Simulador SiNoC

Hiago Mayk Gomes de Araújo Rocha
e-mail: mayk@ppgsc.ufrn.br

August 23, 2017

Diante da dificuldade de encontrar simuladores de redes em chip de alto nível de abstração fáceis de usar, de aprender e que tenham implementadas todas as métricas que se deseja avaliar neste trabalho, foi proposta a implementação de um simulador. O conteúdo aqui descrito foi obtido do trabalho de [1].

O SiNoC, nome vindo da junção entre Simulador e *Network on Chip*, foi implementado na linguagem de programação Java e tem o objetivo de avaliar as aplicações aqui estudadas quanto às métricas que ele implementa. Esse simulador é de código aberto, ou seja, é disponibilizado para que qualquer pessoa possa usar, contribuir para acrescentar funcionalidades ou corrigir eventuais *bugs* que possa apresentar. O código fonte do SiNoC está disponível no GitHub.

A ferramenta SiNoC se restringe, na versão atual, a fazer simulações apenas em MPSoCs com topologia *mesh* 2D de qualquer dimensão, conforme ela foi inicialmente idealizada, porém isso não proíbe de, em trabalhos futuros, poder ser inserida a opção de realizar simulações com topologias genéricas de NoCs.

1 Métricas Calculadas

O simulador possui uma série de métricas que ele calcula e apresenta como resultado. Entre elas, as principais são:

- **Latência da aplicação:** essa métrica contabiliza a latência do pacote que mais demorou para ser entregue à sua tarefa destino, fornecendo, pois, a latência total de comunicação da aplicação. Pode ser obtida em unidades (pacotes) ou em *flits*.
- **Latência em *hops* da aplicação:** essa métrica avalia qual a maior distância, em *hops*, percorrida por um pacote da aplicação.
- **Somatório das latências:** corresponde ao somatório total das latências dos pacotes.
- **Latência média dos pacotes:** é a divisão do somatório total das latências dos pacotes pela quantidade de pacotes transferidos. Ela diz, em média, quanto os pacotes demoram para ser entregues.
- **Quantidade de enlaces acessados:** contabiliza quantos enlaces foram acessados.

- **Reuso dos enlaces:** contabiliza quantos enlaces foram reusados. Ou seja, quantos enlaces foram usados pelo menos duas vezes. Pode ser obtida tanto em unidades (pacotes) ou em *flits*.
- **Taxa de reuso dos enlaces:** contabiliza quantas vezes os enlaces foram reusados. Pode ser obtida tanto em unidades (pacotes) ou em *flits*.
- **Total de *flits* da aplicação:** contabilização do total de *flits* que a aplicação possui.
- **Somatório dos acessos aos enlaces:** contabiliza os acessos a todos os enlaces da NoC. Pode ser obtida em unidades (pacotes) ou em *flits*.

2 Visão da Implementação

Para dar uma visão de como o SiNoC foi estruturado, usando o paradigma de orientação a objetos, apresentamos na Figura 1 um simples diagrama de classes, que descreve as suas classes principais e como elas estão relacionadas. Neste diagrama, observa-se que o projeto é dividido em pacotes. O pacote de Entidades, contém as classes que são usadas para representar entidades do simulador, como Roteador, Pacote, Enlace, entre outras. Elas armazenam os resultados de simulação para cada um desses componentes. No pacote Roteamentos, temos todas as classes de roteamento implementadas, entre elas, temos a classe Roteamento, que é a que possui os atributos e métodos que todos os roteamentos que forem implementados nesse simulador devem ter acesso. Com isso, se for implementado um novo roteamento, este deve herdar da classe Roteamento. No pacote Mapeamentos, temos as implementações dos mapeamentos contidos no simulador. No pacote EntradaSaida, temos a classe de Entrada de dados, a qual implementa entrada por arquivo ou manual. O pacote Principal contém apenas a classe Calcula a qual é responsável por instanciar as classes necessárias para a execução do simulador. Nesta classe temos também uma simples interface de linha de comando, que é usada para tornar mais fácil e agradável o uso da ferramenta.

