Exploração Eficiente em Espaços de Projeto de Comunicação em Plataformas Multiprocessadoras Baseadas em Barramentos

Guilherme Esmeraldo[1], Edna Barros[2]

[1]Instituto Federal de Educação, Ciência e Tecnologia do Ceará (IFCE) – *campus* Crato
Crato, CE – Brasil

[2]Centro de Informática
Universidade Federal de Pernambuco (UFPE) – Recife, PE – Brasil

guilhermealvaro@ifce.edu.br, ensb@cin.ufpe.br

Abstract. *Embedded systems have became very complex due to the new applications requirements, thereby needing more than one processor, different types of memory and peripherals. To make more efficient the communication among these devices, the designer must customize the on-chip communication structure. When using buses for the communication, many parameters must be considered, since they can directly impact on the performance of the entire system. This paper presents a new approach that, compared to traditional simulation approaches, aims to increase the performance of the exploration task of the bus configuration options included in the design space.*

Resumo. *Sistemas embarcados se tornaram muito complexos, devido aos novos requisitos das aplicações, necessitando assim de mais de um processador, diferentes tipos de memória e dispositivos variados. Para tornar a comunicação mais eficiente entre esses dispositivos, deve-se customizar sua estrutura de comunicação. Quando se utiliza barramentos para a comunicação, muitos parâmetros têm que ser considerados, pois podem impactar diretamente no desempenho de todo o sistema. Este trabalho apresenta uma nova abordagem que permite explorar com grande desempenho, em relação às abordagens tradicionais de simulação, as opções de configuração de barramento de todo o espaço de projeto.*

1. Introdução

Devido ao desenvolvimento das tecnologias de circuitos integrados (CIs), que hoje permitem integrar mais de um bilhão de transistores em um único chip [BOUAJILA et al. 2011], aos requisitos mais rígidos das aplicações e ciclos cada vez menores para lançamento de novos produtos, os sistemas embarcados, bem como seu projeto, passaram a ficar mais complexos. Então surgiram os *Systems-on-Chip* (*SoCs*). Um *SoC* é um CI que implementa muitas ou todas as funções de um sistema eletrônico completo [JERRAYA and WOLF 2004], podendo incluir memórias, microprocessador, interfaces com periféricos, lógica de controle de E/S, conversores de dados e outros componentes que abrangem sistemas computacionais completos [WILD, HEKERDOF and

OHLENDORF 2006]. Com a redução do custo e do tamanho dos microprocessadores embarcados, dez ou mais microprocessadores podem ser integrados em um único chip para formar um *Multi-Processor System-on-Chip* (*MPSoC*). Na prática, um *MPSoC* é um *SoC* que inclui vários processadores conectados por uma arquitetura de comunicação [JOO, KIM and HA 2009].

Arquitetura de comunicação *on-chip* refere-se a uma estrutura física que integra os componentes de processamento e armazenamento de *SoCs* e disponibiliza um mecanismo para troca de dados e informações de controle entre eles [JERRAYA and WOLF 2004]. Arquiteturas baseadas em barramentos são atualmente bastante populares para implementação da estrutura de comunicação *on-chip* em projeto de *MPSoCs*, porque são simples de projetar e eficientes de implementar, com consequente redução do tempo de projeto, além de ocupar pequenas áreas no chip e ser extensível [SHANTI and AMUTHA 2012].

Porém, por ser um recurso compartilhado, o barramento torna-se o fator limitante no desempenho da comunicação de um sistema embarcado [LAHIRI and RAGHUNATHAN 2004]. Além disso, selecionar e reconfigurar arquiteturas de comunicação baseadas em barramentos para atender a todos os requisitos de comunicação da aplicação, é um processo que consome muito tempo [PASRICHA et al. 2004]. Isto deve-se ao grande conjunto de opções de configuração ao se considerar: diferentes topologias de barramentos customizáveis, protocolos de comunicação, tipos de transferências, larguras de barramentos de dados e endereço, frequências de operação, tamanho dos *buffers* e esquemas de arbitragem. Especialistas podem, através de experiência e intuição, acelerar o processo de escolha de uma configuração, porém, em sistemas com dezenas de parâmetros de configuração, essa tarefa pode se tornar muito difícil e, então, os especialistas podem ser levados a cometer erros e escolher uma configuração ótima local e não global [SHIN et al. 2004].

