Pontifícia Universidade Católica do Rio Grande do Sul
Faculdade de Informática
Curso de Bacharelado em Ciência da Computação

Arquitetura Multiprocessada em SOCs: Estudo de Diferentes Topologias de Conexão

Proposta de Trabalho de Conclusão I

Autores:

Aline Vieira de Mello

Leandro Heleno Möller

Orientador:

Fernando Gehm Moraes

Porto Alegre, Agosto de 2002.

Índice

Índice de Figuras

Índice de Tabelas

1 Introdução

A maior parte do desenvolvimento de métodos e ferramentas para o projeto de sistemas digitais das últimas quatro décadas (60 a 90) derivaram de necessidades relacionadas com o projeto de computadores e sistemas periféricos. Contudo, a ênfase de pesquisa tem se deslocado gradativamente para um mercado muito mais vasto, o de sistemas eletrônicos que entram na composição de produtos de uso específico. Exemplos são automóveis, aeronaves, eletrodomésticos e dispositivos de comunicação pessoal tais como telefones celulares e *pagers.* Sistemas embarcados ou sistemas embutidos são para todos os efeitos, sistemas computacionais que executam uma função específica. Eles possuem a mesma estrutura geral de um computador, mas a especificidade de suas tarefas faz com que não sejam nem usados nem percebidos como um computador.

De acordo com vários autores [HAM97] [PAT97], em menos de 7 anos já existirão no mercado circuitos integrados compostos por mais de um bilhão de transistores. Com esta capacidade de integração, pode-se imaginar a inclusão de um sistema computacional completo em um único chip, o que cria o conceito de SOC (System On a Chip). A vantagem na utilização de SOCs está na interface de comunicação, pois nos sistemas atuais o maior gargalo é a perda de desempenho causada pela troca de informações entre o hardware e o software executados em CIs distintos. Caso os componentes de hardware e software estejam integrados em um único CI, o desempenho global do sistema tende a ser muito maior. Além do mais, a possibilidade de realizar um SOC pode reduzir o time-to-market e criar novas relações de desempenho entre o hardware e o software.

É neste contexto que entram em cena os núcleos de hardware (*Cores*), módulos de hardware pré-caracterizados, usados no desenvolvimento de sistemas computacionais integrados. Através do reuso destes componentes é possível atingir às exigências do mercado atual e reduzir a distância entre a quantidade de recursos disponíveis e a produtividade das equipes de projeto.

1.1 Processadores Embarcados

Os processadores embarcados estão presentes em todos os lugares. Uma estimativa realizada nos lares de classe média americanos indicou que existem, em média, 40 a 50 processadores em cada uma delas. Os processadores estão presentes em aparelhos de TV, videocassete, máquinas de lavar e secar, controles remotos, aparelhos de som, forno microondas, lavadora de louça, cafeteiras, geladeiras, videogames e muitos outros. Para se

ter uma idéia, os novos carros tem em média 12 processadores chegando a 65 em uma Mercedes, e os computadores pessoais contém aproximadamente 10 processadores.

Os processadores embarcados são componentes que permitem integrar em um mesmo dispositivo a parte software com a parte hardware do sistema, conduzindo a redução de custo e melhor desempenho. Estes processadores embarcados, como MIPS e ARM, não são mais utilizados apenas em ASICs. A evolução na tecnologia de semicondutores permite hoje integrar estes processadores em FPGA, e ao mesmo tempo disponibilizar milhares de portas lógicas para o usuário.

A família Excalibur, da Altera, disponibiliza dois processadores embarcados: NIOS e ARM. O NIOS é um *firm core*, que é integrado à lógica do usuário, descrita em VHDL ou Verilog. Já o processador ARM é disponibilizado na forma de um *hard core*, apresentando um desempenho superior ao NIOS.

A Figura 1 ilustra o dispositivo EPXA10, o qual contém um processador ARM 922T, de arquitetura RISC 32 bits, que opera a 200MHz. Além do processador ARM, cuja implementação pode ser observada no canto direito superior, há disponível para o usuário 256Kbytes de RAM porta simples, 128Kbytes de RAM dupla porta, 1 milhão de portas lógicas para implementar lógica do usuário e mais de 1000 pinos de entrada e saída.

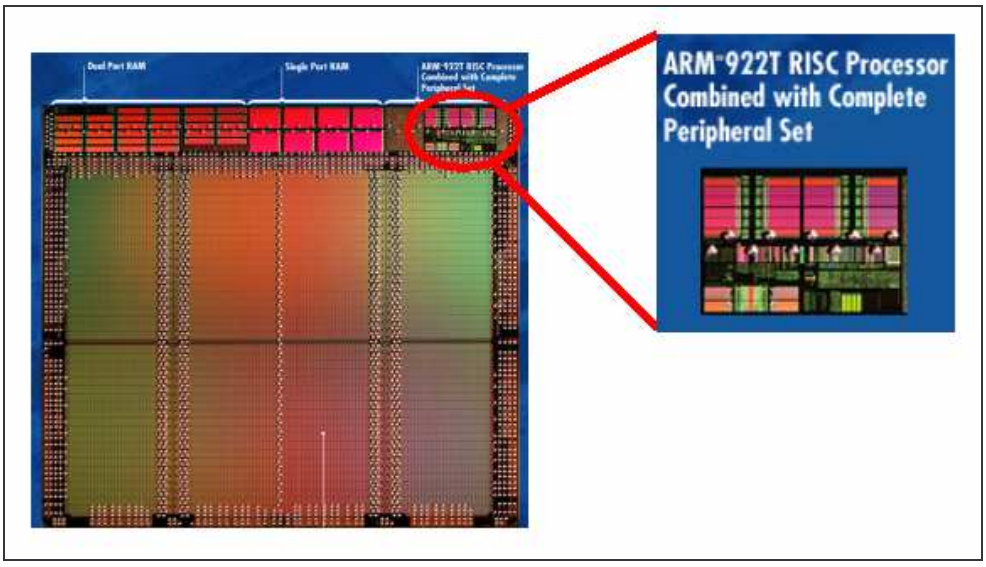

Figura 1 - Dispositivo EPXA10 com processador ARM 922T

O Nios é um processador RISC de propósito geral, configurável, e que pode ser combinado com a lógica do usuário em um dispositivo lógico programável da Altera. Algumas características do processador Nios são: (*i*) conjunto de instruções de 16 bits; (*ii*)

largura de dados com 16 ou 32 bits; (*iii*) execução de uma instrução por ciclo de clock; (*iv*) suporte para memória *on-chip* e *off-chip*; (*v*) desempenho superior à 50 MIPS.

A família Empower, da Xilinx, também disponibiliza dois processadores embarcados: MICROBLAZE (*firm core*) e POWERPC (*hard core*).

O Microblaze é um processador RISC de 32 bits que suporta uma largura de barramento de 32 ou 16 bits. Possui 32 registradores e segue a arquitetura Harvard. Suporta acesso a memória do próprio chip (BlockRAMs) ou memória externa.

Para o desenvolvimento deste trabalho, utilizaremos um processador embarcado simples, desenvolvido localmente. O objetivo do trabalho é explorar e mostrar que multiprocessamento é factível em FPGAs. Assim sendo, não há razão para utilizar processadores complexos, o que desviaria a atenção do trabalho para o problema de estudar e implementar tais processadores, e não a rede de conexão.

1.2 Multiprocessadores

Baseados nos conceitos descritos acima surge a idéia de um sistema multiprocessado, na qual cada núcleo de hardware pode ser um processador de propósito geral (General Purpose Processor - GPP). Teoricamente, a junção de *N* processadores pode conduzir a uma melhoria do desempenho em *N* vezes, atingido uma capacidade de processamento superior a qualquer sistema monoprocessado. São definidas duas classes gerais de multiprocessadores: memória compartilhada e troca de mensagem (ou passagem de mensagem) [HWA93] [KUM94] [PAT96], descritas a seguir:

1.2.1 Multiprocessadores de memória compartilhada

Nos multiprocessadores de memória compartilhada, todos os processadores compartilham um espaço de endereçamento global, o qual pode ser fisicamente centralizado ou distribuído. A sincronização e a comunicação entre os processos de uma mesma aplicação ocorre através do acesso a variáveis compartilhadas em memória. Em função de como o compartilhamento de memória é realizado, podem ser definidas as subclasses descritas a seguir.

