Universidade Federal do Rio Grande do Norte
Centro de Tecnologia
Departamento de Engenharia de Computação e Automação
Programa de Pós-Graduação em Engenharia Elétrica

$CAD\mathbb{Z}$: Uma Metodologia de Projeto baseada em Z para Redes-em-Chip

Karla Darlene Nepomuceno Ramos

Orientador: Prof. Dr. Ivan Saraiva Silva

Tese de Doutorado apresentada ao Programa de Pós-Graduação em Engenharia Elétrica da UFRN (Área de concentração: Engenharia de Computação) como parte dos requisitos para obtenção do título de Doutora em Engenharia Elétrica.

Natal –RN, Brasil
Maio, 2007

CAD$\mathbb{Z}$: UMA METODOLOGIA DE PROJETO BASEADA EM Z PARA REDES-EM-CHIP

Karla Darlene Nepomuceno Ramos

Tese de Doutorado aprovada em 31 de maio de 2007 pela banca examinadora composta pelos seguintes membros:

Profa. Dra. Anamaria Martins Moreira, DIMAp/UFRN

Prof. Dr. Benjamín René Callejas Bedregal, DIMAp/UFRN

Profa. Dra. Cláudia Maria Fernandes de Araújo Ribeiro, DI/UERN

Prof. Dr. Ivan Saraiva Silva (Orientador), DIMAp/DCA/UFRN

Prof. Dr. José Antônio Gomes de Lima, DI/UFPB

Prof. Dr. Luiz Affonso Henderson Guedes de Oliveira, DCA/UFRN

Resumo

O rápido crescimento do mercado de sistemas embarcados e a crescente capacidade de integração de circuitos têm gerado muitos desafios para o projeto de sistemas de hardware e software integrados, oferecendo inúmeras oportunidades de pesquisa. O desempenho de um sistema embarcado complexo, o qual é caracterizado pela alta complexidade estrutural e diversidade funcional, está vinculada diretamente à infra-estrutura de comunicação. Nos últimos anos a Rede-em-Chip surgiu como estrutura de comunicação para interligar os componentes de sistemas completos inseridos em um único chip. Apesar de apresentar algumas características das redes tradicionais, o projeto de uma Rede-em-Chip não é uma tarefa trivial, pois lida com restrições peculiares dos sistemas embarcados que elevam o nível de complexidade do mesmo. É fundamental que o projetista de hardware disponha de metodologias e ferramentas CAD que o auxiliem no projeto de Redes-em-Chip para atender os requisitos de projeto face às exigências de tempo para mercado. Esta tese está situada exatamente nesse contexto e propõe a metodologia *CAD$\mathbb{Z}$* (*Computer Aided Design based on Z)*, a qual utiliza métodos formais como suporte à análise qualitativa de propriedades do sistema de comunicação, mesmo antes do desenvolvimento do próprio sistema. A metodologia *CAD$\mathbb{Z}$* deve ser utilizada nos estágios iniciais do projeto, sendo possível sua incorporação a um fluxo de projeto tradicional, onde a análise qualitativa pode ser seguida pela geração de código e análise quantitativa. A metodologia é validada mediante estudos de caso que comprovam a efetividade da mesma.

Abstract

The fast market growth of embedded systems and the growing capacity of circuits integration have created many challenges for the integrated hardware and software system design, offering countless research opportunities. The performance of a complex embedded system, which is characterized by the high structural complexity and functional diversity, it is related directly to the communication infrastructure. In the last years the Network-on-Chip emerged as interconnection structure to interconnect the components of complex embedded systems. Despite presenting some characteristics of the computer network, network-on-chip project is not a trivial task, because it deals with peculiar restrictions of the embedded systems that elevate the level of complexity of the same. It is ideal making use of methodologies and CAD tools that help the hardware designer on network-on-chip design to assist the requirements of its projects. This thesis proposes the methodology (Computer Aided Design based on Z), which uses formal methods as support to the qualitative analysis of properties of the communication system, even before the development of the own system. The methodology should be used in the initial phase of the project, being possible to be incorporated to a flow of traditional project, where the qualitative analysis can be followed by the code generation and quantitative analysis. The methodology is validated by studies cases that prove its effectiveness.

Sumário

Lista de Figuras

Lista de Abreviaturas

ABS	Anti-lock Breaking System
ACL2	A Computational Logic for Applicative Common Lisp
AMBA	Advanced Microcontroller Bus Architecture
ASIC	Application Specific Integrated Circuit
CAD	Computer Aided Design
CLB	Configurable Logic Block
DSM	Deep Sub-Micron
DSP	Digital Signal Processor
DRAM	Dynamic Random Access Memory
DTMF	Dual Tone MultiFrequential *MultiFrequential Code*-MFC
DVD	Digital Versatile Disc
FFT	Fast Fourier Transform
FIFO	First In, First Out
FLITS	Flow Control Digits
FPGA	Field Programmable Gate Array
GOPS	Giga Operations per Second
HDTV	High-Definition Television
IP	Intellectual Property
ISO	International Organization for Standardization
ITRS	International Technology Roadmap for Semiconductors
MFC	MultiFrequential Code
MID	Mobile Information Device
MOPS	Mega Operations per Second
MPEG	Moving Picture Experts Group
MPSoC	Multiprocessor System-on-Chip
NoC	Network-on-Chip
OSI	Open Systems Interconnection
PDA	Personal Digital Assistants
PVS	Prototype Verification System
RAW	Reconfigurable Architecture Workstation
RISC	Reduced Instruction Set Computer
SAF	Store-and-Forward
SoC	System-on-Chip
TOPS	Tera Operations per Second
VCT	Virtual Cut-Through
VDM	Vienna Development Method
VHDL	Very High Speed Integrated Circuit Hardware Description Language
VERILOG	Verifying Logic
VOPD	Video Object Plane Decoder

1. Introdução

Esta tese de doutorado trata da estrutura de comunicação para sistemas embarcados complexos. Nesta tese, considera-se um sistema embarcado complexo todo e qualquer sistema computacional completo implementado em um único circuito integrado, denominado em inglês *System-on-Chip* ou simplesmente SoC (Jerraya, 2004a). Uma estrutura SoC é basicamente composta por elementos de computação e armazenamento, que são conectados por elementos de comunicação (Goossens et al., 2005a).

É uma prática comum no desenvolvimento de SoCs o reuso de blocos pré-projetados e pré-verificados, denominados blocos de Propriedade Intelectual, em inglês Intellectual Property – IP, os quais podem ser: memórias, processadores e blocos de aplicação específica. Os elementos de comunicação por sua vez podem ser do tipo barramento ou Rede-em-Chip, em inglês *Network-on-Chip* – NoC (Guerrier, 2000; Jantsch, 2003). O barramento é o elemento de comunicação mais utilizado, porém as Redes-em-Chip são apontadas como a solução para suportar a diversidade funcional e a estrutura complexa dos SoCs. O *International Technology Roadmap for Semiconductors* – ITRS prevê a existência, até o final da década, de SoCs integrando de dezenas a centenas de elementos de computação, consumindo 4 bilhões de transistores no total (ITRS, 2003).

Tanto a academia quanto a indústria vêm empreendendo esforços no sentido do desenvolvimento de ferramentas computacionais, também conhecidas por *Computer Aided Design* – CAD e de metodologias de projeto para SoC. Porém, ainda existe muito espaço para contribuição, uma vez que o rápido crescimento do mercado de sistemas embarcados, por conseguinte de SoCs, tem gerado muitos desafios para o projeto de sistemas de hardware e software integrados, oferecendo inúmeras oportunidades de pesquisa.

Esta tese tem como foco principal o elemento de comunicação de Rede-em-Chip, mais precisamente o uso de métodos formais para analisar qualitativamente as propriedades do sistema de comunicação que precede o desenvolvimento do próprio sistema.

A contextualização desta tese, bem como a motivação que levou ao desenvolvimento da mesma, e ainda, a definição do problema, abordagens relacionadas ao projeto de elementos de comunicação para SoC, e a solução proposta serão abordadas nas seções que seguem.

1.1 Contexto

Os sistemas embarcados são projetados para executarem alguma tarefa específica. Diferente dos computadores de propósito geral, estes sistemas apresentam restrições de tempo-real, potência e custo. Atualmente, estão presentes em muitas atividades humanas e executam um papel importante na vida moderna, podendo ser encontrados em equipamentos eletrônicos, na indústria, em sistemas de controle de vôo, em equipamentos médicos, em sistemas automotivos e em sistemas robóticos.

A crescente capacidade de integração de circuitos tem motivado a integração de sistemas de computação completos, gerando um SoC (Jerraya, 2004a). Os SoCs oferecem sistemas cada vez menores, com maior poder de processamento e com menor consumo de energia. Considerando que aplicações diferentes apresentam requisitos distintos, o projeto de SoCs pode variar consideravelmente, impactando o trabalho de desenvolvimento do projetista, principalmente do ponto de vista da comunicação.

A comunicação entre os componentes de um sistema integrado, tradicionalmente, é realizada por meio de canais ponto-a-ponto dedicados ou canais multi-ponto compartilhados. Do ponto de vista do desempenho, os canais ponto-a-ponto são mais eficientes, pois cada comunicação ocorre independentemente das demais por meio de canais exclusivos. Entretanto, a reusabilidade dessa abordagem é limitada, uma vez que necessita de uma estrutura específica para cada projeto. A alternativa de canais multi-ponto compartilhados, também conhecidos por barramento, apresenta limitações quanto a escalabilidade da largura de banda e paralelismo de comunicação. Contudo, a estrutura do barramento agrega a vantagem de poder ser reutilizado em diferentes projetos reduzindo o tempo de desenvolvimento dos mesmos.

Mediante as restrições de reusabilidade, escalabilidade e largura de banda das abordagens de comunicação citadas, tendo em vista o aumento da capacidade de integração, permitindo um maior número de componentes em uma pastilha de silício, novos desafios relacionados à interconexão de componentes na arquitetura de um SoC foram introduzidos. Esse aspecto tem sido objeto de pesquisas tanto na academia como na indústria. Nos últimos anos a Rede-em-Chip surgiu como estrutura de comunicação para interligar os componentes de um SoC (Guerrier, 2000; Dally, 2001).

A Rede-em-Chip é uma estrutura de comunicação que tem requisitos de desempenho semelhantes àqueles das redes de computadores e *clusters* de computadores paralelos. Portanto, é natural que a Rede-em-Chip adote características e técnicas da área de redes e de *clusters* de processamento paralelo (Ye, 2004). Essas redes têm como vantagens: a largura de banda escalável, o uso de conexões ponto-a-ponto de curta distância e o paralelismo na comunicação. Apesar de apresentar algumas desvantagens como: maiores custos e limitações quanto ao consumo de energia, esses problemas tendem a ser resolvidos, ou atenuados, com o crescente avanço da tecnologia de integração, com transistores de canais dimensionados a menos de um micro-metro e a adoção de soluções arquiteturais (Zeferino, 2003a).

Uma típica arquitetura de Rede-em-Chip é composta de roteadores e canais físicos (em inglês, *links*) entre roteadores, fornecendo uma infra-estrutura de comunicação escalável que pode suportar múltiplos fluxos de dados concorrentes (Benini, 2004). Esta estrutura conduz a um alto nível de paralelismo quando os canais físicos podem operar simultaneamente. Por exemplo, mais de uma comunicação pode ser realizada em paralelo, desde que sejam utilizadas rotas disjuntas. A comunicação é completamente independente, resultando em um melhor desempenho da rede como um todo.

Apesar da Rede-em-Chip apresentar algumas características similares às redes tradicionais, seu projeto não é uma tarefa trivial, visto que as peculiaridades de sistemas embarcados devem ser consideradas. É indubitável que restrições de espaço e consumo de energia adicionam complexidade ao tratamento de aspectos comuns como capacidade de armazenamento, controle de fluxo e técnicas de roteamento. Adicionalmente, situações indesejadas tais como *deadlock, starvation* e *livelock*, bem como qualidade de serviço (em inglês, *Quality of Service-QoS*) também devem ser

consideradas (Benini, 2002; Ye, 2004). Todos esses aspectos citados tornam clara a necessidade de suporte automático para o processo de projetos de Redes-em-Chip.

Tradicionalmente, os projetistas de hardware utilizam ferramentas de simulação durante as fases iniciais do desenvolvimento da arquitetura (Edwards, 2000). Um dos principais benefícios desse tipo de ferramenta está relacionado à análise quantitativa, tais como as avaliações da área utilizada, da performance e do consumo de energia. Entretanto, o processo de simulação está comumente limitado à geração de teste automático ou simulação baseada na aplicação. A conseqüência prática da aplicação desse método é a dependência de análise sobre um conjunto pré-definido de vetores de testes dentro do espaço de simulação.

O método qualitativo está baseado no uso de técnicas formais para projeto de hardware (Rushby, 1993; Goossens, 2005b; Schmaltz, 2005). A principal vantagem de métodos formais é a possibilidade de provar a validade dos requisitos essenciais do projeto. Como a linguagem formal tem fundamento matemático, é possível formalmente extrair e verificar as propriedades desejadas do espaço de estado abstrato. Além disso, como o método formal também é incremental, a complexidade de um dado projeto pode ser administrada pelo uso de estratégias de refinamento sucessivo. Por exemplo, para avaliar os aspectos de interconexão entre os componentes de um SoC, detalhes da arquitetura específica podem ser considerados irrelevantes durante uma especificação formal (abstrata) preliminar.

A natureza qualitativa de uma análise formal é definida pela avaliação das propriedades do sistema que precede o desenvolvimento do próprio sistema, isto é, antes da geração do código. Considerando a alta complexidade dos SoCs, principalmente aqueles baseados em Redes-em-Chip, o poder de abstração de métodos formais representa uma ferramenta valiosa, contribuindo para a minimização de erros nas fases iniciais do projeto. Essa minimização de erros influencia diretamente na redução de custo do projeto e na redução do tempo para o mercado, do inglês *time-to-market*. No caso específico de Redes-em-Chip, as características dos métodos formais também beneficiam fortemente o processo de desenvolvimento do projeto, melhorando a capacidade de reuso e conseqüentemente a qualidade dos futuros produtos.

1.2 Motivação

O grau de complexidade dos projetos de sistemas computacionais embarcados é extremamente alto, uma vez que envolve conceitos pouco analisados pela computação de propósito geral, tais como: portabilidade, consumo de energia sem perda de desempenho, baixa disponibilidade de memória, necessidade de segurança, confiabilidade e curto tempo de projeto. Portanto, evidencia-se a necessidade de adoção de novas técnicas e recursos adequados e precisos, a fim de atingir os objetivos impostos pelo avanço tecnológico.

Outro aspecto que contribui para a adoção de novos paradigmas de projeto é o fato das soluções tecnológicas atuais estarem cada vez mais direcionadas à aplicação. Por exemplo, os sistemas projetados para ambiente inteligente (AmI), que é considerado o novo paradigma para a eletrônica de consumo, são baseados em processamento de sinais digitais de alta velocidade com uma carga computacional na ordem de 10 MOPS para processamento de áudio, 3 GOPS para processamento de vídeo, 20 GOPS para interfaces de conversação entre várias línguas e até 1 TOPS para geração de vídeo sintético (Boekhorst, 2002). O poder computacional exigido por este tipo de sistema não pode ser suprido por apenas um único processador em um circuito integrado, mas por vários. Desta forma, a solução é o desenvolvimento de Multiprocessadores em um único circuito integrado, ou seja, o uso de MPSoC, do inglês *Multi-Processor System-on-Chip* (Cesário, 2002; Bertozzi, 2005).

A performance de um MPSoC, que é caracterizado por alta complexidade estrutural e diversidade funcional, está vinculada diretamente à infra-estrutura de comunicação, que exerce a função de acomodar adequadamente os recursos dos processadores que integram o circuito integrado. Considerando a complexidade da estrutura MPSoC, que pode envolver até dezenas ou centenas de processadores heterogêneos, diferentes estruturas de memória e componentes de aplicação específica, a utilização da estrutura de barramento para realizar a comunicação entre esses componentes não seria adequada. O barramento dentre outras restrições é inerentemente não escalável. Contudo, a Rede-em-Chip apresenta características de modularidade e escalabilidade que suportam o crescente nível de integração dos SoCs e MPSoCs.

Apesar das Redes-em-Chip poderem adotar conceitos e metodologias das redes de computadores, existem vantagens e limitações da tecnologia sub-micrônica (em inglês, *Deep Sub-Micron* – DSM) que precisam ser consideradas no desenvolvimento das arquiteturas e protocolos de tais redes. Além disso, os SoCs deparam-se com requisitos de tempo-real que tornam extremamente necessária a previsibilidade na utilização dos componentes de comunicação, computação e armazenamento a fim de poder construir um sistema integrado de tempo-real.

É fundamental que o projetista de hardware disponha de metodologias e ferramentas CAD que o auxiliem no projeto de Redes-em-Chip para atender aos requisitos de projeto do SoC, face às exigências de tempo para mercado. Apesar de já existirem muitas pesquisas nesse sentido, ainda são escassos os trabalhos que tratam de uma metodologia completa para projeto de Redes-em-Chip e ferramentas CAD (Bertozzi, 2005).

É interessante que o projetista de uma estrutura complexa de comunicação, como no caso de uma Rede-em-Chip, possa, nas fases iniciais do projeto, analisar propriedades vitais do aspecto da comunicação a fim de reduzir os custos com verificação nas fases posteriores do projeto. Juergen Ruf em (Ruf, 2006), afirma que com o crescimento da capacidade dos circuitos integrados, a fase de verificação tornou-se o maior gargalo nos custos de projeto chegando a consumir até 80% (oitenta por cento) do custo total do mesmo.

1.3 Definição do Problema

A contínua evolução tecnológica impõe crescentes desafios ao projetista de Chip. Esses desafios envolvem aspectos da estrutura física de interconexão, sincronização, integridade do sinal, consumo de energia, desempenho, comunicação e, conseqüentemente, produtividade.

Apesar dos desafios citados, o ciclo de projeto de chip entre 1997 e 2002 foi reduzido em 50% devido à demanda de mercado, enquanto a complexidade de projeto aumentou

50 vezes no mesmo período (Bjerregaard, 2006). A densidade dos transistores contribui para o aumento da complexidade do projeto de um SoC, pois favorece a inserção de mais componentes permitindo o desenvolvimento de novas aplicações, que possivelmente serão executadas concorrentemente.

Segundo Luca Benini (Benini, 2004), o maior desafio de um projeto MPSoC é o fornecimento da funcionalidade correta, que depende dentre outros fatores, da comunicação *on-chip*. Para Jian Liu (Liu, 2003), dentre os desafios impostos pela tecnologia DSM a comunicação tem se apresentado como a maior restrição de projeto.

Considerando todos os aspectos abordados acima, os projetistas de Chip têm buscado soluções para conectar diferentes blocos IP para otimizar e controlar as propriedades elétricas, a fim de evitar interferência entre sinais (*cross-talk*), reduzir a dissipação de energia, aumentar a velocidade de propagação de sinais e compartilhar os recursos de interconexão entre muitos fluxos de comunicação, tornando mais eficiente o uso da estrutura física de interconexão.

Devido à complexidade estrutural e a diversidade funcional dos SoCs, várias Redes-em-Chip foram desenvolvidas para auxiliar o projeto de circuitos integrados. As Redes-em-Chip propostas buscam estruturar e gerenciar a estrutura de interconexão na tecnologia DSM; permitir uma melhor utilização dessa estrutura por meio do compartilhamento; escalar melhor o barramento; oferecer um consumo de energia confiável e eficiente; separar a computação da comunicação por meio de interfaces bem definidas, permitindo que os blocos IPs e a interconexão sejam projetados de forma isolada e integrados mais facilmente (Radulescu et al., 2005; Ramos, 2004).

Enquanto existem ferramentas completamente desenvolvidas e consolidadas (Lyonnard, 2001; O'Nils, 2001; Synopsys, 2006; Mentor, 2006) para o projeto de blocos IPs (processadores, memórias, componentes de aplicação específica) que agilizam o projeto de um SoC, o projeto dos elementos de comunicação de um SoC ainda consiste em uma área de pesquisa (Goossens, 2005a). Seria desejável e essencial que existissem ferramentas para processos de síntese automática de hardware para Redes-em-Chip, a fim de tornar os projetos de SoCs mais rápidos e eficientes, no entanto o que existe são ferramentas *ad hoc* (Kumar, 2002; Hu, 2004; Ost, 2005).

Do mesmo modo que ainda se busca solução para síntese automática de hardware de Redes-em-Chip, pesquisa-se também sobre novas metodologias de projeto (Kumar, 2002; Siegmund, 2003; Jalabert, 2004). Uma variedade de metodologias surgiu ao longo dos últimos anos. Essas metodologias tipicamente utilizam a prototipagem e a simulação como instrumento de validação do projeto.

Apesar de alguns desenvolvimentos de Redes-em-Chip utilizarem a prototipagem como forma de validação (Holsmark, 2003; Lee, 2006), essa técnica é cara em relação ao tempo gasto para o desenvolvimento do projeto, e só pode ser executada após definição de muitos detalhes de projeto.

A técnica de simulação é uma técnica numérica que permite uma análise quantitativa, como a avaliação da área utilizada, da performance e do consumo de energia. Apesar da validação por simulação ser um recurso bastante utilizado pelos projetistas (Siguenza-Tortosa, 2002; Madsen, 2003; Pestana, 2004), esse recurso apresenta algumas restrições (Carro, 2003). A principal delas é o grande número de casos de testes necessário para validação exaustiva da descrição de um sistema, fato que obriga o projetista pragmaticamente a limitar-se a uma cobertura parcial dos mesmos. Outra restrição é em relação ao nível de abstração, ou seja, quanto mais baixo o nível de abstração, maior o número de casos de teste, uma vez que o detalhamento da descrição será maior, e, por conseguinte, mais lenta será a simulação.

Um trabalho de simulação de um dia pode cobrir somente milissegundos do tempo real. Deste modo, um projeto validado por simulação corre o risco de somente após a obtenção do produto final, os erros e as omissões serem detectados (Vahid, 2001). Esses problemas despendem tempo e custo de fabricação. Portanto, é notória a necessidade de técnicas de análise e validação que possam cobrir todos os casos de testes, a fim de garantir um produto final de acordo com as especificações funcionais definidas nas fases iniciais do projeto.

O interesse por métodos para projetos de Redes-em-Chip, que resolvam os problemas de confiabilidade e tempo para mercado, tem crescido recentemente. Considerando: (1) a necessidade de novos métodos que agilizem e assegurem o processo de desenvolvimento de Redes-em-Chip; (2) a necessidade de evitar o desenvolvimento de

metodologias e ferramentas particulares de projeto; (3) e a necessidade de recursos que auxiliem o projetista a analisar qualitativamente os requisitos funcionais da estrutura de comunicação nas fases iniciais do projeto, esta tese trata do problema das restrições das técnicas de simulação, que consiste na detecção de erros e problemas somente nas fases finais de desenvolvimento de projeto.

A solução para o problema de simulações exaustivas e lentas, que utilizam um conjunto pré-definido de vetores de testes que não abrangem todos os possíveis casos, é a utilização de métodos formais. Por meio da utilização de métodos formais é possível realizar uma verificação abrangente do comportamento do sistema em um alto nível de abstração a fim de reduzir os riscos de obter um produto fora das especificações.

1.4 Abordagens Relacionadas ao Projeto de Elementos de Comunicação para SoC

Várias abordagens para o desenvolvimento de Redes-em-Chip já foram apresentadas tanto pela academia como pela indústria. As seguintes Redes-em-Chip são exemplos da utilização dessas abordagens: SPIN (Adriahantenaina *et al*., 2003), Nostrum (Millberg, 2004), Cliché (Kumar, 2002), QNoC (Bolotin, 2004), Æthereal (Goossens *et al*., 2002), aSoC (Liang et *al*. 2000, 2004) entre outras (Dally, 2001; Karim, 2002; Wiklund, 2003).

As implementações de Redes-em-Chip, geralmente, adotam metodologias particulares e utilizam as linguagens VHDL e VERILOG para a descrição de hardware. A inexistência de uma linguagem que permita a descrição de funções a serem implementadas em hardware ou software de forma imparcial em relação a sua implementação, é que ocasiona o uso de várias linguagens para o desenvolvimento de sistemas embarcados (Carro, 2003).

Apesar da linguagem C permitir a geração de software para uma variedade de processadores, a mesma não oferece o grau de abstração necessário a uma linguagem de especificação. Do mesmo modo, a linguagem C++ além de não dispor dos recursos de uma linguagem de especificação, ainda guarda restrições com relação ao tamanho do

software. Outro aspecto que torna as linguagens C e C++ ineficazes para todo projeto do sistema é a inadequação semântica das mesmas para a descrição de aspectos de hardware.

Com relação às linguagens de descrição de hardware, VHDL e VERILOG, estas, por meio de ferramentas comerciais, têm a característica de serem utilizadas como entrada para simulação e síntese automática de circuitos. Entretanto, a semântica dessas linguagens está orientada apenas para a descrição de hardware, não sendo adequado utilizá-las para a descrição de software.

A fim de combinar as vantagens da linguagem C++ com uma semântica para descrição de hardware, foi criada a linguagem SystemC (SystemC, 2006), a qual provê descrição de hardware, a partir de uma biblioteca de funções, dentro do contexto de C++. Apesar dos esforços empreendidos, nenhuma dessas linguagens atende simultaneamente os requisitos de cobertura de múltiplos domínios e níveis de abstração em relação a implementações de software e hardware (Carro, 2003).

As linguagens citadas acima são orientadas à simulação. Portanto, quando utilizadas no projeto de SoCs, exigem na fase de verificação do sistema que o sistema completo, inclusive a estrutura de interconexão, seja simulado. Como já foi comentado, a técnica de simulação apresenta limitações e por meio dela não é possível garantir a cobertura do pior caso para todas as configurações do sistema, uma vez que é utilizada como entrada uma dada amostra.

Os possíveis problemas que surgem durante a simulação, em geral, são resolvidos por meio de ajustes dos parâmetros, necessitando que o sistema seja novamente simulado. Gangwal cita em (Gangwal, 2005) três problemas que podem ocorrer nesses sistemas: 1) longo tempo de simulação a cada mudança; 2) Numerosas modificações devido à interdependência do sistema, e 3) o comportamento do pior caso não é necessariamente coberto.

Uma forma alternativa de verificar as propriedades de um sistema completo é por meio de modelos analíticos. A modelagem matemática e a lógica formal são técnicas que auxiliam a especificação e verificação tanto de sistemas computacionais como de

software. Essas técnicas, também conhecidas como métodos formais, usualmente são utilizadas como opções para otimizar os custos e atender as restrições de projeto. Os métodos formais executam um importante papel em muitas atividades incluindo certificação, reuso e segurança (NASA, 1995).

A adoção de técnicas formais em projetos de *Systems-on-Chip*, particularmente no aspecto da comunicação, vem ganhando espaço em diferentes iniciativas de pesquisas. Provavelmente, a característica de poder detectar falhas nas fases iniciais do ciclo de vida do projeto, ao contrário das linguagens de descrição de hardware que só detectam as falhas na fase de simulação ou verificação funcional, tem incentivado o uso de métodos formais em diferentes projetos, conforme serão comentados a seguir.

Schmaltz e Borrione apresentaram em (Schmaltz, 2004a; Schmaltz, 2004b; Schmaltz, 2005) a especificação formal da Rede-em-Chip Octagon (Karim, 2002) na lógica *A Computational Logic for Applicative Common Lisp* – ACL2 (Kauffman, 2000). Em (Schmaltz, 2004a), os autores desenvolveram uma metodologia de especificação e verificação funcional para os primeiros passos de módulos virtuais para comunicação parametrizada. Em (Schmaltz, 2004b) foi definido e validado um algoritmo de roteamento, um algoritmo de escalonamento e as provas que uma mensagem ou é perdida, ou eventualmente alcança o destino esperado sem ser modificada. Em (Schmaltz, 2005) um modelo de Rede-em-Chip denominado GeNoC é apresentado. O modelo identifica funções e restrições que devem ser satisfeitas para provar a corretude da GeNoC. Esses trabalhos apresentam como contribuição um modelo genérico para Redes-em-Chip e uma biblioteca específica de funções e teoremas. A biblioteca pode ser usada em projetos com funcionalidades semelhantes, para provar a corretude das operações de leitura e escrita, provando, por exemplo, se as mensagens realmente alcançam o destino sem serem modificadas.

Tsiopoulos e Waldén (Tsiopoulos, 2006), considerando o papel importante executado pelos roteadores, utilizaram o formalismo *B Action System* (Waldén, 1998) para desenvolver um método de desenvolvimento composicional de esquemas de roteamento assíncrono para projeto de sistemas de Redes-em-Chip.

A Rede-em-Chip Æthereal é uma estrutura de comunicação particular que foi desenvolvida nos laboratórios de pesquisa da Philips (Goossens, 2005c), e tem como princípio que o serviço garantido da entrega de dados ordenados, sem perda e não corrompidos, bem como outros aspectos de QoS, são essenciais para a construção eficiente de SoCs complexos. Gebremichael (Gebremichael et al., 2005) produziu um modelo formal modular em *Prototype Verification System* – PVS (Crow, 1995), no qual um critério de corretude foi estabelecido para conseguir ausência de *deadlock*. Outros métodos usando formalismo são apresentados em (Gangwal, 2005), os quais usam um método analítico para computar latência, vazão e requisitos de armazenamento para a rede Æthereal.

Por meio do formalismo é possível documentar um projeto complexo e, auxiliar os projetistas a esclarecer as escolhas de projetos e resolver inconsistências problemáticas nas fases iniciais do ciclo de vida. Um método analítico, por exemplo, pode verificar um sistema sem a necessidade de simulações.

Apesar do uso de métodos formais no contexto de sistemas embarcados ser sujeito de diversos esforços de pesquisa, a sua efetiva inclusão na metodologia de projeto e principalmente o acréscimo da análise qualitativa baseada em propriedades ainda permanece uma questão pouco explorada.

1.5 Solução Proposta

Mediante a definição do problema exposto na seção 1.3 e as abordagens existentes sobre a busca de soluções para o problema da complexidade do projeto de Redes-em-Chip, a solução proposta nesta tese consiste no desenvolvimento de uma metodologia baseada na descrição formal de um modelo especificado em um alto nível de abstração, que contenha as propriedades da estrutura de comunicação das Redes-em-Chip, de modo a identificar possíveis falhas nas fases iniciais do ciclo do projeto.

A importância das abordagens relacionadas ao projeto de Redes-em-Chip para SoC é clara e de alguma forma contribui para o método proposto nesta tese. Contudo, essas

abordagens (1) são comumente definidas para uma arquitetura específica de Rede-em-Chip (Gangwal 2005; Gebremichael, 2005), que obviamente restringe o reuso em diferentes projetos, ou (2) são exclusivas para um componente (Tsiopoulos, 2006) ou propriedade específica de Rede-em-Chip (Schmaltz, 2005; Jalabert, 2004).

Na metodologia proposta, denominada *CAD*$\mathbb{Z}$ (*Computer Aided Design based on Z*), uma Rede-em-Chip pode ser genericamente definida e sucessivamente refinada em uma Rede-em-Chip mais específica, conservando as propriedades definidas e garantidas nas especificações realizadas anteriormente. Isto se deve ao uso de procedimentos formais para lidar com a descrição funcional do sistema em diferentes níveis de abstração. Essa característica facilita o trabalho do projetista, pois permite que o mesmo mantenha sob controle a complexidade do projeto.

Outra característica fundamental desta metodologia refere-se à reusabilidade, que é obtida a partir do aproveitamento de elementos especificados. Tais elementos referem-se às propriedades estáticas e dinâmicas de componentes presentes em circuitos que apresentem funcionalidade semelhante. Dessa forma, é possível reduzir o tempo de projeto, uma vez que somente propriedades de novos componentes ou as propriedades derivadas da associação entre componentes precisam ser validadas. Em termos práticos, tais elementos podem compor uma biblioteca em um ambiente de projeto, a exemplo dos ambientes CAD. É importante ressaltar que a reusabilidade se dá em diferentes níveis de granularidade, ou seja, de elementos simples e de elementos compostos. Por exemplo, uma especificação completa de Rede-em-Chip pode ser totalmente reutilizada em diferentes projetos de SoC ou MPSoC.

Embora o desenvolvimento do ambiente de projeto não faça parte do escopo desta tese, a metodologia proposta constitui relevante contribuição para sua futura implementação. Uma clara contribuição refere-se à interação do projetista com o ambiente, explicitada pela definição dos passos a serem seguidos, o que facilita a identificação de atores, de requisitos de interface e de funcionalidades que o ambiente deve fornecer. Outra contribuição refere-se ao aproveitamento de elementos formais presentes nesta tese (posteriormente detalhados), que serão armazenados em biblioteca disponibilizada pelo ambiente.

A linguagem de especificação formal utilizada na metodologia proposta, notação Z (Spivey, 1992), é baseada em conceitos matemáticos fundamentados na teoria dos conjuntos e lógica de primeira ordem. Algumas características básicas da notação Z guiaram esta escolha. O primeiro é o nível de maturidade da notação, recentemente definido como padrão ISO (ISO, 2002). O segundo refere-se à disponibilidade de ferramentas para suporte a atividades formais, tais como verificadores de tipos, provadores de teorema e animadores de especificação formal (Hazel, 1997; Saaltink, 1997a, 1997b; Utting, 2005). Adicionalmente, a expressividade da notação Z permite iniciar uma especificação com um alto nível de abstração e, por meio de refinamentos sucessivos, é possível atingir um nível mais concreto sem perda da precisão e integridade das propriedades definidas.

O uso de métodos formais na descrição e verificação de propriedades e comportamento de sistemas de computador não é novidade (Cohen, 1989; Bowen, 1996; van Harmelen, 1995). Apesar de algumas críticas sobre a viabilidade de seu uso em projetos industriais, principalmente em relação ao tempo de treinamento e habilidades necessárias, a confiabilidade alcançada pelas técnicas formais em muitos casos justifica sua adoção (Cohen, 1989; NASA, 1995).

A metodologia *CAD*$\mathbb{Z}$ está dividida em três fases: Modelagem, Integração e Análise Qualitativa. A primeira trata da modelagem, por meio da especificação formal das propriedades estáticas e dinâmicas dos elementos da Rede-em-Chip. A segunda fase trata da integração das especificações obtidas na fase de modelagem. Esta fase consiste efetivamente na criação da Rede-em-Chip. A terceira fase consiste da análise qualitativa que tem como objetivo garantir o atendimento aos requisitos do projeto.

Considerando que a metodologia proposta deve ser utilizada nos estágios iniciais do projeto, é importante observar que é possível sua incorporação a um fluxo de projeto tradicional, onde a análise qualitativa pode ser seguida pela geração de código e análise quantitativa. Para tal, será necessário incorporar ao ambiente de projeto, um mecanismo para converter a especificação formal em código na linguagem desejada. Existem várias propostas neste sentido, dentre elas (Ramkarthik, 2006; Utting, 2003). Neste caso, é correto supor que o código obtido terá maior precisão em relação ao atendimento de requisitos do projeto e possivelmente será menos susceptível a falhas.

1.6 Estrutura do Documento

Esta tese está estruturada em seis capítulos, incluindo este que delimita o escopo da pesquisa, e apresenta as razões que motivaram o desenvolvimento da mesma. Os demais capítulos deste documento tratam dos temas e recursos necessários para a contrução e apresentação da metodologia $CAD\mathbb{Z}$.

O capítulo 2 aborda sobre os aspectos teóricos das áreas de Sistemas-em-Chip e de Redes-em-Chip. Na seção de Sistemas-em-Chip são abordadas as características desse tipo de sistema, bem como, os desafios que os projetitas enfrentam ao desenvolvê-lo. A seção de Redes-em-Chip apresenta os principais requisitos de projeto do desenvolvimento de Redes-em-Chip, além de apresentar os elementos que caracterizam e descrevem uma Rede-em-Chip, e que constituem a base da metodologia $CAD\mathbb{Z}$.

Considerando que a metodologia $CAD\mathbb{Z}$ é baseada na linguagem de especificação formal denominada Z, o terceiro capítulo é dedicado a uma visão geral da metodologia $CAD\mathbb{Z}$, uma breve noção do que seja a notação Z e apresenta as especificações dos elementos envolvidos no desenvolvimento de Redes-em-Chip.

O ambiente da metodologia $CAD\mathbb{Z}$ é apresentado no capítulo 4. Neste capítulo são identificadas as equipes envolvidas na metodologia, bem como os papéis de cada equipe no desenvolvimento do projeto de Redes-em-Chip, cuja ênfase é dada à análise qualitativa nas fases iniciais do projeto.

