A alocação de tarefas em sistemas com múltiplos processadores, considerando a confiabilidade das rotas

Task allocation in multiprocessor systems considering reliability of routes

Autor: Francisco Laércio de Morais
Orientador: Gustavo Girão Barreto da Silva
Colaboradores: Mônica Magalhães Pereira, Ivanovitch Medeiros Dantas da Silva

Resumo

Ao longo dos anos a crescente necessidade computacional motivou fabricantes de processadores a criarem produtos com alta frequência de operação. Com o tempo, percebeu-se que o grande consumo de potência tornou-se proibitivo e que o poder de miniaturização de transistores deveria ser utilizado de uma maneira diferente. A criação de sistemas multiprocessados tenta utilizar soluções de paralelismo para contornar estes desafios. Tais sistemas utilizam estruturas fortemente inspiradas em redes de computadores para favorecer o paralelismo de comunicação. Este cenário lida com desafios de dependabilidade (tolerância a falhas e confiabilidade) e comunicação, dado que o alto grau de miniaturização dos transistores influencia diretamente na taxa de falhas dos componentes e na robustez de comunicação. Neste projeto pretende-se investigar e medir a confiabilidade de soluções de comunicação em sistemas multiprocessados, além de investigar métricas e algoritmos de alocação de tarefas que considerem a maximização do paralelismo e da confiabilidade.

Palavras-chave: MPSoCs.NoCs.confiabilidade.tolerância a falhas

Abstract

Over the years the growing computational necessity motivated processors manufacturers to create products with high frequency operation. Over time, it was realized that the great power consumption has become prohibitive and that the power of miniaturization of transistors should be used in a different way. The creation of multiprocessor systems attempt to use parallelism solutions to overcome these challenges. These systems use highly inspired structures in computer networks to facilitate the parallel communications. This scenario deals with challenges of dependability (reliability and fault tolerance) and communication, as the high degree of miniaturization of transistors directly influences the failure rate of the components and the robustness of communication. In this project we intend to investigate and measure the reliability of communication solutions in multiprocessor systems, and investigate metrics and task allocation algorithms that consider maximizing parallelism and reliability.

Keywords: MPSoCs.NoCs.reliability.fault tolerance

1. Introdução

Em sistemas computacionais convencionais, o uso de elementos de processamento com frequências de operação cada vez maiores apresenta pouco ganho em desempenho (proporcionalmente falando) e um elevado consumo energético. Diante deste problema e graças ao alto grau de integração que temos hoje em dia, sistemas computacionais mudaram o foco para explorar soluções compostas de múltiplos processadores em um único chip. Estas soluções são chamadas de MPSoCs – do inglês, Multiprocessor System-on-Chip [1].

Com o objetivo de ter um mecanismo de comunicação mais eficiente, foi proposta uma solução de comunicação em MPSoCs fortemente inspirada em redes de computadores convencionais chamada Redes-em-Chip – do inglês, Networks-on-Chip (NoC) [2]. Uma NoC é uma estrutura de comunicação composta por vários roteadores interconectados seguindo uma certa topologia (i.e. Mesh, Torus, Cubo, Árvore, etc). Cada roteador é associado a um recurso (processadores, memórias ou

módulos de I/O, por exemplo). Além do seu alto grau de escalabilidade, NoCs tem como principal característica a capacidade de prover comunicação paralela entre os componentes do sistema.

Diante deste cenário com vários processadores e a capacidade de comunicação paralela, um novo desafio a ser estudado é a capacidade de alocar tarefas no sistema de modo que a execução da aplicação atinja o seu máximo potencial de desempenho. Esta não é uma tarefa trivial já que certas tarefas têm maior necessidade de comunicação do que outras. Determinar qual alocação de tarefas entre os processadores do sistema por si só já é algo que tem gerado diversas análises e estudos. Entretanto, devido a fatores como miniaturização tecnológica, desgaste físico e interferências eletromagnéticas, algumas das conexões entre os componentes são passíveis de falha [3]. Este é um problema grave que tem motivado diversos estudos ao longo dos anos, com o objetivo de determinar soluções que reparem ou superem falhas que ocorram dentro do chip de modo a manter o sistema funcionando.

