Redes de Interconexão Multiestágios em Plataformas Reconfiguráveis para Sistemas Embarcados

Júlio Cesar Goldner Vendramini[1], Ricardo Ferreira[1]

[1]Depto de Informática – Universidade Federal de Vicosa 36570-000, Vicosa, Brazil

{julio.vendramini,ricardo}@ufv.br

Abstract. *This work presents an evaluation of multistage interconnection network (MIN) for embedded systems, mapped on FPGAs. The total area in LUTs, the latency, the data width, the number of reconfiguration bits as well as the switch complexity are analyzed. The experimental results show that the FPGA synthesis tools are enable to capture the MIN regularity, the area and the reconfiguration bits have complexity O(N Lg N), and the latency has complexity O(Lg N). Therefore, MINs are good candidates as an interconnection network for parallel embedded systems on FPGA.*

Resumo. *Este trabalho apresenta uma avaliação das redes multiestágio como mecanismo de comunicação para sistemas embarcados sintetizados em FPGA. As redes são comparadas levando em consideração área em LUTs, a latência, os bits de configuração, a capacidade dos comutadores e a largura da palavra. Como resultado é mostrado que a regularidade das redes é capturada pelas ferramentas de síntese de FPGA, apresentando complexidade O(N Lg N) em área e em bits de configuração e O(Lg n) em latência, mostrando a viabilidade do uso das redes em sistemas embarcados.*

1. Introdução

Sistemas embarcados e computação paralela são dois grandes desafios da atualidade. As redes de interconexão tem um papel fundamental na implementação de soluções paralelas. Além disso, os sistemas embarcados demandam desempenho e flexibilidade para se adaptar ao mercado muito dinâmico. Os sistemas reconfiguráveis como os FPGAs são uma alternativa interessante. Entretanto a alta flexibilidade dos FPGAs elevam a complexidade da síntese e mapeamento, aumentando o tempo de projeto dos sistemas. Este trabalho se enquadra em arquiteturas paralelas de sistemas embarcados compostas por vários elementos de processamento (EP) e/ou módulos de memória. Este trabalho apresenta as redes multiestágios como uma alternativa para interconexão dos EPs e memórias em FPGAs.

As redes multiestágios já foram usadas em sistemas de emulação e prototipação baseados em Multi-FPGAs [Barbie,1999]. Para emular um sistema eram necessários vários chips de FPGAs e redes de interconexão entre eles. Ao invés de usar chips crossbar, que tem custo $O(n^2)$, alguns trabalhos propuseram dedicar um ou mais FPGAs as interconexões, aumentando a flexibilidade. Este artigo difere dos anteriores ao propor o uso das redes multiestágios internamente no FPGA, em conjunto com elementos de processamento e memória para implementação eficiente de sistemas embarcados.

Este texto está estruturado da seguinte maneira. Primeiro são apresentados os trabalhos correlatos com multiestágio e suas implementações em VLSI e FPGA. Na seção 3, as redes são classificadas e os algoritmos de roteamento são apresentados. Na seção 4 apresentamos uma série de experimentos de implementação das redes em FPGA. A seção 5 finaliza com as principais conclusões e trabalhos futuros.

2. Trabalhos Correlatos

As redes multiestágios [Clos,1953][Benes,1965] foram inicialmente propostas nas décadas de 50/60 para o sistema de telefonia. Na década de 70, redes mais simples como as redes Omega [Lawrie,1975], foram propostas para interconexão de processadores e memória em máquinas paralelas MIMD e SIMD. O custo da rede Omega era menor assim como sua capacidade de roteamento. Porém alguns padrões de comunicação como os algoritmos de FFT e ordenação são realizados de forma eficiente. Durante os anos 70/80 até o início dos anos 90, muitos estudos dos tipos, classes de equivalência para redes bloqueantes e rearranjáveis [Wu,1980] [Yeh,1992], bem com algoritmos paralelos de roteamento foram realizados [Yeh,1992] e outras redes equivalentes sugiram como as fat-tree [Leiserson,1985]. A rede se mostrava um mecanismo paralelo de comunicação e tolerância a falhas, porém como programá-la rapidamente, era um dos grandes desafios para construção das máquinas paralelas. Muitos trabalhos teóricos sobre os limites inferiores do problema foram feitos [Çam,1999], porém estes algoritmos paralelos não foram implementados. Como redes comutadoras de pacotes, as multiestágios foram usadas em redes ATM e supercomputadores no final dos anos 90 com comutadores ópticos. Na comutação de pacotes, a rede é auto-roteável, porém pode gerar congestionamento das filas dos comutadores intermediários quando a carga de trabalho se eleva.

Durante os anos 80 e 90 foram feitos estudos e implementações físicas em VLSI para avaliar o potencial de implementações dos comutadores e conexões das redes multiestágios em silício [Dinizt,1999][Dehon,2000]. Apesar da complexidade O(N Lg N) em termos de número de comutadores, as implementações VLSI mostram um custo $O(N^2)$ em área de silício [Dinizt,1999], e exigiam também conexões com multi-camadas. Para circuitos reconfiguráveis como FPGA, as redes também foram avaliadas como um mecanismo de interconexão programável. Em [Dehon,2000], uma nova arquitetura física para FPGA com o uso de multiestágios para interligar as LUTs foi proposta, ao invés dos tradicionais elementos chaveadores em 2D que interligam segmentos de barramentos em clusters (tipo ilha). Entretanto, a construção eficiente do circuito depende de recursos multi-camadas para os fios. Ademais, nenhum FGPA comercial foi construído com redes multiestágios e continuam a ser fabricados no estilo ilha.

Com a popularização dos FPGAs nos anos 90, alguns trabalhos propunham sistemas grandes compostos de vários chips FPGA e uma arquitetura de comunicação entre eles, conhecidas como arquiteturas multi-FPGA. Ao contrário de construir um novo FPGA no nível físico, a proposta é usar os FPGAs comerciais. Além de usá-los para realizar a computação, como por exemplo a emulação de circuitos, os FPGAs eram usados também para interconexão. A ideia era uma rede de interconexão no nível lógico, sintetizando a multiestágio sobre o FPGA, como as redes Clos [Barbie,1999], Folded-

Clos [Lin,1997], redes crossbar parciais [Ejnioui,1999]. Neste caso, o FPGA inteiro é usado para implementar uma rede e interligar os outros chips de FPGA que realizam computação. São usadas redes Clos que tem um custo maior (3n Lg n) estágios com comutadores 2x2 ou apenas três níveis e comutadores NxK, MxM e KxN.

Recentemente, trabalhos propõem o uso de redes em arquiteturas reconfiguráveis de grão grosso [Tanigawa,2008][Ferreira,2008][Ferreira,2009] e em sistema com multiprocessadores MPSoCs [Neji,2008]. Em [Tanigawa,2008], uma arquitetura em linhas com redes *Benes* sintetizadas em VLSI é proposta, ou seja, propõe construir fisicamente a rede em circuito para interligar unidades funcionais. Os resultados mostraram que o layout é bem compacto, cerca de 1.000 vezes menor que um processador multicore. Mesmo com um tempo de relógio 4 vezes maior, a versão multiestágio apresenta o mesmo tempo de execução para aplicações multimídia quando comparada aos processadores Multi-core. Porém o mapeamento das aplicações é feito manualmente na arquitetura. Semelhante a abordagem anterior, porém acoplado a um processo de tradução binária, onde em tempo de execução o código binário da aplicação é mapeado na arquitetura reconfigurável, o uso de rede multiestágio foi proposto em [Ferreira,2008]. Neste trabalho, as redes *Omega* [Lawrie,1975] são usadas para interligar em linhas de um arranjo de unidades funcionais acopladas a um processador MIPS. Um algoritmo de posicionamento e roteamento é feito em hardware, mostrando que o roteamento pode ser realizado em tempo de execução. Este trabalho também demanda a construção física em VLSI de um circuito reconfigurável com as redes.

Posteriormente, em um trabalho anterior dos autores [Ferreira,2009], as redes *Omega* foram usadas em FPGAs no nível lógico. A ideia era melhorar a capacidade de roteamento de um grid de ALUs para grafos de fluxos de dados. Os grafos eram mapeados no grid que não possui capacidade de roteamento, apenas comunicação direta dos vizinhos (norte,sul,leste e oeste). Quando dois vértices do grafo de fluxo de dados não eram vizinhos na arquitetura, a comunicação deles é roteada na multiestágio. Este trabalho mostrou dois resultados interessantes. Primeiro, as multiestágios, como recurso de roteamento em nível lógico em FPGA, apresentaram um área menor que a integração de capacidade de roteamento nas unidades do grid. Segundo, mesmo usando uma rede com conflitos (redes *Omega*), como a demanda de roteamento era da ordem de 30% dos sinais, a multiestágio realiza com sucesso o roteamento. Ademais, o algoritmo de roteamento, mesmo implementado em software, executa em milisegundos que possibilita sua integração em ambientes de compilação Justi-in-Time. Foram usadas redes com largura de 32 bits para as palavras de dados. Mais recentemente, as redes em FPGA juntamente com elementos de computação [Ferreira,2010], foram usadas em Bioinformática para modelar grafos de rede reguladoras de genes dinamicamente, os resultados preliminares apresentaram aceleração de duas ordens de grandeza. No caso das redes de genes, a largura de 1 bit é suficiente para implementação dos modelos.

Recentemente, um trabalho avalia a área e o desempenho dos FPGAs com as redes multiestágios em sistemas multiprocessadores do tipo Multiprocessors System on Chip (MPSoCs) [Neji,2008]. São sintetizados até 8 processadores miniMIPS, onde cada um possui uma memória de instrução dedicada e usam a rede para comunicar com 8 módulos de memória de dados. Os resultados mostram que o tempo de síntese e a área das multiestágios é bem menor quando comparada as redes crossbar. A complexidade da

rede é O(N Lg N) após a síntese. Porém não se diz qual a largura da palavra, os miniMIPs são em geral de 32 bits. Isoladamente as redes são avaliadas com até 64 entradas. Um algoritmo de multiplicação de matrizes é usado para validar o sistema.

Motivado por trabalhos anteriores dos autores [Ferreira,2009][Ferreira,2010], e o estado atual dos FPGAs, este trabalho difere dos anteriores ao propor uma avaliação de diversos aspectos das redes sintetizadas sobre FPGAs. Diferente do trabalho [Neji,2008], buscamos uma caracterização e estimativa de custo para diferentes larguras de bits e redes bloqueantes e rearranjáveis. Em [Neij,2008] apenas redes bloqueantes são avaliadas. Ademais, redes com até 512 entradas são avaliadas. O trabalho de [Neji,2008] é baseado em ferramentas da Altera, este trabalho é baseado em ferramentas Xilinx. Os novos FPGAs possuem um grande volume de LUTs, multiplicadores embarcados e módulos de memória. Por exemplo, uma Virtex6 possui aproximadamente 150.000 Luts e 416 módulos de memória. A ideia deste trabalho é explorar qual o espaço ocupado pela rede como um recurso de comunicação reconfigurável no FPGA. O espaço restante pode ser usado para implementar sistemas embarcados paralelos com elementos de processamento e/ou memória. Serão avaliadas: a área em LUTs, os comutadores, o roteamento, a latência, os bits de configuração e a largura da palavra. Este trabalho visa uma caracterização das interconexões gerando opções de projeto para sistemas embarcados.

3. Redes Multiestágios

As interconexões podem ser diretas ou indiretas. Os elementos de computação são interligados ponto a ponto em conexões diretas como por exemplo em uma malha bidimensional. Nas conexões indiretas, os elementos se comunicam através de comutadores. As conexões diretas são estáticas, rápidas, tem custo baixo mas perdem em flexibilidade. As conexões indiretas são dinâmicas e oferecem flexibilidade. As redes crossbar são um exemplo de rede dinâmica, porém o custo em comutadores é $O(N^2)$ para comunicar N elementos. As redes multiestágios surgiram nos anos 50/60 como uma alternativa com o custo O(N lg N) em comutadores, e desde então tem sido muito estudadas.

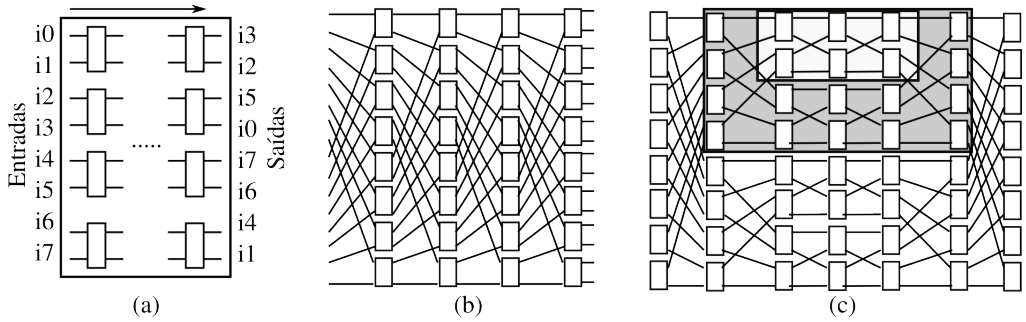

Figura 1. Multiestágios: (a) Permutação de Entrada; (b) Omega; (c) Benes.

Uma rede multiestágio possui N entradas e N saídas, é formada por uma série de colunas de comutadores (estágios), onde os comutadores do estágio j se conectam aos comutadores do estágio j+1. Formalmente a rede é definida pela tupla M(N,E,R,P,B), onde N é o número de entradas e saídas, E é o número de estágios, R é o radix do comutador, P é o padrão de permutação de bits entre os estágios e B é o número de configurações para cada comutador. Na seção 3.1 iremos classificar as redes e na seção

3.2 apresentamos os algoritmos de roteamento.

3.1. Classificação

As redes são classificadas como bloqueantes, rearranjáveis, estritamente não bloqueantes e não bloqueantes. Uma conexão completa da rede equivale a uma permutação de entrada como ilustra a Figura 1(a), onde as entradas i_0 à i_7 se conectam as saídas o_3,o_7,o_1,o_0,o_6,o_2,o_5 e o_4, respectivamente.

Uma rede é bloqueante quando não consegue rotear todas as permutações de entrada/saída. A rede *Omega* ou *Butterfly* são exemplos de redes bloqueantes. A Figura 1(b) ilustra uma rede Omega [Lawrie,1976] com 16 entradas/saídas. Se considerarmos comutadores 2x2, a rede Omega terá Lg_2 N estágios, no exemplo da Figura 1(b) tem 4 estágios. Para facilitar a nomenclatura, os termos logaritmos são da base 2. O número de comutadores é N/2*Lg N e o número de comutadores que um sinal tem que atravessar entre a entrada e a saída da rede é Lg N, ou seja, o atraso é O(Lg N) e o custo em comutadores O(N Lg N). O padrão de conexão entre os estágios que determina o tipo de rede. Estas redes são conhecidas com redes de permutação de bits. No caso da rede *Omega*, o padrão de bit é uma rotação à esquerda no endereço da linha, também conhecido como *perfect-shuffle*. Por exemplo, a linha 1 ou 000**1** em binário irá conectar a linha 2 ou 00**1**0 em binário como ilustrado na Figura 1(b), ou seja a linha $x_3x_2x_1x_0$ conecta a linha $x_2x_1x_0x_3$ para 16 entradas. Depois de realizada a rotação, o sinal passa pelo comutador do estágio, que pode trocar o último bit, por isso a rede *Omega* também é conhecida com *Shuffle-Exchange*. Apesar de bloqueante, a rede *Omega* realiza vários padrões com conectar a entrada i_j à saída o_{j+t} onde t é um inteiro e representa uma translação da entrada, outros padrões são detalhados em [Lawrie,1975]. A *Butterfly* conecta a linha $x_{n-1}...x_i...x_1x_0$ na linha $x_{n-1}...x_0...x_1x_i$ ou seja troca o bit x_i pelo x_0 em função do estágio da rede. Tanto a *Omega* quando a *Butterfly* são conhecidas com redes de *Bayan* onde uma entrada pode alcançar qualquer saída, e existe apenas um único caminho para rotear uma entrada em uma saída. Esta característica simplifica o roteamento que será detalhado na seção 3.2. Vários padrões de redes bloqueantes foram propostos nos anos 70, e no inicio da década de 80 foi demonstrado que estes padrões eram equivalentes [Wu,1980].

Uma rede é rearranjável se consegue realizar todas as permutações, desde de que as conexões já estabelecidas possam ser rearranjadas entre alguns entradas e saídas da rede para não gerar conflitos. A rede *Benes* [Benes,1965] ilustrada na Figura 1(c) é um exemplo de rede rearranjável. Entretanto o custo da rede *Benes* é o dobro da rede *Omega*, tem pelo menos 2*(Lg N)- 1 estágios, o que aumenta também o atraso. Os algoritmos de roteamento serão discutidos na seção 3.2. Assim como as redes bloqueantes, as redes rearranjáveis são organizadas em classes [Yeh,1992].

A rede é estritamente não bloqueante, quando as requisições de conexão entre entrada e saída chegam de forma assíncrona e é sempre possível roteá-las. A rede será não bloqueante se for necessário respeitar um protocolo de roteamento para não alterar as conexões pre-estabelecidas. A rede *Clos* é um exemplo de rede não bloqueante. Entretanto, o custo dos comutadores aumenta significativamente, como ilustra a Figura

2(a) que apresenta uma rede *Clos* genérica com 3 estágios proposta em [Clos,1953]. O primeiro estágio tem m comutadores n x k, o segundo estágio tem k comutadores m x m e o último estágio tem m comutadores k x n, para n entradas/saídas.

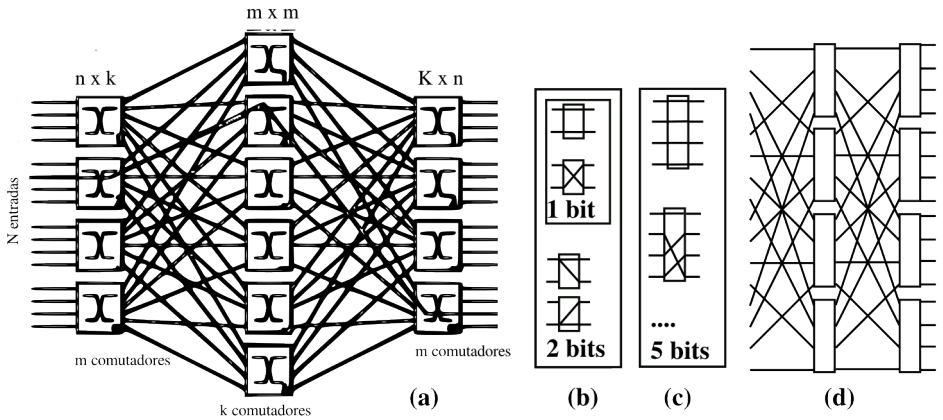

Figura 2. Multiestágios e Comutadores: (a) Clos; (b) 2x2; (c) 4x4 (d) Radix4.

O comutador da rede pode ser simples como um comutador 2x2 (veja Figura 2(b)) ou mais complexo com MxK entradas e saídas como os comutadores da rede Clos (veja Figura 2(a)). Quando maior o comutador, menor será o número de estágios. Ou seja, tem um custo benefício entre a complexidade do comutador e o número de estágios. Para comutadores 2x2 ou radix 2, uma rede bloqueante terá Lg_2N estágios como vimos para a rede Omega na Figura 1(b). Agora, se usarmos comutadores 4x4 ou radix 4, o número de estágios será Lg_4N como ilustrado na Figura 2(c) e na Figura 2(d) para uma rede Omega radix 4 com 16 entradas que terá apenas 2 estágios. Um comutador simples 2x2 tem 1 bit de configuração se opera no modo direto ou cruzado e 2 bits de configuração se faz um broadcast local (veja Figura 2(b)). A configuração broadcast local serve para broadcast ou multicast global, quando uma entrada da rede é conectada a um subconjunto de saídas da rede. Porém, o uso de multicast dobra o número de bits para radix 2. O comutador 4x4 terá 24 configurações simples 1 para 1 (5 bits) e 16 bits se considerarmos todas as possibilidade com multicast. A Tabela 1 ilustra o número de bits para diversos tamanho de redes *Omega*.

Tabela 1: Número de Bits de Configuração para os comutadores em Redes Omega com e sem capacidade de multicast. NA – não se aplica.

Tamanho N	Omega 2x2	Omega 2x2 Multicast	Omega 4x4	Omega 4x4 Multicast
8	12	24	NA	NA
16	32	64	40	128
32	80	160	NA	NA
64	192	384	240	768
128	448	896	NA	NA
256	1024	2048	1280	4096

Podemos observar que a radix4 precisa de mais bits de configuração, e fica restrita a valores de N potência de 4. Uma memória de configuração da rede irá armazenar os bits dos comutadores. Para uma rede de 256 entradas e radix2 temos 1024 bits ou 128 bytes para cada configuração da rede.

A rede pode ser usada para conectar elementos como ALUs [Ferreira,2009], elementos de processamento com recursos acoplados (registros, controle, alu) ou processadores simples [Neji,2008]. Pode ser usada para compartilhar recursos como multiplicadores ou módulos de memória, para acesso paralelo a módulos de memória. Se usada como uma rede global para interligar todas as ALU, simplifica o mapeamento, uma vez que pode-se posicionar em qualquer ALU que a rede fará o roteamento global. As redes já foram usadas para comutação por pacotes (ATM, supercomputadores, NoC). Neste trabalho avaliamos comutações por circuito mais adequadas para conectar ALUs ou elementos de processamento simples. O trabalho para MPSoCs usa comutação por pacotes [Neji,2008] entre os processadores miniMIPS. A Figura 3 ilustra diversas opções de arquiteturas onde as redes podem ser usadas. A largura da palavra é o número de bits que cada linha de dados transmite. Redes com 1 bit podem ser úteis para computação modeladas como autômatos celulares ou para sincronização dos elementos de processamento, enviando sinais de controle no lugar de sinais de dados.

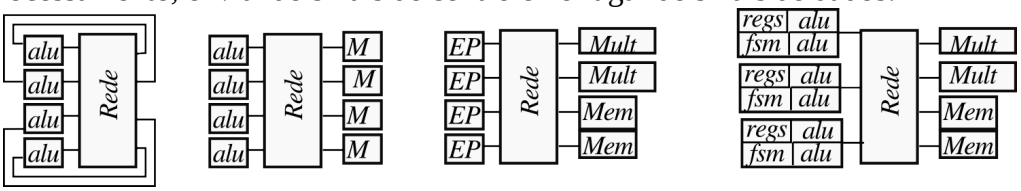

Figura 3. Diferentes Arquiteturas com redes de interconexão

3.2. Roteamento

O roteamento em multiestágio depende do tipo de padrão de bits entre os estágios e do tipo da rede. Nesta seção iremos descrever o roteamento em redes bloqueante *Omega* com e sem estágios extras e o roteamento em redes rearranjáveis.

A rede *Omega* é conhecida como auto-roteável, e pode ser usada com comutação por pacote baseado nos endereços de origem e destino. Se calcula um XOR dos endereços, e o resultado é percorrido bit a bit do mais significativo ao menos significativo, se o bit for 1, o comutador liga cruzado, se o bit for 0 o comutador ligado direto. Cada par origem destino tem um único caminho. Um conflito ocorre quando dois pares de entradas/saídas passam pela mesma linha de um comutador em um determinado estágio. Para verificar os conflitos, o roteamento pode ser feito usando janelas. Suponha uma palavra de roteamento formada pela concatenação dos endereços de origem e destino, ou seja, $O_{n-1}O_{n-2}....O_1O_0D_{n-1}D_{n-2}...D_1D_0$. Para facilitar a explicação vamos supor uma rede com 16 entradas como a ilustrada na Figura 4(a). Uma janela de 4 bits irá deslocar sobre a palavra de roteamento. A entrada 1 ou em binário 0001 é conectada a saída 6 ou em binario 0110, fazendo o caminho 0**0010**110, 00**0101**10, 000**1011**0 e 0001**0110**, onde a janela é mostrada em negrito. A janela representa a linha em cada estágio. Neste exemplo no primeiro estágio passará pela linha 2, depois pela linha 5, depois pela linha 11 e finalmente a linha 6 de destino. Quanto um outro par entrada e saída, tenta conectar pode ocorrer conflito se for passar pela mesma linha no mesmo

estágio. Por exemplo, a entrada 9 não pode se conectar a saída 4, pois gera a palavra 10010100, que terá conflito no primeiro estágio 1**0010**100 na linha 2, como ilustra a Figura 4(a).

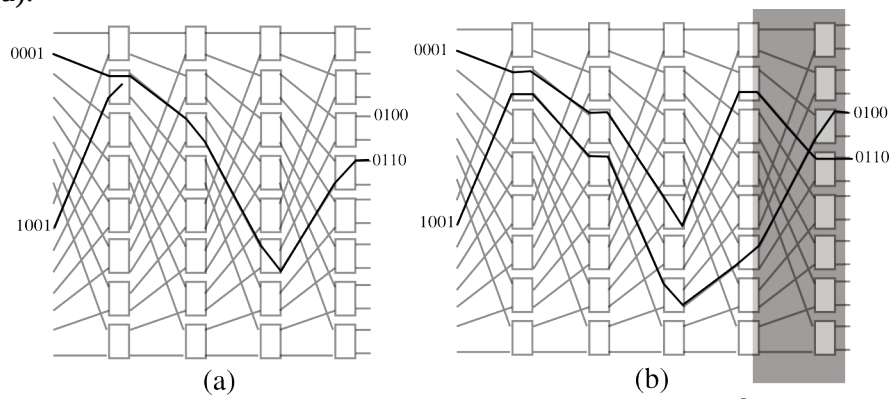

Figura 4. Roteamento em Rede Omega: (a) Caminho Único e Conflito; (b) Estágio Extra.

Uma solução para melhorar a capacidade de roteamento da Omega sem aumentar muito o custo é acrescentar estágios extras. Ao adicionar um estágio, no lugar de um caminho teremos dois caminhos. Cada estágio, dobra o número de caminhos. O roteamento é semelhante. Agora temos um bit a mais no caminho. Para conectar a entrada 0001 na saída 0110, teremos a palavra 0001X0110, onde X pode ser 0 ou 1. Suponha 0. Teremos o caminho 0**0010**0110, 00**0100**110, 000**1001**10, 0001**0011**0 e 00010**0110**, ou seja passa pelas linhas 2, 4, 9, 3 e finalmente 6. Para ligar sem conflito a entrada 9 na saída 4, pode-se usar X=1, no roteamento e nenhum conflito é gerado, como ilustra a Figura 4(b). A medida que a rede cresce aumenta a probabilidade de conflito. Apesar de vários estudos, ainda é um problema em aberto determinar a capacidade de roteamento com estágios extras de uma rede Omega [Gazit, 1989].

Uma implementação em hardware para o roteamento em redes Omega com estágios extras foi apresentada recentemente em [Ferreira,2008]. Para cada comutador, uma unidade simples de controle é gerada. As unidades são interligadas seguindo o padrão da multiestágio e um decodificador de prioridade faz a seleção de caminhos. Esta implementação pode ser realizada em FPGA em ambientes dinâmicos. Se a aplicação for compilada, um roteamento estático em software cuja a complexidade é polinomial O(N Lg N) pode ser usado.

Para redes rearranjáveis, a maioria dos algoritmos supõe conhecer a priori a permutação completa entrada/saída. Vários algoritmos sequenciais com o primeiro trabalho proposto por [Waksman,1968], e posteriormente várias versões paralelas ao longo das décadas de de 80 e 90 também foram propostas [Çam,1999]. A motivação para as versões paralelas era o uso da rede em computadores paralelos. A rede tem baixa latência O(Lg N), porém para executar o roteamento era necessário O(N Lg N). Apesar de várias propostas, os modelos de computadores que executavam os algoritmos paralelos eram teóricos, e não se encontra resultados de implementações em máquinas paralelas na literatura. O algoritmo básico pode ser executado rapidamente em um processador simples. A ideia é programar a rede, estágio por estágio, dos estágios externos para os mais internos. A cada passo se programa N comutadores e em Lg N passos se programa a rede. Um ponto em aberto, são implementações em hardware de

roteamento rearranjável para Benes.

A maioria dos algoritmos aborda apenas a conexão 1 para 1, de um entrada para uma saída sem multicast. Se considerarmos multicast além de aumentar o número de bits de configuração, aumenta-se a complexidade do roteamento. O multicast pode ser interessante em algumas aplicações.

4. Resultados Experimentais

As redes multiestágios foram descritas com um código parametrizado em VHDL em função do radix, número de estágios, bits de configuração, largura da palavra, etc. O objetivo foi avaliar o custo das redes e como as ferramentas de síntese de FPGA capturam uma descrição alto nível das redes. Como ferramenta usamos o ISE da XILINX versão 11 e como arquitetura alvo usamos uma Virtex6 que possui módulos de memória e multiplicadores embarcados. A Tabela 2 apresenta os resultados para redes Omega com vários tamanho e com a largura de bits variando de 1, 8, 16 e 32 bits.

Tabela 2: Área, Tamanho das Redes Multiestágios em Luts para N variando de 32 a 512 entradas/saídas e com largura de bits: 1, 8, 16 e 32 bits. Sintetizado em uma Virtex6 com 150730 Luts.

	32	64	128	256	512
1 bit	392	425	1108	1838	4017
8 bits	1227	2443	6039	12592	29107
16 bits	2251	4747	11501	24880	57779
32 bits	4299	9355	22765	49456	115123

Podemos observar da Tabela 2, que o espaço ocupado no FPGA para a rede *Omega* em função do tamanho da entrada cresce com a complexidade O(N Lg N), ou seja, a ferramenta de síntese captura a regularidade da rede. Em termos de espaço, vemos que uma rede de 512 entradas com largura de 32 bits ocupa 70% do FPGA. Entretanto, a complexidade de um sistema com 512 conexões de 32 bits é bem elevada. Para outros contextos, como uma rede com 128 conexões de 32 bits, apenas 15% do FPGA é ocupado, sendo que o restante pode ser usado para implementação de unidades de processamento. Outro ponto é que ao aumentar de 1 bit para 8 bits, a área em LUTs aumentou em média apenas 5,7 vezes, ao aumentar para 16 bits em relação a 1 bit, a área aumentou em 11 vezes em média, e para 32 bits o aumento médio foi de 22 vezes. Mostrando que a regularidade da rede é um aspecto importante capturado pela síntese. Além disso, as redes com largura pequena em bits apresentam um custo proporcionalmente maior.

A Tabela 3 mostra os resultados de latência para as redes da Tabela 2. Podemos observar que a latência não varia com a largura de bits, apenas o número de estágios. A cada coluna estamos acrescentando um estágio a mais na rede, ao dobrar o seu tamanho. Por exemplo, a rede de 32x32 tem 5 estágios e latência de 1,97 ns. Ao dobrar para uma rede de 64x64 teremos 6 estágios e a latência de 2,28, ou seja, 0,33 ns para atravessar um estágio a mais. Podemos observar que a latência aumenta em média 0,34 ns por

estágio, ou seja, a rede tem um comportamento linear em atraso. Se adicionarmos 1 estágio, a latência aumenta 0,34 ns. Este recurso pode ser usado para aumentar a capacidade de roteamento da rede com estágios extras. Cada estágio extra dobra o número de caminhos para um par de entrada e saída. Entretanto, uma rede bloqueante pode ter no máximo 2 Lg N estágios, mais estágios não alteram a capacidade de roteamento.

Tabela 3: Latência (ns) Redes Multiestágios para N variando de 32 a 512 entradas e saídas e com largura de bits: 1, 8, 16 e 32 bits. Sintetizado em uma Virtex6.

	32	64	128	256	512
1 bit	1,97	2,28	2,64	2,95	3,32
8 bits	1,97	2,28	2,64	2,95	3,32
16 bits	1,97	2,28	2,64	2,95	3,32
32 bits	1,97	2,28	2,64	2,95	3,32

Outro ponto importante é o numero de módulos de memória para armazenar as configurações. A Tabela 4 apresenta os resultados. Uma virtex6 tem 416 módulos, para a configuração de 512 são usados 72, ou seja, apenas 17% dos módulos. A primeira coluna com redes com 32 entradas usa LUTs para armazenar a configuração, enquanto que as outras colunas com redes de 64 à 512 entradas, a configuração é armazenada nas memórias embarcadas. A ferramenta da Xilinx faz a escolha pelas LUTs para reduzir o uso dos recursos do FPGA na rede de 32, porém para as outras possibilidades, a ferramenta verifica que os módulos são mais adequados para armazenar as configurações. Cada memória pode armazenar até 512 linhas, ou seja, sem aumentar o custo da implementação, podemos gerar 512 configurações diferentes de interconexão e programá-las no FPGA estaticamente ou dinamicamente. Ou seja, uma computação que envolva até 512 padrões diferentes de comunicação entre os elementos de processamento e memória pode ser configurada. No caso de uma rede com 128 entradas, usamos apenas 14 módulos, e 17% das LUTs do FPGA, sendo que praticamente todos os módulos de memória (mais de 400) juntamente com 80% do FPGA poderão implementar recursos de computação.

Tabela 4: Módulos de Memória sintetizado em uma Virtex6 com 416 módulos.

	32	64	128	256	512
Número de Módulos	0	6	14	32	72

Para avaliar o impacto da rede junto com unidade funcionais, a Tabela 5 apresenta resultados de síntese com ALUs de 8, 16 e 32 bits conectadas a uma rede com 32 entradas. Como cada ALU tem duas entradas e uma saída, a rede tem as 32 saídas conectadas as entradas das ALU, e as 16 saídas das ALUs juntamente com 16 entradas externas são conectadas a entrada da rede. As entradas externas servem para alimentar as ALUs. Podemos observar na Tabela 5, que o tamanho da rede em separado pode ser usado com uma base de estimativa da área ocupada. Ao realizar a síntese em conjunto a

área total foi próxima da estimativa da soma em separado da área das ALUs e da rede.

Tabela 5: Área das ALU sintetizadas em separado e em conjunto com uma rede Omega de 32 entradas.

largura	8bits	16 bits	32 bits
Área em LUTs das 16 ALUs	1824	3824	7920
Área em LUTs 16 ALUs + Rede sintetizado em separado	2848	5872	12016
Área em LUTs das ALUs + Rede sintetizado em conjunto	2910	6187	12228

Para aumentar o desempenho da rede podemos fazer uso de pipeline, ao inserir registros entre os estágios da rede. A Tabela 6 mostra os resultados de área e latência para a rede com registros entre os estágios. A vantagem é a redução do tempo de relógio para 0,5 ns. O aumento médio em área foi de 30% para uma rede com largura de 8 bits. Os módulos de memória podem armazenar a configuração da execução em pipeline, aumentando a vazão dos resultados.

Tabela 6: Rede com Pipeline (Registros entre os estágios). Largura de 8 bits

	32	64	128	256	512
Área com pipeline	1549	3370	7403	17104	37294
Área sem pipeline	1227	2443	6039	12592	29107
Acréscimo	26,00%	38,00%	23,00%	36,00%	28,00%
Atraso em ns	0,52	0,52	0,52	0,52	0,52

O padrão butterfly também foi sintetizado e apresentou resultados semelhantes a Omega. A radix4 ocupa um espaço menor. O tamanho da radix2 evolui com O(N Lg N), mostrando que a síntese em FPGA não aumenta a complexidade de custo em área da rede. Já o padrão radix4 com comutadores 4x4 reduz a área pois o número de estágios é $N\ Lg_4N$. A Tabela 7 apresenta os resultados para síntese com radix 4 para uma rede de 1 bit de largura e de 32 bits de largura. Os resultados são comparados com a rede radix2. Em alguns casos não se aplica (NA) pois o radix4 tem que ser potência de 4. Para a rede com largura de 1 bit, a radix4 ocupa uma área em LUTs duas vezes menor. Para a largura de 32 bits, a área da radix4 é em torno de 60% da área da radix2. A latência reduz de 10 a 20% com a radix4. Se for considerado multicast, o número de bits de configuração para a radix4 será o dobro da radix2. Trabalhos futuros são necessários para avaliar a número de conflitos em função do radix. Para máquinas paralelas, um trabalho recente apresentou uma versão plana dos comutadores agrupados por linha e como comunicação entre as linhas [Kim,2007]. A abordagem é baseada em redes com padrão *butterfly*. As interligações e o roteamento perde um pouco em regularidade e um estudo detalhado deve ser feito para avaliar esta alternativa.

Tabela 7: Rede Radix 4 e Radix 2. Largura de 1 e 32 bits. Área em LUTs e atraso em ns.

	16	32	64	128	256
1 bit, Radix2 – Área (LUTs)	-	392	425	1108	1838
1 bit, Radix4 – Área (LUTs)	32	NA	192	NA	1024
32 bits, Radix2 – Área (LUTs)	-	4299	9355	22765	49456
32 bits, Radix4 – Área (LUTs)	1024	NA	6144	NA	32768
Radix2 – Latência (ns)	-	1,97	2,27	2,64	3,20
Radix4 – Latência (ns)	1,46	NA	2,08	NA	2,69

Para redes rearranjáveis, como estudo de caso, a rede *Benes* foi sintetizada. Um código VHDL parametrizado também foi gerado. A Tabela 8 mostra os resultados para área em LUTs e o atraso em ns considerando uma largura de 8 bits para palavra. Podemos observar que ao dobrar o tamanho da rede, a latência aumenta em média 0,60 ns, pois em uma rede *Benes*, dobrar o tamanho significa incluir dois estágios a mais. Em relação a rede Omega bloqueante com largura de 8 bits, a área em LUTs é 70% maior. Para aplicações com compilação estática as redes Benes são interessantes pois não geram conflitos e o roteamento tem complexidade O(N Lg N) e pode ser facilmente incorporado no compilador.

Tabela 8: Rede Benes. Largura de 8 bits. Área em LUTs e atraso em ns.

	32	64	128	256	512
Área em Luts para 8 bits de largura	1792	4224	9984	23040	52224
Latência em ns	3.45	4,07	4,75	5,43	6,11

5. Conclusão

Este trabalho propõe um estudo dos custos de implementação de redes multiestágios em FPGA para implementação de sistemas embarcados paralelos. As redes podem ser descritas com código parametrizável em VHDL. As ferramentas de síntese em FPGA capturam a regularidade da rede com a complexidade O(N Lg N) em área e em bits de configuração bem como a latência com complexidade O(Lg N). Os módulos de memória embarcados nos FPGAs de última geração constituem uma boa alternativa para armazenar internamente várias configurações (até 512 em uma Virtex6). O espaço ocupado pelas redes é reduzido e é viável a síntese de várias unidades de processamento simples como ALUs. As unidades podem ser homogêneas ou heterogêneas, uma vez que a rede é um mecanismo global de roteamento, e não é necessário se aplicar algoritmos de posicionamento, uma vez que todas as posições são equivalentes. Trabalhos futuros incluem a implementação de algoritmos para sistemas embarcados com arquiteturas paralelas. Outro ponto importante é a implementação em hardware no FPGA de algoritmos de roteamento bloqueantes e rearranjáveis, bem como a implementação em softcore do roteamento internamente no FPGA em conjunto com as redes e unidades de

processamento para suporte de compilação Just-in-Time (JIT).

Referências

Barbie, J., e Reblewski, F. (1999) "Emulation System having a Scalable Multi-level Multi-Stage Hybrid Programmable Interconnection Network", US Patent 5907679.

Benes, V. (1965) “Mathematical Theory of Connecting Networks and Telephone Traffic”, Academic Press, New York.

Clos, C. (1953) “A study of non-blocking switch networks”, *Bell System Tech. J*. 32:407-425.

Çam, H. and Fortes, J. A. 1999. Work-Efficient Routing Algorithms for Rearrangeable Symmetrical Networks. IEEE Trans. Parallel Distrib. Syst. v10(7)

DeHon, A. (2000) “Compact, multilayer layout for butterfly fat-tree” In *Proceedings of the 12th ACM symposium on Parallel algorithms and architectures*, pp.206-215.

Dinitz, Y., Even, S., Kupershtok, R., and Zapolotsky, M. (1999). Some compact layouts of the butterfly. In 11Th *ACM Symposium on Parallel Algorithms and Architectures.*

Ejnioui, A., Ranganathan, N. (1999). Multi-terminal net routing for partial crossbar-based multi-FPGA systems. In Proceedings of the ACM/SIGDA Seventh international Symposium on Field Programmable Gate Arrays

Ferreira, R. S. ; Laure, M. ; Rutizig, M. ; Beck, A. C. ; Carro, L. (2008). “Reducing interconnection cost in coarse-grained dynamic computing through multistage network.” IEEE International Conference on Field Programmable Logic and Applications (FPL), p. 47-52.

Ferreira, R. S. ; Damiany, A. ; Vendramini, J. ; Teixeira, T. ; Cardoso, J. (2009). On Simplifying Placement and Routing by Extending Coarse-Grained Reconfigurable Arrays with Omega Networks. In: 5th International Workshop on Applied Reconfigurable Computing,

Ferreira, R. S. ; Vendramini, J. ; Carvalho, L. (2010). Simulação em FPGA de Redes Reguladoras com Topologia Livre de Escala. In: VI Jornadas sobre Sistemas Reconfiguráveis, Aveiro, Portugal.

Gazit, I. and Malek, M. (1989). On the Number of Permutations Performable by Extra-Stage Multistage Interconnection Networks. *IEEE Trans. Comput*. v38(2)

Kim, J., Dally, W. J., and Abts, D. (2007). Flattened butterfly: a cost-efficient topology for high-radix networks. In *Proceedings of the 34th Annual international Symposium on Computer Architecture* (San Diego, California, USA). ISCA '07. ACM, New York, NY, 126-137.

Lawrie, D.H. (1975) “Access and Alignment of Data in an Array Processor”, IEEE Trans. on Computers, V24 (12)

Leiserson, C. E. (1985) “Fat-trees: Universal networks for hardware efficient supercomputing.” IEEE Transactions on Computers, v34 (10) p. 892–901,

Lin,S., Lin,Y. e Hwang, T.(1997)“Net Assignment for the FPGA-Based Logic

Emulation System in the Folded-Clos Network Structure", IEEE Trans. CAD, v16(3)

Neji, B., Aydi, Y., Ben-atilallah, R., Meftaly, S., Abid, M., Dykeyser, J-L. (2008) Multistage Interconnection Network for MPSoC: Performances study and prototyping on FPGA, IEEE Design and Test Workshop (IDT)

Tanigawa, K., Zuyama, T., Uchida, T. , e Hironaka, T. (2008) “Exploring compact design on high throught coarse-grained reconfigurable architectures”, IEEE *Proceedings of the International Workshop on Field-Programmable Logic (FPL).*

Yeh, Y-M., e Tse-yun Feng, T-Y (1992) “On a Class of Rearrangeable Networks”, *IEEE Trans. on Computers*, v41(11), pp. 1361—1379.

Wu, C-L., Feng, T-Y. (1980). “On a Class of Multistage Interconnection Networks.” *IEEE Trans. Comput.* V29 (8), 694-702.

Waksman, A. (1968) “A Permutation Network”, J. ACM, vol. 15, no. 1, pp. 159-163