Framework para Análise de Mapeamento de Tarefas em Sistemas Multriprocessados Intra-Chip usando Planejamento Automático

Jean Carlo Hamerski
Pontifícia Universidade Católica do Rio Grande do Sul
jean.hamerski@acad.pucrs.br

Resumo

O uso de MPSoCs para processamento de aplicações em sistemas computacionais já é uma realidade. Os MPSoCs são sistemas compostos por um grande número de elementos de processamento encapsulados em um único chip com a capacidade de executar aplicações de forma paralela. Em MPSoCs, essas aplicações são executadas na forma de tarefas comunicantes que trocam mensagens entre si. A escolha de qual elemento de processamento irá executar uma determinada tarefa impacta no tempo de execução e na energia consumida pela aplicação como um todo. Este trabalho apresenta a implementação de um framework para análise de mapeamento de tarefas em sistemas multiprocessados intra-chip (MPSoC) que faz uso de funcionalidades de planejamento automático, em especial PDDL, para definir o esquema de mapeamento no MPSoC visando minimizar o tempo de execução da aplicação, ao mesmo tempo em que reduz o consumo de energia. Foi realizada uma modelagem do domínio e problema em PDDL com o objetivo de aproximar tarefas comunicantes que trocam um volume maior de dados entre si. O framework desenvolvido é composto por dois planejadores automáticos (Metric-FF e MauPlanner) e quatro aplicações modelo (MPEG-4, VOP, MWD e RBERG), sendo possível também variar o tamanho do MPSoC. Os experimentos realizados mostram que o uso de planejadores automáticos para definição do mapeamento de tarefas em MPSoCs tem uma limitação em relação ao tamanho do MPSoC.

1. Introdução

A tecnologia *Very-Large-Scale Integration* (VLSI) e sua evolução quanto à redução do tamanho do transistor permitiu a integração de múltiplos núcleos de processamento em um mesmo chip, formando um sistema computacional completo comumente chamado de *Multiprocessor System-on-Chip* (MPSoC).

Os MPSoC executam várias aplicações em paralelo. O impacto da distribuição das tarefas das aplicações nos respectivos elementos de processamento (PEs) e a consequente carga de trabalho gerada por essa comunicação impacta na confiabilidade do sistema e no tempo de execução das aplicações (Chandra 2014) (Mandelli et al. 2015). Em especial, o tempo de execução da aplicação é afetado quando não se leva em consideração o comportamento da comunicação efetuada entre as tarefas que compõem a aplicação (Huang, Yuan, and Xu 2009). Nesse sentido, o método utilizado para definir qual PE irá receber determinada tarefa é crucial quando se deseja minimizar tanto o tempo de execução da aplicação como o consumo de energia provocado pela sua execução.

O problema de mapeamento de tarefas é um problema conhecido na área de sistemas operacionais distribuídos e tem ganhado atenção em vários trabalhos na área de MPSoCs. Algumas das abordagens utilizadas para resolver esse problema são baseadas em computação bioinspirada (Wang et al. 2011), Simulate Annealing (Orsila et al. 2007) e algoritmos genéticos (Bonilha, dos Santos, and Indrusiak 2014).

Este trabalho apresenta um framework para investigação do problema de mapeamento de tarefas usando uma abordagem de planejamento automático. O framework desenvolvido utiliza os planejadores *Metric-FF* (Hoffmann 2003) e *MauPlanner* (Magnaguagno and Meneguzzi 2013). Sobre o planejador *MauPlanner*, foi efetuada uma modificação para habilitá-lo a trabalhar com funções de custo. O framework também inclui o modelo desenvolvido para o domínio do problema de mapeamento de tarefas, assim como a modelagem de 4 aplicações usadas para realização de experimentos. O objetivo do framework desenvolvido é possibilitar a exploração de soluções de planejamento automático para resolução do problema de mapeamento de tarefas em MPSoCs. Não há relatos de trabalhos semelhantes que relacionam esse problema com planejadores automáticos.

O restante deste documento é dividido como segue. Na seção 2 são definidos o domínio e as instâncias dos problemas modelados. A seção 3 apresenta detalhes do framework desenvolvido. Na seção 4 são relatados os experimentos realizados. Por fim, a seção 5 apresenta as conclusões sobre a utilização de planejadores automáticos para resolução de problemas de mapeamento de tarefas em MPSoCs.

2. Formalização do Domínio e do Problema

O problema de mapeamento de tarefas é uma derivação do problema de atribuição quadrática (do inglês, quadratic assignment problem - QAP). QAP (Burkard et al. 1998) é provado ser um problema NP-completo, ou seja, não existe uma

solução para resolvê-lo em tempo polinomial (isso para dimensões suficientemente grandes do problema).