Neste projeto pretende-se realizar uma análise de confiabilidade de uma rede em chip considerando a confiabilidade de comunicação entre todos os possíveis pares de roteadores existentes nela. Neste estudo serão consideradas variáveis como topologia e algoritmos de roteamento. Com esta análise em mãos, é possível estabelecer os trechos mais confiáveis desta rede e utilizar esta informação para guiar a alocação de tarefas no sistema. Assim, pode-se maximizar o paralelismo do sistema espalhando as tarefas entre os processadores até um ponto onde a confiabilidade é tolerável. Adicionalmente, o nicho de aplicação explorado pelo trabalho é vital para o desenvolvimento de aplicações de missão crítica (militar, aeroespacial, segurança, veicular), área ainda em desenvolvimento no país. As metodologias propostas podem contribuir para o desenvolvimento de software com alto valor agregado, preenchendo uma lacuna científica do país.

2. Materiais, técnicas e métodos

Para cumprir os objetivos do projeto a metodologia inicial adotada foi passar os conceitos básicos envolvidos no escopo do projeto para o bolsista. Nessa primeira etapa ele ficou encarregado de ler alguns documentos da área, slides de

aulas sobre o tema, elaborados pelo professor Ivanovitch para uma disciplina de tópicos especiais que tinha como tema abordado a “Análise de confiabilidade de Sistemas Críticos”, o livro e também foi apresentada ao bolsista um sistema que já trabalha com árvore de falhas (algo que será posteriormente utilizado no projeto). Com isso foi possível inserir o aluno no escopo do projeto e passar para a próxima etapa, que foi o início do desenvolvimento da aplicação que faria simulações de comunicações entre tarefas.

Nessa segunda etapa foi elaborada uma aplicação que, dado uma NoC (organizada em uma topologia pré conhecida, que é passada para a aplicação através de uma matriz de adjacência) e grafo de tarefas padronizado (STF) [4], a aplicação faz a alocação dessas tarefas na NoC e gera um arquivo que contem os caminhos utilizados nas comunicações entre essas tarefas. Como não se tinha algoritmos adaptativos, a falha de um roteador no caminho entre tarefa A e B, nesse cenário, é o suficiente para que a comunicação entre essas duas tarefas falhe. Partindo dai, houve a inclusão da ferramente já existente que trabalha com árvores de falhas, a SHARPE tool [5].

Na terceira etapa foi feito uma nova aplicação que faz a tradução dos caminhos gerados pela primeira aplicação para o arquivo de entrada da ferramenta SHARPE. A partir desse arquivo de entrada, é possível montar uma árvore de falha dentro dessa ferramenta e extrair dela várias informações sobre a NoC que está sendo representada pela árvore de falha gerada, a principal delas seria a confiabilidade, que é calculada baseado nos roteadores usadas em cada comunicação e na confiabilidade de cada roteador (ou seja, dado que um roteador possui uma taxa de falha, essa taxa é levada em consideração no calculo da confiabilidade da NoC).

Na quarta etapa foi feito um aprimoramento da primeira aplicação desenvolvida, agora ela também gera estatísticas sobre a NoC, como a taxa de uso de cada roteador, em quantas comunicações cada um foi usado, etc. Nessa etapa também foi introduzido um módulo de injeção de falhas, que permite fazer a mesma simulação das comunicações entre as tarefas só que agora simulando que alguns roteadores não estão funcionando. Também é gerado estatísticas sobre os testes