A visão geral da metodologia $CAD\mathbb{Z}$ introduzida no capítulo 3, as especificações, objeto desse mesmo capítulo, bem como a dinâmica da aplicação da metodologia $CAD\mathbb{Z}$ apresentada no capítulo 4 são validadas por meio de estudos de caso apresentados no capítulo 5.

O último capítulo refere-se à conclusão do trabalho e apresenta as contribuições advindas do uso da metodologia $CAD\mathbb{Z}$ que retém o conhecimento e favorece uma análise qualitativa nas fases iniciais de projeto. Além das contribuições, também são apontados trabalhos futuros que aprimorarão a metodologia proposta.

2. Sistema-em-Chip e Rede-em-Chip

A tecnologia computacional está cada vez mais presente no cotidiano das pessoas. Dispositivos desenvolvidos especificamente para atender uma demanda específica do usuário estão se tornando uma prática comum. A funcionalidade dos dispositivos de informação móvel, do inglês *Mobile Information Device* – MID tais como telefones celulares e assistentes pessoais digitais, do inglês *Personal Digital Assistants* – PDA, é ampliada à medida que se torna possível inserir mais componentes em um único circuito integrado.

Além dos MIDs, a grande popularização do computadores pessoais, da Internet e das redes sem fio tem forçado o aumento do poder de processamento e uma integração cada vez maior dos componentes (Bergamaschi, 2002). Portanto, a partir da disponibilidade do silício e visando oferecer um maior poder de processamento com menor consumo de energia, a partir de dispositivos menores, surgiu a tecnologia de sistemas integrados em um único chip, também conhecida por *System-on-Chip* – SoC.

Com o surgimento da tecnologia SoC, a exigência por elementos de comunicação para efetivar a interconexão de forma escalável e paralela entre os diversos componentes de um SoC se tornou imperativa. Mediante as limitações da estrutura tradicional dos barramentos, as Redes-em-Chip se apresentam como uma solução viável para interconexão de componentes heterogêneos como aqueles que integram os SoCs.

Assim como os SoCs apresentam características distintas daquelas conhecidas na arquitetura de computadores tradicionais, as Redes-em-Chip também se deparam com peculiaridades inexistentes na rede de comunicação tradicional. As seções seguintes tratam dos vários aspectos envolvidos no projeto de SoCs e Redes-em-Chip.

2.1 Sistema-em-Chip

A alta disponibilidade de silício e o curto tempo de projeto têm transformado a maneira como os sistemas são projetados. Atualmente, é possível integrar mais de um giga (10^9) transistores em um circuito integrado, sendo possível implementar em um único chip, muitas ou todas, as funções de um sistema eletrônico completo. Um SoC pode ser composto por múltiplos processadores, e neste caso recebe a denominação de MPSoC, do inglês *Multi-Processor System-on-Chip*.

Cientistas prevêem, que dentro de uma década seja possível integrar um tera (10^{12}) componentes em um chip, sejam transistores ou uma nova tecnologia (Vassiliadis et al., 2006). Desta forma, o que se prevê é uma computação ubíqua, onde a computação estará dentro do ambiente e não em computadores separados. As pessoas que comungam com essa idéia, esperam que a computação embarcada no ambiente e os objetos do dia-a-dia permitam que as pessoas interajam com os dispositivos de processamento de informação mais naturalmente e casualmente do que o fazem atualmente (Wikipédia, 2006).

Dentro desse contexto percebe-se que cada vez mais serão projetados sistemas para atender uma demanda específica e não de propósito geral. Uma arquitetura de SoC ou MPSoC geralmente é projetada para aplicação específica, e envolve pelo menos duas importantes características que tornam o projeto de SoC ou MPSoC distinto dos projetos de arquiteturas de computadores de propósito geral. Essas características são as operações em tempo real e o baixo consumo de energia.

A tecnologia que permite o projeto de MPSoC apresenta outras características importantes referentes ao desempenho e ao custo, as quais influenciam aplicações que produzem grandes volumes de produtos, tais como: comunicação, redes de computadores e multimídia. Essas aplicações utilizam padrões que exigem muita computação e algoritmos complexos que não podem ser executados por um hardware simples (Wolf, 2004).

Os SoCs e MPSoCs podem ser encontrados em diferentes categorias de produtos, tais como *set-top-boxes*[1], câmeras digitais, aparelhos de DVD, impressoras; robôs industriais; aviação avançada como o sistema de direção de mísseis e o sistema de controle de vôo; equipamentos médicos e projetos automotivos, como injeção eletrônica e sistema de freio ABS.

Usualmente, os SoCs são projetados para aplicações nas quais um grande volume de dados chega a intervalos regulares e conhecidos para processamento, essas aplicações são citadas na literatura, em inglês, como *streaming applications*. Entretanto, aplicações modernas que são executadas, por exemplo, em ambientes inteligentes (Boekhorst, 2002) e em dispositivos de informação móvel (MID) demandam processamento complexo distinto daquele utilizado por *streaming applications*.

Um exemplo de aplicação complexa é a compressão de vídeo MPEG (Moving Picture Experts Group), que codifica um vídeo como uma seqüência de quadros. Para reduzir o volume de dados, a codificação MPEG utiliza algoritmos sofisticados para redução de redundância espacial, redução de redundância temporal e predição de movimento que é realizada com quadros no passado e no futuro em relação ao quadro que está sendo codificado. A complexidade desse tipo de aplicação, pode ser tratada pela tecnologia MPSoC que reúne processadores embarcados e hardware digital de aplicação específica. A codificação de vídeo de alta definição em tempo real exige taxas extremamente altas de computação. Certamente, o uso de computadores de propósito geral, para esse fim, não forneceria o desempenho necessário, ou o custo desse tipo de arquitetura não seria compatível com a aplicação.

Um sistema computacional completo compreende hardware e software, no caso dos MPSoCs os projetistas de hardware e de software interagem mais do que se estivessem projetando um sistema computacional tradicional. Em um MPSoC um problema pode ser resolvido ou por hardware ou por software, a decisão dependerá dos requisitos de desempenho, consumo de energia e tempo de projeto (Jerraya, 2004a).

[1] O termo **set-top box** (STB) descreve um dispositivo que conecta a televisão a alguma fonte externa de sinal, e transforma o sinal em conteúdo exibido na tela (Wikipedia, 2006)

Em (Wagner *et al.*, 2004), os autores constatam que o projeto de sistemas embarcados complexos como MPSoC está sendo largamente dominado pelo software, e acrescentam que perspectivas de mercado indicam que até 90% do esforço de projeto de sistemas embarcados está sendo na parte de software. Jerraya e Wolf em (Jerraya, 2004a) afirmam que o software é um dos maiores desafios para os projetistas de MPSoC, uma vez que o software que executa no processador deve ser de alto desempenho, de tempo real e de baixo consumo de energia.

Tanto os projetistas de hardware como os de software enfrentam desafios distintos daqueles encontrados nos projetos onde os sistemas de hardware e software são desenvolvidos independentemente. O software a ser integrado, por exemplo, deve ser extremamente confiável e deve atender restrições de projeto típicas de hardware, tais como baixo consumo de energia e temporização confiável. O fato de um problema poder ser resolvido por hardware, por software ou por ambos torna o projeto de um MPSoC muito interessante e desafiante.

Até certo ponto, o conhecimento adquirido pelos projetistas no desenvolvimento de sistemas grandes e poderosos da computação tradicional pode auxiliá-los no projeto de MPSoCs. O projeto de MPSoC contém requisitos especiais que o projetista do mundo das estações de trabalho e dos computadores pessoais, possivelmente, nunca se deparou. Existem aspectos que estão centrados na tecnologia, e outros no usuário da tecnologia. Por exemplo, enquanto o usuário da computação de propósito geral suporta reinicio da máquina, o usuário de um telefone celular não admite que durante uma ligação o aparelho seja reiniciado. A tecnologia sem fio utilizada em dispositivos móveis impõe limites sobre o que pode ser realizado com relação ao tamanho da tela, memória e poder de processamento, diferente da tecnologia utilizada em sistemas grandes e poderosos de computação tradicional.

Além das restrições da tecnologia, é preciso atender às expectativas dos usuários de computadores e consumidores de produtos eletrônicos. Por exemplo, enquanto os usuários de computador pessoal e de estações de trabalho estão acostumados a trabalhar com telas grandes, grande quantidade de memória e tecnologia relativamente baixa confiabilidade, os consumidores de produtos eletrônicos querem dispositivos simples,

intuitivos e confiáveis. Para esses últimos, é inaceitável reiniciar um telefone celular em meio a uma conversa.

Visando atender às restrições de projeto, de tecnologia e dos usuários da tecnologia os MPSoCs são compostos por multiprocessadores heterogêneos, com funções específicas que proporcionam menor consumo de energia. Contudo, tornam a programação mais difícil. Apesar dos MPSoCs poderem ser constituídos por processadores simétricos, o mais provável é serem compostos de multiprocessadores heterogêneos.

Um celular moderno, por exemplo, pode ter de 4 a 8 processadores incluindo um ou mais processadores do tipo *Reduced Instruction Set Computer* – RISC para interface de usuário, processamento de protocolos e outras funções de controle; um processador de sinais digitais (*Digital Signal Processor* – DSP) para codificação e decodificação de voz e interface de radio; um processador de áudio para reprodução de músicas; um processador de imagem para a funcionalidade da câmera; e um processador de vídeo-fone com a capacidade de transmitir e receber imagem visual e voz. Dispositivos como este demandam por requisitos de baixo custo, desempenho razoável e consumo mínimo de energia, que podem ser obtidos pelo uso de processadores de aplicação específica (Martin, 2006).

O mercado atualmente dispõe de muitos MPSoCs para atender a vários tipos de aplicações com diferentes exigências. Por exemplo, a ST Nomadik (ST, 2006) e a TI Omap (TI, 2006) disponibilizam MPSoCs para atender às exigências de multimídia móvel, que necessita de alto desempenho e baixo consumo de energia. A Philips oferece o MPSoC Nexperia (Nexperia, 2006) que disponibiliza altíssimo desempenho para televisão de alta definição *(High-Definition Television* – HDTV), e não tem restrições de consumo de energia como os dispositivos de multimídia móvel. A Intel (Intel, 2006) e a Cisco (Cisco, 2006) oferecem arquiteturas heterogêneas de processadores de rede para processarem pacotes a altas taxas de comunicação. Enfim, para atender os diferentes requisitos de diversas aplicações, existem MPSoC diferenciados cujo projeto têm muitos desafios a serem superados.

Dentre os muitos desafios existentes para projetar um MPSoC, o aumento do desempenho, o consumo de energia, a integridade de sinal, a tecnologia de integração, o custo e a comunicação podem ser citados como desafios relevantes a serem suplantados.

O custo de produção de um SoC é influenciado pelo tamanho do chip, logo é interessante minimizar o tamanho do chip. Para atingir este objetivo é extremamente importante utilizar soluções dedicadas, desenvolvidas para necessidades específicas, pois desta forma é possível atender requisitos importantes de projeto, tais como: minimização de área, redução de consumo de energia e aumento de desempenho (Keutzer, 2000).

Apesar das soluções dedicadas apresentarem as vantagens citadas acima, essas soluções consistem da integração em um único chip de elementos de computação, cujas tecnologias usualmente possuem características distintas, ampliando, desta forma, os desafios do projeto de MPSoCs. Por exemplo, os chips de memória de acesso aleatório dinâmico (em inglês *Dynamic Random Access Memory* – DRAM) precisam de renovação (em inglês, *refresh*) para manter carregado seus padrões de bits. No caso de uma DRAM embarcada, geralmente, é necessária mais renovação que uma memória DRAM não embarcada, implicando em um maior consumo de energia (Crowder et al., 1998).

A minimização do consumo de energia é uma preocupação constante dos projetistas, pois à medida que a complexidade dos microprocessadores e dos programas aumenta, a energia se torna uma restrição de projeto muito importante. Uma das preocupações primordiais do projeto de dispositivos móveis é em relação à duração da carga da bateria, por outro lado esses dispositivos necessitam de memória, as quais são as maiores consumidoras de energia tanto estática como dinâmica. Esses extremos exigem metodologias de projeto eficientes que levem em consideração a aplicação e a previsibilidade de algumas ocorrências.

Uma outra fonte de preocupação no projeto de MPSoCs é a estrutura de comunicação, que adiciona outros desafios e implicam em uma maior complexidade de projeto. Por exemplo, a comunicação ponto-a-ponto por ser projetada sob medida é mais rápida, porém é menos reusável e pode gerar problema quando muitos componentes precisam

se comunicar. O barramento pode ser reusado, mas só permite uma transação por ciclo, logo, torna impossível a paralelização. Os barramentos hierárquicos como o Core Connect (IBM, 2006) e o AMBA (ARM, 2006) só permitem paralelismo restrito. Uma solução emergente são as Redes-em-Chip, porém sua eficiência depende de fatores, como: quantidade de conexões físicas para efetuar o canal, política de prioridade de mensagens, topologia, estratégia de chaveamento, entre outros fatores que precisam ser tratados com muita cautela e rigor.

Para superar os vários desafios existentes no desenvolvimento de Sistemas-em-Chip, ressaltando aqueles citados acima, foram desenvolvidas várias metodologias de projeto (Keutzer, 2000; Savage, 2000; Vincentelli, 2001; Pogge, 2002). Um aspecto importante e adotado nas metodologias para aumentar a produtividade dos SoCs é a utilização de componentes pré-projetados e pré-verificados, ou seja, o reuso de componentes. A técnica de reuso (Keating, 2002) favorece a redução do custo do projeto e seu tempo de desenvolvimento.

O paradigma de projeto baseado em plataforma (Keutzer, 2000), por exemplo, é viabilizado pela técnica de reuso que permite a construção, em poucos meses, de um sistema dedicado com milhões de transistores, cuja arquitetura de hardware e software é específica para um domínio de aplicação. Apesar do projeto baseado em plataforma ser desenvolvido para um domínio específico, o mesmo inclui a característica de parametrização permitindo a definição do número de componentes, da estrutura de comunicação, do tamanho da memória e dos dispositivos de entrada e saída, entre outras definições.

Considerando o paradigma de projeto baseado em plataforma, o projeto de um SoC embarcado pode consistir em encontrar uma plataforma que atenda aos requisitos da aplicação, como desempenho e consumo de energia (Carro, 2003). Segundo Rowson e Vincentelli (Rowson, 1997), uma das maiores regras de projeto de SoC é tornar claro a separação entre computação e comunicação, pois desta forma é possível construir um projeto modular e independente.

2.2 Redes-em-Chip

Assumindo que o projeto de um SoC envolve centenas de blocos IPs que são pré-projetados e pré-verificados, e que funcionam de acordo com suas especificações, o problema então passar a ser a comunicação.

Nos últimos anos, as Redes-em-Chip têm surgido como solução de comunicação confiável e uma alternativa para interconexão *on-chip*, tradicionalmente realizada como uma tarefa *ad-hoc*. O roteamento *ad-hoc* de conexões em circuitos integrados de aplicação específica, como menciona Ron Ho em (Ho, 2001), está cada vez mais enfrentando desafios como atrasos e integridade de sinal na tecnologia DSM. Como solução de interconexão dos componentes em SoC e MPSoC, a Rede-em-Chip busca resolver as restrições de reusabilidade, escalabilidade, largura de banda e carga capacitiva dos modelos tradicionais de interconexão tais como os canais multi-ponto compartilhados ou os canais ponto-a-ponto dedicados.

Shashi Kumar *et al*. (Kumar, 2002) definem Redes-em-Chip como uma arquitetura que é essencialmente uma infra-estrutura de comunicação *on-chip* composta pelas camadas física, de enlace e de rede do Modelo de Referência (*Reference Model* – RM) para Interconexão de Sistemas Abertos (*Open Systems Interconnection* – OSI) (Day, 1983). Kees Goossens *et al*. (Goossens, 2002) afirmam que uma Rede-em-Chip é uma arquitetura de hardware juntamente com um modelo de programação.

Luca Benini (Benini, 2002) compara as Redes-em-Chip às redes de interconexão para computadores paralelos de alta performance com múltiplos processadores, no qual cada processador é um chip individual. Na realidade, as Redes-em-Chip adotam características e metodologias de projeto usadas na computação paralela e na área de redes de sistemas de computação. Portanto, muitos dos princípios fundamentais adquiridos nas décadas de 80 e 90 são usados como base para as pesquisas atuais.

Assim como uma rede de interconexão para computadores paralelos, duas características importantes de Redes-em-Chip são a topologia e o protocolo de comunicação. Enquanto a topologia está inerentemente relacionada à arquitetura da rede, ou seja, ao *layout* do sistema de comunicação no chip; o protocolo especifica

como os recursos da rede serão usados durante a operação do sistema, ou seja, define por meio de mecanismos de comunicação a forma como as mensagens serão transferidas pela rede. Segundo Sgroi (Sgroi *et al*., 2001), uma comunicação confiável entre vários componentes requer protocolos bem definidos, constituídos por regras que ditam como as interações entre os componentes devam acontecer, de modo a alcançar as exigências de desempenho e comunicação do sistema, enquanto os recursos físicos (área e potência) são minimizados.

As seções 2.2.1 e 2.2.2 apresentam, com base em (Duato, 1997; Hwang, 1993, Benini, 2004), aspectos relacionados à topologia e aos mecanismos de comunicação de Redes-em-Chip, respectivamente. O objetivo dessas seções é revisar os conceitos de redes tradicionais que se adéquam às Redes-em-Chip, uma vez que o foco desta tese está na metodologia para projeto de redes em chip.

Considerando o conceito de Redes-em-Chip apresentado por Kees Goossens *et al.* em (Goossens, 2002), que diz "uma Rede-em-Chip é uma arquitetura de hardware juntamente com um modelo de programação", a seção 2.2.3 aborda sobre os modelos de programação que permitem os programadores explorarem o poder computacional da arquitetura heterogênea e paralela dos MPSoCs.

2.2.1 Tipos de Redes de Interconexão

Considerando o aspecto da topologia, basicamente, as Redes-em-Chip podem ser classificadas em redes diretas e redes indiretas. Entretanto, segundo Duato (Duato, 1997) a topologia de redes de interconexão pode ser dividida em quatro classes principais: rede compartilhada (barramento), rede direta, rede indireta e rede híbrida. Essa taxonomia utilizada na computação paralela tradicional, foi mostrada em (Benini, 2004) como podendo ser aplicada ao domínio MPSoC, que possui diferentes tipos de interconexões que apresentam características que se adéquam a essa taxonomia.

A rede compartilhada é a topologia de interconexão mais simples, cujo meio de transmissão é compartilhado por todos os dispositivos de comunicação. Nestas redes, somente um dispositivo pode utilizar a rede a cada momento, logo é necessário um

mecanismo de árbitragem para decidir “quem” se apropriará da rede em um dado instante. Uma vantagem desse tipo de interconexão é o suporte dado à difusão da informação em que todos os dispositivos podem monitorar as atividades da rede e receber a informação transmitida sobre o meio compartilhado.

Devido à largura de banda limitada desse tipo de rede, um meio compartilhado só suporta um número limitado de dispositivos, ou seja, do ponto de vista da escalabilidade essa estrutura de rede é limitada. No domínio MPSoC muitas arquiteturas adotam esse tipo de topologia. A ARM (ARM, 2006) desenvolveu o barramento AMBA, que implementa uma hierarquia de barramentos, para atender às necessidades existentes na estrutura de interconexão dos projetos MPSoC.

A rede direta, também conhecida por rede ponto-a-ponto ou rede baseada em roteadores, é uma rede que supera o problema de escalabilidade das redes compartilhadas. Os componentes de uma rede direta possuem um bloco de interface de rede, denominado roteador, acoplado ao próprio componente constituindo um nó. Os nós são conectados diretamente um ao outro pela rede. Portanto, quando o número de nós de um sistema aumenta, a largura de banda de comunicação também aumenta.

A arquitetura *Reconfigurable Architecture Workstation* – RAW (Agarwal, 1999), que consiste de um sistema paralelo descentralizado com armazenamento local, foi desenvolvido pelo *Massachusetts Institute of Technology* – MIT. Esta arquitetura é um SoC totalmente programável, que utiliza roteadores acoplados aos blocos computacionais para realizar a comunicação entre os mesmos. A arquitetura RAW constitui um exemplo de SoC com uma estrutura de comunicação baseada na rede direta.

A rede indireta conecta os componentes por meio de uma ou mais chaves. A arquitetura de rede indireta apresenta-se como uma solução para projetos de interconexão escalável. A conexão entre os nós é realizada por meio de um conjunto de chaveadores, cuja configuração do caminho de comunicação pode ser alterado no decorrer do tempo.

As definições de redes direta e indireta são muito próximas, entretanto, em geral, pode-se dizer que as redes diretas são compostas por roteadores ligados diretamente a um

componente, enquanto que as redes indiretas dispõem de um conjunto de chaveadores que conduzirão a informação. Como exemplo de rede indireta, no domínio SoC, pode-se citar o FPGA - *Field Programmable Gate Array* Spartan-II da Xilinx (Xilinx, 2006), que é composto por uma matriz de blocos lógicos configuráveis (*Configurable Logic Blocks* – CLBs) que são conectados via uma hierarquia de canais de roteamento. Logo, cada chip tem uma rede indireta sobre uma estrutura homogênea. Outro exemplo de rede indireta é o Virtex-II da Xilinx (Xilinx, 2006) que são FPGAs com vários elementos configuráveis que dão suporte ao projeto de processadores de sinais digitais reconfiguráveis. A interconexão programável do Virtex-II é realizada mediante chaves de roteamento.

As redes híbridas, em geral, combinam os mecanismos das redes compartilhadas, direta e indireta. Essa combinação, em relação à rede compartilhada, resulta no aumento de largura de banda, e com relação às redes direta e indireta reduz a distância entre os nós. No domínio MPSoC, o padrão AMBA 2.0 especifica três protocolos: o barramento de alta performance (AMBA *High performance Bus* – AHB), o barramento de sistema avançado (*Advanced System Bus* – ASB) e barramento de periférico avançado (*Advanced Peripheral Bus* – APB), cujo objetivo é atender os diversos componentes de uma arquitetura que demandam diferentes exigências de comunicação e largura de banda.

2.2.1.1 Topologias

Existem diferentes formas de organizar a interconexão entre os vários nós de uma rede. A topologia de rede descreve o fluxo de dados através da rede (topologia lógica) e a estrutura física de como os dispositivos estão conectados (topologia física). Em geral, a topologia consiste na organização da rede sob a forma de um grafo, no qual os roteadores são os vértices do grafo e os canais são os arcos.

Segundo Kai Hwang (Hwang,1993), o desempenho de uma rede de interconexão é afetada pelos seguintes fatores: funcionalidade, latência da rede, largura de banda, complexidade do hardware e escalabilidade.

A funcionalidade está relacionada ao modo como a rede suporta o roteamento de dados, tratamento de interrupção, sincronização e coerência. A latência da rede se refere ao pior caso na transferência de uma mensagem através da rede. A taxa máxima de transferência de dados está relacionada à largura de banda da rede, enquanto que os custos de implementação com conexões, roteadores, árbitros e interface lógica estão relacionados à complexidade do hardware. A escalabilidade está relacionada à habilidade da expansão da rede com uma performance escalável.

Tendo em vista as peculiaridades dos SoCs, várias topologias de rede de interconexão foram propostas para diferentes projetos MPSoC, as quais estão ilustradas na Figura 2.1. As topologias de redes diretas mais utilizadas são a malha (*mesh*), o toróide (*torus*) e o n-cubo. Enquanto que as topologias que mais se destacam nas redes indiretas são o *crossbar* e as redes multiestágio. Existem outras topologias alternativas tanto para a rede direta como indireta (Jantsch, 2003). A topologia da árvore gorda (*fat-tree*), por exemplo, foi utilizada na rede indireta denominada SPIN, desenvolvida por Guerrier e Greiner (Guerrier, 2000).

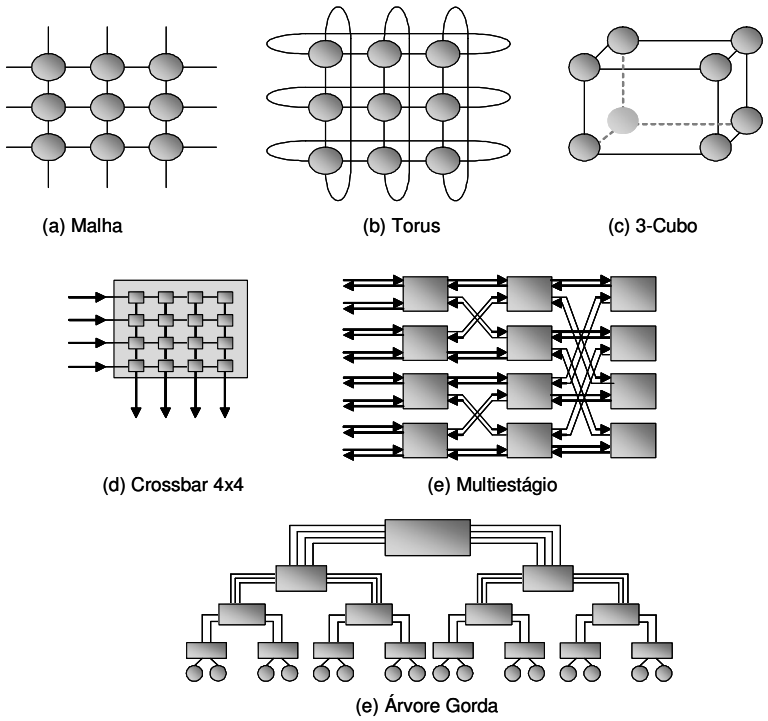

Figura 2.1 Topologias de redes de interconexão (Hwang, 1993; Duato, 1997)

Dally e Towles em (Dally, 2001) apresentaram um estudo sobre a utilização das topologias *torus* e *mesh*. Mediante a grande disponibilidade de conexões existente nos MPSoCs, algumas vezes é recomendado o uso da topologia *torus* que oferece o dobro da demanda de conexões e da largura de bisseção[2] da topologia *mesh*. Entretanto, se a dissipação de potência for crítica, o recomendado é a utilização da *mesh* em substituição a *torus*.

O projeto de MPSoCs, usualmente, tem restrições severas de desempenho que necessita da escolha adequada de uma topologia que atenda da melhor forma possível os fatores de funcionalidade, latência da rede, largura de banda, complexidade do hardware e escalabilidade, citados anteriormente. Apesar das Redes-em-Chip adotarem as mesmas técnicas utilizadas em rede de computadores tradicionais, as mesmas dispõem de características arquiteturais distintas, como a abundância de conexões, que faz com que os sinais de controle não sejam serializados ou transmitidos com os dados como é feito nas redes de computadores paralelos, mas através de sinais de controle dedicados. Outra característica é a restrição do consumo de energia que limita o uso de área utilizada para o armazenamento de dados (*buffers*). Segundo Lei e Kumar (Lei, 2003) a maior desvantagem do paradigma de Redes-em-Chip é o possível aumento no atraso da comunicação.

2.2.2 Mecanismos Básicos de Rede

As redes de interconexão dependem de vários fatores para atingir o desempenho desejado. Em geral, fatores como escalabilidade, particionabilidade, distância, restrições físicas e de custo, entre outros, afetam os serviços básicos da rede que consistem em transportar um fluxo de bit de um ponto a outro da mesma.

Os serviços oferecidos por uma rede de interconexão são diferenciados, por exemplo, pela qualidade da velocidade, atraso e erros. Esses serviços são produzidos por mecanismos, tais como chaveamento, roteamento, arbitragem, controle de fluxo, e

[2] Quando uma rede é cortada em duas metades iguais, o número mínimo de canais ao longo do corte é chamado bisseção do canal de largura b. (Hwang, 1993)

memorização. Apesar da existência de tais mecanismos, existe a possibilidade de um fluxo de bits não alcançar o seu destino. Essa ocorrência pode ser ocasionada por três situações conhecidas como *deadlock*, *livelock* e *starvation*.

As próximas sub-seções tratam sobre as situações indesejadas acima citadas, bem como dos mecanismos básicos de rede. Tendo em vista que alguns mecanismos de rede abordam sobre as situações de *deadlock*, *livelock* e *starvation*, optou-se por iniciar as subseções comentando sobre as situações que impedem um pacote de chegar ao seu destino, para em seguida abordar sobre os mecanismos de rede.

2.2.2.1 *Deadlock, Livelock* e *Starvation*

Os pacotes são transportados pela rede através de canais físicos e de *buffers* existentes nos roteadores ou fora deles. Entretanto, os *buffers* têm capacidade finita e algumas vezes não podem atender a demanda de comunicação, ocorrendo, portanto, situações indesejadas que impedem a recepção do pacote pelo destinatário. Estas situações serão abordadas a seguir.

Deadlock

O deadlock consiste em uma situação que pode adiar a entrega do pacote no destino indefinidamente, uma vez que um conjunto de pacotes pode ficar bloqueado para sempre na rede. A dependência cíclica de recursos na rede ocasiona essa situação. A Figura 2.2 ilustra um exemplo de *deadlock* envolvendo quatro roteadores e quatro pacotes.

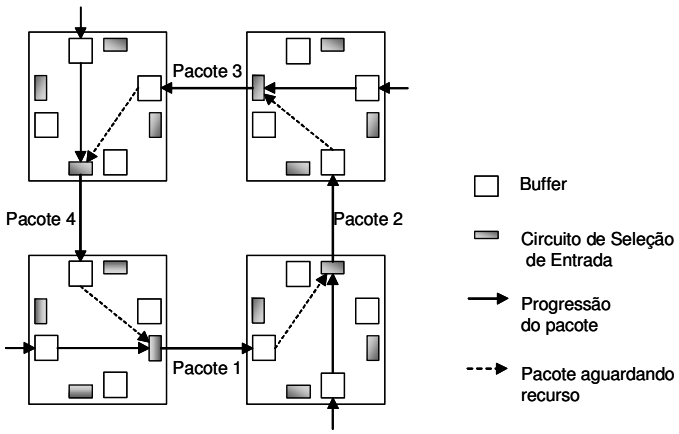

Figura 2.2 Deadlock envolvendo quatro pacotes (Ni, 1993)

A dependência cíclica ocorre devido a existência de quatro pacotes no segmento de rede ilustrada na Figura 2.2. Cada um dos pacotes mantém um canal de enlace, mas requer outro canal já alocado para um outro pacote. Por exemplo, o *packet 1* para avançar precisa do canal que está sendo ocupado pelo packet 2. A Figura 2.3 exibe mais claramente a dependência cíclica existente na Figura 2.2.

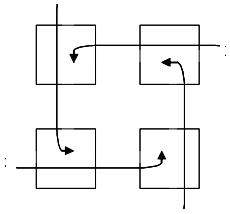

Figura 2.3 Dependência Cíclica

Do ponto de vista da comunicação, para solucionar o problema do *deadlock* foram desenvolvidas as seguintes estratégias: Prevenir, Evitar, Detectar e Recuperar. A estratégia de prevenção garante os recursos a um pacote de modo que o *deadlock* nunca ocorra. Nessa estratégia, o escalonador aloca todos os recursos necessários antes do início da transmissão do pacote. Na estratégia de evitar, o escalonador controla a alocação dos recursos, ou seja, os recursos são requisitados à medida que um pacote avança pela rede. Uma forma de evitar o *deadlock* é limitando a quantidade de voltas que o algoritmo de roteamento pode realizar, pois desta forma impede-se a ocorrência de ciclos. Nas estratégias de detecção e recuperação, o escalonador concede os recursos

ao processo tão logo eles se tornem disponíveis, sem nenhuma verificação. Quando o *deadlock* é detectado, o escalonador interrompe alguns recursos para recuperar o sistema da situação de *deadlock*.

Das estratégias acima, a mais conservadora é a estratégia prevenir, que conduz a uma baixa utilização dos recursos da rede. A estratégia de evitar *deadlock* é menos conservadora, uma vez que os recursos só são alocados à medida que o pacote avança pela rede. A detecção é uma estratégia otimista, que só deve ser utilizada em sistemas que a ocorrência de *deadlock* seja rara.

Adotando essas estratégias, vários trabalhos foram publicados visando à solução do problema de *deadlock* com diferentes tipos de chaveamento (Dally, 1987, 1993; Duato, 1993; Chiu, 2000; Wu, 2002). No domínio das Redes-em-Chip também existem muitos esforços para prevenir, evitar, detectar e recuperar o sistema de problemas de deadlock (Li, 2006; Palesi, 2006; Gebremichael, 2005)

Algumas soluções apresentadas para o problema de *deadlock* contemplam canais virtuais (Duato, 1993). Um canal virtual é um canal lógico entre dois nós, que é formado por um *buffer* no nó de origem, um canal físico entre os nós origem/destino e um *buffer* no nó de destino (ver Figura 2.4). Os canais virtuais podem resolver o problema de deadlock enquanto conseguirem alto desempenho. Entretanto, requer grande espaço de armazenamento elevando o consumo de energia, o que dificulta a implementação em Redes-em-Chip, uma vez que esta estrutura de comunicação tem restrições quanto ao consumo de energia.

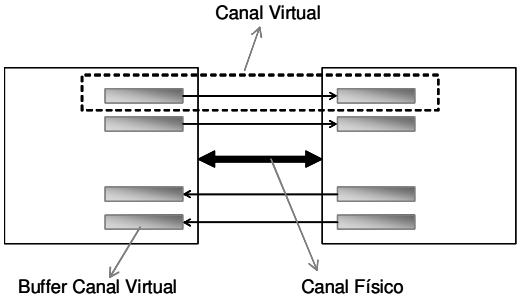

Figura 2.4 Canal Virtual (Duato, 1997)

Outra solução adotada para obtenção de roteamento livre de *deadlock,* foi proposta em (Dally, 1987), e consiste de um grafo de dependência de canal seguido de uma ordem entre os canais. O roteamento é restrito aos canais visitados em ordem crescente ou decrescente para eliminar os ciclos do grafo de dependência de canal. Além das soluções citadas, outros algoritmos foram propostos no sentido de proibir no mínimo uma volta em cada ciclo de roteamento possível. Em (Glass, 1998) foram propostos os algoritmos *west-first, north-last* e *negative-first*.

Livelock

A situação *livelock* ocorre quando os canais necessários para um pacote chegar ao destino estão sempre ocupados, e o pacote fica trafegando permanentemente pela rede. Segundo Duato *et al.* (Duato, 1997), uma das formas para previnir essa situação é utilizar apenas caminhos mínimos, ou seja, caminhos mais curtos, entre a origem e o destino.

A tolerância à falha consiste no principal motivo para utilização de caminhos não mínimos. Entretanto, mesmo quando caminhos não mínimos são utilizados, é possível previnir *livelock*, neste caso é só limitar a quantidade de operações de desvio de caminhos não mínimo.

Starvation

Diz-se que ocorre *starvation* quando um pacote em um *buffer* de entrada, ao competir com outros pacotes por uma mesma saída, pode ficar bloqueado permanentemente se o recurso requisitado for sempre concedido a outros pacotes.

Quando ocorre *starvation* uma política de arbitragem deve ser utilizada para decidir qual dos *buffers* de entrada deve ser conectado a saída. Um esquema *round-robin*, muitas vezes, é suficiente para garantir a justiça do uso de recursos. Se a prioridade de alguns pacotes for maior em relação a outros, alguma largura de banda deve ser reservada para os pacotes de baixa prioridade. Esta solução pode ser realizada por meio da limitação de pacotes de alta prioridade, ou por meio da reserva de alguns canais virtuais ou *buffers* para os pacotes de baixa prioridade.

2.2.2.2 Tipos de Chaveamento

O chaveamento é o principal mecanismo da arquitetura de interconexão para rotear a informação da origem para o destino. O chaveamento tem por função transferir uma mensagem da entrada de um roteador para qualquer um de seus canais de saída.

As principais técnicas de chaveamento são denominadas chaveamento de circuitos e chaveamento de pacotes. A primeira delas pressupõe a existência de um caminho dedicado de comunicação entre os nós, enquanto que a segunda transfere os dados por meio de pacotes. Um pacote é uma unidade de informação que contém detalhes sobre o roteamento e seqüenciamento dos dados e mantém uma estrutura semelhante à de uma mensagem, com um cabeçalho (*header*), uma carga útil (*payload*) e um terminador (*tail*). Assim como uma rede de interconexão para computadores paralelos, muitos modelos de Redes-em-Chip adotam a técnica de redes chaveadas de pacote.

O chaveamento de circuito consiste no estabelecimento de um caminho físico completo entre os nós fonte e destino para transferência de uma mensagem. O nó fonte insere na rede um cabeçalho de roteamento com o endereço de destino e algumas informações de controle. O cabeçalho avança pela rede reservando canais para o estabelecimento do circuito. Se o canal desejado já estiver alocado, o cabeçalho fica bloqueado até que esse canal lhe seja alocado. Quando o cabeçalho atinge o nó destino, uma informação de reconhecimento é enviada ao nó fonte através do circuito estabelecido. Em seguida é efetuada a transferência dos dados da mensagem. O circuito pode ser desfeito durante o avanço do terminador da mensagem em direção ao destino. A passagem do terminador sinaliza para cada roteador que o canal alocado pode ser liberado.

