PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO GRANDE DO SUL

FACULDADE DE INFORMÁTICA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

ARQUITETURAS RECONFIGURÁVEIS DE GRÃO GRANDE E PROCESSADORES RECONFIGURÁVEIS

Trabalho Individual I

Mestrando: Leandro Heleno Möller

Orientador: Prof. Fernando Gehm Moraes

Porto Alegre, Maio de 2004

Sumário

Lista de Figuras

Listas de Abreviaturas

ASIC	*Application Specific Integrated Circuit*
ASIP	*Application Specific Instruction-Set Processor*
CAD	*Computer Aided Design*
DCT	*Discrete Cosine Transform*
DMA	*Direct Memory Access*
DPU	*Datapath Unit*
DSP	*Digital Signal Processor*
FPGA	*Field Programmable Gate Array*
IP	*Intellectual Property*
LUT	*Look-Up Table*
PE	*Processing Element*
RAM	*Random Access Memory*
RAP	*Reconfigurable Arithmetic Processing Unit*
RDPA	*Reconfigurable Datapath Architecture*
RDPU	*Reconfigurable Datapath Unit*
RFU	*Reconfigurable Functional Unit*
RISP	*Reconfigurable Instruction-Set Processor*
RPU	*Reconfigurable Processing Unit*
SOC	*System On Chip*
NOC	*Network On Chip*
ULA	Unidade Lógica e Aritmética

1 INTRODUÇÃO

Sistemas embarcados possuem como pré-requisitos baixo consumo de energia e elevado poder de processamento. Circuitos Integrados para Aplicações Específicas (*Application Specific Integrated Circuits* - ASICs) atendem a esta demanda, implementando um hardware dedicado e otimizado para alcançar estes pré-requisitos. Contudo, avanços tecnológicos acrescentam novas exigências, como flexibilidade, dificilmente atendidas por ASICs. Um exemplo de dispositivo que permite modificação do hardware após a fabricação é o FPGA. Um benefício para usuários de FPGAs é poder configurar com maior grau de liberdade o produto, permitindo até estendê-lo para outros usos. Os fabricantes também se beneficiam desta tecnologia por permitir a fácil atualização dos produtos após os mesmos serem vendidos. Além destes, o meio ambiente é favorecido pela redução do número de equipamentos eletrônicos com componentes nocivos à natureza jogados fora.

Essa flexibilidade em termos de hardware pode ser explorada sob outros aspectos além do aumento do ciclo de vida dos produtos, como por exemplo, a miniaturização dos mesmos. Já que o hardware é passível de ser alterado, o próprio sistema pode garantir que apenas os recursos em uso sejam mantidos em hardware durante o funcionamento do produto. O resultado é um produto que reutiliza o hardware para diferentes tarefas, reduzindo assim o tamanho do produto final e o consumo de energia. Além disso, o produto pode ser tolerante a falhas, sendo capaz de detectar certos defeitos de hardware e modificar a si mesmo de forma a contornar o problema e manter-se em funcionamento.

Entretanto, a tendência não é escolher entre ASICs ou FPGAs, mas que coexistam ambas tecnologias em um mesmo sistema embarcado para melhor atingir requisitos de poder de processamento, área, consumo de potência e flexibilidade. Combinar estas tecnologias permite acelerar computações antes executadas apenas em software, fazendo-se uso de hardware reconfigurável.

Todavia, existem algumas dificuldades para que FPGAs tornem-se amplamente utilizados. A primeira delas é o alto consumo de potência de FPGAs quando comparados com ASICs, conforme relatado por Becker et al. [BEC03]. Uma segunda dificuldade é prover um mecanismo de comunicação entre áreas reconfiguráveis em arquiteturas atuais. Outra dificuldade é o gerenciamento de substituições de áreas reconfiguráveis do sistema.

O objetivo deste trabalho é investigar as arquiteturas reconfiguráveis de grão grande, por estas serem mais indicadas para a reconfiguração de núcleos de Propriedade Intelectual (*Intellectual Property* - IP). Também é propósito deste trabalho realizar um estudo de processadores reconfiguráveis, de forma a viabilizar a implementação de sistemas que se transformam em tempo de execução para melhor resolver uma gama específica de problemas. A longo prazo, almeja-se a implementação de processadores reconfiguráveis conectados a partir de uma rede de comunicação intra-chip (*Network On Chip* - NOC), seguindo a tendência "*sea-of-processors*" [HEN03].

Com o intuito de situar o trabalho sobre arquiteturas reconfiguráveis de grão grande, a seção 1.1 classifica os diferentes tipos de sistemas digitais reconfiguráveis, enquanto a seção 1.2 provê uma breve introdução sobre topologias de redes de interconexão. A seção 1.3 apresenta as motivações deste trabalho. Após a apresentação do estado da arte em arquiteturas reconfiguráveis de grão grande (capítulo 2) e processadores reconfiguráveis (capítulo 3), o capítulo 4 apresenta as

conclusões e trabalhos futuros.

1.1 Classificação de Arquiteturas Reconfiguráveis

Após o surgimento dos primeiros sistemas configuráveis [BER89], diversas arquiteturas com diferentes características foram criadas, sendo hoje difícil escolher um dispositivo frente a tantas opções. Por isso faz-se necessária a organização de tais dispositivos em grupos, de forma a melhor analisar os prós e os contras perante um problema a ser resolvido. A taxonomia de Olimpo [RAD98] permite classificar os diferentes tipos de sistemas reconfiguráveis a partir de uma estrutura em árvore (Figura 1), onde cada nível aborda uma característica da arquitetura.

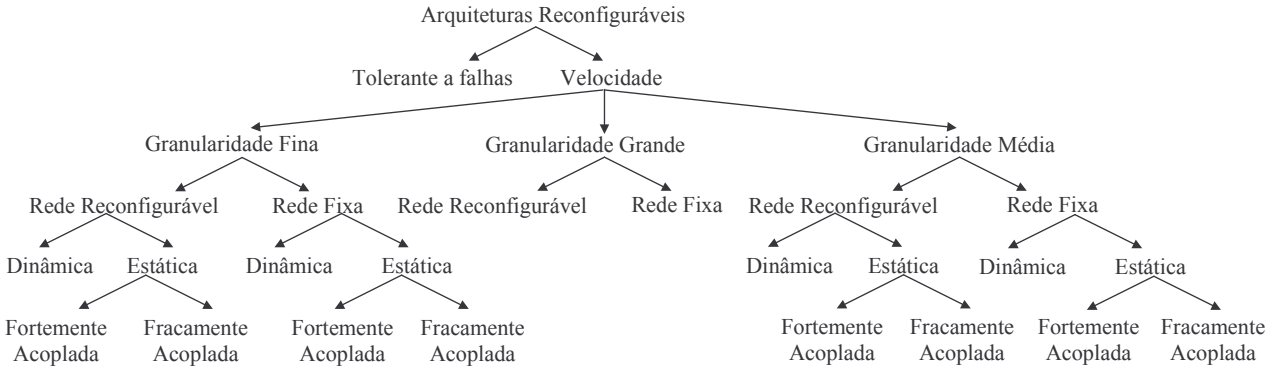

Figura 1 – Representação gráfica do critério de Olimpo.

A primeira característica é referente ao objetivo da arquitetura reconfigurável, que freqüentemente buscam um ganho de desempenho e **velocidade** na implementação de algoritmos. Entretanto, a arquitetura também pode ser voltada para **sistemas tolerantes** à falha, as quais possuem a capacidade de detectar um defeito de hardware e se auto-corrigir.

A segunda característica da arquitetura reconfigurável é a granularidade de suas células lógicas, ou seja, é o menor bloco reconfigurável que compõe a arquitetura. São chamadas de arquiteturas reconfiguráveis de **grão fino** arquiteturas que possuem como menor bloco reconfigurável *flip-flops* e *look-up tables* (LUTs), que são capazes de implementar somadores e funções lógicas. Arquiteturas de **granularidade média** possuem como menor bloco reconfigurável ULAs, que são freqüentemente utilizadas para acelerar computações com cálculos intensivos. Arquiteturas de **grão grande** normalmente contêm processadores, DSPs, memórias ou hardware dedicado como elementos de processamento interconectados da melhor forma a resolver um problema específico.

A terceira característica é em relação à flexibilidade da rede de interconexão dos elementos de processamento. Uma arquitetura com **rede fixa** é geralmente mais barata, simples e se adapta bem a computações intensivas. Uma arquitetura com **rede reconfigurável** permite modificar a interconexão entre os elementos de processamento, provendo uma maior escalabilidade da arquitetura.

A quarta característica é quanto à configuração do sistema. Arquiteturas **dinâmicas** não são controladas por um dispositivo externo, elas são ditas sistemas envolventes ou bio-inspiradas, pois não existe uma lista de configurações possíveis, mas sim sucessivas gerações de novas configurações em tempo de execução. Por outro lado, arquiteturas **estáticas** possuem um dispositivo externo que controla a configuração.

As arquiteturas estáticas podem ainda ser subdivididas conforme a proximidade a um processador principal. Arquiteturas estáticas **fracamente acopladas** são conectadas a um processador como um co-processador, o que pode tanto ocorrer em uma mesma plataforma ou diferentes plataformas. Arquiteturas estáticas **fortemente acopladas** atuam como unidades de execução do processador, manipulando dados diretamente de seus registradores.

1.2 Topologias de Redes de Interconexão

Na seção anterior a taxonomia de Olimpo buscou classificar os diferentes tipos de arquiteturas reconfiguráveis. Uma das características abordadas foi a granularidade dos blocos lógicos das arquiteturas. Esta seção tem por objetivo prover uma breve introdução sobre como os blocos lógicos podem ser interconectados. Para isso faz-se necessário o estudo de topologias de redes de interconexão, que tiveram origem nas redes de computadores.

A forma como elementos de uma rede são ligados entre si é dada pela rede de interconexão [DER02]. Alguns exemplos de topologias de redes de interconexão são apresentados na Figura 2. Estas redes podem ser agrupadas em duas classes principais, as redes estáticas e as redes dinâmicas.

Nas **redes estáticas,** um nodo qualquer possui uma conexão fixa dedicada e direta a outro(s) da rede. Por este motivo estas redes são também chamadas de ponto-a-ponto, porque se o nodo que recebeu a mensagem não for o destinatário da mesma ele é encarregado de passá-la adiante. São exemplos de redes estáticas as topologias apresentadas na Figura 2(a-n).

Nas **redes dinâmicas** não existe uma topologia fixa que defina o padrão de comunicação entre os elementos. A conexão é estabelecida em tempo de execução e a rede de interconexão adapta-se dinamicamente para permitir a transferência de dados. São exemplos de redes dinâmicas as topologias apresentadas na Figura 2(o-t).

Os principais critérios para a avaliação de uma topologia são quantas ligações diretas cada elemento possui (**grau do nó**), a maior distância entre dois elementos quaisquer da rede (**diâmetro**), o fato da conexão entre dois pontos **bloquear** o estabelecimento de outra conexão entre quaisquer outros dois pontos e o custo da rede, que cresce em função do número de ligações e de sua capacidade de transferência (**vazão** e **latência**).

Uma das redes mais utilizadas são os barramentos (Figura 2o-p) por serem de fácil implementação e baixo custo. Entretanto, este tipo de topologia apresenta diversas desvantagens [BEN02]: (*i*) apenas uma troca de dados pode ser realizada por vez, pois o meio físico é compartilhado por todos os elementos, 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 elementos que podem ser conectados ao barramento é muito baixo, tipicamente na ordem de dezena.

Em contraponto, podem ser citadas as matrizes (Figura 2q-r) que permitem a conexão direta entre quaisquer dois elementos que não estejam já se comunicando. No entanto, o seu custo cresce de forma quadrática em relação ao número de elementos interligados, muitas vezes tornando o seu custo proibitivo para redes grandes.