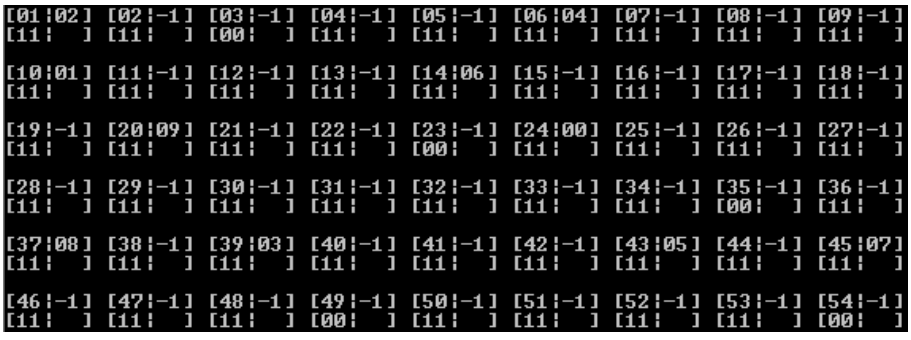
realizados com a injeção de falhas. Nesta quarta o sistema começa a se parecer mais com o que acontece na vida real, uma vez que até então era considerado que os roteadores sempre funcionariam com a taxa de falhas em 0%.

Partindo dai, numa quinta etapa, foi desenvolvido um algoritmo de roteamento adaptativo simples, que já consegue evitar que algumas comunicações falhem ao tomar decisões sobre o caminho que a comunicação deve usar e desviando de roteadores com falhas. Também foi feito testes e levantado as estatísticas utilizando o algoritmo adaptativo, e comparando com os experimentos que utilizam o algoritmo não adaptativo é visível o resultado positivo. Pretende-se, agora, tornar o algoritmo adaptativo mais robusto para a realização de novos testes e, a partir dos dados coletados, se ter uma ideia de como se comporta a relação entre algoritmo de roteamento e a alocação das tarefas na NoC, para que possamos desenvolver um sistema que, baseado nas comunicações que serão realizadas e no algoritmo de roteamento utilizado na NoC, fazer a alocação das tarefas de forma a minimizar a taxa de falhas nessas comunicações.

3. Resultados

Como resultados obtidos até agora podemos mostrar os documentos gerados pela nossa aplicação nos experimentos executados até agora:

Figura 1 – Representação de uma NoC na aplicação.



Fonte: *printscreen* do console de saída da aplicação.

A figura 1 é o *printscreen* do console da aplicação após a execução de uma simulação. Ela mostra a representação de uma NoC que foi organizada utilizando a topologia *mesh*. Cada conjunto de três valores representa algum dado sobre o roteador em questão. O valor do canto esquerdo superior é a identificação do roteador, o do canto direito superior é referente a aplicação/tarefa que está alocada

nos recursos daquele roteador (o valor -1 nesse campo indica que não há nenhuma aplicação alocada nos recursos do roteador), e o do canto esquerdo inferior é referente ao funcionamento do roteador ou de seus recursos (11 representa que está tudo funcionando, 00 indica que foi injetada uma falha nesse roteador ou em seus recursos, e na simulação ele não poderá ser utilizado).