1.2.1.1 Modelo UMA

No modelo UMA *(Uniform Memory Acces*s), os múltiplos processadores são ligados por meio de uma rede de interconexão a uma memória global centralizada. Essa memória

central é formada por módulos disjuntos, os quais podem ser acessados independentemente um do outro por diferentes processadores da máquina. Para reduzir o tráfego na rede, a cada processador pode ser associada uma memória local privativa para o armazenamento de dados locais e instruções de programa. Essas máquinas recebem esse nome porque o tempo de acesso à memória compartilhada é igual para todos os processadores.

1.2.1.2 Modelo NUMA

No modelo NUMA *(No*n-*Uniform Memory Acces*s), cada processador possui uma memória local, a qual é agregada ao espaço de endereçamento global da máquina. Dessa forma, podem existir até três padrões de acesso à memória compartilhada. O primeiro, e o mais rápido, é aquele onde a variável compartilhada está localizada na memória local do processador. O segundo padrão refere-se ao acesso a um endereço na memória central. Já o terceiro, e o mais lento, diz respeito ao acesso a uma posição localizada em uma memória local de outro processador. Dois modelos alternativos de máquina NUMA são mostrados na Figura 2.

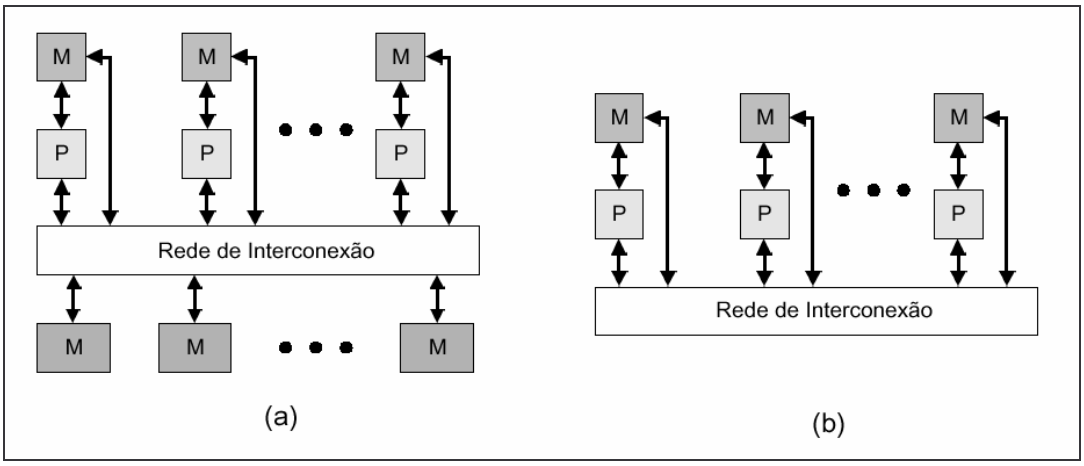

Figura 2 - Modelo NUMA: (a) com memória central; (b) sem memória central.

O modelo mostrado na Figura 2b é também chamado de multiprocessador de memória compartilhada distribuída (DSM - *Distributed Shared Memory*), pois toda a memória do sistema é distribuída entre os processadores da máquina, não havendo uma memória central.

1.2.1.3 Modelos COMA e CC-NUMA

As máquinas COMA *(Cache-Only Memory Access)* são um caso particular das

arquiteturas NUMA onde as memórias locais dos processadores são convertidas em caches. Todas as caches formam o espaço de endereçamento global e o acesso às caches remotas é auxiliado por meio de diretórios distribuídos. Já as máquinas CC-NUMA *(Cache-Coherent Non-Uniform Memory Acces*s) utilizam memória compartilhada distribuída e diretórios de cache, como, por exemplo, a máquina DASH de Stanford [LEN90]. Essas máquinas têm sido outra alternativa para a construção de máquinas escaláveis com memória compartilhada.

1.2.2 Multiprocessadores de troca de mensagem

Um multiprocessador de troca de mensagem é constituído por múltiplos processadores com memória local privativa e sem acesso à memória remota. Não existe compartilhamento de memória e a comunicação entre os processadores ocorre exclusivamente pela troca de mensagens através da rede de interconexão, normalmente através do uso de bibliotecas de comunicação como PVM ou MPI. Por esses motivos, essa arquitetura também recebe o nome de modelo NORMA *(NO-Remote Memory Acces*s), ou seja, sem acesso à memória remota. Um exemplo de arquitetura de troca de mensagem é ilustrado na Figura 3.

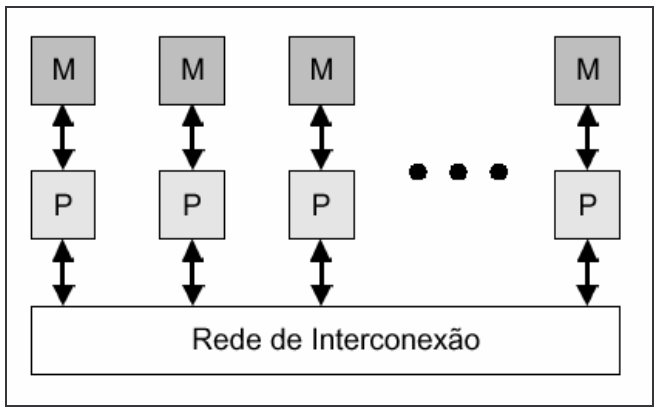

Figura 3 - Modelo NORMA.

1.3 Redes de Interconexão

A Figura 4 [MAD97] ilustra uma arquitetura genérica de um SOC. Os núcleos de hardware (*Core*) são integrados através de uma rede de interconexão comercial ou adaptada, com um controlador e funções de interface com o meio externo [PAL02]. Caso os núcleos de hardware sejam obtidos de diferentes fontes, a integração dos módulos e o teste do sistema podem ser difíceis, podendo até haver necessidade de que os núcleos de

hardware sejam reprojetados, para adequá-los a um protocolo de interface comum. Para que o núcleo de hardware não seja reprojetado é construído um módulo (w*rapper* - [BRA02] [IYE02] [KOR02]) que encapsula o núcleo de hardware e o conecta à rede de interconexão.

A forma mais usual de interconectar núcleos é através de um barramento. Esta forma de conexão é discutida na Seção 1.3.1. Devido aos problemas de escalabilidade e desempenho, observa-se que esta forma de conexão está migrando aos poucos para uma rede intra-chip, a qual denomina-se NOC (network-on-chip). NOCs são discutidas na Seção 1.3.2.

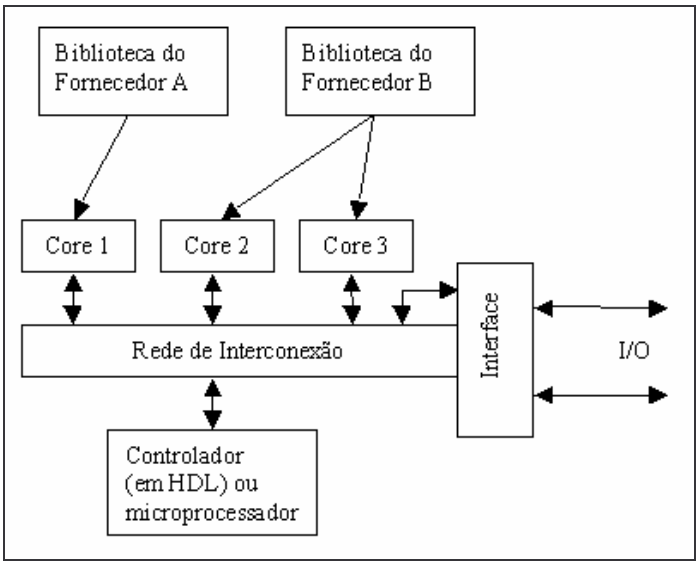

Figura 4 - Arquitetura SOC Genérica.

1.3.1 Barramento

A forma usual de interconexão entre os núcleos de hardware é a de barramento, havendo diversos padrões adotados pela indústria, como o *CoreConnect* [IBM99] da *IBM, AMBA* [ARM02] da *ARM* e *WISHBONE* [SIL02] da *Silicore*. Estas arquiteturas de barramento são geralmente vinculadas à arquitetura de um processador, tal como o *PowerPC* ou o *ARM* [BER00].

A Figura 5 [BER00] ilustra um SOC baseado na arquitetura de barramento *CoreConnect*.