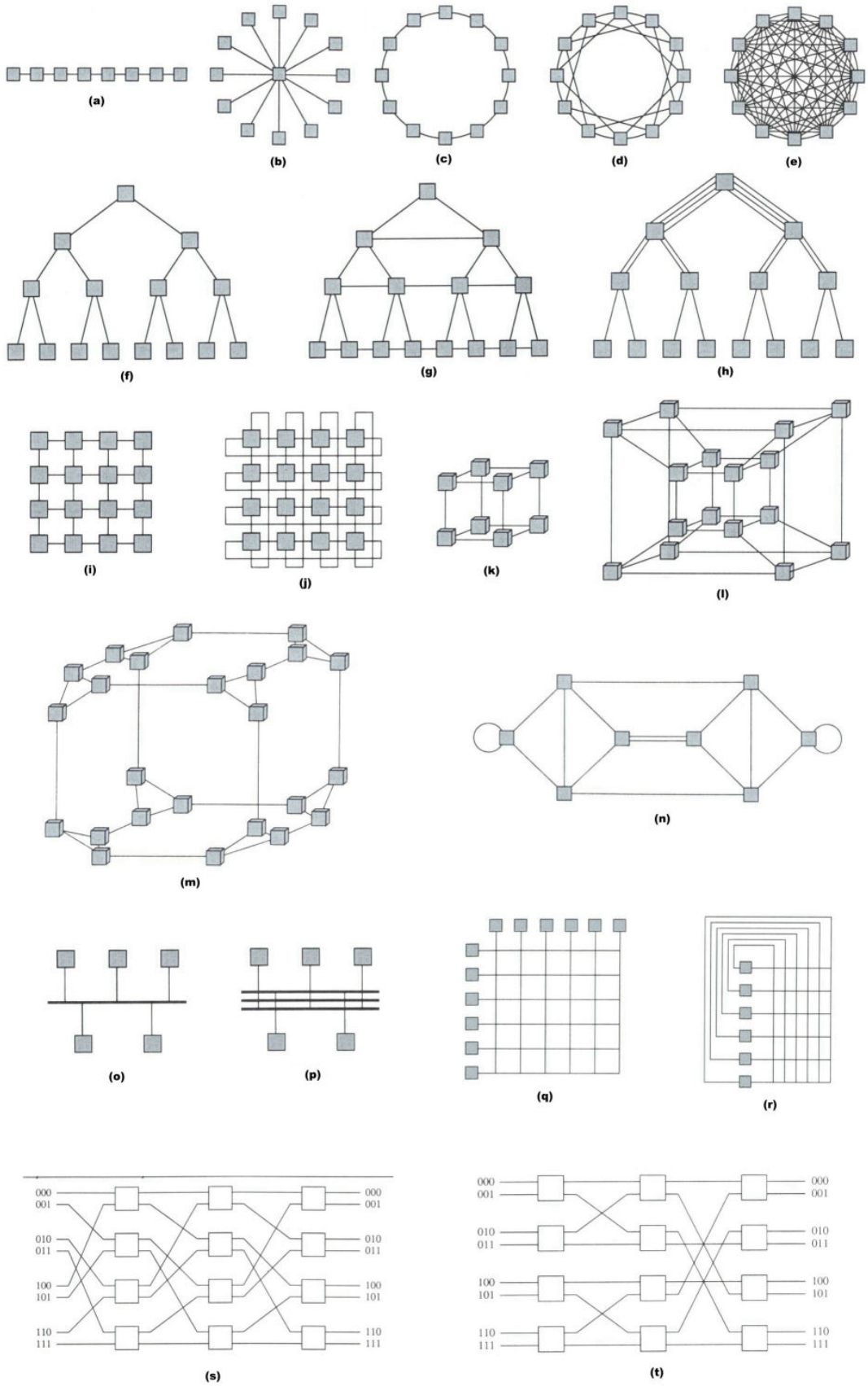

Figura 2 – Exemplos de topologias de redes de interconexão estáticas (a-n) e dinâmicas (o-t): (a) array linear; (b) estrela; (c) anel; (d) anel chordal; (e) totalmente conectada; (f) árvore binária; (g) x-tree; (h) árvore fat-tree; (i) malha ou grelha; (j) torus; (k) hipercubo de grau 3; (l) hipercubo de grau 4; (m) ccc – cube connected cycles; (n) grafo de DeBrujn; (o)barramento único; (p) barramento múltiplo; (q) matriz unilateral; (r) matriz bilateral (*crossbar*); (s) rede Omega; (t) rede Banyan.

1.3 Motivações

Com a evolução dos computadores pessoais, pessoas de diferentes áreas de estudo se dedicaram a utilizar a programação para criar seus próprios programas de computador de caráter pessoal ou profissional. A computação configurável abre novas fronteiras para a evolução da programação. Equipamentos eletrônicos podem ser criados e testados por usuários que não necessariamente são especialistas em hardware, podendo ter apenas conhecimento de linguagem programação, como por exemplo C++ ou SystemC. Entretanto, em pouco tempo a computação configurável estará sendo utilizada sem ao menos ser percebida em diversos equipamentos, pois apresentam flexibilidade e desempenho, que são características muito pleiteadas na tecnologia atual.

Este trabalho direciona o estudo para dispositivos reconfiguráveis de grão grande, porque dentre os classificados na seção 1.1, estes apresentam um menor número de bits para configurar um sistema, assim apresentando um menor tempo de configuração e uma maior eficiência quando comparado com módulos implementados em arquiteturas de grão menores.

Nesta pesquisa também estão sendo focados processadores reconfiguráveis, pois a tendência é que os processadores devem ser reconfigurados para aceitar novas instruções que sejam implementadas em hardware, assim aumentando sua flexibilidade e seu desempenho, ao mesmo tempo em que a área do núcleo do processador permanece inalterada.

Unindo processadores reconfiguráveis a NOCs e utilizando arquiteturas reconfiguráveis de grão grande, que é a tecnologia com maior propensão a melhores resultados em termos de velocidade de reconfiguração e desempenho, uma boa contribuição pode ser extraída do presente trabalho.

2 DISPOSITIVOS RECONFIGURÁVEIS DE GRÃO GRANDE

Arquiteturas reconfiguráveis de grão fino, como caracterizado na seção 1.1, possuem como menor bloco reconfigurável *flip-flops* e LUTs. Estas arquiteturas reconfiguráveis se popularizaram com os FPGAs, tornando-se naturais candidatas para a prototipação de sistemas digitais por sua flexibilidade. No entanto, projetos de sistemas digitais atuais tratam operações com palavras de dezenas de bits, tornando arquiteturas de grão fino ineficientes em termos de área, roteamento, consumo de energia e desempenho.

Arquiteturas reconfiguráveis de grão grande possuem unidades funcionais em nível de palavra e menos chaves de roteamento interconectando elementos de processamento (*Processing Element* - PE). Isto simplifica o roteamento e reduz o atraso entre os PEs, aumentando o desempenho da arquitetura. Além disso, chaves consomem mais energia e área que conexões diretas. O processo de síntese e reconfiguração também são mais rápidos em dispositivos de grão grande, pois a complexidade é reduzida devido a um menor número de chaves de roteamento e elementos reconfiguráveis.

Este capítulo apresenta o estado da arte dispositivos reconfiguráveis de grão grande, fazendo ao final um resumo das arquiteturas apresentadas.

2.1 KressArray-I

O KressArray-I [HAR95], publicado por Rainer Kress et al. da Universidade de Kaiserslautern, pode ser considerado como a primeira arquitetura de grão grande reconfigurável, pois suas Unidades de Blocos de Dados (*Datapath Unit* - DPU) manipulam dados de 32 bits. O conjunto de DPUs do KressArray-I é chamado de Arquitetura de Blocos de Dados Reconfiguráveis (*Reconfigurable Datapath Architecture* - RDPA), possuindo topologia em malha e DPUs com dois registradores de entrada e dois registradores de saída, conforme apresentado na Figura 3. As operações são dirigidas a dados, ou seja, elas são avaliadas logo que todos os operandos estão disponíveis, utilizando para isso um protocolo semelhante a um *handshake* para a sincronização de dados entre DPUs.

O KressArray-I provê barramentos locais e globais. O barramento local interconecta os vizinhos e é implementado como uma malha unidirecional para reduzir gastos em área. O barramento global interconecta cada DPU da arquitetura, sendo utilizado para entrada e saída do KressArray-I e para propagação de resultados intermediários de DPUs distantes. Outra alternativa de projeto empregada para reduzir área foi a multiplexação das interconexões globais, passando assim a tarefa de gerência para um escalonador externo. O escalonamento do acesso a dados é feito pelo ambiente de software, que calcula o tempo de execução de toda a aplicação e escalona os dados buscando atingir o melhor desempenho.

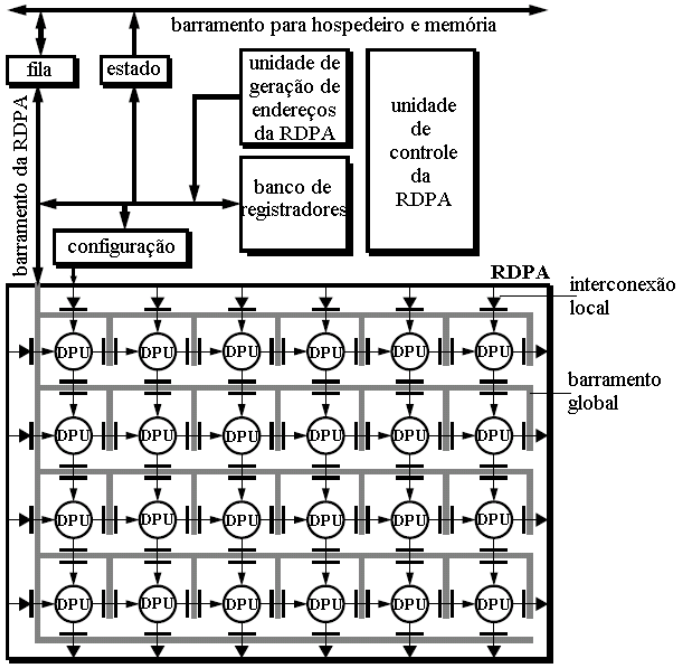

Figura 3 – Arquitetura do KressArray-I

2.2 RaPiD

O RaPiD (*Reconfigurable Pipeline Datapaths*) [EBE96] foi proposto por Carl Ebeling et al., da Universidade de Washington. A arquitetura é constituída de um array de unidades de execução, configurados para funcionar linearmente como um *pipeline*. A Figura 4 mostra a estrutura da célula utilizada na primeira versão do RaPiD. A célula compreende um multiplicador de inteiros, três ULAs, seis registradores de propósito geral e três memórias locais. O RaPiD contém de 8 a 32 dessas células.

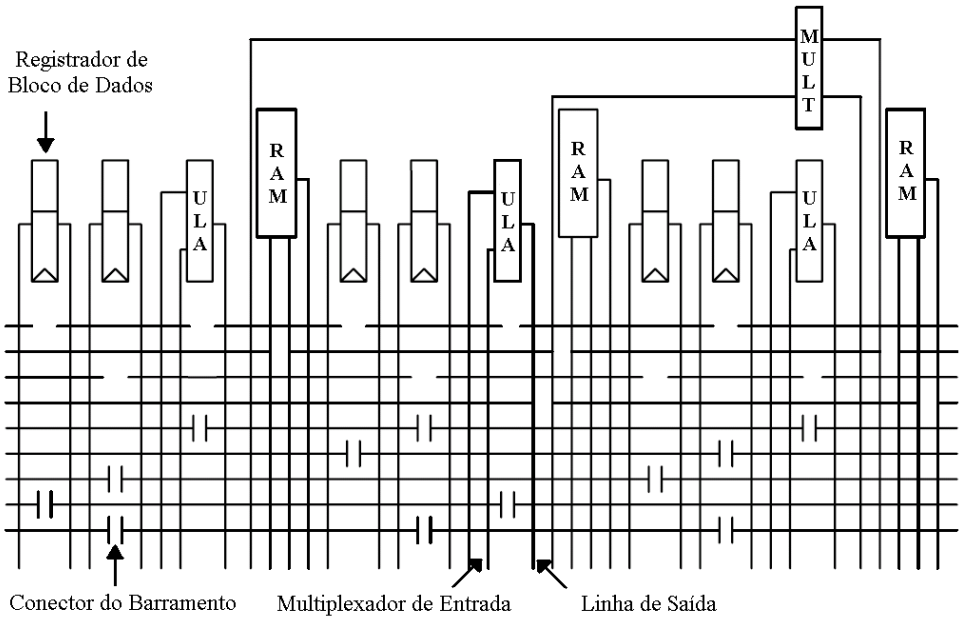

Figura 4 – Célula básica do RaPiD.

As unidades de execução são interconectadas usando um conjunto de barramentos que percorrem todo o array. As unidades de execução utilizam um *multiplexador* N:1 para selecionar os dados de entrada de N-2 barramentos em linhas adjacentes. As duas entradas adicionais provêm

linhas de retorno (para utilizar a ULA como acumulador) ou a constante zero (para limpar ou manter o valor dos registradores). As saídas das unidades de execução conectam-se aos barramentos através de *tristates*.

Os barramentos em diferentes linhas são segmentados em diferentes comprimentos para implementar uma conexão otimizada entre os diferentes recursos disponíveis. Em algumas linhas, os segmentos adjacentes podem ser interconectados com um conector de barramento para estender a linha até outras unidades de execução.

O RaPiD é a primeira abordagem que integra memória às células reconfiguráveis, o que implica em concorrência de acessos à mesma. A integração dos registradores do *pipeline* ao barramento e as múltiplas portas para o mundo externo colaboram para alta taxa de transferência de dados na arquitetura.

2.3 MATRIX

A arquitetura MATRIX (*Multiple ALU Architecture wiTh Reconfigurable Interconnect eXperiment*) [MIR96] foi proposta por Ethan Mirsky e André DeHon do MIT. Ela possui topologia em malha que conecta Unidades Básicas Funcionais (*Basic Functional Unit* - BFU) de 8 bits. Cada BFU (Figura 5) contém 256 palavras de 8 bits de memória, uma ULA de 8 bits e uma unidade de multiplicação e uma unidade de controle implementada através de PLA.

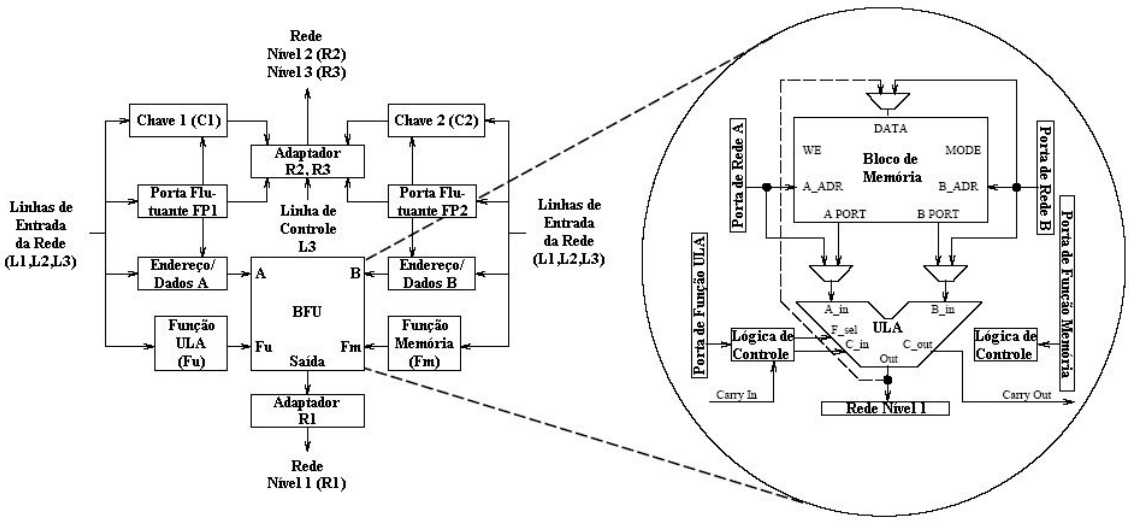

Figura 5 – Arquitetura da Unidade Básica Funcional.

