleos IP é feita através de pinos virtuais. Estes pinos são implementados através de *buffers tristate*.

Foi criada uma abordagem que utiliza duas camadas de *buffers*, uma camada pertencendo ao controlador e outra pertencendo ao núcleo IP, como mostra Figura 1. A conexão de núcleos IP ao controlador é feita por linhas comuns de roteamento.

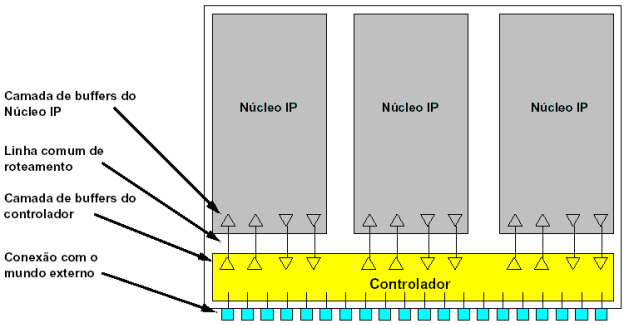

Figura 1 - Estrutura de um SoC usando o método proposto por Palma.

Junto ao controlador, no núcleo IP deve ser inserido em um invólucro (*wrapper*), contendo módulos de envio e recepção de dados para/do barramento, dependendo de solicitações externas.

2.3.4 Reconfiguração Parcial via Projeto Modular

Projeto Modular, em seu objetivo inicial, é utilizado para desenvolver projetos de forma distribuída. Um projetista é responsável pela manutenção e atualização do arquivo de nível hierárquico mais alto e os projetistas restantes são responsáveis pelos módulos que compõem o projeto. Estes são sintetizados separadamente. No final do fluxo do Projeto Modular, todos os módulos são reunidos em um único projeto junto ao módulo de maior nível hierárquico, resultando no projeto final, pronto para ser configurado no FPGA.

Este fluxo utiliza um conjunto de ferramentas do pacote ISE (*Integrated Software Environment*), responsáveis por mapear, posicionar, rotear e gerar arquivos de configuração, parciais ou totais. O Projeto Modular permite que um módulo seja modificado de forma independente, não afetando os demais módulos que compõe o projeto. Os módulos e o arquivo de maior nível hierárquico são sintetizados separadamente.

De forma alternativa, é possível gerar módulos reconfiguráveis [13]. A partir desta premissa, é possível configurar o FPGA com um arquivo de configuração total inicial e, logo após, configurar sucessivos arquivos de configuração parciais no FPGA, para a execução dinamicamente reconfigurável.

Com o uso do fluxo de Projeto Modular, o usuário não precisa se preocupar com vários detalhes inerentes à implementação do projeto. O nível de abstração do fluxo esconde especificidades arquiteturais do usuário, permitindo que este se detenha mais no desenvolvimento do projeto no nível HDL e arquivos de restrições. No entanto, existem diversas fases da execução do Projeto Modular efetuadas manualmente.

3. INTERFACES DE COMUNICAÇÃO PADRONIZADAS

As interfaces de comunicação padrão podem ser classificadas em abordagem centrada no meio de comunicação e abordagem centrada na interface de comunicação.

3.1 Abordagem Centrada no Meio de Comunicação (*bus-centric*)

Esta abordagem refere-se a núcleos IP dependentes do sistema de interconexão. Os núcleos devem seguir o mesmo protocolo do meio de comunicação ou serem adaptados para poderem acessar o meio de comunicação. A maioria das interfaces centradas no meio de comunicação são baseadas em barramentos. A Figura 2 mostra um modelo baseado nesta abordagem.

CoreConnect [17] é um exemplo de interface de comunicação baseada na arquitetura de barramento desenvolvida pela IBM que provê integração e reutilização de núcleos IP em SoCs. Como mostrado na Figura 3, esta interface é composta por um barramento de alto desempenho (PLB - *Processor Local Bus*), barramento periférico interno ao CI (OPB - *On-chip Peripheral Bus*), um circuito que une estes dois barramentos (*bus bridge*), e um barramento de registradores de controle de dispositivo (DCR - *Device Control Register*).

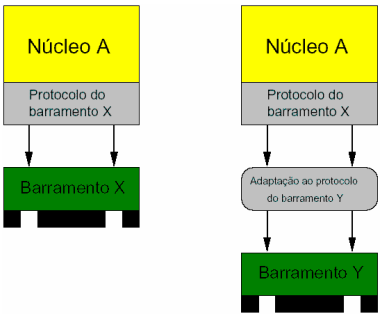

Figura 2 – Exemplo da desvantagem principal de comunicação baseada na abordagem centrada no meio de comunicação. Adaptações de um núcleo com um protocolo de comunicação diferente do barramento deve ser feita para conectar o núcleo ao barramento.

