RCS-2: Projeto de uma Chave *Crossbar* Reconfigurável

Henrique C. Freitas Milene B. Carvalho Carlos A. P. S. Martins
Grupo de Sistemas Digitais e Computacionais
Instituto de Informática
Programa de Pós-graduação em Engenharia Elétrica
Pontifícia Universidade Católica de Minas Gerais
cota@pucminas.br, milene@ieee.org, capsm@pucminas.br

Resumo

Os equipamentos de redes apresentaram nos últimos anos avanços consideráveis em sua arquitetura de processamento e chaveamento de pacotes de dados resultando no surgimento de unidades de chaveamento dedicadas para trabalhar com o aumento de qualidade e demanda de serviço. Entre elas podemos citar a chave crossbar. No entanto, soluções reconfiguráveis para este tipo de unidade de chaveamento estão surgindo na literatura com o foco apenas no FPGA. Este artigo apresenta uma arquitetura de chave crossbar com dois níveis de reconfiguração, para o aumento da flexibilidade e desempenho. Os resultados apresentados neste artigo são referentes às simulações dos modelos em Redes de Petri e da síntese em FPGA da chave crossbar reconfigurável em relação ao modelo original de uma chave crossbar tradicional.

1. Introdução

Nos últimos anos as pesquisas envolvendo arquiteturas de interconexão [4, 5, 12] têm aumentado consideravelmente, muito em função da própria evolução da Internet e da grande demanda por serviços. Existe uma necessidade crescente de novos equipamentos de rede com alto desempenho para suprir as necessidades relativas à qualidade de serviço. Normalmente encontramos estes equipamentos em pontos de concentração nas empresas provedoras de acesso ou conteúdo.

Como exemplo podemos citar os roteadores [21, 24], que recebem toda a carga de trabalho de uma determinada instituição por se tratar muitas vezes do *gateway* de entrada de uma rede local. Outra situação típica que demanda arquiteturas de interconexão de alto desempenho [1] é referente aos ambientes SANs (*System Área Network*) [3], onde existem servidores e ambientes de armazenagem (*storage*) que precisam de um *switch* de alto desempenho.

As arquiteturas de computadores multi-processadas [5, 16, 19, 23] também precisam de uma rede de interconexão de alto desempenho para manter a comunicação entre os processadores. Atualmente as pesquisas relacionadas aos processadores de rede [4, 6, 7] (dedicados para redes de comunicação de dados) estão caminhando para projetos de arquiteturas *System-on-Chip* (SoC) e *Network-on-Chip* (NoC). Neste tipo de arquitetura [25] vários blocos, que não normalmente são externos e acessados através de interfaces do processador, são inseridos no mesmo chip (SoC). É necessário que se tenha uma rede de interconexão interna (NoC), para manter o alto desempenho de comunicação entre todos os blocos construtivos do *chip*.

A computação reconfigurável [13, 17] tem surgido como uma opção para alcançar um maior desempenho e flexibilidade para diversas arquiteturas inclusive chaves *crossbar*. Neste artigo mostramos o uso deste conceito no tópico 2 (proposta) e no tópico 3 (resultados). Os trabalhos correlatos estão descritos no tópico 4, logo após a apresentação dos resultados.

O objetivo principal do nosso artigo é apresentar a proposta da chave *crossbar* de 2 bits, verificando através dos resultados obtidos pela modelagem em Redes de Petri (RdP) [8] e pela síntese em FPGA (*Field Programmable Gate Array*) [14, 18, 26], que há um aumento de desempenho no uso desta proposta em relação ao modelo original de uma chave *crossbar* tradicional.

A motivação para este trabalho se deve ao fato da pequena quantidade de pesquisas relacionadas à chave *crossbar* reconfigurável, além de estarmos abordando dois níveis de reconfiguração, o que normalmente não

é apresentado em outros artigos, conforme será discutido em trabalhos correlatos.

2. Proposta da RCS-2

Uma chave *crossbar* tradicional (TCS – *Traditional Crossbar Switch*) [4, 5, 6] é uma matriz de conexões, que permite um conjunto de entradas se comunicarem com um conjunto de saídas através do fechamento dos nós intermediários ou pontos de interseção entre as linhas (entradas) e as colunas (saídas).

Neste tipo de chave *crossbar* existe um limitante de flexibilidade que impacta diretamente no desempenho. Uma linha ou coluna da chave *crossbar* deve possuir apenas um nó fechado. Não é permitido neste tipo de arquitetura que uma entrada envie dados simultaneamente a mais de uma saída, ou várias entradas enviando dados a uma mesma saída. Isto impossibilita um *broadcast* ou *multicast* [22].