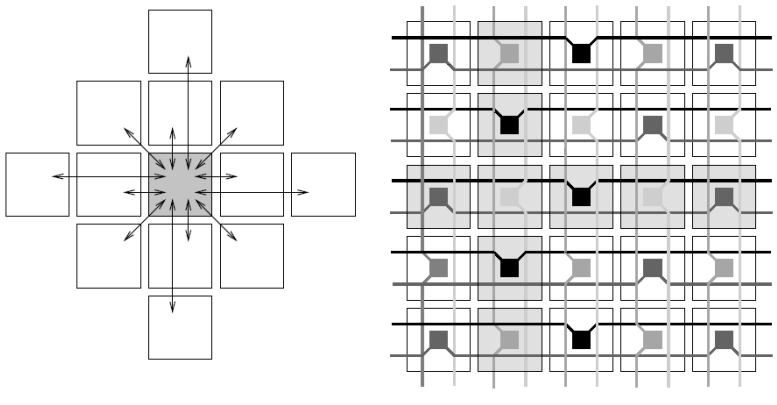

Figura 6 – (a) interconexão com vizinhos próximos;
(b) interconexão com vizinhos a uma distância de 4 posições.

A rede MATRIX é uma coleção hierárquica de barramentos de 8 bits que possui a opção de dinamicamente trocar as conexões. A rede interconecta os vizinhos mais próximos (Figura 6a), vizinhos a uma distância de quatro posições (Figura 6b) e também interconecta BFUs por linhas globais. Essa estrutura hierárquica de interconexão atinge alta taxa de transferência de dados e, em contraponto, dificulta o mapeamento da aplicação.

2.4 RAW

O projeto RAW (*Reconfigurable Architecture Workstation*) [WAI97] foi publicado por Elliot Waingold et al. do MIT. Ele é baseado na interconexão por rede malha (4x4) de 16 módulos replicados. Cada módulo é construído como um processador simples (MIPS R2000), com ULA, 32 KB de cache de dados, 96 KB de cache de instruções, um *pipeline* de ponto flutuante de 4 estágios, um roteador de comunicação estática e dois roteadores de comunicação dinâmica. O chip multiprocessado contém ao total 122 milhões de transistores, 1657 pinos e executa 16 instruções em paralelo por ciclo de relógio. Ele também possui 2 MB de RAM estática L1 distribuída, provendo 57 GB/s de largura de banda de memória on-chip e 14 portas de 32 bits que transferem dados a uma velocidade de 7,5 Gb/s de uma largura de banda total de 25 GB/s.

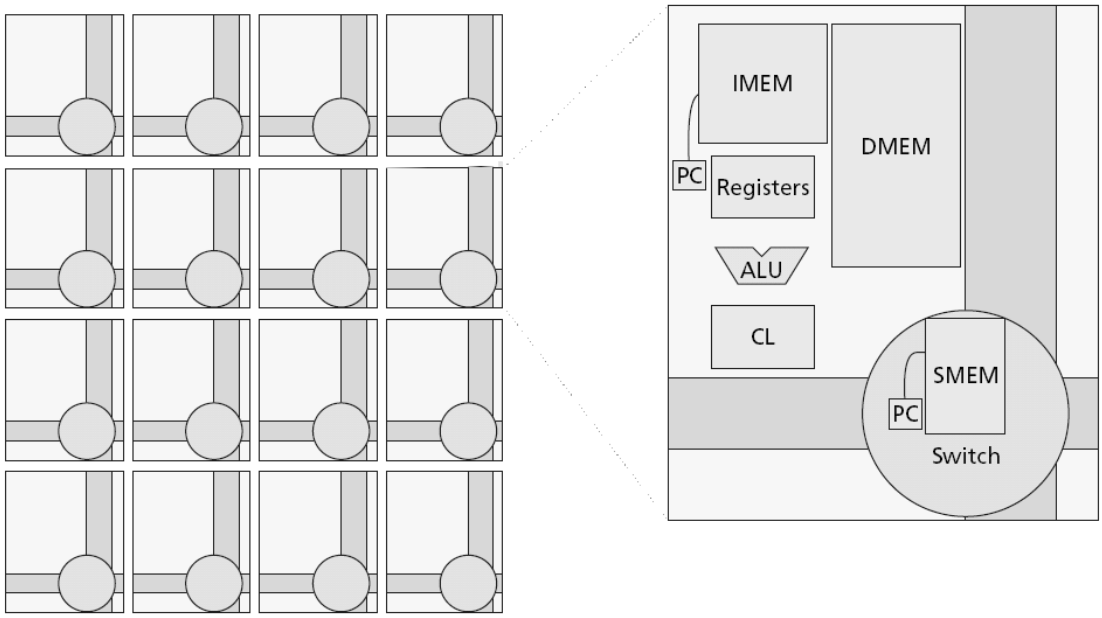

Figura 7 – Arquitetura do RAW.

Cada módulo se comunica diretamente apenas com seus quatro vizinhos por canais bidirecionais de 32 bits. O roteador estático dos módulos provê uma comunicação de baixa latência voltada para software e outras aplicações que necessitam de previsão do tempo de resposta para efetuar uma comunicação em tempo de compilação. O roteador dinâmico executa operações como interrupções e busca de dados que não foram encontrados na cache.

Além disso, fica a cargo do compilador controlar e otimizar os recursos do sistema, implementar uma memória global distribuída, resolver dependências de dados, prever saltos, renomear registradores e realizar o roteamento.

2.5 Pleiades

O Pleiades [RAB97] foi proposto por Jan Rabaey da Universidade da Califórnia em Berkeley. Ele é composto por microprocessadores e elementos de computação heterogêneos (satélites). O modelo da arquitetura fixa as primitivas de comunicação tanto entre o microprocessador e os satélites quanto entre cada satélite. Para cada domínio de aplicação (comunicação, codificação de voz, codificação de vídeo) uma arquitetura diferente pode ser criada. A Figura 8 apresenta o modelo heterogêneo do Pleiades.

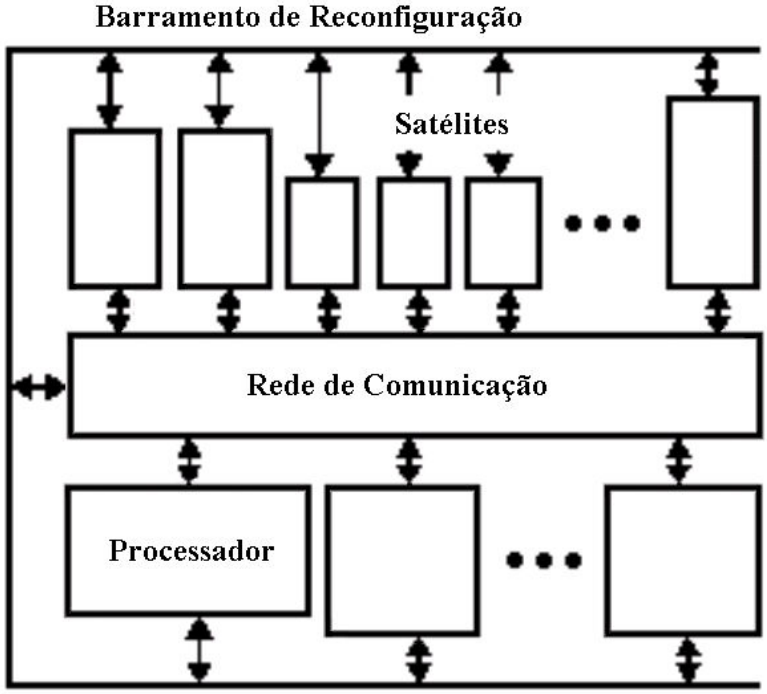

Figura 8 – Modelo heterogêneo do Pleiades.

O Pleiades objetiva o baixo consumo de potência, provendo uma plataforma computacional que mistura granularidades de programação e permitindo o mapeamento de aplicações que melhor se adaptam a arquitetura alvo. Os satélites podem ser microprocessadores, FPGAs ou unidades reconfiguráveis de grão grande. O controle principal é do microprocessador que simplifica a implementação de aplicações. A heterogeneidade da arquitetura também habilita a integração de memórias aos satélites, o qual pode ser acessado pela rede de comunicação.

O Pleiades possui um sistema de configuração e controle distribuído, onde cada satélite é equipado com uma interface que possibilita a troca de dados com outros satélites, sem envolver um controlador global. A configuração das interconexões permite construir um cluster de satélites, enquanto o controle permite especificar o tipo de operação a ser executada pelo satélite. Do ponto de vista do processador todos os registradores de controle são partes do mapa de memória e as configurações são escritas do ponto de vista do processador.

2.6 KressArray-III

O KressArray-III [HAR97] é uma versão posterior do KressArray-I (seção 2.1) que passou a chamar as DPUs de RDPUs. Nesta arquitetura as conexões passaram a ser bidirecionais, conectando-se às quatro RDPUs vizinhas. Aqui “bidirecional” significa que a direção é selecionada

em tempo de configuração, ou seja, ela é fixa durante a execução. Também foi inserido nesta arquitetura um hardware adicional responsável por paralelizar o acesso à memória, permitindo acesso consistente em modo rajada.

No KressArray-III as RDPUs podem servir tanto para roteamento quanto para a aplicação de uma determinada função. O conjunto de funções também foi estendido, agora podendo executar qualquer operação aritmética da linguagem C.

2.7 CHESS

O CHESS [MAR99] foi desenvolvido por Alan Marshall et al. da HP Labs e possui uma topologia em malha. A unidade de computação fundamental é a ULA de quatro bits, que possui um conjunto de 16 instruções básicas. Ao contrário de outras arquiteturas aritméticas reconfiguráveis, o CHESS possui uma arquitetura no formato de um tabuleiro de xadrez (Figura 9). Cada ULA contem barramentos de quatro bits de largura de entrada e saída em cada um dos seus quatro lados. Isto possibilita o envio e recebimento de dados de/para qualquer uma das oito ULAs à sua volta.

Em tempo de execução qualquer uma das chaves pode ser utilizada como memória de 16 palavras de 4 bits. Neste modo todas as chaves permanecem desconectadas, no entanto, barramentos que passam sobre a chave ainda podem continuar operando. Logo, se um grande número de chaves for utilizado como RAM a capacidade de roteamento fica reduzida. Para evitar este problema, o CHESS suporta a inserção de blockRAMs.

O CHESS provê a capacidade de encadeamento de ULAs, o que é muito útil para computações seriais de nibbles. As instruções executadas pelas ULAs podem ser estáticas ou dinâmicas. Instruções estáticas são armazenadas como parte da configuração fixa da ULA. Instruções dinâmicas são geradas pelo encadeamento de ULAs. Isso permite que instruções sejam modificadas ciclo-a-ciclo, suportando determinadas execuções como macros, implementando processadores específicos ou provendo um efeito de arquitetura reconfigurável de grão fino.

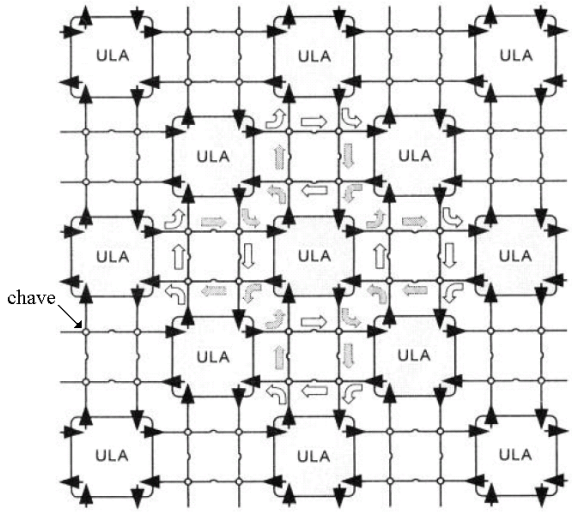

Figura 9 – Arquitetura CHESS

A arquitetura do CHESS foi otimizada para reduzir atrasos de comunicação, favorecendo interconexões locais entre os elementos de processamento e não possuindo barramentos globais. Interconexões longas podem ser obtidas pela conexão de múltiplos segmentos de barramentos ou pela utilização do CHESS como um *pipeline*.

2.8 DReAM

O DReAM (*Dynamically Reconfigurable Architecture for Mobile Systems*) [BEC00] foi desenvolvido na Universidade de Darmstadt por Jürgen Becker et al. O DreAM possui uma topologia em malha de unidades de processamento reconfiguráveis (*Reconfigurable Processing Unit* - RPU). Ele foi fabricado pela Mietec/Alcatel com tecnologia CMOS 0.35µm utilizando processo de células padrão *standard cells*. O DReAM é composto pela interconexão de 8 componentes, conforme apresentado na Figura 10. Enquanto o barramento local liga RPUs vizinhas, a interconexão global é composta por barramentos segmentados por chaves.

A estrutura da RPU (Figura 11) foi otimizada para requisitos de sistemas de comunicação móvel, sendo desenvolvida para executar manipulações de dados aritméticos. A RPU consiste em duas RAMs de 16 por 8 bits dupla porta, um módulo que implementa o Protocolo de Entrada e Saída de dados (PES), uma Unidade de Execução Compartilhada (UEC) e duas unidades de execução de 8 bits dinamicamente reconfiguráveis, chamadas de Unidades Reconfiguráveis de Processamento Aritmético (*Reconfigurable Arithmetic Processing Unit* - RAP).

As RAMs de dupla porta são utilizadas tanto como LUTs nas operações de multiplicação quanto como memória de dados ou como fila para armazenar resultados intermediários. Cada RAP integra dois deslocadores e uma ULA.