Núcleos de freqüência mais baixa são conectados ao OPB, reduzindo o tráfego no PLB, o que aumenta o desempenho geral do sistema. Existe um árbitro que decide qual núcleo IP vai fazer o acesso no barramento de cada vez. O PLB é usado para implementar comunicação de alto desempenho e baixa latência para núcleos IP integrados ao SoC. O OPB é um barramento de entrada e saída, criado para aliviar congestionamento de dados entre periféricos e o PLB, aumentando o desempenho do sistema como um todo.

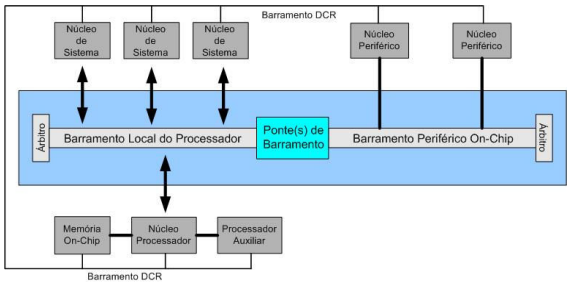

Figura 3 - Arquitetura de barramento *CoreConnect.*

Outro exemplo é o barramento AMBA, *Advanced Microcontroller bus Architecture* [18], o qual é um padrão aberto desenvolvido pela ARM Corp. O barramento de sistema (*System Bus*) interconecta processadores embarcados, controladores de DMA, memórias-on-chip e interfaces. Ele é um barramento de alto desempenho e de alta largura de banda que suporta gerenciamento de barramentos com vários módulos mestres para maximizar o desempenho do sistema.

O barramento periférico (*Peripheral Bus*) foi projetado para periféricos de propósito geral tais como UARTs (*Universal Asynchronous Receiver/Transmitter*), e portas de entrada e saída. Para a conexão do barramento de sistema com o barramento periférico, é necessário um circuito de adaptação (*bridge*).

3.2 Abordagem Centrada na Interface de Comunicação (*core-centric*)

Esta abordagem torna possível projetar e verificar núcleos IP como uma unidade independente do sistema de interconexão. Esta abordagem facilita o desenvolvimento de núcleos IP, já que o projetista irá se concentrar somente no desenvolvimento do núcleo, e não em como este núcleo interage com o ambiente. Assim, é possível integrar um núcleo a qualquer sistema sem a necessidade de alteração em termos de largura de banda, freqüência e sinais de interface deste núcleo. A Figura 4 ilustra a abordagem *core-centric* através de um exemplo de arquitetura.

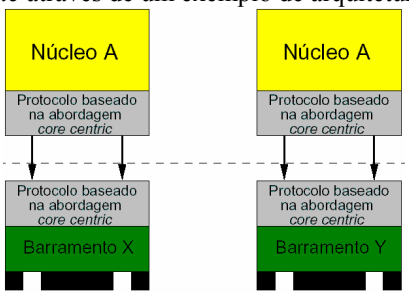

Figura 4 - Exemplo de arquitetura baseada na abordagem *core-centric*.

Entre os protocolos de comunicação baseados neste modelo cita-se OCP. O *Open Core Protocol* (OCP) é um protocolo de comunicação, que provê interface de comunicação entre núcleos no desenvolvimento de SoCs. OCP pode ser aplicado a qualquer tipo núcleo IP, inclusive barramentos.

Os núcleos IP são empacotados pelo protocolo. Desta forma, este protocolo é independente do sistema de interconexão adotado. O uso deste protocolo de comunicação minimiza problemas de conectividade entre núcleos IP, possibilitando o reuso de um núcleo IP de um determinado projeto. Desta forma, o uso do OCP torna mais simples a integração deste núcleo, tanto para o provedor quanto para o integrador do núcleo.

4. PROJETO MODULAR

Usando o fluxo do Projeto Modular descrito em [13], é possível desenvolver projetos com reconfiguração parcial e dinâmica. Projeto Modular, em seu objetivo inicial, é utilizado para desenvolver projetos de forma distribuída. Executando o Projeto Modular de forma alternativa, é possível desenvolver módulos reconfiguráveis sob forma de bitstreams parciais. Deste modo, é possível inicializar o FPGA com um bitstream e então fazer os sucessivos downloads de bitstreams parciais no FPGA.