Figura 2 – Um dos arquivos de saída de uma simulação na aplicação.

```
XY Padrao Nao Adaptativo

-- Uso dos roteadores [Qtd de comunicacoes que usam cada roteador]

Router 1: 0     |   Router 2: 0     |   Router 3: 0     |   Router 4: 0     |   Router 5: 0
Router 6: 0     |   Router 7: 0     |   Router 8: 1     |   Router 9: 0     |   Router 10: 1
Router 11: 1    |   Router 12: 1    |   Router 13: 1    |   Router 14: 2    |   Router 15: 0
Router 16: 0    |   Router 17: 1    |   Router 18: 0    |   Router 19: 1    |   Router 20: 0
Router 21: 0    |   Router 22: 0    |   Router 23: 1    |   Router 24: 0    |   Router 25: 0
Router 26: 1    |   Router 27: 0    |   Router 28: 1    |   Router 29: 0    |   Router 30: 0
Router 31: 1    |   Router 32: 1    |   Router 33: 0    |   Router 34: 0    |   Router 35: 1
Router 36: 1    |   Router 37: 1    |   Router 38: 0    |   Router 39: 0    |   Router 40: 1
Router 41: 1    |   Router 42: 0    |   Router 43: 1    |   Router 44: 1    |   Router 45: 1
Router 46: 0    |   Router 47: 0    |   Router 48: 0    |   Router 49: 1    |   Router 50: 2
Router 51: 3    |   Router 52: 3    |   Router 53: 5    |   Router 54: 1    |

-- Uso dos roteadores [Qtd de comunicacoes que usam cada roteador/Total de comunicacoes]

Router 1: 0.00 %    |   Router 2: 0.00 %    |   Router 3: 0.00 %    |   Router 4: 0.00 %    |   Router 5: 0.00 %
Router 6: 0.00 %    |   Router 7: 0.00 %    |   Router 8: 11.11 %   |   Router 9: 0.00 %    |   Router 10: 11.11 %
Router 11: 11.11 %  |   Router 12: 11.11 %  |   Router 13: 11.11 %  |   Router 14: 22.22 %  |   Router 15: 0.00 %
Router 16: 0.00 %   |   Router 17: 11.11 %  |   Router 18: 0.00 %   |   Router 19: 11.11 %  |   Router 20: 0.00 %
Router 21: 0.00 %   |   Router 22: 0.00 %   |   Router 23: 11.11 %  |   Router 24: 0.00 %   |   Router 25: 0.00 %
Router 26: 11.11 %  |   Router 27: 0.00 %   |   Router 28: 11.11 %  |   Router 29: 0.00 %   |   Router 30: 0.00 %
Router 31: 11.11 %  |   Router 32: 11.11 %  |   Router 33: 0.00 %   |   Router 34: 0.00 %   |   Router 35: 11.11 %
Router 36: 11.11 %  |   Router 37: 11.11 %  |   Router 38: 0.00 %   |   Router 39: 0.00 %   |   Router 40: 11.11 %
Router 41: 11.11 %  |   Router 42: 0.00 %   |   Router 43: 11.11 %  |   Router 44: 11.11 %  |   Router 45: 11.11 %
Router 46: 0.00 %   |   Router 47: 0.00 %   |   Router 48: 0.00 %   |   Router 49: 11.11 %  |   Router 50: 22.22 %
Router 51: 33.33 %  |   Router 52: 33.33 %  |   Router 53: 55.56 %  |   Router 54: 11.11 %  |

-- 7 Comunicacoes tiveram sucesso.
-- Porcentagem de falha: 22.22 %


-- Foram avaliadas 9 comunicacoes:

App 1 to App 0 > Path: 51 52 53
App 2 to App 0 > Path: Nao foi possivel seguir a partir do router 16.
App 3 to App 0 > Path: 31 40 49 50 51 52 53
App 4 to App 0 > Path: 8 17 26 35 44 53
App 5 to App 1 > Path: 14 23 32 41 50 51
App 6 to App 5 > Path: 37 28 19 10 11 12 13 14
App 7 to App 0 > Path: 43 52 53
App 8 to App 6 > Path: Nao foi possivel seguir a partir do router 39.
App 9 to App 0 > Path: 36 45 54 53
```

Fonte: *printscreen* do arquivo.