A latência do chaveamento de circuito depende da demora em estabelecer a conexão mais a demora em transmitir os dados. Este tipo de chaveamento, apesar de consumir recursos, uma vez que uma conexão reserva todos os recursos entre a origem e o destino, costuma ser utilizado quando o tráfego de mensagens é caracterizado por mensagens longas e não muito freqüentes. A latência e o processamento da comunicação sobre um caminho fixo são geralmente previsíveis, podendo, então,

fornecer uma alta qualidade de serviço que pode ser essencial em controle, videoconferência e outras aplicações de tempo-real (Walrand, 1996)

A arquitetura do roteador da Rede-em-Chip Æthereal conceitualmente tem duas partes, que foram definidas como roteador de vazão garantida (*Guaranteed-Throughput* – GT) e roteador de melhor esforço (*Best-effort* – BE) (Rijpkema, 2003). O roteador que oferece o serviço de processamento garantido utiliza chaveamento de circuito com acesso múltiplo por divisão de tempo (*Time Division Multiple Access* – TDMA), e o roteador que desempenha o serviço de melhor esforço adota o chaveamento de pacote.

A técnica de chaveamento de pacote pode ser implementada por três modos distintos: *Store-and-Forward* – SAF, *Virtual Cut-Through* – VCT e W*ormhole*. O chaveamento *Store-and-Forward* é utilizado em muitas redes de computadores tradicionais e consiste no recebimento de todo o pacote por um roteador, onde o pacote é armazenado; o roteador identifica o destino da mensagem, seleciona uma porta de saída com base em algum critério de roteamento e repassa todo o pacote adiante para um roteador vizinho ou para o nó de roteamento local (ver Figura 2.5). A técnica SAF introduz atraso não determinístico devido ao enfileiramento de mensagens nos nós intermediários (Benini, 2004).

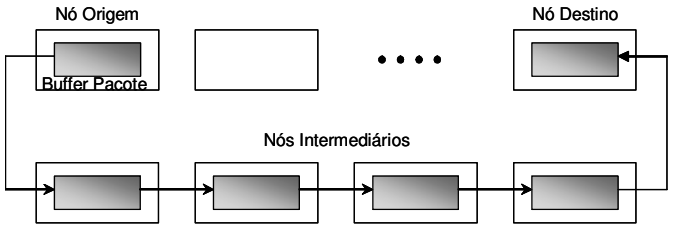

Figura 2.5 Chaveamento Store-and-Forward (Hwang, 1993)

Considerando, que nas Redes-em-Chip, os recursos de armazenamento apresentam custo elevado em termos de área e consumo de energia, a técnica de chaveamento *Store-and-Forward* não se apresenta como a técnica mais adequada para este tipo de arquitetura de comunicação *on-chip*.

O chaveamento de pacote *Virtual Cut-Through* foi proposto como uma alternativa ao chaveamento *Store-and-Forward*. Quando o cabeçalho do pacote contendo as informações de roteamento chega a um roteador e o canal de saída desejado encontra-se disponível, o restante do pacote (carga útil e terminador) desvia o *buffer*, reduzindo a latência da comunicação. Um pacote só é armazenado em *buffer* se o canal desejado estiver indisponível.

O chaveamento de pacote *Wormhole* é uma variação do chaveamento *Virtual Cut-Through*, e foi originalmente desenvolvido para *clusters* de computadores paralelos (Duato, 1997). Esta técnica divide a mensagem em pacotes e os pacotes em *flits* (*flow control unit*). Os *flits* avançam pela rede em modo *pipeline*. Quando o *flits* cabeçalho alcança um nó que não tem canal de saída disponível, todos os *flits* são bloqueados onde estão (ver Figura 2.6).

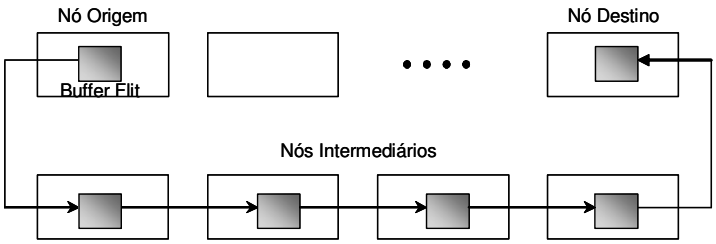

Figura 2.6 Chaveamento *Wormhole* (Hwang, 1993)

A grande vantagem do chaveamento *Wormhole* são os requisitos de armazenamento que são menores que os requisitos adotados nas abordagens SAF e VCT, tornando possível o desenvolvimento de roteadores pequenos e rápidos. Segundo Lionel Ni (Ni, 1993), *Wormhole* é a técnica de chaveamento mais utilizada, uma vez que apresenta mais vantagens em relação à utilização da rede e ao custo dos roteadores.

Wormhole é apontada como a técnica de chaveamento ideal para as Redes-on-Chip, pois a mesma apresenta as vantagens de oferecer uma latência baixa, e necessitar de pouco espaço de armazenamento. Recentemente, vários projetos de Redes-on-Chip foram desenvolvidos utilizando a abordagem *Wormhole* como solução para o chaveamento de pacotes (Dally, 2001; Goossens, 2002; Dall'Osso, 2003; Adriahantenaina, 2003).

Apesar da técnica *Wormhole* apresentar mais vantagens em relação às demais técnicas de chaveamento, e por este motivo ser mais utilizada tanto nas estruturas de redes tradicionais com nas estruturas *on-chip*, essa técnica apresenta maiores preocupações com relação aos pacotes que podem bloquear uns aos outros de maneira circular, de modo que nenhum pacote possa avançar, criando desta forma uma situação de *deadlock*.

2.2.2.3 Roteamento

O roteamento consiste em um mecanismo de comunicação que define o caminho a ser utilizado por uma mensagem para alcançar seu destino. O algoritmo de roteamento escolhido interfere diretamente no desempenho de comunicação da rede de interconexão.

Conforme Lionel Ni, em (Ni, 1993), o roteamento pode ser classificado quanto ao lugar onde as decisões de roteamento são tomadas e quanto à adaptatividade. Quanto ao lugar, o roteamento pode ser feito na fonte, distribuído ou centralizado. No roteamento fonte o nó fonte (emissor) seleciona todo o caminho a ser percorrido pelo pacote antes de enviá-lo. Neste caso, cada pacote é acrescido da informação do roteamento, e não pode ser mais alterado depois de sair da fonte. No roteamento distribuído, cada roteador, ao receber o pacote, decide como será realizado o roteamento enquanto o pacote trafega na rede. Neste caso, o algoritmo de roteamento deve determinar para qual vizinho o pacote será enviado. Essa decisão deve ser executada rapidamente a fim de reduzir a latência na rede. No roteamento centralizado, um controlador central estabelece os caminhos.

Com relação à adaptatividade o roteamento pode ser classificado como determinístico e adaptativo. No roteamento determinístico o caminho entre a fonte e o destino será sempre o mesmo, enquanto que no roteamento adaptativo o caminho dependerá das condições da rede, isto é, o algoritmo utiliza informações sobre o tráfego e os canais para evitar caminhos congestionados ou com falha. Se o algoritmo de roteamento selecionar apenas canais de saída que aproximam cada vez mais o pacote do seu

destino, é chamado de algoritmo mínimo, caso contrário, é chamado não-mínimo, ou seja, o algoritmo seleciona canais que levam o pacote a se afastar do seu destino.

Uma das principais vantagens do uso do roteamento determinístico consiste na simplicidade do projeto de roteadores. Devido à lógica simplificada, o roteamento determinístico fornece uma baixa latência quando a rede não está congestionada. Entretanto, quando a taxa de injeção de pacotes na rede aumenta, a tendência é haver uma degradação de vazão, uma vez que os roteadores não podem resolver dinamicamente o congestionamento da rede.

Por outro lado, o roteamento adaptativo está preparado para evitar canais congestionados, e oferecer melhor vazão, por meio de caminhos de roteamento alternativo. Entretanto, devido à lógica extra, necessária para decidir sobre o melhor caminho de roteamento, o roteamento adaptativo tem uma maior latência comparada ao roteamento determinístico, considerando baixos níveis de congestionamento de rede (Hu, 2005).

2.2.2.4 Arbitragem

A arbitragem é um mecanismo que busca resolver conflitos internos no roteador. Diz-se que ocorreu uma colisão de recurso, quando um pacote não pode prosseguir porque o recurso que ele precisa já está sendo utilizado por outro pacote. A solução para esses casos é uma política, que deve decidir se o pacote deve ser descartado, bloqueado no lugar onde está, recebido e armazenado temporariamente, ou desviado para um outro caminho.

O árbitro de um roteador pode ser implementado de forma centralizada, ou distribuída. Na abordagem centralizada, o árbitro consiste de um único módulo que avalia as requisições para uso dos canais de saída. A arbitragem é executada com base em uma visão global da utilização dos canais de saída e determina qual desses canais será utilizado por cada pacote. Na abordagem distribuída, o árbitro é constituído por múltiplos módulos, sendo que cada um deles é associado a um canal de saída e possui

uma visão local limitada ao canal ao qual está associado. Porém, sua arquitetura é mais simples e essa abordagem permite a construção de roteadores mais rápidos.

2.2.2.5 Controle de Fluxo

Em geral, uma rede dispõe de recursos como: canais e *buffers*. A alocação dos recursos necessários para um pacote avançar pela rede é feita através do mecanismo de controle de fluxo, o qual regula o tráfego nos canais. O objetivo de uma boa política de controle de fluxo é evitar o congestionamento do canal, enquanto reduz a latência da rede (Ni, 1993).

O controle de fluxo normalmente é realizado no nível de enlace. A forma mais comum de efetuar o controle de fluxo é por meio de um *buffer First In, First Out* – FIFO de entrada no nó destino e um sinal de retorno ao nó origem, cujo objetivo é informar se há espaço disponível nesse *buffer*. Essa informação pode sinalizar a disponibilidade de um crédito no nó destino, o nó origem só enviará um dado se existir crédito disponível no destino. Uma outra abordagem do controle de fluxo é o "aperto de mão" (*handshake*), que consiste no envio, por parte do nó origem, de uma informação referente à intenção desse nó enviar uma mensagem para o nó destino. Essa informação é enviada através de um sinal de validação, e a confirmação de disponibilidade de espaço no nó destino é enviada através de um sinal de reconhecimento (*acknowledge*).

A Figura 2.7 ilustra o protocolo *handshake* entre dois roteadores *Wormhole*. Entre os roteadores adjacentes existe um canal de dados unidirecional (*channel*) e um sinal de controle *Request*/*Acknowledge* (R/A). Quando o roteador B está pronto para receber um flit, ou seja, o *buffer* flit está disponível, ele puxa o sinal para *low* (a). Quando o roteador A está pronto ele eleva o sinal para *high* e transmite o flit *i* através do canal (b). Enquanto o flit estiver sendo recebido por B, o sinal R/A é mantido *high* (c). Depois que o flit *i* é removido do *buffer* de B (d), o ciclo se repete para a transmissão do próximo flit (i + 1) até que todo pacote seja recebido.

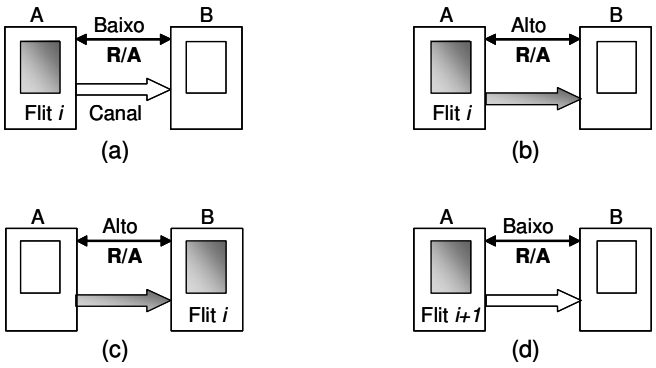

Figura 2.7 Protocolo Handshake (Ni, 1993)

Outra forma de controlar os recursos disponíveis de uma rede é por meio de canais virtuais (ver Figura 2.4). Essa abordagem busca resolver o problema do *deadlock* em redes com chaveamento *Wormhole*, e consiste em organizar o *buffer* de entrada em filas de profundidade menor, denominadas canais virtuais, a fim de obter uma coleção de filas, as quais podem ser alocadas independentemente umas das outras. Desta forma, se um canal virtual estiver bloqueado por algum motivo, o canal físico ainda poderá ser usado por outro canal virtual.

Em Redes-em-Chip a implementação do controle de fluxo geralmente é baseado em crédito, uma vez que esse tipo de controle produz uma rede que nunca descarta dados. Desta forma, uma transmissão só é realizada quando o receptor tem espaço suficiente em *buffer*.

2.2.2.6 Memorização

Em virtude da possível indisponibilidade de canais de saída, quando do recebimento de pacotes na entrada dos roteadores, há a necessidade de implementar algum esquema de memorização (*buffer*) a fim de manter os pacotes bloqueados pelo roteador. A forma de implementação desses *buffers* interfere no desempenho do roteador. Os *buffers* podem ser organizados de forma independente ou compartilhada, bem como, podem se posicionar na entrada, saída ou serem centralizados. De acordo com as formas de organização dos *buffers*, existem diferentes técnicas de implementação de memorização:

memorização centralizada compartilhada, memorização na entrada e memorização na saída (Tamir, 1992).

A memorização centralizada compartilhada, também conhecida por *Centrally-Buffered, Dynamically-Allocated* – CBDA, é utilizada para armazenar os pacotes bloqueados de todas as portas de entrada e o espaço de endereçamento é dinamicamente distribuído entre os pacotes bloqueados. No pior caso essa abordagem deve oferecer uma largura de banda igual à soma das larguras de banda de todas as portas. Essa abordagem oferece uma utilização do espaço de memória melhor do que aquelas que alocam esse espaço previamente e estaticamente às portas de entrada. Nessa técnica pode ocorrer o problema de enchimento do *buffer*, quando uma saída está em uso por uma entrada e outra entrada com pacotes destinados a essa saída continuar a receber dados. A solução para esse problema é limitar o espaço disponível a cada porta.

A memorização na saída utiliza a estratégia de particionar o espaço de memória entre as saídas, que no caso podem implementar a alternativa FIFO. Essa implementação necessita que cada buffer suporte a demanda simultânea das portas de entrada, bem como, requer um controle de fluxo interno entre portas de entrada e portas de saída do roteador.

A memorização na entrada consiste na distribuição do espaço de memória sob a forma de partições entre as portas de entrada. Essa abordagem pode ser implementada por meio de várias estratégias, como: *buffer* First-In, First-Out – FIFO, *buffer Statically Allocated, Fully Connected* – SAFC, *buffer Statically Allocated, Multi-Queue* – SAMQ, *buffer Dynamically Allocated, Multi-Queue* – DAMQ.

A estratégia *buffer* FIFO implementa um espaço de memória fixo onde os dados são lidos na mesma ordem em que são escritos. Nessa alternativa de memorização pode ocorrer o problema do bloqueio na cabeça de linha (*Head-of-Line blocking* – HOL), que consiste no bloqueio do primeiro pacote do *buffer* por motivo de colisão ou pelo pacote ser maior que o espaço disponível no *buffer*, neste caso o canal físico também ficará bloqueado e nenhum pacote poderá utilizar esse canal. Essa abordagem é a alternativa mais simples e de menor custo.

A memorização na entrada SAFC apresenta uma solução para o bloqueio HOL da técnica FIFO. Essa técnica consiste em dividir cada *buffer* de entrada em N partições com tamanho igual a 1/N do tamanho do *buffer* original. Cada partição é atribuída a uma porta de saída em especial. Essa técnica gera alguns inconvenientes, tais como: custo adicional referente ao controle do núcleo de chaveamento e ao controle dos N *buffers* por porta de entrada; limitação do espaço de armazenamento total e um controle de fluxo mais complexo.

A alternativa de memorização na entrada SAMQ consiste em multiplexar as saídas dos *buffers* de entrada atribuídos a uma mesma porta de saída. Essa estratégia elimina alguns inconvenientes da SAFC, pois reduz o custo do *crossbar* (Silva, 2004).

2.2.3 Modelos de Programação

As seções anteriores apresentaram a infra-estrutura de hardware das Redes-em-Chip. Entretanto, as Redes-em-Chip também contemplam o nível da aplicação e neste nível os programadores precisam de um modelo de programação para efetivamente explorar o poder computacional e a flexibilidade das arquiteturas MPSoCs bastante heterogêneas.

Anterior ao surgimento do projeto de MPSoC, a busca por poder computacional fez com que a indústria de computadores buscasse soluções além da aceleração do relógio (*clock*). Baseando-se no conceito de paralelismo que consiste em obter resultados mais rápidos por meio da divisão de tarefas grandes e complexas em tarefas pequenas, distribuídas em vários processadores para serem executadas simultaneamente, adotou-se a solução dos computadores paralelos. Os computadores paralelos são constituídos basicamente por processadores, módulos de memória e rede de interconexão, que juntos buscam resolver um problema computacional.

Para exploração de arquiteturas paralelas, são necessários modelos de programação paralela (*Parallel Programming Models* – PPM). Segundo David Skillicorn (Skillicorn, 1998), um modelo é uma máquina abstrata que fornece algumas operações no nível de programação, e necessita de implementação para cada operação no nível inferior

(*hardware*). Os modelos de programação são interfaces que separam as propriedades de alto nível (*software*) das propriedades de baixo nível (*hardware*), e fornecem aos programadores de aplicação uma visão abstrata do *hardware*.

Na computação paralela tradicional, existem vários modelos de programação, como por exemplo: memória compartilhada, passagem de mensagem, *Bulk Synchronous Parallelism* – BSP, OpenMP, LogP, paralelismo de dados, entre outros (Foster, 1995; Jerraya, 2004a). Os modelos de programação estão relacionados à arquitetura de memória utilizada no ambiente computacional, uma vez que a comunicação entre os processadores depende da arquitetura da memória utilizada.

Os tipos de arquitetura de memória são: compartilhada e distribuída. Na arquitetura de memória compartilhada os processadores operam independentemente, mas compartilham o recurso de uma única memória; somente um processador acessa um endereço na memória por vez. Na arquitetura de memória distribuída os processadores operam independentemente onde cada um possui sua própria memória; os dados são compartilhados através da rede usando um mecanismo de passagem de mensagem (*Message Passing*).

Devido aos tipos de arquitetura de memória existentes, o paralelismo abrange características como: particionamento, que está relacionado com a decomposição de um programa em tarefas (*threads*); mapeamento, que está relacionado à alocação das tarefas nos processadores; comunicação e sincronização.

Em geral, os modelos de programação mais utilizados são memória compartilhada e passagem de mensagem. Esses modelos, como já foi mencionado anteriormente, estão diretamente relacionados à arquitetura de memória utilizada no ambiente de programação. No modelo de memória compartilhada a comunicação é executada implicitamente, acontecendo por meio do acesso de tarefas paralelas a posições do espaço de endereço compartilhado. No modelo de passagem de mensagem as tarefas têm espaço de endereço separado e a comunicação entre as mesmas é executada explicitamente por meio de primitivas como, por exemplo, *send()* e *receive()*. Neste modelo, os processos possuem acesso à memória local. A transferência dos dados entre

os processos requer operações de cooperação entre cada processo de forma que para cada operação de envio exista uma operação de recebimento.

Benini e De Micheli, em (Benini, 2004), observam alguns pontos positivos e negatrivos de ambos modelos. Os autores declaram que a paralelização de um código seqüencial, ou a escrita de um código paralelo, geralmente, é mais fácil de ser desenvolvido por meio do modelo de memória compartilhada, ao invés do modelo de passagem de mensagem. Entretanto, os autores também apontam que o modelo de memória compartilhada exige um intenso suporte de hardware para atingir alto desempenho, enquanto que o modelo de troca de mensagem – apesar da escrita difícil – atinge alto desempenho até mesmo em arquiteturas que implementam canais muito simples de comunicação .

O paradigma de passagem de mensagem dispõe da facilidade e generalidade de poder construir um mecanismo para troca de mensagens em qualquer linguagem, bem como adequar-se à ambientes distribuídos. Entretanto, existem algumas limitações que não o tornam muito viável, que consistem no fato do programador ser diretamente responsável pela paralelização, bem como, os custos de comunicação podem tornar extremamente proibitiva a transmissão de mensagens em um dado ambiente.

Os modelos de programação paralela comentados nesta seção são comumente utilizados pela computação de propósito geral. Entretanto, no domínio SoC, além de lidar com aspectos convencionais da programação paralela, lida-se também com características específicas, tais como a heterogeneidade e o paralelismo massivo. A próxima seção discorre sobre as questões dos modelos de programação paralela para SoC/MPSoC.

2.2.3.1 Modelos de Programação Paralela para SoC/MPSoC

A natureza dos sistemas multiprocessadores em um único chip guarda semelhanças com sistemas tradicionais de computação paralela, uma vez que consiste de processadores conectados, entre si, por meio de uma rede de interconexão. Entretanto, não se pode aplicar diretamente os modelos da computação paralela tradicional aos SoCs, pois estes

têm computações em tempo-real, restrições de área e energia, bem como, devem fornecer conexões apropriadas de entrada e saída.

O modelo de programação paralela para SoC deve atender as características fundamentais do sistema. As aplicações SoC, em geral, exigem alta performance e previsibilidade. Portanto, é necessário explorar as características específicas do SoC/MPSoC para usar os modelos de programação paralela convencionais de forma mais eficiente.

Segundo Jerraya (Jerraya, 2004b), as diferenças entre a programação paralela convencional e a programação MPSoC surgem mediante dois aspectos do projeto de software para MPSoC, que são a aplicação e a arquitetura. Os modelos de programação paralela convencionais oferecem suporte a qualquer tipo de programa. Considerando que o MPSoC oferece suporte apenas à aplicação específica, o modelo de programação para este tipo de sistema não precisa contemplar todo tipo de programação, entretanto deve ser eficiente para programação MPSoC.

As diferenças entre programação paralela convencional e programação MPSoC, quanto ao aspecto da arquitetura, consistem em duas características relevantes do MPSoC que são: a heterogeneidade dos componentes e o paralelismo massivo. Um MPSoC possui diferentes tipos de elementos de processamento, o sistema de memória pode ser heterogeneamente distribuído, a topologia de interconexão entre os elementos de processamento e a memória, também, podem ser heterogêneos o que torna a programação paralela para MPSoC mais complexa que a programação paralela convencional.

Do ponto de vista arquitetural, existem projetos de pesquisa tanto na academia como na indústria que propõem arquiteturas MPSoC de alto desempenho para aplicações de alto desempenho. Entretanto, o que se observa é que ainda existem limitações do número e tipos de processadores integrados; estrutura de memória; limitações quanto a escolha de componentes IPs e de redes de interconexão, bem como falta de suporte no nível alto de abstração.

Os modelos de programação em MPSoC devem oferecer acessos seguros e balanceados aos recursos de hardware, para tanto as tarefas devem ser alocadas e escalonadas para os componentes que são integrados por uma rede de interconexão. Além da alocação e do escalonamento de tarefas existe a preocupação com a transferência de dados entre a memória externa e os blocos de memória *on-chip*. A sincronização é um outro aspecto importante dos modelos de programação. A sincronização se torna mais complexa em sistemas cujo paralelismo é bastante utilizado, porém devido a muitas pesquisas já realizadas, algumas primitivas como semáforos, monitores e regiões críticas têm sido utilizados como solução para sincronização (Rego, 2006).

Segundo Luca Benini (Benini, 2004), o modelo de troca de mensagem é o mais adequado para software de aplicação de Redes-em-Chip. A motivação para essa afirmação baseia-se em três razões:

a) O software para sistemas de aplicação on-chip é desenvolvido mediante linguagens de especificação que usam fortemente a comunicação explícita entre as tarefas de execução paralela (modelos de fluxos de dados);

b) O desenvolvimento do código embarcado objetiva tradicionalmente o alto desempenho sobre recursos de hardware limitado, e, neste caso, o modelo de troca de mensagem pode conseguir um maior desempenho;

c) Por fazer comunicação explícita, o modelo de troca de mensagem localiza as fontes principais de latência e de transferências na execução do programa.

Levando em consideração a previsibilidade do modelo de troca de mensagem, que constitui uma das características importante e necessária em aplicações *on-chip*, o modelo de troca de mensagem se adéqua melhor ao SoC.

Na seção a seguir serão apresentadas algumas metodologias de Redes-em-Chip propostas para sistemas complexos como MPSoCs.

2.2.4 Metodologias para Projeto de Redes-em-Chip

As metodologias, em geral, são propostas para gerar um produto útil, e englobam algumas propriedades típicas como: funcionalidade, custo de fabricação, desempenho, consumo de energia entre outras propriedades (Wolf, 2001).

Além da função, desempenho e consumo de energia, um fluxo de projeto envolve objetivos muito importantes que devem ser alcançados, tais como: tempo para o produto estar disponível no mercado (*time-to-market*), custo e qualidade do projeto. Em geral, o produto que chega primeiro ao mercado pode conquistá-lo e já definir as preferências dos usuários para futuras versões do produto. O atraso na chegada do produto ao mercado leva a perda econômica, segundo a curva de retorno financeiro em relação ao tempo, conforme ilustrado na Figura 2.8.

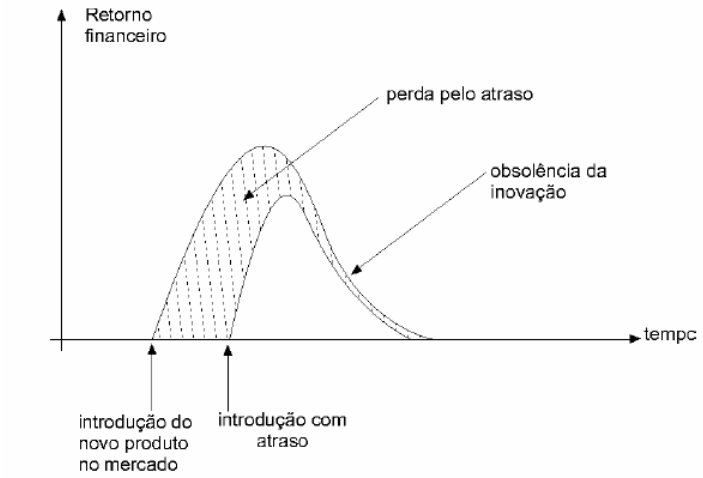

Figura 2.8 Retorno financeiro em relação ao tempo (Carro, 2003)

Tipicamente, o processo de projeto de um produto busca conciliar a obtenção das especificações do produto com as metas de custos pré-estabelecidas. O custo de projeto envolve os custos com pessoal e equipamentos envolvidos no projeto, que devem ser distribuídos pelas unidades vendidas.

Enquanto os usuários, comumente, exigem produtos de baixo custo e de qualidade, o mercado exige um rápido desenvolvimento dos mesmos. Portanto, é muito importante

que no início do projeto sejam definidos os níveis de corretude, confiabilidade e usabilidade para se obter um produto final de alta qualidade. Uma boa metodologia é decisiva para a construção de sistemas que trabalhe corretamente.

Para que as propriedades citadas acima sejam contempladas nos projetos de SoCs e MPSoCs, é extremamente importante a utilização de técnicas de reuso de componentes, arquiteturas, aplicações e implementações (Keating, 2002; Ramos, 2004). Projetos baseados em plataforma costumam adotar a técnica de reuso. Frank Vahid, por exemplo, em (Vahid, 2001) apresentou uma metodologia de projeto baseada em plataforma que além de reusar componentes, reusa também arquiteturas e topologias. A idéia fundamental é que uma arquitetura projetada para uma aplicação seja também eficiente e adequada para muitas aplicações semelhantes (Kumar, 2002). Do ponto de vista da estrutura de comunicação, em (Jantsch, 2003) é observado que sistemas baseados em Redes-em-Chip são economicamente viáveis se os mesmos podem ser usados em produtos diferentes, e se o projeto pode ser reusado em áreas de aplicações distintas.

Desde o surgimento da alternativa de Redes-em-Chip como estrutura de interconexão, várias metodologias de projeto foram introduzidas. Shashi Kumar *et al.* em (Kumar, 2002) apresentaram uma arquitetura e uma metodologia de projeto específica para uma topologia de malha bi-dimensional. Esta metodologia é constituída de duas fases, as quais geram um produto concreto a partir do mapeamento da aplicação em uma arquitetura de um modelo genérico de Rede-em-Chip. Esta metodologia necessita de simuladores para validação das várias opções arquiteturais e de comunicação.

Siegmund *et al*. apresentaram em (Siegmund, 2003) uma metodologia para modelar e sintetizar Redes-em-Chip em um fluxo de projeto baseado em SystemC. A metodologia é baseada em uma especificação de alto nível com refinamento incremental e simulação de um protocolo de comunicação. Jalabert *et al.* (Jalabert, 2004) propuseram uma ferramenta denominada *xpipescompiler* para instanciar Redes-em-Chip baseada em topologias definidas pelo usuário. A ferramenta permite que uma topologia seja estudada a partir de qualquer número de roteadores, interconexões com restrições de largura de banda e interfaces de rede para conectar componentes externos à Rede-em-Chip.

Kees Goossens *et al.* considerando que os serviços garantidos são a base para um modelo de programação, bem como uma exigência para as arquiteturas de hardware das Redes-em-Chip, propuseram em (Goossens, 2002, 2003) uma metodologia para fornecer serviços garantidos e de melhor esforço. Outras metodologias foram propostas para avaliar o projeto e implementações de características de Redes-em-Chip, tais como: tamanhos de pacote, modos de chaveamento, memorização, controle de fluxo entre outras características.

Aspectos relativos à utilização de área de silício e consumo de energia também foram abordados em outras propostas metodológicas. Segundo Vitkovski *et al.* (Vitkovski, 2004) a dissipação de energia dos canais de interconexão dos circuitos integrados gira em torno de até 50%. Para as Redes-em-Chip esse aspecto ainda se tornará mais crítico, devido a uma maior largura de banda, bem como uma maior área de silício. Observando esses aspectos, Luca Benini em (Benini, 2001) apresentou uma metodologia para reduzir o consumo de energia nas Redes-em-Chip, a qual consiste em técnicas aplicadas às camadas física, de enlace e de rede do modelo OSI reduzido.

Ainda considerando o aspecto da otimização do consumo de energia em Redes-em-Chip, Hu e Marculescu buscaram a minimização da energia por meio de um algoritmo que mapeia os componentes IPs em uma arquitetura de Rede-em-Chip regular e genérica (Hu, 2003). Ye *et al.* propuseram em (Ye, 2004), um modelo de energia para os roteadores da rede, por meio da definição da medida E_{bit}, que corresponde à energia consumida quando um bit de dado é transferido pelo roteador. Bhojwani *et al.* desenvolveram uma heurística, em (Bhojwani, 2005), que é capaz de controlar o consumo da energia de pico, aspecto crítico em sistemas com restrição de energia.

Uma Rede-em-Chip, do ponto de vista das características de implementação, usualmente, passa por descrições de diferentes níveis de abstração. Para validação funcional alguns pesquisadores implementam a Rede-em-Chip em FPGA (Brebner, 2003; Bobda, 2004; Silva, 2004). Contudo, as diversas implementações que utilizam FPGA são limitadas em flexibilidade e não permitem a realização de testes completos de uma Rede-em-Chip sobre o silício (Genko, 2005). Outra estratégia de implementação, é portar arquiteturas de Redes-em-Chip para FPGA a fim de validar

características baseadas em simulações de linguagens de descrição de hardware (Zeferino, 2003b; Moraes, 2004).

A linguagem de descrição de hardware VHDL é largamente empregada para avaliar latência e *throughput*, entre outras características de Redes-em-Chip. Em (Siguenza-Tortosa, 2002), por exemplo, os autores utilizaram VHDL para avaliar as características de canais virtuais em uma Rede-em-Chip com topologia baseada em malha. SystemC e XML também são utilizadas para avaliação de detalhes de diferentes alternativas arquiteturais de Redes-em-Chip. Pestana *et al.* utilizaram SystemC e XML para especificar roteadores e interfaces de rede, e testar alternativas de projetos de Redes-em-Chip baseada em malha (Pestana, 2004). A linguagem VHDL, usualmente, apresenta um elevado tempo de simulação em relação à linguagem SystemC, então algumas metodologias adotam SystemC ou uma combinação de VHDL/SystemC (Chan, 2004) para avaliar arquiteturas de Redes-em-Chip.

Considerando limitada a velocidade de simulação e a precisão de estimativas das linguagens acima citadas, Genko *et al*. (Genko, 2005) propuseram um método de emulação visando uma rápida exploração das principais características de Redes-em-Chip parametrizáveis. Outras propostas sugerem o uso de linguagens de alto nível como C e C++. Wiklund *et al.* (Wiklund, 2004), por exemplo, propuseram um simulador em C++ para melhorar a velocidade de simulação em precisão de ciclo. Contudo, as linguagens de programação de alto nível não dispõem de recursos semânticos para descrição de aspectos de hardware.

Em geral, as metodologias propostas acima são definidas para uma topologia específica ou para componentes e tecnologias de Redes-em-Chip personalizadas. Ademais, utilizam basicamente como técnicas de validação a simulação, emulação e validações em FPGAs. Conforme mencionado na seção 1.3, a simulação apresenta restrições quanto ao grande número de casos de testes necessários para validar parcialmente um sistema, enquanto a emulação, apesar de apresentar um tempo menor de execução em relação à simulação, dificilmente detectará erros de temporização (Nam, 2000). Por outro lado, as validações utilizando FPGAs enfrentam o empecilho da exploração arquitetural dos parâmetros envolvidos no projeto, tais como: tamanho do pacote,

número de roteadores ou topologias que implicam em um re-projeto da implementação física sobre o FPGA o qual consome muito tempo.

Considerando as limitações das técnicas de validação, acima citadas, amplamente utilizadas nos projetos de desenvolvimento de sistemas, ressalta-se a necessidade de metodologias que eliminem, ainda nas fases iniciais de projeto, o maior número possível de falhas, a fim de diminuir o tempo e os altos custos das fases de implementação e testes, bem como oferecer uma maior corretude ao sistema. Segundo Potter *et al.* (Potter, 1996), uma grande porcentagem dos erros encontrados nas fases de implementação e testes são decorrentes de imprecisões nas fases iniciais do projeto.

A complexidade crescente dos sistemas tem motivado o uso de melhores técnicas de especificação e verificação. Uma das técnicas adotadas em diferentes pesquisas é o método formal. O método formal pode assumir diferentes formas, que pode variar desde notações matemáticas feitas com especificações em linguagem natural, até especificação formal completa utilizando linguagens de especificação com uma semântica precisa (NASA, 1995).

Provavelmente, a possibilidade de poder detectar falhas nas fases iniciais do ciclo de vida do projeto, ao contrário das linguagens de descrição de hardware que só detectam as falhas na fase de simulação ou verificação funcional, tem incentivado o uso de métodos formais em diferentes projetos. Existem muitas pesquisas em diferentes domínios de aplicação que utilizam métodos formais como instrumento de especificação, algumas delas estão relacionadas com a verificação de sistemas automotivos (Beyer, 2005), operadores de ponto-flutuante (Sawada, 2002), microprocessadores (Sawada, 1998) e ferramentas de verificação simbólica (Sammane, 2004).

Considerando as metodologias de projeto no domínio de comunicação dos MPSoCs, que utilizam métodos formais, pode-se citar a pesquisa de Schmaltz e Borrione, que utilizou a lógica ACL2 (Kauffman, 2000) para especificar formalmente a Rede-em-Chip Octagon (Karim, 2002) e desenvolver uma metodologia de especificação e verificação funcional para os primeiros passos de módulos virtuais para comunicação parametrizada (Schmaltz, 2004a). Outrossim, Schmaltz e Borrione em (Schmaltz, 2004b; Schmaltz,

2005) desenvolveram algoritmos de roteamento e escalonamento e provaram que uma mensagem, sem ser alterada, em algum momento no futuro alcançará o seu destino ou será perdida. Esses trabalhos desenvolveram um modelo genérico para Redes-em-Chip e uma biblioteca específica de funções e teoremas que pode ser utilizada em projetos com funcionalidades semelhantes.

Visando o desenvolvimento de um método para roteamento assíncrono para sistemas baseados em Redes-em-Chip, Tsiopoulos e Valdén (Tsiopoulos, 2006) utilizaram o formalismo *B Action System* (Waldén, 1998) para especificação formal dos roteadores de comunicação. Outros métodos usando o formalismo são apresentados em (Gangwal, 2005), que utiliza um método analítico para calcular a latência, *throughput* e requisitos de armazenamento para a Rede-em-Chip Æthereal da Philips.