Este fluxo foi usado para o desenvolvimento de sistemas dinamicamente reconfiguráveis pela geração de bitstreams parciais dos módulos reconfiguráveis em um maior nível de abstração do que as outras técnicas de reconfigurabilidade mostradas na Seção 2, sem o conhecimento de especificidades arquiteturais do FPGA, cujo conhecimento detalhado das mesmas é imprescindível no desenvolvimento de sistemas digitais reconfiguráveis na maioria das ferramentas para reconfiguração parcial e dinâmica.

O Projeto Modular é dividido em três fases: (i) *Fase do Orçamento Inicial* - nesta fase, determina-se qual é a estrutura do arquivo de nível hierárquico mais alto do projeto. Atribuições de restrições de área e temporização também são realizadas nesta fase; (ii) *Implementação dos Módulos Ativos* - cada módulo é sintetizado e implementado separadamente. Esta fase gerará bitstreams parciais de cada módulo usado na reconfiguração; (iii) Montagem Final - a terceira e última fase do Projeto Modular, une-se todos os módulos do projeto ao módulo de maior nível hierárquico (top) e realiza-se a síntese física gerando um ou mais arquivos de configuração total.

4.1 Preparação dos Módulos e Fases do Projeto Modular

4.1.1 Preparação da Entrada do Fluxo

Antes da execução do fluxo do Projeto Modular, é necessário definir os módulos *top* e os módulos que constituem o projeto. O módulo *top* (módulo de maior nível hierárquico) é apenas uma *casca* que une os módulos hierarquicamente inferiores. Os módulos são apenas instanciados no módulo *top* e conectados diretamente ou usando *bus macros*.

Bus macro é um componente (macro) pré-definido e pré-roteado, fornecido pelo fabricante, constituído por oito *buffers tristate* para assegurar a comunicação de 4 bits entre módulos fixos/reconfiguráveis e reconfiguráveis/ reconfiguráveis (Figura 5). O principal objetivo da bus macro é prover uma forma de controlar a interface de comunicação entre dois módulos, a partir de qualquer um dos módulos.

As instâncias das *bus macros* devem ser declaradas no módulo *top* para conexão dos módulos que compõe o projeto. *Buffers* e compensadores de atraso de propagação (DLLs) do sinal de *clock* devem ser instanciados no módulo *top* para que não ocorra escorregamento de relógio e dessincronização do circuito.

Depois da síntese dos módulos, deve-se organizar os arquivos de síntese gerados em diretórios recomendados pelo fabricante [13] para a execução do fluxo do Projeto Modular de modo que as ferramentas não possam sobrescrever dados de uma fase à outra, causando problemas na geração de arquivos com dados incorretos ao decorrer da execução do fluxo como um todo.

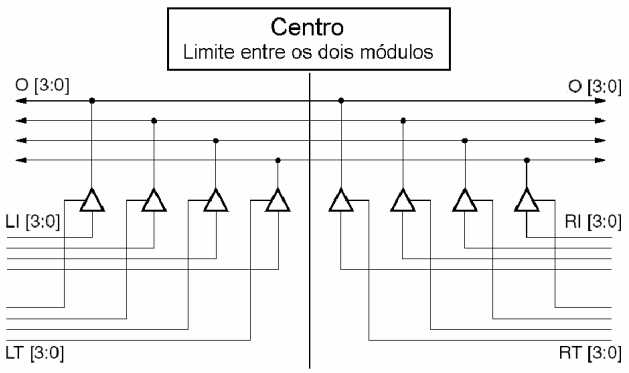

Figura 5 - Esquemático de uma bus macro, formada por 8 tristates.

Depois que todos os módulos são sintetizados inicia-se a execução do fluxo do Projeto Modular.

4.1.2 Ferramental para a Execução do Fluxo

Existem várias ferramentas para execução do fluxo disponibilizadas pela Xilinx. São elas:

- *NgdBuild:* Esta ferramenta aceita como entrada arquivos EDIFs, e como saída, gera arquivos de extensão NGO (arquivo intermediário que contém descrição lógica do projeto, preservando componentes originais e sua hierarquia interna), NGD (Xilinx Netlist Generic Database), UCFs (User Constraint File) e arquivos de componentes pré-roteados (arquivos NMC (Netlist Macros). Esta ferramenta converte um arquivo netlist de acordo com o modelo e fabricante do dispositivo reconfigurável, reduzindo todos os componentes em primitivas internas no formato da Xilinx e impondo restrições de acordo com o arquivo UCF, gerando um arquivo NGD;
- *Floorplanner:* Ferramenta que aceita como entrada um arquivo NGD para a realização da planta-baixa, ou seja, agrupa módulos por área e fixa em um determinado local pelo usuário componentes primitivos da Xilinx;
- *Map:* programa que mapeia uma descrição lógica, tais como blocos lógicos e componentes. Remove também lógica não usada, associa os PADs nos seus blocos lógicos de entrada e saída associados, mapeia a lógica dentro dos componentes (CLB, IOBs, etc). Esta ferramenta aceita como entrada arquivos NMC e NGD e como saída gera arquivo de restrições físicas de extensão PCF (Physical Constraints File) e também gera um arquivo NCD (Native Circuit Description). É gerado também um arquivo de extensão NGM, o qual contém informação do projeto físico produzido pelo MAP;
- *PAR (Place and Route*): o arquivo NCD criado pelo MAP é a entrada do PAR, junto com as restrições físicas (PCF). PAR realiza o roteamento e o posicionamento das linhas de conexão utilizando dois critérios básicos. O primeiro deles é baseado no custo. São utilizadas várias tabelas de pesos e custos para estimar comprimento de conexões, restrições impostas pelo PCF e recursos de roteamento disponíveis. Outro critério é a temporização. O roteamento e posicionamento são feitos seguindo a análise de restrições de tempo. O arquivo gerado pelo PAR é um NCD roteado e posicionado, pronto para ser utilizado pelo gerador de bitstreams;
- *FPGA Editor:* Ferramenta opcional utilizada para visualizar os componentes e o roteamento dentro do FPGA. Esta ferramenta permite também editar conteúdo dentro de LUTs e modificar o roteamento;
- *PIMCreate (Physically Implemented Modules Creator):* PIMs são projetos já roteados e posicionados. PIMCreator automaticamente copia os arquivos NGO, NGM e NGD para diretórios apropriados para a execução do Projeto Modular, na preparação na fase da Montagem Final. O utilitário também renomeia arquivos de acordo com necessidades do Projeto Modular.
- *BitGen:* Produz o bitstream para configuração de um dispositivo de configuração da Xilinx. A entrada desta ferramenta é um arquivo NCD roteado e posicionado gerado pelo PAR. Este tipo de arquivo contém todas as informações de configuração, que definem toda a lógica interna e interconexões, num dispositivo configurável específico. O bitstream pode, através de uma ferramenta de download, ser enviado para o FPGA para configuração.

4.1.3. Fase Orçamento Inicial

Primeiramente, determina-se o arquivo *top* que deverá ser inserido no projeto, bem como atribuições de restrições de área e temporização para cada módulo. Cada módulo reconfigurável do projeto será encapsulado por um módulo *top*. A fase do Orçamento Inicial é aplicada em todos os módulos *top*. Nesta fase são feitos: (a) definição de pinagem e restrições de tempo; (b) posicionamento de elementos para sincronismo de *clock*, *bus macros*, e fontes de alimentação das mesmas; (c) posicionamento dos módulos no FPGA (definição da planta-baixa).

A Figura 6 apresenta o fluxo de execução da fase do Orçamento Inicial.

A ferramenta NGDBuild é executada para gerar os arquivos necessários para as demais fases do Projeto Modular, posicionando toda a lógica do módulo *top* e agregando blocos que não foram expandidos que representam módulos. As entradas desta ferramenta são os arquivos EDIF já sintetizados, o arquivo que contém a *bus*

macro pré-roteada (NMC) e o arquivo de restrições que é gerado inicialmente pelo usuário e depois então é editada alternativamente pela ferramenta *Floorplanner*. Arquivos NGD e NGO são gerados, porém apenas este último será usado como entrada da próxima fase.

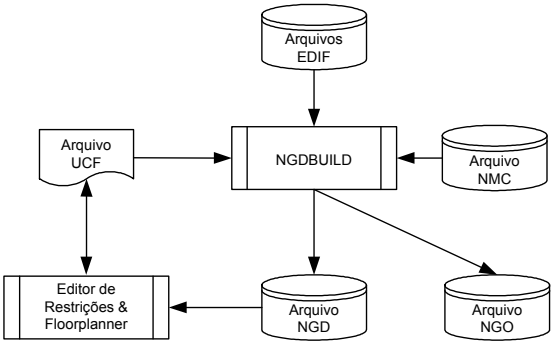

Figura 6 - Fluxo da fase do Orçamento Inicial do Projeto Modular.

A ferramenta *Floorplanner* é executada nesta fase para definição da planta-baixa, fixando os locais onde os módulos serão roteados e posicionados nas próximas fases do Projeto Modular. Os demais componentes (LUTs, *bus macros*, etc) também são fixados com o uso desta ferramenta. Esta ferramenta poderá ser reexecutada tantas vezes quantas forem necessárias para obtenção de melhores resultados como mostra a Figura 6.

4.1.4 Fase Implementação do Módulo Ativo

Na fase da Implementação do Módulo Ativo, ocorre o posicionamento, mapeamento, roteamento e a geração dos bitstreams parciais que representam cada módulo do projeto. O bitstream parcial gerado através de módulo reconfigurável será configurado no dispositivo configurável depois da execução do fluxo do Projeto Modular. O fluxo desta fase é apresentada na Figura 7.

As entradas deste fluxo são as mesmas utilizadas pela fase do Orçamento Inicial. Entretanto, usa-se mais um arquivo de entrada gerado na fase do Orçamento Inicial com extensão NGO. NGDBuild gera um arquivo que será usado como entrada na ferramenta de mapeamento (MAP). Esta gera restrições físicas e um arquivo NCD não roteado, que serão usados na ferramenta de posicionamento e roteamento (PAR). Finalmente, são gerados o bitstream parcial e uma estrutura de diretórios chamada PIM (*Physically Implemented Modules*), onde serão armazenados arquivos necessários para a execução da terceira e última fase do Projeto Modular.

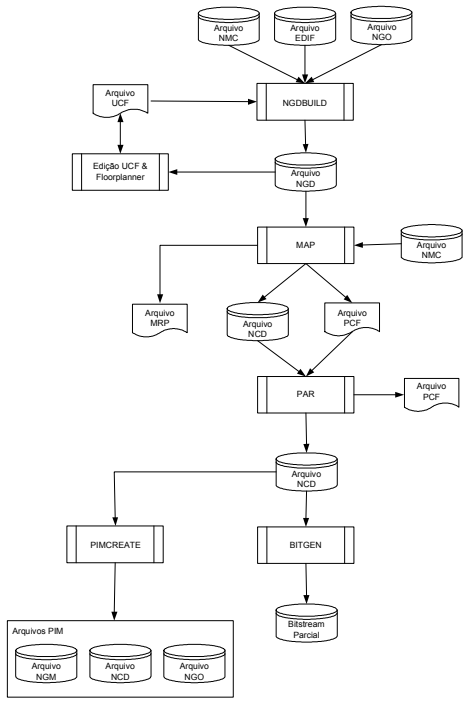

Figura 7 - Fluxo ferramental da fase Implementação do Módulo Ativo do Projeto Modular.

4.1.5 Fase Montagem Final

Finalmente, na fase Montagem Final, todos os módulos implementados fisicamente que fazem parte do módulo *top* são agregados ao mesmo. O posicionamento, mapeamento e roteamento são feitos de maneira unificada, anexando todos os módulos em um único projeto. O bitstream total do projeto final é criado nesta fase, pronto para ser prototipado.

De acordo com a Figura 8, a ferramenta NGDBuild lê o arquivo EDIF do módulo *top*, o arquivo de restrições, o arquivo onde estão contidas informações inerentes para implementar fisicamente a *bus macro* e todo o conjunto de arquivos PIM gerados na fase anterior. O restante deste fluxo da fase Montagem Final é semelhante ao fluxo da fase anterior, exceto pelo fato da ausência da execução da ferramenta *PIMCreator* para criar a estrutura de diretórios PIM. O bitstream total gerado nesta fase é resultado do produto final é já pode ser configurado no FPGA.

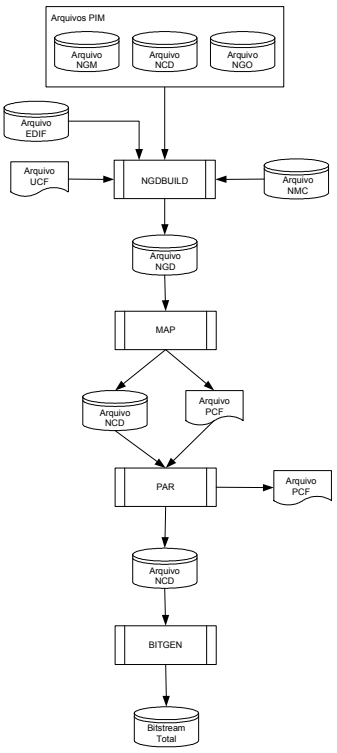

Figura 8 - Fluxo ferramental da fase Montagem Final do Projeto Modular.