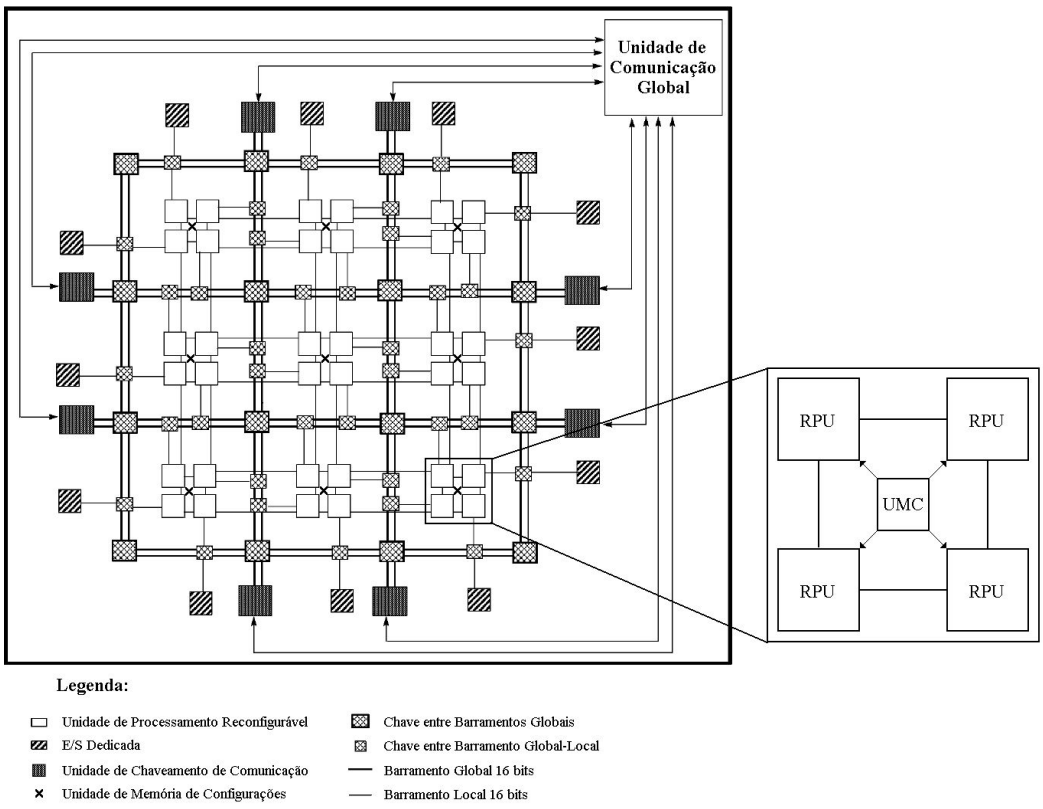

Figura 10 – Arquitetura DReAM

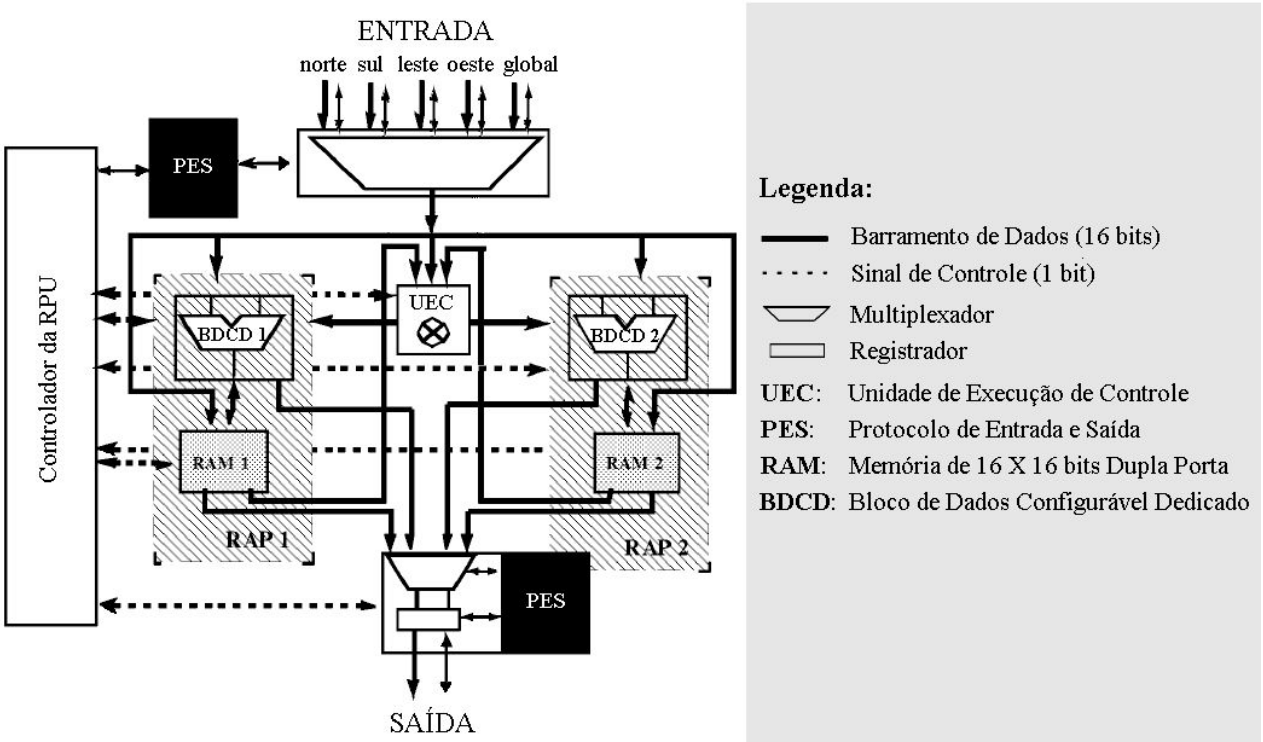

Figura 11 – Estrutura da RPU

2.9 Resumo das arquiteturas

Arquiteturas reconfiguráveis de grão grande possuem uma estrutura regular de bloco de dados replicados e interconectados por uma rede[1]. Dentre as arquiteturas reconfiguráveis estudadas neste capítulo, percebe-se que apenas as topologias em malha e em array foram utilizadas para a interconexão de PEs. Acredita-se que esta escolha deve-se a regularidade da implementação física e facilidade de implementação dos algoritmos responsáveis pela transferência de dados.

Quando malhas são utilizadas para interconectar PEs em arquiteturas reconfiguráveis, é fundamental que passos de posicionamento e roteamento antecedam a construção da arquitetura, pois a qualidade deste passo pode ter impacto direto no desempenho da aplicação [HER01]. Devido ao baixo número de PEs em arquiteturas reconfiguráveis de grão grande, o posicionamento e o roteamento são normalmente menos complexos nessas arquiteturas que em FPGAs.

A organização dos PEs em malha favorece a conexão com os vizinhos mais próximos, o que pode tornar o tempo de comunicação com PEs distantes muito longo, em termos de ciclos de relógio. Por este motivo, algumas arquiteturas que utilizam a topologia malha apresentam interconexões secundárias. Exemplos destes casos são o KressArray, o MATRIX e o DReAM.

O KressArray é uma abordagem com um grande bloco de dados, possuindo um barramento global que interconecta todos os PEs. O MATRIX foi o primeiro a estender os PEs a pequenos processadores, apresentando memórias locais. O RAW apresenta PEs com processadores RISC completos e memórias de instruções e dados. O CHESS interconecta os PEs na forma de um tabuleiro de xadrez, possuindo memórias embarcadas. O DReAM apresenta uma complexa rede de interconexão voltada para aplicações multimídia.

Uma forma mais simples de interconexão de PEs é por array. Essa estrutura é interessante

[1] Esta afirmação não se aplica ao Pleiades (seção 2.5), pelo motivo deste não ser uma arquitetura, mas sim um modelo para implementação de arquiteturas reconfiguráveis.

para aplicações com um fluxo uniforme e geralmente são utilizadas como *pipelines*, onde cada PE executa um estágio do *pipeline*. Essa é uma abordagem que funciona bem para aplicações com apenas uma linha de execução. Este capítulo apresentou apenas o RaPiD como exemplo de arquitetura que interconecta os PEs em array. O RaPiD é baseado na idéia de prover aceleração em aplicações que executam computações intensivas e de estrutura regular. Outra arquitetura que poderia ser citada como array, porém com uma estrutura circular, é o Systolic Ring [SAS02], voltado para aplicações multimídia.

Outras características importantes a serem observadas em arquiteturas reconfiguráveis de grão grande são granularidade e memória embarcada. Estas possuem uma ligação direta com o tipo de aplicação a ser tratada, pois é partir da aplicação que é possível refinar a arquitetura com os componentes que melhor resolvem o problema. Estas e outras características das arquiteturas reconfiguráveis de grão grande são apresentadas na Tabela 1.

Tabela 1 – Resumo das arquiteturas reconfiguráveis de grão grande.

Projeto e Referência	Ano	Arquitetura	Granularidade	Conexão	Aplicação Alvo	Memória Embarcada
KressArray-I	1995	Malha	32 bits	Conexão aos vizinhos e barramento segmentado	Indefinido	Não
RaPiD	1996	Array	16 bits	Barramento segmentado	*Pipeline*	Sim
Matrix	1996	Malha	8 bits Parametrizável	Conexão aos vizinhos, conexão a cada 4, barramento global	Indefinido	Sim
RAW	1997	Malha	8 bits Parametrizável	Conexão aos vizinhos	Indefinido	Sim
KressArray-III	1997	Malha	32 bits	Conexão aos vizinhos, linhas longas, barramento global	Indefinido	Não
Morphosys	1998	Malha	16 bits	Conexão aos vizinhos, conexão a cada 2 ou 3, barramento global	Indefinido	Não
CHESS	1999	Malha	4 bits	Conexão aos vizinhos	Multimídia	Sim
DReAM	2000	Malha	8 e 16 bits	Conexão aos vizinhos e barramentos segmentados	*Wireless*	Sim

É interessante observar que muitas das arquiteturas reconfiguráveis apresentadas na Tabela 1 não possuem uma aplicação alvo definida. Estas arquiteturas buscam tanto a flexibilidade dos FPGAs quanto desempenho de arquiteturas reconfiguráveis de grão grande para resolver problemas genéricos. Segundo Hartenstein [HAR01], uma arquitetura reconfigurável universal é obviamente uma ilusão, pois sempre será possível otimizá-la para obter um melhor desempenho em uma aplicação específica.

3 PROCESSADORES RECONFIGURÁVEIS

Aplicações com computações intensivas passam em média 90% do tempo em apenas 10% do código [HEN90]. Isto permite otimizar o processador utilizado para executar os trechos de código responsáveis por este maior consumo de tempo de processamento. A implementação de um hardware específico é uma alternativa possível para alcançar um ganho de desempenho significativo. Deve-se observar que nem sempre o hardware dedicado é a solução para ganho de desempenho, pois em determinados casos o gargalo é o conjunto de operações de entrada/saída ou acessos à memória externa.

Processadores com um conjunto de instruções específicas para a aplicação (*Application-Specific Instruction-Set Processor* - ASIP) provêm flexibilidade e desempenho a um custo de área adicional para cada nova função implementada em hardware. Processadores com um conjunto de instruções reconfiguráveis (*Reconfigurable Instruction-Set Processor* - RISP) não apresentam um custo em hardware para cada nova instrução em hardware, mas sim um custo pré-definido de área onde lógica reconfigurável pode ser programada em tempo de execução.

Este capítulo apresenta o estado da arte em processadores reconfiguráveis. Um resumo dos processadores estudados e um quadro comparativo são apresentados ao final do capítulo.

3.1 PRISM-I

O PRISM-I (*Processor Reconfiguration through Instruction-Set Metamorphosis*) [ATH93] teve sua primeira publicação por Peter Athanas et al. da Universidade do Estado da Virginia. Ele foi desenvolvido como prova de conceito para mostrar a viabilidade de incorporar um conjunto de instruções configurável a um processador de propósito geral. O PRISM-I consiste na interconexão de um nodo de processamento Armstrong com uma plataforma com hardware reconfigurável por meio de um barramento, conforme ilustrado na Figura 12.

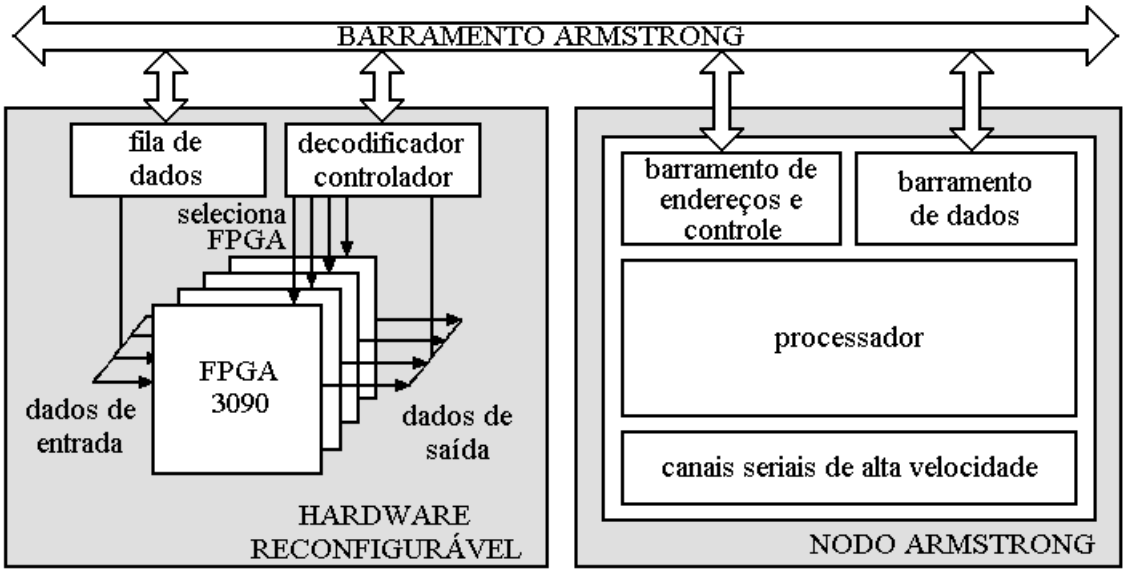

Figura 12 – Arquitetura do PRISM-I.