Para o problema de mapeamento de tarefas em um MPSoC 2x2, o tempo para encontrar a melhor solução usando métodos exaustivos é relativamente baixo, enquanto que para uma NoC 10x10, esse tempo pode ser inaceitável se todas as soluções forem exploradas. O uso de heurísticas para resolução do problema de mapeamento é indicado para tornar viável a resolução do problema (Carvalho and Moraes 2009).

Este trabalho faz uso de planejadores automáticos com heurísticas de função de custo para explorar as soluções do problema de mapeamento automático em um MPSoC. Antes de entrar nos detalhes da modelagem realizada, é importante citar algumas definições e simplificações levadas em consideradação no desenvolvimento desse framework:

- a abordagem de mapeamento utilizada é estática, ou seja, o mapeamento é definido em tempo de projeto; tal abordagem é adotada em arquiteturas cuja proposta seja executar um conjunto limitado e predefinido de tarefas de aplicações especificas;
- os PEs são interligados em uma topologia de malha;
- o MPSoC é homogêneo, ou seja, todos os PEs são iguais;
- as aplicações são modeladas através de grafos;
- cada PE pode receber somente uma tarefa;
- modelos temporais não são considerados, ou seja, um PE que terminou a execução de uma tarefa não pode receber outra tarefa para ser executada.

O problema de mapeamento explorado no presente trabalho leva em consideração um MPSoC composto por PEs interligados na forma de malha. Uma instância de um MPSoC 3x3 pode ser visualizado na Figura 1-a. Para fins de representação do MPSoC, neste trabalho 'x' representa linha e 'y' representa coluna, e a numeração dos PEs é incrementada da esquerda para direita e de cima para baixo.

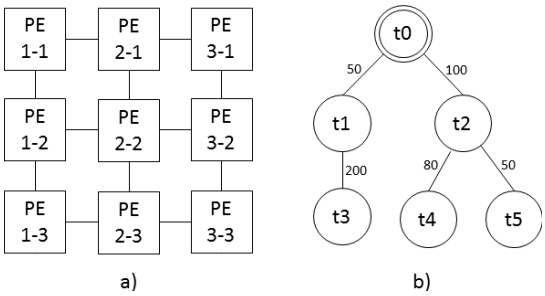


Figura 1: a) MPSoC com 9 elementos de processamento organizados em uma matriz de dimensão 3x3; b) Aplicação com 6 tarefas, sendo t0 a tarefa inicial.

O modelo de aplicação implementado nestre trabalho considera uma modelagem empregada em diversos trabalhos onde a aplicação é representada por um grafo. No grafo, os vértices representam as tarefas da aplicação, e as arestas representam a comunicação efetivada pelas tarefas conectadas. O peso das arestas representa o volume de dados transferido entre as tarefas. A Figura 1-b exibe um exemplo de modelo de aplicação com 6 tarefas, onde a tarefa t0 é a tarefa inicial (representada pelo círculo com linha dupla).

Existem modelos mais detalhados para representar a aplicação, os quais modelam também informações como o tempo de computação das tarefas (Marcon et al. 2005) e a largura de banda requerida para comunicação (Rhee, Jeong, and Ha 2004), os quais não são cobertos no presente trabalho.

Outros trabalhos representam a aplicação por um grafo dirigido. No presente trabalho, a aplicação é representada por um grafo não-dirigido, sendo que uma aresta entre dois nodos no grafo representa a dependência entre uma tarefa que está em execução e que requisita o mapeamento de outra tarefa com a qual irá efetivar a comunicação. Por exemplo, na Figura 1-b, a tarefa 't1' é dependente da tarefa 't3', sendo que o volume de dados transferido entre essas duas tarefas é correpondente ao peso da aresta (200).

Uma aplicação sempre tem apenas uma tarefa inicial, a qual, em um sistema multiprocessado, será iniciada tão logo a aplicação seja executada pelo usuário. As demais tarefas são executadas na medida em que são requisitadas por outras tarefas em execução.

2.1 Modelagem em PDDL

Para modelagem do domínio e problemas relatados no início dessa seção foi utilizada a linguagem PDDL (Planning Domain Definition Language).

Com PDDL é possível modelar todo o problema a ser solucionado por meio de objetos, predicados, ações, estado inicial e objetivo final. Uma vez que a modelagem esteja bem definida e havendo uma solução, a ferramenta exibirá a sequência de ações da solução encontrada. No caso do problema de mapeamento de tarefas, o conjunto de ações resultantes representará a ordem e o posicionamento de cada tarefa no seu respectivo PE.