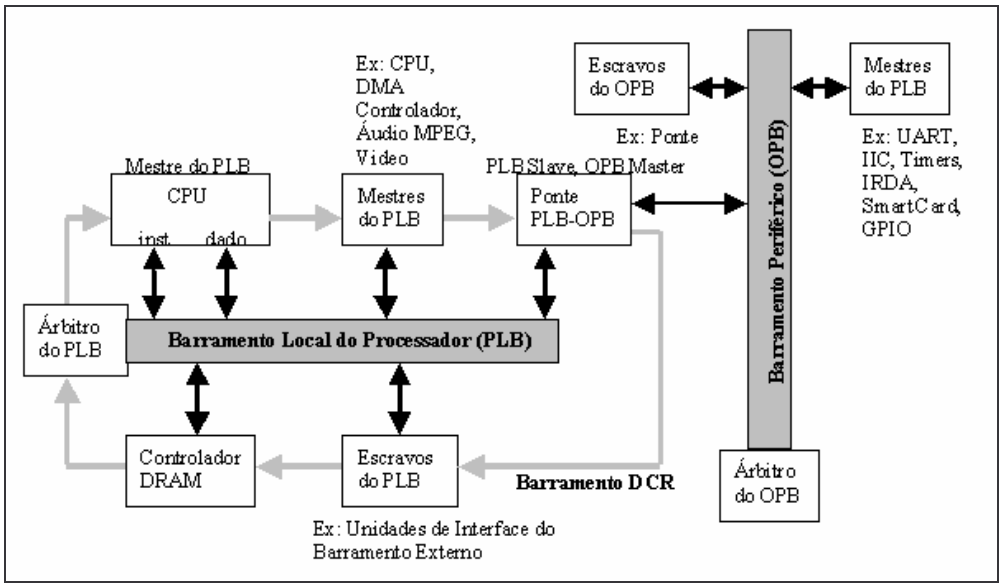

Figura 5 - SOC que usa a arquitetura de barramento *CoreConnect.*

A arquitetura *CoreConnect* da *IBM* fornece três barramentos para interconectar núcleos de hardware e lógica personalizável:

- Barramento Local do Processador (*Processor Local Bus - PLB*): usado para interconectar núcleos de hardware com alto desempenho, grande largura de banda, tais como o PowerPC, controladores DMA e interfaces de memória externa.
- Barramento Periférico (*On-Chip Peripheral Bus - OPB*): usado para interconectar periféricos que trabalham com baixas taxas de dados, tais como portas seriais, portas paralelas, UARTs (Universal Assynchronous Receiver Transmiter), e outros núcleos de hardware com pequena largura de banda.
- Barramento de Registradores de Controle de Dispositivos (*Device Control Register Bus - DCR*): caminho de baixa velocidade, usado para passar configuração e informações de estado entre o processador e outros núcleos de hardware.

Se comparado com a arquitetura SOC genérica (Figura 4), o PLB seria equivalente ao barramento de interconexão, enquanto que o OPB é responsável pela interface I/O.

Outra arquitetura que deve ser citada é a *WISHBONE*. Esta especificação pode ser utilizada para *soft*, *firm* e *hard cores*, já que *firm* e *hard cores* são geralmente concebidos a partir de *soft cores*. A especificação não requer o uso de ferramentas de desenvolvimento ou dispositivos-alvo (hardware) específicos [SIL02].

Os desenvolvedores do *WISHBONE* foram influenciados por três fatores principais. Primeiro, havia a necessidade de uma solução boa e confiável para a integração de núcleos

de hardware em SOCs. Segundo, havia a necessidade de uma especificação de interface comum para facilitar as metodologias de projeto estruturadas para grandes equipes de projeto. Terceiro, eles foram influenciados pelas soluções de integração de sistemas tradicionais, fornecidos por barramentos de microcomputador como o PCI, por exemplo.

De fato, a arquitetura do *WISHBONE* é análoga a um barramento de microcomputador, sendo que: (*i*) oferece uma solução flexível para integração que pode ser facilmente adaptada à uma aplicação específica; (*ii*) oferece uma variedade de ciclos de acesso ao barramento e de larguras de caminhos de dados para atender a diferentes sistemas; e (*iii*) permite que os núcleos de hardware sejam projetados por vários fornecedores.

Os projetistas do *WISHBONE* criaram uma especificação robusta o suficiente para assegurar a compatibilidade entre IP *cores* e, ao mesmo tempo, sem restringir a criatividade do projetista e do usuário final. Em Janeiro de 2001, a organização *OpenCores* adotou o *WISHBONE* como um padrão de conectividade entre seus núcleos de hardware. Ele foi escolhido por ser o único a atender os requisitos adotados por esta instituição. É um barramento flexível, simples, e o único completamente aberto atualmente, pois sua criadora, a empresa *Silicore Corporation*, tornou-o aberto ao domínio público [OPE01].

Ao contrário de arquiteturas como a *CoreConnect*, o *WISHBONE* tem uma estrutura simples, composta de um único barramento, como mostra a Figura 6. Um sistema com muitos componentes pode incluir duas interfaces *WISHBONE*: uma para blocos que exigem alto desempenho e outro para periféricos de baixo desempenho.

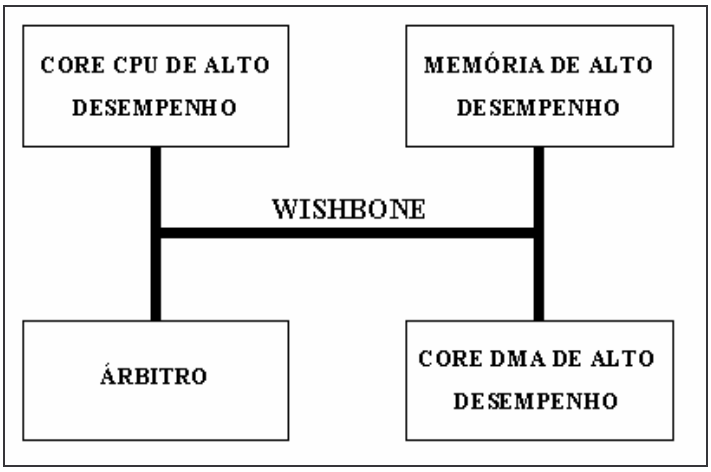

Figura 6 - Arquitetura *WISHBONE.*

A interconexão por barramento é simples, sob o ponto de vista de implementação,

apresentando entretanto diversas desvantagens [BEN02]: (*i*) apenas uma troca de dados pode ser realizada por vez, pois o meio físico é compartilhado por todos os núcleos de hardware, reduzindo o desempenho global do sistema; (*ii*) necessidade de mecanismos inteligentes de arbitragem do meio físico para evitar desperdício de largura de banda; (*iii*) a escalabilidade é limitada, ou seja, o número de núcleos de hardware que podem ser conectados ao barramento é muito baixo, tipicamente na ordem da dezena; (*iv*) o uso de linhas globais em um circuito integrado com tecnologia submicrônica impõe sérias restrições ao desempenho do sistema devido às altas capacitâncias e resistências parasitas inerentes aos longos fios. Estas desvantagens podem ser parcialmente contornadas através do uso de, por exemplo, hierarquia de barramentos, onde o problema continua existindo, sendo apenas minimizado.

1.3.2 Network On Chip - NOC

Uma forma de atacar os problemas dos barramentos é através da utilização de redes internas ao circuito integrado [DAL01] [WIN01], mecanismo denominado de NOC – *Network On Chip*. Estas redes herdam conceitos utilizados em telecomunicações, que é a organização da transferência de informação em camadas e protocolos [SGR01].

Uma rede qualquer é composta por nodos de processamento e nodos de chaveamento. Por exemplo, a rede em anel (Figura 7) é uma topologia bastante simples e econômica. Cada nodo de chaveamento possui ligações para dois nodos de chaveamento vizinhos e para um nodo de processamento local.