No projeto de *MPSoCs* há uma clara tendência em se utilizar a abordagem de Projeto Baseado em Plataformas (PBP) [ROWSON and SANGIOVANNI-VINCENTELLI 1997]. Nesta abordagem, o sistema a ser desenvolvido é, inicialmente, especificado através de uma descrição em alto nível, como descrições textuais e/ou diagramas. As funções do sistema, contidas nessa especificação, são selecionadas para serem implementadas em *software* (que será mapeado para microprocessadores) ou em *hardware* (componentes de *hardware* de finalidade específica, implementados a partir de reuso de outros componentes, ou compilados a partir de ferramentas de síntese) [LAHIRI, RAGHUNATHAN and DEY 2001]. Estes componentes de *hardware* fazem parte de uma arquitetura predefinida, conhecida como plataforma, que pode ser modificada para que sua estrutura possa ser adaptada às necessidades da aplicação.

Desde que os componentes da plataforma irão compartilhar dados e se comunicar para implementar as funcionalidades do sistema, deve-se refinar os parâmetros de configuração da arquitetura de comunicação para atender às necessidades de comunicação dos componentes. Pesquisadores têm trabalhado no desenvolvimento de técnicas de análise considerando diferentes métricas, tais como desempenho, potência, custo do sistema, etc., para guiar os passos de particionamento/mapeamento. Nestes trabalhos, técnicas para particionamento automático podem: 1) ignorar completamente comunicação entre os componentes ou 2) utilizar modelos simples de

comunicação para guiar o passo de particionamento/mapeamento [LAHIRI, RAGHUNATHAN and DEY 2001].

Este trabalho apresenta uma nova abordagem para dar suporte ao projeto da estrutura de comunicação em plataformas multiprocessadoras. A abordagem inclui uma técnica de análise híbrida capaz de estimar o desempenho de comunicação de uma plataforma para todas as opções de configuração de barramento contidas no espaço de projeto. A técnica proposta suporta a análise da comunicação nos processos de seleção e refinamento das arquiteturas de comunicação após a aplicação ter sido particionada e mapeada para os componentes da plataforma.

O restante do artigo está dividido da seguinte maneira: Na seção a seguir, são apresentados os trabalhos relacionados. A Seção 3 apresenta a abordagem proposta. Na Seção 4, é apresentado o estudo de caso bem como resultados experimentais e, por fim, na Seção 5, são demarcadas as conclusões.

2. Trabalhos relacionados

Dado o aumento da complexidade dos sistemas, o número de parâmetros das arquiteturas de comunicação das plataformas tem aumentado consideravelmente, levando à necessidade de examinar milhares de configurações diferentes para se obter a melhor arquitetura [LAHIRI, RAGHUNATHAN and DEY 2001]. Dependendo do tamanho do espaço de projeto, esse processo pode consumir muito tempo, dias ou meses, até se encontrar a melhor arquitetura ou apenas uma que atenda a todas as restrições de projeto.

Para acelerar a busca no espaço de projeto, muitos trabalhos têm focado no desenvolvimento de técnicas para suporte à análise de comunicação, disponibilizando estimativas do impacto da arquitetura de comunicação no desempenho e consumo de potência total do sistema [JERRAYA and WOLF 2004]. Neste trabalho, estas técnicas foram divididas, de um modo geral, em três categorias: (1) técnicas baseadas em simulação, (2) abordagens baseadas em estimação estática e (3) técnicas híbridas. São técnicas bastante distintas mas, em muitos casos, são complementares para avaliação de mecanismos de comunicação.

O primeiro grupo inclui as abordagens baseadas na simulação completa de sistema. Estas abordagens incluem modelos de simulação que podem incluir componentes de hardware e sua comunicação em diferentes níveis de abstração [PASRICHA, DUTT and BEN-ROMDHALE 2005][ROWSON and SANGIOVANNI-VINCENTELLI 1997]. O uso de abstrações de comunicação permite estabelecer um compromisso entre precisão e desempenho das estimativas. Entretanto, essas técnicas requerem uma simulação completa do sistema para avaliar cada ponto do espaço de projeto, implicando em alto custo computacional.