O nodo de processamento Armstrong é baseado em um processador M68010 da Motorola,

que roda a uma freqüência de 10MHz. A plataforma com hardware reconfigurável possui 4 FPGAs 3090 da Xilinx [XIL04], cada um com seis mil portas lógicas equivalentes e 144 pinos de entrada e saída. A interconexão das duas plataformas foi feita pela interface de coprocessador de 16 bits do M68010.

Para avaliar o sistema, o processador fez chamadas de funções implementadas em hardware reconfigurável. As operações testadas consumiram de 48 a 72 ciclos de relógio para completar a operação e entregar a resposta de volta ao processador.

Os testes executados, embora tenham sido iniciais e prejudicados pelo baixo desempenho do meio de comunicação, mostraram que é possível obter ganhos com a implementação de instruções em hardware reconfigurável. Dentre as diversas operações executadas, a que obteve melhor ganho foi a função de logaritmo na base 2, executando 54 vezes mais rápido que em software.

3.2 PRISM-II

O PRISM-II [WAZ93] foi a versão seguinte ao PRISM-I (seção 3.1). Esta versão teve como principais objetivos estender o número de instruções suportadas pelo compilador e reduzir o tempo de comunicação entre o processador e o hardware reconfigurável. Enquanto o PRISM-I leva em média 100ns para devolver a resposta ao processador, o tempo gasto na nova versão é de 30ns. Outra modificação feita foi no tamanho da palavra de dados, que no PRISM-I era fixa em 32 bits, e que no PRISM-II é de 64 bits para entrada e 32 bits para saída. O PRISM-I também possuía a limitação que uma instrução tinha que caber em um único FPGA. A arquitetura do PRISM-II permite que uma única operação seja dividida em até três FPGAs.

Para prover um melhor balanceamento entre hardware e software o processador M68010 foi substituído pelo AMD Am29050, que roda a 33MHz e possui um desempenho aproximado de 28 MIPS. Os FPGAs 3090 foram substituídos pelos 4010 com aproximadamente 20 mil portas lógicas equivalentes e 160 pinos de entrada e saída. Todas essas modificações resultaram em um fator de aceleração de aproximadamente 86 vezes em relação a software para uma função de reversão de bits em uma palavra de 32 bits. Isso mostra que essa versão obteve um ganho significativo de desempenho em relação ao PRISM-I, pois quando a mesma função foi executada na primeira versão ela obteve um fator de aceleração de apenas 26 vezes em relação a software.

3.3 Nano Processor

O Nano Processor [WIR94] foi proposto por Michael Wirthlin et al. da Universidade de Brigham Young. O seu principal objetivo é poder ser implementado em FPGAs da época, por isso ele dispõe de uma arquitetura simples, ocupando pouca área. Ele possui seis instruções nativas e permite um total de 32 instruções. O processador é organizado hierarquicamente em três níveis, conforme apresentado na Figura 13.

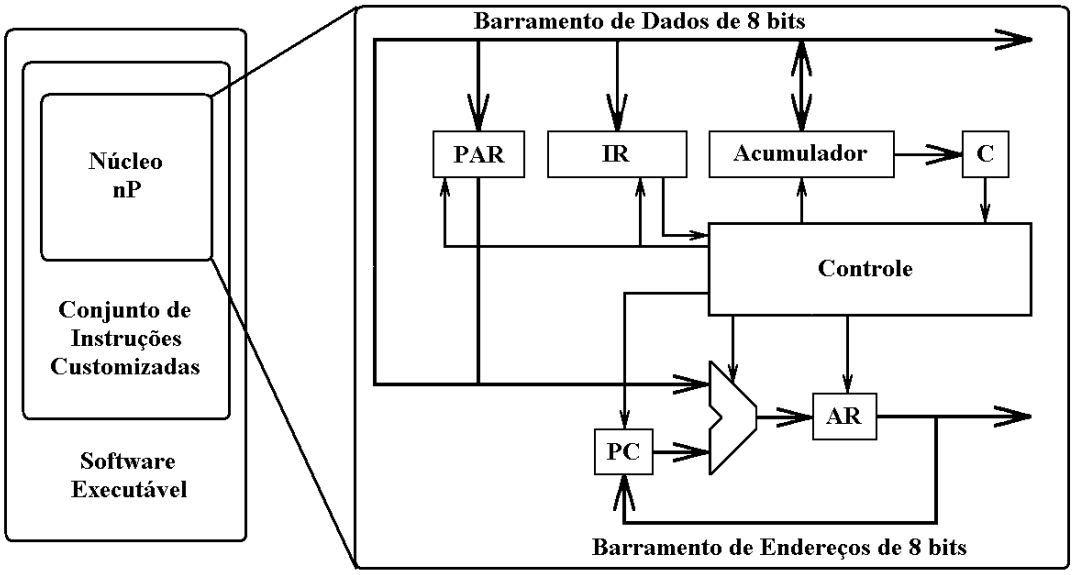

Figura 13 – Organização e arquitetura do Nano Processor.

O nível mais interno do processador é o núcleo do Nano Processor, que possui as seis instruções nativas e não foi desenvolvido para ser modificado. Ele possui um registrador de instruções (*Instruction Register* - IR) de 5 bits, um registrador de página de endereços (*Page Address Register* - PAR) de 3 bits, um contador de programa (*Program Counter* - PC) de 8 bits, um registrador de endereços (*Address Register* - AR) de 8 bits e um acumulador de 8 bits com 1 bit de *carry*. Nesta arquitetura apenas instruções aritméticas de adição e subtração foram implementadas. O núcleo do Nano Processor ocupa 40 CLBs da família 3000 de FPGAs da Xilinx [XIL04].

O nível intermediário do processador é chamado de Conjunto de Instruções Customizadas. Nela são implementadas as instruções específicas ao problema a ser tratado pelo processador. Essas instruções podem ser selecionadas de uma biblioteca previamente desenvolvida ou criadas a partir de editores de esquemáticos ou ferramentas de síntese. O nível mais externo é o Software Executável. Nele é implementado a aplicação em código assembly onde são feitas as chamadas das instruções nativas e customizadas.

O Nano Processor foi utilizado para uma plataforma da National Technologies. A plataforma inclui dois FPGAs 3090 da Xilinx [XIL04], duas memórias SRAMs 32k x 8, uma memória DRAM de 1 Mb, um codec stereo de 16 bits e uma interface com o PC. O resultado obtido com esta plataforma foi 240 vezes mais rápido que o obtido com um processador 486 executando a 33 MHz para a aplicação de *saturating mixer*.

3.4 DISC

O DISC (*Dynamic Instruction Set Computer*) [WIR95] foi proposto por Michael Wirthlin et al. da Universidade de Brigham Young, um ano após a publicação do Nano Processor (seção 3.3). Ele foi desenvolvido para suportar modificações no seu conjunto de instruções em tempo de execução. Para isso ele deve ser implementado em dispositivos que suportem reconfigurações dinâmicas parciais, como, por exemplo, os FPGAs fabricados pela Algotronix, Atmel, National Semiconductor e Xilinx. No DISC as instruções são implementadas como módulos de hardware e são substituídas quando não existe mais espaço disponível para a instrução seguinte. Essa abordagem possui a vantagem de não só utilizar uma área em hardware virtualmente maior que a existente fisicamente, mas também de permitir implementar funções complexas com melhor

desempenho quando comparado a software. A arquitetura do DISC e alguns exemplos de instruções são apresentados na Figura 14.

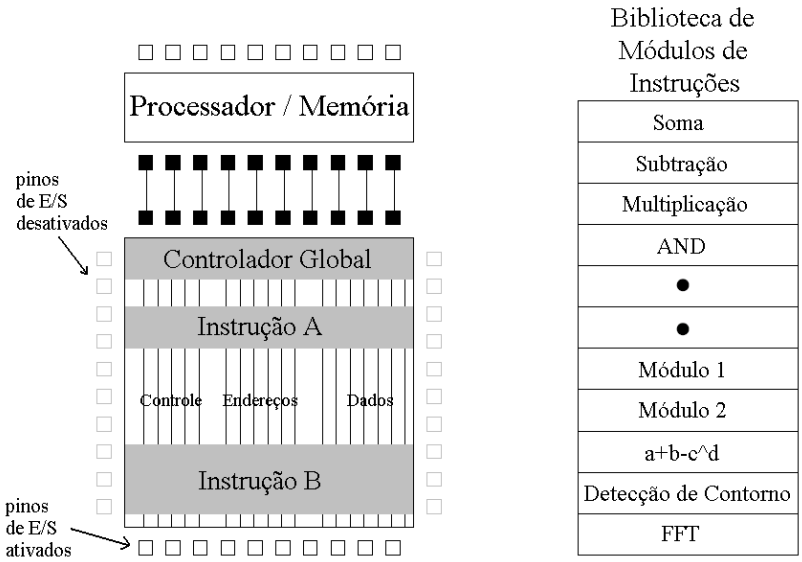

Figura 14 – Arquitetura do DISC e exemplo de biblioteca de módulos de instruções.

Antes de iniciar o DISC, um programa é carregado na memória, o módulo de controle global é carregado em hardware e o espaço de instruções dinâmicas é liberado. A seguir o DISC inicia a execução do programa. Quando for feita a chamada de uma instrução que não se encontra configurada, o processador é bloqueado e uma requisição de configuração de instrução é enviada ao computador hospedeiro pelos pinos de entrada e saída do sistema. O computador hospedeiro verifica o tamanho do módulo requisitado, seleciona onde o mesmo será posicionado, remove um ou mais módulos se não existir espaço suficiente no dispositivo e reposiciona o módulo para onde ele havia sido selecionado. A seguir, o computador hospedeiro envia a nova configuração para o sistema, o processador sai do estado de bloqueio e a instrução é executada.

O principal módulo do DISC é o controlador global, que possui a tarefa de gerenciar os recursos de memória, os canais de comunicação e efetuar a reconfiguração dinâmica dos módulos de instrução. O diagrama de blocos do controlador global é apresentado na Figura 15.

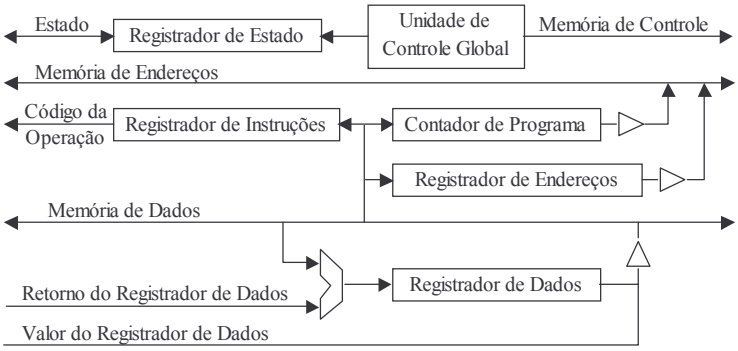

Figura 15 – Diagrama de blocos do controlador global.

3.5 OneChip

O OneChip [WIT96] teve a sua primeira publicação por Ralph Wittig et al. da Universidade de Toronto. Nesse projeto é defendida a idéia que o processador deve estar fortemente acoplado ao

hardware reconfigurável, minimizando os bloqueios do processador com uma comunicação rápida e eficiente entre eles.

Os recursos reconfiguráveis foram integrados no estágio de execução do processador, sendo chamados de Unidades Funcionais Programáveis (*Programmable Functional Unit* - PFU). As PFUs são adicionadas em paralelo às Unidades Funcionais Básicas (*Basic Functional Unit* - BFU) do processador. As PFUs implementam funções específicas a uma aplicação, podendo ser circuitos seqüenciais ou combinacionais. As PFUs também podem ser utilizadas como lógica de cola para que o processamento de uma determinada função venha do mundo externo. As BFUs são responsáveis pelas operações lógicas e aritméticas elementares de qualquer aplicação.

O OneChip foi prototipado em um Transmogrifier-1, que possui quatro FPGAs 4010 da Xilinx [XIL04], dois chips de interconexão Aptix AX1024 e quatro memórias SRAMs 32k x 9. O processador DLX [HEN03] baseado no MIPS foi particionado em três FPGAs, deixando um FPGA inteiro e a lógica restante dos outros dois para PFUs do usuário. O OneChip obteve um fator de aceleração de quase 50 vezes na aplicação de um DCT (*Discrete Cosine Transform*) em relação a execução em uma Workstation SGI Indy rodando a 150MHz com processador MIPS R4400.

3.6 Garp

O Garp [HAU97a] teve sua primeira publicação por John Hauser da Universidade da Califórnia em Berkeley e foi projetado com o principal objetivo de acelerar laços de aplicações de propósito geral. O projeto implementa um coprocessador conectado ao MIPS, que consiste em uma malha bidimensional de CLBs interconectadas por recursos de roteamento programáveis (Figura 16). A célula lógica básica da matriz reconfigurável é uma unidade de 2 bits com quatro entradas de 2 bits cada. Quatro canais de 32 bits de dados e um canal de 32 bits de endereços comunicam o processador a matriz reconfigurável.