Por meio do formalismo é possível documentar um projeto complexo e, auxiliar os projetistas a esclarecer as escolhas de projetos e resolver inconsistências problemáticas nas fases iniciais do ciclo de vida. Um método analítico, por exemplo, pode verificar um sistema sem a necessidade de simulações.

Devido à complexidade dos MPSoCs que contêm múltiplos componentes de hardware e software em torno de uma estrutura de comunicação, e à grande variedade de soluções possíveis objetivando atender os requisitos de desempenho, consumo de energia, área ocupada, entre outros requisitos de projeto, é essencial que o projeto do sistema seja realizado em níveis de abstração elevados (Carro, 2003).

Portanto, considerando as observações constantes em (Carro, 2003) e com o intuito de fornecer um suporte mais sólido às técnicas de simulação largamente utilizadas, será apresentada na próxima seção uma modelagem baseada em linguagem de especificação formal, a qual se constitui um método de projeto que procura validar, tão cedo quanto possível, as propriedades desejadas de um sistema de Redes-em-Chip.

3. A Metodologia *CAD*$\mathbb{Z}$

A tendência atual é que a diversidade das exigências computacionais dos MPSoCs, utilizados por novas aplicações, aumente a cada dia, exigindo um bom desempenho da estrutura de comunicação que interliga os diversos elementos computacionais. Conforme discutido em (Kumar, 2002; Jantsch, 2003; Benini, 2004), está claro que um dos problemas fundamentais do projeto de sistemas embarcados, mais especificamente dos MPSoCs, é a estrutura de comunicação.

A metodologia *CAD*$\mathbb{Z}$ difere das metodologias tradicionais pela utilização de Z na modelagem formal, que traduz as exigências específicas de cada projeto. Por meio da modelagem formal em Z, busca-se permitir ao usuário definir de forma rigorosa e incremental as propriedades fundamentais do elemento de comunicação, possibilitando a avaliação qualitativa das propriedades nas fases iniciais do projeto. Esta busca tem como conseqüência a obtenção de uma biblioteca de funções, que poderão ser reusadas para circuitos de funcionalidade semelhante. A Figura 3.1 descreve graficamente as fases da metodologia *CAD*$\mathbb{Z}$ e sua inclusão em um fluxo tradicional de projeto.

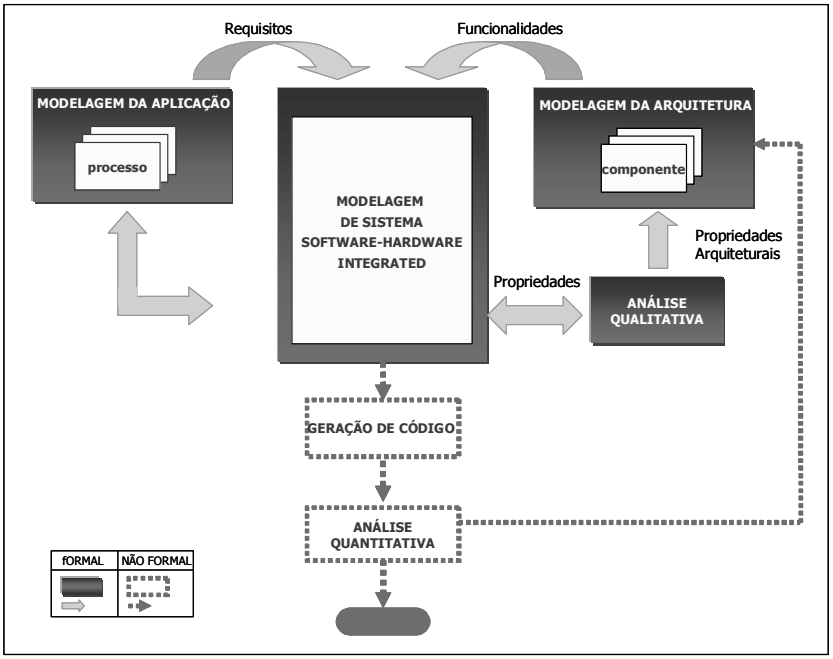

Figura 3.1 Metodologia *CAD*$\mathbb{Z}$

Assim sendo, as fases da metodologia proposta estão concentradas na parte formal que antecede as fases da parte não formal. Esta parte consiste das seguintes etapas: Modelagem, Integração e Análise Qualitativa. A fase de Modelagem é sub-dividida em modelagem da aplicação (*Application Modeling*) e da arquitetura (*Architecture Modeling*). Esta separação permite que diferentes equipes trabalhem em paralelo.

Como a ênfase de aplicação da metodologia proposta nesta tese é o projeto de redes-em-chip, a modelagem da aplicação foi simplificada de forma que seus requisitos (*requirements*) são expressos através do grafo de comunicação das tarefas (*Process*), enquanto que a modelagem da arquitetura envolve especificação formal da infra-estrutura de comunicação da Rede-em-Chip traduzindo a funcionalidade (*functionality*) requerida.

A fase de Integração tem como objetivo a junção entre os elementos obtidos na fase de modelagem, ou seja, aplicação e arquitetura, gerando então uma especificação integrada de software e hardware (*integrated software-hardware system modeling*). Nesta fase são feitas verificações de compatibilidade entre os vários elementos.

A fase seguinte, Análise Qualitativa (*Qualitative Analysis*), corresponde à análise de propriedades (*properties*) que devem ser atendidas pelo projeto integrado, ou seja, refere-se ao mapeamento entre os requisitos da aplicação e a funcionalidade da arquitetura. Exemplos de propriedades incluem latência, vazão, performance, consumo de energia e área utilizada. Como resultado desta análise qualitativa, é possível que a necessidade de ajustes seja identificada. Considerando que as aplicações têm demandas que devem ser atendidas, tais ajustes só recaem sobre propriedades arquiteturais (*architectural properties*), assim sendo, um novo ciclo de modelagem arquitetural é iniciado como o objetivo de adequar a arquitetura aos requisitos impostos pela aplicação.

Obtida uma especificação que satisfaça as propriedades exigidas, um mecanismo de geração de código é usado. Embora existam iniciativas de automatização deste processo (Carro, 2003) no fluxo do projeto, tal mecanismo pode ser manual, ou seja, a partir da especificação formal validada um programador escreve o código em uma linguagem de programação. Dado o rigor característico da aplicação de métodos formais, é correto

supor que tal especificação propicie uma maior clareza e, portanto contribua para minimização de erros de codificação.

Após a geração do código são iniciados testes, que são técnicas de análise quantitativa (*quantitative analysis*). Esta etapa segue as orientações de planejamento comuns à validação de software. Caso os resultados demonstrem a necessidade de ajustes, é re-iniciado um novo ciclo de modelagem arquitetural a exemplo da análise qualitativa, descrita anteriormente.

A metodologia *CAD*$\mathbb{Z}$, portanto, tem o claro objetivo de aumentar a qualidade dos projetos pela incorporação de técnicas formais em fases iniciais do projeto. Embora tais técnicas possam representar um esforço adicional da equipe de projetistas, a metodologia prevê o reuso de especificações, o que na prática significa que é possível reduzir os impactos de aprendizado identificando projetistas especificadores que trabalharão na especificação formal de componentes que serão utilizados pelos demais em diferentes projetos. Considerando, os custos normalmente associados aos projetos, tal esforço pode significar a identificação rápida de erros que podem comprometer o seu sucesso.

Dada a complexidade dos elementos envolvidos em projetos de redes-em-chip, a estratégia adotada se baseia em refinamentos sucessivos. Inicialmente, definem-se os elementos arquiteturais em um nível de abstração alto que serão sucessivamente refinados até o nível que o projetista especificador julgar necessário. Considerando que a abstração é realizada por meio da notação Z, a próxima seção apresenta uma visão geral da mesma, de forma a facilitar a compreensão.

3.1 Notação Z

Conforme citado na seção 1.5, a modelagem formal utiliza linguagem de especificação formal, mais precisamente a notação Z, a qual é baseada na teoria de conjuntos e orientada para construção de modelos. A forma básica utilizada pela notação Z é denominada esquema (em inglês *schema*), que é usado para introduzir axiomas de

funções. Os modelos são construídos pela especificação de uma série de esquemas usando tipicamente um estilo de transição de estado.

A teoria dos conjuntos, como mencionado anteriormente, é um dos fundamentos da notação Z onde os tipos básicos são conjuntos definidos no início da especificação. Um dado conjunto inicial, denominado em inglês *given set*, é um poderoso elemento abstrato de Z, representado por nomes dentro de colchetes, do qual uma especificação formal é construída. Conjuntos enumerados também são permitidos na notação Z. Relações e funções são elementos chaves para a modelagem abstrata e estruturas de programação de dados tais como tabelas, listas, matrizes, etc. Lógica de primeira ordem e operações da teoria dos conjuntos são utilizadas para especificar propriedades dinâmicas e invariantes de dados.

O elemento básico da estrutura de especificação em Z é o esquema (Spivey, 1992). Os esquemas são compostos por declaração e predicado, conforme exibe a Figura 3.2. A parte da declaração de um esquema declara as entidades que serão melhor definidas na parte do predicado.

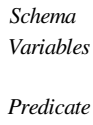

Figura 3.2 The Z Schema

Os esquemas em Z são utilizados para descrever os aspectos estáticos e dinâmicos de um sistema. Os aspectos estáticos estão relacionados com o estado global do sistema, bem como com os relacionamentos entre seus componentes, também denominado invariante. Um controle rígido sobre a integridade do estado é acompanhado pelo invariante durante qualquer operação que altera o estado. Os aspectos dinâmicos incluem todas as operações que manipulam os elementos dos estados, mesmo em tarefas de consulta (Spivey, 1992).

Os esquemas podem ser combinados de várias formas, este aspecto da notação Z é conhecido como cálculo de esquema (*schema calculus*) e contribui bastante, tanto do

ponto da notação como conceitual para facilitar a separação de conceitos na especificação Z.

Para ilustração, será considerado um simples exemplo de uma especificação genérica de uma Rede-em-Chip, que na Figura 3.3 está denominada pelo acrônimo *NoC*. Uma Rede-em-Chip é composta por roteadores e canais entre esses roteadores. A Figura 3.3 exibe um esquema dessa abstração, onde a existência de um canal entre dois roteadores *r1* e *r2* é modelado pelo fato que o par *(r1,r2)* pertence à relação *links*. A parte do predicado expressa que todos os roteadores conectados por um *link* devem ser declarados como sendo roteadores dessa *NoC*, ou seja, ambos *r1* e *r2* devem pertencer ao componente *routers* da *NoC*. Em seguida é declarado que não existe um canal entre um roteador e ele mesmo. O uso do operador de conjunto potência (em inglês, *power set*) significa, de maneira prática, que toda *NoC* é constituída de elementos do tipo *ROUTER*. Esse esquema de estado global, a grosso modo, corresponde à parte dos atributos de uma classe no paradigma orientado a objeto e pode ser mais tarde instanciado para definir instâncias de NoC, como no esquema parcial exibido na Figura 3.4, onde se define que um SoC é constituído de uma NoC.

```
[ROUTER]

   NoC
routers:   ROUTER
links:     ROUTER    ROUTER

  r1, r2: ROUTER     r1  r2     links
     r1    routers    r2    routers    r1    r2     r2  r1     links
```

Figura 3.3 Exemplo de uma NoC genérica

```
  SOC
anoc: NoC
...
```

Figura 3.4 Esquema Parcial SoC

Uma vez que as características do estado global tenham sido definidas, pode-se definir as operações para manipular as variáveis do estado, nesta ilustração, roteadores e canais, semelhante aos métodos no paradigma orientado a objeto. Entretanto, Z não é uma

estrutura orientada a objeto. Encapsulamento, por exemplo, não é suportado. Uma operação para adicionar um link à *NoC* pode ser definida pelo esquema ilustrado pela Figura 3.5.

AddLinkRouters
ΔNoC
r1?: ROUTER
r2?: ROUTER

r1? $\in$ *routers*
r2? $\in$ *routers*
r1? $\neq$ *r2?*
routers' = *routers*
links' = *links* $\cup$ {*r1? r2?*}

Figura 3.5 Esquema *AddLinkRouters*

O Esquema acima especifica uma operação que pode alterar o estado de uma *NoC*, indicado por Δ *NoC* na parte declarativa, e deve receber duas entradas *r1?* e *r2?* (em Z, as variáveis de entrada são indicadas por um ponto de interrogação no fim do nome da variável), ambos roteadores. Essa operação não tem saída (em Z, as variáveis de saída são indicadas por um ponto de exclamação no fim do nome da variável). Os valores das variáveis de estado antes da execução da operação são indicadas por seus nomes e seus valores, depois da execução da operação são indicados por seus nomes seguidos de um apóstrofe. Na parte do predicado são declaradas as pré e pós-condições da operação.

As pré-condições se referem somente às variáveis de entrada e ao estado global antes da execução das operações especificadas, enquanto que as pós-condições se referem ao estado global após execução da operação especificada. Pré-condições expressam as condições que devem ser verdadeiras antes da chamada da operação, de tal forma que a execução da operação preserve o invariante. Pós-condições definem as condições que devem ser verdade, além do invariante, após a execução da operação, isto é, o comportamento especificado da operação. Neste exemplo, as pré-condições da operação *AddLinkRouters* declaram que *r1?* e *r2?* são diferentes e pertencem a *NoC* (*r1?* $\in$ routers, *r2?* $\in$ routers e *r1?* $\neq$ *r2?*), e as pós-condições declaram que após a execução da operação nenhuma alteração deve ter ocorrido ao conjunto de roteadores da *NoC* e que

mais um canal deve ter sido adicionado, conectando *r1?* a *r2?* (*links' = links* $\cup$ *{(r1?, r2?)}*).

A corretude do comportamento especificado nessa operação, em relação às restrições sobre as variáveis do estado global (o invariante), pode ser verificada por meio da verificação formal. Uma obrigação de prova pode ser extraída para garantir que o estado alcançado após a execução da operação é válido, isto é, satisfaz ao invariante da *NoC*, enquanto as pré-condições da operação são respeitadas. Esta tarefa, usualmente, é executada por ferramentas.

Finalmente, o refinamento pode ser executado por meio da inserção de componentes de estado, transformando aqueles existentes em tipos de dados mais concretos (implementáveis), restringindo o comportamento das operações, etc. Os refinamentos também podem ser formalmente verificados.

A ferramenta utilizada para analisar as especificações em Z é a ferramenta Z/EVES (Saaltink, 1997a), que pode ser usada para análise da estrutura gramatical, verificação de tipos, verificação de domínio, expansão de esquema, cálculo de pré-condição e provas de teoremas. Outra ferramenta utilizada como suporte, é o animador de especificação Possum que interpreta e enumera estruturas especificadas usando a notação Z (Hazel, 1997).

3.2 Modelagem *CAD*$\mathbb{Z}$

Esta seção trata das modelagens da aplicação e da infra-etrutura de comunicação das Redes-em-Chip, que são realizadas a partir da notação Z. A sub-seção 3.2.1 aborda sobre a modelagem da aplicação e seus requisitos, enquanto que a sub-seção 3.2.2 trata da modelagem da infra-estrutura de comunicação das Rede-em-Chip, que inclui as propriedades relacionadas aos requisitos da aplicação, os mecanismos básicos de rede e as estruturas de interconexão.

A Figura 3.6 exibe a modelagem da infra-estrutura de comunicação por meio de perspectivas distribuídas em três camadas. A primeira camada (*Properties*) representa a

camada mais abstrata. Nesta camada estão as propriedades relacionadas aos requisitos da aplicação. Para que seja possível atender os requisitos das propriedades associadas à aplicação, existem os mecanismos básicos de rede que estão agrupados em uma camada com um nível menor de abstração (*Basic Network Mechanisms*). Os mecanismos básicos de rede geralmente estão vinculados a uma estrutura de interconexão, cujas topologias estão representadas na última camada (*Interconnection Structures*).

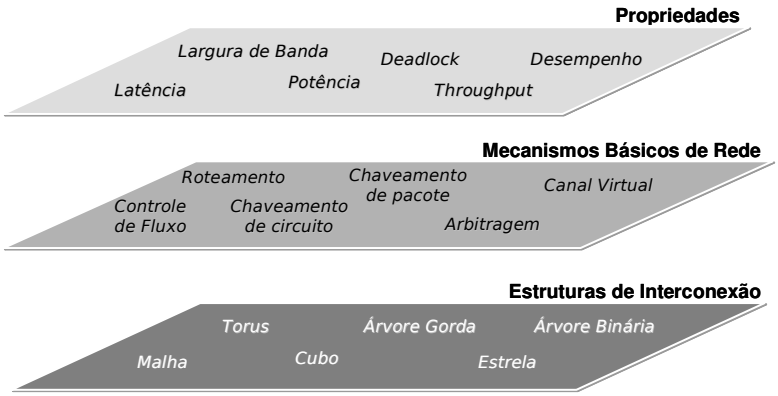

Figura 3.6 Perspectiva da Modelagem

As camadas acima se interrelacionam e apresentam um movimento cíclico. Por exemplo, supondo que o principal requisito de uma aplicação seja nível baixo de consumo de energia, ou seja, a propriedade energia. Então, os mecanismos básicos de rede que serão utilizados devem apresentar características de baixo consumo de energia, bem como, a topologia utilizada deve atender o baixo consumo de energia exigido pela aplicação.

3.2.1 Modelagem da Aplicação

O projeto de um SoC envolve uma questão muito importante com relação à generalidade e ao desempenho. Enquanto a generalidade promove a reusabilidade do hardware, do sistema operacional e de técnicas de desenvolvimento, a busca por um melhor desempenho utiliza estruturas de aplicação específica. Um projeto baseado em Redes-em-Chip, por exemplo, nem sempre será a melhor solução para todo tipo de aplicação. Porém, espera-se que projetos baseados em Redes-em-Chip forneçam boas

soluções para produtos que possam ser reconfigurados e programados, produtos que tendem a evoluir com o lançamento de novas gerações, versões, devido a reusabilidade da Rede-em-Chip, bem como para aplicações com uma mescla de tarefas heterogêneas (Kumar, 2002).

As aplicações comumente possuem requisitos que estão associados às propriedades de uma estrutura de interconexão (ver primeira camada Figura 3.6). Estes requisitos, na metodologia $CAD\mathbb{Z}$, são modelados como um conjunto enumerado (ver Figura 3.7) e fazem parte da modelagem da aplicação. Contudo, na modelagem da aplicação os requisitos são associados a níveis. Por exemplo, supondo que um dos requisitos da aplicação seja a latência de comunicação, este requisito será modelado por meio da informação referente ao nível de latência esperado, ou seja, se é baixa, média, alta, máxima ou mínima. Portanto, na metodologia $CAD\mathbb{Z}$ a modelagem da aplicação contempla os requistos da mesma.

requirements :: *latency*
power
throughput
bandwidth
performance
deadlock

Figura 3.7 Modelagem dos requisitos de uma aplicação

Assim como nos trabalhos de (Lei, 2003; Murali, 2004; Jalabert, 2004; Hansson, 2005), a aplicação é especificada como um grafo dirigido G(P, R). O grafo exibe a estrutura de uma aplicação, em que cada nó do grafo corresponde a um processo da aplicação que pertence a P. As arestas do grafo representam a ligação entre os processos e pertencem a R. O rótulo das arestas pode ser associado a diferentes requisitos. Por exemplo, em uma dada análise os valores das arestas podem representar o atraso máximo de comunicação, ou o consumo de energia máximo, ou ainda a largura de banda entre dois nós.

A formalização da aplicação (ver Figura 3.8) consiste na definição do conjunto de processos (*processes*) que compõem à estrutura da aplicação, da ligação (*commProcess*) e do rótulo (*label*) entre os processos, dos requisitos da aplicação (*requirements*) com

seus respectivos níveis (*requireLevel*), bem como da quantidade de processos da aplicação (*processesSize*).

Application
processes: Process
commProcess: Process Process
label: Process Process
requireLevel: requirements level
processesSize:

processesSize 2
dom *label commProcess*
p1, p2: Process p1 processes p2 processes p1 p2
p1 p2 commProcess

Figura 3.8 Esquema do grafo de comunicação da aplicação

3.2.2 Modelagem da Infra-Estrutura de Comunicação

Considerando que uma Rede-em-Chip é uma estrutura de interconexão, a mesma pode ser descrita por sua topologia, política de roteamento e controle de fluxo (Carro, 2003). Segundo Duato *et al*. (Duato, 1997), existem outros parâmetros que também são utilizados para descrever uma rede de interconexão, por exemplo, a arbitragem. Esta tese descreve as Redes-em-Chip segundo a topologia, roteamento, chaveamento, arbitragem e controle de fluxo, necessários para atender as propriedades inerentes ao sistema.

Esta seção trata do conjunto de termos e das definições necessárias para descrição de uma infra-estrutura de comunicação baseada em Redes-em-Chip, as quais são modeladas formalmente, segundo a metodologia $CAD\mathbb{Z}$, a fim de permitir uma análise qualitativa da estrutura por parte do projetista do sistema.

3.2.2.1 Propriedades

De um modo geral, o desenvolvimento de um projeto contempla propriedades que precisam ser validadas. Uma rede de interconexão, por exemplo, pode ter o seu desempenho avaliado pela sua largura de banda (*bandwidth*), vazão (*throughput*), latência (*latency*) máxima, potência máxima consumida na comunicação, atraso (delay) máximo no canal físico, etc. A Figura 3.9 exibe algumas propriedades relacionadas aos sistemas de interconexão.

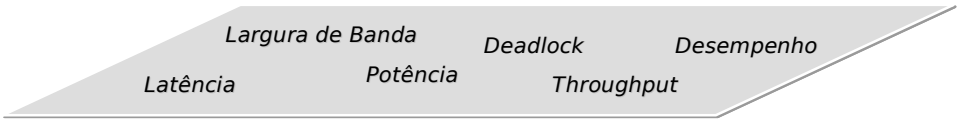

Figura 3.9 Propriedades

As propriedades são declaradas inicialmente em um alto nível de abstração, ou seja, no início da análise as mesmas podem indicar apenas o nível desejado (alto, médio, baixo, máximo, mínimo) e no decorrer da análise essas propriedades são refinadas podendo expressar faixas de valores que estão dentro daquele nível inicialmente declarado. A seguir são apresentadas algumas modelagens de propriedades, que exemplificam o refinamento citado acima.

3.2.2.1.1 Latência (*Latency*)

A latência de comunicação é uma propriedade que envolve a soma dos vários atrasos que podem ocorrer na transferência de dados entre os nós de origem e destino de uma Rede. A latência da rede é muito dependente da abordagem de chaveamento utilizada, ou seja, se o chaveamento é realizado por meio de circuito ou de pacotes.

A latência de uma rede chaveada por pacote, sem contenção, segundo (Ni, 1993) é definida pelo tempo necessário para uma mensagem de comprimento *L* passar através de um canal de largura de banda *B*, vezes a quantidade de nós (*D*) entre a origem e o destino, ou seja, (L/B) D, onde L é o comprimento (*Length*) do pacote, B é a largura de banda (*Bandwidth*) e D é a distância ou comprimento do caminho entre os nós de

origem e destino. Esta definição é adequada para o chaveamento *Store-and-Forward* – SAF, e para os demais tipos de chaveamento esta definição é ajustada.

Conforme introduzido por (Kermani, 1979), no chaveamento *Virtual-Cut-Through* – VCT o pacote só será armazenado em um nó intermediário se o próximo canal requisitado estiver ocupado. Então, a latência da rede para este chaveamento é definida como $(L_h/B)D + L/B$, onde L_h é o comprimento do campo do cabeçalho. No caso do chaveamento *Wormhole,* onde um pacote é dividido em *flits* para transmissão, e o *flit* cabeçalho orienta a rota, a latência é definida por $(L_f/B)D + L/B$, onde L_f é o comprimento de cada *flit*.

O chaveamento de circuito, conforme mencionado na seção 2.2.2.2, constrói um circuito físico entre a origem e o destino, cujos canais integrantes do circuito são reservados exclusivamente para o mesmo. Portanto, neste tipo de chaveamento não há necessidade de *buffers* nos nós intermediários e a latência é definida por $(L_c/B)D + L/B$, onde Lc é o comprimento do pacote de controle transmitido na fase de estabelecimento do circuito.

Duato, em (Duato, 1997), esclarece que a latência das técnicas de chaveamento envolve o tempo que um roteador leva para rotear um pacote, o tempo de propagação do pacote no canal (*link*) entre os roteadores e o tempo de um pacote ser enviado do *buffer* de entrada para o *buffer* de saída do roteador, este último depende do esquema de memorização (seção 2.2.2.6) utilizado. Esta definição de Duato é aplicada às Redes-em-Chip.

A latência, além de ser uma propriedade da infra-estrutura de rede, consiste em um requisito da aplicação. Mediante esse contexto, a latência é especificada inicialmente, como um requisito da aplicação que apresenta um nível de exigência, por exemplo, latência máxima. Como a latência resulta da soma dos vários tempos abordados no parágrafo anterior, esta propriedade é refinada de modo a apresentar os elementos que compõem a propriedade. A Figura 3.10 exibe o refinamento da propriedade latência, que representa a especificação do atraso envolvido no roteamento de um pacote na rede.

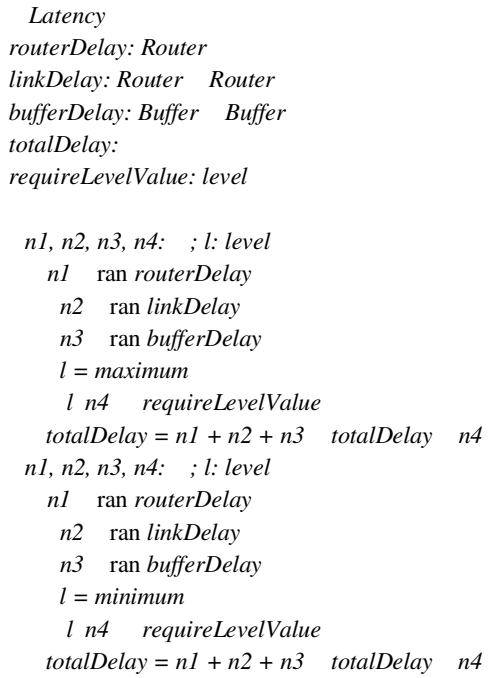

Figura 3.10 Esquema referente à Latência de comunicação

O esquema *Latency* é constituído pela função *routerDelay* que define o tempo gasto pelo roteador para rotear um pacote, pela função *linkDelay* que define o tempo de propagação do pacote através do link entre dois roteadores. O tempo de envio de um pacote armazenado no *buffer* de entrada para o *buffer* de saída é representado pela função *bufferDelay*. A variável *totalDelay* representa a soma de todos os atrasos existentes na comunicação entre roteadores adjacentes. A função *requireLevelValue* associa um valor ao nível de atraso exigido pela aplicação.

Considerando que em alguns projetos os *buffers* são implementados em registradores, o esquema *Delay* pode ser composto apenas das funções que definem os atrasos referentes aos roteadores (*routerDelay*) e aos *links* (*linkDelay*).

Para especificar a latência total de uma rota é necessário identificar o atraso dos componentes que compõem a rota. Então, o esquema *RoutingLatency*, ilustrado na Figura 3.11, associa a cada componente da rota o *delay* respectivo. Este esquema consiste da conjunção dos esquemas *Routing* e *Latency*, bem como, das restrições necessárias para vincular os atrasos dos roteadores. Neste caso, por meio da conjunção

de esquema um novo esquema é formado pela mescla das declarações e conjunção dos predicados (Woodcock, 1996). Em Z, e por conseguinte na metodologia $CAD\mathbb{Z}$, por meio da conjunção de esquemas é possível especificar diferentes aspectos de uma especificação separadamente, e combiná-las para formar uma descrição completa.

$$
\begin{array}{l}
\quad RoutingLatency \\
Routing \\
Latency \\
totalRouteLatency: ID \\
auxl: \quad Router \quad Router \\
auxr: \quad Router \\
\\
\mathrm{dom}\ totalRouteLatency = \mathrm{dom}\ routeInfo \\
\mathrm{dom}\ auxl = \mathrm{ran}\ routeElements \\
\quad id: ID \quad id \quad \mathrm{dom}\ routeElements \\
\quad\quad \mathrm{dom}\ routeElements\ id \quad \mathrm{ran}\ routeElements\ id \quad \mathrm{dom}\ routerDelay \\
\quad id: ID; r1, r2: Router \quad id \quad \mathrm{dom}\ routeElements \quad r1\ r2 \quad routeElements\ id \\
\quad\quad r1\ r2 \quad \mathrm{dom}\ linkDelay \\
\quad id: ID; r1, r2: Router \\
\quad\quad id \quad \mathrm{dom}\ routeElements \\
\quad\quad\quad r1\ r2 \quad routeElements\ id \\
\quad\quad routeElements\ id \quad \mathrm{dom}\ auxl \\
\quad\quad r1\ r2 \quad \mathrm{dom}\ linkDelay \quad r1 \quad \mathrm{dom}\ routerDelay \quad r2 \quad \mathrm{dom}\ routerDelay \\
\quad route: \quad Router \quad Router \quad route \quad \mathrm{dom}\ auxl \\
\quad\quad route = \quad auxl\ route = 0 \\
\quad\quad\quad route \\
\quad\quad\quad\quad link: Router \quad Router \\
\quad\quad\quad\quad link \quad route \\
\quad\quad\quad\quad\quad link \quad \mathrm{dom}\ linkDelay \\
\quad\quad\quad\quad\quad route \setminus link \quad \mathrm{dom}\ auxl \\
\quad\quad\quad\quad auxl\ route = linkDelay\ link + auxl\ route \setminus link \\
\quad router1: \quad Router \quad router1 \quad \mathrm{dom}\ auxr \\
\quad\quad router1 = \quad auxr\ router1 = 0 \\
\quad\quad\quad router1 \\
\quad\quad\quad\quad router2: Router \\
\quad\quad\quad\quad router2 \quad router1 \\
\quad\quad\quad\quad\quad router2 \quad \mathrm{dom}\ routerDelay \\
\quad\quad\quad\quad\quad router1 \setminus router2 \quad \mathrm{dom}\ auxr \\
\quad\quad\quad\quad auxr\ router1 \\
\quad\quad\quad\quad = routerDelay\ router2 + auxr\ router1 \setminus router2 \\
\quad id: ID \\
\quad\quad id \quad \mathrm{dom}\ routeElements \\
\quad\quad\quad \mathrm{dom}\ routeElements\ id \quad \mathrm{dom}\ auxr \\
\quad\quad\quad \mathrm{ran}\ routeElements\ id \quad \mathrm{dom}\ auxr \\
\quad\quad\quad routeElements\ id \quad \mathrm{dom}\ auxl \\
\quad\quad totalRouteLatency\ id \\
\quad\quad = auxl\ routeElements\ id + auxr\ \mathrm{dom}\ routeElements\ id \\
\quad\quad\quad + auxr\ \mathrm{ran}\ routeElements\ id
\end{array}
$$

Figura 3.11 Esquema da rota com a latência associada

O esquema *RoutingLatency* (ver Figura 3.11) inclui também as variáveis *auxl* e *auxr*, que auxiliam no cálculo recursivo do atraso provocado pelos *links* (*auxl*) e roteadores (*auxr*) que compõem a rota, como também insere a função *totalRouteLatency*, a qual acumula a soma de todos os atrasos dos componentes da rota. As sentenças constantes no invariante do esquema restringem o cálculo da latência exclusivamente para as rotas cujos identificadores são comuns.

3.2.2.1.2 Deadlock

Conforme abordado na seção 2.2.2.1 o *deadlock* consiste em uma situação que pode postergar a entrega de um pacote indefinidamente, uma vez que um conjunto de pacotes pode ficar bloqueado para sempre na rede. Segundo Ni e McKinley (Ni, 1993), a situação de *deadlock* ocorre quando é permitido que um pacote que está ocupando algum recurso, requisite um outro recurso. Os recursos podem ser *buffers* ou *links*. Comumente, os *buffers* constituem os recursos utilizados no chaveamento *Store-and-Forward* e no chaveamento *Virtual Cut-Through*, enquanto que os *links* são os recursos utilizados no chaveamento de circuito e *Wormhole*.

Os *buffers* são componentes internos dos roteadores, mas que podem ser implementados por registradores. A especificação dos mesmos é realizada em um nível menos abstrato, onde os componentes internos do roteador são especificados. Em um alto nível de abstração, os roteadores são visualizados como uma caixa preta. Considerando, a especificação de uma Rede-em-Chip na fase inicial do projeto, a qual é realizada em um nível de abstração alto, a especificação da configuração de *deadlock* considera os *links*.

O esquema *DeadlockConfiguration* apresentado na Figura 3.12 exibe uma configuração de ocorrência de *deadlock*, especificada de acordo com a situação definida em (Ni, 1993). Na parte da declaração são incluídos os esquemas *NoC* e *Routing*, os quais juntos especificam as rotas de uma NoC. Além da inclusão dos esquemas *NoC* e *Routing*, são declaradas também as funções *pacHoldLink* e *pacReqLink*, que definem a ocupação e a requisição de um *link* por um pacote, respectivamente. A variável *deadlock* indica se existe ou não a ocorrência de *deadlock*.

Na parte do predicado constam as restrições que os pares de roteadores das imagens das funções *pacHoldLink* e *pacReqLink* devem fazer parte do conjunto de canais físicos da NoC, que por sua vez integram as rotas desta NoC. Outra restrição é com relação aos pacotes que estão utilizando o canal físico, os quais não podem usar um canal e requisitar outro *link*, ou seja o domínio da função *pacHoldLink* deve ser diferente do domínio da função *pacReqLink*.

```
  DeadlockConfiguration
NoC
Routing
pacHoldLink: Packet    Router    Router
pacReqLink: Packet     Router    Router
deadlock: yesNo

ran pacHoldLink    links
ran pacReqLink    links
dom pacHoldLink    dom pacReqLink
  p1, p2, p3, p4: Packet; r1, r2, r3, r4: Router
     p1   r1  r2      pacHoldLink
      p2   r2  r3      pacHoldLink
      p3   r3  r4      pacHoldLink
      p4   r4  r1      pacHoldLink
     p1    dom pacReqLink
     p2    dom pacReqLink
     p3    dom pacReqLink
     p4    dom pacReqLink    deadlock = yes
  id: ID    id    dom routeElements    routeElements id    links
```

Figura 3.12 Esquema de configuração de deadlock

3.2.2.1.3 Consumo de Energia (*Power*)

A crescente complexidade dos sistemas embarcados tem tornado o consumo de energia uma restrição de projeto muito importante. Do ponto de vista das estruturas de Rede-em-Chip, o consumo de energia, entre outros aspectos, está estreitamente relacionado ao chaveamento realizado pelos roteadores no momento de escolher a melhor rota e evitar congestionamentos. O chaveamento contribui em uma parte significante do consumo de energia de uma Rede-em-Chip (Ye, 2002).

Considerando a importância do consumo de energia em um projeto de Redes-em-Chip, Ye *et al*. (Ye, 2002) propôs um modelo de energia para roteadores de rede. Este modelo consiste na definição da energia consumida quando um bit de dados é transferido através do roteador. Este modelo considera três fontes de consumo de energia em uma rede de interconexão: (1) as chaves internas do roteador, (2) os canais físicos (*links*) entre roteadores e (3) os *buffers* internos do roteador. O modelo, portanto, consiste da fórmula $E_{bit} = E_{S_{bit}} + E_{L_{bit}} + E_{B_{bit}}$, onde $E_{S_{bit}}$ representa a energia consumida pelo chaveamento (*Switching*) feito pelo roteador, $E_{L_{bit}}$ representa a energia consumida pelo canal físico (*Link*) e $E_{B_{bit}}$ representa a energia consumida pelo *Buffer*.

Considerando uma Rede-em-Chip em que os *buffers* são implementados por registradores, Ye, em (Ye *et al*., 2003), e Hu, em (Hu, 2003), propuseram calcular a energia do bit por meio da fórmula $E_{bit} = E_{S_{bit}} + E_{L_{bit}}$, onde E_{bit} representa a soma dos consumos de energia do chaveamento feito pelo roteador e pelo canal físico entre roteadores.

O esquema *Energy*, exibido na Figura 3.13, apresenta a especificação do consumo de energia para o roteamento de mensagens/pacotes em uma Rede-em-Chip, utilizando a metodologia *CAD*$\mathbb{Z}$ e baseado nos modelos citados acima. O esquema *Energy* é constituído pelas funções que definem os consumos de energia efetuados pelo chaveamento feito pelo roteador (*routerEnergy: Router*), pelo link entre roteadores (*linkEnergy: Router Router*) e pelos *buffers* (*bufferEnergy: Buffer Buffer*). A variável *totalEnergy* representa a soma dos consumos de energia de cada nó e a função *requireLevelValue* define o valor referente ao requisito consumo de energia (power) da aplicação.