Esta limitação está associada ao fato da chave *crossbar* ser uma unidade de chaveamento espacial, mas de implementação temporal de topologias. A solução que encontramos está no uso da computação reconfigurável para fazer com que uma chave *crossbar* também seja capaz de implementar topologias no espaço, evitando o atraso no chaveamento de um nó de conexão sempre que necessário.

No WSCAD 2004 foi apresentado a chave *crossbar* de 1 bit (RCS-1: *Reconfigurable Crossbar Switch*) [9]. Esta arquitetura é muito semelhante à arquitetura apresentada neste artigo, porém há uma mudança na quantidade de bits usados para reconfigurar as topologias implementadas sobre a RCS-2.

Antes de apresentar o funcionamento desta nova versão, é importante ressaltar a proposta de reconfiguração em dois níveis da RCS-2:

Nível 1: Neste primeiro nível de reconfiguração há uma mudança da implementação da chave *crossbar* no FPGA. Nesta reconfiguração é feita a mudança, por exemplo, do tamanho da arquitetura, alterando quantidade de entradas e saídas.

Nível 2: No segundo nível, ocorre a alteração da topologia implementada sobre a chave *crossbar*. Neste nível não há mudança no tamanho da chave, mas na implementação sobre a chave.

É importante ressaltar que reconfigurar é mudar a forma. Por este motivo o FPGA é programável e não reconfigurável. Não aumentamos o tamanho do FPGA, mas da implementação sobre o mesmo. Esta analogia pode ser aplicada nos dois níveis de reconfiguração apresentados e está ilustrada na figura 1.

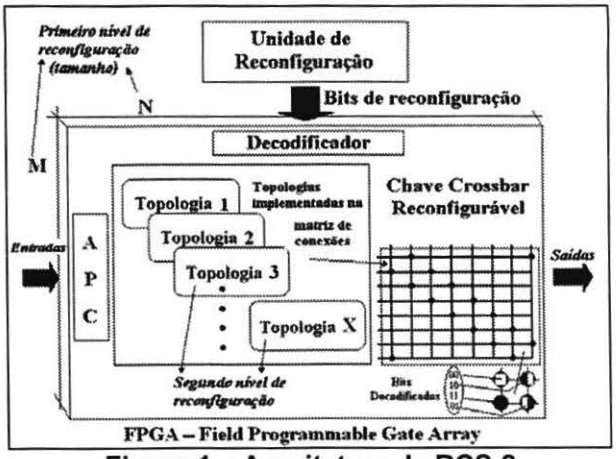

Figura 1 – Arquitetura da RCS-2

A arquitetura da chave *crossbar* de 2 bits possui os seguintes blocos construtivos:

- Matriz de conexões: responsável por oferecer o meio onde os nós serão fechados para estabelecer a comunicação entre entradas e saídas.
- Decodificador: bloco responsável por decodificar os bits de reconfiguração e gerar os bits para fechar ou abrir um nó.
- APC (Analisador de Pré-cabeçalho): Este bloco é responsável por analisar a informação de um pré-cabeçalho, acrescentado por um processador de rede para o encaminhamento de um pacote.

A unidade de reconfiguração é um bloco externo à RCS-2 é tem como funções, reconfigurar a implementação da RCS-2 sobre o FPGA e também a implementação das topologias sobre a RCS-2.

Cada nó da RCS-2 possui um registrador responsável por armazenar os bits decodificados para reconfiguração da topologia. Estes bits (figura 1) possuem o seguinte significado:

- Bits 00: Nó aberto. Somente a unidade de reconfiguração pode reconfigurar este nó.
- Bits 01: Nó aberto. A unidade de reconfiguração e instruções de um processador de rede podem alterar este estado para fechado.
- Bits 11: Nó fechado. Somente a unidade de reconfiguração pode reconfigurar este nó.
- Bits 10: Nó fechado. A unidade de reconfiguração e instruções de um processador de rede podem alterar este estado para aberto.

A unidade de reconfiguração pode usar qualquer uma das quatro combinações durante a reconfiguração. No entanto, um processador tem acesso somente às combinações 01 e 10. Isto significa que, se um nó estiver aberto com a combinação 00, o processador não

conseguirá, através de suas instruções, alterar este nó para fechado em 10.

Estas combinações de dois bits são uma mudança em relação à proposta apresentada no WSCAD 2004, onde a RCS-1 só trabalhava com 1 bit, sem oferecer a flexibilidade de reconfiguração para um processador. No entanto, as combinações 00 e 11 são de uso restrito da unidade de reconfiguração, mantendo a segurança em situações onde não é possível permitir que instruções de um programa em execução alterem uma topologia implementada.

É importante ressaltar, que as topologias são implementadas na RCS-2 através destas quatro combinações de bits, conforme está ilustrado pela figura 1 e 2 (topologia completa implementada no espaço).