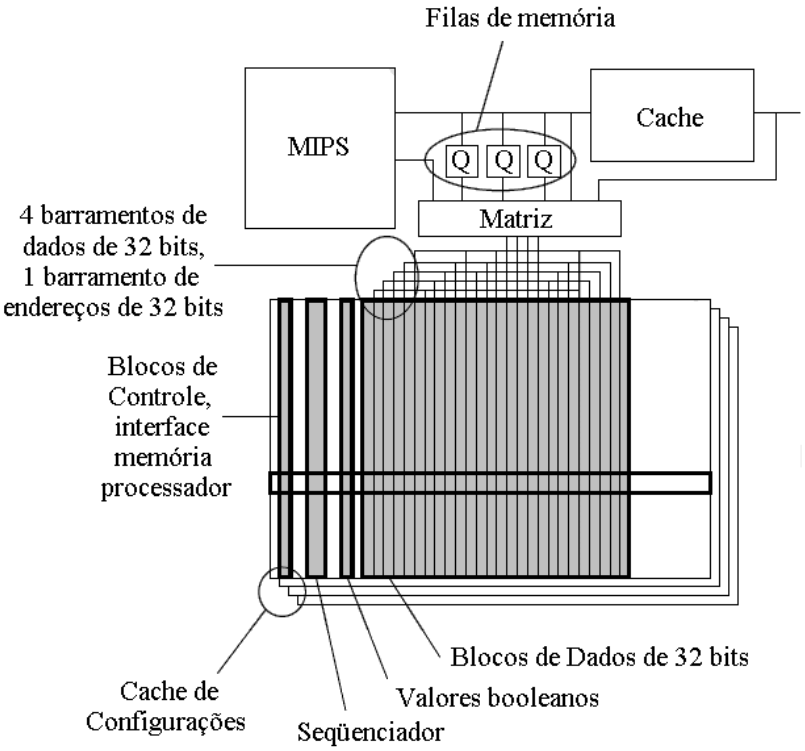

Figura 16 – Arquitetura do Garp.

A matriz reconfigurável possui pelo menos 24 blocos lógicos em uma coluna, dos quais 16 blocos a partir da primeira posição são conectados ao barramento compartilhado processador-memória. Na tentativa de minimizar os acessos ao barramento compartilhado, o projeto emprega a utilização de uma memória cache que pode armazenar quatro configurações totais da matriz ou diversas configurações parciais. Uma reconfiguração total buscada da cache pode entrar em funcionamento em até cinco ciclos de relógio.

3.7 Chimaera

O Chimaera [HAU97b] foi proposto por Scott Hauck et al. da Universidade de Washington. Ele possui Unidades Funcionais Reconfiguráveis (*Reconfigurable Functional Unit* - RFU) pequenas o suficiente para serem integradas no próprio processador, buscando minimizar o gargalo de comunicação entre processador e FPGA. Esta abordagem permite utilizar a lógica reconfigurável para acelerar as computações críticas das aplicações, enquanto aproveita a flexibilidade dos processadores para a criação de software. A Figura 17 apresenta a arquitetura do Chimaera.

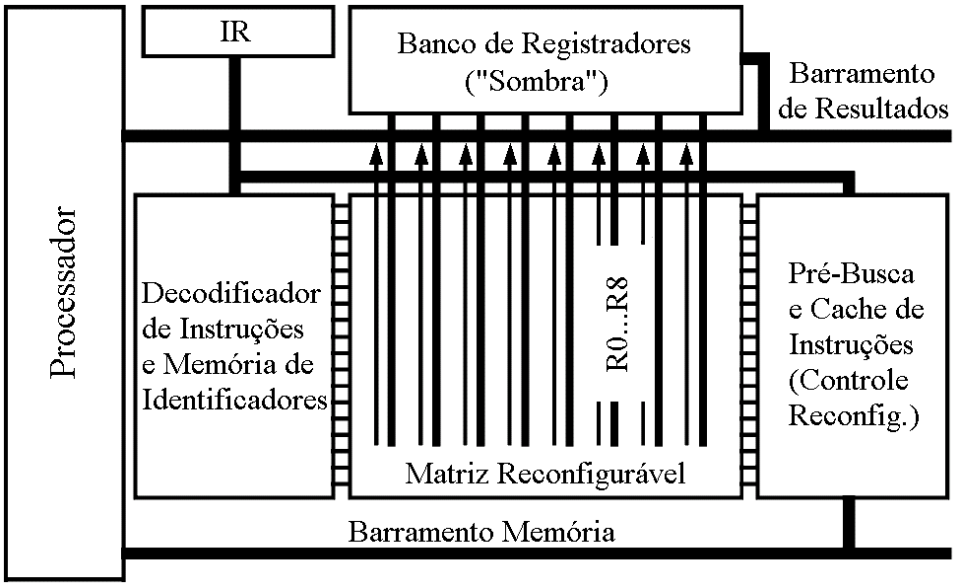

Figura 17 – Arquitetura do Chimaera.

O principal componente do sistema é a matriz reconfigurável, que possui uma arquitetura própria baseada no Triptych [EBE95], na família FLEX 8000 da Altera [ALT04] e no PRISC [RAZ94]. A matriz recebe entradas do banco de registradores chamado “sombra”, pois este possui cópia de um subconjunto de valores do banco de registradores do processador. Cada modificação feita no banco de registradores do processador é propagada pelo barramento, assim atualizando o “sombra”. As instruções que forem carregadas na matriz podem utilizar uma ou mais linhas de RFUs. Sempre que uma linha de RFUs for configurada, uma memória conectada à matriz armazena, em uma posição específica para a linha da RFU, o identificador da instrução que foi carregado na linha. No momento em que uma operação da matriz for executada, as linhas de saída da matriz que possuem o identificador da instrução são habilitadas a colocar o resultado no barramento.

Enquanto o processador executa instruções nativas, o módulo de pré-busca verifica a cada ciclo de relógio o Registrador de Instruções (*Instruction Register* - IR), de forma “especulativa”. Isso permite iniciar a execução da operação na matriz antes mesmo da chamada ter sido feita pelo processador, assim reduzindo ou eliminando o bloqueio do mesmo à espera do resultado. Quando a

chamada é de fato executada pelo processador, a matriz escreve o resultado no barramento de resultados caso a operação já tenha sido computada, caso contrário ela bloqueio o processador até a computação ser efetuada. Também pode ocorrer da operação não estar presente na matriz, assim causando um bloqueio no processador até que a operação seja buscada da cache de instruções ou da memória, parcialmente configurada na matriz e executada. Neste caso o tempo de reconfiguração pode ser significativo, logo o programador deve evitar ficar recarregando a RFU constantemente.

O Chimaera possui nove registradores que podem servir como origem de uma operação a ser executada nas RFUs. O programador deve saber quais destes registradores devem ser carregados antes de chamar a operação a ser executada na matriz, pois quando o processador chama uma operação implementada na RFU é passado como parâmetro somente o identificador da operação e o registrador de retorno.

Uma célula da matriz reconfigurável com sua estrutura de roteamento é apresentada na Figura 18. Em cada linha da matriz, existe uma célula por bit do processador. Por exemplo, um processador de 32 bits possui 32 células por linha, portanto uma célula N tem acesso ao enésimo bit dos registradores R0 a R8. Cada célula possui quatro entradas (I1-I4) e quatro saídas (O1-O4) que são utilizadas para comunicação com outras células.

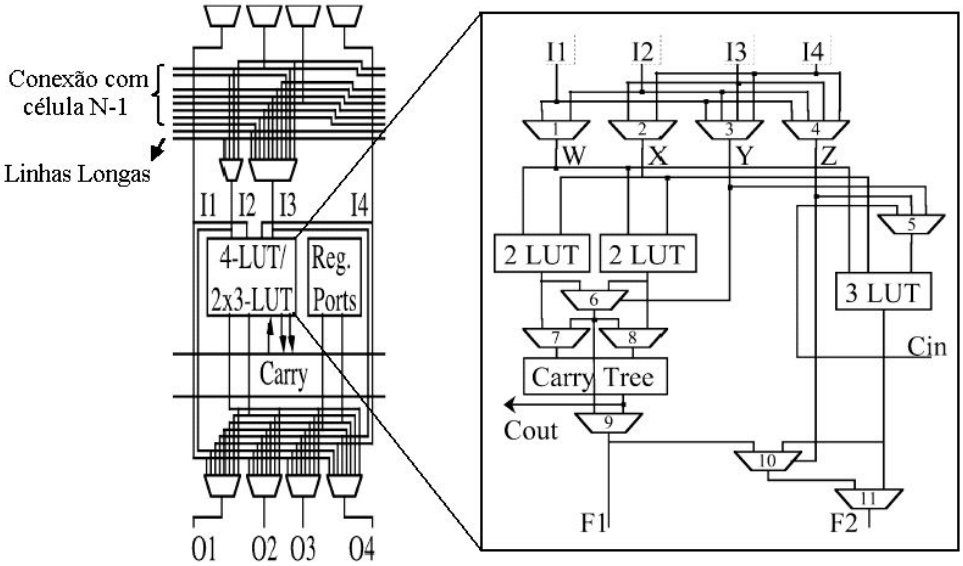

Figura 18 – Célula da matriz reconfigurável e estrutura de roteamento do Chimaera.

As RFUs do Chimaera foram desenvolvidas para tratar com eficiência instruções típicas de um processador como AND, OR, XOR, XNOR, operações aritméticas simples, condições de salto e combinações destas. Essa arquitetura, embora simples, apresentou desempenhos até 160 vezes superiores do que a execução do processador sem lógica reconfigurável.

3.8 Morphosys

O Morphosys foi proposto por Hartej Singh et al. [SIN98] da Universidade da Califórnia em Irvine e consiste em sete componentes principais (Figura 19): um processador baseado no MIPS (Tiny_RISC), uma malha 8x8 de células reconfiguráveis, uma memória de contextos, uma fila, uma cache de dados, um controlador de DMA e duas memórias principais (uma do processador e outra de contextos da malha reconfigurável).

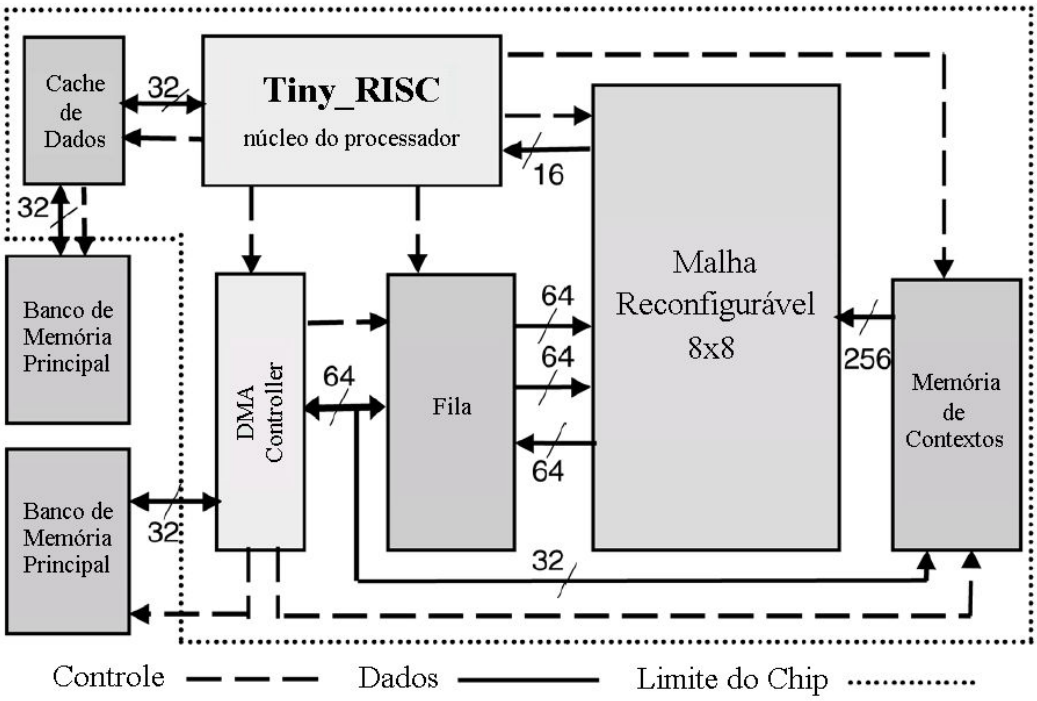

Figura 19 – Arquitetura do MorphoSys.

Cada célula da malha reconfigurável pode ser individualmente reconfigurada. Ela é composta de uma ULA de 16 bits com módulo de multiplicação acoplado, uma unidade de deslocamento e dois multiplexadores que selecionam as entradas da célula. Cada célula também possui um registrador de saída e um banco de registradores.

Enquanto o processador executa as tarefas seqüenciais e de controle de transferência de dados entre o hardware reconfigurável e a memória, o hardware reconfigurável explora o paralelismo disponível do algoritmo da aplicação. O processador RISC possui duas classes de instruções específicas para controlar os componentes do MorphoSys. Uma dessas classes é responsável por carregar contextos da memória principal para a memória de contextos e a outra é responsável por iniciar transferências de dados entre a memória principal e a fila, essa classe é chamada de instruções de DMA.

A fila é organizada em dois conjuntos (0 e 1) de dois bancos (A e B) de 128 palavras de 16 bits. Um barramento de operandos de 128 bits conecta a fila com a matriz de células reconfiguráveis. Todas as células reconfiguráveis de uma mesma linha compartilham 16 bits do barramento de operandos. A transferência de dados entre a fila e a matriz de células reconfiguráveis pode executar apenas em modo entrelaçado, por exemplo, os dados podem ser acessados apenas alternando entre banco A e B. Enquanto este método aumenta a velocidade de acesso consecutivo a memória, ele restringe o acesso a dados aleatórios.

Uma característica importante da malha reconfigurável é a sua interconexão em três níveis, o qual pode ser vista na Figura 20. O primeiro nível acessa o vizinho mais próximo da mesma linha/coluna. O segundo nível provê conectividade completa com a linha/coluna do mesmo quadrante. O terceiro nível provê conectividade inter-quadrante, que consiste em barramentos que percorrem todas as linhas e colunas, cruzando as bordas dos quadrantes.