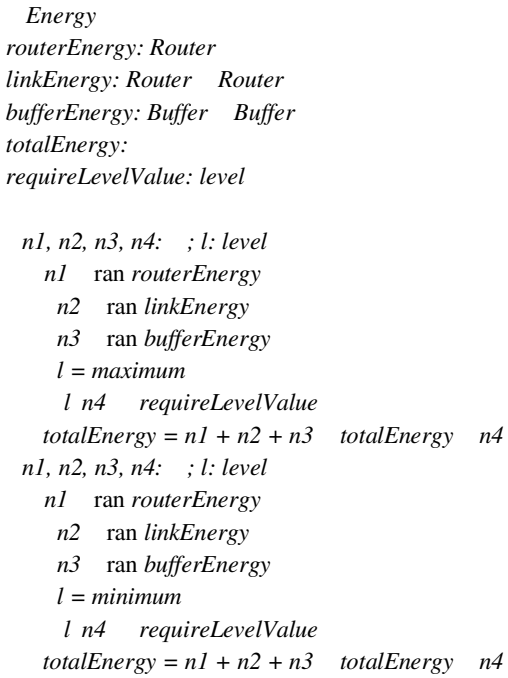

Figura 3.13 Esquema do consumo de energia

A imagem de cada função, representada pelo símbolo dos números naturais, indica o resultado final dos algoritmos que podem ser implementados para obter o resultado de cada um desses consumos. Por exemplo, o consumo de energia dos *buffers* pode ser obtido por meio da fórmula $E_{B_{bit}} = E_{access} + E_{ref}$, onde E_{access} é a energia consumida por cada operação de acesso (leitura ou escrita) ao *buffer*, e E_{ref} é a energia consumida por cada operação de *refresh*, no caso da memória ser do tipo DRAM. Portanto, a imagem da função *bufferEnergy* consiste do resultado da fórmula acima.

3.2.2.2 Mecanismos Básicos de Redes

Os serviços oferecidos pelas redes compreendem o transporte de fluxos de bits de um nó a outro, em formatos específicos, sobre um conjunto de rotas. Esses serviços são diferenciados pela qualidade, e são produzidos por meio de mecanismos, tais como: roteamento, chaveamento e controle de fluxo, conforme exibe a segunda camada da

Figura 3.6, abaixo representada por meio da Figura 3.14. As modelagens desses mecanismos serão apresentadas nas sub-seções seguintes.

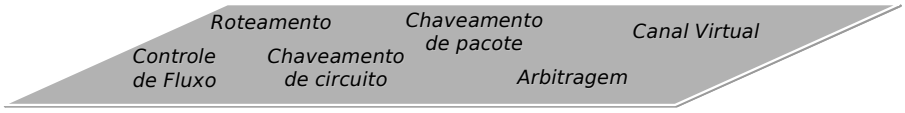

Figura 3.14 Mecanismos Básicos de Redes

3.2.2.2.1 Roteamento (*Routing*)

O roteamento estabelece o caminho seguido por cada pacote ou mensagem. Com base nas definições de roteamento citadas em (Dally, 1987; Duato, 1993), o roteamento é modelado pelo esquema *Routing*, o qual é constituído por duas funções: *routeInfo* e *routeElements*. A rota é um caminho entre dois roteadores, ou seja, de um nó origem para um destino, que pode incluir muitos *links* intermediários. Portanto, a função *routeInfo* identifica cada rota por um identificador exclusivo do tipo ID, e a função *routeElements* define o caminho ou o conjunto de todos os *links* intermediários que constituem uma rota. A Figura 3.15 exibe o esquema de roteamento, cuja parte inferior (predicados) exige que os identificadores dos pares origem/destino seja o mesmo da rota propriamente dita.

Routing
routeInfo: ID *Router* *Router*
routeElements: ID *Router* *Router*

dom *routeInfo* = dom *routeElements*
id: ID; r1, r2: Router
id dom *routeInfo* *r1 r2* ran *routeInfo* *r1 r2* = *routeInfo id*
r1 dom *routeElements id*
r1 ran *routeElements id*
r2 ran *routeElements id*
r2 dom *routeElements id*
r: Router *r* ran *routeElements id* *r* *r2*
r dom *routeElements id*
r: Router *r* dom *routeElements id* *r* *r1*
r ran *routeElements id*

Figura 3.15 Esquema do Roteamento

Conforme mencionado na seção 2.2.2.3, o algoritmo de roteamento pode ser classificado como determinístico e adaptativo. Tipicamente, o roteamento determinístico fornece um único caminho entre cada par de nós, entretanto no roteamento adaptativo o caminho depende das condições dinâmicas da rede. Enquanto o roteamento determinístico define uma rota como saída, o roteamento adaptativo define um conjunto de rotas possível.

Os tipos de roteamento acima citados são modelados por meio dos esquemas *DeterministicRouting* e *AdaptiveRouting*, os quais são exibidos na Figura 3.16 e na Figura 3.17, respectivamente. Estes esquemas refinam o esquema *Routing*. O refinamento ocorre por meio da inclusão de *Routing* na parte declarativa dos mesmos.

Nos esquemas *DeterministicRouting* e *AdaptiveRouting* as partes declarativas incluem as declarações contidas no esquema *Routing* e as variáveis *throughputLevel*, *pacInjRate*, *complexityLevel* e *latencyLevel*, as quais representam o nível de vazão (*throughputLevel*) da rede, a taxa de injeção de pacotes na rede (*throughputLevel*), o nível de complexidade de implementação (*complexityLevel*) e o nível de latência do roteamento. Na parte do predicado, esses esquemas se diferenciam pelos valores assumidos pelas variáveis e pela quantidade de rotas entre os nós origem e destino.

DeterministicRouting
Routing
throughputLevel: level
pacInjRate: level
complexityLevel: level
latencyLevel: level

id: ID; r1, r2: Router
dom *routeInfo* *ID*
id dom *routeInfo*
r1 r2 ran *routeInfo*
r1 r2 = *routeInfo id* # dom *routeInfo* = 1

Figura 3.16 Esquema Roteamento Determinístico

Tipicamente, o nível de complexidade de implementação do roteamento determinístico é baixo (*low*), uma vez que os roteadores utilizam uma lógica simples de roteamento. Quando o nível de injeção de pacotes na rede é alto (*pacInjRate* = *high*), o nível de

vazão é baixo (*throughputLevel = low)*, devido ao congestionamento de pacotes. No roteamento adaptativo, em geral, o nível de complexidade de implementação do roteamento é alto (*complexityLevel = high*), pois os pacotes podem decidir por um caminho sem congestionamento. Por conseguinte, neste tipo de roteamento a latência do roteamento é maior (*latencyLevel = high*) que no roteamento determinístico. Quando o nível de injeção de pacotes na rede é alto (*pacInjRate = high*) o nível de vazão também é alto (*throughputLevel = high).*

AdaptiveRouting
Routing
throughputLevel: level
pacInjRate: level
complexityLevel: level
latencyLevel: level

id: ID; r1, r2: Router
dom *routeInfo* *ID*
id dom *routeInfo*
r1 r2 ran *routeInfo*
r1 r2 = *routeInfo id* # dom *routeInfo* 1

Figura 3.17 Esquema Roteamento Adaptativo

Considerando que o roteamento adaptativo consiste em uma relação que implica em um conjunto de rotas de saída, existem diferentes políticas de seleção para definir a rota que deve ser escolhida para envio do pacote. As políticas de seleção largamente utilizadas são *round robin* (Zeferino, 2004; Kavaldjiev, 2004), *buffer level* (Hu, 2005), *random* (Palesi, 2006), dentre outras (Glass, 1998). Entretanto, conforme menciona Martínez *et al.* (Martínez, 2003) esta seleção pode ser realizada pelo status do roteador e do *link*, que podem assumir os valores de ocupado ou livre, e, portanto, os roteadores e *links* ocupados serão saltados.

A política de seleção definida na metodologia *CAD*$\mathbb{Z}$ considera que (1) uma rota é composta por pares de roteadores e que esses pares possuem um status; (2) por meio das funções *routerStatus* e *linkStatus*, integrantes do esquema *Status* (vide Figura 3.18) é possível identificar o status de cada roteador e *link* e (3) a seleção pode ser aleatória ou baseada em propriedades estáticas ou dinâmicas. A política de seleção é representada

pelo esquema *FreeRouting*, o qual especifica as rotas cujos roteadores e *links* possuem o status livre (em inglês, *free*).

Status
routerStatus: Router status
linkStatus: Router Router status

Figura 3.18 Esquema Status

O esquema *FreeRouting* é constituído pelo esquema *RoutingWithStatus*, que por sua vez representa a conjunção dos esquemas *Routing* e *Status*, e dos invariantes da parte do predicado que ratificam que os roteadores que compõem a rota são os mesmos das funções do esquema *Status*. A Figura 3.19 exibe o esquema *RoutingWithStatus*, enquanto que a Figura 3.20 exibe o esquema *FreeRouting* que representa a política de seleção. Este esquema, além da inclusão do esquema *RoutingWithStatus*, insere também os elementos *freeRouting, freeRouter* e *freeLink.* O elemento *freeRouting* é definido como o conjunto dos identificadores de pares origem/destino, bem como de rotas da Rede-em-Chip que possuem o status *free*; o elemento *freeRouter* representa o conjunto dos roteadores com status *free* e a função *freeLink* define os pares de roteadores cujos status são *free*.

RoutingWithStatus
Routing
Status

id: ID id dom *routeElements*
dom *routeElements id* ran *routeElements id* dom *routerStatus*
id: ID; r1, r2: Router id dom *routeElements r1 r2 routeElements id*
r1 r2 dom *linkStatus*

Figura 3.19 Esquema das rotas com status

FreeRouting
RoutingWithStatus
freeRouting: ID
freeRouter: Router
freeLink: Router Router

freeRouting dom *routeInfo*
freeRouter dom *routerStatus*
id: ID; r1, r2: Router
id dom *routeElements*
r1 r2 *routeElements id*
r1 r2 dom *linkStatus*
linkStatus r1 r2 = free
r1 dom *routerStatus*
r2 dom *routerStatus*
routerStatus r1 = free
routerStatus r2 = free *id* *freeRouting*

Figura 3.20 Esquema representativo da Política de Seleção

3.2.2.2.2 Chaveamento (*Switching*)

Conforme discutido na seção 2.2.2.2, as principais técnicas de chaveamento são o chaveamento de circuito e o chaveamento de pacote, este último classificado em três tipos: *Store-and-Forward* – SAF, *Virtual-Cut-Through* – VCT e *Wormhole*. Dada as características de cada tipo de chaveamento apresentadas naquela seção, a seguir serão exibidas as modelagens dos tipos de chaveamento por meio da metodologia *CAD*$\mathbb{Z}$.

As técnicas de chaveamento envolvem dois aspectos básicos e relevantes que são a memorização e a latência, que corresponde ao tempo gasto na transmissão de uma mensagem ou pacote pela rede. Portanto, esses aspectos constituem os elementos básicos da modelagem de cada tipo de chaveamento.

A modelagem dos tipos de chaveamento é realizada a partir do tipo de caminho (*pathType*), o qual pode ser classificado como dedicado (*dedicated*) ou não dedicado (*noDedicated*), e da unidade básica (*transferUnit*) utilizada pelo chaveamento para efetuar a transferência de dados através da rede, que pode ser uma mensagem, um pacote ou um *flit*. O tipo enumerado *pathType*, exibido na Figura 3.21, define os valores dos elementos *path* declarados em cada tipo de técnica de chaveamento.

pathType :: *dedicated* *noDedicated*

Figura 3.21 Tipo Enumerado relativo ao chaveamento

A modelagem das técnicas de chaveamento de circuito e de pacote são exibidas por meio das Figura 3.22 e Figura 3.23, respectivamente. A modelagem destas técnicas incluem o esquema *RoutingWithStatus* além dos elementos *transferUnit* e *path*, que são comuns em ambas modelagens, porém os elementos *bufferSize* e *latencyLevel*, constantes na modelagem do chaveamento de circuito, só serão considerados nas modelagens das técnicas SAF, VCT e *Wormhole*, as quais resultam do refinamento da técnica de chaveamento de pacote.

CircuitSwitching
RoutingWithStatus
transferUnit: Message
path: pathType
bufferSize: Size
latencyLevel: level

bufferSize = none
id: ID; r1, r2: Router
id dom *routeElements*
r1 r2 *routeElements id*
r1 r2 dom *linkStatus*
linkStatus r1 r2 = busy *path = dedicated*

Figura 3.22 Esquema Chaveamento de Circuito

PacketSwitching
RoutingWithStatus
transferUnit: Packet
path: pathType

id: ID; r1, r2: Router
id dom *routeElements*
r1 r2 *routeElements id*
r1 r2 dom *linkStatus*
linkStatus r1 r2 = busy *linkStatus r1 r2 = free*
path = noDedicated

Figura 3.23 Esquema de Chaveamento de Pacote

A parte do predicado do esquema *CircuitSwitching* define o tipo mensagem como unidade de transferência e restringe os valores dos elementos *path* e *bufferSize* a *dedicated* e *none*, respectivamente. Esses valores são assumidos devido às características do tipo de chaveamento, o qual constrói um caminho dedicado (*path = dedicated*), que não necessita de *buffers* (*bufferSize = none*) para transmitir uma mensagem (*transferUnit = Message*), uma vez que é estabelecido um circuito e a mensagem é transferida sem contenção. Outra restrição está relacionada ao status dos links que compõem a rota que assumem o valor ocupado (*busy*).

O elemento *latency* não está restrito a um valor, devido à possibilidade de variação da latência, uma vez que a latência no chaveamento de circuito envolve o tempo de estabelecimento do circuito mais o tempo da transferência da mensagem. Se na fase de estabelecimento do circuito existir algum *link* desejado com o status ocupado (*busy*), então haverá uma espera pela liberação desse canal, acarretando em uma latência *medium* ou *high*. Por outro lado, a latência será *low* (*latencyLevel = low*), pois a fase de transferência é realizada sem contenção contribuindo para uma latência baixa.

As técnicas de chaveamento de pacotes SAF, VCT e *Wormhole*, como mencionado, são modeladas a partir do refinamento do esquema *PacketSwitching*. No chaveamento SAF cada pacote aloca os recursos necessários para avançar de nó para nó. Isso implica em uma sobrecarga adicional à comunicação, pois cada pacote deve conter um cabeçalho de endereçamento e os roteadores gastam um tempo para efetuar o roteamento individual dos mesmos. Considerando que neste tipo de chaveamento um pacote só pode ser enviado após ter sido completamente recebido, possivelmente haverá um aumento na latência de comunicação devido ao tamanho do pacote, bem como, um aumento no custo da rede devido à necessidade de espaço para armazenamento.

Considerando os aspectos acima citados, a modelagem do chaveamento SAF, representado na Figura 3.24, define na parte do predicado do esquema os valores *large* e *high* para as variáveis *bufferSize* e *latencyLevel*, respectivamente.

StoreAndForward
PacketSwitching
bufferSize: Size
latencyLevel: level

bufferSize = large
latencyLevel = high

Figura 3.24 Esquema do Chaveamento *Store-and-Forward*

No chaveamento VCT o pacote só será armazenado em um nó intermediário se o próximo canal requisitado estiver ocupado. Portanto, como existe a possibilidade de armazenamento do pacote, então o chaveamento VCT adota a mesma definição do chaveamento SAF para o valor da variável *bufferSize* (*bufferSize = large*). Entretanto, como o objetivo deste tipo de chaveamento é a redução da latência, o valor do nível de latência da técnica VCT é considerado inferior ao nível da técnica SAF, ou seja, a variável *latencyLevel* assume o valor *medium* (*latencyLevel = medium*). A Figura 3.25 exibe a modelagem do chaveamento VCT.

VirtualCutThrough
PacketSwitching
bufferSize: Size
latencyLevel: level

bufferSize = large
latencyLevel = medium

Figura 3.25 Esquema do Chaveamento *Virtual Cut-Through*

A técnica de chaveamento *Wormhole*, indicada em diversos trabalhos como a técnica mais adequada para utilização em Redes-em-Chip com chaveamento de pacote (Dally, 2001; Kumar, 2002; Benini, 2004), é definida pelas mesmas funções utilizadas nos chaveamentos SAF e VCT, porém acrescida da restrição do tamanho do *buffer* que é menor em relação às outras técnicas, uma vez que a capacidade do *buffer* está relacionada ao tamanho do *flit*. Considerando que os requisitos de *buffer*, bem como a latência, são menores no chaveamento *Wormhole*, os valores atribuídos às variáveis *bufferSize* e *latencyLevel* foram *small* e *low*, respectivamente. A Figura 3.26 exibe a modelagem do chaveamento *Wormhole*.

Wormhole
PacketSwitching2
bufferSize: Size
latencyLevel: level

bufferSize = small
latencyLevel = low

Figura 3.26 Esquema do Chaveamento *Wormhole*

3.2.2.2.3 Arbitragem (*Arbitration*)

O mecanismo de arbitragem conforme mencionado na seção 2.2.2.4 busca resolver os problemas de colisão de pacotes. A modelagem do mecanismo de arbitragem é realizada de acordo com as definições constantes em (Hwang, 1993), que estabelece quatro estratégias para resolver a colisão entre dois pacotes: Armazenamento Temporário, Bloqueio, Descarte e Desvio. Apesar das Redes-em-Chip só adotarem alguns tipos de estratégia, todas as políticas definidas em (Hwang, 1993) são modeladas a seguir. Para a modelagem das referidas políticas considerou-se que os canais de entrada e saída pertencem ao mesmo nó.

A primeira política refere-se ao armazenamento temporário, a qual foi proposta de acordo com a técnica de chaveamento VCT e consiste em armazenar um dos pacotes envolvidos na colisão em um *buffer* temporário, enquanto o *link* de saída está ocupado com o outro pacote. Quando o *link* tornar-se disponível o pacote armazenado no *buffer* será transmitido. Este *buffer* deve ser implementado em uma memória local a fim de evitar possíveis ciclos. A política de armazenamento temporário está ilustrada na Figura 3.27.

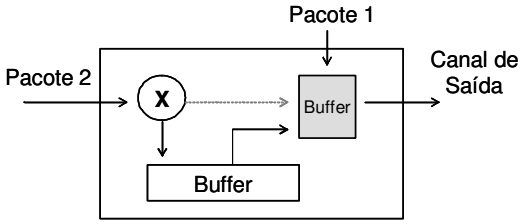

Figura 3.27 Política de Armazenamento Temporário – *Buffering* (Hwang, 1993)

A modelagem da política de armazenamento temporário (*buffering*) é visualizada por meio das Figura 3.28 e Figura 3.29 que exibem as partes da declaração e do predicado do esquema *Buffering*, respectivamente. A parte declarativa consiste do esquema *VirtualCutThrough* e dos elementos *inLink, outLink* e *bufferTemp*. A função *inLink* define um *link* (*Router x Router*) com um ou vários pacotes de entrada (P *Packet*), enquanto que a função *outLink* define um *link* (*Router x Router*) com um pacote de saída (*Packet*). O elemento *bufferTemp* define o *buffer* temporário que é do tipo pacote (*bufferTemp = packet*).

Buffering
VirtualCutThrough
inLink: Packet Router Router
outLink: Packet Router Router
bufferTemp: Packet

Figura 3.28 Parte declarativa do esquema *Buffering*

A primeira parte do predicado expressa que o segundo pacote é armazenado temporariamente em um *buffer*, enquanto o primeiro pacote é transmitido, mantendo o *link* de saída ocupado. A segunda parte do predicado expressa a disponibilidade do canal e o envio do segundo pacote.

dom *outLink* dom *inLink*
p1, p2: Packet; r2, r4, r5, r6: Router; s1, s2: status
p1 r2 r5 inLink
p2 r4 r5 inLink
s1 = busy
s2 = free
ran *outLink* = *r5 r6*
r5 r6 s1 linkStatus bufferTemp = p2 dom *outLink* = *p1*
p1, p2, none: Packet; r2, r4, r5, r6: Router; s1, s2: status
p1 r2 r5 inLink
p2 r4 r5 inLink
s1 = busy
s2 = free
ran *outLink* = *r5 r6*
r5 r6 s2 linkStatus bufferTemp = none dom *outLink* = *p2*

Figura 3.29 Parte do predicado do esquema *Buffering*

A política de Bloqueio, ilustrada na Figura 3.30, é utilizada pela técnica de chaveamento *Wormhole* e consiste no bloqueio do segundo pacote, enquanto o primeiro pacote ocupa o *link* de saída durante sua transmissão.

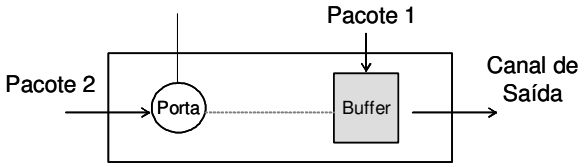

Figura 3.30 Política de Bloqueio (Hwang, 1993)

O esquema *Blocking*, exibido na Figura 3.31, modela a política de bloqueio. Este esquema refina a técnica de chaveamento *Wormhole*, além de declarar também os elementos *inLink* e *outLink*, já comentados no esquema da política de armazenamento temporário. A parte do predicado expressa que para dois pacotes de entrada em *links* diferentes (*p1 r1 r2 inLink, p2 r3 r4 inLink*) um será enviado para o *link* de saída e manterá o *status* do *link* de saída ocupado (...*s1 = busy...* ran *outLink = rx ry rx ry s1 linkStatus* dom *outLink = p1 ...*); entretanto, o outro será mantido bloqueado (... *s2 = blocked ... p2 r3 r4 inLink r3 r4 s2 linkStatus*).

Blocking
Wormhole
inLink: Packet Router Router
outLink: Packet Router Router

p1, p2: Packet; r1, r2, r3, r4, rx, ry: Router; s1, s2: status
p1 r1 r2 inLink
p2 r3 r4 inLink
s1 = busy
s2 = blocked
ran *outLink = rx ry*
rx ry s1 linkStatus
dom *outLink = p1*
p2 r3 r4 inLink
r3 r4 s2 linkStatus

Figura 3.31 Esquema da Política de Bloqueio

Uma outra política que busca evitar a colisão de pacotes é a política de Descarte, que consiste em excluir o pacote que está impedido de passar. A Figura 3.32 ilustra esta política, e a Figura 3.33 apresenta a modelagem da mesma por meio do esquema *Discard*.

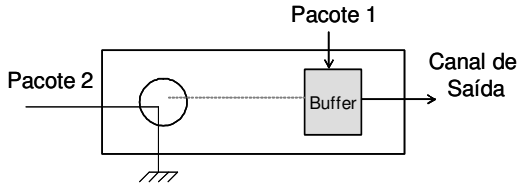

Figura 3.32 Política de Descarte (Hwang, 1993)

A modelagem da política de Descarte consiste do esquema *VirtualCutThrough* e da inclusão dos elementos *inLink* e *outLink* representando os canais de entrada e saída com seus respectivos pacotes. O predicado expressa que para todo dois pacotes de entrada em *links* diferentes (*p1 r1 r2 inLink, p2 r3 r4 inLink*) o primeiro será enviado para o *link* de saída e manterá o *status* do *link* de saída ocupado (...*s1* = *busy*... ran *outLink* = *rx ry rx ry s1 linkStatus* dom *outLink* = *p1* ...); entretanto, o segundo será excluído, ou seja, deixará de pertencer ao *link* de entrada (*p2 r3 r4 inLink*). Essa política raramente é utilizada, devido ao possível desperdício elevado de recursos, bem com o risco da perda do pacote após o descarte.

```
  Discard
VirtualCutThrough
inLink:   Packet    Router   Router
outLink: Packet    Router   Router

 p1, p2: Packet; r2, r4, r5, r6: Router; s1: status
     p1   r2  r5    inLink
       p2   r4  r5    inLink
     s1 = busy
     ran outLink =   r5  r6
       r5  r6  s1   linkStatus
   dom outLink =  p1     p2   r4  r5    inLink
```

Figura 3.33 Esquema da Política de Descarte

A política de desvio ilustrada na Figura 3.34, e modelada pelo esquema ***Detour***, oferece maior flexibilidade no roteamento do pacote, uma vez que o pacote é desviado para um outro link. Entretanto, essa política pode utilizar mais canais que o necessário para alcançar o destino. Ademais, corre o risco de entrar em um ciclo *livelock* que desperdiça recursos da rede.

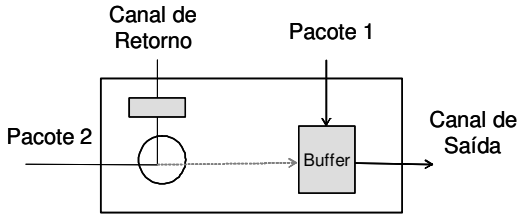

Figura 3.34 Política de Desvio (Hwang, 1993)

O esquema *Detour*, exibido na Figura 3.35, consiste do refinamento do esquema *VirtualCutThrough* e inclui os elementos *inLink* e *outLink*. As sentenças inseridas no predicado do esquema definem que para todo dois pacotes em *links* diferentes (*p1 r1 r2 inLink, p2 r3 r4 inLink*), o primeiro pacote assumirá o *link* de saída (... *s1 = busy* ...ran *outLink = rx ry rx ry s1 linkStatus,* dom *outLink = p1*) e o segundo pacote será desviado (... *s2 = detour... p2 r3 r4 inLink, r3 r4 s2 linkStatus*).

Detour
VirtualCutThrough
inLink: Packet Router Router
outLink: Packet Router Router

p1, p2: Packet; r2, r4, r5, r6: Router; s1, s2: status
p1 r2 r5 inLink
p2 r4 r5 inLink
s1 = busy
s2 = detour
ran *outLink = r5 r6*
r5 r6 s1 linkStatus
dom *outLink = p1*
p2 r4 r5 inLink
r4 r5 s2 linkStatus

Figura 3.35 Esquema da Política de Desvio

As estratégias modeladas nesta seção são implementadas em redes de interconexão de computadores paralelos. Em Redes-em-Chip, normalmente, uma transmissão só é realizada quando o receptor tem espaço suficiente em seu *buffer*, não existindo, portanto, a utilização das estratégias de descarte e desvio de pacotes.

3.2.2.2.4 Controle de Fluxo (*Flow Control*)

O mecanismo de controle de fluxo estabelece um diálogo entre roteadores adjacentes, liberando e interrompendo o avanço da informação. Assim como o chaveamento está firmemente associado ao roteamento, o controle de fluxo está rigorosamente associado aos algoritmos de gerenciamento de *buffers* que definem como os *buffers* serão requisitados e liberados, bem como, definem como os pacotes bloqueados serão tratados.

A modelagem do controle de fluxo é realizada com base no protocolo *handshake* que, assim como o controle baseado em crédito, produz uma Rede-em-Chip que não descarta nem desvia pacotes. Esta modelagem exibida na Figura 3.36 considera os *buffers* do roteador do tipo flit, uma vez que inclui o esquema Wormhole na parte declarativa da especificação. Além da inclusão do esquema Wormhole, é inserido na parte das declarações a função *controlLine* que representa a linha de controle *Request/Acknowledge* comentada na sub-seção 2.2.2.5 e exibida na Figura 2.7.

Considerando que o esquema Wormhole incluído na modelagem consiste no refinamento dos esquemas de roteamento e chaveamento, é necessário que a descrição mais detalhada do mecanismo de controle de fluxo seja consistente com as demais especificações anteriores.

Handshake
Wormhole
controlLine: Router Router level

dom *controlLine* ran *routeElements*
r1, r2: Router; s1, s2, s3: status; l1: level
r1 s1 routerStatus
r2 s2 routerStatus
r1 r2 dom *linkStatus*
s1 = busy
s2 = free
s3 = free
l1 = low r1 r2 l1 controlLine r1 r2 s3 linkStatus
r1, r2: Router; s1, s2, s3: status; l1: level
r1 s1 routerStatus
r2 s2 routerStatus
r1 r2 dom *linkStatus*
s1 = busy
s2 = free
s3 = busy
l1 = high r1 r2 l1 controlLine r1 r2 s3 linkStatus
r1, r2: Router; s1, s2, s3: status; l1: level
r1 s1 routerStatus
r2 s2 routerStatus
r1 r2 dom *linkStatus*
s1 = free
s2 = busy
s3 = busy
l1 = high r1 r2 l1 controlLine r1 r2 s3 linkStatus

Figura 3.36 Esquema do controle de fluxo baseado no protocolo *handshake*

3.2.2.3 Estruturas de Interconexão

Baseando-se na taxonomia de redes de interconexão apresentada na seçao 2.2.1 e representada, em parte, na camada de Estruturas de Interconexão da Figura 3.6, a infra-estrutura de Redes-em-Chip mais popular é a direta ou regular, devido a sua previsibilidade e facilidade de projeto. Em (Hemani, 2000; Liang, 2000; Dally, 2001; Kumar, 2002) são exploradas várias possibilidades de uso e vantagens das Redes-em-Chip regulares. Dentre as várias topologias regulares, a topologia em malha (em inglês, *mesh*) bi-dimensional é uma das mais utilizadas. Esta topologia vem sendo largamente adotada em diferentes projetos (Liang, 2000; Kumar, 2002; Ye, 2004; Shang, 2004; Rego, 2004), e por razões de simplicidade e clareza, a topologia *mesh* será adotada

também nesta tese como a topologia para as descrições, ilustrações e estudo de caso. Entretanto, enfatiza-se que o método proposto pode ser aplicado a qualquer topologia.

A topologia está relacionada ao posicionamento dos nós da rede no espaço. A topologia de uma Rede-em-Chip é representada por um grafo dirigido G(N, C), onde cada vértice n_i pertence ao conjunto de nós N e representa um nó da topologia, a aresta dirigida (n_i, n_j) denotada como c(n_i, n_j) = (n_i, n_j) pertence ao conjunto de canais C e representa uma comunicação direta entre os vértices n_i e n_j. O peso da aresta l(n_i, n_j) denotado por *Bandwidth* (n_i, n_j), representa a largura de banda disponível para a aresta c(n_i, n_j).

A topologia *mesh* de dimensão *n*, por exemplo, é definida formalmente como consistindo de $k_{n-1} x\, k_{n-2} x \ldots x\, k_1 x\, k_0$ nós, onde $k_i \geq 2$ é o número de nós ao longo da dimensão *i*. Um nó X é representado por *n* coordenadas, $(x_{n-1}, x_{n-2}, \ldots, x_0)$, $0 \leq x_i \leq k_i - 1$, $0 \leq i \leq n - 1$. Se X e Y são vizinhos, então o canal da dimensão *i* do nó X está na direção positiva em relação ao nó Y quando $x_i = y_i - 1$, ou na direção negativa quando $x_i = y_i + 1$.

Na metodologia $CAD\mathbb{Z}$ cada nó é representado por um roteador denominado *Router* e cada canal ou *link* é representado por um par de roteadores (*Router, Router*). O esquema *NoC*, representado pela Figura 3.37, define a estrutura de uma Rede-em-Chip, a qual pode ser refinada para uma topologia específica, por exemplo, a *Mesh.* O esquema ***NoC*** contém na parte declarativa o elemento ***routers*** que define o conjunto de roteadores da *NoC* (*Router*) e a relação ***links*** que define o canal entre roteadores (*Router Router*). A parte do predicado expressa que todos os roteadores conectados por um *link* devem ser declarados como sendo roteadores dessa *NoC*.

```
NoC
routers:  Router
links:  Router  Router
routersSize:

r1, r2: Router  r1 r2  links  r1  routers  r2  routers  r1  r2
```

Figura 3.37 Esquema de uma topologia de Rede

O refinamento da ***NoC*** para uma topologia específica, conforme citado no parágrafo anterior, é realizado por meio do esquema ***NoCMesh2D***, o qual insere na parte declarativa variáveis e funções necessárias para descrição de uma topologia *Mesh* bi-dimensional (2D), conforme apresentado na Figura 3.38

NoCMesh2D
NoC
dimx: $_1$
dimy: $_1$
place:
routerplace: Router

dimx 1
dimy 1
routersSize 4
place = 0 .. *dimx* - 1 0 .. *dimy* - 1
dom *routerplace* = *routers*
ran *routerplace* = *place*
x, y: *x* 0 *x y* *place*
routerplace ~ *x y* *routerplace* ~ *x* - 1 *y* *links*
x, y: *x* *dimx* - 1 *x y* *place*
routerplace ~ *x y* *routerplace* ~ *x* + 1 *y* *links*
x, y: *y* 0 *x y* *place*
routerplace ~ *x y* *routerplace* ~ *x* *y* - 1 *links*
x, y: *y* *dimy* - 1 *x y* *place*
routerplace ~ *x y* *routerplace* ~ *x* *y* + 1 *links*

Figura 3.38 Esquema *NoCMesh2D*

As restrições constantes na parte do predicado, as quais devem ser sempre satisfeitas, sentenciam que os valores das dimensões ***x*** e ***y*** devem ser maior que um (1), que cada local da topologia é definido por uma coordenada, e cada roteador R(i,j) está posicionado em uma dessas coordenadas, bem como, está conectado a outros roteadores R(i+/-1,j) e R(i, j+/-1), se eles existirem.

O esquema ***NoCTree*** expressa outro refinamento da especificação ***NoC***, entretanto é realizado a partir da definição das características de uma topologia de árvore, conforme exibe a Figura 3.39.

$$
\begin{array}{l}
\quad NoCTree \\
NoC \\
level: \ _1 \\
levelnodes: \ _1 \\
place: \ _1 \ \ _1 \\
downlink: \ _1 \ \ _1 \ \ _1 \ \ _1 \\
routerplace: Router \ \ _1 \ \ _1 \\
\\
routersSize \quad 3 \\
place \quad 1 \mathinner{..} level \quad 1 \mathinner{..} levelnodes \\
\quad i, j: \ _1 \\
\qquad i \ j \quad place \\
\qquad\quad i = 1 \quad j = 1 \\
\qquad\qquad l: \ _1 \quad i - 1 \ l \quad place \quad j = 2 * l \quad j = 2 * l - 1 \\
\quad i1, i2: 1 \mathinner{..} level; \ j1, j2: 1 \mathinner{..} levelnodes \\
\qquad i1 \ j1 \quad place \quad i2 \ j2 \quad place \\
\qquad i2 = i1 + 1 \quad j2 = 2 * j1 \quad j2 = 2 * j1 - 1 \\
\qquad\quad i1 \ j1 \quad i2 \ j2 \quad downlink \\
\mathrm{dom}\ routerplace = routers \\
\mathrm{ran}\ routerplace = place
\end{array}
$$

Figura 3.39 Esquema *NoCTree*

3.3 Considerações Finais

Este capítulo destinou-se à modelagem *CAD*$\mathbb{Z}$ efetuada através de elementos formais, oriundos das perspectivas de propriedade, mecanismos básicos de rede e estrutura de interconexão. Os esquemas formais obtidos e apresentados podem ser conjugados, originando novos esquemas, possibilitando a geração de novas descrições a partir de elementos já existentes, o que torna a metodologia *CAD*$\mathbb{Z}$ um instrumento facilitador para o projetista de sistema.

4. Aplicando a metodologia $CAD\mathbb{Z}$ no Projeto de Rede-em-Chip

Este capítulo tem como finalidade mostrar a dinâmica da aplicação da metodologia $CAD\mathbb{Z}$ apresentada no capítulo 3, cuja ilustração é reapresentada por meio da Figura 4.1. Como já foi abordado, a ênfase da metodologia $CAD\mathbb{Z}$ está na análise qualitativa, que corresponde a terceira fase da mesma.

