HOLOS
ISSN 1807 - 1600

EXPLORAÇÃO DE ESPAÇO DE PROJETO DO ROTEAMENTO NA ARQUITETURA IPNOSYS

R. V. COSTA FILHO, S. R. F. ARAÚJO, D. F. L. NUNES, J. L. SILVA, I. A. C. ALVES e W. K. S. DANTAS
Programa de Pós-Graduação UFERSA/UERN
valtercostaf@yahoo.com.br

Artigo submetido em janeiro/2013 e aceito em abril/2014

DOI: 10.15628/holos.2014.1909

RESUMO

Neste artigo, propõe-se algoritmos de roteamento diferenciados para avaliar o impacto destes sobre o desempenho geral da arquitetura IPNoSys. É avaliada, também, a quantidade de retransmissões realizadas por cada algoritmo a fim de apontar soluções para melhor distribuição de carga na rede e com melhor impacto no desempenho das aplicações nessa arquitetura

PALAVRAS-CHAVE: IPNoSys, Roteamento, Multiprocessador, Rede-em-chip, Arquitetura Dataflow.

THE ROUTING PACKAGE IMPACT ON IPNOSYS ARCHITECTURE PERFORMANCE

ABSTRACT

This paper proposes different routing algorithms to assess the impact of different routes taken by the regular packages on the overall performance of the architecture IPNoSys. It is evaluated the amount of retransmissions performed by each algorithm in order to identify possible solutions to load balancing and better performance when considering applications running over this system.

KEYWORDS: IPNoSys, Routing, Multiprocessor, Network-on-chip, Dataflow Architecture.

HOLOS
ISSN 1807 - 1600

1 INTRODUÇÃO

As redes-em-chip surgiram como uma alternativa viável à conectividade de um número cada vez maior de componentes intra-chip. Com o futuro do desempenho dos computadores atuais pautado no paralelismo da computação, os multiprocessadores/*multicores* viram nesta solução um grande potencial de ampliação da quantidade de elementos processantes. Porém, apesar das vantagens de comunicar paralelamente, de forma não bloqueante, com tolerância a falhas, tornando o sistema escalável e a possibilidade de reuso dos componentes, as redes-em-chip incrementam consideráveis custos no projeto e latência na comunicação (ZEFERINO, 2003).

Apesar de LEE, (2008) ressaltar que a ideia chave em projetos envolvendo NoCs é desacoplar a computação da comunicação envolvida entre os pontos de um sistema, (ARAÚJO, 2012) confirma essa ideia com a proposta de uma arquitetura não-convencional nesta perspectiva. A partir da união das características das redes intra-chip aos sistemas computacionais em fila, foi sugerido o sistema IPNoSys (*Integrated Processing NoC System*) (ARAÚJO, 2012).

Pela analogia entre as filas das *queue machines* e os pacotes das redes-em-chip, geralmente transmitidos sequencialmente, a IPNoSys transmite instruções e dados através da rede de interconexão. Os roteadores, devidamente adaptados com unidade lógica aritmética e estruturas de controle, executam as instruções na medida em que elas trafegam na rede. Essa abordagem, relativamente nova, sugeriu oportunidades no tocante à execução orientada a fluxo de dados, podendo valorizar a paralelização do processamento dos dados e, consequentemente, alcançar melhor desempenho.

Em uma arquitetura, em sua essência baseada em redes-em-chip, é natural que o desempenho esteja atrelado a alguns aspectos como roteamento, método de chaveamento, memorização, arbitragem, controle de fluxo, entre outras características das redes-em-chip. Contudo, o quanto a eficiência da arquitetura IPNoSys estaria ligada à rota dos pacotes que a percorre? Para responder essa pergunta e avaliar outros aspectos do sistema, propõe-se, neste trabalho, novos algoritmos de roteamento para IPNoSys.

Este artigo está organizado da seguinte forma: a seção Arquitetura IPNoSyS faz uma breve revisão da arquitetura IPNoSys. A seção Algoritmos de Roteamento elenca as propostas de algoritmos adicionais para avaliar possíveis melhorias no desempenho da arquitetura. A seção Resultados apresenta detalhes dos experimentos e realiza a comparação dos resultados obtidos. A seção Considerações Finais aborda uma rápida reflexão acerca de alguns pontos de melhoria objetivando o desempenho global do sistema e a seção Trabalhos Futuros discorre sobre tendências observadas na pesquisa.

2 ARQUITETURA IPNOSYS

A arquitetura IPNoSys é estruturada com base em uma rede-em-chip em topologia grelha 2-D. Os roteadores foram substituídos por RPUs (*Routing and Processing Unit*) capazes de executar instruções e rotear os pacotes. Apenas as MAUs (*Memory Access Unit*) e IOMAU (*In/Out Memory Access Unit*) estão conectadas aos cantos da rede. Estas são unidades que gerenciam operações sobre a memória, com a diferença de que a IOMAU contempla, também, a gestão de dispositivos

HOLOS
ISSN 1807 - 1600

de entrada e saída. A memória é compartilhada, sendo o espaço de endereçamento dividido igualmente entre as MAUs e IOMAU. A IOMAU dispõe do IONode, que liga o dispositivo de entrada/saída à memória e transfere automaticamente os dados. O modelo da arquitetura é ilustrado na Figura 1.

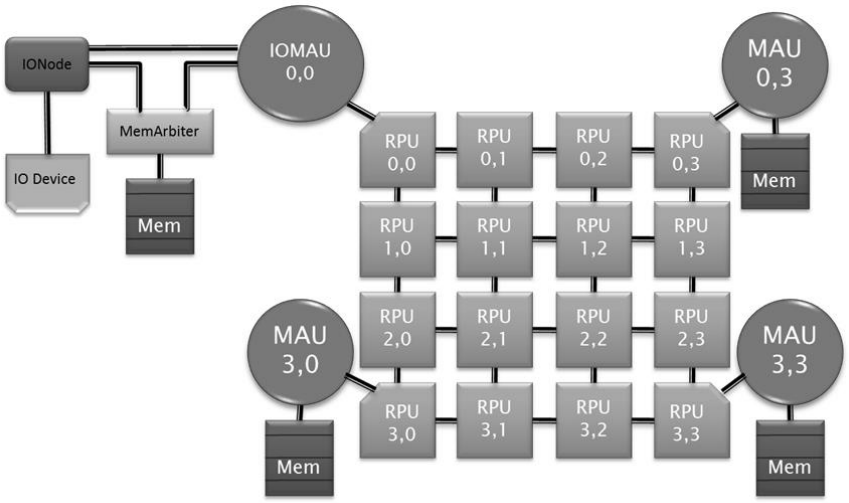

Figura 1 – Arquitetura da IPNoSys.

Diferenciando-se de uma rede-em-chip comum, onde a carga útil dos pacotes não necessariamente interessa aos roteadores, na IPNoSys, os pacotes carregam instruções e operandos a serem processados durante o percurso na rede. O formato dos pacotes que trafegam na IPNoSys está ilustrado na Figura 2. Estes são formados por *flits* (*FLow control unIT*) contendo uma palavra de 32 bits da carga útil e 4 bits de controle. A técnica de chaveamento de pacotes utilizada é *wormhole*. O primeiro *flit* de cabeçalho é responsável por armazenar informações de destino, origem, quantidade de instruções no pacote, re-roteamento e tipo do pacote. Ele é encaminhado primeiro na rede e os demais *flits* seguem pela mesma rota.

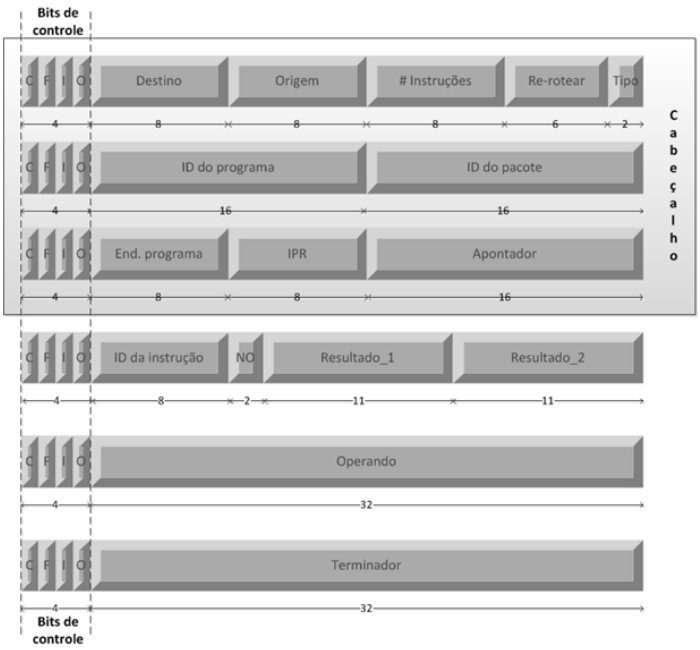

Figura 2 - Formato dos pacotes que trafegam no sistema IPNoSys.

HOLOS
ISSN 1807 - 1600

As aplicações executadas na IPNoSys são programadas em formato de pacotes por meio da linguagem PDL (*Package Description Language*) e armazenadas nas memórias localizadas nos quatro cantos da IPNoSys. O modelo de execução proposto em (ARAÚJO, 2012) estabelece que os pacotes contendo instruções e dados são processados ao passo em que trafegam pelo sistema. Ao ser injetado por uma das MAUs ou IOMAU, o conteúdo do pacote é avaliado pela primeira RPU. A primeira instrução é então executada. O resultado é inserido em posições específicas do pacote enquanto os demais *flits* são transmitidos para a próxima RPU. Esta segunda RPU executa a próxima instrução na sequência e retransmite o resultado da mesma maneira, bem como o resto do pacote, tal como realizado pela primeira RPU. Estas operações se repetem a cada RPU no caminho do pacote até que todo o processamento seja concluído ou seja executada uma instrução de chamada de sistema, por meio do qual o pacote fica bloqueado na RPU corrente aguardando o recurso solicitado.

Os pacotes que trafegam na IPNoSys chamados regulares são aqueles que contém instruções e dados a serem processados. Diferentemente do que é observado em uma NoC comum, estes pacotes não possuem destino específico no sistema e devem ser roteados de maneira a balancear a execução de instruções entre as RPUs. Apesar do processamento realizado pelas RPUs provocar, em média, uma redução no tamanho desses pacotes (ARAÚJO, 2009), boa parte são suficientemente grandes de maneira que precisam ser encaminhados em trajetórias que permitam, além da (i) distribuição de carga computacional, o (ii) encaixe do maior número possível de pacotes no sistema. Como solução ao problema, ARAÚJO, (2008) propôs o algoritmo de roteamento *Spiral Complement*.

3 ALGORITMOS DE ROTEAMENTO

Quando considera-se o comportamento de um sistema como a IPNoSys, as rotas percorridas pelos pacotes influenciam diretamente no desempenho da arquitetura. É esperado que diferentes métodos de roteamento favoreçam características específicas inerentes ao sistema e apontem caminhos para o desenvolvimento de técnicas mais eficazes em distribuir a execução e extrair resultado na IPNoSys. Para evidenciar como a rota dos pacotes influencia o sistema, propomos três novos algoritmos. A seguir, descreve-se o algoritmo originalmente utilizado na IPNoSys, o *Spiral Complement*, além de dois novos algoritmos determinísticos e um aleatório.

3.1 Spiral Complement

O algoritmo Spiral Complement é executado em cada roteador de forma distribuída. Ele tem esse nome devido à forma espiralada do caminho desenhado pelo pacote ao percorrer o sistema. Para rotear os pacotes contendo instruções e dados, a IPNoSys utiliza-se das informações de origem, destino e re-rotear, contidas no primeiro flit. O método consiste em inicialmente definir um destino temporário e, a partir disso, executar o algoritmo de roteamento XY para movimentar o pacote da origem até este destino. O primeiro destino temporário é encontrado executando o complemento de um do valor binário das coordenadas do endereço atual. A técnica XY é caracterizada pela movimentação do pacote primeiramente na direção do eixo X e, em seguida, na direção Y, atingindo o destino escolhido. Neste ponto, caso o pacote não tenha sido completamente computado, a RPU recalcula um novo destino para o pacote. Este cálculo utiliza-se do *bit* re-rotear e é executado sobre o endereço da RPU atual. Caso a RPU esteja localizada em

HOLOS
ISSN 1807 - 1600

um dos cantos da IPNoSys, o novo destino é encontrado a partir do complemento de um do endereço atual subtraindo uma unidade da coordenada X ou Y encontrada. Caso a RPU não esteja em um dos cantos, o novo destino é encontrado apenas com o complemento de um do endereço cartesiano atual. A Figura 3 mostra dois passos na sequência do roteamento *Spiral Complement* para um pacote em execução.

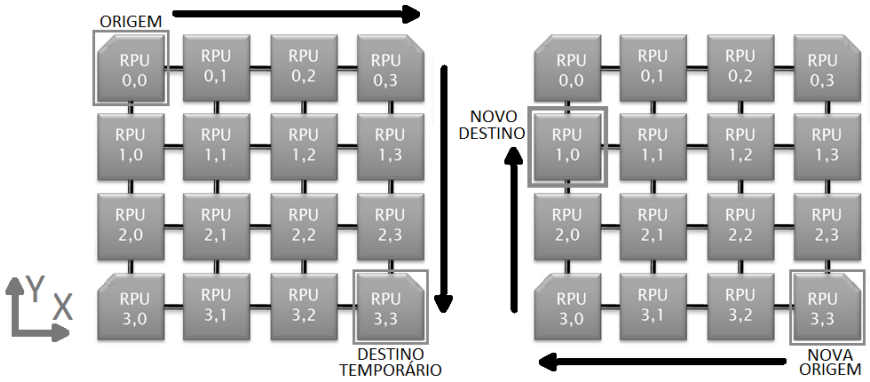

Figura 3 - Passos intermediários do roteamento *Spiral Complement*.

Dependendo do comprimento do pacote, do tráfego ou das dimensões da rede, poderá ocorrer um *deadlock*, isto é, o pacote pode ficar impedido de prosseguir até o destino temporário devido ao preenchimento total do *buffer* da próxima RPU. Para contornar este problema, foi adotada a execução localizada. Esta técnica consiste em executar o pacote na mesma RPU, até que o impedimento seja resolvido ou o pacote seja totalmente computado.

Na Figura 4-A tem-se o caminho percorrido pelo pacote ao ser injetado pela respectiva MAU e, na Figura 4-B, a estimativa de distribuição das execuções considerando um pacote ideal[1]. Com base nessa distribuição, espera-se uma média de 381 instruções executadas por RPU e um desvio padrão médio de aproximadamente 61 instruções, isto é, 15,89% da média de execuções.

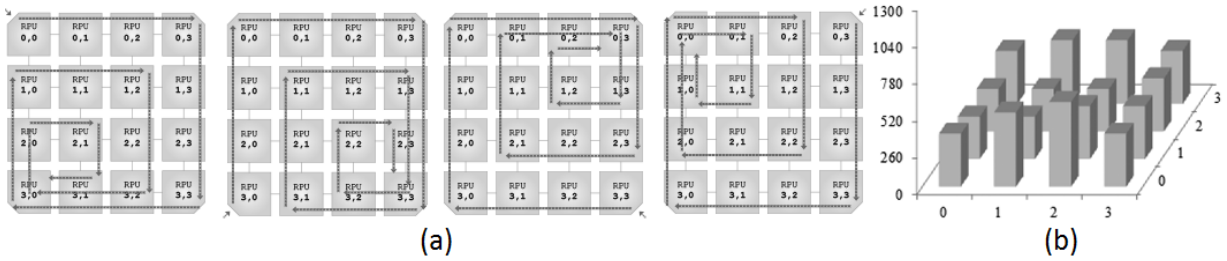

Figura 4 – (a) Rotas produzidas pelo algoritmo *Spiral Complement* e o (b) gráfico de distribuição de execuções de um pacote ideal.

3.2 Roteamentos em Ziguezague

Utilizando as mesmas informações de destino, origem e re-roteamento, que foram necessárias para o algoritmo *Spiral Complement*, pode-se chegar a um conjunto de diferentes rotas percorridas pelos pacotes. Adicionalmente à proposta de (Cruz, 2013) (Cruz et al, 2013), foram propostas duas novas diferentes rotas em Ziguezague. A ideia é provocar uma melhora na distribuição das execuções e a diminuição das condições de colisão entre pacotes. Responsáveis

[1] Exemplo hipotético de pacote, contendo 6100 instruções, onde cada RPU executa uma única instrução e encaminha o pacote para a RPU seguinte, não existindo colisões nem chamadas de sistema.

HOLOS
ISSN 1807 - 1600

pelas execuções localizadas e que resulta em um mau balanceamento de carga computacional no sistema. A seguir serão apresentados os algoritmos de roteamentos propostos neste artigo.

3.2.1 Ziguezague 1

Neste método, o pacote entra na rede por uma das MAUs e é transmitido até a RPU mais distante do sistema. Em seguida faz-se o caminho de volta ao nó inicial. Neste ponto, o processo se repete, fazendo com que o pacote nunca pare. Em determinados pontos, os pacotes que estão trafegando para frente, podem se chocar com os que estão no sentido contrário. Na IPNoSys, este problema só afeta o desempenho da rede quando os caminhos tem a mesma direção e sentido, porque dois pacotes não podem ocupar a mesma porta do roteador. Para contornar isso, a IPNoSys usa o mecanismo de canal virtual. No canal virtual, os pacotes podem dividir a porta do roteador, porém a utilização de canais virtuais pode prejudicar o desempenho da rede. Para os pacotes oriundos das MAUs posicionadas na parte superior da rede, o pacote deve caminhar descrevendo um "L" no sentido XY. Para os pacotes vindos das MAUs inferiores, o sentido adotado é o YX. Essa ordem serve para distribuir os pacotes de forma mais homogênea na rede e diminuir o número de colisões.

No tocante a implementação, o algoritmo simplificou o cálculo da rota, pois fixa quatro rotas específicas baseando-se, para isso, na RPU de entrada do pacote na IPNoSys. Reduzindo bastante a complexidade do sistema, pois não se faz necessário calcular qual o próximo destino do pacote a cada retransmissão. A Figura 5-A nos mostra os 4 caminhos que um pacote pode percorrer quando é utilizado o algoritmo Ziguezague 1. Enquanto a Figura 5-B, a estimativa de distribuição das execuções considerando um pacote ideal.

Para o Ziguezague 1, foram desenhadas, na Figura 5, as rotas dos pacotes e estimadas as distribuições das execuções imaginando um pacote ideal, conforme feito anteriormente. Foi observado que, apesar de uma média de aproximadamente 381 instruções por RPU, o desvio padrão médio é de 53 instruções (14% da média) para o referido método. Esta medida informa o quanto se espera que a execução seja distribuída na rede, desconsiderando as colisões entre pacotes. É esperado que esta técnica concentre mais execuções no centro do sistema, facilitando a entrada de mais pacotes pelas RPUs diretamente ligadas às MAUs e IOMAU.

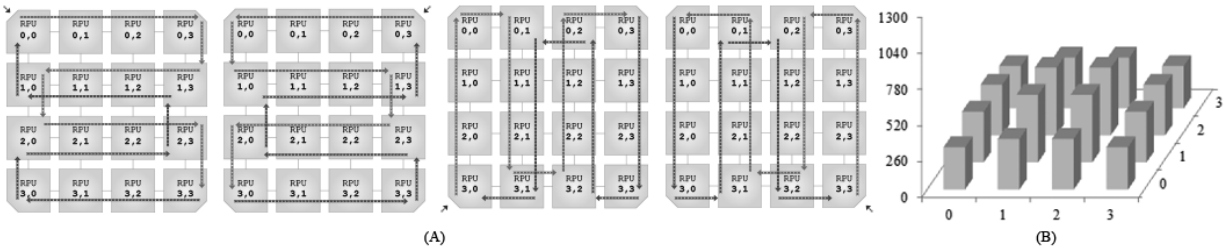

Figura 5 – (a) Rotas produzidas pelo algoritmo Ziguezague 1 e (b) gráfico de distribuição de execuções de um pacote ideal.

3.2.2 Ziguezague 2

Neste método de roteamento, o pacote entra por uma das MAUS e é transmitido pelas RPUs em um roteamento que segue um padrão de dois tempos, onde no primeiro tempo é iterada apenas uma das coordenadas, fazendo com que o pacote seja direcionado para a RPU da extremidade direita oposta. No segundo tempo, é realizado um roteamento XY similar ao *spiral*

HOLOS
ISSN 1807 - 1600

complement. Essa modificação, visa à flexibilização do tamanho da rede (problema crucial no roteamento determinístico). Além disso, o algoritmo visa uma melhor distribuição de pacotes na rede.

O algoritmo usa 3 bits para fazer o roteamento, (i) o primeiro bit define se a coordenada cresce ou decresce, o segundo bit define se a direção do roteamento do primeiro ciclo é vertical (RPUs 0,0 e 3,3) ou horizontal (RPUs 0,3 e 3,0) e o terceiro bit define se o roteamento está no primeiro ou segundo ciclo.

A rota percorrida pelo pacote ao utilizar o método descrito está ilustrado na Figura 6-A. A execução de um pacote ideal, ilustrada na Figura 6-B, mostra que o desvio padrão esperado é nulo para o Ziguezague 2. Isto é, o referido algoritmo tenderia a balancear a carga computacional por todas as RPUs do sistema.

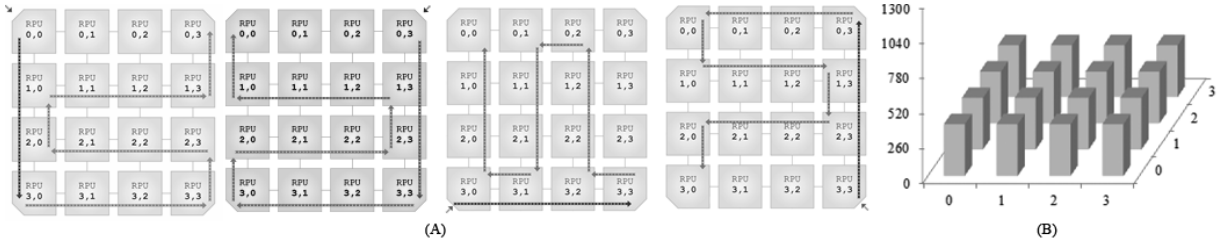

Figura 6 – (a) Rotas produzidas pelo algoritmo Ziguezague 2 e (b) gráfico de distribuição de execuções de um pacote ideal.

3.3 Roteamento Aleatório

Na tentativa de criar rotas que encaixassem a maior quantidade de pacotes na IPNoSys, surgiu o algoritmo aleatório. Neste algoritmo a origem e o destino do pacote não são úteis, o roteamento agora economiza esses bits. O algoritmo, em contrapartida, é orientado a encaixar o pacote na rede na tentativa de preterir condições de colisão.

Após a execução da instrução, o pacote é roteado para a RPU mais livre, isto é, para aquela que contiver o maior *buffer* disponível para alocação do pacote. Em caso de empate entre portas viáveis o algoritmo sortearia a RPU vencedora. Assim, o pacote seguiria percorrendo a rede pelos caminhos onde as RPUs, em tese, estivessem menos ocupadas. Para esse método, as rotas e a distribuição de carga não foram plotadas, já que ambas variam aleatoriamente com a execução, não ocorrendo uma rota preferencial como observado nos algoritmos de roteamento anteriores.

4 RESULTADOS

Para avaliar o desempenho da arquitetura frente aos novos algoritmos de roteamento, considera-se a aplicação DCT-2D (*Two Dimensional Discrete Cosine Transform*). Esta aplicação é uma operação útil em uma das etapas do processo de compressão de imagens JPEG e foi executada sobre uma imagem sintética de 16x16 *pixels*. A IPNoSys permite que o cálculo completo seja feito simultaneamente por quatro fluxos de execução. A Figura 7 esquematiza a imagem 16x16 *pixels* dividida igualmente pelas MAUs e IOMAU em quatro blocos de 8x8 *pixels*. Cada fluxo de execução da aplicação DCT-2D que parte das MAUs e IOMAU são independentes e processam a imagem separadamente. A aplicação foi executada completamente a partir da memória, não utilizando operações de entrada e saída.

HOLOS
ISSN 1807 - 1600

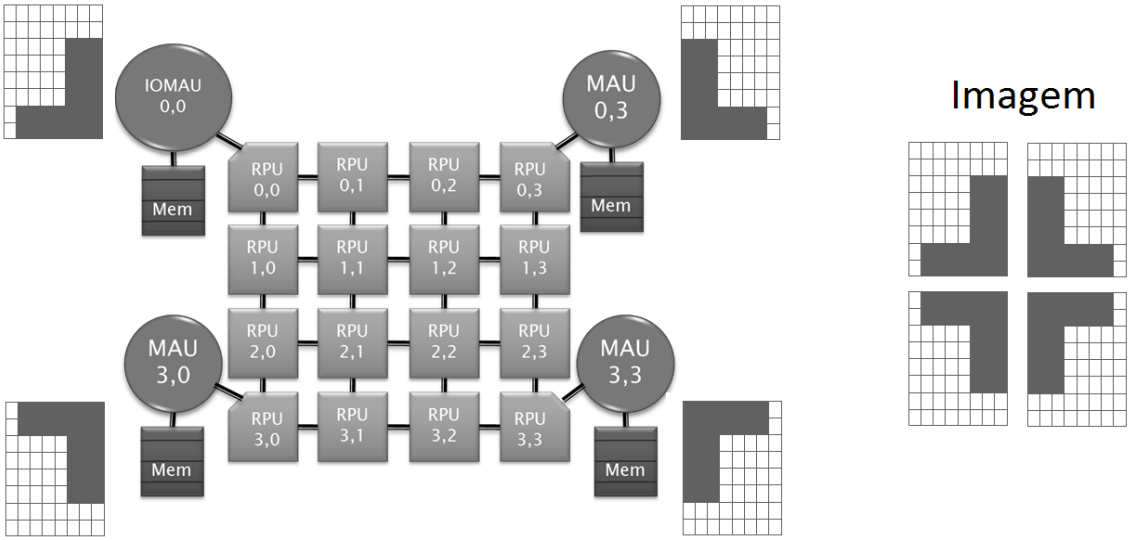

Figura 7 – Esquema exemplificando os quatro fluxos de execução da aplicação DCT-2D.

Os dados foram obtidos através de diversas repetições da aplicação DCT-2D para cada método de roteamento separadamente. Apenas o sistema que usou o algoritmo aleatório apresentou resultados distintos entre si. Este resultado poderia ser confundido pela natureza do método randômico, porém, em uma aplicação real do sistema IPNoSys, as variações de resultados seriam experimentadas por todas as técnicas propostas. Tendo em vista que o fluxo de execução nessas aplicações é assíncrono, disparando eventos aleatoriamente no tempo e produzindo diferentes atrasos na execução, reflexo das imprevisíveis colisões entre pacotes e execuções localizadas.

Todos os resultados foram expressos em porcentagem dos valores obtidos para o algoritmo *Spiral Complement*. Na Figura 8-A é possível constatar que os tempos decrescem proporcionalmente ao passo que o uso das RPUs se eleva, conforme Figura 8-B. Isto é, quanto maior o tempo de uso das unidades operativas, maior é o desempenho alcançado. Porém, apesar de melhorar o desempenho, as RPUs gastam apenas cerca de 2,5% do tempo realizando computação útil, o resto do tempo é gasto em roteamentos, transmissões e aguardando pacotes a serem computados.

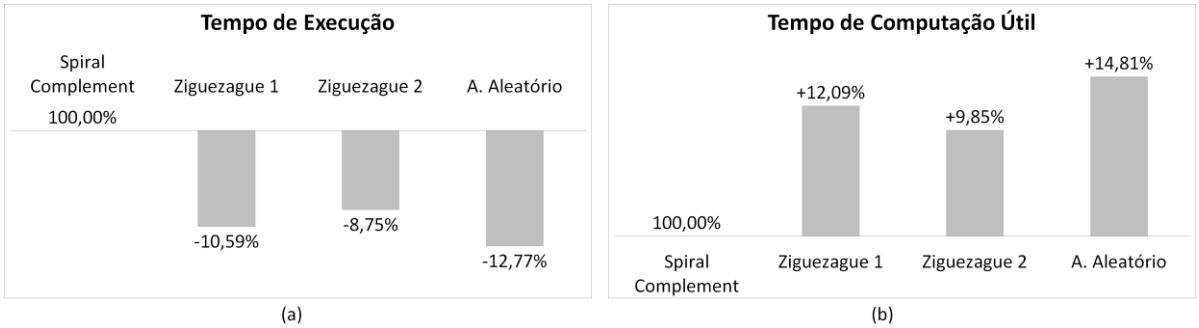

Figura 8 – (a) Tempo de execução da aplicação DCT-2D e o (b) tempo de computação útil.

A Figura 9-A mostra que todos os métodos propostos experimentam uma redução de cerca de 40% na quantidade de *flits* transmitidos quando comparados ao algoritmo original. Provavelmente a energia economizada com retransmissões foi, em parte, útil na execução da

HOLOS ISSN 1807 - 1600

computação, visto que quanto mais as RPUs foram utilizadas (cerca de 9% a 15% a mais) não se observou um aumento do consumo de energia na mesma proporção, conforme Figura 9-B.

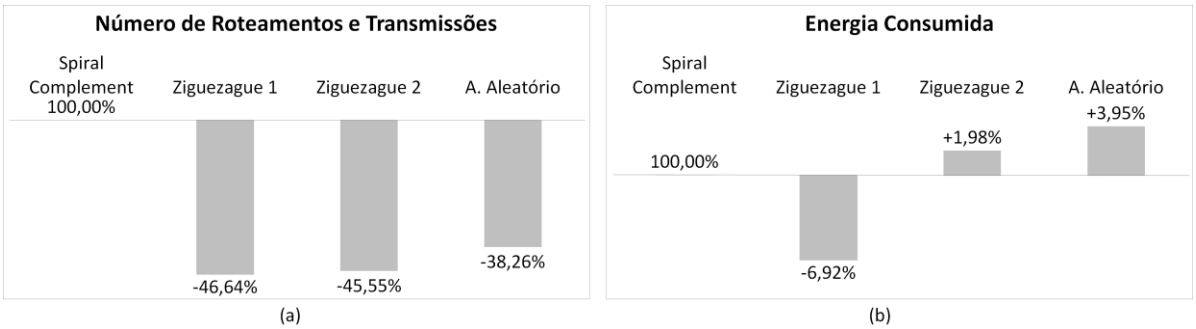

Figura 9 – (a) Número de roteamentos e transmissões totais e (b) energia consumida.

A partir da distribuição das execuções mostradas na Figura 10, pode-se avaliar o quanto as colisões nas RPUs ligadas às MAUs e IOMAU prejudicam o desempenho do sistema. Para todos os métodos é evidente que a maioria das instruções (entre 60% a 85% delas) é computada nestas RPUs, decorrente de colisões e execuções localizadas. Isso impede que os pacotes penetrem e as execuções experimentem uma melhor distribuição no sistema.

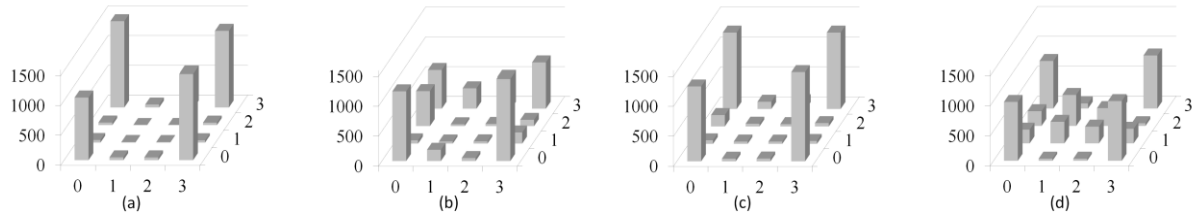

Figura 10 – Execuções por RPU para o algoritmo (a) *Spiral Complement*, (b) Ziguezague 1, (c) Ziguezague 2 e (d) Algoritmo Aleatório - caso médio.

5 CONSIDERAÇÕES FINAIS

A melhora apresentada pelas diferentes propostas de roteamento para o pacote dentro da IPNoSys é acompanhada da diminuição das transmissões e da uniformidade da distribuição das execuções. Entretanto, o sistema testado usa a transmissão dos pacotes de forma recorrente, não apenas realizando a distribuição das tarefas, mas continuamente, produzindo um atraso desnecessário na execução da computação.

O algoritmo aleatório proposto mostrou um desempenho melhor em termos de tempo de execução total da aplicação DCT-2D frente aos outros algoritmos. Este resultado foi seguido pelo aumento no tempo de uso de RPU (14,81% maior, quando comparado ao *Spiral Complement*). Apesar disto, o algoritmo Ziguezague 1 obteve desempenho semelhante, conseguindo uma redução ainda maior no número de retransmissões e na energia total requerida pelo sistema. Portanto, do ponto de vista energético, este algoritmo ganha destaque frente aos demais.

6 TRABALHOS FUTUROS

Foi observado que, na IPNoSys, onde a computação útil está organizada em pacotes, obtém-se melhor desempenho na medida em que a arquitetura diminui o tempo entre a entrada

HOLOS ISSN 1807 - 1600

do pacote no sistema e a saída dos resultados gerados. Como proposta, novas versões do sistema IPNoSys podem explorar uma maior agilidade na distribuição de pacotes na rede, minimizando as transmissões entre *buffers* de RPUs vizinhas e, consequentemente os atrasos na execução. Uma proposta interessante poderia ser extraída da ideia de chaveamento de pacotes VCT (*Virtual Cut Through*), proposto inicialmente por (Kermany et al., 1979). Neste caso, o pacote de tamanho *m* seria injetado na rede, sendo repassado para a RPU que possuir *buffer* na quantidade necessária para alocá-lo. O mecanismo de VCT possibilitaria o salto dos *buffers* intermediários, realizando uma espécie de chaveamento de circuito em direção à RPU que aloca e executa o pacote.

Outra ideia seria transformar o sistema IPNoSys em uma arquitetura essencialmente *dataflow*. Onde pacotes de controle seriam injetados, informando como as RPUs deveriam processar e rotear os pacotes de dados que trafegassem por essas. Logo, todo pacote identificado por um *tag*[2] específico seria computado e roteado conforme o que foi previamente definido pelos pacotes de configuração/reconfiguração. Os resultados obtidos seriam retirados por meio de portas locais às RPUs que compõe o sistema. Esta abordagem é interessante do ponto de vista do desempenho, em contrapartida várias modificações seriam necessárias no esquema de computação e na arquitetura da IPNoSys, criando-se uma plataforma completamente distinta da original.

7 REFERÊNCIAS

1. ARAÚJO, S. R. F. d. Projeto de Sistemas Integrados de Propósito Geral Baseados em Redes em Chip - Expandindo as Funcionalidades dos Roteadores para Execução de Operações. PhD thesis, Universidade Federal do Rio Grande do Norte, 2012.
2. ARAÚJO, S. e OLIVEIRA, B. C. e SILVA, I. S. Using NoC routers as processing elements. In: SBCCI, Natal, Brazil, ACM, 2009.
3. ARAÚJO, S. e OLIVEIRA, B. C. e COSTA, M. e SILVA, I. S. IPNoSys: uma nova arquitetura paralela baseada em redes em chip. In: IX Simpósio em Sistemas Computacionais - WSCAD SSC 2008. SBC, 2008.
4. CRUZ, M. O. d. Roteamento em zigue-zague para a plataforma ipnosys. Trabalho de conclusão de curso de ciência da computação pela Universidade Federal do Rio Grande do Norte, 2013.
5. CRUZ, M. O., KREUTZ, M., FERNANDES, S. An experimental evaluation of a combination of features on the IPNoSys. In: III Simpósio Brasileiro de Engenharia de Sistemas Computacionais (SBESC 2013), Niterói/RJ, 2013.
6. KERMANI, P. and KLEINROCK, L. Virtual cut-through: a new computer communication switching technique. Computer Networks, 3:267–286, 1979.
7. LEE, S. E., BAHN, J. H., YANG, Y. S., AND BAGHERZADEH, N. A generic network interface architecture for a networked processor array (nepa). In Lecture Notes in Computer Science, volume 4934, pages 247–260. Architecture of Computing Systems – ARCS, 2008.
8. ZEFERINO, C. A. Redes-em-Chip: Arquiteturas e Modelos para Avaliação de Área e Desempenho. PhD thesis, Universidade Federal do Rio Grande do Sul, 2003.

[2] Numeração que identifica um conjunto de dados a ser computado e roteado pela RPU previamente configurada.