Na descrição da modelagem implementada no presente trabalho, os PEs e tarefas foram definidos como objetos, sendo representados pelos seguintes predicados:

- *pe ?p*: 'p' é um elemento de processamento;
- *task-ini ?t*: 't' é uma tarefa inicial;
- *task-noini ?t*: 't' é uma tarefa não-inicial;

Também foram criados outros predicados para representar algumas propriedades importantes na modelagem:

- *task-dep ?t1 ?t2*: tarefa 't1' é dependente da tarefa 't2';
- *not-mapped ?t*: tarefa 't' ainda não foi mapeada em nenhum PE;
- *at ?t ?pe*: tarefa 't' está mapeada no elemento de processamento 'pe';
- *not-full ?pe*: elemento de processamento 'pe' não está cheio (com alguma tarefa mapeada nele).

As ações que são aplicadas sobre os estados da modelagem para que o estado objetivo seja alcançado foram representadas da seguinte forma:

- *map-task-ini ?pe ?t*: mapeia 't' em 'pe', com a pré-condição de que 't' seja uma tarefa inicial ainda não mapeada e que 'pe' esteja livre;
- *map-task-noini ?pe1 ?pe2 ?t1 ?t2*: mapeia 't2' em 'pe2', com a pré-condição de que 't2' seja uma tarefa não-inicial e que tenha sido requisitada pela tarefa 't1' já mapeada no 'pe1'; 'pe2' precisa estar livre nesse caso; além disso 't1' deve ser dependente de 't2'.

Uma importante funcionalidade necessária para utilização de planejadores automáticos para resolução de problemas como o de mapeamento de tarefas é a implementação de funções de custo no planejador. Com essa funcionalidade, cada ação tem um custo numérico e, consequentemente, a solução composta por um conjunto de ações terá um custo total. O planejador então escolhe a melhor solução maximizando ou minimizando o custo total, dependendo de como se modela o problema.

No presente trabalho, a ação *map-task-noini* tem um custo representado pela equação demonstrada em (1), onde *comm(?t1,?t2)* representa o volume de dados transferidos entre 't1' e 't2' e *cost(?pe1,?pe2)* representa o número de saltos do caminho mais curto entre os elementos de processamento 'pe1' e 'pe2' na matriz do MPSoC, todos esses valores especificados na descrição do estado inicial do problema. Por exemplo, para o MPSoC da Figura 1-a e para a aplicação da Figura 1-b, a equação definida em (1) retornaria o valor 240 (=80*3) para o par de tarefas 't2' e 't4', mapeados respectivamente em 'PE1-1' e 'PE2-3'.

$$comm(?t1, ?t2) * cost(?pe1, ?pe2) \quad (1)$$

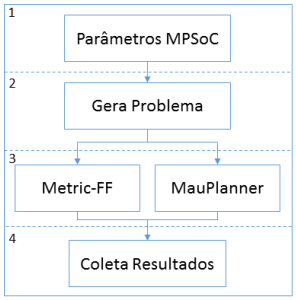
O objetivo da função de custo é direcionar o planejador para que ele mapeie pares de tarefas comunicantes próximas umas das outras, sendo que pares de tarefas que trocam um volume maior de dados tenham prioridade. Com isso, espera-se que o tempo de execução total da aplicação seja reduzido, uma vez que o atraso na comunicação entre as tarefas que trocam uma maior quantidade de dados será menor. Consequentemente, o consumo de energia também tende a reduzir, uma vez que o chaveamento da rede de comunicação do MPSoC será menor se comparado a um mapeamento aleatório das tarefas.

3. Framework

Com o objetivo de possibilitar a exploração das soluções de planejamento automático para diferentes configurações de MPSoCs e aplicações, foi criado dentro do presente trabalho um framework composto pelos seguintes componentes:

- descrições em PDDL do domínio, problemas e aplicações;
- planejador *Metric-FF* (Hoffmann 2003);
- planejador *MauPlanner* (Magnaguagno and Meneguzzi 2013), com modificações adicionais impementadas para aceitar funções de custo na modelagem do problema;
- script para generalização de configurações do MPSoC para diferentes tamanhos do MPSoC.