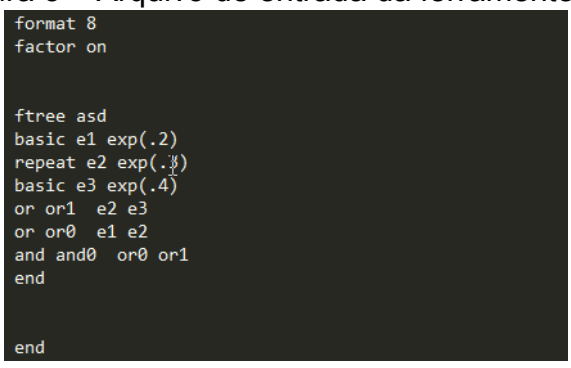
A figura 2 é o *printscreen* de um dos arquivos gerados pela aplicação após uma simulação. Para cada algoritmo de roteamento executado no cenário de simulação criado, é gerado um arquivo similar a esse. O arquivo traz o uso de cada roteador, ou seja, quantas vezes ele foi usado durante a simulação, o que nos dá a

informação de quais pontos da NoC estão sendo mais utilizados para aquele algoritmo de roteamento. Claro, isso vai depender de como as aplicações foram alocadas, e o ideal é que não haja uma sobrecarga de um roteador específico ou de uma região da NoC isolada. Essa informação será bastante útil para testar o algoritmo de alocação dinâmica, quando ele for desenvolvido. Logo abaixo, temos as informações sobre cada comunicação que existiu durante a simulação. Se alguma comunicação não obteve êxito em ser completada é especificado em qual roteador o algoritmo de roteamento falhou, e caso ela tenha êxito, é especificado o caminho percorrido pela informação nessa comunicação, ou seja, quais roteadores foram utilizados nessa comunicação. Essa informação é importante para se analisar o comportamento do algoritmo de roteamento e detectar cenários de falhas que ele não é capaz de se adaptar para que a comunicação não falhe.

Outro resultado obtido, este em um módulo secundário, foi a automação na criação de arquivos de entrada da ferramente SHARPE Tool. Esse arquivo representa uma árvore de falhas (*fault tree*), que será utilizada para calcular a confiabilidade da NoC. A árvore de falhas é gerado a partir da informação de quais são os caminhos utilizados nas comunicações, ou seja, quais os roteadores utilizados. Com ela é possível obter a informação de quais roteadores são os mais relevantes, ou seja, roteadores cuja sua falha comprometem mais o funcionamento do sistema, e quais são os menos relevantes, que seriam roteadores cujo a falha não interferem no funcionamento do sistema. Como dito, este é um módulo secundário, por isso não houve muito esforço direcionado a ele nessa etapa do projeto.

Abaixo está uma imagem de um arquivo gerado para uma NoC bem simples:

Figura 3 – Arquivo de entrada da ferramente SHARPE Tool.



Fonte: *printscreen* do arquivo.

4. Discussão

No nosso projeto estamos buscando uma solução para acelerar o processo de exploração de espaço de projeto, avaliando diferentes soluções de alocação de aplicações em diferentes cenários e plataformas, consequentemente, diminuindo o tempo que se levaria para produtos que se beneficiam nesses aspectos serem lançados no mercado. Como o projeto ainda não tem resultados concretos finais, não há como confrontar com resultados de outras literaturas. Em relatórios futuros esses confrontos serão relatados.

5. Conclusão

Nessa etapa do projeto foram feitos avanços nas ferramentas que auxiliarão para que o objetivo final seja alcançado, além do aprofundamento do conhecimento do bolsista na área. Espera-se que com os resultados obtidos até agora seja possível agilizar a segunda etapa do projeto de forma mais rápida e eficaz, uma vez que as informações necessárias para auxiliar nessa segunda etapa estarão a disposição de maneira bastante simplificada.

Referências

[1]Ahmed Jerraya and Wayne Wolf, “Multiprocessor Systems-on-Chips“, 2004.

[2]Luca Benini and Giovanni De Micheli, "Networks on Chips: A New SoC Paradigm", IEEE Computer, January 2002, pp. 70-78.

[3]DEHON, A.; NAEIMI, H. Seven Strategies for Tolerating Highly Defective Fabrication. IEEE Design & Test, Los Alamitos, Jul.-Aug. 2005. 306-315.

[4]http://www.kasahara.elec.waseda.ac.jp/schedule/

[5]http://people.ee.duke.edu/~kst/