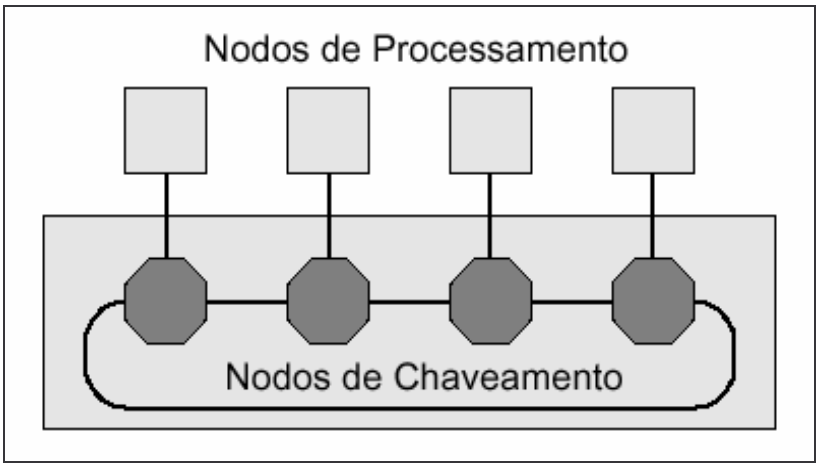

Figura 7 - Topologia em anel.

Os nodos de processamento (Figura 8a) são responsáveis pela execução das

subtarefas do algoritmo paralelo e possuem pelo menos um processador e uma interface para a rede de interconexão (chamada *interface de rede*), podendo ter, ainda, memória local, discos e outros periféricos.

Os nodos de chaveamento (Figura 8b) realizam a transferência de mensagens entre os nodos de processamento. Em geral, eles possuem um núcleo de chaveamento (ou chave), uma lógica para roteamento e arbitragem (abreviado por R&A) e portas de comunicação para outros nodos de chaveamento e, dependendo da topologia, para um nodo de processamento local. As portas de comunicação incluem canais de entrada e de saída, os quais podem possuir, ou não, *buffers* para o armazenamento temporário de informações. As portas possuem ainda um controlador de enlace para a implementação do protocolo físico de comunicação. Na Figura 8b, a porta de interface com o nodo de processamento não inclui controlador de enlace, pois, nesse caso, é sugerido que a ligação entre os nodos é implementada por meio de trilhas de circuito. Alternativamente, ela poderia ser realizada por meio de um enlace.

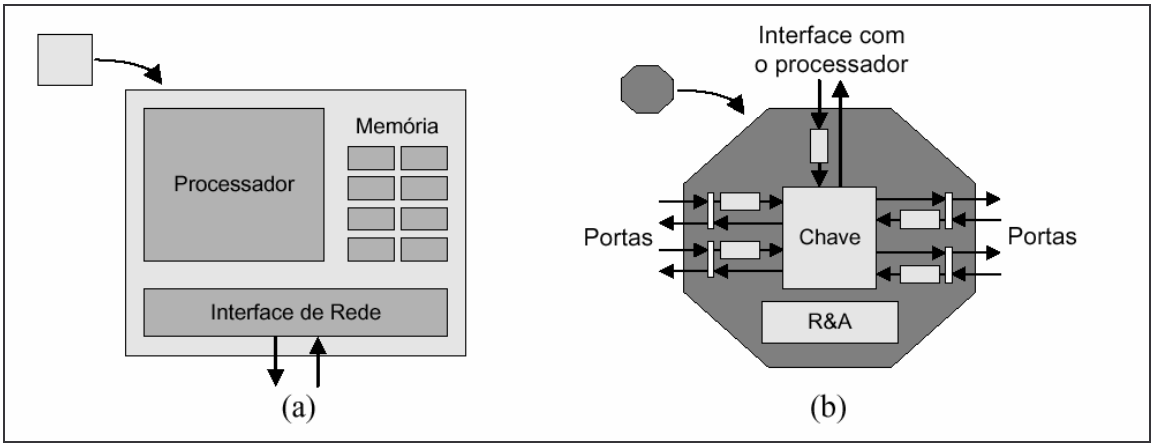

Figura 8 - Nodos: (a) de processamento; (b) de chaveamento.

A ligação física entre dois nodos de chaveamento é denominada enlace (*link*). Um enlace possui um ou dois canais físicos de comunicação e é implementado sob a forma de um cabo elétrico ou óptico. Dependendo da topologia da rede, um nodo de processamento também pode ser ligado a um nodo de chaveamento através de um enlace.

Em SOCs os enlaces são implementados através do roteamento entre os módulos. É neste quesito, roteamento, que as topologias NOC superam as topologias de barramento. Nas topologias NOC, as conexões são locais, entre módulos de chaveamento próximos, o que reduz o comprimento total de roteamento e por conseqüência aumenta o desempenho elétrico. Já nas topologias de barramento, as conexões são globais, o que acarreta perda de

desempenho devido aos fios longos.

As informações trocadas pelos nodos fonte e destino de uma comunicação são organizadas sob a forma de mensagens. Em geral, uma mensagem possui três partes: um cabeçalho (*header*), um corpo de dados (*payload*) e um terminador (*trailer*), sendo que o cabeçalho e o terminador formam um envelope ao redor do corpo de dados da mensagem. No cabeçalho são incluídas informações de roteamento e controle utilizadas pelos nodos de chaveamento para propagar a mensagem em direção ao nodo destino da comunicação. O terminador, por sua vez, inclui informações usadas para a detecção de erros e para a sinalização do fim da mensagem.

Tipicamente, as mensagens são quebradas em pacotes para transmissão. Um pacote é a menor unidade de informação que contém detalhes sobre o roteamento e seqüenciamento dos dados e mantém uma estrutura semelhante a de uma mensagem, com um cabeçalho, um corpo de dados e um terminador. Um pacote é constituído por uma seqüência de *phits*[1], cuja largura depende da largura física do canal.

Uma rede de interconexão pode ser caracterizada pela sua topologia e pelas estratégias utilizadas para roteamento, controle de fluxo, chaveamento e arbitragem que ela utiliza. Essas características são definidas brevemente abaixo:

- *Topologia*: é o arranjo dos nodos e canais sob a forma de um grafo.
- *Roteamento*: determina como uma mensagem escolhe um caminho dentro desse grafo.
- *Controle de fluxo*: lida com a alocação de canais e *buffers* para uma mensagem que atravessa esse grafo.
- *Chaveamento*: define como e quando um canal de entrada é conectado a um canal de saída selecionado pelo algoritmo de roteamento.
- *Arbitragem*: determina qual canal de entrada pode utilizar um determinado canal de saída.

As redes de interconexão para multiprocessadores adotam muitos dos conceitos empregados em redes de computadores. Por exemplo, muitas redes de interconexão são estruturadas em camadas que encapsulam funções equivalentes àquelas definidas para os níveis hierárquicos do modelo de referência OSI (*Open System Interconnection*), um padrão internacional de organização de redes de computadores proposto pela ISO (*International Organization for Standardization*) [DAY83]. O objetivo de uma estrutura de protocolo em níveis é delimitar e isolar funções de comunicações a camadas. Cada nível deve ser

[1] Phit (unidade de transferência de dados) representa a largura do canal de dados.

pensado como um programa ou processo, quer implementado por hardware ou software, que se comunica com o processo correspondente na outra máquina. As regras que governam a conversação de um nível "K" qualquer são chamadas de protocolo de nível "K". O modelo da ISO possui sete níveis de protocolos, como pode ser observado na Figura 9.

Figura 9 - Camadas do modelo OSI.

A arquitetura da rede é formada por níveis, interfaces e protocolos. Cada nível oferece um conjunto de serviços ao nível superior, usando funções realizadas no próprio nível e serviços disponíveis nos níveis inferiores. Os nodos de chaveamento de redes de interconexão que são estruturados em camadas hierárquicas implementam algumas das funções dos níveis inferiores (físico, enlace, rede) do modelo OSI, descritas abaixo:

- Nível físico: realiza a transferência de dados em nível de bits através de um enlace.
- Nível de enlace: efetua a comunicação em nível de quadros (grupos de bits). Se preocupa com o enquadramento dos dados e com a transferência desses quadros de forma confiável, realizando o tratamento de erros e o controle do fluxo de transferência de quadros.
- Nível de rede: faz a comunicação em nível de pacotes (grupos de quadros). Responsável pelo empacotamento das mensagens, roteamento dos pacotes entre a origem e o destino da mensagem, controle de congestionamento e contabilização de pacotes transferidos.

1.3.3 Topologias de Redes de Interconexão

Uma rede de interconexão pode ser caracterizada pela estrutura como seus nodos são

interligados. Essa estrutura é tipicamente representada por um grafo G(N,C) onde *N* representa o conjunto de nodos (de processamento e/ou de chaveamento) da rede e *C* representa o conjunto de canais de comunicação. Quanto à topologia, as redes de interconexão para multiprocessadores podem ser agrupadas em duas classes principais, as redes diretas e as redes indiretas.