O simulador, como visto no diagrama de classes da Figura 1, já implementa duas estratégias de roteamento, a XY e a XY_YX, e também três estratégias mapeamento, o $\mathcal{V}1$_decrescente, $\mathcal{V}1$_crescente e o mapeamento Sequencial, as quais serão explicadas em detalhe no Capítulo 5, deste trabalho. Apesar disso, por ser uma ferramenta *open source*, ela pode ser facilmente estendida para implementar muitas outras estratégias de roteamento e mapeamento.

3 Configurações Padrão

Por simplicidade, no SiNoC, a largura de banda dos canais e os *buffers* dos roteadores são infinitos. O simulador já tem os mecanismos de comunicação préfixados, não sendo possível a alteração desses. As configurações dos mecanismos de comunicação são os seguintes:

- **Chaveamento:** é feito por pacote, da forma *store-and-forward*, logo, os pacotes são subdivididos em *flits*. Quando um grafo de uma aplicação é dado como entrada para a ferramenta, cada comunicação entre suas tarefas

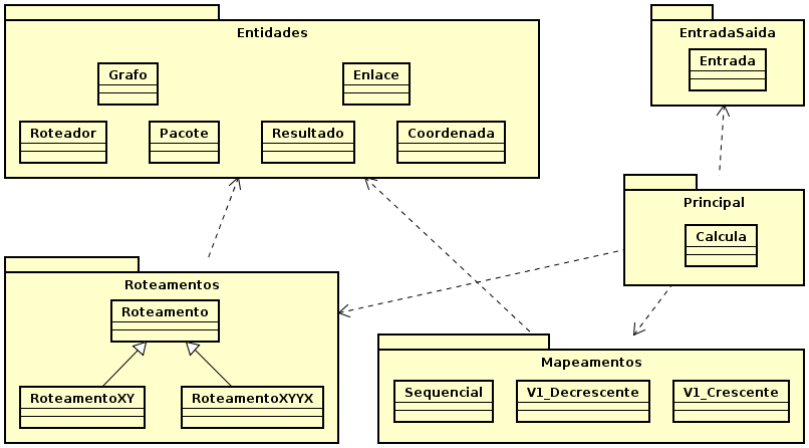

Figure 1: Representação do diagrama de classes da ferramenta SiNoC.

é considerada como um pacote, e os *flits* são os pesos relacionados a cada uma dessas comunicações.

- **Controle de fluxo:** como os *buffers* dos roteadores são infinitos, não é implementado nenhum controle de fluxo específico. Entretanto, pode-se dizer que temos um controle de fluxo baseado em créditos onde a quantidade de créditos é infinita.
- **Roteamento:** como já visto, o simulador implementa dois tipos de roteamento, sendo um não adaptativo, XY, e o outro adaptativo, XY_YX sendo subclassificado de semi adaptativo, por evitar que os caminhos percorridos pelos pacotes se distancie de sua tarefa de destino.
- **Arbitragem:** é feita de forma distribuída, onde roteamento e a arbitragem são realizados separadamente. Assim, o algoritmo de roteamento apenas diz ao roteador qual direção o pacote deve seguir, e o árbitro de cada roteador é quem vai dizer qual melhor caminho imediato a seguir para que o pacote percorra para aquela determinada direção.
- **Memorização:** é feita de forma centralizada em cada roteador, onde os pacotes que chegam de qualquer direção são armazenados em um único *buffer*, o qual usa a política *First-In First-Out* (FIFO). Entretanto, caso em um dado instante da simulação, chegue mais de um pacote em um determinado roteador, eles são armazenados no *buffer* de forma crescente de acordo com seus tamanhos, em *flits*. Caso o tamanho seja igual, a alocação é feita de forma arbitrária.

4 Exemplo de Uso

Para exemplificar o uso da ferramenta, e explicar como é usada na versão atual, iremos apresentar como realizar um teste para uma das aplicações aqui avaliadas. A aplicação escolhida foi a Pipe, a qual será detalhadamente apresentada e descrita no capítulo que trata dos resultados experimentais, Capítulo 6.

Primeiramente, ao rodar o executável do simulador ou importar o projeto em alguma plataforma de desenvolvimento da linguagem Java, irá ser apresentado o menu de seleção do mapeamento que se deseja realizar, conforme apresentado na Figura 2. Nessa Figura, pode-se selecionar (1) a entrada do mapeamento por console, ou seja, onde o usuário digita no terminal qual núcleo receberá qual tarefa, (2) o mapeamento $\mathcal{V}1$_decrescente, (3) $\mathcal{V}1$_crescente, (4) Sequencial e (5) para a entrada do mapeamento por arquivo. A opção 5 foi implementada para que pudessem ser aplicadas no simulador as soluções geradas pelo algoritmo BLS. Essa opção pega uma lista de mapeamentos e aplica nas estruturas internas do simulador. A opção 0, por fim, encerra a execução do simulador.

A Figura 2 ilustra a seleção da opção 3 (mapeamento $\mathcal{V}1$_crescente), indicado pelo número 3, em verde, na linha mais abaixo da Figura.

```
=================    SiNoC    =================
1               Mapeamento por console;
2               Mapeamento V1_Decrecente;
3               Mapeamento V1_Crescente;
4               Mapeamento Sequencial;
5               Mapeamento por arquivo;
0               Sair.
>>> 3
```

Figure 2: Representação do menu de mapeamento da ferramenta SiNoC.

Na Figura 3, temos a entrada do grafo da aplicação, a qual é dada pela seguinte sequência: quantidade de tarefas, quantidades de comunicações e, nas linhas subsequentes, as comunicações que são feitas, seguindo a ordem de tarefa de origem, tarefa de destino e quantidade de *flits* transferidos na comunicação.

Nesta figura vemos a inserção (números em verde), do grafo da aplicação Pipe na ferramenta SiNoC. Observe a ordem de inserção dos parâmetros, onde, primeiramente, é inserida a quantidade de tarefas da aplicação, que neste caso são 10. Logo após, é inserida a quantidade de comunicações feitas entre essas tarefas, que neste caso são 9. As linhas seguintes, representam as comunicações que são feitas, como já dito, onde a tarefa 0 se comunica com a tarefa 1 com uma quantidade de 2000 *flits*, a tarefa 1 se comunica com a tarefa 2 com uma quantidade de 2000 *flits*, e assim vai, até a última linha, onde temos que a tarefa 8 se comunica com a tarefa 9 também com uma quantidade de 2000.

```
Digite o grafo de entrada:

10
9
0 1 2000
1 2 2000
2 3 2000
3 4 2000
4 5 2000
5 6 2000
6 7 2000
7 8 2000
8 9 2000
```

Figure 3: Representação da inserção do grafo da aplicação no simulador SiNoC.

Na Figura 4, temos a inserção dos parâmetros de dimensão da NoC que

será simulada, sendo o primeiro valor referente às linhas e o segundo às colunas das matrizes que representam os núcleos do MPSoC e os roteadores da NoC que serão criadas internamente no simulador. Nesta figura, vemos a inserção (números em verde) de dois valores 4, referentes as linhas e colunas, ou seja, a simulação será feita em cima de um MPSoC baseado em NoC com topologia *mesh* 2D com dimensão 4x4.

```
Digite a quantidade de linhas e colunas da rede:
4
4
```

Figure 4: Representação da inserção do tamanho da NoC no simulador SiNoC.

Na Figura 5, temos a representação do menu da escolha de roteamentos, no qual temos dois tipos de simulações de roteamento. Um deles, denominado de $Full$, assume que as comunicações são feitas de forma sequencial, ou seja, um pacote é transmitido por vez na rede, e cada pacote só sai de seu destino para sua origem se, e somente se, não houver outros pacotes trafegando na rede. Já o outro, o qual é chamado de $ByStep$, assume paralelismo total nas comunicações das tarefas e são os dados gerados por essa simulação que esse trabalho avalia.

Nessa Figura são apresentadas as possíveis opções de simulação sendo (1) o roteamento XY não paralelo, (2) XY paralelo, (3) XY_YX não paralelo, (4) XY_YX paralelo, (5) a execução dos dois roteamentos XY e XY_YX paralelos e por último (6) a execução de todos os roteamentos implementados pela ferramenta. Na Figura 5 vemos a seleção da opção 6, para a execução de todos os roteamentos nas duas formas de simulação.

```
========================================
1               Rotemento XY Full
2               Rotemento XY By Step
3               Rotemento XY_YX Full
4               Rotemento XY_YX By Step
5               Rotemento XY e XY_YX By Step
6               Todos
>>>6
```

Figure 5: Representação do menu de roteamentos da ferramenta SiNoC.

Na Figura 6, ilustramos como são apresentados os resultados obtidos pela ferramenta, os quais são divididos por modos de simulação, seguidos pelas métricas calculadas em cada tipo de simulação. Os resultados apresentados nessa figura mostram apenas alguns dos dados que se pode obter dessa ferramenta. Outros dados podem ser obtidos, dependendo de quais aspectos se deseja avaliar na simulação. É possível obter métricas para cada enlace, pacote, roteador, ou outro componente específico, ou até mesmo avaliar a configuração dos mapeamentos gerados por cada algoritmo aplicado.

References

[1] Hiago Mayk Gomes de Araújo Rocha. O problema do mapeamento: Heurísticas de mapeamento de tarefas em mpsocs baseados em noc. B.S. thesis, Universidade Federal do Rio Grande do Norte, 2017.

```
ALGORITMO XY FULL:
Latencia: 11
Enlaces acessados: 10
Somatorio de acessos aos enlaces: 11
Enlaces reusados: 1
Acessos aos enlaces reusados: 1
Taxa de reuso dos enlaces: 9.0%

ALGORITMO XY BY STEP:
8      -      9      ->     2000    Hops: 2      Latencia: 2
1      -      2      ->     2000    Hops: 1      Latencia: 1
7      -      8      ->     2000    Hops: 1      Latencia: 1
2      -      3      ->     2000    Hops: 1      Latencia: 1
0      -      1      ->     2000    Hops: 2      Latencia: 2
6      -      7      ->     2000    Hops: 1      Latencia: 1
3      -      4      ->     2000    Hops: 1      Latencia: 1
4      -      5      ->     2000    Hops: 1      Latencia: 1
5      -      6      ->     2000    Hops: 1      Latencia: 1
(Pacote maior latencia) Lantencia: 2
(Pacote maior latencia) Hops: 2
(Pacote maior hop) Lantencia: 2
(Pacote maior hop) Hops: 2
Somatório das latencias: 11
Latencia Média: 1
Enlaces acessados: 10
Somatorio de acessos aos enlaces: 11
Enlaces reusados: 1
Reuso dos enlaces: 1
Taxa de reuso dos enlaces: 9.0%
Total Flits:18000
Somatorio de acessos aos enlaces (em flits): 22000
Reuso dos enlaces (em flits): 4000
Taxa de reuso dos enlaces (em flits): 18.0%

ALGORITMO XY_YX FULL:
Latencia: 11
Enlaces acessados: 10
Somatorio de acessos aos enlaces: 11
Enlaces reusados: 1
Acessos aos enlaces reusados: 1
Taxa de reuso dos enlaces: 9.0%

ALGORITMO XY_YX BY STEP:
8      -      9      ->     2000    Hops: 2      Latencia: 2
1      -      2      ->     2000    Hops: 1      Latencia: 1
7      -      8      ->     2000    Hops: 1      Latencia: 1
2      -      3      ->     2000    Hops: 1      Latencia: 1
0      -      1      ->     2000    Hops: 2      Latencia: 2
6      -      7      ->     2000    Hops: 1      Latencia: 1
3      -      4      ->     2000    Hops: 1      Latencia: 1
4      -      5      ->     2000    Hops: 1      Latencia: 1
5      -      6      ->     2000    Hops: 1      Latencia: 1
(Pacote maior latencia) Lantencia: 2
(Pacote maior latencia) Hops: 2
(Pacote maior hop) Lantencia: 2
(Pacote maior hop) Hops: 2
Somatório das latencias: 11
Latencia Média: 1
Enlaces acessados: 10
Somatorio de acessos aos enlaces: 11
Enlaces reusados: 1
Reuso dos enlaces: 1
Taxa de reuso dos enlaces: 9.0%
Total Flits:18000
Somatorio de acessos aos enlaces (em flits): 22000
Reuso dos enlaces (em flits): 4000
Taxa de reuso dos enlaces (em flits): 18.0%
```

Figure 6: Representação dos resultados obtidos pela ferramenta SiNoC.