3. Resultados

Neste tópico apresentamos os resultados obtidos através da modelagem e simulação em Redes de Petri [8] e da síntese em FPGA [14, 17, 18, 26].

3.1. Modelagem usando Redes de Petri

Para verificar o desempenho da proposta da RCS-2 foi realizada a modelagem em Redes de Petri da RCS-2 e da TCS. Foi simulada uma topologia completa com quatro máquinas, conforme figura 2, para as seguintes situações:

- *Multicast* da máquina 1 para máquinas 2, 3 e 4.
- *Multicast* da máquina 2 para máquinas 1, 3 e 4.
- *Multicast* da máquina 3 para máquinas 1, 2 e 4.
- *Multicast* da máquina 4 para máquinas 1, 2 e 3.

A RCS-2 é responsável por implementar esta topologia no espaço. As figuras 3 e 4 ilustram o modelo da RdP para a RCS-2.

Estas duas figuras são complementares. A figura 3 é referente ao modelo usado na simulação e também pode ser aplicada à RCS-1. A figura 4 é referente apenas à RCS-2, pois apresenta o modelo de uma RdP colorida para a chave *crossbar* reconfigurável de dois bits com as quatro combinações possíveis {00, 01, 10, 11}.

A figura 3 apresenta uma RdP com quatro entradas, onde cada uma delas é representada por quatro *places*, para que seja possível simular o *multicast*. Cada nó é representado por um rótulo (Ex: 11, 23, 33, 42) e cada saída por um *place*. Existem quatro *places* chamados de Conflito Sx, responsáveis por definir uma ordem quando mais de uma entrada está enviando dados para uma mesma saída simultaneamente (conflito).

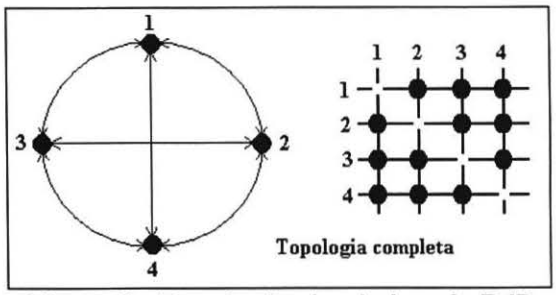

Figura 2 – Topologia simulada pela RdP

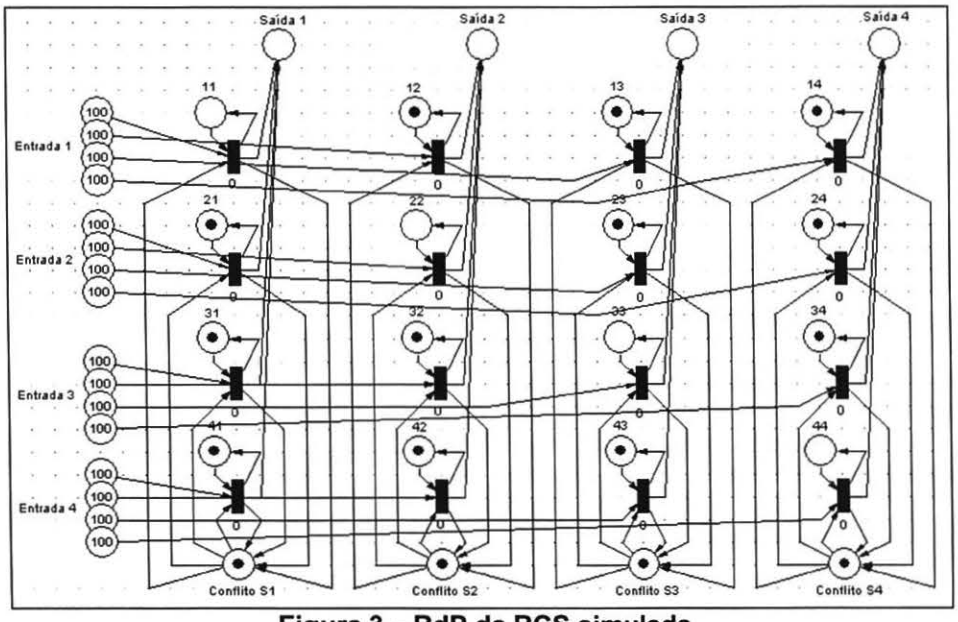

Figura 3 – RdP da RCS simulada

A figura 4 apresenta um modelo em RdP colorida que especifica formalmente o comportamento da reconfiguração com os quatro bits usados pela unidade de reconfiguração e os dois bits usados por um processador. Através deste modelo a unidade de reconfiguração pode alterar os dois bits do Nó 34, por qualquer uma das quatro combinações possíveis. O processador pode alterar usando as duas combinações {01 e 10}, sendo que esta ação somente será possível se não houver no Nó 34 as combinações {00 e 11}. Na figura o Nó 34 possui os bits 11, mas esta não é uma condição para a transição T2 disparar, portanto, o processador não consegue alterar estes bits. A entrada e saída são representadas respectivamente pelos *places* Entrada 3 e Saída 4. O *place* Nó 34 com a ficha 11 habilita o disparo da transição T3 e o envio da ficha E3 para o *place* Saída 4.

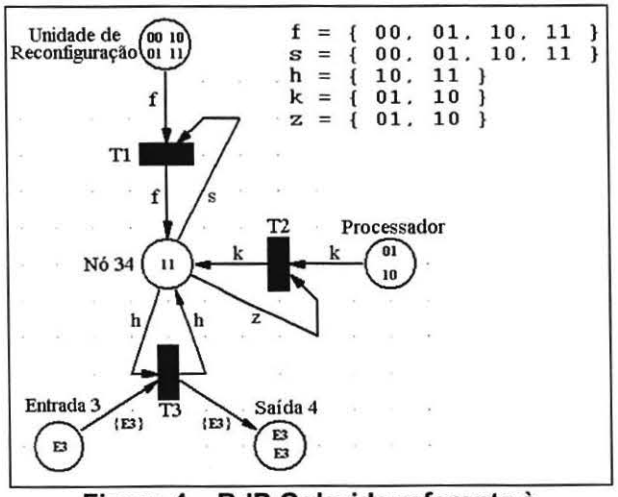

Figura 4 – RdP Colorida referente à reconfiguração para a RCS-2

A verificação do modelo da RdP da figura 3 foi realizada usando o simulador Visual Object Net ++ [20] e teve como cenário a seguinte situação (figura 2):

- Cada pacote de dados é uma ficha na RdP.
- Cada máquina terá que fazer um *multicast* para as demais máquinas da topologia, conforme quantidade de fichas (pacotes) da tabela 1.

Tabela 1 – Resultados da simulação da RdP

Qtd. fichas	TCS	RCS
	Tempo de simulação (s)	
10	44	30
20	86	60
50	216	150
100	432	300

O tempo de simulação da tabela 1 é referente ao tempo gasto por cada modelo de RdP para terminar de encaminhar todas as fichas de suas respectivas entradas para as saídas.

Com 10 fichas em cada uma das quatro entradas, a simulação termina quando cada uma das quatro saídas possuir 30 fichas. Este procedimento foi adotado para as demais quantidades de fichas (20, 50 e 100) e ao término de cada simulação foi coletado o tempo gasto.

Esta simulação foi realizada também para uma chave *crossbar* tradicional. O modelo desta chave possui algumas alterações em relação ao modelo da figura 3:

- Os *places* referentes aos nós foram retirados, com exceção dos *places* da diagonal, que permaneceram sem fichas.
- Foi acrescentado um controle para conflito em cada uma das quatro linhas.

O gráfico da figura 5 apresenta uma comparação relativa ao tempo gasto para terminar a simulação em cada uma das situações de entrada (10, 20, 50 e 100 fichas). O objetivo é mostrar quanto tempo cada uma das arquiteturas de chave *crossbar* levou para encaminhar todas as fichas para as quatro saídas. Segundo o resultado obtido, o desempenho da proposta de chave *crossbar* reconfigurável foi melhor em todos os cenários, apresentando um ganho médio de 1,4.

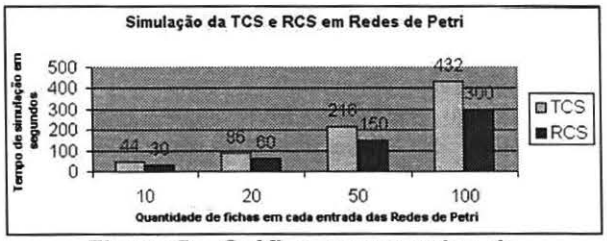

Figura 5 – Gráfico comparativo de desempenho

Em um contexto de redes de comunicação de dados um desempenho 1,4 vezes maior da RCS em relação à proposta de chave *crossbar* tradicional pode resultar em uma menor quantidade de pacotes descartados ou retransmitidos devido aos baixos tempos de espera em fila.

3.2. Síntese em FPGA

Para possibilitar o primeiro nível de reconfiguração e para verificar nossa proposta, codificamos a RCS-2 usando uma linguagem de descrição de hardware (HDL – *Hardware Description Language*) [18]. A

linguagem escolhida para tal codificação foi VHDL (*VHSIC – Very High Speed Integrated Circuit - Hardware Description Language*). Esta escolha foi motivada pela facilidade de se codificar circuitos usando tal linguagem e por se tratar de um padrão, possibilitando a portabilidade do código para diversas ferramentas e dispositivos para síntese.

Além da utilização de uma linguagem que permite a portabilidade de nossa implementação, foi necessária uma codificação parametrizada para atingirmos o primeiro nível de reconfiguração totalmente. Para isso, criamos um pacote que possui todas os parâmetros que podem ser variados do projeto da RCS-2. Desta maneira, uma alteração da configuração, como número de portas de entrada ou saída, é refletida no código pela modificação de uma variável deste pacote.

Para possibilitar o segundo nível de reconfiguração, o módulo decodificador, representado na Figura 1, foi implementado. O decodificador recebe 3 conjuntos de bits: tipo de reconfiguração (2 bits), endereçamento ($\log_2^{(\text{número\_de\_entradas x número\_de\_entradas})}$ bits) e bits de configuração (2 bits). Os tipos de reconfiguração permitidos são: nó, linha ou coluna. O primeiro configura uma conexão e os outros configuram uma linha ou coluna inteira da matriz de conexões. O número de bits de endereçamento é o número necessário para endereçar qualquer conexão da chave *crossbar*. O tempo para a reconfiguração é definido pelas características do dispositivo onde ele é implementado e pelo número de entradas e saídas da chave *crossbar*. Qualquer reconfiguração na RCS-2 gasta o mesmo tempo porque os nós da linha/coluna são configurados em paralelo.

O código implementado e, conseqüentemente, a arquitetura da chave *crossbar* foram verificados através de simulações comportamentais da ferramenta ModelSim XE II da empresa Model Technology. Simulamos alguns fechamentos de pontos da matriz de conexão, configurações e o envio de alguns pacotes. Os resultados encontrados foram equivalentes aos de simulações manuais.

Para a análise dos recursos de um dispositivo necessários para a implementação da RCS-2, consultamos os relatórios gerados pela ferramenta Xilinx ISE 6 de um projeto com 8 portas de entrada (e de saída). Utilizamos como dispositivo alvo o FPGA Spartan 2 xc2s200-6fg256 da Xilinx. Os valores encontrados estão na tabela 2.

Para a análise temporal de uma das possíveis implementações (síntese) da RCS-2, realizamos simulações pós-síntese também através da ferramenta ModelSim XE II usando o mesmo dispositivo e configuração usados anteriormente. Os tempos encontrados podem ser observados na tabela 3.

Tabela 2 – Recursos do no dispositivo FPGA Spartan 2 xc2s200-6fg256 da Xilinx utilizados na implementação da RCS-2 com 8 portas de entrada

Recursos utilizados	Número	Porcentagem
Slices	366	15%
Flip Flops	160	3%
LUTs (*Look Up Tables*)	681	14%

Considerando uma chave *crossbar* tradicional (TCS) implementada usando a mesma tecnologia, podemos considerar a mesma latência e freqüência da implementação da RCS-2 da tabela 3. Conseqüentemente, o tempo necessário para a transmissão de um pacote é o mesmo também. Na tabela 4 são exibidos os tempos necessários para alguns padrões de comunicação, tanto para a TCS quanto para a RCS-2, ambas com 8 entradas.

Como a RCS-2 é capaz de fazer *multicast* e *broadcast* usando apenas a topologia implementada na chave o tempo necessário para se transmitir pacotes para uma ou mais máquinas é o mesmo. Como se pode fechar qualquer nó da RCS-2, qualquer grupo de *multicast* pode ser montado e, portanto, todos os grupos possuem o mesmo tempo de execução. No entanto, na TCS, cada pacote deve ser enviado individualmente, portanto, quanto mais nós são envolvidos na comunicação, maior o tempo necessário para realizá-la.

Tabela 3 – Tempos da RCS-2 implementada no dispositivo FPGA Spartan 2 xc2s200-6fg256 da Xilinx

Inicialização da chave *crosssbar*	23,137 ns
Configuração de um nó da matriz de conexões	8,780 ns
Latência entre portas da chave *crosssbar*	12,895 ns
Freqüência de bits suportada pela *crossbar*	1GHz
Overhead da utilização da *crossbar* para transmissão de 1500 Bytes	6012,9 ns

Utilizando os tempos desta implementação, simulamos uma aplicação real para verificar as vantagens da utilização da RCS-2. Supondo uma rede formada por 8 nós e onde o padrão de comunicação é o de uma topologia anel unidirecional, figura 6. Nela, os nós são logicamente conectados da maneira

representada pelas setas, mas são fisicamente conectados através da chave *crossbar*. Estas conexões são representadas pelas linhas tracejadas. Internamente, esta topologia é implementada na RCS-2 através do fechamento das conexões representadas pelos círculos pretos da figura 7. Na TCS a topologia é temporal, as conexões são realizadas ao longo do tempo. Como consideramos que as duas chaves são implementadas usando-se a mesma tecnologia, qualquer comunicação nestas duas chaves gasta o mesmo tempo.

Tabela 4 - Tempos da TCS e RCS-2 para executar alguns padrões de comunicação transferindo um pacote (1.500 Bytes)

Padrão de comunicação	TCS	RCS-2
Ponto a ponto	6012,9 ns	6012,9 ns
Multicast – 2 nós	12025,8 ns	6012,9 ns
Multicast – 3 nós	18038,7 ns	6012,9 ns
...	...	...
Multicast – 7 nós	42090,3 ns	6012,9 ns
Broadcast	48103,2 ns	6012,9 ns

Considerando a rede descrita anteriormente, em um dado momento, o nó 1 precisa enviar pacotes para todos os demais nós da rede. A topologia anel unidirecional não é indicada para esta comunicação, já que uma mensagem para o nó 8, por exemplo, precisaria passar por todos os demais. Para minimizar este problema, poderia ser implementada a topologia estrela, representada na figura 7 pelas linhas internas ao anel. Para implementar esta topologia na chave *crossbar* seria necessário fechar as conexões representadas pelos círculos brancos. No entanto, como estas conexões estão em mesmas linhas e/ou colunas de outras conexões, uma TCS não conseguiria implementar a topologia estrela espacialmente.

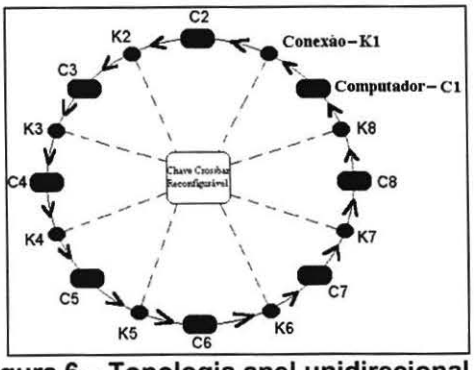

Figura 6 – Topologia anel unidirecional

Para implementar a topologia estrela, a TCS precisa fechar as conexões uma a uma ao longo do tempo. Como já estava implementada a topologia anel unidirecional, seria necessário abrir as demais conexões. Então, seriam necessárias diversas comunicações ponto a ponto. Portanto, em nossa situação exemplo, seriam necessários 7 x 6012,9ns para enviar o pacote para todos os demais 7 nós. Para o nó 1 mandar *n* pacotes para os demais, seriam necessários *n* x 7 x 6012,9ns. Pode-se observar que, a medida que o número de pacotes e/ou o número de nós da rede aumenta, o tempo necessário para a comunicação cresce proporcionalmente.

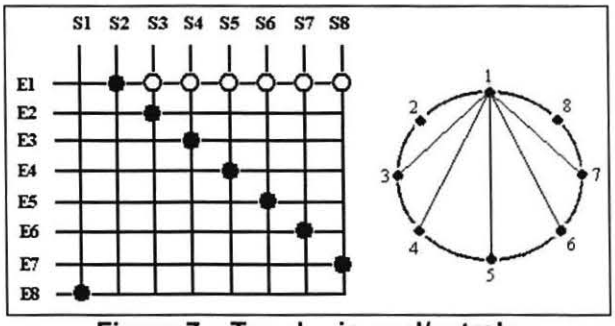

Figura 7 – Topologia anel/estrela

A mesma topologia pode ser implementada na RCS-2 de maneira diferente. Como a RCS-2 permite que mais de uma conexão seja fechada em uma mesma linha ou coluna, é possível implementar a topologia fechando as conexões representadas pelos círculos brancos (figura 7). Com isto, apenas 6012,9ns são gastos para que o nó 1 envie um pacote para todos os demais nós. Deste modo, para o nó 1 mandar *n* pacotes para os demais, seriam necessários *n* x 6012,9ns. No entanto, para fechar todos estes nós é necessário reconfigurar a RCS-2. Esta reconfiguração é uma reconfiguração de segundo nível, não sendo necessário reconfigurar o dispositivo. Esta reconfiguração, neste problema exemplo, gastaria 8,780ns + 8,780ns. O primeiro tempo é para reconfigurar a linha da entrada 1 da RCS-2, fechando todos os nós, e o segundo para abrir o nó que conecta a entrada 1 à saída 1. Portanto, mesmo com o *overhead* de reconfiguração, o tempo de comunicação da RCS-2 seria menor que o da TCS.

4. Trabalhos Correlatos

Apresentamos neste tópico alguns trabalhos envolvendo chaves *crossbar* e que abordam a aplicação de conceitos relativos à computação reconfigurável.

No entanto, podemos ressaltar que nenhuma das pesquisas ou artigos encontrados abordam questões

relativas aos dois níveis de reconfiguração, conforme nossa proposta apresenta.

O artigo *A High I/O Reconfigurable Crossbar Switch* [2], apresenta resultados no uso do FPGA para implementação de uma chave *crossbar*. Neste artigo, Fewer aborda questões envolvendo a reconfiguração das células de forma parcial em tempo de execução para aumentar a flexibilidade de interconexão. Esta pesquisa aborda reconfiguração da implementação da chave *crossbar* sobre o FPGA, não havendo, portanto, uma proposta de nova arquitetura, conforme apresentamos na proposta da RCS-2.

Em *Fast Reconfigurable Crossbar Switching in FPGAs* [10] o objetivo é propor a integração entre um FPGA e uma chave *crossbar* para reduzir custo de parte de um circuito dedicado, alterando a implementação da chave *crossbar*. Esta pesquisa também não apresenta uma proposta de arquitetura de chave *crossbar*, permanecendo o conceito de reconfiguração aplicável somente na implementação da mesma sobre o FPGA.

A dissertação de mestrado *VHDL Implementation of a High-Speed Symmetric Crossbar Switch* [15] aborda apenas a implementação da chave crossbar sobre um FPGA. Foram implementados blocos construtivos tais como: portas de entrada, escalonador (definição de rotas) e a unidade de chaveamento (matriz de conexões). Não há uma proposta de arquitetura reconfigurável, apenas a implementação sobre o FPGA.

A única pesquisa que encontramos com uma proposta de arquitetura é a *FLEXBAR: A crossbar switching fabric with improved performance and utilization* [11]. No entanto, a proposta contribui com novas arquiteturas para as camadas de hardware das portas de entrada e para o escalonador. O objetivo é diminuir ao máximo as mudanças na arquitetura de uma chave *crossbar* tradicional. Neste artigo não é apresentado o conceito de reconfiguração e não é mencionada implementação em FPGA, o que nos faz entender que a proposta de arquitetura de chave *crossbar* não é reconfigurável.

Em relação à nossa proposta (RCS-2), podemos ressaltar que existem dois níveis de reconfiguração, sendo que o segundo nível (programação da chave *crossbar* – reconfiguração de topologias) é independente de plataforma.

5. Conclusões

O projeto da chave *crossbar* reconfigurável tem seguido método baseado na evolução de sua arquitetura, que resultou na proposta da RCS-2.

Os resultados apresentados neste artigo têm como objetivo verificar a proposta de arquitetura reconfigurável de dois níveis através da modelagem e simulação em Redes de Petri e síntese em FPGA.

Foi verificado através da simulação em Redes de Petri que a RCS-2 conseguiu um ganho considerável de desempenho em relação a TCS. Como conseqüência, podemos diminuir o gargalo de uma rede de comunicações de dados, reduzindo o atraso de rede, os possíveis descartes de pacotes, aumentando a vazão de dados. A relação de 1,4 vezes de ganho de desempenho favorável à RCS indica uma maior capacidade de escoamento de dados em um tempo menor de trabalho.

O primeiro nível de reconfiguração da RCS-2 pôde ser alcançado através da codificação da arquitetura usando uma linguagem de descrição de hardware, permitindo que ela seja implementada em diversos dispositivos. Já o segundo nível de reconfiguração pôde ser atingido com as modificações na matriz de conexões (conforme arquitetura da figura 1). Estas modificações geram um *overhead*, no entanto, através de nossos experimentos constatamos que ele é menor que o ganho conseguido pela implementação das topologias na RCS-2. Portanto, podemos concluir que a RCS-2 possui melhor desempenho que uma chave *crossbar* tradicional.

Uma unidade de chaveamento de dados baseada em chave *crossbar* possui outros blocos necessários para o funcionamento e armazenamento de pacotes de dados. Sendo assim, *buffers* reconfiguráveis para alocação de dados temporários serão acrescentados no projeto da chave *crossbar*. Portanto, podemos destacar como trabalhos futuros:

- Projeto de *buffers* reconfiguráveis para as portas de entrada;
- Verificação através de um modelo analítico e teoria de filas do impacto de desempenho;
- Descrição em VHDL e síntese em FPGA dos *buffers* e da chave *crossbar* reconfigurável.

6. Agradecimentos

Aos colegas do Grupo de Sistemas Computacionais e Digitais, ao Programa de Pós-graduação em Engenharia Elétrica, ao Instituto de Informática, à Pró-reitoria de Pesquisa e Pós-graduação e a PUC Minas

pelo incentivo e suporte, que possibilitou nossa participação no WSCAD 2005.

7. Referências

[1] Buyya, R., High Performance Cluster Computing, Volume 1, Prentice Hall, 1999

[2] C. Fewer, et al., "A High I/O Reconfigurable Crossbar Switch", *11th Annual IEEE Symposium on Field-Programmable Custom Computing Machines*, April 09-13, 2003, Napa, California, pp.3-10, 2003

[3] C. Ulmer, C. Wood, S. Yalamanchili, "Active SANs: Hardware Support for Integrating Computation and Communication", *Workshop on Novel Uses of System Area Networks at HPCA (SAN 2002)*, February 2002

[4] Comer, D. E., "Network Systems Design Using Network Processors", Prentice Hall, 2003

[5] De Rose, C. A. F. e P. O. A. Navaux, "Arquiteturas Paralelas", Série Livros Didáticos, Editora Sagra Luzzatto, 2003

[6] Freitas, H. C., "Proposta e Desenvolvimento de Processador de Rede com Chave *Crossbar* Reconfigurável", dissertação de mestrado apresentada em 04 de novembro de 2003

[7] Freitas, H. C., C. A. P. S. Martins, "Processadores de Rede: Conceitos, Arquiteturas e Aplicações" (Capítulo de Livro), Minicurso apresentado na *III Escola Regional de Informática RJ/ES*, Vitória, 07 de outubro, 2003, pp.127-166

[8] Girault, C., R. Valk, "Petri Nets for Systems Engineering – A Guide to Modeling, Verification and Applications", Springer-Verlag, 2003

[9] H. C. Freitas, C. A. P. S. Martins, "Chave *Crossbar* Reconfigurável para Implementação Dinâmica de Topologias em Redes de Interconexão de Dados", *Workshop em Sistemas Computacionais de Alto Desempenho*, Foz do Iguaçu, 27 a 29 de outubro, 2004, pp.74-81

[10] H. Eggers, P. Lysaght, H. Dick, G. McGregor, "Fast Reconfigurable Crossbar Switching in FPGAs", *6th International Workshop on Field-Programmable Logic and Applications*, Springer-Verlag LNCS 1142, 1996, pp.297-306

[11] J. Chang, S. Ravi, and A. Raghunathan, "FLEXBAR: A crossbar switching fabric with improved performance and utilization", *IEEE Custom Integrated Circuits Conference (CICC)*, May 2002

[12] J. Turner and N. Yamanaka, "Architectural choices in large scale atm switches," *IEICE Transactions*, vol. E81-B, no. 2, pp. 120–137, 1998.

[13] K. Compton, S. Hauck, "Reconfigurable Computing: A Survey of Systems and Software", *ACM Computing Surveys*, Vol. 34, No. 2, June 2002, pp. 171-210

[14] M. Glesner, A. Kirschbaum, "State-of-the-Art in Rapid Prototyping", *XI Brazilian Symposium on Integrated Circuit Design*, SBCCI'98, Búzios, Rio de Janeiro, 1998, pp.60-65

[15] M. Keyvani, "VHDL Implementation of a High-Speed Symmetric Crossbar Switch", Dissertação de Mestrado, School of Engineering Science Communication Networks Laboratory, Simon Fraser University, August, 2001

[16] Marsan, M. A., G. Balbo, G. Conte, "Performance Models of Multiprocessor Systems", Computer Systems Series, MIT Press, second printing, 1988

[17] Martins, C. A. P. S., E. D. M. Ordonez, J. B. T. Corrêa e M. B. Carvalho, "Computação Reconfigurável: conceitos, tendências e aplicações", Jornada de Informática 2003, Congresso da Sociedade Brasileira de Computação, Capítulo 8, 2003

[18] Ordonez, E. D. M., F. D. Pereira, C. G. Penteado, R. A. Pericini, "Projeto, Desempenho e Aplicações de Sistemas Digitais em Circuitos Programáveis (FPGAs)", Bless Gráfica e Editora Ltda, 2003

[19] Patterson, D. A., J. L. Hennessy, Organização de Computadores a Interface Hardware/Software, Campus, terceira edição, 2005

[20] R. Drath, "Visual Object Net ++", Technical University of Ilmenau, Alemanha, URL: http://www.systemtechnik.tu-ilmenau.de/~drath/index_e.htm

[21] S. Keshav, R. Sharma, "Issues and Trends in Router Design", *IEEE Communications Magazine*, Vol. 36, No. 5, May 1998, pp.144-151

[22] Tanenbaum, A. S., "Redes de Computadores", Editora Campus, quarta edição, 2003

[23] T. Ungerer, B. Robic, J. Silc, "Multithreaded Processors", *British Computer Society*, 2002

[24] T. Wolf and J. Turner, "Design Issues for High Performance Active Routers", *International Zurich Seminar on Broadband Communications*, Zurich, Switzerland, February 2000, pp. 199-205

[25] W. D. Mensch. Jr. and D. A. Silage, "System-on-chip Design Methodology in Engineering Education", *International Conference on Engineering Education, ICEE2000 (IEEE/CS)*, Taipei, Taiwan, August 2000, pp. 224-228

[26] Xilinx Development System, "Synthesis and Simulation Design Guide", 2002