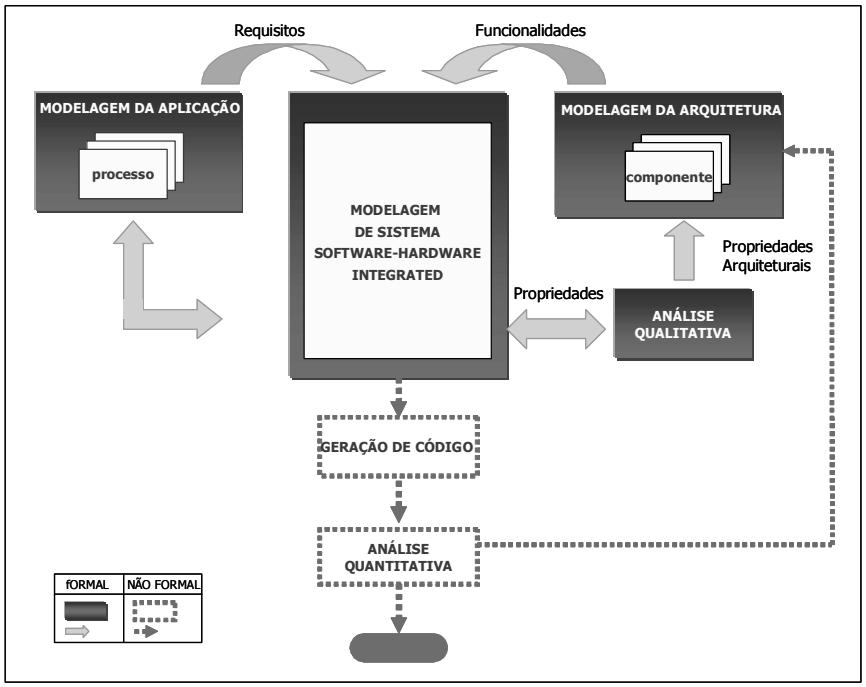

Figura 4.1 Metodologia $CAD\mathbb{Z}$

Na metodologia $CAD\mathbb{Z}$ a análise qualitativa pode ser classificada em genérica e específica. A primeira visa avaliar propriedades em uma arquitetura genérica, ou seja, adequada a qualquer tipo de aplicação. A segunda operacionaliza uma estratégia de otimização, pois considera uma aplicação específica ou um domínio de aplicações. Embora ambas sejam importantes, a segunda representa melhor os benefícios de características diferenciais da metodologia.

Esta tese propõe que a aplicação da metodologia apresentada na Figura 4.1 seja realizada por três equipes: equipe de modelagem dos requisitos, equipe de especificação e equipe de projeto do sistema. A Figura 4.2 exibe as equipes que integram a metodologia, as perspectivas de modelagem (ver Figura 3.6) associadas a cada equipe, bem como, apresenta uma visão geral das atividades realizadas pelas equipes e os resultados das mesmas.

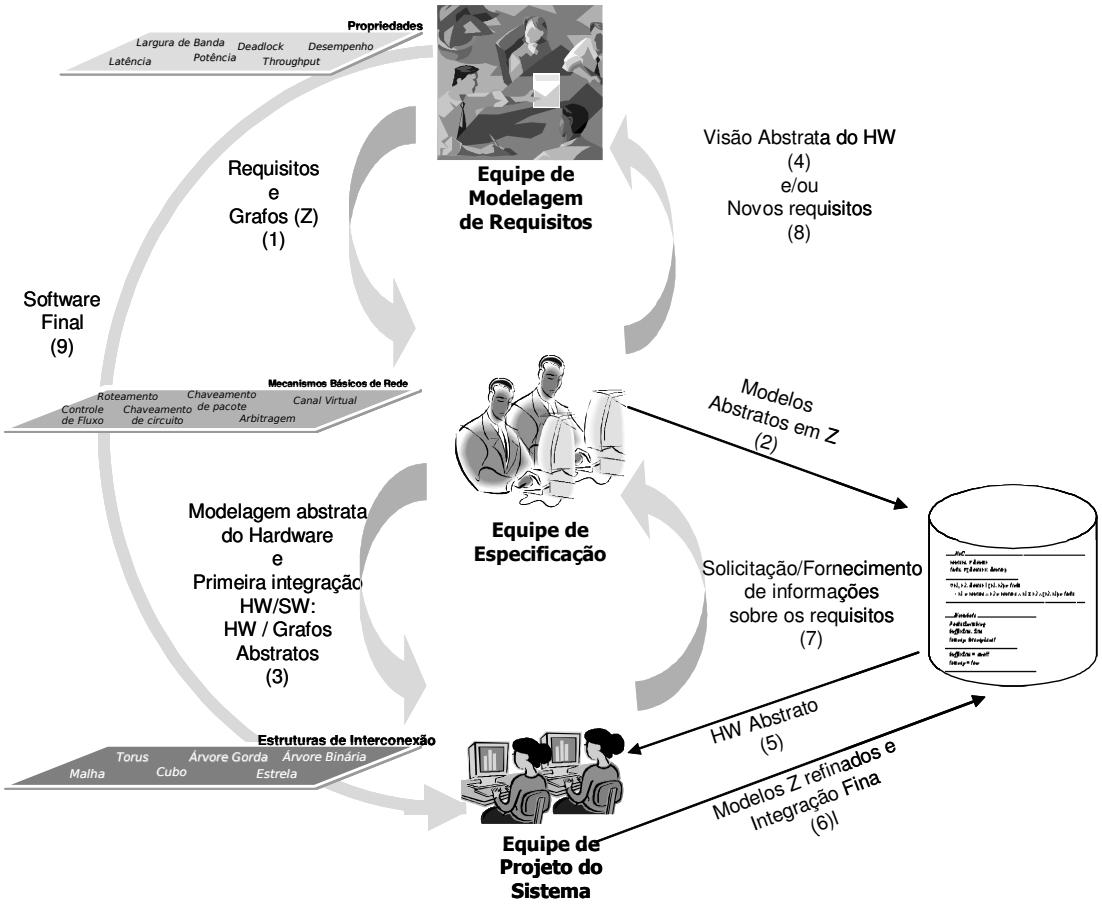

Figura 4.2 Dinâmica da Metodologia *CADℤ*

Tomando por base as fases da metodologia *CADℤ* reapresentadas na Figura 4.1, a seguir serão apresentados os trabalhos desenvolvidos por cada equipe, bem como os resultados obtidos pelas mesmas. Objetivando facilitar a compreensão, é feita sistematicamente uma associação entre as definições presentes no capítulo 2, as fases da

metodologia definidas no capítulo 3 e a aplicação propriamente dita da metodologia $CAD\mathbb{Z}$.

4.1 Modelagem de Requisitos

Na metodologia $CAD\mathbb{Z}$, a equipe de modelagem de requisitos é responsável pelo levantamento dos requisitos da aplicação com base nas propriedades inerentes às estruturas de interconexão. A Figura 4.3 exibe as atividades que são executadas exclusivamente por esta equipe e os resultados que a mesma gera para as demais equipes.

Figura 4.3 Equipe de Modelagem: atividades e resultados

Considerando que a metodologia $CAD\mathbb{Z}$ é uma metodologia com ênfase no projeto de hardware, mais especificamente no desenvolvimento de Redes-em-chip, a fase de modelagem da aplicação foi propositadamente simplificada. Assim sendo, a aplicação é representada por meio de um grafo, onde cada nó corresponde a um processo e as arestas representam os requisitos da referida aplicação. Desta forma, o grafo pode representar mais de uma visão específica dos requisitos da aplicação, ou seja, pode existir um grafo referente ao requisito de consumo de energia, outro referente à latência, e assim por diante.

O objetivo desta estratégia de modelagem é enfatizar aspectos de comunicação, explicitando a relação existente entre o desempenho da aplicação e a estrutura física de comunicação. O desenvolvimento de arquiteturas genéricas para interconexão é largamente justificado pela sua abrangência a vários domínios de aplicação e pela

redução do custo de produção baseado em reuso. Contudo, a especialização da infra-estrutura de comunicação pode ser utilizada como meio efetivo para otimizar o desempenho, uma vez que o mapeamento entre as necessidades de comunicação de uma aplicação específica e os recursos de comunicação existentes, certamente influenciará no desempenho da aplicação. É precisamente esse aspecto que norteia a metodologia $CAD\mathbb{Z}$.

Embora a metodologia $CAD\mathbb{Z}$ tenha características similares à metodologia Co-Design (Lavagno, 1996), inicialmente, é realizado um levantamento dos requisitos da aplicação. Este levantamento é realizado pela equipe de modelagem de requisitos em alto nível de abstração, ou seja, os requisitos podem ser representados por níveis abstratos ou por valores representativos ou indicativos (máximo, mínimo, médio). Por exemplo, o requisito de uma aplicação pode ser a latência máxima permitida para comunicação entre processos.

Os requisitos são modelados em Z como um conjunto enumerado (ver Figura 3.7), e inseridos na modelagem da aplicação por meio da relação *requireLevel*. A Figura 4.4 exibe o esquema da modelagem da aplicação em Z. Nesse esquema são descritos: o conjunto de processos (nós) do grafo (*processes*), a comunicação entre os processos (*commProcess*), o rótulo da aresta (*label*), o requisito da aplicação seguido do seu nível de exigência (*requireLevel*) e a quantidade de processos da aplicação (*processesSize*).

Application
processes: Process
commProcess: Process Process
label: Process Process
requireLevel: requirements level
processesSize:

 # processes = processesSize
dom *label = commProcess*
p1, p2: Process p1 p2 commProcess
p1 processes p2 processes p1 p2

Figura 4.4 Esquema da modelagem da aplicação

O resultado do levantamento dos requisitos e da modelagem da aplicação consiste na obtenção de um grafo que expressa as restrições referentes aos requisitos levantados. A

Figura 4.5 exibe o grafo de uma dada aplicação, cujas arestas representam o requisito da aplicação referente ao atraso máximo de comunicação entre os processos (nós) que se comunicam. Este atraso não deve levar em consideração qualquer característica do hardware, uma vez que a equipe de modelagem de requisitos da aplicação não toma nenhuma decisão a este respeito e, para realizar seu trabalho, não necessita fazer qualquer suposição a cerca do hardware. Ao contrário, esta equipe mapeia os requisitos intrínsecos da aplicação para que as outras equipes definam os recursos de hardware capazes de atendê-los.

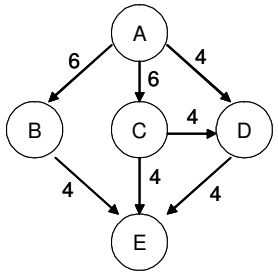

Figura 4.5 Grafo da aplicação referente ao atraso máximo de comunicação (baseado em Hwang, 1993)

Os grafos gerados pela equipe de modelagem de requisitos são repassados para a equipe de especificação. Entretanto, a equipe de modelagem, ao longo da análise qualitativa do projeto de Redes-em-Chip, poderá ser consultada novamente para efetuar o levantamento de outros requisitos que não foram abordados inicialmente. Por exemplo, a equipe de modelagem de requisitos pode realizar uma modelagem primária da aplicação considerando apenas os requisitos temporais de comunicação entre os processos. Entretanto, as equipes seguintes podem identificar duas ou mais soluções de hardware que atendam aos requisitos temporais, apresentando, contudo, características distintas quanto ao custo, consumo de potência, área e desempenho final. A equipe de modelagem de requisitos da aplicação pode ser chamada a se posicionar com relação a estes requisitos.

A equipe de modelagem também deverá receber da equipe de especificação, visões abstratas do hardware para a partir destas, gerar os componentes de software da aplicação que podem ser repassados diretamente para a equipe de projeto do sistema.

4.2 Especificação dos Elementos

A equipe de especificação é a segunda equipe proposta para a execução da metodologia $CAD\mathbb{Z}$. Esta equipe trabalha com base na segunda camada das perspectiva de modelagem (ver Figura 3.6). As atividades desta equipe, bem como, os resultados gerados pela mesma são exibidos na Figura 4.6.

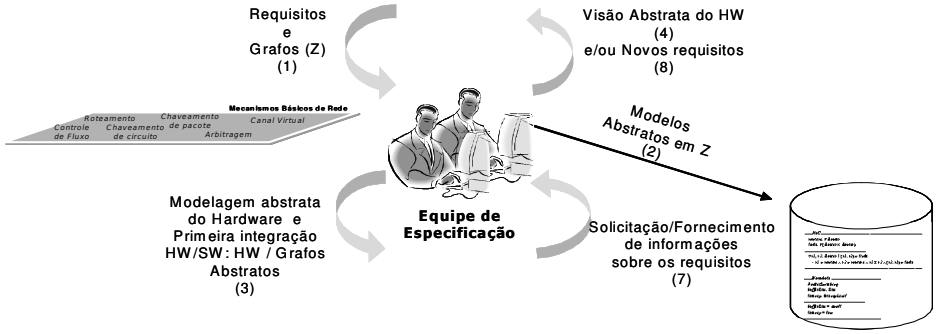

Figura 4.6 Equipe de Especificação: atividades e resultados

A equipe de especificação é responsável pela geração dos modelos abstratos referentes à camada dos mecanismos básicos de rede. Ainda considerando níveis altos de abstração, porém realizando um primeiro mapeamento entre os requisitos da aplicação e os níveis de serviço oferecidos pelo hardware, esta equipe realiza o levantamento e a especificação dos elementos que caracterizam uma Rede-em-Chip, tais como controle de fluxo, roteamento, arbitragem, chaveamento, memorização. Os modelos abstratos gerados ficam armazenados em uma biblioteca de especificação que será utilizada pela equipe de projeto de sistema.

O objetivo da tarefa de geração de modelos abstratos é identificar os elementos necessários para a definição da Rede-em-Chip adequada para atender os requisitos da aplicação definidos anteriormente. Na realidade, é indiferente para esta equipe se a estrutura de interconexão será barramento ou Rede-em-Chip, entretanto, como esta tese é baseada em Redes-em-Chip, os modelos abstratos aqui utilizados são gerados com base nesse tipo de estrutura.

Outra atividade de responsabilidade da equipe de especificação é a geração da primeira integração entre software/hardware, ou seja, a integração entre os requisitos da

aplicação e as descrições abstratas do hardware. A equipe de especificação repassa a primeira integração HW/SW para a equipe de projeto do sistema e para a equipe de modelagem de requisitos. A partir desse momento é estabelecido o início do desenvolvimento paralelo HW/SW. Eventualmente, a equipe de especificação poderá fazer novas solicitações sobre novos requisitos da aplicação.

Os modelos abstratos dos elementos da infra-estrutura de comunicação são gerados com base nos requisitos da aplicação, os quais são recebidos da equipe de modelagem de requisitos. O mecanismo de roteamento, por exemplo, é modelado a partir da especificação das possíveis rotas existentes e do critério de adaptatividade, ou seja, se o roteamento é determinístico ou adaptativo (ver seção 2.2.2.3). A Figura 4.7 exibe a parte declarativa da modelagem do roteamento determinístico. Entretanto, a modelagem do roteamento pode incluir outros critérios, tais como número de destinos (*unicast* ou *multicast*), modos de implementação (baseado em tabela ou em máquina de estados), etc. O fator determinante na especificação dos mecanismos básicos de rede serão os requisitos da aplicação.

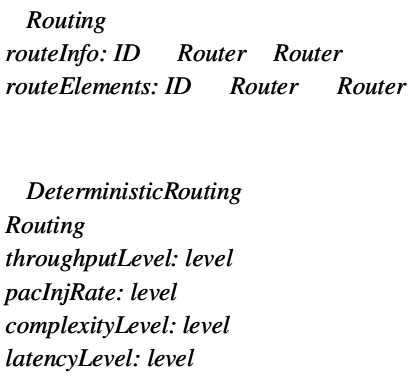

Figura 4.7 Modelagem parcial do mecanismo roteamento

O esquema *DeterministicRouting* da Figura 4.7 inclui o esquema *Routing*, que especifica as possíveis rotas, e apresenta quatro elementos que permitem a caracterização em alto nível do tipo de roteamento segundo a adaptatividade. O roteamento determinístico apresenta uma lógica simplificada (*complexityLevel* = *low*) em termos de projeto de roteadores, como conseqüência, esse tipo de roteamento permite uma latência baixa (*latencyLevel* = *low*), na ausência de congestionamento.

Entretanto, quando a taxa de injeção de pacotes aumenta (*pacInjLevel* = *high*) a vazão é degradada (*throughput* = *low*), uma vez que os roteadores não podem resolver dinamicamente o congestionamento da rede.

Por outro lado, o roteamento adaptativo que também é descrito com os mesmos elementos do esquema *DeterministicRouting*, apresenta uma configuração distinta em relação aos níveis de vazão, taxa de injeção de pacotes, complexidade lógica e latência. O roteamento adaptativo tem a capacidade de evitar congestionamento (*pacInjRate* = *high*), uma vez que os pacotes podem evitar links congestionados pelo uso de caminhos alternativos e desta forma aumentando o nível de vazão (*throughput* = *high*). Contudo, devido a lógica extra utilizada para decidir o caminho a ser utilizado (*complexityLevel* = *high*), o roteamento adaptativo apresenta uma latência de roteamento maior (*latencyLevel* = high) que o roteamento determinístico.

Considerando a aplicação que originou o grafo da Figura 4.5, e com base nos elementos que caracterizam o roteamento, a equipe de especificação define o roteamento mais adequado ao atendimento do requisito de latência máxima.

Com relação ao chaveamento, na seção 3.2.2.2.2 foram modelados os principais tipos de chaveamento, que correspondem aos modelos abstratos de chaveamento que podem ser gerados pela equipe de especficação, a partir de informações disponibiliadas pela equipe de modelagem de requisitos.

Considerando o grafo da aplicação exibido na seção 4.1, e os modelos abstratos de chaveamento, cujas declarações são exibidas na Figura 4.8, a equipe de especificação pode concluir que é possível atender ao requisito de latência máxima, que a aplicação exige, por meio do chaveamento de circuito que reserva recursos estabelecendo um caminho dedicado (*path* = *dedicated*). Entretanto, o chaveamento *wormhole* também pode atender o requisito de latência máxima, mas com maior consumo de energia, uma vez que esse tipo de roteamento utiliza *buffers* (*bufferSize* = *small*), que não são necessários no chaveamento de circuito (*bufferSize* = *none*).

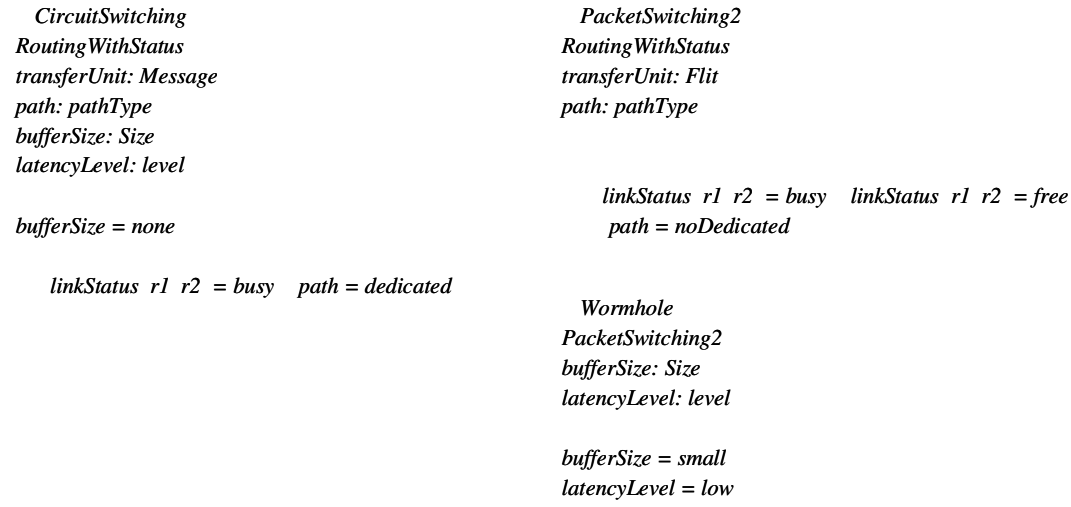

Figura 4.8 Esquemas parciais de chaveamentos de circuito e wormhole

Situações como esta, descrita acima, são solucionadas pela equipe de especificação a partir de solicitações à equipe de modelagem sobre novos requisitos. Na situação acima, a equipe de especificação solicita à equipe de modelagem informações sobre o requisito consumo de energia. A equipe de modelagem fornece novos grafos com base no novo requisito solicitado.

A arbitragem é outro mecanismo básico de rede modelado pela equipe de especificação. As estratégias utilizadas pelo mecanismo de arbitragem para solucionar conflitos de utilização de canais por dois ou mais pacotes constituem informações relevantes que são capturadas na fase de especificação. Na seção 3.2.2.2.3 foram exibidos alguns modelos abstratos das estratégias de arbitragem. As estratégias mais adequadas para Redes-em-Chip são as estratégias de armazenamento temporário e de bloqueio, uma vez que as Redes-em-Chip utilizam controles que na maioria dos casos, não admite descarte ou desvio de pacotes.

Considerando que a técnica de chaveamento wormhole tenha sido a técnica mais adequada para atender os requisitos da aplicação do grafo da Figura 4.5, a estratégia de arbitragem denominada bloqueio é a estratégia indicada para resolver o problema de colisão de pacotes, uma vez que esta estratégia foi desenvolvida para atender *buffers* do tipo flit, os mesmos utilizados pela técnica wormhole. A Figura 4.9 exibe o modelo

abstrato da técnica wormhole, o qual é incluído na especificação da estratégia de bloqueio.

```
   Wormhole
PacketSwitching2
bufferSize: Size
latencyLevel: level

bufferSize = small
latencyLevel = low


   Blocking
Wormhole
inLink:    Packet     Router    Router
outLink: Packet     Router    Router
```

Figura 4.9 Modelos (parciais) utilizados pelo mecanismo de arbitragem

Com relação ao controle de fluxo, a equipe de especificação gera modelos abstratos das técnicas de implementação desse mecanismo, as quais podem variar entre controle de fluxo baseado em crédito, em canais virtuais, em *slack buffer* ou ainda por meio do protocolo "aperto de mão" (*handshake*). Os modelos abstratos de controle de fluxo visam controlar a transmissão de pacotes entre nós adjacentes, de modo a garantir que o nó destino só receberá pacotes se houver espaço disponível em *buffer*, pois as Redes-em-Chip ainda não toleram descartes nem desvios de pacotes.

O modelo abstrato do protocolo *handshake* descrito na seção 3.2.2.2.4, cuja parte das declarações está apresentado na Figura 4.10 pode ser utilizado para controlar a comunicação entre os processos da aplicação representada no grafo da Figura 4.5, uma vez que este protocolo é baseado na técnica de chaveamento *wormhole*, a qual foi supostamente selecionada para atender o requisito de latência da aplicação.

```
   Handshake
Wormhole
controlLine: Router    Router      level
```

Figura 4.10 Modelo do protocolo *handshake*

Além da geração dos modelos abstratos referentes aos mecanismos básicos de rede, a equipe de especificação gera também os modelos abstratos das topologias de Redes-em-Chip. Apesar das Redes-em-Chip adotarem princípios e características das redes de interconexão utilizadas em computadores paralelos (ver seção 2.2), existem restrições e requisitos diferentes para implementação desses dois tipos de redes.

As topologias utilizadas nas redes de interconexão para computadores paralelos, por exemplo, utilizam estruturas complexas como as tri-dimensionais (3-D). As Redes-em-Chip, no entanto, são construídas utilizando estruturas bi-dimensionais (2-D) e de árvore, as quais são mais adequadas às tecnologias atuais de fabricação (Zeferino, 2003a).

Na seção 3.2.2.3 foram gerados alguns modelos abstratos de topologias de Redes-em-Chip, os quais foram contruídos a partir da estrutura básica de uma Rede-em-Chip que são os roteadores e *links*. A Figura 4.11 exibe a parte das declarações dos modelos das estruturas de interconexão geradas.

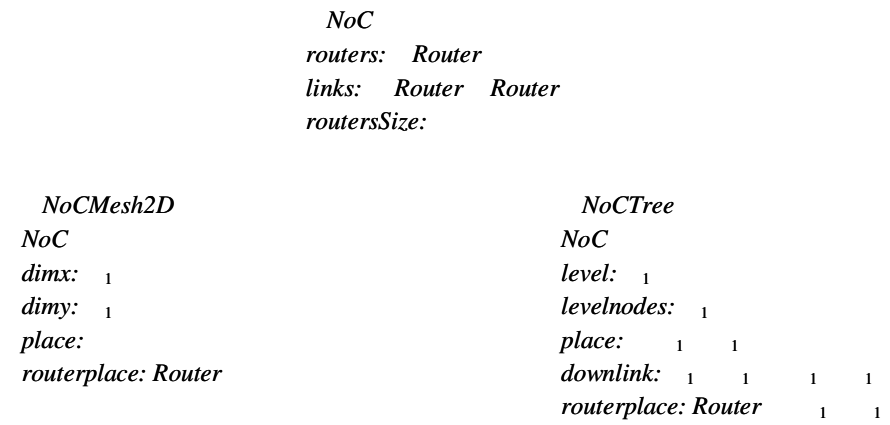

Figura 4.11 Modelos parciais de estruturas de interconexão

A partir da geração dos modelos abstratos referentes aos mecanismos básicos de rede e às estruturas de Redes-em-Chip, bem como, do grafo de processos da aplicação, a equipe de especificação gera a primeira integração de Software/Hardware que corresponde a segunda fase da metodologia *CAD*$\mathbb{Z}$ reapresentada na Figura 4.1. A integração consiste na efetiva compatibilização entre a modelagem da aplicação e a

modelagem da arquitetura. A Figura 4.12 exibe a modelagem em Z da conjunção dos esquemas de Rede-em-Chip (NoC) e da aplicação (*Application*) representando o Sistema Integrado Hardware/Software.

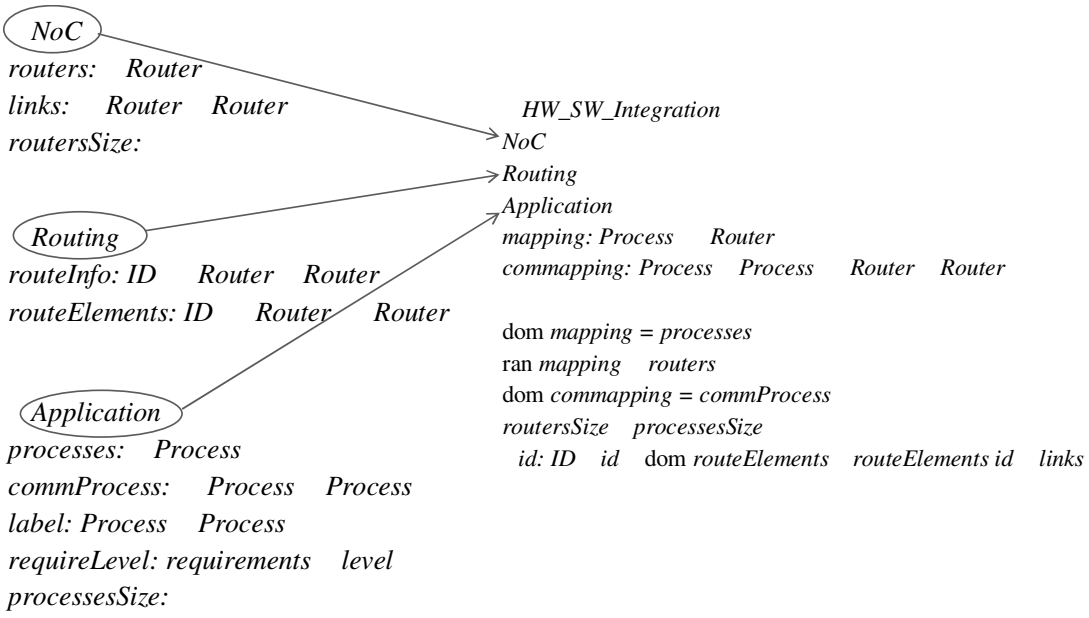

Figura 4.12 Modelo abstrato da integração Hardware/Sofware

O esquema *HW_SW_Modeling* representa a modelagem do sistema integrado de HW/SW, o qual faz uma conjunção dos esquemas *NoC* e *Application*, bem como, insere as funções *mapping* e *commapping*, que indicam, respectivamente, o mapeamento dos processos nos roteadores e dos *labels* nos *links*. Este esquema além de preservar as restrições impostas aos esquemas *NoC* e *Application*, o mesmo insere restrições ao mapeamento dos processos na Rede-em-Chip, exigindo por exemplo que para cada processo deve existir um roteador, ou seja, a quantidade de roteadores deve ser maior ou igual a quantidade de processos.

Sendo assim, as fases de modelagem e integração da metodologia *CAD*$\mathbb{Z}$ se inter-relacionam em uma abordagem tipicamente iterativa, resultando em um sistema integrado de software e hardware (ver Figura 4.1), o qual é rapassado para a equipe de projeto de sistema. A partir do esquema da primeira integração HW/SW a equipe de

projeto do sistema efetuará os refinamentos necessários para análise do atendimento de requisitos da aplicação.

4.3 Projeto do Sistema

A terceira equipe proposta para executar a metodologia *CADℤ* é a equipe de Projeto do Sistema, a qual é responsável pelo refinamento dos modelos abstratos gerados pela equipe de especificação ou obtidos de outros projetos. A Figura 4.13 destaca as atividades executadas e os resultados obtidos por esta equipe.

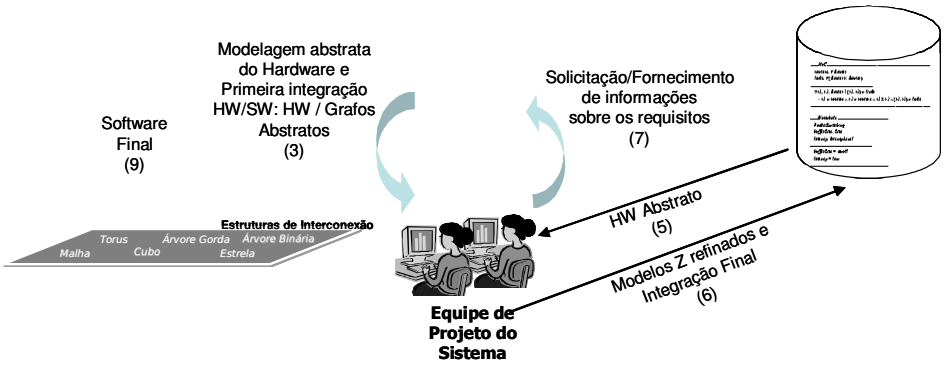

Figura 4.13 Equipe de Projeto do Sistema: atividades e resultados

A equipe de Projeto do Sistema trabalha focada na terceira camada da Figura 3.6 Perspectiva da Modelagem. Esta equipe é responsável pelas atividades de modelagem refinada do hardware e modelagem Hardware/Software (HW/SW) definitiva. A primeira atividade consiste no refinamento de modelos abstratos de hardware, podendo inclusive refinar modelos abstratos obtidos de outros projetos. A segunda atividade é realizada mediante informações sobre o software final, advindas da equipe de modelagem de requisitos, e informações da modelagem HW/SW produzidas pela equipe de especificação, as quais serão utilizadas pela equipe de projeto para o refinamento definitivo da integração HW/SW.

Os resultados dos refinamentos são armazenados na biblioteca de epecificação, podendo ser utilizados em outros projetos. A execução da tarefa de refinamento consiste na

produção de modelos de mais baixo nível de abstração que incluem informações qualitativas, principalmente, e quantitativas que contribuem para a seleção de uma implementação específica que atenda aos requisitos especificados pela equipe de modelagem dos requisitos. Por exemplo, tratando do roteamento a equipe de projeto pode detalhar os custos (atrasos, potência, latência, etc.) envolvidos em cada implementação possível e considerada pela equipe.

Aspectos não relacionados à aplicação propriamente dita, mas considerados de extrema importância para a implementação final, tais como testabilidade e tolerância a falhas, devem ser considerados pela equipe de projeto. Definição e compatibilização de interfaces, principalmente quando componentes especificados para outros projetos são reutilizados, também devem ser tratados.

Retomando como exemplo o grafo da aplicação apresentado na Figura 4.5, a equipe de projeto tem conhecimento que o principal requisito da aplicação é a latência máxima entre a comunicação dos processos, que não deve exceder os valores dos rótulos das arestas do grafo. Ademais, a equipe de projeto, também, já conhece os mecanismos básicos de rede adequados ao atendimento dos requisitos da aplicação. Contudo, devido as diferentes formas de atendimentos desses requisitos, a equipe de projeto poderá solicitar à equipe de especificação novas visões da aplicação referente a novos requisitos, por exemplo, consumo de energia. As novas demandas geradas para a equipe de especificação, caracterizam o movimento cíclico explicitado pela Figura 4.2.

Enquanto a equipe de especificação observa o atendimento do requisito com base em níveis (alto, baixo, máximo, etc), a equipe de projeto analisa o atendimento do requisito do ponto de vista menos abstrato. Por exemplo, o requisito latência das técnicas de chaveamento consiste da soma (1) dos atrasos que um roteador leva para rotear um pacote, (2) da propagação do pacote no canal físico e (3) do tempo de armazenamento em *buffer* (ver seção 3.2.2.1.1). A equipe de projeto, neste exemplo, preocupa-se exatamente com os atrasos individuais de cada elemento, bem como a forma de implementação dos mesmos.

Considerando os modelos abstratos do chaveamento wormhole, da propriedade latência e da latência de uma rota é possível constatar o exemplo citado acima. O modelo

abstrato do chaveamento wormhole, gerado pela equipe de especificação, descreve uma das características dessa técnica, que é proporcionar níveis baixos de latência (*latencyLevel = low*) (ver Figura 4.14 a). A latência, por sua vez, é descrita como a composição dos atrasos do roteador (*routerDelay*), do canal físico (*linkDelay*) e do *buffer* (*bufferDelay*) (ver Figura 4.14 b). Nesta descrição, os atrasos não são representados como níveis, mas como valores que devem estar inseridos em um dada faixa de valores mínimos e máximos que correspondem aos níveis mínimo, máximo, etc. O módulo abstrato *RoutingLatency*, que representa o cálculo da latência de uma rota auxilia na análise qualitativa, para verificar se a soma dos atrasos que compõem a latência se enquadram nas faixas de valores exigidas pela aplicação (ver Figura 4.14 b).

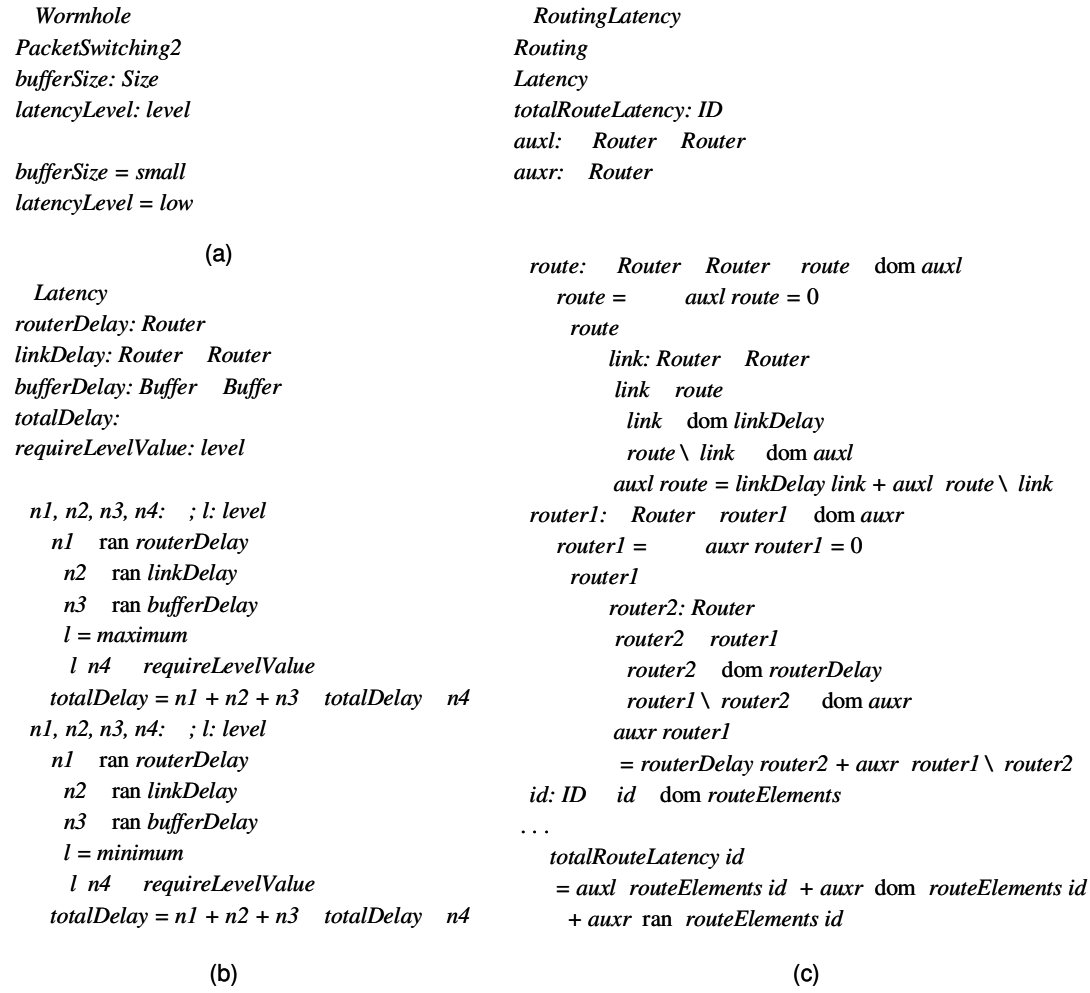

Figura 4.14 Modelos abstratos que lidam com a latência de comunicação

A partir do modelo abstrato da primeira integração HW/SW, a equipe de projeto pode gerar modelos abstratos refinados como o exibido na Figura 4.15, o qual insere restrições sobre a latência das rotas no modelo HW/SW.