Um dos algoritmos utilizados para avaliar o desempenho do MorphoSys foi o de DCT, sendo executado em 370ns. Este tempo foi mais de três vezes menor que a execução do algoritmo em um computador com processador Pentium MMX com instruções otimizadas.

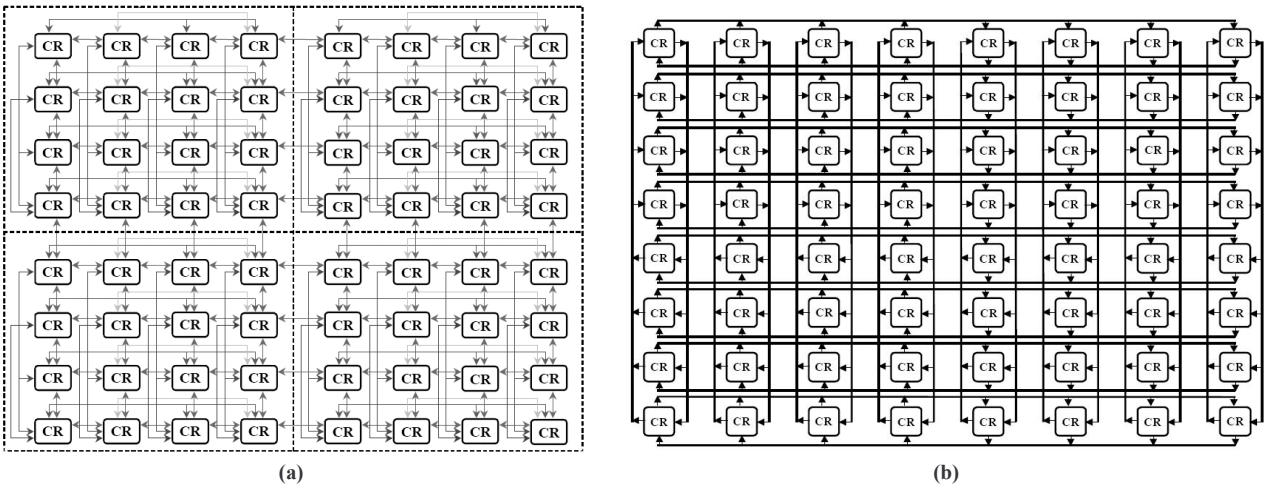

Figura 20 – Conectividade das 8x8 Células Reconfiguráveis do MorphoSys em três níveis: (a) primeiro e segundo níveis; (b) terceiro nível.

3.9 NAPA

O NAPA (*National Adaptive Processing Architecture*) [RUP98] foi proposto por Charlé Rupp et al. da National Semiconductor Corporation. Ele é composto de uma rede denominada ToggleBus que interconecta um ou mais processadores, conforme apresentado na Figura 21. Um dos processadores da arquitetura é responsável por comunicar o sistema com um computador hospedeiro e gerar o vetor de controle (C) que define o tipo de fluxo de dados e parâmetros para cada ciclo do ToggleBus. Blocos de memória externa (M) se comunicam diretamente com os processadores da arquitetura por linhas de endereços (E) e dados (D). Cada processador possui acesso a rede de comunicação ToggleBus pelas portas P e Q. Além disso cada processador possui 128 pinos de entrada e saída para atuar diretamente com sensores externos.

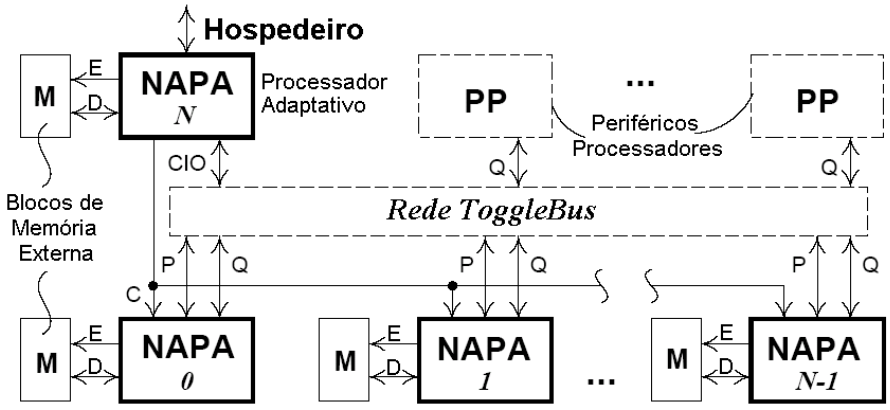

Figura 21 – Arquitetura do NAPA.

O NAPA integra lógica reconfigurável (*Adaptive Logic Processor* - ALP) com um processador escalar fixo (*Fixed Instruction Processor* - FIP) de maneira fortemente acoplada. Na primeira implementação do NAPA, as ALPs possuiam 6144 células reconfiguráveis (64 colunas por 96 linhas) e 1608 linhas de roteamento global. O processador utilizado foi o CompactRISC que é um pequeno processador RISC de 32 bits que roda a 50 MHz. Não foram apresentados dados de desempenho do sistema no artigo [RUP98].

3.10 OneChip-98

O OneChip-98 [JAC99] é uma versão posterior do OneChip (seção 3.5), publicada por Jeffrey Jacob e Paul Chow ambos da Universidade de Toronto. A primeira versão do OneChip com processador, memória e lógica reconfigurável integrados obteve quase 50 vezes superior desempenho que uma estação de trabalho da época. A partir dele, detectou-se que o gargalo não era mais a interface entre o processador e a lógica reconfigurável, mas sim a interface destes com a memória. Por este motivo, o OneChip-98 (Figura 22) busca uma maior taxa de transferência de dados com a memória, permitindo o acesso concorrente a mesma pela lógica reconfigurável e pelo processador.

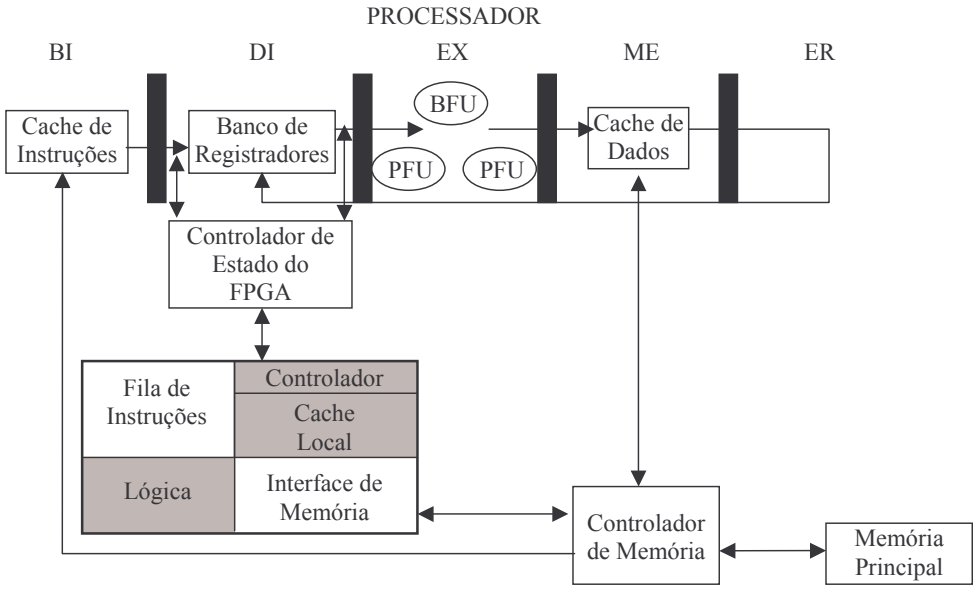

Figura 22 – Arquitetura do OneChip-98.

Para aumentar a taxa de transferência com a memória principal, caches foram adicionadas ao sistema, implementando uma hierarquia de memória semelhante à empregada em computadores pessoais. Conseqüentemente, um mesmo dado pode ser trazido tanto para a cache do processador quanto para a cache do FPGA, causando um possível problema de coerência de caches. O tratamento deste problema é analisado no artigo [JAC99].

O OneChip-98 foi prototipado em um Transmogrifier-2 com dois PLDs Altera Flex10K50. A aplicação de um algoritmo de DCT obteve uma aceleração de dez vezes sobre um Ultra 2 rodando a 300MHz. Os autores apontam como aplicações alvo problemas que acessam muito a memória.

3.11 PipeRench

O PipeRench [GOL99] foi proposto por Seth Goldstein et al. da Universidade de Carnegie Mellon. A arquitetura é composta por uma rede de elementos lógicos e de armazenamento interconectados, que atuam como um coprocessador. Esta arquitetura é organizada de forma a implementar um *pipeline* de configurações, sendo particularmente interessante para aplicações baseadas em fluxos uniformes (como processamento de imagens ou sons), ou qualquer computação simples e regular sobre grandes conjuntos de pequenos elementos de dados.

A Figura 23 apresenta uma visão abstrata de dois estágios do PipeRench e uma visão

detalhada de um estágio. Cada PE contém uma ULA e um conjunto de registradores para comunicação entre estágios (registrador de passagem). Cada ULA contém LUTs e controle extra para propagação de vai um, detecção de zero, e outros bits de controle. Lógica combinacional pode ser implementada usando um conjunto de N ULAs de B bits. Funções combinacionais complexas podem ser obtidas com o cascateamento das linhas de propagação de vai um, através da rede de interconexão.

A partir das redes de interconexão os PEs podem acessar operandos dos estágios anteriores que estão armazenados nos registradores de saída, bem como saídas de outros PEs do mesmo estágio. O barramento global é utilizado tanto pela aplicação para entrada e saída de dados quanto pelos PEs para alcançar os seus destinos, porque os estágios do *pipeline* de uma dada aplicação podem estar fisicamente em qualquer um dos estágios da arquitetura.

Com este dispositivo há um ganho considerável de desempenho sobre processadores convencionais. Enquanto estes forçam uma serialização de operações intrinsecamente paralelas, PipeRench explora o paralelismo dessas operações, além de aumentar a flexibilidade e diminuir o tempo de projeto do sistema.

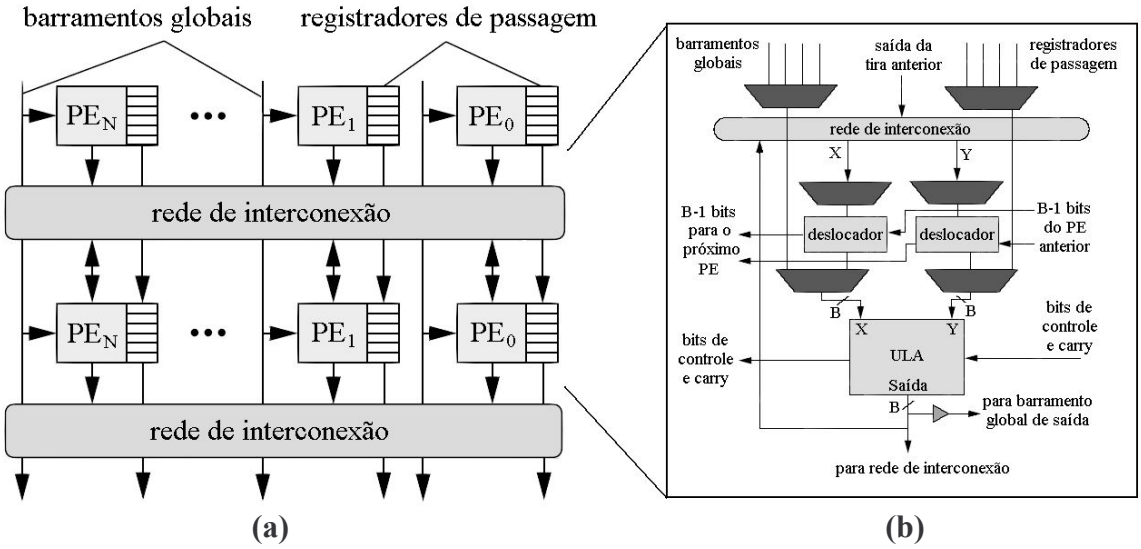

Figura 23 – Arquitetura do PipeRench: (a) diversas linhas de PEs e redes de interconexão formam a arquitetura; (b) arquitetura interna de um estágio do PipeRench.

3.12 Resumo dos processadores reconfiguráveis

Analisar a aplicação encontrando os melhores trechos de código a serem otimizados é essencial para não dar o "tiro pela culatra". Por isso deve-se ter cuidado para que o tempo de execução total do trecho de código em software não seja superior ao somatório do tempo de (i) reconfiguração; (ii) comunicação entre processador e lógica reconfigurável e vice-versa; (iii) tempo de execução em hardware.

para não efetuar reconfigurações sucessivas, que podem não somente não obter ganho como também reduzir o desempenho do sistema.

A maior parte dos processadores estudados utiliza a lógica programável para acelerar computações críticas do processador, dessa forma contornando o problema tanto de efetuar

constantes configurações quanto o baixo desempenho dos processadores em determinadas situações.

O Garp é um sistema composto de um microprocessador com um coprocessador reconfigurável. Ele apresenta PEs baseados em LUTs com blocos de dados de apenas dois bits. De fato, esta arquitetura se aproxima de um FPGA e é considerada com freqüência de grão fino.

O MorphoSys é um sistema híbrido que consiste de um processador RISC com um coprocessador reconfigurável.

Enquanto o RaPiD utiliza m modelo de reconfiguração estática, o PipeRench conta com reconfiguração dinâmica, permitindo configurar um PE a cada ciclo de relógio. Além disso, a maioria das interconexões com os vizinhos são unidirecionais, pois estas progridem dentro da arquitetura de uma extremidade para a outra como um pipeline.

Tabela 2 – Resumo dos Processadores Reconfiguráveis.

Projeto e Referência	Prism-I	Prism-II	Nano Processor	DISC	OneChip	Garp	Chimaera	Morpho Sys	NAPA	OneChip 98	Pipe Rench
Ano	1993	1993	1994	1995	1996	1997	1997	1998	1998	1999	1999
Processador	M68010	Am29050	RISC	CISC	DLX	MIPS	MIPS	TinyRISC	RISC	DLX	Genérico
Aplicação	Genérico	Genérico	Genérico	Genérico	Genérico	Genérico	Multi-mídia	Genérico	Genérico	Genérico	Multi-mídia
Acoplamento	Anexado	Coproc	RFU	RFU	RFU	Coproc	RFU	Coproc	Coproc	RFU	Anexado
Instruções	Todas	Todas	Persona-lizadas	Todas	Todas	Stream	Persona-lizada	Todas	Todas	Stream	Stream
Tabela de Config.	Não	Não	Não	Sim	Não	Sim	Sim	Sim	Não	Sim	-
Operandos	Fixo	Fixo	Fixo	Flexível	Fixo	Fixo	Fixo	Fixo	Fixo	Fixo	Fixo
Banco de Registradores	Não	Não	Sim	Sim	Sim	Sim	Sim	Sim	Não	Sim	Não
Portas de Memória	Não	Não	Sim (1)	Sim (1)	Sim (1)	Sim (1)	Não	Sim (1)	Sim (2)	Sim (1)	Sim (1)
Granularidade	Fina	Fina	Fina	Fina	Fina	Fina	Fina	Grande	Fina	Fina	Grande
Tipo Lógica Reconfig.	3090	4010	3000	CLAy-31	4010	Persona-lizada	Persona-lizada	Persona-lizada	Persona-lizada	Persona-lizada	Persona-lizada
Dinamicam. Reconfig.	Não	Não	Não	Parcial	Não	Sim	Sim	Sim	Sim	Sim	Sim
Método Reconfig.	Soft	Soft	Soft	Soft	Soft	Soft	Controla-dor	Controla-dor	Controla-dor	Controla-dor	Controla-dor
Realocável	Não	Não	Não	Sim	Não	Sim	Sim	Não	Sim	Não	Sim
Aceleração	50	86	240	23,5	40	43	160	6,5	-	32	189

Para uma análise bem sucedida destas arquiteturas, algumas características importantes devem ser consideradas ...

Portanto, o que deve ser levado em conta é a relação flexibilidade da arquitetura (complexidade de roteamento adicionada para resolver diversos problemas) versus desempenho obtido. Por esse motivo as estruturas das arquiteturas reconfiguráveis são um fator importante para não restringir a velocidade de comunicação com a memória e com os recursos reconfiguráveis. Por isso a conseqüência de aumento da complexidade do roteamento pode acabar limitando a comunicação entre diversos pontos da arquitetura.

Entretanto, a qualidade do suporte das computações variam de arquitetura para arquitetura: algumas possuem memórias embarcadas permitindo acessos concorrentes a dados. Outras possuem uma memória tão pequena que não se adapta bem para certas aplicações. Além disso, outras arquiteturas possuem tantos recursos de roteamento que acabam pecando pelo alto atraso entre os

recursos.

4 CONCLUSÃO

Embora a computação reconfigurável seja flexível a ponto de permitir modificar partes do hardware em tempo de execução e algumas vezes alcançar resultados até duas ordens de grandeza melhores que a execução puramente software, uma abordagem utilizando somente computação reconfigurável parece não ser a melhor alternativa para a implementação de sistemas. A principal dificuldade hoje é que sistemas complexos demandam muita área para sua implementação. Isto exige muitas reconfigurações para cumprir a tarefa apenas com hardware reconfigurável, e tem como conseqüência a degradação do desempenho do sistema como um todo.

Entretanto, é possível que no futuro tenha-se dispositivos tão grandes e com tempos de reconfiguração tão curtos que viabilizem a implementação de sistemas compostos essencialmente de lógica reconfigurável. Enquanto este avanço tecnológico não é alcançado, duas abordagens em direção a um aumento de desempenho podem ser obtidas com arquiteturas reconfiguráveis de grão grande e com processadores reconfiguráveis, como pode ser verificado nos estudos de caso apresentados neste trabalho.

As arquiteturas reconfiguráveis de grão grande, apresentadas no capítulo 2, contribuíram para a análise de quais tipos de elementos de processamento são mais apropriados para um conjunto de problemas, permitindo escolher uma arquitetura existente ou provendo idéias para a construção de novas arquiteturas. A rede de interconexão de elementos de processamento é uma característica importante a ser levada em conta na escolha de uma arquitetura reconfigurável, pois ela pode limitar o desempenho dos elementos de processamento quando uma carga elevada de transferência de dados for exigida da arquitetura.

Processadores reconfiguráveis, apresentados no capítulo 3, mostraram diversas formas de fazer uso da lógica reconfigurável sem perder a facilidade de programação dos processadores. A maior parte destes utiliza a lógica reconfigurável como um coprocessador, o que resulta em um dos casos estudados desempenho 160 vezes superior que uma abordagem sem lógica reconfigurável. Ganhos tão significativos são atingidos quando é possível contornar o problema de baixo desempenho do processador em situações críticas, como repetição de um pequeno trecho de código em hardware.

Ainda que processadores reconfiguráveis e arquiteturas de grão grande sejam os principais aspectos a serem considerados para a construção de um sistema com elevado poder de processamento, é necessário que existam compiladores e ferramentas de CAD que auxiliem o processo de desenvolvimento. Os compiladores devem aceitar instruções de comunicação entre o processador e a lógica reconfigurável, de forma a permitir a troca de dados e as chamadas das diferentes funções implementadas em lógica reconfigurável. As ferramentas de CAD devem prover um ambiente para criar hardware para a arquitetura reconfigurável e suportar o co-desenvolvimento e a co-simulação de hardware e software.

Será feita na continuação deste trabalho um estudo aprofundado do estado da arte em NOCs e a implementação de uma NOC que permita a reconfiguração parcial e dinâmica dos módulos conectados a ela. Exemplos de módulos que serão substituídos da NOC são processadores, memórias e módulos de hardware dedicado. A NOC em si não será reconfigurada para que a mesma continue operando corretamente com o restante dos módulos em funcionamento. O trabalho será

implementado utilizando os dispositivos Virtex da Xilinx [XIL04] que permitem reconfiguração parcial e dinâmica do sistema.

5 REFERÊNCIAS BIBLIOGRÁFICAS

[ALT04] Altera, Inc. http://www.altera.com.

[ATH93] Athanas, P.; Silverman, H; "**Processor Reconfiguration through Instruction-Set Metamorphosis**". Computer, Volume: 26 (3), Mar. 1993, pp.11-18.

[BEC00] Becker, J.; Piontek, T.; Glesner, M.; "**DReAM: A Dynamically Reconfigurable Architecture for Future Mobile Communications Applications**". In: Field Programable Logic and Applications (FPL), 2000, pp. 312-321.

[BEC03] Becker, J.; Huebner, M.; Ullmann, M. "**Power Estimation and Power Measurement of Xilinx Virtex FPGAs: Trade-offs and Limitations**". In: Brazilian Symposium on Integrated Circuits and Systems Design (SBCCI), 2003, pp. 283-288.

[BEN02] Benini, L.; De Micheli, G. "**Networks on Chips: a New SoC Paradigm**". Computer, volume: 35 (1), Jan. 2002, pp. 70-78.

[BER89] Bertin, P.; Roncin, D.; Vuillemin, J. "**Introduction to Programmable Active Memories**". Systolic Array Processors, Prentice Hall, Englewood Cliffs, N.J., 1989, pp. 300-309.

[DER02] De Rose, C.; Navaux, P. "**Arquiteturas Paralelas**". Brasil, Porto Alegre, Sagra Luzzato, 2002, pp. 75-85.

[EBE95] Ebeling, C.; McMurchie, L.; Hauck, S.; Burns, S. "Placement and Routing Tools for the Triptych FPGA". IEEE Transactions on Very Large Scale Integration Systems, Volume: 3 (4), Dec. 1995, pp. 473-482.

[EBE96] Ebeling, C.; Cronquist, D.; Franklin, P. "**RaPiD - Reconfigurable Pipelined Datapath**". In: Field Programmable Logic and Applications (FPL), 1996, pp. 126-135.

[GOL99] Goldstein, S.; Schmit, H.; Moe, M.; Budiu, M.; Cadambi, S.; Taylor, R.; Laufer, R. "**PipeRench: a Coprocessor for Streaming Multimedia Acceleration**". In: International Symposium on Computer Architecture (ISCA), 1999, pp. 28-39.

[HAR95] Hartenstein, R.; Kress, R. "**A Datapath Synthesis System for the Reconfigurable Datapath Architecture**". In: Asia and South Pacific Design Automation Conference (ASP-DAC), 1995, pp. 479-484.

[HAR97] Hartenstein, R.; Becker, J.; Herz, M.; Nageldinger, U. "**An Innovative Platform for Embedded System Design**". In: Architekturen von Rechensystemen (ARCS), 1997, pp. 143-152.

[HAR01] Hartenstein, R. "**A decade of reconfigurable computing: a visionary retrospective"**. In: Design, Automation and Test in Europe (DATE), 2001, pp. 642-649.

[HAU97a] Hauser, J.; Wawrzynek, J. "**Garp: a MIPS Processor with a Reconfigurable Coprocessor**". In: FPGAs for Custom Computing Machines (FCCM), 1997, pp. 12-21.

[HAU97b] Hauck, S.; Fry, T.; Hosler, M.; Kao, J. "**The Chimaera Reconfigurable Functional Unit**". In: FPGAs for Custom Computing Machines (FCCM), 1997, pp. 87-96.

[HEN90] Hennessy, J.; Patterson, D. "**Computer Architecture: A Quantitative Approach**". Estados Unidos, San Mateo, Morgan Kauffmann Publishers, 1990, 594 p.

[HEN03] Henkel, J. "**Closing the SoC Design Gap**". Computer, volume: 36 (9), Sep. 2003, pp. 119-121.

[HER01] Herz, M. "**High Performance Memory Communication Architectures for Coarse-Grained Reconfigurable Computing Architectures**". Tese de doutorado, Universidade de Kaiserslautern, Alemanha, 2001.

[JAC99] Jacob, J.; Chow, P. "**Memory Interfacing and Instruction Specification for Reconfigurable Processors**". In: Field Programmable Gate Arrays (FPGA), 1999, pp. 145-154.

[MAR99] Marshall, A.; Stansfield, T.; Kostarnov, I.; Vuillemin, J.; Hutchings, B. "**A Reconfigurable Arithmetic Array for Multimedia Applications**". In: Field Programmable Gate Arrays (FPGA), 1999, pp. 135-143.

[MIR96] Mirsky, E.; DeHon, A. "**MATRIX: A Reconfigurable Computing Architecture with Configurable Instruction Distribution and Deployable Resources**". In: FPGAs for Custom Computing Machines (FCCM), 1996, pp. 157-166.

[RAB97] Rabaey, J. "**Reconfigurable Computing: The Solution to Low Power Programmable DSP**". In: International Conference on Acoustics, Speech, and Signal Processing (ICASSP), 1997, pp. 275-278.

[RAD98] Radunovic, B. "**A Survey of Reconfigurable Computing Architectures**". In: Field Programable Logic and Applications (FPL), 1998, pp. 376-385.

[RAZ94] Razdan, R.; Brace, K.; Smith, D.; "**PRISC software acceleration techniques**". In: IEEE International Conference on Computer Design (ICCD), VLSI in Computer & Processors, 1994, pp. 145-149.

[RUP98] Rupp, C.; Landguth, M.; Garverick, T.; Gomersall, E.; Holt, H.; Arnold, J.; Gokhale, M. "**The NAPA Adaptive Processing Architecture**". In: FPGAs for Custom Computing Machines (FCCM), 1998, pp. 28-37.

[SAS02] Sassatelli, G.; Torres, L.; Benoit, P.; Gil, T.; Diou, C.; Cambon, G.; Galy, J. "**Highly scalable dynamically reconfigurable systolic ring-architecture for DSP applications**". In: Design, Automation and Test in Europe (DATE), 2002, pp. 553-558.

[SIN98] Singh, H.; Lee, M.; Lu, G.; Kurdahi, F.; Bagherzadeh, N.; Filho, E. "**MorphoSys: a Reconfigurable Architecture for Multimedia Applications**". In: Brazilian Symposium on Integrated Circuits and Systems Design (SBCCI), 1998. pp. 134-139.

[WAI97] Waingold, E.; Taylor, M.; Srikrishna, D.; Sarkar, V.; Lee, W.; Lee, V.; Kim, J.; Frank, M.; Finch, P.; Barua, R.; Babb, J.; Amarasinghe, S.; Agarwal, A. "**Baring It All to Software: Raw Machines**". IEEE Computer, volume: 30(9), Sep. 1997, pp. 86-93.

[WAZ93] Wazlowski, M.; Agarwal, L.; Lee, T.; Smith, A.; Lam, E.; Athanas, P.; Silverman, H.; Ghosh, S. "**PRISM-II Compiler and Architecture**". In: FPGAs for Custom Computing Machines (FCCM), 1993, pp. 9-16.

[WIR94] Wirthlin, M.; Hutchings, B.; Gilson, K.; "**The Nano Processor: a Low Resource Reconfigurable Processor**". In: FPGAs for Custom Computing Machines (FCCM), 1994, pp. 23-30.

[WIR95] Wirthlin, M.; Hutchings, B. "**A Dynamic Instruction Set Computer**". In: FPGAs for Custom Computing Machines (FCCM), 1995, pp. 99-107.

[WIT96] Wittig, R.; Chow, P. "**OneChip: an FPGA Processor with Reconfigurable Logic**". In: FPGAs for Custom Computing Machines (FCCM), 1996, pp. 126-135.

[XIL04] Xilinx, Inc. http://www.xilinx.com.