4.2 Crítica do Processo

Apesar do fluxo do Projeto Modular conceber módulos reconfiguráveis sem a manipulação detalhada de bits, o fluxo mostrou-se com pouca automatização na execução de suas fases. A falta de automatização pode resultar em eventuais erros ao longo da execução do fluxo como um todo.

Além disso, o número de sinais que atravessam as bus macros é determinado pelo número de tristates em uma coluna inteira do FPGA. O número de bits dos sinais que atravessam as bus macros deve ser menor ou igual ao número de tristates numa dada coluna do FPGA. Este fator é limitante para a concepção de projetos integrados que são desenvolvidos através deste fluxo.

4.3 Proposta de Ferramental

Observando o padrão do fluxo do Projeto Modular, suas entradas e estruturas de diretório, uma ferramenta para automatizar o fluxo foi desenvolvida, da codificação até a geração dos bitstreams parciais e totais. A ferramenta, denominada *MDLauncher* (*Modular Design Launcher*), é capaz de executar praticamente todos os passos do fluxo. O usuário apenas informa quais módulos são reconfiguráveis, e disponibiliza os arquivos de restrição para o usuário (UCF) com a planta-baixa definida. A ferramenta também disponibiliza o recurso de edição e inclusão de bus macros na fase de codificação dos arquivos HDL de maneira automática. Dentro da ferramenta, pode-se escolher a ferramenta de síntese lógica (*Leonardo Spectrum*, XST, FPGA *Express*, etc). *MDLauncher* foi desenvolvido usando linguagem de programação JAVA.

5. ESTUDOS DE CASO E RESULTADOS OBTIDOS

Nesta Seção serão apresentados alguns estudos de caso que demonstraram o funcionamento de circuitos reconfiguráveis com a interface de comunicação OCP desenvolvidos através do fluxo do Projeto Modular.

5.1Estudo de Caso Simples

5.1.1 Calculadora

Foi desenvolvida utilizando dois núcleos IP inicialmente: um núcleo responsável pela captura dos dados programados nas chaves do FPGA pelo usuário e outro pela operação. O núcleo de captura é um núcleo fixo, ou seja, é um núcleo que não se reconfigura. Porém o núcleo de operação é reconfigurável aceitando dois tipos de cores: um para soma e outro de subtração.

A calculadora constitui-se de dois módulos reconfiguráveis assíncronos. Esta calculadora foi prototipada no FPGA e foram feitos dois testes: primeiramente carrega-se o dispositivo reconfigurável com um bitstream inicial de soma e logo após carrega-se o dispositivo com um módulo reconfigurável. Depois foi feito outro teste utilizando um bitstream inicial de subtração e então se carregou um módulo reconfigurável de soma. O circuito se comportou de maneira desejada. Neste primeiro estudo de caso, a reconfiguração parcial e dinâmica foi factível

5.1.2 Contador

O circuito contador realiza uma contagem de 0 a 9 em uma determinada freqüência. Existe um módulo fixo que captura os dados de entrada que é o passo (Δ) de contagem (contagem pode ser $t_1 \rightarrow 0 + \Delta$, $t_2 \rightarrow t_1 + \Delta$, $t_3 \rightarrow t_2 + \Delta$) e o módulo reconfigurável é responsável pela implementação do contador. No entanto, foram criados dois módulos reconfiguráveis: um contador de freqüência de 1 Hertz e outro contador de freqüência de 4 Hertz. Foram feitas sucessivas configurações no FPGA e os resultados foram corretos. Alternaram-se os dois módulos reconfiguráveis e os dois tiveram o comportamento desejado.

Este estudo de caso utiliza dois módulos reconfiguráveis síncronos.

Também foi desenvolvida uma versão deste contador

usando duas áreas reconfiguráveis, obtendo-se o comportamento desejado no momento da reconfiguração de cada uma das áreas separadamente.

5.1.3 Contador com Interface OCP

O contador mencionado acima foi implementado com o encapsulamento de protocolo de comunicação OCP entre o módulo fixo e o módulo reconfigurável. O protocolo foi validado através da ferramenta *CoreCreator*. O circuito foi prototipado e o comportamento dos dois cores foram desejados. Aqui foi feita a reconfiguração parcial e dinâmica usando núcleos IP com interfaces de comunicação padronizadas.