Um esquema que exibe o fluxo de execução do framework é exibido na Figura 2-a. Cabe salientar que esse fluxo é executado de forma automática a partir da definição dos parâmetros do MPSoC realizado na etapa 1, onde são definidos a dimensão 'X' e 'Y' do MPSoC e o planejador a ser usado. Na etapa 2, a partir dos parâmetros definidos na etapa anterior, o framework gera a descrição PDDL do problema (a descrição PDDL do domínio é unica para qualquer configuração do framework). Na etapa 3, as descrições PDDL do domínio e problema são usadas como entrada do planejador escolhido para essa execução. Por fim, na etapa 4 são coletados os resultados do planejador, ou seja, a sequência de ações que define o mapeamento das tarefas nos respectivos PEs, além de outras informações de desempenho do planejador.



a)

MPEG		
ti	tj	p
0	1	20
1	2	20
1	3	200
1	4	20
1	5	1200
1	6	200
1	8	200
1	9	800
5	7	200
6	7	200
8	10	400
9	10	900
10	11	1360
10	12	480

VOP		
ti	tj	p
0	1	20
0	2	20
1	3	110
2	8	90
3	4	110
4	5	110
5	6	110
5	7	200
6	7	110
7	8	110
8	9	110
9	10	110
10	11	110
10	12	110
11	12	200

MWD		
ti	tj	p
0	1	110
1	2	110
2	3	110
3	9	110
0	4	110
4	5	200
4	6	110
6	7	110
7	8	110
8	9	110
9	10	110
10	11	110

RBERG		
ti	tj	p
0	1	110
1	2	110
2	3	110
0	4	110
1	4	110
1	5	200
2	5	110
2	6	110
3	6	110
4	7	110
5	7	110
5	8	110
6	8	110
7	9	110

b)

Figura 2: a) Esquema do framework com as etapas de execução; b) Aplicações e respectivas dependências e volume de dados entre par de tarefas.

As descrições em PDDL do domínio e problemas seguem a modelagem detalhada na seção 2.1. Como citado, também foram criados modelos descritos em PDDL de 4 aplicações, comumente utilizadas em trabalhos de mapeamento de tarefas, as quais são: MPEG-4, VOP (Video Object Plane), MWD (Multi-Window Display), e RBERG (Integral de Romberg). A Figura 2-b apresenta 4 tabelas com as dependências e volume de dados (*p*) transferido entre o respectivo par de tarefas (*ti* e *tj*). Os valores de volume de dados, assim como o esquema de dependência, utilizados no presente trabalho, foram retirados do trabalho realizado em (Carvalho and Moraes 2009).

4. Resultados

Para avaliar a modelagem realizada em conjunto com os planejadores e aplicações citados da seção 3, foi realizado uma série de experimentos. Especificamente foi analisado o tempo de execução dos planejadores para diferentes configurações do MPSoC. Os experimentos foram realizados em um computador com processador Intel Xeon W3540 (2,93GHz) quad-core, com 6GB de RAM (DDD3-1066).

Em relação ao planejador *MauPlanner*, foram realizados experimentos para avaliar o seu determinismo na escolha de uma solução otimizada de mapeamento de tarefas em um MPSoC variando sua dimensão em 3x3 e 4x4. Para esse experimento, foi utilizado um modelo de aplicação com 4 tarefas. Para o modelo de MPSoC 3x3, o planejador levou um

tempo de 0,18s para gerar um plano otimizado. Ao testar o planejador no modelo de MPSoC 4x4, o mesmo não retornou um resultado após 6 horas de execução, pois avalia um conjunto muito grande de ações ao abrir a árvore de soluções de mapeamento dentro do planejador. Por isso, esse planejador não foi utilizado nos experimentos com um número maior de PEs ou com modelos de aplicações com maior número de tarefas.

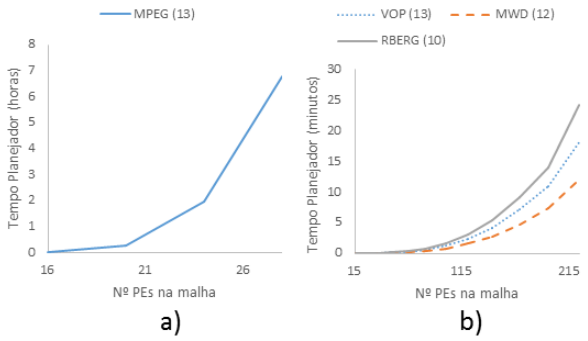


Figura 3: Tempo do Planejador *Metric-FF* variando dimensão do MPSoC para: a) aplicação MPEG; b) aplicações VOP, MWD e RBERG.