Nas redes *diretas*, cada nodo de chaveamento possui um nodo de processamento associado, e esse par pode ser visto como um elemento único dentro da máquina, tipicamente referenciado pela palavra nodo, como ilustra a Figura 10. Pelo fato de utilizarem nodos de chaveamento tipo roteador, as redes diretas são também chamadas de redes baseadas em roteadores [DUA97]. Uma outra denominação utilizada é a de redes estáticas, pois as ligações entre os nodos não mudam durante a execução do programa paralelo [HWA93].

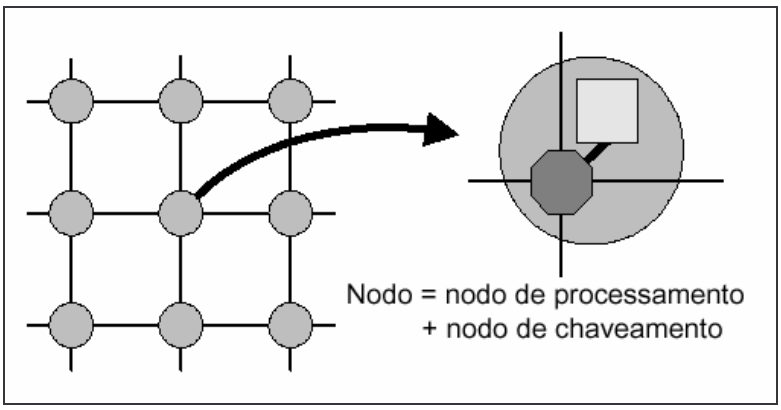

Figura 10 - Nodos de redes diretas.

As topologias de redes diretas estritamente ortogonais mais utilizadas são a grelha (ou malha) n-dimensional (Figura 11a), o toróide (Figura 11b) e o hipercubo (Figura 11c).

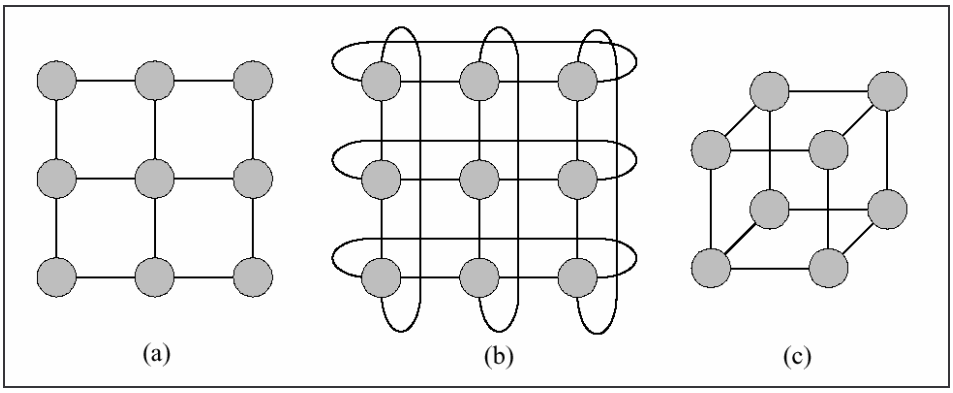

Figura 11 - (a) Grelha 2D 3x3; (b) Toróide 2D 3x3; (c) Hipercubo 3D.

Nas redes *indiretas*, o acoplamento entre os nodos de processamento e os nodos de chaveamento não ocorre no mesmo nível das redes diretas. Na visão unificada cada par nodo de processamento - nodo de chaveamento é visto como um elemento. Os nodos de processamento possuem uma interface para uma rede de nodos de chaveamento baseados em chaves. Cada chave possui um conjunto de portas bidirecionais para ligações com outras chaves e/ou com os nodos de processamento. Somente algumas chaves possuem conexões para nodos de processamento e apenas essas podem servir de fonte ou destino de uma mensagem. A topologia da rede é definida pela estrutura de interconexão dessas chaves.

Duas topologia clássicas de redes indiretas se destacam: o crossbar e as redes multiestágio. Para conexão indireta de N nodos de processamento, o crossbar (Figura 12) é a topologia ideal, pois consiste de uma única chave $N \times N$. Embora seja mais econômico que uma rede direta completamente conectada (a qual necessitaria de N roteadores, cada um com um crossbar $N \times N$ interno), o crossbar possui uma complexidade da ordem de N^2, o que torna o seu custo proibitivo para redes grandes.

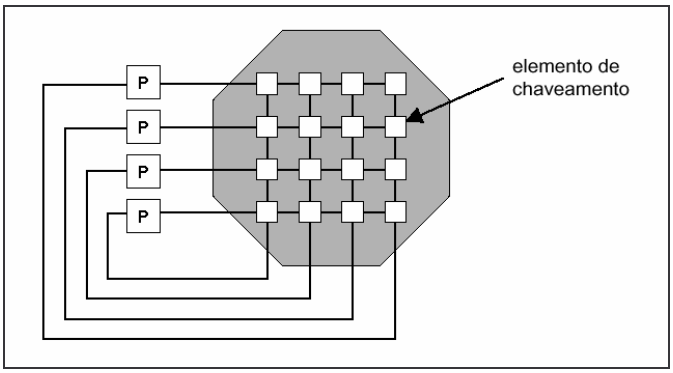

Figura 12 - Crossbar 4 x 4.

Além das redes vistas acima, existem inúmeras outras topologias de redes diretas e indiretas propostas com objetivos específicos, como por exemplo, minimizar o diâmetro da rede para um determinado número de nodos e grau do nodo [FEN81] [HWA93] [CUL98]. Entre essas topologias, podem ser citadas o cubo conectado por ciclos (Figura 13a), a árvore, a árvore gorda (Figura 13b), a estrela, a rede banyan, a banyan-hipercúbica, a pirâmide e a rede De Brujin, entre outras.

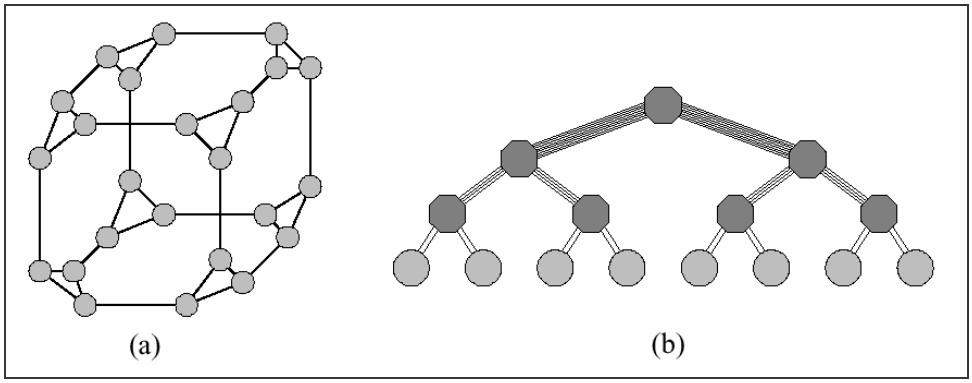

Figura 13 - (a) Cubo conectado por ciclos 3D; (b) Rede indireta em árvore gorda.

Esta proposta de trabalho de conclusão está organizada da seguinte forma. A seção 2 apresenta os objetivos do trabalho. A seção 3 contém a descrição da proposta assim como os tópicos a serem estudados para a realização do trabalho. A seção 4 apresenta o cronograma de atividades do grupo. Os recursos necessários são descritos na seção 5. Finalmente, na seção 6 as referências bibliográficas desta proposta de trabalho de conclusão são apresentadas.

2 Objetivos e Resultados Esperados

O primeiro objetivo deste trabalho é estudar a possibilidade de implementar um sistema multiprocessado em dispositivos programáveis, com conexão por barramento. Estes processadores embarcados podem tanto operar em paralelo para resolver um problema complexo, quanto operarem independentemente. A motivação para estudar este tipo de sistema deve-se ao fato que tradicionalmente multiprocessamento resulta em ganho de desempenho. Entretanto, estes sistemas são implementados com circuitos discretos. Implementando-se todo o sistema em único dispositivo pode-se obter um incremento de desempenho, dada a proximidade física dos componentes, e a possibilidade de explorar mecanismos dedicados de comunicação, os quais não são viáveis de se implementar em sistemas discretos.