5.2 Processador Reconfigurável

Neste estudo de caso consiste em um processador *load-store* de 16 bits chamado de R8 [19]. O processador R8 reconfigurável (Figura 9) consiste em dois módulos: um módulo fixo chamado de R8 e um outro chamado de coprocessador reconfigurável. Para que o processador possa acessar o coprocessador reconfigurável, foram inseridas algumas instruções no processador R8 para inicialização, seleção, escrita e leitura do coprocessador reconfigurável. A parte fixa é subdividida em três módulos: processador R8, memória mapeada em BRAMs do FPGA e um módulo serial usado para interagir com o PC hospedeiro para carregar a memória mapeada com programas em código-nativo. Todos estes elementos foram interconectados a um sistema de barramento. Foram desenvolvidos 3 coprocessadores aritméticos reconfiguráveis (multiplicação, divisão e raiz quadrada) para execução e avaliação deste estudo de caso. Foi desenvolvido inicialmente um software para acessar e executar cada um dos coprocessadores. Para cada reconfiguração, o processador R8 bloqueia a sua execução, e o coprocessador entra no estado de execução. Quando o coprocessador termina sua execução, ele avisa o processador R8 para que este possa continuar a sua execução.

O tempo de reconfiguração foi medido e levou 10 ms [20] para reconfigurar um arquivo bitstream de 50kb. A Figura 10 mostra um gráfico comparativo entre os tempos de execução do software em relação ao número de operações e o tempo de reconfiguração e execução do coprocessador em relação também ao número de operações.

A partir de um certo número de operações, as retas se encontram. Isto significa que a partir de 750 operações, uma única reconfiguração parcial do coprocessador de multiplicação é viável comparando com o tempo de execução em software. Uma observação importante é que quando o software começou a ser executado, o coprocessador também começou a ser reconfigurado. A execução do coprocessador só é inicializada depois da reconfiguração (a partir do tempo de 10 ms).

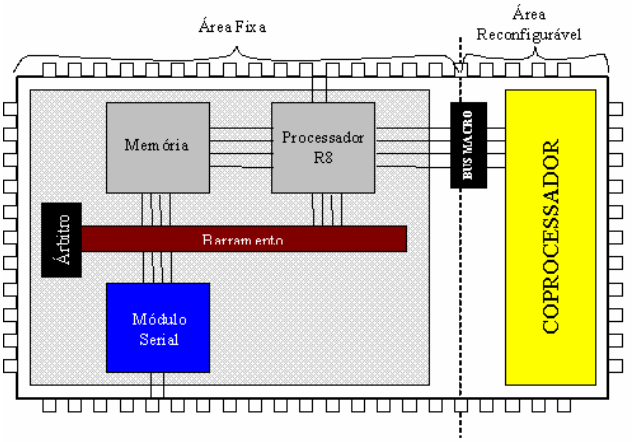

Figura 9 - Uma visão geral do sistema R8R disposto no FPGA

.

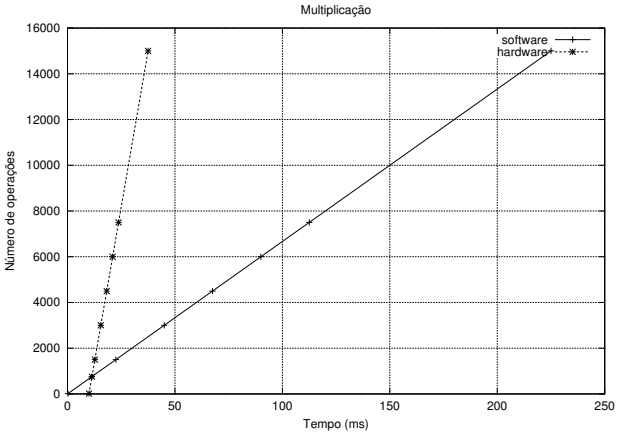

Figura 10 - Gráfico comparativo entre os tempos de reconfiguração e execução do coprocessador de multiplicação e o tempo de execução do software ambos em relação ao número de operações.

6. CONCLUSÕES

O fluxo do Projeto Modular ainda levanta uma série de questões que precisam ser resolvidas como: qual é o menor espaço definido na ferramenta *Floorplanner* para um determinado módulo; qual é o menor bitstream parcial que se pode extrair do fluxo; como diminuir de maneira eficiente os escorregamentos de relógio, os quais podem acarretar em problemas e dessincronização no circuito; como saber se um determinado circuito necessita de recursos adicionais (*buffers*, restrições de temporização) de modo a serem inseridos automaticamente pela ferramenta proposta (*MDLauncher*) e finalmente se é possível utilizar múltiplas áreas reconfiguráveis em sistemas razoavelmente complexos (um processador com várias áreas reconfiguráveis).

Dentro deste questionamento, no que se refere à interface de comunicação OCP, também existem questões que são levantadas e devem ser resolvidas como desempenho de um circuito reconfigurável com OCP no FPGA quanto à velocidade de execução e quanto à dissipação de potência; e finalmente como reduzir recursos gastos (buffers tristates) com a implantação de OCP no contexto da reconfiguração parcial e dinâmica utilizando Projeto Modular.

Através de todos os estudos de caso realizados, conclui-se que o fluxo do Projeto Modular pode ser usado para desenvolvimento de sistemas digitais reconfiguráveis num alto grau de abstração, e que a ferramenta proposta para automatização do fluxo diminui consideravelmente o tempo de projeto do sistema reconfigurável.

7. REFERÊNCIAS BIBLIOGRÁFICAS

[1] G. Martin and H. Chang. Tutorial – System on Chip Design. In *9th International Symposium on Integrated Circuits, Devices & Systems*, Singapore, September 2001.

[2] R. Bergamaschi and W. Lee. Designing System-on-Chip Using Cores. In *37th Design Automation Conference – DAC'00*, pages 420-425, 2000.

[3] OCP International Partnership. The Importance of Socket in SOC Design. OCP White Paper. Disp em http://www.ocpip.org/data/sockets_socdesign.pdf

[4] L. Paulson. Reconfiguring Wireless Phones with Adaptative Chips. Computer v.36,n.9, p9-11,2003.

[5] E. Bezerra; F. Vargas and M. Gough. Improving Reconfigurable Systems Reliability by Combining Periodical Test and Redundancy Technique: a case study. ***Journal of Eletronic Testing: Theory and Applications – JETTA***, v.17, n.3, p.701-711, 2001.

[6] G. Estrin. Parallel Processing in a Restructurable Computer System. ***IEEE Tranasctions on Computers***, v.EC-12, p.747-755, 1963.

[7] ATMEL AT40K Series Configuration. 1999. (Disp. Em http://www.atmel.com/dyn/resources/prod_documents/DOC1009.PDF).

[8] XILINX Inc. Virtex Series Configuration Architecture User Guide. 2001. (Disp. em http://www.xilinx.com/xapp/xapp151.pdf)

[9] S. Hauser and J. Wawrzynek. Garp: a mips processor with a reconfigurable coprocessor. In: *Proceedings of the IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM'97)*, 1997, USA.

[10] J. Arnold; D. Buell and E. Davis. Splash 2. In: *4th Annual ACM Symp. On Parallel Algorithms and Architecture*, 1992. p. 316-322.

[11] A. Marshall; T. Stansfield; I. Kostarnov; J. Vuillemin and B. Hutchings. A Reconfigurable Arithmetic Array for Multimedia Applications. In: Proceedings of the 1999 ACM/SIGDA 17th international symposium of FPGAs, 1999.

[12] XILINX, Inc. Virtex II Plataform Handbook. 2002. (Disp. em www.xilinx.com/publications/products/v2/handbook/ug002.pdf)

[13] D. Lim and M. Peattie. Two flows for partial reconfigurations: module based or small bit manipulations. 2002 (Xilinx Application Note – XAPP290).

[14] S. Guccione; D. Levi and P. Sundararajan, Jbits: a java-based interface for reconfigurable computing. In: Military and Aerospace Application of Programmable Devices and Technologies Conference – MAPLD, 1999.

[15] S. Mcmillan; S. Guccione. Partial run-time reconfiguration using JRTR In: *Proceedings of Field-Programmable Logic and Applications*, 1999

[16] J. Palma, Métodos para desenvolvimento e distribuição de IP-cores. 2002. (Dissertação de Mestrado, Pontifícia Universidade Católica do Rio Grande do Sul/Faculdade de Informática – PUCRS/FACIN, Porto Alegre, RS).

[17] IBM. The CoreConnect bus architecture. 1999. (Disp. em http://www.chips.ibm.com/products;coreconnect).

[18] ARM Corp. AMBA 2.0 Specification. 2000. (Disp. em http://www.arm.com/armtech/AMBA_Spec).

[19] F. Moraes and N. Calazans; R8 Processor – Architecture and Organization Specification and Design Guidelines, 2003. Disp. em http://www.inf.pucrs.br/~gaph/Projects/R8/public/R8_arq_spec_eng.pdf

[20] E. Carvalho, RSCM – Controlador de Configurações para Sistemas de Hardware Reconfigurável – Dissertação de Mestrado – Pontifícia Universidade Católica do Rio Grande do Sul/Faculdade de Informática – PUCRS/FACIN – Porto Alegre,RS)