HW_SW_RoutingLatency
HW_SW_Integration
RoutingLatency

dom *routerDelay* = ran *mapping*
dom *linkDelay* *links*
id: ID *id* dom *routeElements* *routeElements id* *links*
n1, n2: *n1 = totalDelay* *n2* ran *label* *n1* *n2* *n1* *n2*
r1: requirements; l1, l2: level; n1:
r1 l1 *requireLevel* *l2 n1* *requireLevelValue* *l1 = l2*

Figura 4.15 Esquema HW/SW integrado com referência à latência

Por meio do refinamento da primeira integração HW/SW a equipe de projeto, utilizando a análise qualitativa, define a estrutura adequada para atender os requisitos da aplicação gerando a integração HW/SW refinada. Por exemplo, tomando como base o grafo apresentado na Figura 4.5, a integração HW/SW refinada representa a alocação dos processos nos componentes ligados aos roteadores, conforme Figura 4.16.

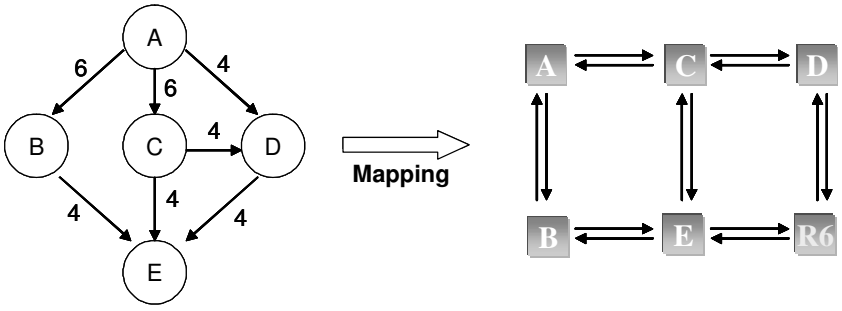

Figura 4.16 Integração Hardware/Software refinada

É importante observar que a obediência às restrições impostas pela aplicação, no que se refere à comunicação, é traduzida pela Rede-em-Chip resultante desse mapeamento. A análise qualitativa também pode ser estendida para outras propriedades.

A natureza qualitativa desta análise diz respeito à flexibilidade na definição de propriedades e a possibilidade de avaliá-las em fase inicial de projeto. Embora análises

quantitativas tradicionais possam complementar a análise qualitativa, a sua aplicação só se dá nas fases finais do projeto.

4.4 Considerações Finais

Este capítulo apresentou a dinâmica de utilização da metodologia $CAD\mathbb{Z}$, demonstrando sua aplicabilidade. Uma metodologia de projeto visa definir claramente os papéis das equipes envolvidas, identificando as fases da metodologia e as equipes que as executam. Também uma metodologia especifica o produto a ser obtido em cada fase. Dessa forma, é correto supor que a adoção de uma metodologia facilita a comunicação e contribui para o aumento da produtividade das equipes.

5. Estudos de Caso

Este capítulo tem como objetivo validar a metodologia $CAD\mathbb{Z}$ a partir de estudos de caso que demonstrem as características e benefícios da metodologia. São apresentados dois estudos de caso. O primeiro está focado na análise formal do impacto de algoritmos de roteamento, que é um dos mais importantes mecanismos de rede. O segundo estudo de caso refere-se a projeto já desenvolvido e documentado na literatura. Embora tal projeto não tenha obviamente utilizado a metodologia $CAD\mathbb{Z}$ durante o seu desenvolvimento, a análise da aplicabilidade e benefícios da utilização da metodologia proposta é explicada sem prejuízo da compreensão.

5.1 Análise Formal de Mecanismos Básicos de Rede

Esta seção exemplifica uma abordagem formal para análise de mecanismos básicos de rede, mais especificamente de algoritmos de roteamento. O objetivo é mostrar que é possível validar propriedades diretamente da especificação formal, através de ferramentas específicas.

Neste estudo de caso, documentado em (Ramos, 2006a; Ramos, 2006b), são avaliados três algoritmos de roteamento. O primeiro refere-se ao menor caminho, o segundo é baseado na rota de menor latência e o terceiro considera o estado dos links.

A ferramenta utilizada, denominada Possum (Hazel, 1997), é classificada como ferramenta para animação de especificação Z. Como os resultados obtidos por esta ferramenta produzem arquivos muito extensos, devido à inclusão do estado anterior e posterior de cada operação, foi adotada a estratégia de simplificação para apresentação destes resultados.

Embora estes resultados tenham sido obtidos através de exemplos simples, os mesmos indicam claramente o potencial para a análise qualitativa, prevista na metodologia $CAD\mathbb{Z}$.

A forma mais simples de definir a melhor rota de uma dada origem até um destino específico é baseado na rota mais curta, ou seja, a rota que tem a menor quantidade de roteadores intermediários. Esta abordagem não considera outras propriedades, por exemplo, a latência. Uma representação gráfica desta situação é mostrada na Figura 5.1, onde as linhas bidirecionais em destaque representam a rota mais curta entre R1 e R3.

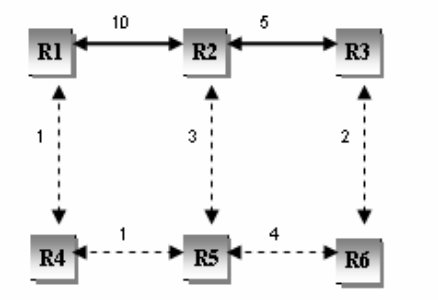

Figura 5.1 A rota mais curta de um cenário (Ramos, 2006a)

O resultado parcial da animação efetuada pela ferramenta Possum pode ser observada na Figura 5.2, onde o menor caminho obtido é destacado. A operação *GetShortestRoute* representa a execução do algoritmo. As variáveis de saída denominadas *sroute!* e *shortestRoute!* representam as rotas possíveis e a menor rota, respectivamente.

```
66 ARCHITECTURE: !!P_C_S!!GetShortestRoute {("R1","R3")/sourceDestination?}
!!P_C_F!!
   ...
  sroute! :=
  {("R#4", {("R1", "R2"), ("R2", "R3")}),
    ("R#5", {("R1", "R2"), ("R2", "R5"), ("R5", "R6"), ("R6", "R3")}),
    ("R#6", {("R1", "R4"), ("R2", "R3"), ("R4", "R5"), ("R5", "R2")}),
    ("R#7", {("R1", "R4"), ("R4", "R5"), ("R5", "R6"), ("R6", "R3")})},
   shortestRoute! := {("R1", "R2"), ("R2", "R3")},
   ...
```

Figura 5.2 Resultado parcial da operação *GetShortestRoute* (Ramos, 2006a)

Uma técnica de roteamento mais complexa considera a rota de latência mínima. Esta técnica melhora potencialmente a comunicação através da Rede-em-Chip, uma vez que um link entre roteadores adjacentes pode apresentar uma latência maior do que rotas

com links intermediários. Esta técnica considera a latência total da rota, isto é, a soma da latência de cada *link* que compõe a rota. A Figura 5.3 exibe a latência entre os roteadores adjacentes e a rota com menor latência.

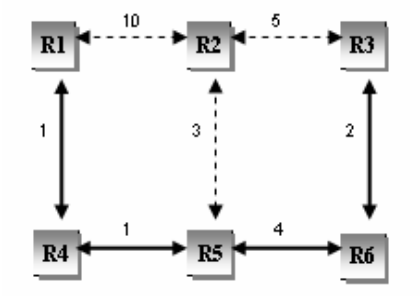

Figura 5.3 A melhor rota de um cenário considerando a latência total (Ramos, 2006a)

O resultado parcial da execução deste algoritmo, formalmente especificado como *GetBestLatencyRoute*, pode ser visualizado na Figura 5.4. A latência total neste exemplo é calculada em 8 unidades de tempo e a melhor rota é mostrada pela variável de saída *bestlatencyroute*.

```
67 ARCHITECTURE: !!P_C_S!!GetBestLatencyRoute
{("R1","R3")/sourceDestination?}    !!P_C_F!!
...
  bestlatency!:= 8
  bestlatencyroute!:={("R1","R4"),("R4","R5"),("R5","R6"),("R6","R3")}
...
```

Figura 5.4 Resultado parcial da operação *GetBestLatencyRoute*

O terceiro e último algoritmo de roteamento analisado leva em consideração o estado dos *links*, que é uma importante propriedade arquitetural, principlamente quando o paralelismo em comunicação é possível. Os valores que podem ser atribuídos ao estado são: reservado (*reserved*), ocioso (*idle*), ocupado (*busy*) ou desligado (*off*). Somente os *links* com estado ocioso são considerados na identificação de rotas possíveis. A Figura 5.5 ilustra esta situação.

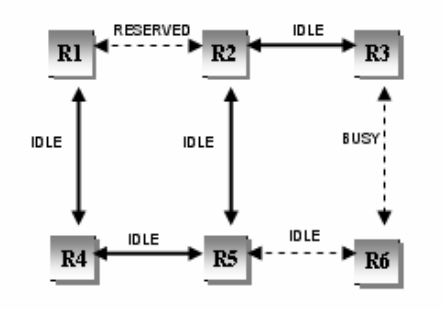

Figura 5.5 A melhor rota de um cenário considerando o estado dos *links*

Apesar dos algoritmos de roteamento terem sido apresentados isoladamente, é possível combiná-los em algoritmos mais complexos, a fim de obter melhores rotas. A análise qualitativa baseada na especificação formal pode ser considerada uma ferramenta essencial para o projetista, uma vez que um alto nível de abstração da especificação independe de implementação específica, portanto, decisões sobre a arquitetura alvo e seus componentes podem ser postergadas.

A consequência prática disto é a liberdade do projetista em lidar com a funcionalidade da arquitetura durante a fase inicial de desenvolvimento. Embora focado em algoritmos de roteamento, o processo de raciocínio utilizado pode ser estendido a outros mecanismos básicos de rede, tais como chaveamento, controle de fluxo e arbitragem.

5.2 Processamento de Vídeo – Vídeo Object Plane Decoder (VOPD)

Este estudo de caso trata da aplicação de processamento de vídeo, mais precisamente do mapeamento da decodificação de vídeo efetuada pelo *Vídeo Object Plane Decoder* (VOPD) (van der Tol, 2002; Murali, 2004). O objetivo desse estudo de caso, conforme documentado em (Murali, 2004) é mapear os processos da aplicação em uma Rede-em-Chip, de modo a satisfazer as restrições de largura de banda da Rede-em-Chip e minimizar o atraso da comunicação.

O decodificador VOP consiste em uma arquitetura dedicada cuja estrutura básica pode ser visualizada por meio da Figura 5.6.

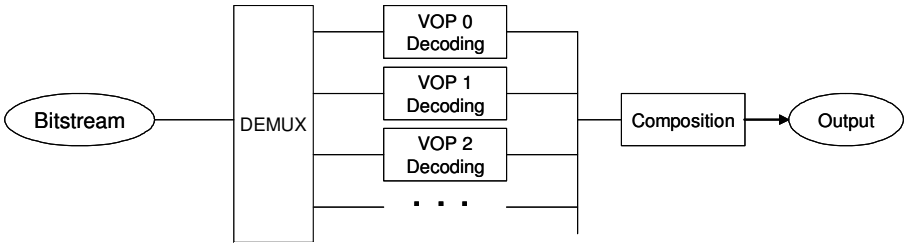

Figura 5.6 Estrutura de um decodificador VOP

O projeto de decodificadores também pode envolver a utilização de processadores de sinais digitais (DSP). DSP são microprocessadores especializados em processamento de áudio e vídeo, quer o decodificador seja utilizado em tempo-real ou não (Wolf, 2001). Este tipo de componente é bastante utilizado, principalmente em eletrônica de consumo, tais como instrumentos musicais, e dispositivos móveis.

DSPs podem ser utilizados como filtros, a fim de melhorar a qualidade do sinal digital. Isto é possível pela capacidade que os DSPs têm de repetir uma instrução complexa em extrema velocidade, por exemplo, as Transformadas Rápidas de Fourier (*Fast Fourier Transform* – FFT). Estes filtros são muito utilizados em telecomunicações para decodificar a sinalização *Dual Tone MultiFrequential* – DTMF, *MultiFrequential Code*- MFC, etc.

Neste estudo de caso é dada ênfase à dinâmica da aplicação da metodologia *CAD*$\mathbb{Z}$, conforme apresentada no capítulo 4.

5.2.1 Modelagem de Requisitos

Uma característica fundamental da metodologia *CAD*$\mathbb{Z}$ é integrar os requisitos da aplicação ao projeto da arquitetura, gerando uma especificação integrada de software e hardware. O levantamento dos requisitos da aplicação resulta em um grafo cujos valores das arestas podem representar largura de banda, latência máxima, energia máxima consumida, etc. A equipe de modelagem de requisitos é responsável por definir uma, ou

mais de uma visões específicas dos requisitos do projeto na forma de um grafo modelado em Z.

Devido o foco da tese estar na comunicação, nesse estudo de caso, informações referentes aos nós (processos) foram omitidas, sendo definidas as arestas dos *links*, que representam a visão específica da aplicação em relação à demanda de largura de banda necessária para a comunicação entre os processos ou elementos de processamento. A Figura 5.7 mostra o grafo de processos gerado segundo a documentação constante em (Murali, 2004).

É importante observar que para a modelagem da aplicação é suficiente definir os processos e os requisitos da aplicação. Assim sendo, os processos foram genericamente identificados no grafo (v1, v2,..,vn). Em um cenário real, tais processos podem ser identificados pela função ou componente genérico, por exemplo, DEMUX, VOP 0 Decoding, etc., conforme visualizado na Figura 5.6.

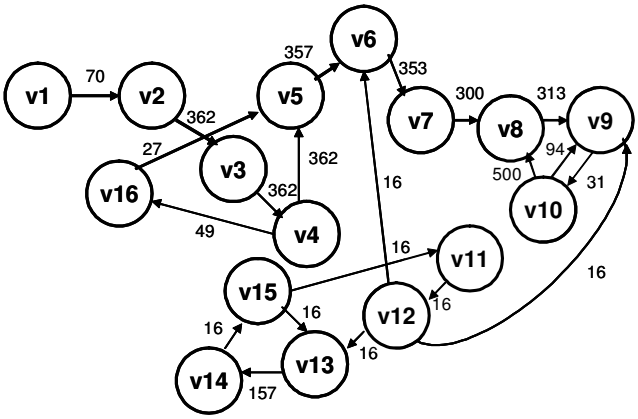

Figura 5.7 Modelagem da aplicação (Murali, 2004)

A modelagem em Z do grafo da Figura 5.7 é realizada a partir do esquema *Application* (ver Figura 5.8 a) que especifica o conjunto de processos da aplicação (v1..v16), a comunicação entre os processos ((v1,v2), (v2,v3),...,(v15,v13), etc.), os rótulos das arestas (((v1,v2),70), ((v2,v3),362), etc.), o requisito da aplicação seguido do seu nível de exigência ((*bandwidth*, *minimum*)) e a quantidade de processos da aplicação (16).

Enquanto o esquema *Application* define o grafo da Figura 5.7, o esquema *Bandwidth* (ver Figura 5.8 b) associa um valor ao nível mínimo, ou máximo, exigido para a

largura de banda necessária entre os processos. Este valor é associado por meio da função *requireLevelValue*, e dos invariantes constantes na parte do predicado do esquema *Bandwidth*.

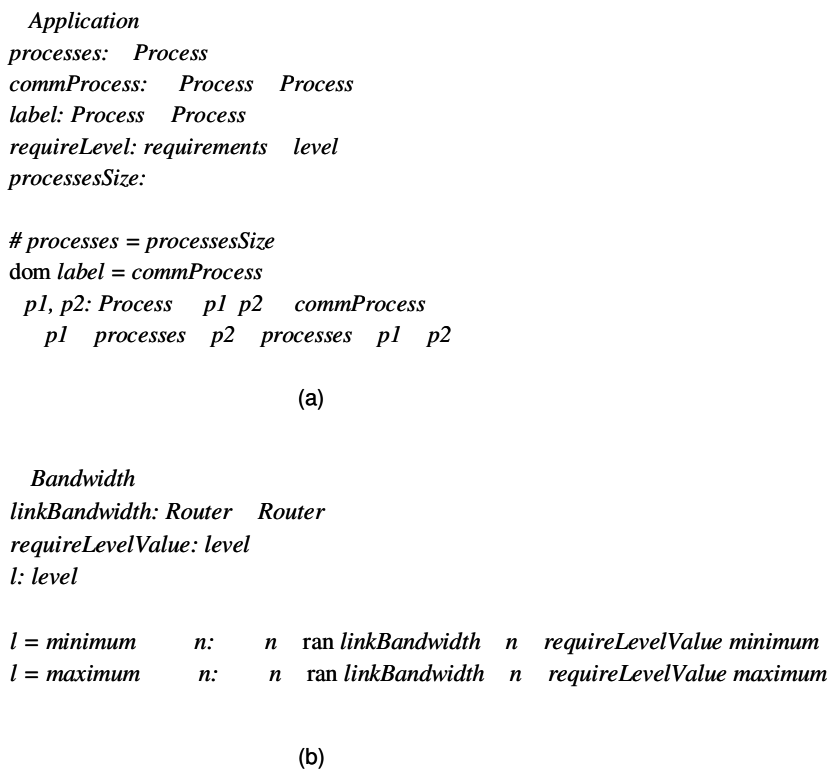

Figura 5.8 Modelagem da aplicação com relação à visão específica da largura de banda

É importante observar que a modelagem da arquitetura, realizada pela equipe de projeto do sistema (terceira equipe), e a modelagem da aplicação, realizada pela equipe de modelagem de requisitos (primeira equipe), podem ser efetuadas concomitantemente, uma vez que a equipe de modelagem de requisitos efetua a modelagem dos requisitos da aplicação e as equipes de especificação e de projeto do sistema fazem a integração Hardware/Software.

5.2.2 Especificação dos Elementos

De acordo com a documentação constante em (Murali, 2004), a topologia considerada para implementação da estrutura de interconexão foi a topologia em malha com dezesseis nós (ver Figura 5.9). Esta topologia foi adotada devido a sua ampla utilização em vários projetos, porém os autores ressaltam que as técnicas utilizadas no trabalho não se limitam a topologia em malha.

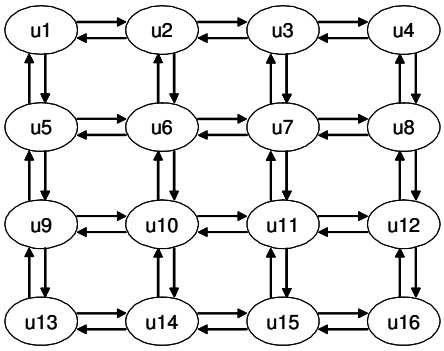

Figura 5.9 Rede-em-Chip de 16 núcleos (Murali, 2004)

Na metodologia *CAD*$\mathbb{Z}$, a partir do levantamento realizado pela equipe de especificação dos possíveis elementos já especificados, ou não, que poderão atender aos requisitos da aplicação é possível sugerir alguns elementos que se adéquam ao projeto. A integração inicial de Hardware/Software é realizada pela equipe de especificação, a qual analisa as várias possibilidades de hardware para o atendimento dos requisitos da aplicação. Neste estudo de caso, por exemplo, o esquema *HW_SW_Integration* pode assumir a topologia *NoCMesh2D* ou *Tree*, conforme modelagens exibidas na Figura 5.10.

A topologia a ser selecionada pela equipe de projeto do sistema deve respeitar a restrição constante no esquema *HW_SW_ Integration* a qual exige que a quantidade de roteadores da estrutura de interconexão seja superior ou igual a quantidade de processos da aplicação.

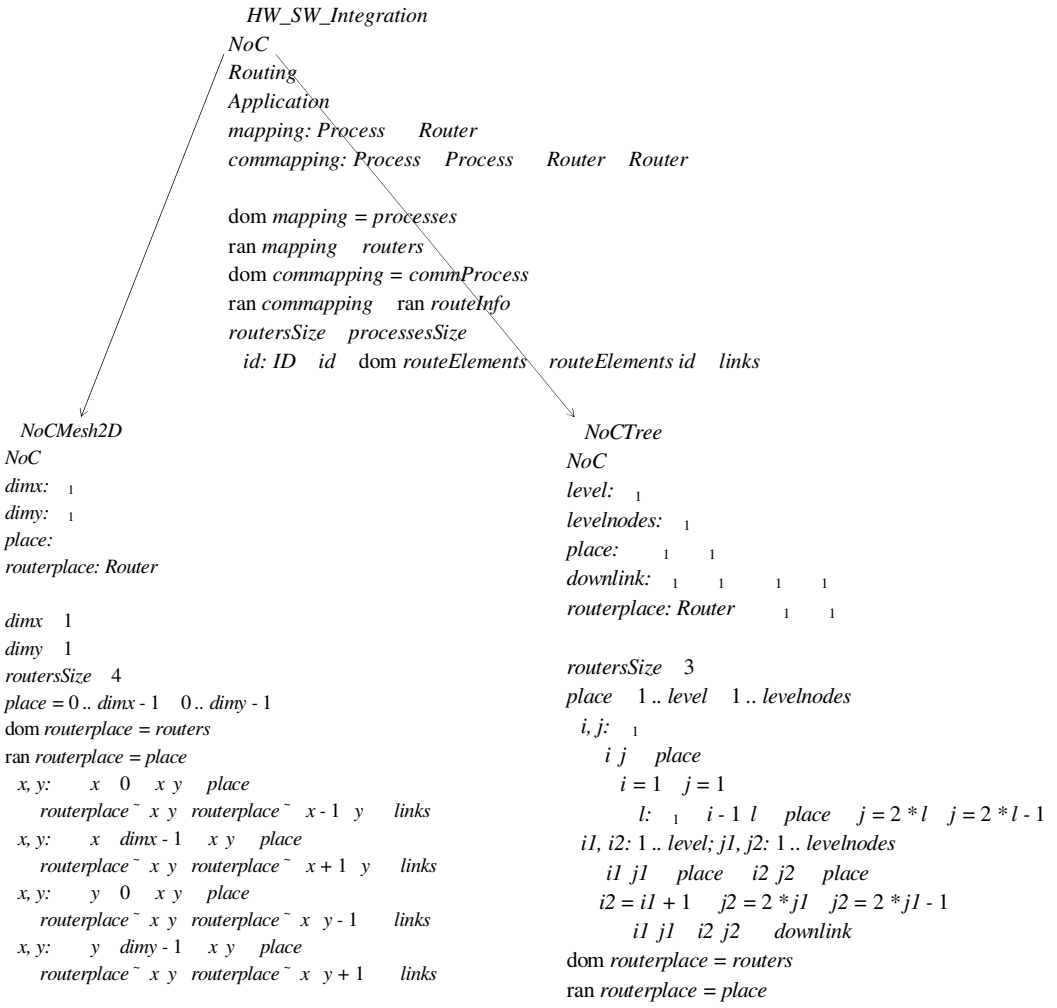

Figura 5.10 Modelagens Z de opções de estruturas de interconexão

Em (Murali, 2004), os autores apresentam um algoritmo de mapeamento que utiliza o roteamento do menor caminho entre os nós de uma arquitetura em malha. Utilizando a metodologia *CAD*$\mathbb{Z}$ é possível por meio da conjunção dos esquemas *Application*, *NoC* e *Routing* definir as rotas com a menor quantidade de roteadores entre dois nós que se comunicam. Como o esquema *HW_SW_Integration* consiste na conjunção dos esquemas *Application* e *NoC*, o mapeamento utilizando o roteamento do caminho mínimo pode ser realizado pela conjunção dos esquemas *HW_SW_Integration* e *Routing* (ver Figura 5.11)

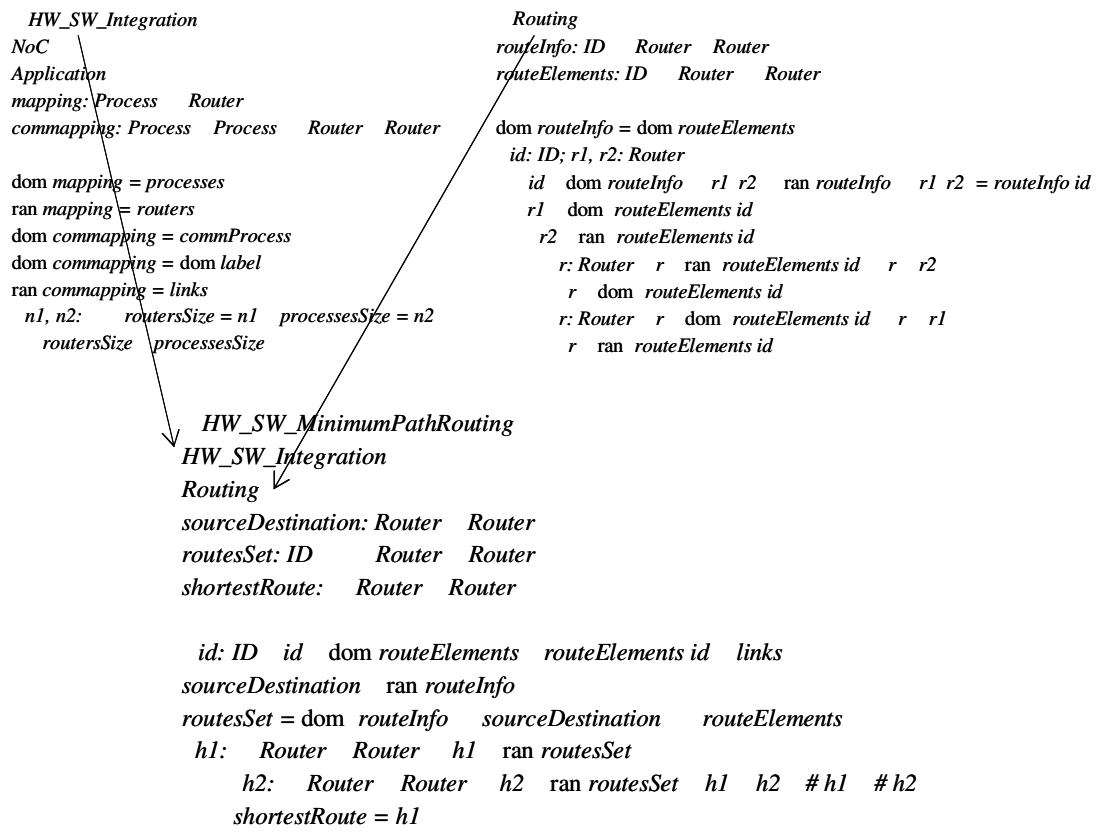

Figura 5.11 Modelagem de mapeamento com base no roteamento de menor caminho

Como a modelagem é realizada em um nível de abstração alto, a equipe de especificação utiliza a conjunção de esquemas como uma técnica de refinamento sucessivo para obter especificações cada vez mais concretas dos elementos envolvidos. Este refinamento também é realizado pela equipe de projeto. É importante lembrar que a reusabilidade é uma característica da metodologia *CAD*$\mathbb{Z}$ e, portanto, esse esforço de especificação deve ser maior no início de sua utilização, devendo diminuir à medida que novos projetos são desenvolvidos.

5.2.3 Projeto do Sistema

A partir de algoritmos de mapeamento que satisfazem as restrições de largura de banda e minimizam o atraso na comunicação apresentados em (Murali, 2004), os processos da aplicação referente ao decodificador VOP são mapeados em uma Rede-em-Chip do tipo malha, gerando a arquitetura exibida na Figura 5.12.

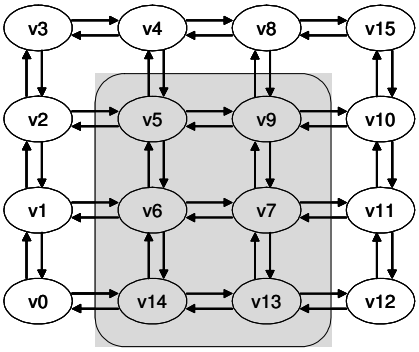

Figura 5.12 Mapeamento da Aplicação na Rede-em-Chip (Murali, 2004)

Na metodologia *CAD*$\mathbb{Z}$, a equipe de projeto do sistema utilizando os modelos gerados pela equipe de especificação refina o modelo *HW_SW_Integration*, gerando o modelo *HW_SW_RefinedIntegration* (ver Figura 5.13) o qual define a topologia malha bi-dimensional (*NoCMesh2D*) como a estrutura de interconexão da Rede-em-Chip.

HW_SW_RefinedIntegration
NoCMesh2D
Application
mapping: Process Router
commapping: Process Process Router Router

dom *mapping* = *processes*
ran *mapping* = *routers*
ran *mapping* = dom *routerplace*
dom *commapping* = *commProcess*
dom *commapping* = dom *label*
ran *commapping* = *links*
n1, n2: $_1$ *n1 = dimx n2 = dimy routersSize = n1 * n2*
n1, n2: routersSize = n1 processesSize = n2
routersSize processesSize

Figura 5.13 Esquema de integração HW/SW Refinado

Enquanto o esquema *HW_SW_Integration* (ver Figura 5.11) define uma estrutura de roteadores e canais físicos, independente da topologia utilizada, o esquema *HW_SW_RefinedIntegration* define a topologia malha como a estrutura de interconexão dos roteadores. A partir da definição da topologia utilizada, novas restrições são inseridas àquelas já existentes. Por exemplo, é necessário garantir que a quantidade de roteadores (*routersSize*) da Rede-em-Chip seja igual ao produto das dimensões (*dimx*, *dimy*) da malha.

A partir do esquema *HW_SW_RefinedIntegration* é possível fazer análises qualitativas, de propriedades da estrutura de comunicação tais como largura de banda, latência, consumo de energia, entre outras (ver Figura 3.9). Para tanto, é necessário que tais propriedades estejam especificadas.

Por meio do esquema *HW_SW_Band_RefinedIntegration*, por exemplo, a equipe de projeto do sistema pode analisar os níveis de exigências da propriedade de largura de banda por parte das aplicações. A Figura 5.14 exibe a modelagem da análise qualitativa referente à largura de banda, a qual consiste da conjunção dos esquemas *HW_SW_RefinedIntegration* e *Bandwidth* (ver Figura 5.8 b), bem como de restrições que devem ser respeitadas e se manterem verdadeiras ao longo da análise.

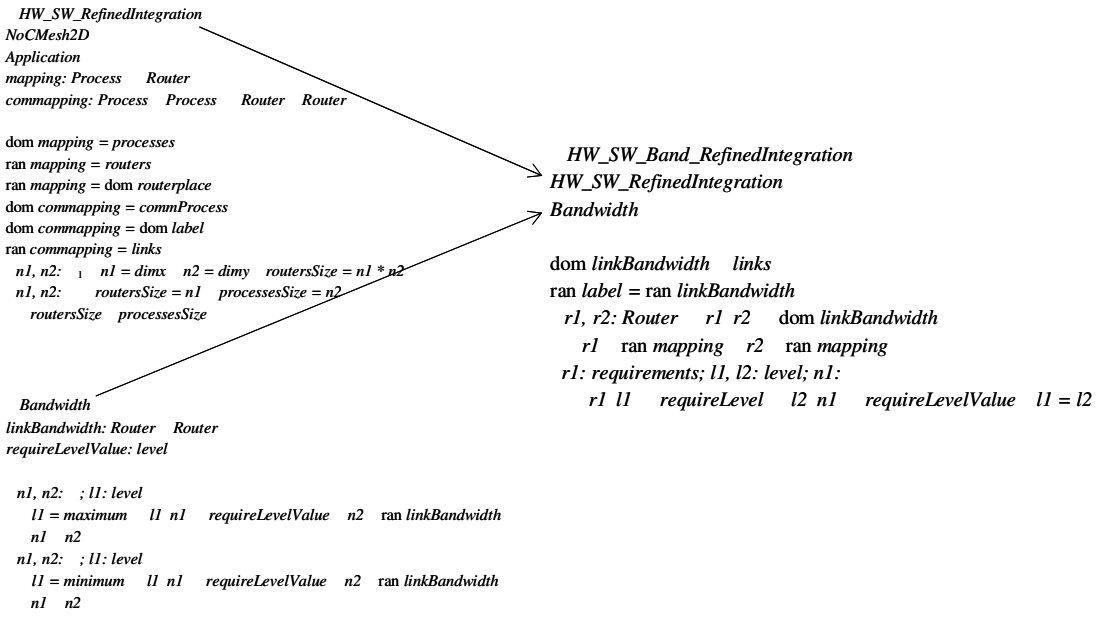

Figura 5.14 Análise qualitativa referente à largura de banda

Após o refinamento do esquema *HW_SW_Integration*, para uma determinada topologia, a equipe de projeto de sistema refina o esquema *HW_SW_MinimumPathRouting* (ver Figura 5.15) para a estrutura de Rede-em-Chip do tipo malha. A partir deste esquema é possível mapear, assim como em (Murali, 2004), os processos da aplicação segundo a rota mais curta, atendendo a restrição de largura de banda da aplicação. A inclusão da modelagem *HW_SW_Band_RefinedIntegration* (ver Figura 5.14) na parte declarativa do esquema abaixo caracteriza o mapeamento dos processos da aplicação segundo a rota mais curta, atendendo a restrição de largura de banda da aplicação.

HW_SW_RefinedMinimumPathRouting
HW_SW_Band_RefinedIntegration
Routing
sourceDestination: Router Router
routesSet: ID Router Router
shortestRoute: Router Router

id: ID id dom *routeElements routeElements id links*
sourceDestination ran *routeInfo*
routesSet = dom *routeInfo sourceDestination routeElements*
h1: Router Router h1 ran *routesSet*
h2: Router Router h2 ran *routesSet h1 h2 # h1 # h2*
shortestRoute = h1

Figura 5.15 Esquema do mapeamento com base no roteamento de menor caminho a partir de uma topologia específica de Rede-em-Chip

Na metodologia *CAD*$\mathbb{Z}$, a latência da comunicação entre os processos da aplicação em uma estrutura de Rede-em-Chip pode ser analisada pelo esquema de especificação da propriedade latência (*Latency*) associada ao esquema *HW_SW_RefinedIntegration*. O esquema resultante *HW_SW_Latency_RefinedIntegration* exibido na Figura 5.16 além de mesclar as restrições existentes nos esquemas conjugados, insere restrições referentes a análise qualitativa. As duas últimas restrições do esquema

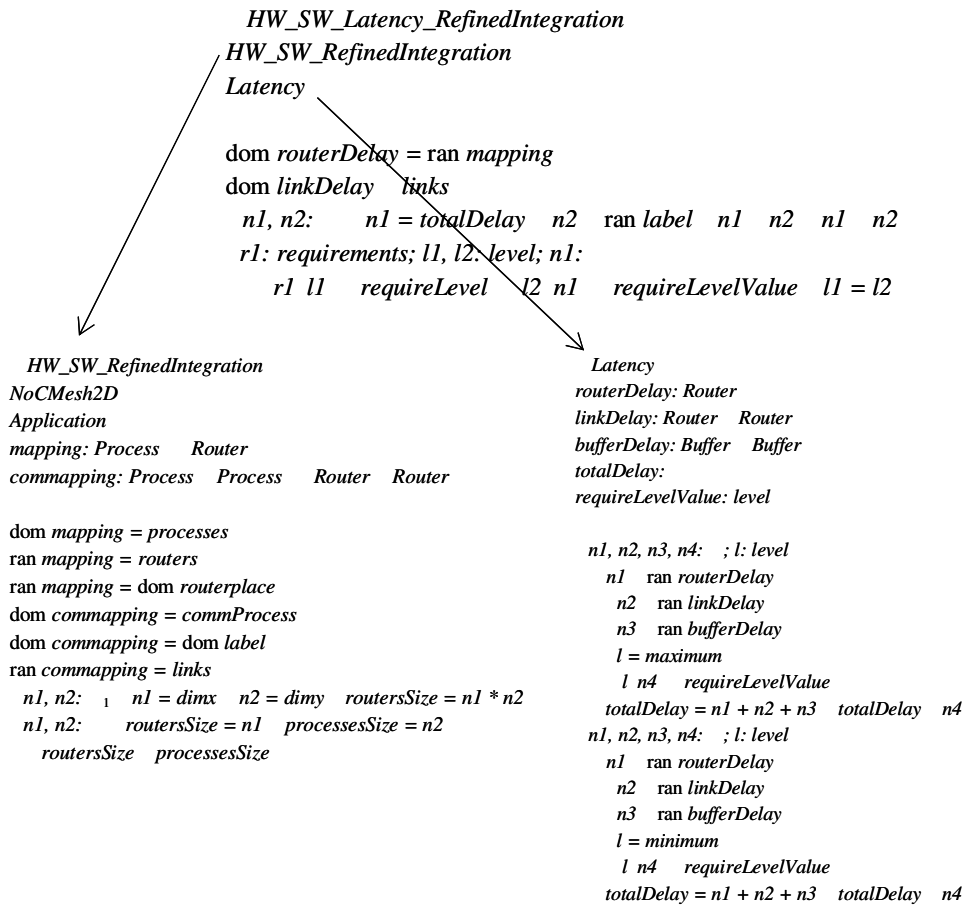

Figura 5.16 Análise qualitativa referente à latência

5.3 Considerações Finais

Este capítulo apresentou dois estudos de caso que validam a metodologia *CAD*$\mathbb{Z}$. Enquanto o primeiro estudo mostra que é possível, utilizando ferramentas específicas, validar propriedades diretamente da especificação formal, o segundo mostra que é possível realizar diferentes mapeamentos de processos com base nos requisitos da aplicação.