Com relação ao planejador *Metric-FF*, foram realizados experimentos para avaliar o tempo que o planejador levou para gerar uma solução otimizada de mapeamento de tarefas para as aplicações citadas, variando-se o tamanho do MPSoC. A Figura 3-a mostra o resultado do tempo de planejamento para a aplicação MPEG em configurações de MPSoC 4x4 (16 PEs), 5x5 (25 PEs), variando-se até 15x15 (225 PEs). Como pode ser visualizado, para essa aplicação o planejador aumenta exponencialmente o tempo de planejamento, o que não é um comportamento desejável, pois para tamanhos ainda maiores de MPSoC, o planejador demoraria horas para geração do resultado.

Já para as outras aplicações (Figura 3-b), o tempo que o planejador levou para gerar o resultado foi satisfatório para tamanhos de MPSoC de até 36 PEs, aumentando esse tempo exponencialmente para tamanhos de MPSoC ainda maiores.

A comparação entre os dois planejadores não foi realizada, uma vez que o planejador *Metric-FF* utiliza algoritmos de otimização sobre o conjunto de soluções, os quais ainda não foram implementadas na versão atual do planejador *MauPlanner* com função de custo. Com essa otimização, o planejador não avalia todas as possíveis soluções de mapeamento e sim somente aquelas que são mais "promissoras". Em versões futuras, esses algoritmos serão integrados ao planejador *MauPlanner*.

5. Conclusão

Ao fim desse trabalho, pode-se verificar que é possível resolver problemas de mapeamento de tarefas em MPSoCs utilizando planejamento automático, porém, para problemas que precisam explorar um grande número de soluções, o uso dos planejadores automáticos testados torna-se inviável devido ao tempo que eles levam para gerar uma solução otimizada.

Em trabalhos futuros, pretende-se implementar algoritmos de otimização sobre o planejador *MauPlanner* com função de custo e reavaliar sua aplicação nesse tipo de problema. A partir das melhorias a serem realizadas no planejador *MauPlanner*, os resultados dos mapeamentos serão utilizados em experimentos de tempo de execução de aplicações em um MPSoC real, possibilitando uma avaliação qualitativa dos resultados alcançados quanto ao tempo de execução e consumo de energia.

Outra possibilidade é explorar a utilização de planejadores implementados de hardware, o que pode ser útil em arquiteturas multiprocessadas como a utilizada no presente trabalho.

Referências

Bonilha, I.; dos Santos, O.; and Indrusiak, L. 2014. Heuristics for mapping real-time applications to noc-based architectures using genetic algorithms. In *Computing Systems Engineering (SBESC), 2014 Brazilian Symposium on*.

Burkard, R. E.; Çela, E.; Pardalos, P. M.; and Pitsoulis, L. S. 1998. The quadratic assignment problem.

Carvalho, E., and Moraes, F. 2009. Mapeamento dinâmico de tarefas em mpsocs heterogêneos baseados em noc.

Chandra, V. 2014. Quantifying workload dependent reliability in embedded processors. In *Design Automation Conference (ASP-DAC), 2014 19th Asia and South Pacific*.

Hoffmann, J. 2003. The metric-ff planning system: Translating "ignoring delete lists"to numeric state variables. *J. Artif. Int. Res.* 20(1):291–341.

Huang, L.; Yuan, F.; and Xu, Q. 2009. Lifetime reliability-aware task allocation and scheduling for mpsoc platforms. In *Design, Automation Test in Europe Conference Exhibition, 2009. DATE '09.*, 51–56.

Magnaguagno, M. C., and Meneguzzi, F. R. 2013. Implementação de um planejador baseado em busca heurística.

Mandelli, M.; Ost, L.; Sassatelli, G.; and Moraes, F. 2015. Trading-off system load and communication in mapping heuristics for improving noc-based mpsocs reliability. In *Quality Electronic Design (ISQED), 2015 16th International Symposium on*, 392–396.

Marcon, C.; Calazans, N.; Moraes, F.; Susin, A.; Reis, I.; and Hessel, F. 2005. Exploring noc mapping strategies: an energy and timing aware technique. In *Design, Automation and Test in Europe, 2005. Proceedings*, 502–507 Vol. 1.

Orsila, H.; Kangas, T.; Salminen, E.; Hämäläinen, T. D.; and Hännikäinen, M. 2007. Automated memory-aware application distribution for multi-processor system-on-chips. *J. Syst. Archit.* 53(11):795–815.

Rhee, C.-E.; Jeong, H.-Y.; and Ha, S. 2004. Many-to-many core-switch mapping in 2-d mesh noc architectures. In *Computer Design: VLSI in Computers and Processors, 2004. ICCD 2004. Proceedings. IEEE International Conference on*, 438–443.

Wang, J.; Li, Y.; Chai, S.; and Peng, Q. 2011. Bandwidth-Aware Application Mapping for NoC-Based MPSoCs. 1.