O segundo objetivo do trabalho é explorar outras formas de comunicação entre processadores (ou núcleos de hardware), que não seja a de barramento. Estas redes genéricas, implementadas em um SOC, são denominadas de NOCs.

Ao final deste trabalho de conclusão espera-se:

1. Ter um sistema biprocessado implementado em hardware, com comunicação por barramento, operando com esquema de memória NUMA/NORMA (definição em função do decorrer do trabalho).
2. Ter um sistema com ***n*** processadores validados funcionalmente e se possível em hardware, com comunicação por barramento.
3. Implementação em hardware de um nodo de chaveamento.
4. Implementação e validação de pelo menos uma das topologias de rede de interconexão utilizando o nodo de chaveamento.
5. Integração da rede de interconexão a um conjunto de processadores, com validação funcional, e se possível em hardware.

Espera-se atingir os objetivos 1 e 2 ao final do Trabalho de Conclusão 1 e os objetivos 3 a 5 ao final do Trabalho de Conclusão 2.

3 Descrição da Proposta

Este trabalho será desenvolvido sobre a plataforma XSV800 da fabricante XESS, que contém o FPGA (Field-Programmable Gate Array) XCV800 fabricado pela Xilinx [XIL02]. Este dispositivo têm o equivalente a oitocentas mil portas lógicas e permite a inserção e remoção de núcleos de hardware em tempo de execução (reconfiguração parcial e dinâmica). Este FPGA também é dotado de 28 blocos de RAM (BlockRAMs) de 4096 bits cada. A plataforma contém dois displays de 7 segmentos, comunicação serial, comunicação pela porta paralela, entrada para teclado, saída para monitor VGA, push buttons e switch buttons para a configuração de parâmetros. O principal motivo da seleção desta plataforma foi a existência deste recurso no GAPH (Grupo de Apoio ao Projeto de Hardware).

Este trabalho visa, primeiramente, implementar um sistema em hardware (Figura 14) capaz de comunicar o hospedeiro às BlockRAMs através da porta serial. Este módulo de hardware para controle é composto pelo barramento e pela política de arbitragem. Exemplo de funcionamento esperado para esta etapa é:

(1) o hospedeiro envia para a plataforma comandos pela serial, utilizando o software desenvolvido;

(2) estes comandos são recebidos pelo módulo serial, e o *wrapper* da serial em função da decodificação do comando faz uma requisição ao árbitro. Exemplo de comando: “escreve na memória 2000 palavras”;

(3) o árbitro julga a requisição, liberando o acesso ao barramento pelo *wrapper* da serial. O módulo serial escreve os dados provenientes do hospedeiro na memória, enquanto tiver acesso ao barramento.

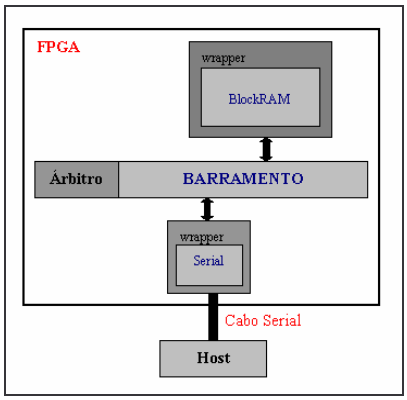

Figura 14 - Topologia de Barramento

Para obtenção do sistema em hardware descrito, deve-se:

- desenvolver um software que envie o código objeto a ser executado pelo processador para a plataforma e depois receba respostas de suas computações;
- desenvolver um núcleo de hardware que receba o código objeto da serial e envie resposta(s) para o hospedeiro;
- desenvolver um núcleo de hardware que escreva nas BlockRAMs;
- definir as prioridades de acesso ao barramento (política de arbitragem do barramento) e definir prioridades de requisição de interrupção do processador;
- interconectar os pinos de acordo com suas prioridades;
- definir quais são os clocks do sistema e conectar os clocks corretos a cada núcleo de hardware, bem como a lógica de controle de clock apropriada;
- definir as entradas e saídas do circuito integrado, incluindo nestes os pinos necessários para a metodologia de teste que será empregada;
- definir o módulo para comunicação entre os núcleos de hardware e o barramento (w*rapper*).

A etapa seguinte conectará um processador ao barramento, formando um sistema monoprocessado (Figura 15a). O processador alvo para esta implementação é o R8. O processador R8 é uma organização Von Neumann (memória de dados/instruções unificada), load/store, com CPI entre 3 e 4, barramento de dados e endereços 16 bits. Esta arquitetura é praticamente uma máquina RISC. Optou-se pelo uso deste processador por sua simplicidade, por poder ser alterado caso haja a necessidade, pelo conhecimento prévio proporcionado pela disciplina de Organização de Computadores ministrada no curso de Ciência da Computação e o baixo consumo de área do FPGA.

O processador é descrito na linguagem VHDL e passa por um processo de síntese lógica, síntese física (posicionamento e roteamento), análise temporal e finaliza pela construção de um arquivo que configura o hardware (*bitstream*). Este *bitstream* é enviado para a plataforma por um programa específico que utiliza a interface serial ou paralela do computador.

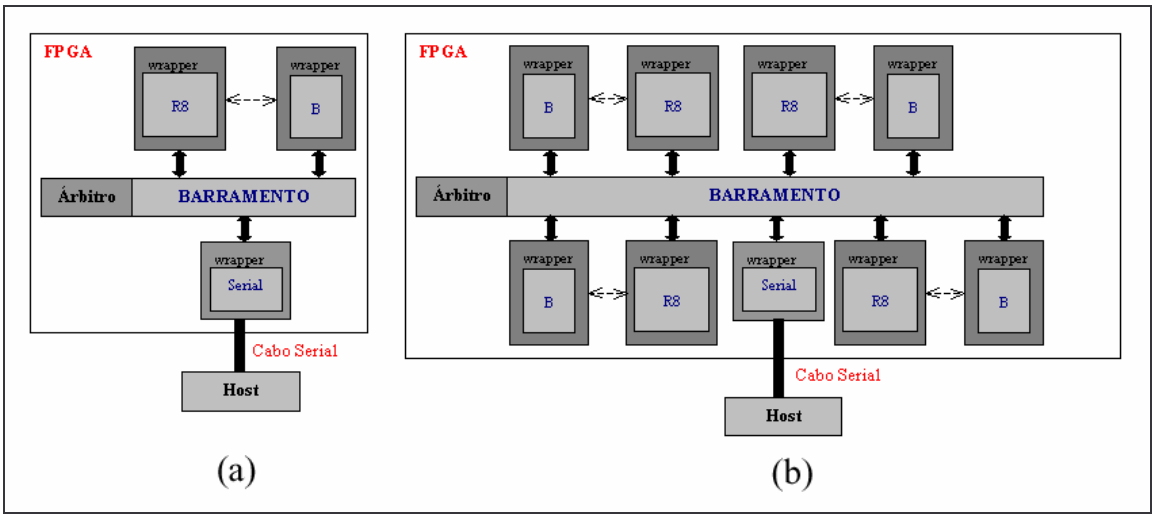

Figura 15 - Sistema: (a) monoprocessado; (b) multiprocessado.

É importante observar na Figura 15 que existe uma comunicação entre o processador e a memória. Esta comunicação é opcional e será decidida se existirá ou não no decorrer do trabalho. Sua presença é para permitir um acesso rápido à memória pelo processador, fazendo com que o módulo de memória atue como uma *cache*.

A próxima etapa é estabelecer a conexão entre ***n*** processadores (Figura 15b) e realizar a análise de desempenho obtido em comparação com as implementações anteriores.

Após coletados os dados referentes às interconexões com a topologia de barramento, será iniciado o estudo sobre Network on Chip (NOC). Este estudo será concluído com a implementação e simulação de pelo menos uma topologia de NOC, por exemplo como apresentada na Figura 16. Durante a fase de implementação será desenvolvido o nodo de chaveamento, que é composto por um núcleo de chaveamento, uma lógica para roteamento e arbitragem e portas de comunicação para outros nodos de chaveamento. A implementação das portas de comunicação incluem canais de entrada e de saída, os quais podem possuir, ou não, *buffers* para o armazenamento temporário de informações. As portas possuem ainda um controlador de enlace para a implementação do protocolo físico de comunicação. A implementação do enlace, que possuirá um ou dois canais físicos de comunicação, a definição do formato das mensagens e pacotes e a integração de ***n*** nodos de processamento à rede de interconexão, também fazem parte desta fase. Na fase de simulação pretende-se através da execução de programas paralelos avaliar o desempenho e identificar os gargalos da topologia implementada.