Considerando que todos os elementos especificados são disponibilizados em uma biblioteca, facilitando o reuso dos mesmos em outros projetos, em termos práticos significa dizer que a biblioteca de especificações, formaliza todo um conhecimento

adquirido em desenvolvimento de projetos desta natureza. Essa é precisamente um dos grandes benefícios da adoção da metodologia $CAD\mathbb{Z}$

6. Conclusões e Perspectivas Futuras

Esta tese apresentou uma nova metodologia de projeto baseada em Z, denominada $CAD\mathbb{Z}$. A utilização das noções de conjuntos, relações, funções e suas várias propriedades e operações, junto com as idéias de quantificação universal e existencial utilizadas pela notação Z, permitiram a construção de maneira clara e direta de especificações para projetos de Redes-em-Chip, bem como, para análise qualitativa das propriedades vitais de comunicação. Um dos resultados práticos da usabilidade de Z em projetos de Rede-em-Chip e na análise qualitativa é a redução de custos com verificação nas fases posteriores do projeto.

Embora tenha sido enfatizada a estrutura de Redes-em-Chip, a metodologia apresentada pode ser aplicada em projetos que utilizam outro tipo de estrutura de interconexão.

A elaboração da metodologia considerou as seguintes premissas:

- A comunicação representa um aspecto crítico no projeto de sistemas-em-chip (SoC);
- A estrutura de interconexão pode ser utilizada como base para a otimização do projeto;
- A análise da qualidade do projeto deve acontecer nas fases iniciais do desenvolvimento;
- O reuso representa uma forma efetiva para acelerar o desenvolvimento do projeto;
- A especificação formal é uma importante forma de reter o conhecimento adquirido favorecendo o reuso;
- A complexidade dos projetos atuais é um parâmetro para a adoção de metodologia;
- A simplicidade da metodologia $CAD\mathbb{Z}$ facilita a sua adoção.

A metodologia $CAD\mathbb{Z}$ adota o paradigma Co-Design a partir do qual são considerados os requisitos da aplicação e as funcionalidades que a arquitetura deverá prover. A

modelagem realizada pela metodologia *CAD*$\mathbb{Z}$ é feita no nível de abstração alto, permitindo o refinamento sucessivo de modelos abstratos de hardware e software para projetos de sistemas embarcados.

Por meio de estudo de caso foi possível mostrar o uso da metodologia em projetos reais, bem como, que uma Rede-em-Chip pode ser genericamente definida e sucessivamente refinada em uma Rede-em-Chip mais específica, conservando as propriedades definidas e garantidas nas especificações realizadas anteriormente.

Outra característica apresentada pela metodologia *CAD*$\mathbb{Z}$ refere-se à reusabilidade, que é obtida a partir do aproveitamento de elementos especificados e é considerado fundamental para acelerar o desenvolvimento do projeto. Desta forma, a metodologia retém em uma biblioteca de especificações o conhecimento adquirido no projeto, a fim de que possa ser utilizado novamente.

Ruídos na documentação do projeto, por exemplo, ambigüidade ou omissão podem inviabilizar o reuso. Essa justificativa fundamentou o uso de linguagem formal Z, cuja base matemática permite descrever com exatidão as características essenciais dos elementos usados no projeto. Embora use de rigor matemático, a notação Z é relativamente fácil de aprender, pois é baseada em teoria dos conjuntos. A expressividade é outro ponto forte desta linguagem, que permite descrever elementos em alto nível de abstração e refiná-los sucessivamente, até uma descrição que pode servir de base para a codificação em uma linguagem de programação.

Projetos de hardware são extremamente complexos, dada a complexidade dos elementos envolvidos e as restrições que devem ser consideradas durante todo o projeto. Quanto maior a complexidade mais forte é a justificativa para adoção de uma metodologia de projeto. Uma das vantagens da metodologia de projeto refere-se à definição clara de papés e atribuições, o que facilita sem dúvida o trabalho em equipe, favorece o aumento da produtividade e consequentemente reduz o tempo total do projeto.

Contudo para que uma metodologia possa ser efetivamente usada é importante que a equipe reconheça suas vantagens e principalmente assimile-a de forma rápida e natural. Dessa forma, quanto mais simples a metodologia, maiores as chances de sua adoção. A

simplicidade, portanto, é um princípio que norteia a metodologia $CAD\mathbb{Z}$, o que fica visível no número reduzido de fases e tarefas, embora a complexidade seja inerente a projetos dessa natureza.

6.1 Resumo das Contribuições

Esta seção resume as principais contribuições deste trabalho de tese. Embora as contribuições efetivas de uma metodologia só possam ser validadas após seu uso em situações reais, o que não seria possível neste caso, a validação com os estudos de caso descritos no Capítulo 5, permite inferir os benefícios da metodologia $CAD\mathbb{Z}$.

A primeira contribuição obtida neste trabalho é a identificação clara do papel que a estrutura de interconexão exerce no projeto como um todo. Dado o aumento significativo da quantidade de elementos em um único SoC, a forma como a comunicação entre esses elementos se dá pode ser um fator decisivo para a obtenção do desempenho esperado. Sistematizar a preocupação com esse fato durante todo o desenvolvimento do projeto é uma forma de trabalhar para o sucesso e alinhar toda a equipe na consecução deste objetivo.

Considerar a comunicação um fator crítico para sucesso do projeto, permite focar a estrutura de interconexão em esforços de otimização. Esta estratégia é fundamentada na elaboração de um modelo de comunicação que traduz a necessidade de comunicação da aplicação. A representação desta necessidade em um grafo permite ainda associar pesos aos nós (processos) e *links*, e com isso caracterizar melhor aspectos associados aos processos e a comunicação entre eles, respectivamente. Esta estratégia de otimização é também uma contribuição.

Outra contribuição e principal diferencial entre a metodologia $CAD\mathbb{Z}$ e as demais é a identificação de linguagens formais como um elemento chave na retenção de conhecimento de projetos e base para o reuso. Ao utilizar este tipo de linguagem durante as fases iniciais, é possível antecipar a avaliação de propriedades do projeto, e

consequente identificação em tempo hábil de potenciais problemas, que possam vir a comprometer o sistema.

Embora críticos de métodos formais possam apontar o formalismo como uma dificuldade, principalmente pela necessidade de treinamento e pelo uso de operadores matemáticos não usuais, dois aspectos precisam ser considerados nesta análise. O primeiro refere-se à utilização de um pequeno subconjunto de operadores da linguagem, que é originária da teoria dos conjuntos e, portanto largamente conhecidos. O segundo refere-se ao reuso, característica da metodologia $CAD\mathbb{Z}$, que possibilita a redução gradativa da atividade de especificação. Outro aspecto, que depende obviamente do tamanho da equipe de projeto, refer-se ao fato de que nem todos os envolvidos no projeto precisem assimilar e usar o formalismo.

É importante observar que as especificações formais representam uma descrição rigorosa e não-ambígua que podem ser utilizadas para auxiliar a geração de código. Esta geração pode ser manual ou automatizada. Existem várias propostas de geração automática de código a partir de especificações formais, por exemplo, de Z para Java. Detalhes sobre a utilização das especificações formais dos elementos em software ambiente para projeto representam também uma contribuição significativa deste trabalho.

6.2 Perspectivas Futuras

Esta seção aponta atividades futuras, identificadas a partir deste trabalho de tese. Embora a metodologia possa ser aplicada de forma não automatizada, a criação de um software que simule um ambiente real de projeto é sem dúvida um importante trabalho a ser desenvolvido. As especificações formais dos elementos, a identificação de equipes e as tarefas da metodologia $CAD\mathbb{Z}$ serão de extrema importância no desenvolvimento deste ambiente.

É importante validar a utilização da metodologia em projetos diferenciados a fim de ampliar a base de conhecimento sobre as propriedades associadas a esses projetos. Por

exemplo, é importante conhecer precisamente quais os impactos que outras estruturas de interconexão, além das redes-em-chip, exercem sobre o projeto, bem como compreender “se” e “como” as mesmas podem ser utilizadas na otimização do projeto.

Outra atividade a ser executada no futuro, é a avaliação sobre o aprofundamento da metodologia referente ao refinamento dos modelos de hardware.

Referências Bibliográficas

Adriahantenaina, A., Charlery, H., Greiner, A., Mortiez, L. and Zeferino, C. A., (2003), SPIN: A Scalable, Packet Switched, on-chip Micro-network, Proc. Design, Automation and Test in Europe, Conference and Exhibition.

Agarwal, A. (1999), Raw Computation, Scientific American, August. Disponível em: http://www.sciamdigital.com

ARM (2006), Disponível em: http://www.arm.com/products/solutions/AMBAHomePage.html

Benini, L. and De Micheli, G. (2001), Powering Networks on Chip, Proc. of the 14th Int. Symp. on System Synthesis, pp. 33-38.

Benini, L. and De Micheli, G. (2002), Networks on chips: A new SoC paradigm. IEEE Computer, 35(1):70–80.

Benini L. and De Micheli, G. (2004), Networks on Chip: A new paradigm for component-based MPSoC design," in A. Jerrraya and W. Wolf Editors, Multiprocessors Systems on Chips, Morgan Kaufman, pp. 49-80.

Bergamaschi, R. and Cohn, J. (2002), The A to Z of SoCs, Computer Aided Design, ICCAD 2002. IEEE/ACM International Conference

Bertozzi, D., Jalabert, A., Murali, S., Tamhankar, R.R., Stergiou, S., Benini, L. and De Micheli, G. (2005), NoC Synthesis Flow for Customized Domain Specific Multiprocessor Systems-on-Chip. IEEE Transactions on Parallel and Distributed Systems, 16, Issue 2:113–129.

Beyer, S., Bohm, P., Gerke, M., Hillebrand, M., der Rieden, T., Knapp, S., Leinenbach, D. and Paul, W.J. (2005) Towards the formal verification of lower system layers in automotive systems. Computer Design: VLSI in Computers and Processors.

Bhojwani, P., Mahapatra, R., Kim, E. and Chen, T. (2005), A Heuristic for Peak Power Constrained Design of Network-on-Chip (NoC) based Multimode Systems, VLSI Design, 18th International Conference on Publication, pp. 124-129.

Bjerregaard, T., Mahadevan, S. (2006), A Survey of Research and Practices of Network-on-chip, ACM Computing Surveys, Vol. 38.

Bobda, C., Majer, M., Koch, D., Ahmadinia, A. and Teich, J. (2004), A Dynamic NoC Approach for Communication in Reconfigurable Devices, Proc. in Field-Programmable Logic, pp.1032-1036

Boekhorst, F. (2002), Ambient Intelligence, the Next Paradigm for Consumer Electronics: How Will it Affect Silicon, Proc. Int'l Solid State Circuits Conf. 2002, pp. 28-31.

Bolotin, E., Cidon, I., Ginosar, R. and Kolodny, A. (2004), QNoC: QoS Architecture and Design Process for Network on Chip, Journal of Systems Architecture 50(2–3), 105–128.

Bowen, J.P., (1996), Formal Specification and Documentation using Z, International Thomson Computer Press, ISBN 1-850-32230-9.

Brebner, G. and Levi, D. (2003), Networking on Chip with Platform FPGAs, Proc. IEEE International Conference on Field-Programmable Technology.

Carro, L., Wagner, F. (2003), Sistemas Computacionais Embarcados. JAI - XXII Jornada de Atualização em Informática.

Cesário, W. O., Lyonnard, D., Nicolescu, G., Paviot, Y., Yoo, S., Jerraya, A.A., Gauthier, L., Diaz-Nava, Mario (2002), Multiprocessor SoC Platforms: A Component-Based Design Approach, IEEE Design & Test, v.19 n.6, p.52-63.

Chan, J. and Parameswaran, S. (2004), NoCGEN: A Template Based Reuse Methodology for Networks on Chip Architecture, In Proc. International Conference on VLSI Design, p.717.

Chiu, G. (2000), The odd-even turn model for adaptive routing, IEEE Transactions on Parallel and Distributed Systems, vol. 11, pp.729-38.

Cisco (2006), Disponível em: http://www.cisco.com/

Cohen, B. (1989), Justification of Formal Methods for System Specification. Software Engineering Journal, pages 26-35.

Crow, J., Owre, S., Rushby, J., Shankar, N. and Srivas, M. (1995), A tutorial introduction to PVS. In Workshop on Industrial-Strength Formal Specification Techniques, 1995.

Crowder, S., Hannon, R., Ho, H., Sinitsky, D., Wu, S., Winstel, K., Khan, B., Stiffler, S.R. and Iyer, S.S. (1998), Integration of trench DRAM into a high-performance 0.18 μm logictechnology with copper BEOL, International Electron Devices Meeting.

Dall'Osso, M., Biccari, G., Giovannini, L., Bertozzi, D. and Benini, L. (2003) "Xpipes: a Latency Insensitive Parameterized Network-on-Chip Architecture for Multi-Processor SoCs", International Conference on Computer Design, pp.536-539.

Dally, W., Seitz, C. (1987), Deadlock-Free Message Routing in Multiprocessor Interconnection Networks, IEEE Transactions on Computers, Vol. C-36, nr. 5.

Dally, W., and Aoki, H. (1993), Deadlock-free Adaptive Routing in Multicomputer Networks Using Virtual Channels", IEEE Trans. on Parallel and Distributed Systems, April 1993, pp. 466-475.

Dally, W. & Towles, B. (2001), "Route Packets, Not Wires: On- Chip Interconnection Networks", Proc. of Design Automation Conference, pp. 684-689.

Day, J. D. and Zimmerman, H. (1983), The OSI Reference Model. In Proceedings of the IEEE, volume 71, pages 1334–1340.

Duato, J. (1993), A New Theory of Deadlock-Free Adaptive Routing in Wormhole Networks, IEEE Transactions on Parallel and Distributed Systems.

Duato, J., Yalamanchili, S. and Ni, L. (1997), Interconnection Networks: An Engineering Approach. Morgan Kaufmann Publishers.

Edwards, S. A. (2000), Languages for Digital Embedded Systems. Kluwer Academic Publishers, USA.

Foster, I. (1995), Designing and Building Parallel Programs, Addison-Wesley.

Gangwal, O. P., Radulescu, A., Goossens, K., Pestana, S. G. and Rijpkema, E. (2005), Building Predictable Systems on Chip: An Analysis of Guaranteed Communication in the Æthereal Network on Chip. In P. van der Stok, editor, Philips Research Book Series, chapter 1.

Genko, N., Atienza, D., De Micheli, G., Benini, L., Mendias, J.M., Hermida, R. and Catthoor, F. (2005), A Novel Approach for Network on Chip Emulation, International Symposium on Computer Architecture, pp. 2365- 2368.

Gebremichael, B., Vaandrager, F.W., Zhang, M., Goossens, K., Rijpkema, E. and Radulescu, A.(2005), Deadlock Prevention in the Æthereal Protocol. In D. Borrione and W. Paul, editors. Proceedings 13th IFIP Advanced Research Working Conference on Correct Hardware Design and Verification Methods, pp. 345 – 348.

Glass, C. and Ni, L. (1998), The Turn Model for Adaptive Routing. In 25 Years ISCA: Retrospectives and Reprint, pp. 441-450.

Goossens, K., van Meerbergen, J., Peeters, A. andWielage, P. (2002), Networks on Silicon: Combining Best-effort and Guaranteed Services, Proc. Design, Automation and Test in Europe Conference and Exhibition, pp. 423–425.

Goossens, K., Dielissen, J., van Meerbergen, J., Poplavko, P., Radulescu, A., Rijpkema, E., Waterlande, E. and Wielage, P. (2003), Guaranteeing the Quality of Services in Networks on Chip. In A. Jantsch and H. Tenhunen, editors, Networks on Chip. Kluwer.

Goossens, K., Dielissen, J., Grangwal, O. P. and Pestana, S. G. (2005a), A Design Flow for Application-Specific Networks on Chip with Guaranteed Performance to Accelerate SOC Design and Verification, Proc. Design, Automation and Test in Europe Conference and Exhibition.

Goossens, K. (2005b), Formal Methods for Networks on Chips, ACSD'05, Fifth International Conference on Application of Concurrency to System Design, pp188-189.

Goossens, K., Dielissen, J. and Radulescu, A.(2005c), Æthereal network on chip: concepts, architectures, and implementations. IEEE Design and Test of Computers, volume: 22, Issue: 5, pp 414 – 421.

Guerrier P., Greiner A. (2000), A Generic Architecture for on Chip Packet Switched Interconnections, Proc. Design, Automation and Test in Europe Conference and Exhibition, pp. 250–256.

Hansson, A., Goossens, K. and Radulescu, A. (2005), A Unified Approach to Constrained Mapping and Routing on Network-on-Chip Architectures. In Int'l Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS).

Hazel, D., Strooper, P., Traynor, O. (1997), Possum: An Animator for the SUM Specification Language. In Proc. AsiaPacific Software Engineering Conference and Int. Computer Science Conference, pp. 42--51.

Hemani, A., Jantsch, A., Kumar, S., Postula, A., Öberg, J., Millberg, M., Dan Lindqvist, D. (2000), Network on chip: an architecture for billion transistor era, Proc. of IEEE NorChip Conference.

Ho, R., Mai, K., Horowitz, M. (2001), The Future of wires, Proceedings of the IEEE, pp. 490-504.

Holsmark, R., Högberg, M. and Kumar, S. (2003), Modelling and Evaluation of a Network on Chip Architecture Using SDL. Springer Berlin / Heidelberg, Vol. 2708/2003, pages: 166-182.

Hu, J. and Marculescu, R. (2003), Energy-Aware Mapping for Tile-based NoC Architectures Under Performance Constraints, Proc. Asia & South Pacific Design Automation Conference.

Hu, J. and Marculescu, R. (2004), DyAD – Smart Routing for Networks-on-Chip, Design Automation Conference.

Hu, J. (2005), Design Methodologies for Application Specific Networks-on-Chip (Thesis), Carnegie Mellon University - Carnegie Institute Of Technology.

Hwang, K. (1993), Advanced Computer Architecture: parallelism, scalability, programmability, McGraw-Hill Series in Computer Science.

IBM, (2006), http://www-03.ibm.com/chips/products/coreconnect/

Intel (2006), http://www.intel.com/

ISO/IEC 13568:2002 - Information technology (2002), Z formal specification notation - Syntax, type system and semantics. Disponível em: http://www.iso.org

ITRS, (2003) International Technology Roadmap for Semiconductors – Update. Disponível em http://public.itrs.net

Jalabert, A,, Murali, S., Benini, L. and De Micheli, G. (2004), XpipesCompiler: A Tool for Instantiating Application Specific Networks on Chip", Design, Automation and Test in Europe Conference and Exhibition.

Jantsch, A., Tenhunen, H. (2003), Network on chip. Kluwer Academic Publishers.

Jerraya, A. A. and Wolf, W. (2004a) The What, Why, and How of MPSoCs in A. Jerrraya and W. Wolf Editors, Multiprocessors Systems on Chips, Morgan Kaufman, 2004.

Jerraya, A. A. (2004b), Long Term Trends for Embedded System Design. Euromicro Symposium on Digital System Design (DSD'04). Rennes, France.

Karim, F., Nguyen, A. and Dey, S. (2002), An Interconnect Architecture for Networking Systems on Chips, IEEE Micro 22(5), 36–45.

Kauffman, M., Manolios, P. and Moore, J. S. (2000), Computer-Aided Reasoning: An Approach. Kluwer Academic Publishers.

Kavaldjiev, N., Smit, G. and Jansen, P. 2004, A Virtual Channel Router for On-chip Networks, IEEE International SOC Conference, pp. 289-93.

Keating, M. and Bricaud, P. (2002), Reuse Methodology Manual for System-on-Chip Designs, KAP, Third Edition.

Kermani, P. and Kleinrock, L. (1979), Virtual Cut-Through: A New Computer Communication Switching Technique. Computer Networks 3: 267-286.

Keutzer, K. et al. (2000), System-Level Design: Orthogonalization of Concerns and Platform-Based Design. IEEE Transactions on Computer-Aided Design of Integrated Circuits, Vol. 19, No. 12, pp 1523-1543.

Kumar, S., Jantsch, A., Soininen,J.-P., Forsell, M., Millberg, M., Oberg, J., Tiensyrj¨a, K. and Hemani, A. (2002), A Network on Chip Architecture and Design Methodology. In Proc. of VLSI Annual Symposium, pages 105–112.

Lavagno, L., Sangiovanni-Vincentelli, A. and Hsieh, H. (1996), Embedded system co-design: Synthesis and verification. In G. DeMicheli and M. Sami, editors, Hardware/Software Co-Design, pp. 213--242. Kluwer Academic Publishers, 1996.

Lee, H., Ogras, U., Marculescu, R., Chang, N. (2006), Design Space Exploration and Prototyping for on-Chip Multimedia Applications, Design Automation Conference, pp. 137-142.

Lei, T., Kumar, S. (2003), A Two-step Genetic Algorithm for Mapping Task Graphs to a Network on Chip Architecture. Euromicro Symposium on Digital Systems Design (DSD'03),p. 180.

Li, M., Zeng, Q. and Jone, W. (2006), DyXY - A Proximity Congestion-Aware Deadlock-Free Dynamic Routing Method for Network on Chip, Design Automation Conference.

Liang, J., Swaminathan, S. and Tessier, R. (2000), aSOC: A scalable, single-chip communications architecture, In Proc. of the IEEE Int. Conf. on Parallel Architectures and Compilation Techniques

Liang, J., Laffely, A., Srinivasan, S. and Tessier, R. (2004), An architecture and compiler for scalable on-chip communication. IEEE Tran. on Very Large Scale Integration Systems, pp. 711-726.

Liu, J., Shen, M., Zheng, L. and Tenhunen, H. (2003), System level interconnect design for network-on-chip using interconnect IPs. Proc. International Workshop on System-level interconnect prediction, pp. 117-124.

Liu, J., Zheng, L. and Tenhunen, H. (2004), Interconnect Intellectual Property for Network-on-Chip, Journal of Systems Arcitecture, 65-79.

Lyonnard, D., Yoo, S., Baghdadi, A. and Jerraya, A. (2001), Automatic Generation of Application-Specific Architectures for Heterogeneous Multiprocessor System-on-Chip, Design Automation Conference, Proceedings, ACM.

Madsen, J., Mahadevan, S., Virk, K., and Gonzalez, M. (2003), NoC modeling for system-level multiprocessor simulation. In Proc. Real-Time Systems Symposium.

Martin, G. (2006), Overview of the MPSoC Design Challenge, Design Automation Conference.

Martínez, J., Flich, J., Robles, A., López, P. and Duato, J. (2003), Supporting Adaptive Routing in IBA Switches, Journal of Systems Architecture: the EUROMICRO Journal, Vol. 49 , Issue 10-11, pp. 441 – 456.

Mentor (2006), Seamless CVE, http://www.mentor.com/seamless

Millberg, M., Nilsson, E., Thid, R. and Jantsch, A. (2004), Guaranteed Bandwidth Using Looped Containers in Temporally Disjoint Networks Within the Nostrum Network on Chip, Proc. Design, Automation and Test in Europe, Conference and Exhibition.

Moraes, F., Calazans, N., Mello, A., Muller, L. and OST, L. (2004), HERMES: an infrastructure for low area overhead packet-switching networks on chip, Integration, the VLSI Journal, vol. 38-1, pp. 69-93.

Murali, S. and De Micheli, G. (2004), Bandwidth-Constrained Mapping of Cores onto NoC Architectures, Proc. Design, Automation and Test in Europe Conference and Exhibition.

Nam, S., Lee, J., Kim, B., Im, Y., Kwon, Y., Kyung, C. and Kang, K. (2000), Fast development of source-level debugging system using hardware emulation, Proceedings of the Asia and South Pacific-Design Automation, pp. 401-404

NASA (1995), Formal Methods Specification and Verification Guidebook fo Software and Computer Systems. Volume I - Planning and Technology Insertion.

Nexperia (2006), http://www.nxp.com/products/nexperia/

Ni, L. and McKinley, P., (1993) A Survey of Wormhole Routing Techniques in Direct Networks. IEEE Computer Magazine, v. 26, nr. 2, pp. 62-76.

OCP, (2003), Sonics Inc., http://www.ocpip.org/home

O'Nils, M. and Jantsch, A. (2001), Device Driver and DMA Controller Synthesis from HW/SW Communication Protocol Specification, Design Automation for Embedded System, vol. 6(2),Kluwer Academic Publishers.

Ost, L., Mello, A., Palma, J., Moraes, F. and Calazans, N. (2005), MAIA: A Framework for Networks on Chip Generation and Verification. Design Automation Conference, pp. 49-52.

Palesi, M., Holsmark, R., Kumar, S. and Catania V. (2006), A Methodology for Design of Application Specific Deadlock-free Routing Algorithms for NoC systems, Proceedings of the 4th International Conference on Hardware/Software Codesign and System Synthesis, pp. 142-147.

Pestana, S., Rijpkema, E., Radulescu,A., Goossens, K. and Gangwal, O. (2004), Cost-performance trade-offs in NoC: a simulation-based approach. In Proc. Design, Automation and Test in Europe Conference and Exhibition.

Pogge, H. (2002), The Next Chip Challenge: Effective Methods for Viable Mixed Technology SoC's, Proc. of the 39th ACM/IEEE Design Automation Conference, pp. 81-87.

Potter, B., Sinclair, J. and Till, D. (1996), An Introduction to Formal Specification and Z, Prentice Hall, 2nd. Ed.

Radulescu, A., Dielissen, J., Pestana, S., Gangwal, O., Rijpkema, E., Wielage, P. and Goossens, K. (2005), An Efficient on-chip NI Offering Guaranteed Services, Shared-Memory Abstraction, and Flexible Network Configuration. IEEE Trans. on CAD of Integrated Circuits and Systems 24(1): 4-17.

Ramkarthik, S. and Zhang, C. (2006), Generating Java Skeletal Code with Design Contracts from Specifications in a Subset of Object Z, International Conference on Computer and Information Science and International Workshop on Component-Based Software Engineering,Software Architecture and Reuse, pp. 405-411

Ramos, K., Santiago, N., Casillo, L., Augusto, A., Oliveira, J., Silva, I. (2004), Projeto Baseado em Reuso: Implementação de um IP de Processador didático em FPGA com interface OCP, Proceedings X Workshop IBERCHIP, pp. 11-20.

Ramos, K., Ribeiro, C., Moreira, A. and Silva, I. (2006a), A Formal Approach for Network on Chip Design, Proc. International Workshop on SoC and MCSoC Design, pp. 347-358.

Ramos, K., Ribeiro, C., Moreira, A. and Silva, I. (2006b), A Formal Approach for Network on Chip Design, In Ben A. Abderazek; Masahiro Sowa; Ismail K. Ibrahim; Laurence T. Yang, editors, (Submitted) Multicore Systems-on-Chips Book, pp. 1-20.

Rego, R. S. L. S. (2004), When Reconfigurable Architecture Meets Network-on-Chip, Proc. of the 17th Symposium on Integrated Circuits and System Design, pp. 216 – 221.

Rego, R. S. L. S. (2006), Projeto e Implementação de uma Plataforma MP-SoC usando SystemC. Dissertação (Mestrado em Sistemas e Computação) - Universidade Federal do Rio Grande do Norte.

Rijpkema, E., Goossens, K., Radulescu, A., Dielissen, J., van Meerbergen, J., Wielage, P. and Waterlander, E., (2003) Trade Offs in the Design of a Router with Both Guaranteed and Best-Effort Services for Networks on Chip, Proc. Design, Automation and Test in Europe Conference and Exhibition, pp. 350–355.

Rowson, J. and Sangiovanni-Vincentelli, A. (1997), Interface Based Design, Proc. Design Automation Conf., pp. 178–183.

Ruf, J., Drechsler, R. (2006), Formal – Verification – Today and in the Future Design, Automation and Test in Europe.

Rushby, J. (1993), Formal Methods and the Certification of Critical Systems. Technical Report CSL-93-7.

Saaltink, M. (1997a), The Z/EVES User's guide. Technical report, ORA Canadá, Ontário, Canadá. TR-97-5493-06.

Saaltink, M. (1997b), The Z/EVES system. In J. P. Bowen, M. G. Hinchey, and D. Till, editors, ZUM'97: Z Formal Specification Notation, vol. 1212 of Lecture Notes in Computer Science., pp. 72–85.

Sammane, G., Schmaltz, J., Toma, D., Ostier, P. and Borrione, D. (2004), TheoSim: combining symbolic simulation and theorem proving for hardware verification. 17th Symposium on Integrated Circuits and Systems Design.

Savage, W., Chilton, J. and Camposano, R. (2000), IP reuse in the system on a chip era, Proc. of the 13th international symposium on System synthesis.

Sawada, J. and Gamboa, R. (2002), Mechanical Verification of a Square Root Algorithm Using Taylor's Theorem. In Formal Methods in Computer-Aided Design, pp 274-292.

Sawada, J. and Hunt, W. (1998), Processor Verification with Precise Exceptions and Speculative Execution. In CAV 98, volume 1254 of LNCS, Springer.

Schmaltz, J. and Borrione, D. (2005), A Generic Network on Chip Model. Proceedings TPHOLs 2005, Springer LNCS 3603.

Schmaltz, J. and Borrione, D. (2004a), A Functional Approach to the Formal Specification of Networks on Chip. In. Proc. of Formal Methods in Computer-Aided Design. A. J. Hu and A. K. Martin (eds), LNCS 3312, Springer-Verlag, pp 52-66.

Schmaltz, J. and Borrione, D. (2004b), A Functional Specification and Validation Model for Networks on Chip the ACL2 Logic. In Proceedings of the 5th International Workshop on the ACL2 Theorem Prover and its Applications.

Sgroi, M., Sheets, M., Mihal, A., Keutzer, K., Malik, S, Rabaey, J. and Vincentelli, A. (2001), Addressing the System-on-a-Chip Interconnect Woes Through Communication-Based Design. In Proc. of the Design Automation Conference, pp. 667–672.

Shang, L., Peh, L., Kumar, A. and Jha, N. (2004), Thermal Modeling, Characterization and Management of On-chip Networks. In Intl. Symp. on Microarchitecture, pp. 67-78.

Siegmund, R. and Muller, D. (2003), Efficient Modeling and Synthesis of On-chip Communication Protocols for Network-on-chip Design", IEEE International Symposium on Circuits and Systems.

Siguenza-Tortosa, D. and Nurmi, J. (2002), Vhdl-based simulation environment for proteo noc. In Proc. High-Level Design Validation and Test Workshop, pp. 1-6.

Silva, J. I. S. (2004) Malha de Interconexão para Sistemas Integrados em Chip Único. Dissertação (Mestrado em Sistemas e Computação) - Universidade Federal do Rio Grande do Norte.

Skillicorn, D. and Tália, D. (1998), Models and Languages for Parallel Computation. ACM Computing Surveys, vol. 30, issue 2, pp 123 – 169.

Spivey, J. M. (1992), The Z Notation: A Reference Manual, Prentice Hall International, 2nd edition.

ST Nomadik (2006), http://www.st.com/stonline/

Synopsys (2006), CoCentric System Studio, http://www.synopsys.com/products/ designware/system_studio/system_studio.html

SystemC (2006), http://www.systemc.org

Tamir, Y., Frazier, G. (1992), Dinamically-Allocated Multi-Queue Buffers for VLSI Communication Switches. IEEE Trans. On Computers, v. 41, n.6, pp. 725-737.

TI Omap (2006), Texas Instruments, http://www.ti.com

Tsiopoulos, L. and Waldén, M. (2006), Formal Development of NoC Systems in B. TUCS Technical Report No. 751.

Utting, M., Toyn, I., Sun, J., Martin, A., Dong, J., Daley, N. and Currie, D. (2003), ZML: XML Support for Standard Z, Proc.in Third International Conference of B and Z Users, Book Series Lecture Notes in Computer Science, Vol. 2651/2003, pp. 437-456.

Utting, M. (2005), Jaza User Manual and Tutorial. The University of Waikato Hamilton, New Zealand.

Vahid, F. and Givargis, T. (2001), Platform Tuning for Embedded Systems Design, IEEE Computer, vol. 34, Issue 3, pp 112-114.

van der Tol, E. B. and Jaspers, E. G. T. (2002), Mapping of MPEG-4 decoding on a flexible architecture platform, SPIE 2002, pp. 1-13.

van Harmelen, F., Fensel, D. (1995), Formal Methods in Knowledge Engineering. The Knowledge Engineering Review, Vol. 10, No. 4, pp. 345-360.

Vassiliadis, S et al. (2006), Tera-Device Computing and Beyond, Disponível em: http://ec.europa.eu/information_society.

Vincentelli, A. S- and Martin, G. (2001), Platform-based Design and Software Design Methodology for Embedded Systems. IEEE Design & Test of Computers, vol. 18, pp. 23-33.

Vitkovski, A., Haukilahti, R., Jantsch, A. and Nilsson, E. (2004), Low-power and error coding for network-on-chip traffic. NorChip Conference.

Wagner, F., Cesário, W., Carro, L. and Jerraya, A. (2004), Strategies for the Integration of Hardware and Software IP Components in Embedded Systems-on-Chip, Integration, the VLSI Journal, vol. 37 , Issue 4.

Waldén, M. and K. Sere, K. (1998), Reasoning about action systems using the B-Method. In Formal Methods in System Design, Vol. 13, No 1, pp. 5 - 35. Kluwer Academic Publishers.

Wiklund, D. and Liu, D. (2003), Socbus: Switched Network on Chip for Hard Real Time Systems. Proc of the Int'l Parallel and Distributed Processing Symposium (IPDPS).

Wiklund, D., Sathe, S. and Liu, D. (2004), NoC Simulations for Benchmarking, In Proc. System-on-Chip for Real-Time Applications, pp. 269 – 274.

Walrand, J. and Varaiya, P. (1996), High-Performance Communication Networks, Morgan Kaufmann Publishers.

Wikipedia, The Free Encyclopedia (2006), Set-top Box, http://en.wikipedia.org/wiki/Set-top_box

Wikipedia, The Free Encyclopedia (2006), Ubiquitous Computing, http://en.wikipedia.org/ wiki/ Ubiquitous_computing.

Wolf, W. (2001), Computer as Components: principles of embedded computing system design, Morgan Kaufmann Publishers.

Wolf, W. (2004), The Future of Multiprocessor Systems-on-Chips. Design Automation Conference.

Woodcock, J. and Davies, J. (1996), Using Z: Specification, Refinement, and Proof, Prentice-Hall International Series in Computer Science.

Wu, J. (2002), A Deterministic Fault-tolerant and Deadlock-free Routing Protocol in 2-D Meshes Based on Odd-even Turn Model", Proceedings of the 16th international conference on Supercomputing, pp. 67-76.

Xilinx, (2006), http://www.xilinx.com

Ye, T., Benini, L. and De Micheli, G. (2002), Analysis of Power Consumption on Switch Fabrics in Network Routers, Proc. Design Automation Conference.

Ye, T., Benini, L. and De Micheli, G. (2003), Packetized on-chip interconnect Communicationan analysis for MPSoC, Proc. Design Automation and Test In Europe.

Ye, T., Benini, L. and De Micheli (2004), G. Packetization and routing analysis of on-chip multiprocessor networks**.** Journal of Systems Architecture: the EUROMICRO Journal,Volume 50 , Issue 2-3, pages: 81 – 104.

Zeferino, C. A. (2003a), Redes-em-Chip: Arquiteturas e Modelos para Avaliação de Área e Desempenho. Tese de Doutorado. Brasil: Universidade Federal do Rio Grande do Sul.

Zeferino, C. and Susin, A. (2003b), SoCIN: A Parametric and Scalable Network-on-Chip. In: 16th (INT.) Symposium on Integrated Circuits and Systems, pp. 169-174.

Zeferino, C., Kreutz, M. and Susin, A. (2004), RASoC: A Router Soft-Core for Networks-on-Chip. In: Int. Design Automation and Test in Europe, pp. 198-205.

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo