Estudo de Heurísticas para Mapeamento Dinâmico de Tarefas sobre a Plataforma HeMPS

Ezequiel L. Vidal[1], **Aline V. Mello**[1], **Ewerson L.S. Carvalho** [2], **Claudio Schepke**[1]

[1]Universidade Federal do Pampa (UNIPAMPA)
Av. Tiarajú, 810 – 97546-550 – Alegrete – RS – Brasil

[2]Universidade Federal do Rio Grande - (FURG)
Av. Itália, Km 8 – 96203-900 – Rio Grande – RS – Brasil

{ezequielvidal.ti, alinemello, ewersoncarvalho, schepke}@gmail.com

Abstract. *There is a tendency that Multiprocessor System-on-Chip (MPSoC) will be composed of dozens or hundred of processing elements, allowing the execution of many tasks in parallel. Therefore, efficient resource allocation strategies need to be developed. In this context, this work investigates the performance of the heuristics for dynamic task mapping First Free, Nearest Neighbor, Path Load and Best Neighbor in HeMPS MPSoC. The Best Neighbor heuristic presented the best result in relation to the occupation of the communication channels of the MPSoC, with reduction of approximately 32% when compared to the First Free heuristic. However, Best Neighbor presented simulation time up to 24.21% higher than First Free heuristic due to the complexity of its algorithm.*

Resumo. *Há uma tendência que sistemas intrachip multiprocessados (MPSoC) sejam compostos por dezenas ou centenas de elementos de processamento, permitindo a execução de muitas tarefas em paralelo. Assim, estratégias de alocação de recursos eficientes precisam ser desenvolvidas. Neste sentido, este trabalho investiga o desempenho das heurísticas de mapeamento de tarefas First Free, Nearest Neighbor, Path Load e Best Neighbor no MPSoC HeMPS. A heurística Best Neighbor apresentou o melhor resultado em relação à ocupação dos canais de comunicação do MPSoC, com redução de aproximadamente 32% quando comparada a heurística First Free. No entanto, essa heurística apresentou tempo de execução até 24,21% superior a heurística First Free devido à complexidade de seu algoritmo.*

1. Introdução

Multi-Processor Systems on Chip (MPSoCs) são sistemas que integram em um único circuito vários elementos de processamento (PEs), memórias e *IP-Cores* específicos, geralmente interconectados através de uma *Network on Chip* (NoC). Alguns exemplos incluem o processador de 72 núcleos Tile-Gx72 [Mellanox 2015], o processador MPPA de 256 núcleos da Kalray [De Dinechin et al. 2014] e mais recentemente o processador de 1000 núcleos desenvolvido em parceria entre a IBM e a universidade UC Davis [Bohnenstiehl et al. 2016].

O ato de escolher apropriadamente PEs de um MPSoC para alocar as tarefas de uma dada aplicação é denominado de mapeamento de tarefas [Mandelli 2011]. O mapeamento de tarefas está na classe de problemas *NP-Hard* [Garey and Johnson 1979], sendo

um dos problemas de otimização combinatória mais desafiadores que existem. Quando o número de PEs é pequeno, é possível obter soluções ótimas em tempo consideravelmente curto. Entretanto, MPSoCs possuem uma quantidade de PEs cada vez maior, o que torna imprescindível que estratégias de alocação de recursos eficientes sejam desenvolvidas [Ding et al. 2014].

As estratégias de mapeamento consideram diferentes parâmetros, como: tempo de computação da aplicação, comunicação entre as tarefas da aplicação e taxa de ocupação dos canais da rede. [Chen et al. 2012] propõem heurísticas que consideram somente o tempo de computação da aplicação. As heurísticas de [Ng et al. 2015] consideram o tempo de computação da aplicação e a comunicação entre as tarefas da aplicação. Já [Carvalho et al. 2010] propõem heurísticas que consideram tanto a comunicação entre as tarefas da aplicação quanto a taxa de ocupação dos canais da NoC para gerar um mapeamento eficiente.

Com relação ao momento em que o mapeamento é realizado, ele é dito *estático* ou *dinâmico*. O mapeamento estático é concebido em tempo de projeto, tirando proveito do fato de poder usar algoritmos mais complexos e avaliar um maior número de alternativas de mapeamento. No entanto, o mapeamento estático não consegue lidar com cargas de trabalho dinâmicas chegando no MPSoC. Já o mapeamento dinâmico acontece em tempo de execução, portanto consegue lidar com novas tarefas ou aplicações inseridas no sistema. Para isso, exige uma heurística simples e rápida tendo em vista que esta não deve interferir significativamente no desempenho do sistema.

O objetivo deste trabalho é avaliar o desempenho de quatro heurísticas de mapeamento dinâmico de tarefas sobre um MPSoC HeMPS [Carara et al. 2009]: *First Free*, *Nearest Neighbor*, *Path Load* e *Best Neighbor*. O restante deste trabalho está organizado da seguinte forma. A Seção 2 apresenta os trabalhos relacionados sobre mapeamento de tarefas. As heurísticas de mapeamento são descritas na Seção 3. A Seção 4 descreve a metodologia adotada para avaliação. Na Seção 5, os resultados obtidos e uma análise dos mesmos são apresentados. A Seção 6 traz as considerações finais, bem como sugestões de trabalhos futuros.

2. Trabalhos Relacionados

A Tabela 1 apresenta uma lista de trabalhos relacionados sobre mapeamento dinâmico de tarefas em MPSocs. A última linha da tabela corresponde ao presente trabalho. Conforme pode ser observado na Tabela 1, a maioria dos trabalhos relacionados utiliza mapeamento dinâmico para reduzir o consumo de energia e o tempo de execução das aplicações rodando sobre MPSoC, com exceção de [Ost et al. 2013] que foca apenas no consumo de energia. Em [Quan and Pimentel 2015] e [Carvalho et al. 2010] são usados MPSoCs com processadores heterogêneos, enquanto os demais utilizam MPSoCs com processadores homogêneos.

Com relação ao tipo de controle de mapeamento, o centralizado é a estratégia adotada na maioria dos trabalhos, mesmo não sendo escalável. O único trabalho que apresenta um controle distribuído é proposto por [Mendis et al. 2015]. Os trabalhos investigados apresentam diferentes abordagens para o problema de mapeamento. Na abordagem de Remapeamento, após o mapeamento inicial existe a execução periódica de um conjunto de regras que possibilita a migração de tarefas para outros PEs, com o objetivo de me-

Tabela 1. Trabalhos Investigados sobre Mapeamento Dinâmico de Tarefas.

Trabalho	Otimização	Arquitetura	Controle	Abordagem
[Carvalho et al. 2010]	TE, CE	Heterogênea	Centralizado	Evitar Congestionamento
[Ost et al. 2013]	CE	Homogênea	Centralizado	Evitar Congestionamento
[Fattah et al. 2014]	TE, CE	Homogênea	Centralizado	Evitar Congestionamento
[Mendis et al. 2015]	TE, CE	Homogênea	Distribuído	Remapeamento
[Quan and Pimentel 2015]	TE, CE	Heterogênea	Centralizado	Híbrida
[Ng et al. 2015]	TE, CE	Homogênea	Centralizado	Desfragmentação
[Huang et al. 2015]	TE, CE	Homogênea	Centralizado	Evitar Congestionamento
[Singh et al. 2016]	TE, CE	Homogênea	Centralizado	Híbrida
[Seidipiri et al. 2016]	TE, CE	Homogênea	Centralizado	Evitar Congestionamento
Nosso trabalho	TE, CE	Homogênea	Centralizado	Evitar Congestionamento

TE = Tempo de Execução; CE = Consumo de Energia.

lhorar o desempenho do MPSoC. A abordagem Híbrida combina o mapeamento ótimo para as aplicações obtido em tempo de projeto (estático) com um mapeamento dinâmico, baseado no estado atual do sistema. Na abordagem de Desfragmentação, os PEs livres e dispersos (devido a constante alocação e desalocação) são reunidos e redimensionados. Por fim, a abordagem mais utilizada é a de Evitar Congestionamento, em que o foco é diminuir o fluxo de dados nos canais de comunicação, a fim de que não fiquem saturados.

O presente trabalho implementa e avalia as heurísticas de mapeamento dinâmico de tarefas sobre um MPSoC HeMPS [Carara et al. 2009]. Este possui uma arquitetura homogênea com controle centralizado. A avaliação considera o tempo de execução e a ocupação dos canais de comunicação, a fim de evitar congestionamentos. Detalhes da implementação e avaliação são apresentados nas próximas seções.

3. Heurísticas de Mapeamento

Pode-se definir uma aplicação paralela como um conjunto de tarefas que se comunicam entre si. As tarefas executam as instruções da aplicação, de forma paralela. Conforme a Figura 1(a), uma aplicação pode ser representada através de um grafo dirigido cíclico ou acíclico, onde os vértices representam as tarefas que compõem a aplicação, e as arestas representam comunicações entre as tarefas. Os pesos nas arestas indicam as taxas de comunicação (em ambos os sentidos) entre cada par de tarefas comunicantes.

O problema de mapeamento consiste em escolher o melhor posicionamento para cada uma das tarefas das aplicações no MPSoC, ou seja, cada tarefa deve ser atribuída a um elemento de processamento do MPSoC. A Figura 1(b) e Figura 1(c) representam dois exemplos de mapeamentos. No primeiro, as tarefas estão mapeadas aleatoriamente, enquanto o segundo aproxima tarefas comunicantes. Assumindo em ambos os casos um MPSoC com roteamento XY [De Micheli and Benini 2006], pode-se notar que as taxas de ocupação resultantes nos canais diferem, e que o mapeamento otimizado permite melhor aproveitamento dos canais de comunicação do MPSoC, pois exigi menos recursos.

Neste trabalho foram investigadas quatro heurísticas de mapeamento dinâmico de tarefas: *First Free*, *Nearest Neighbor*, *Path Load* e *Best Neighbor*. Essas heurísticas foram escolhidas porque são adequadas à topologia da NoC presente no MPSoC HeMPS e possuem baixa complexidade, reduzindo o impacto no tempo de execução das aplicações.

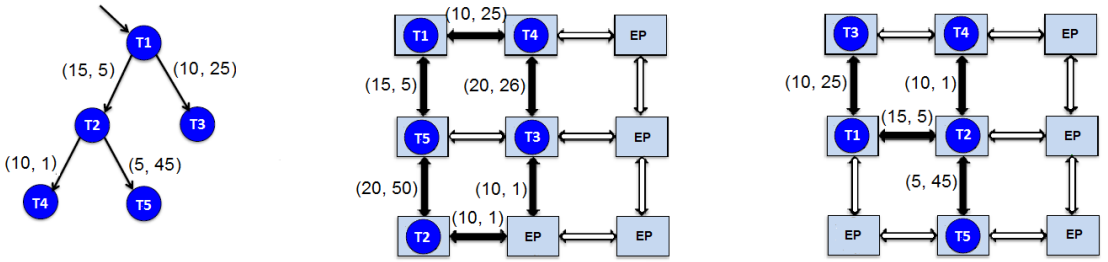


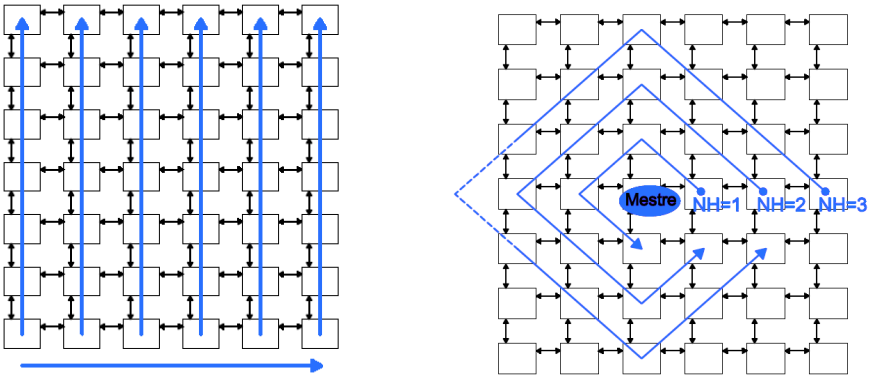
Figura 1. Dois possíveis mapeamentos para uma aplicação.

3.1. First Free (FF)

A heurística *First Free* [Carvalho 2009] seleciona o primeiro recurso livre para mapear uma nova tarefa, sem considerar métricas de desempenho. A escolha deste recurso é diretamente relacionada ao caminho de procura empregado na heurística: da esquerda para direita e de baixo para cima, conforme ilustrado na Figura 2(a).

3.2. Nearest Neighbor (NN)

Na heurística *Nearest Neighbor* [Carvalho 2009], a definição do caminho de procura privilegia a proximidade entre as tarefas comunicantes. Conforme ilustrado na Figura 2(b), a procura por um recurso livre se dá a partir da posição da tarefa mestre (a tarefa que solicitou o mapeamento). A partir desta posição, é seguido um caminho circular, onde os vizinhos são testados de acordo com o número de *hops* necessários para a comunicação.



(a) *First Free.* (b) *Nearest Neighbor.*

Figura 2. Heurísticas de Mapeamento.

3.3. Path Load (PL)

A heurística *Path Load* [Carvalho 2009] possui um caminho de procura idêntico à FF, porém PL considera o peso de comunicação que o mapeamento em determinado recurso provocaria nos canais de comunicação da NoC. A PL calcula o custo para cada recurso livre, realizando o somatório dos pesos dos canais do PE da tarefa mestre (aquela que solicitou o mapeamento) até um PE alvo para a tarefa escrava (aquela que foi solicitada). O PE selecionado é aquele que tem o custo mínimo.

3.4. Best Neighbor (BN)

A heurística *Best Neighbor* [Carvalho 2009] combina a estratégia de busca em espiral de NN com a abordagem de computação de peso dos canais de PL. BN seleciona o melhor vizinho de acordo com a função custo de PL, em vez do primeiro vizinho livre como NN.

4. Metodologia

Esta Seção apresenta a configuração do MPSoC HeMPS, as aplicações e os casos de teste elaborados para investigar as heurísticas, assim como as métricas de avaliação adotadas.

4.1. MPSoC HeMPS

A plataforma HeMPS é um conjunto de ferramentas que permitem a geração e simulação automática de MPSoCs baseado na NoC Hermes (HeMPS) com diferentes configurações, permitindo a exploração do espaço de projeto desde as fases iniciais. A Figura 3 apresenta uma instância de um MPSoC HeMPS [Carara et al. 2009]. Os principais componentes de hardware são o processador Plasma e a NoC Hermes. As tarefas das aplicações a serem executadas são armazenadas em uma memória externa, chamada de repositório de tarefas (*Task Repository*). Além da infra-estrutura de hardware, a estrutura fornece uma infra-estrutura de software que inclui um *microkernel* multitarefa, primitivas de comunicação entre tarefas e suporte para o mapeamento dinâmico de tarefas. O *microkernel* é executado em cada processador, permitindo multitarefa no nível do processador.

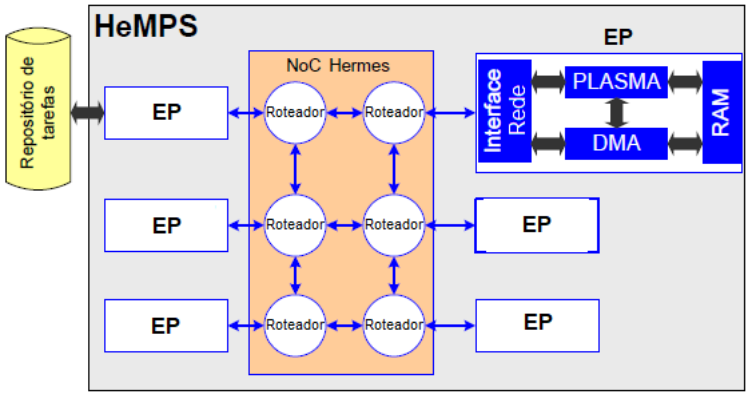


Figura 3. Instância de um MPSoC HeMPS utilizando uma NoC 2x3.

A investigação conduzida neste trabalho é realizada sobre um MPSoC HeMPS com dimensão 5x5, totalizando 25 PEs. O PE 0x0 não recebe nenhuma tarefa pois tem como função gerenciar toda a comunicação. No MPSoC HeMPS, é papel do programador particionar uma dada aplicação paralela em diferentes tarefas. Cada tarefa é descrita na linguagem de programação C e salva em um arquivo.

Para que o mapeamento dinâmico de tarefas funcionasse corretamente, alguns ajustes foram necessários. Embora o grafo de uma dada aplicação já fosse representado em um arquivo na plataforma HeMPS, esse arquivo não era utilizado. A plataforma mapeava as tarefas na ordem de leitura dos arquivos que representavam a aplicação. Portanto, foi preciso realizar modificações no sistema operacional da plataforma para que o grafo da aplicação fosse considerado durante o mapeamento das tarefas.

Adicionalmente, a plataforma não dispõe de uma representação de canais de comunicação em um nível mais alto de abstração, e por isso não fornece *feedback* relativo à ocupação dos canais. Como algumas das heurísticas investigadas consideram um fator custo baseado na ocupação dos canais para decidir o mapeamento, fez-se necessário criar estruturas de dados adicionais na plataforma.

4.2. Casos de Teste

A Figura 4 apresenta as três aplicações utilizadas neste trabalho, representadas por grafos dirigidos, onde os vértices representam as tarefas e as arestas representam o fluxo de dados entre elas. As três aplicações estão disponíveis na plataforma HeMPS [Group 2019]. A aplicação *Dynamic Time Warping* (DTW) é utilizada para encontrar o alinhamento não-linear ótimo entre duas sequências de valores numéricos e representa o conjunto de aplicações que possuem alta taxa de comunicação entre as tarefas. *Moving Picture Experts Group* (MPEG) é uma aplicação de áudio e vídeo que representa as aplicações com carga elevada de processamento. SYNTHETIC é uma aplicação sintética que simula o envio e recebimento de dados, permitindo manipular a taxa de comunicação entre as tarefas.

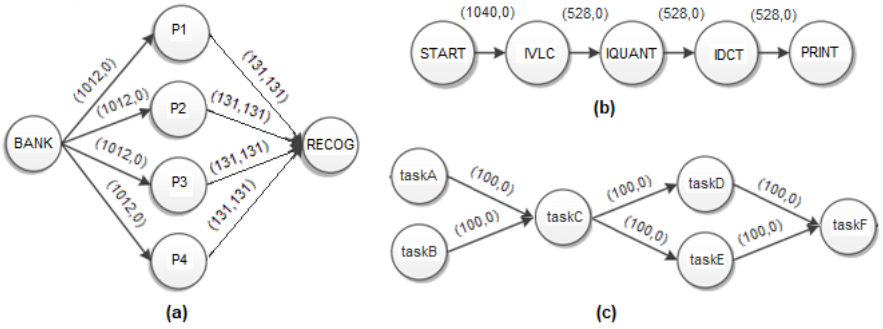


Figura 4. Aplicações utilizadas: (a) DTW, (b) MPEG e (c) SYNTHETIC.

Cinco casos de testes foram elaborados. Nos casos de teste 1, 2 e 3, apenas uma aplicação (DTW, MPEG e SYNTHETIC, respectivamente) é executada no MPSoC. No caso de teste 4, duas aplicações foram executadas paralelamente (DTW e MPEG). Por fim, no caso de teste 5, três aplicações foram executadas em paralelo (DTW, MPEG e SYNTHETIC). Cada caso de teste foi simulado para cada uma das heurísticas (FF, NN, PL e BN), totalizando 20 simulações.

4.3. Métricas de Avaliação

Em geral, um MPSoC permite mais de um tipo de mapeamento. Portanto, é necessário avaliar a qualidade de um dado mapeamento de acordo com um ou mais critérios, a fim de escolher aquele que atende melhor aos requisitos do sistema. Além disso, o impacto que o mapeamento gera no sistema também deve ser considerado, visto que heurísticas lentas irão degradar o desempenho do sistema como um todo.

As heurísticas de mapeamento de tarefas foram avaliadas utilizando as métricas: tempo de mapeamento, tempo de execução e ocupação dos canais de comunicação. O tempo de mapeamento é o tempo dedicado à execução do algoritmo de mapeamento, sendo afetado por sua complexidade. Mesmo que o algoritmo gere um bom resultado, ele

pode demorar para tomar uma decisão. Caso isso ocorra, o tempo de mapeamento de uma aplicação aumenta. O tempo de execução é o tempo total de um caso de teste, englobando o tempo de mapeamento. Ambos os tempos são apresentados em milissegundos (ms). A ocupação dos canais corresponde ao número total de *flits*[1] transmitidos em todos os canais de comunicação do MPSoC. Quando tarefas que se comunicam entre si são mapeadas próximas umas das outras, a tendência é que os canais de comunicação sejam menos exigidos. Em contraste, os canais de comunicação tendem a ficar com cargas maiores de comunicação quando um mapeamento não otimizado é utilizado.

5. Resultados Obtidos

A Figura 5 apresenta os tempos de mapeamento e execução para os casos de teste 1, 2 e 3. A aplicação MPEG possui a maior média de tempo de execução (9,36 ms) devido a sua carga de processamento, seguido de DTW (7,62 ms) e de SYNTHETIC (1,41 ms). Em relação ao tempo de mapeamento, a heurística PL apresentou o pior desempenho em razão da complexidade de seu algoritmo ($O(x^3)$), alcançando um tempo médio de 0,89 ms. A heurística FF apresentou um desempenho médio de 29,92% melhor que PL (0,623 ms), seguido de NN com 29,83% (0,624 ms) e BN com 24,21% (0,674 ms).

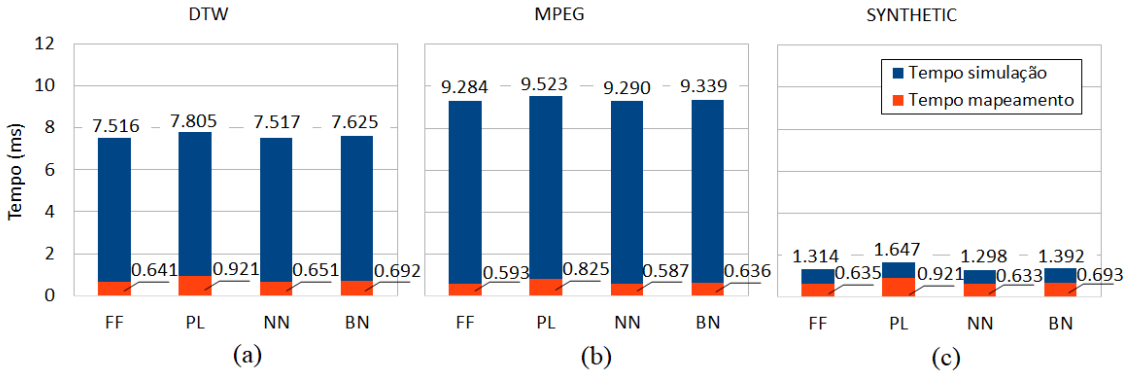


Figura 5. Tempo de mapeamento e execução para os casos de teste 1 (a), 2 (b) e 3 (c).

A Figura 6 apresenta o total de *flits* transmitidos nos canais de comunicação do MPSoC para os casos de testes 1, 2 e 3. A aplicação DTW possui a maior média de *flits* transmitidos entre as tarefas (9.961 *flits*), seguido de MPEG (3.680 *flits*) e de SYNTHETIC (1.175 *flits*). Esses valores estão coerentes com o fluxo de dados apresentado no grafo das aplicações (Figura 4). A heurística BN apresentou as melhores médias de resultados em relação à ocupação dos canais (4.152 *flits*), seguido de NN (4.501 *flits*), PL (4.923 *flits*) e FF (6.179 *flits*). Analisando o ganho em porcentagem, a heurística BN foi 32,80% melhor que a heurística FF, seguido de NN com 27,14% e PL com 20,32%.

A Figura 7 apresenta o tempo de mapeamento e o tempo de execução para os casos de teste 4 e 5, que possuem mais de uma aplicação mapeada no MPSoC. O teste 4 obteve tempo de execução médio de 10,12 ms e o teste 5 apresentou tempo de execução médio de 10,24 ms. Considerando que: (*i*) após mapeadas, as tarefas executam em paralelo no MPSoC; (*ii*) a aplicação MPEG é a mais lenta das aplicações; e (*iii*) faz parte dos casos

[1]Menor unidade de transferência de informação.

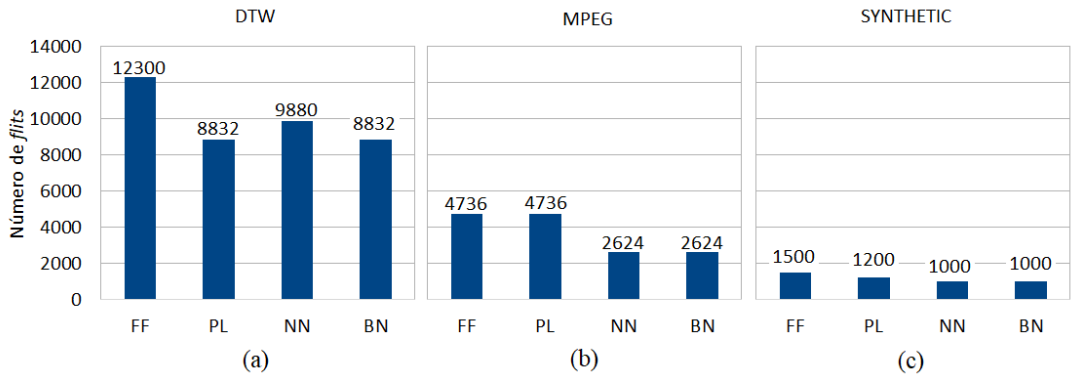


Figura 6. Total de *flits* transmitidos nos casos de teste 1 (a), 2 (b) e 3 (c).

de teste 4 e 5; é natural que ambos apresentem aproximadamente o mesmo tempo de execução. Em relação ao tempo de mapeamento, é possível observar uma diferença entre os casos de teste 4 (tempo médio de 1,41 ms) e 5 (tempo médio de 2,20 ms) devido ao fato de que mais aplicações foram mapeadas no caso de teste 5.

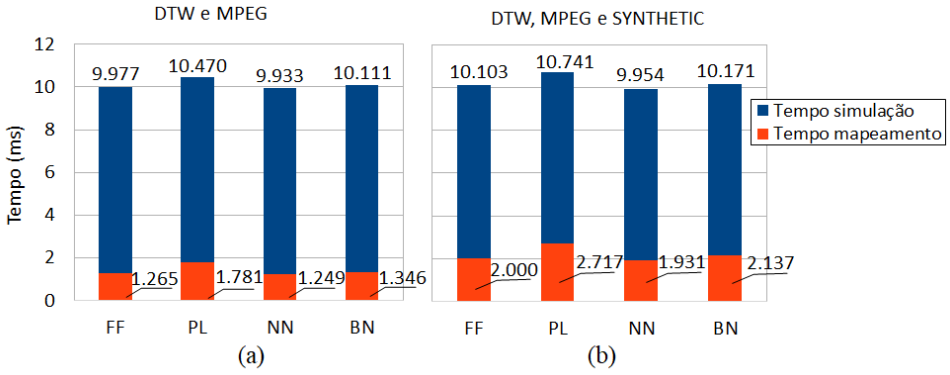


Figura 7. Tempo de mapeamento e execução para os casos de teste 4 (a) e 5 (b).

A Figura 8 apresenta o total de *flits* transmitidos nos canais de comunicação do MPSoC para os casos de teste 4 e 5. O teste 5 obteve um fluxo de dados maior pelo fato de ter uma aplicação a mais que o teste 4. É possível notar também que, para os casos de teste com múltiplas aplicações, os resultados evidenciam como cada heurística de mapeamento influência na ocupação dos canais de comunicação. A heurística BN apresentou a melhor média de ocupação para os dois casos de teste, com 11.856 *flits*, seguido de NN com 13.054 *flits*, PL com 15.258 *flits* e FF com 17.536 *flits*. Analisando a diferença em porcentagem, a heurística BN apresentou ocupação média até 32,39% menor do que FF. As heurísticas NN e PL apresentaram ocupação média de 25,55% e 12,99% menor do que a heurística FF, respectivamente.

6. Considerações Finais

O desempenho de MPSoCs dependem das características de hardware e do software, o que inclui o mapeamento de tarefas adotado. Os resultados apresentados neste artigo mostraram que a heurística *Best Neighbor* ocupou em média 32% menos os canais de comunicação comparado as demais heurísticas investigadas, já que possui um

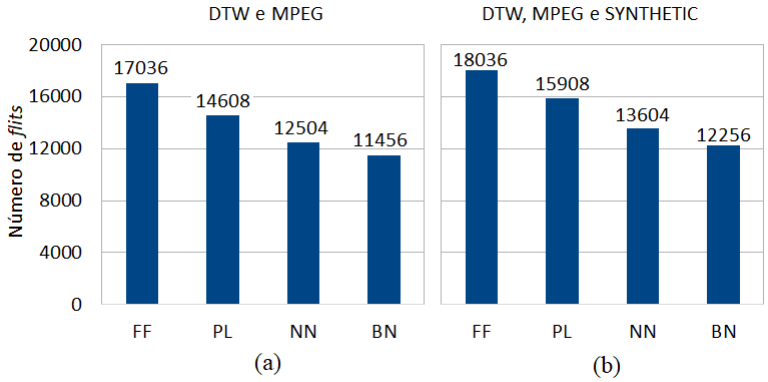


Figura 8. Total de *flits* transmitidos nos casos de teste 4 (a) e 5 (b).

algoritmo mais otimizado. Enquanto as heurísticas *First Free* e *Nearest Neighbor* possuem resultados de tempo de mapeamento e de execução muito semelhantes e menores do que as heurísticas mais elaboradas (*Path Load* e *Best Neighbor*). Portanto, a heurística *Best Neighbor* é indicada para aplicações com alta taxa de comunicação, como a aplicação DTW. Já em aplicações como a MPEG, com elevada carga de processamento, as heurísticas *First Free* e *Nearest Neighbor* são indicadas.

Como trabalho futuro pretendemos repetir as simulações realizadas multiplicando as taxas de comunicação das aplicações para investigar como as heurísticas se comportam com a introdução de novos dados. Ainda, pretendemos investigar outras heurísticas tal como a proposta em [Ng et al. 2015]. Esta implementa a estratégia de migração de tarefas, onde uma dada tarefa já mapeada pode migrar para outro elemento de processamento para reduzir o consumo de energia. Além de investigar estratégias de mapeamento, também faz-se necessário propor novas alternativas.

A plataforma HeMPS com todas as modificações realizadas para a execução desta pesquisa (código, heurísticas e aplicações) encontra-se disponível em: `https://github.com/ezequielvidal/modificacoes_hemps.git`.

Referências

Bohnenstiehl, B. et al. (2016). A 5.8 pJ/Op 115 billion ops/sec, to 1.78 trillion ops/sec 32nm 1000-processor array. In *VLSI Circuits (VLSI-Circuits), 2016 IEEE Symposium on*, pages 1–2, Honolulu, HI, USA. IEEE.

Carara, E. et al. (2009). HeMPS-a framework for NoC-based MPSoC generation. In *Circuits and Systems, 2009. ISCAS 2009. IEEE International Symposium on*, pages 1345–1348, Taipei, Taiwan. IEEE.

Carvalho, E. et al. (2010). Dynamic task mapping for MPSoCs. *IEEE Design & Test of Computers*, 27(5):26–35.

Carvalho, E. L. d. S. (2009). *Mapeamento dinâmico de tarefas em MPSoCs heterogêneos baseados em NoC*. PhD thesis, Pontifícia Universidade Católica do Rio Grande do Sul.

Chen, L., Marconi, T., and Mitra, T. (2012). Online scheduling for multi-core shared reconfigurable fabric. In *Proceedings of the Conference on Design, Automation and Test in Europe*, pages 582–585, Dresden, Germany. EDA Consortium.

De Dinechin, B. D. et al. (2014). Time-critical computing on a single-chip massively parallel processor. In *Design, Automation and Test in Europe Conference and Exhibition (DATE), 2014*, pages 1–6, Dresden, Germany. IEEE.

De Micheli, G. and Benini, L. (2006). *Network on Chips: Technology and Tools*. Morgan Kaufmann Publishers Inc.

Ding, J.-H. et al. (2014). An efficient and comprehensive scheduler on Asymmetric Multicore Architecture systems. *Journal of Systems Architecture*, 60(3):305–314.

Fattah, M. et al. (2014). Adjustable contiguity of run-time task allocation in networked many-core systems. In *Design Automation Conference (ASP-DAC), 2014 19th Asia and South Pacific*, pages 349–354, Singapore. IEEE.

Garey, M. R. and Johnson, D. S. (1979). A Guide to the Theory of NP-Completeness. *WH Freemann, New York*, 70.

Group, G. (2019). Hemps - hermes multiprocessor system on chip. `https://www.inf.pucrs.br/hemps/`. Online; accessed February 2019.

Huang, L.-T. et al. (2015). Wena: Deterministic run-time task mapping for performance improvement in many-core embedded systems. *IEEE Embedded Systems Letters*, 7(4):93–96.

Mandelli, M. G. (2011). Mapeamento dinâmico de aplicações para MPSOCS homogêneos. Master's thesis, Pontifícia Universidade Católica do Rio Grande do Sul.

Mellanox, T. (2015). TILE-Gx72 Processor. Product Brief Description.

Mendis, H., Indrusiak, L., and Audsley, N. (2015). Bio-inspired distributed task remapping for multiple video stream decoding on homogeneous NoCs. In *Embedded Systems For Real-time Multimedia (ESTIMedia), 2015 13th IEEE Symposium on*, pages 1–10, Amsterdam, Netherlands. IEEE.

Ng, J. et al. (2015). Defrag: Defragmentation for efficient runtime resource allocation in noc-based many-core systems. In *Parallel, Distributed and Network-Based Processing (PDP), 2015 23rd Euromicro International Conference on*, pages 345–352, Hong Kong, China. IEEE.

Ost, L. et al. (2013). Power-aware dynamic mapping heuristics for NoC-based MPSoCs using a unified model-based approach. *ACM Transactions on Embedded Computing Systems (TECS)*, 12(3):75.

Quan, W. and Pimentel, A. (2015). A hybrid task mapping algorithm for heterogeneous mpsocs. *ACM Transactions on Embedded Computing Systems (TECS)*, 14(1):14.

Seidipiri, R. et al. (2016). RASMAP: An efficient heuristic application mapping algorithm for network-on-chips. In *Information and Knowledge Technology (IKT), 2016 Eighth International Conference on*, pages 149–155, Hamedan, Iran. IEEE.

Singh, A. K. et al. (2016). Resource and throughput aware execution trace analysis for efficient run-time mapping on MPSoCs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 35(1):72–85.