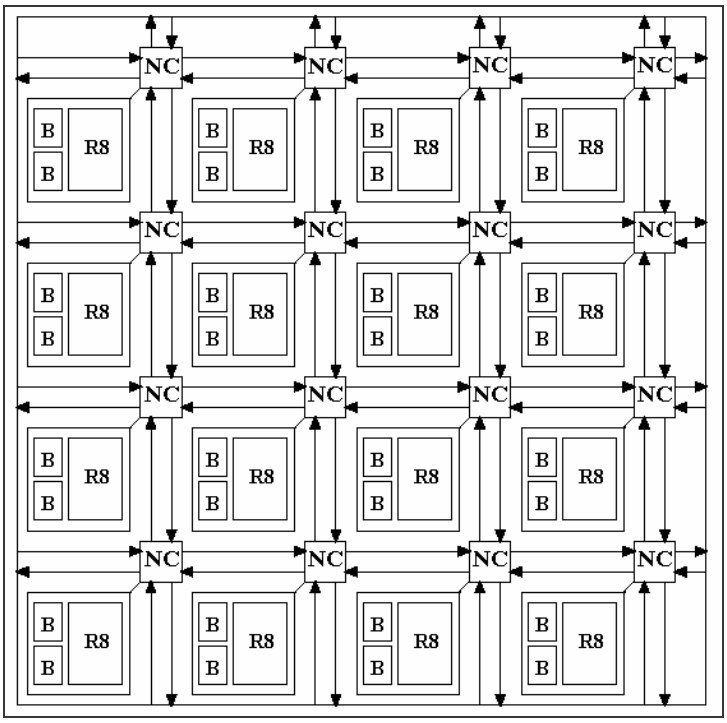

Figura 16 - Exemplo de NOC com 16 nodos de processamento e chaveamento.

O trabalho concluir-se-á com a avaliação das topologias e a apresentação dos resultados obtidos.

4 Cronograma de Atividades

1. *Estudos de técnicas de interconexão de núcleos de hardware através de barramento.*
   Nesta atividade pretende-se aprofundar os conhecimentos das técnicas de interconexão de núcleos, focando a topologia de barramento.
2. *Implementação de um software em Java para comunicação com a porta serial.*
   Nesta tarefa teremos um microcomputador atuando como hospedeiro e uma plataforma de comunicação contendo o dispositivo programável, memória e recursos de entrada e saída. A forma de comunicação entre o hospedeiro e a plataforma é através de uma comunicação serial, padrão RS-232.
   Esta tarefa tem por objetivo construir uma interface em Java para realizar a comunicação entre o hospedeiro e a plataforma. Exemplos de operações realizadas por esta interface: envio dos programas em código objeto para os módulos de memória do FPGA (BlockRAM), envio/recepção de dados e instruções, monitoramento do funcionamento do sistema, etc.
3. *Implementação de um módulo em VHDL para comunicação com a serial.*
   Relacionado à atividade anterior existe a necessidade de um módulo em hardware que estabeleça a comunicação entre o hospedeiro e o dispositivo através da serial. Este módulo será descrito em VHDL e tem a função: receber os dados vindos do hospedeiro e disponibilizá-los ao dispositivo; e enviar os dados do dispositivo para o hospedeiro.
4. *Início da implementação do barramento/árbitro.*
   Relacionado às atividade 2 e 3 existe a necessidade de um módulo que interprete os comandos enviados pela serial e escreva na devida BlockRAM.
   Nesta atividade será iniciada a implementação do barramento que conectará os processadores. A Figura 14 ilustra esta topologia. O sistema será composto por 4 módulos: bloco de memória, controle da serial, árbitro e barramento. A serial receberá comandos do hospedeiro para ler/escrever na memória. O acesso de leitura/escrita será autorizado pelo árbitro. Os módulos 'memória' e 'serial' são vistos como núcleos. A conexão destes implicará no desenvolvimento de *wrappers* que os conectem ao barramento.

5. *Estudo da arquitetura R8.*

   Este estudo tem como objetivo aprimorar os conhecimentos da arquitetura e do funcionamento do processador R8. O aprimoramento possibilitará a conexão de seus pinos a outros módulos e as futuras modificações.

6. *Integração do processador ao barramento.*

   Nesta atividade será acrescentado o processador na estrutura de barramento apresentada na Figura 15a. Nesta atividade, além de desenvolver o *wrapper* para o processador, deve-se decidir como será efetuado o acesso à memória de programa/dados. Ao final desta atividade espera-se ter o sistema validado por simulação funcional.

7. *Prototipação do sistema monoprocessado.*

   Nesta etapa é realizado a conexão dos pinos do sistema ao mundo externo e o processo de síntese lógica, síntese física, análise temporal e geração do bitstream.

   Ao final desta etapa espera-se ter um sistema monoprocessado sendo executado no FPGA, com uma estrutura de arbitragem centralizada (o processador não será o árbitro), escalável, sendo a interface com o mundo externo realizada por um hospedeiro.

8. *Seleção de algoritmos paralelos.*

   A seleção de algoritmos paralelos é necessária para testar a comunicação entre dois ou mais processadores. O critério de seleção será baseado no ganho de desempenho, apresentado pelo algoritmo, em máquinas multiprocessadas.

9. *Conexão e simulação de dois processadores com comunicação através de barramento.*

   Esta atividade será responsável por conectar mais um processador ao barramento e simular a execução dos algoritmos paralelos anteriormente selecionados.

10. *Prototipação do sistema biprocessado com comunicação através de barramento.*

    A prototipação do sistema biprocessado será semelhante ao do sistema monoprocessado, apresentado na atividade 7. Esta atividade será estendida nos meses de dezembro e janeiro.

11. *Escrita do Volume de TC1.*

    Esta atividade será desenvolvida ao longo do trabalho de conclusão 1.

12. *Conexão e simulação de **n** processadores com comunicação através de barramento.*

    A conexão e simulação de ***n*** processadores será semelhante a atividade 9.

13. *Prototipação de sistema com **n** processadores com comunicação através de barramento.*

    A prototipação de sistema com ***n*** processadores será semelhante a atividade 10.

14. *Avaliação do ganho de desempenho e identificação dos gargalos na arquitetura.*

    Esta etapa utilizará simulações desenvolvidas na atividade 12 para avaliar o desempenho e identificar os gargalos na arquitetura. Esta etapa, eventualmente, apresentará avaliações a nível físico.

15. *Estudo aprofundado de redes de interconexão em um SOC.*

    O estudo de NOCs será desenvolvido paralelamente a outras atividades para prover o embasamento teórico as mesmas.

16. *Implementação e validação funcional de um nodo de chaveamento.*

    Esta atividade será responsável pela implementação e validação funcional de um nodo de chaveamento com todos os seus componentes (núcleo de chaveamento, portas de comunicação e lógica para roteamento e arbitragem).

17. *Implementação e validação funcional de ao menos uma topologia de interconexão de núcleos usando o nodo de chaveamento.*

    Escolher uma topologia de interconexão dos nodos de chaveamento e realizar sua implementação será a atividade que validará a NOC.

18. *Integração dos processadores à rede de interconexão.*

    Esta atividade consiste na integração dos processadores à NOC desenvolvida na atividade anterior.

19. *Execução de programas paralelos sobre a rede implementada.*

    A execução de programas paralelos tem como objetivo avaliar o desempenho da rede com um caso real.

20. *Análise de resultados.*

    A atividade final consiste em analisar os resultados obtidos com as topologias implementadas ao longo do trabalho.

21. *Escrita do Volume final de TC2.*

    A escrita do volume será desenvolvida ao longo do trabalho de conclusão 2.

A Figura 17 ilustra a dependência entre as atividades previstas para o presente trabalho de conclusão. As atividades serão executadas por ambos os autores.

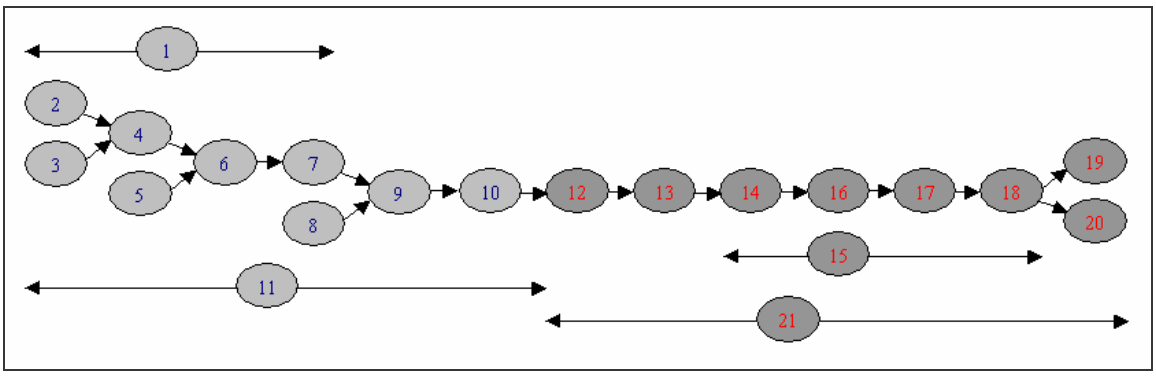

Figura 17 - Relação entre as atividades do trabalho de conclusão.

O cronograma das atividades é definido pela tabela abaixo:

	TC1				TC2					
Tarefas	**Ago**	**Set**	**Out**	**Nov**	**Dez**	**Jan**	**Mar**	**Abr**	**Mai**	**Jun**
1										
2										
3										
4										
5										
6										
7										
8										
9										
10										
11										
12										
13										
14										
15										
16										
17										
18										
19										
20										
21										

Tabela 1 – Cronograma das atividades previstas no trabalho de conclusão.

5 Recursos Necessários

Todos os recursos necessários existem no GAPH (Grupo de Apoio ao Projeto de Hardware), sala 106.34.

5.1 Recursos de Hardware

- Computadores Pentium 1GHz (compatível ou superior)
- Estação de trabalho
- Plataforma de prototipação XSV800

5.2 Recursos de Software

- Sistema Operacional Solaris/Sun e Windows NT
- ISE 4.1
- Active-HDL 5.1
- JDK 1.2.2
- Jbits 2.7 (ou superior)
- Microsoft Visual C++ 6.0
- TextPad 4
- Servidor Web Apache
- Internet Explorer 5.5 (ou superior)
- Microsoft Word

5.3 Fontes de Pesquisa

- Biblioteca Central
- Biblioteca do IPCT
- Internet (acesso ao portal de períodos CAPES)

A geração do código objeto para os processadores, assim como a simulação dos programas, será feita mediante o emprego das ferramentas de montagem/simulações desenvolvidas pela proponente desta proposta, Aline Vieira de Mello, durante seu trabalho como bolsista de Iniciação Científica [MOR01].

Será também considerada a opção do uso de ferramentas para reconfiguração parcial dos blocos de memória do FPGA, para carregar os programas em código objeto diretamente no FPGA, sem utilizar desta forma a interface serial. Estas ferramentas para reconfiguração foram desenvolvidas pelo proponente desta proposta, Leandro Heleno Möller, durante seu trabalho como bolsista de Iniciação Científica [MES01].

6 Referências Bibliográficas

[ARM02] ARM *"AMBA 2.0 Specification"*. Capturado em: http://www.arm.com/armtech/AMBA_Spec (Jul. 2002).

[BEN02] Benini, L. De Micheli,G. *"Networks on chips: a new SOC paradigm"*. Computer, Volume: 35(1), Jan. 2002, pp. 70-78.

[BER00] Bergamaschi, R. A.; Lee, W. R. *"Designing Systems-on-Chip Using Cores"*. In: Design Automation Conference, Los Angeles, California, USA, 2000, pp. 420-425.

[BRA02] Braun, F.; Lockwood, J.; Waldvogel, M. *"Protocol wrappers for layered network packet processing in reconfigurable hardware"*. IEEE Micro, Volume: 22(1), Jan.-Feb. 2002, pp. 66-74.

[CUL98] Culler, D.; Singh, J. P. *"Parallel Computer Architecture: a Hardware Software Approach"*. Los Altos, California: Morgan Kaufmann, 1998, 1100 p.

[DAL01] Dally, W.J.; Towles, B. *"Route packets, not wires: on-chip interconnection networks"*. In: Design Automation Conference, 2001, pp. 684-689.

[DAY83] Day, J.D.; Zimmermman, H. *"The OSI reference model"*. Proceedings of IEEE, Volume: 71, Dec. 1983, pp. 1334-1340.

[DUA97] Duato, J. et al. *"Interconnection Networks"*. Los Alamitos, California: IEEE Computer Society Press, 1997, 515 p.

[FEN81] Feng, T.-Y. *"A Survey of Interconnection Networks"*. IEEE Computer, Volume: 14, Dec. 1981, pp. 12-27.

[HAM97] Hammond, L.; Nayfeh, B. A.; Olukotun, K. "*A Single-Chip Multiprocessor*". IEEE Computer, Volume: 30(9), Sep. 1997, pp. 79-85.

[HWA93] Hwang, K. "Advanced Computer Architecture : Parallelism, Scalability, Programmability". New York : McGraw-Hill, 1993, pp. 213-256.

[IBM99] IBM, *"The RS/6000 SP High Performance Communication Network.IBM"*. Capturado em: http://www.rs6000.ibm.com/resource/technology/sp_sw1/spswp1.book_1.html (Aug. 1999).

[IYE02] Iyengar, V.; Chakrabarty, K.; Marinissen, E.J. *"Efficient wrapper/TAM co-optimization for large SOCs"*. In: Design, Automation and Test in Europe, 2002, pp. 491-498.

[KOR02] Koranne, S. *"Design of reconfigurable access wrappers for embedded core based SOC test"*. In: International Symposium on Quality Electronic Design, 2002, pp. 106-111.

[KUM94] Kumar, V. et al. "*Introduction to Parallel Computing: Design and Analysis of Algorithms*". Redowood City, California : Benjamin/Cummings, 1994, 580 p.

[LEN90] Lenoski, D. et al. "The Directory-Based Cache Coherence Protocol for the DASH Multiprocessor". In: ACM ISCA, 17, 1990, pp. 148-159.

[MAD97] Madisetti, V. K., Shen L. "*Interface Design for Core-Based Systems*". IEEE Design & Test of Computers, Oct.-Dec. 1997, pp. 42-51.

[MES01] Mesquita, D.; Moraes, F.; Palma, J.; Möller, L.; Calazans, N. "*Reconfiguração Parcial e Remota de Cores FPGAs"*. In: IBERCHIP, 2001.

[MOR01] Moraes, F.; Mello, A.; Calazans, N. "*Ambiente de Desenvolvimento de Processador Embarcado para Aplicações de Codesign"*. In: SCR'2001 - Seminário de Computação Reconfigurável. 2001.

[OPE01] Opencores.org. *“Wishbone SoC Interconnection”*. Capturado em: http://www.opencores.com/press/pr_8jan2001.shtml (Aug. 2001).

[PAL02] Palma, J.C.S. "*Métodos de Distribuição e Conexão de IC Cores para Dispositivos Programáveis FPGA"*. Dissertação de Mestrado, Programa de Pós-Graduação em Ciência da Computação, PUCRS, 2002, 108 p.

[PAT96] Patterson, D.; Hennessy, J. L. “*Computer Architecture: A Quantitative Approach*”. San Francisco, California : Morgan Kaufmann, 1996, 760 p.

[PAT97] Patt, Y. N., Patel, S. J., Evers, M., Friendly, D. H., Stark, J. “*One Billion Transistors, One Uniprocessors, One Chip”*. IEEE Computer, Volume: 30 (9), Sep. 1997, pp. 51-57.

[SGR01] Sgroi, M.; Sheets, M.; Mihal, A.; Keutzer, K.; Malik, S.; Rabaey, J.; Sangiovanni-Vincentelli, A. *"Addressing the system-on-a-chip interconnect woes through communication-based design"*. In: Design Automation Conference, 2001, pp. 667-672.

[SIL02] Silicore. *"The WISHBONE Service Center"*. Capturado em http://www.silicore.net/wishbone.htm/ (Jul. 2002).

[WIN01] Wingard, D. *"MicroNetwork-based integration for SOCs"*. In: Design Automation Conference, 2001, pp. 673-677

[XIL02] Xilinx. *"Virtex-II Platform FPGA Overview"*. Capturado em: http://www.xilinx.com (Jul. 2002).