Outros tipos de técnicas buscam, através de modelos matemático/estatísticos [KIM, IM and HA 2003][CHO, CHOI and CHO 2006], estimar estaticamente as métricas do sistema, como, por exemplo, desempenho ou potência. O uso deste tipo de técnica só é possível em sistemas onde as computações e comunicações podem ser previamente agendadas [LAHIRI, RAGHUNATHAN and DEY 2001]. Além disso, podem ser demasiadamente otimistas, por não considerarem os efeitos dinâmicos da

aplicação, como latências por contenção de barramento, ou pessimistas, assumindo o pior caso para contenção [GASTEIER and GLESNER 1999]. Por não incluir detalhes suficientes de comunicação, a utilização deste tipo de técnica pode resultar em estimativas imprecisas.

O último grupo inclui as abordagens híbridas [LAHIRI, RAGHUNATAN and DEY 2001][SHIN et al. 2004][WILD, HEKERDOF and OHLENDORF, 2006][JOO, KIM and HA 2009][LUKASIEWYCZ et al. 2009], que agregam diversos tipos de técnicas, como por exemplo, heurísticas ou estimativas estáticas, às técnicas de simulação completa de sistema. Tais abordagens buscam aumento de desempenho das simulações e/ou exploração, enquanto procuram disponibilizar estimativas precisas da comunicação. Porém, na prática, algumas dessas abordagens não permitem explorar espaços de projeto maiores para refinamento de comunicação.

Este artigo apresenta uma abordagem híbrida que utiliza simulação completa de sistema para obter um perfil de comunicação da aplicação e, a partir dele, estimar o desempenho de comunicação para cada configuração de barramento. Com o uso da técnica proposta, a tarefa de exploração do espaço de projeto de comunicação se torna mais eficiente, pela redução do número de simulações necessárias para exploração do espaço de projeto, e é eficaz, por fazer uso de estimativas precisas de desempenho da comunicação da aplicação. Adicionalmente, a técnica proposta dispõe de suporte ao detalhamento das transações de barramento para que o projetista possa analisar todo o processo de comunicação e identificar os gargalos.

3. Abordagem Proposta

O fluxo da abordagem proposta para o projeto da estrutura de comunicação de uma plataforma virtual inclui cinco fases, as quais são ilustradas na Figura 1.

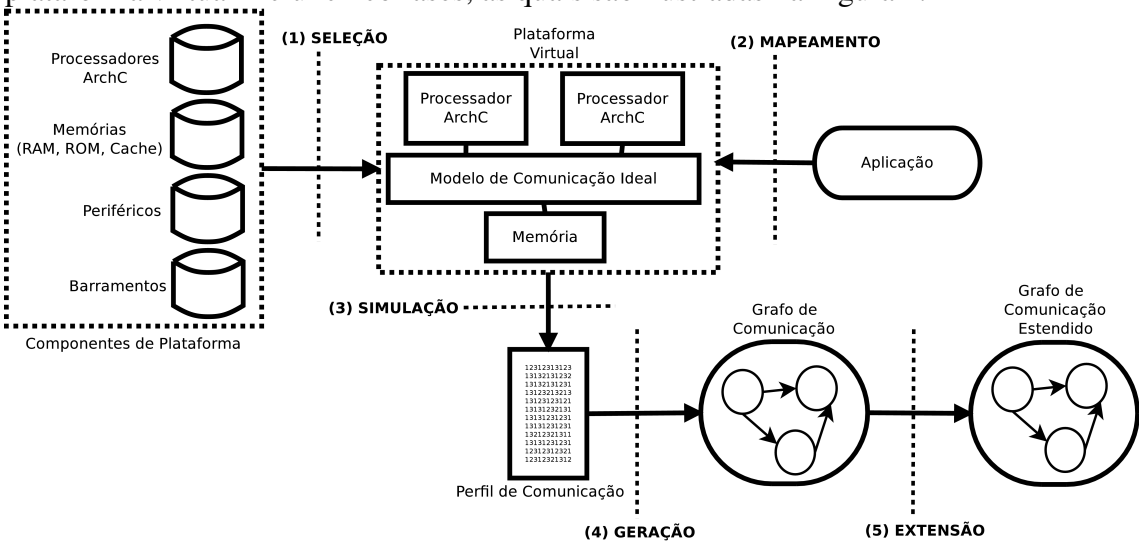

Figura 1. Fluxo proposto para o projeto da estrutura de comunicação.

A primeira e a segunda fase consistem da *(1) seleção* dos componentes de hardware que irão compor a plataforma virtual e do *(2) mapeamento* da aplicação, respectivamente. O modelo de plataforma adotado neste trabalho inclui processadores

descritos em *ArchC* [ARCHC TEAM 2007] e memórias RAM, descritas em *SystemC TLM* [PASRICHA, DUTT and BEN-ROMDHALE 2005], com capacidade de armazenamento, tamanho de palavra e latências parametrizáveis.

Para modelar a comunicação entre esses componentes, foi proposto um modelo de barramento genérico, também descrito em *SystemC*. Este modelo de barramento permite a simulação considerando um modelo de estrutura de comunicação ideal, o qual não introduz latências nos processos de comunicação, que permite transferências em paralelo, desde que os periféricos forneçam suporte para este tipo de acesso, e com diferentes tamanhos de dados. Adicionalmente, este modelo registra as informações de todas as transferências, possibilitando a extração de um perfil da comunicação da aplicação.

A terceira fase, consiste da *(3) simulação* da plataforma virtual com o modelo de barramento genérico, através da execução do programa gerado a partir do código *SystemC*. Nesta fase, é gerado um arquivo em disco, o perfil de comunicação, com o registro de todos os eventos de comunicação realizados no modelo de barramento genérico proposto. Formalmente, o perfil de comunicação é dado por:

> Seja P um perfil de comunicação, que será dado por $P=\{r_k\}$, para $k=1,2,...,n$, onde r_k representa cada um dos registros de transações de barramento contidas no perfil, que será dada pela 3-upla $r_k=(m, e, t)$, onde: m será o identificador do mestre do barramento que originou a transação; e será o evento de transação, que pode ser de *(q)* requisição de uso do barramento, *(c)* envio de sinais de controle, *(d)* envio de dados e *(l)* liberação do barramento; t será número de sequência da transação.

O perfil de comunicação inclui registros de todos os eventos de comunicação entre os componentes da plataforma, porém sem incluir as características específicas do barramento, tais como tamanho de palavra ou latências de transferências e contenção.

Assim, é proposto que esse perfil seja modificado para incluir características de barramentos reais, e que seja utilizado para estimar o desempenho da aplicação na plataforma com barramentos reais. A modificação da estrutura que representa o perfil de comunicação é obtida através de: 1) alteração da sequência de eventos das transações, impondo assim um modelo de prioridades de acesso (escalonamento de transferências por arbitragem) e 2) adição de características específicas de modelos reais de barramentos, tais como largura de barramentos de dados, latências inerentes aos protocolos de transferências e frequência de operação.

O modelo adotado para representar em memória os perfis de comunicação, e com isso otimizar o acesso aos dados, é uma extensão do modelo proposto em [LAHIRI, RAGHUNATHAN and DEY 2001], que utilizou grafos - chamado de Grafos de Comunicação (GC). Neles os vértices são utilizados para representar eventos de computação e comunicação. Tais eventos de comunicação consideram apenas transações atômicas de barramento, indivisíveis, descaracterizando cenários reais de comunicação embarcada, como por exemplo, interrupção de uma transferência por preempção de arbitragem. Assim, a primeira contribuição do trabalho aqui apresentado é a extensão desse modelo de GC, para suportar a modelagem das fases internas de uma transação (requisição de uso do barramento, envio de sinais de controle, transferência de dados e liberação) no perfil de comunicação. Com isso, busca-se tornar as estimativas mais precisas. Formalmente, um GC, neste trabalho, é dado por:

Seja G um grafo de comunicação, dado por $G=(V,E)$, onde V é o conjunto de vértices, que representam as transações, e E o conjunto de arcos, os quais representam a sequência entre as transações em V. Seja a transação v, tal que $v \in V$, v será dada por $v=(m, e, t)$, onde m é o identificador do mestre que iniciou a transação, e é uma 4-upla (q, c, d, l), onde q, c, d, e l representam as quantidades de eventos de cada fase da transação (requisição de uso, sinais de controle, envio de dados e liberação do barramento) e t o número de ordem na sequência das transações. Sejam os vértices $u=(m_u, e_u, t_u)$ e $v=(m_v, e_v, t_v)$ transações do barramento, então o arco $(u, v) \in E$, estabelece ordem de sequência entre as transações u e v, de forma que após a transferência u haverá a transferência v, sendo que $m_u=m_v$ (pertencerão ao mesmo mestre).

A fase seguinte inclui a *(5) extensão* de um GC para embutir características de barramentos reais e com isso obter uma estimativa de desempenho da comunicação sob aquelas configurações. As características do barramento, que terão seus efeitos adicionados ao grafo de comunicação, são definidas no *Perfil de Barramento*. O perfil de barramento é uma descrição dos parâmetros de um modelo incluindo possíveis larguras do barramento de dados, frequências, suporte a transferências com *pipeline*, latências das transferências (quantidade de ciclos necessários para completar as fases de uma transferência), tipos de transferência (simples, rajada, com e sem preempção), além do mecanismo de arbitragem (e prioridades de mestres, quando for o caso). Caso o espaço de projeto inclua diferentes modelos de barramento, podem ser utilizados vários perfis de barramento. Em termos gerais, o processo de extensão de um grafo de comunicação envolve quatro etapas:

1. Fixação de largura do barramento de dados: para incorporar uma largura fixa do barramento de dados, o GC deve ser modificado para suportar a largura de barramento definida no perfil do barramento. Para um mestre com tamanho de palavra maior que a largura do barramento de dados, em seu respectivo ramo no grafo, serão adicionados novos vértices para cada vértice já existente, de forma a simular a divisão de uma transferência em transferências com menor tamanho de palavra. Assim, para um mestre com tamanho de palavra de 64 *bits* e um barramento de dados de 8 *bits*, para cada vértice em seu ramo do grafo, seriam criados sete novos vértices, totalizando assim oito transferências de 8 *bits*. Mestres com tamanho de palavra inferior à largura do barramento continuarão com um vértice apenas para representar a transferência de uma palavra, implicando assim em não alteração no grafo. Para esse mestre seria possível ainda agrupar vários vértices (várias transferências) em apenas um único vértice, simulando assim uma transferência com múltiplas palavras. Assim, teríamos, por exemplo, para um mestre com palavras de 8 *bits*, um vértice que representa uma comunicação de um barramento de 64 *bits* poderia agrupar até oito vértices de comunicação deste mestre simulando, assim, oito transferências de 8 *bits*.

2. Inclusão de política de arbitragem: quando há mais de um mestre no barramento, um grafo de comunicação terá um ramo para cada mestre, representando as respectivas sequências de transferências. Mencionou-se anteriormente que a disposição dos ramos pode representar transferências em paralelo e, como o barramento é um recurso compartilhado, deve-se considerar o uso de um mecanismo de arbitragem para escalonar o uso do barramento entre os mestres. Para tanto é necessário definir inicialmente a política de arbitragem a ser adotada e distribuir as prioridades de uso do barramento a cada um dos mestres. Em seguida, o grafo deverá convergir seus ramos de forma que, ao fim, haja apenas um único ramo. O ramo final conterá todos os vértices

dispostos em ordem seguindo a política de arbitragem e prioridades definidas anteriormente. Consequentemente, o ramo final incluirá, em sua sequência de transferências, as contenções provenientes da arbitragem, assim como ocorre em um modelo de barramento real. Formalmente, o processo de convergência dos ramos de um grafo de comunicação é descrito a seguir:

> Dado $G=(V,E)$, um grafo de comunicação de uma dada plataforma T, e M seja o conjunto de mestres de barramento da plataforma T. M será dado por $M=\{M_k\}$, para $k=1,2,..,n$, e M_k representa o conjunto de transferências de um mestre k, tal que $M_k = \{v_{k1}, v_{k2}, ..., v_{km}\}$ e $v_{kt} \in V$, para $t=1, 2, .., m$ e t representa a ordem de execução das transações de cada mestre. O processo de convergência será dado por:
>
> i) Cria-se um novo grafo de comunicação $G'=(V', E')$, com os conjuntos V' e E' vazios.
>
> ii) Define-se um conjunto $P=(p_1, p_2, .., p_n)$, onde $p_1, p_2, .., p_n$ representam as prioridades de uso do barramento dos mestres $M_1, M_2, .., M_n$, respectivamente.
>
> iii) Cria-se um conjunto M' que conterá os conjuntos $M_1, M_2, ..., M_n$, dispostos ordenadamente de acordo com as prioridades definidas por P.
>
> iv) Remove-se o próximo vértice $v_{jz} \in M_j$, onde $M_j \in M'$ e M_j é o conjunto de vértices do mestre de maior prioridade, e adiciona-se v_{jz} a V'. Caso não existam vértices em M_j, este conjunto será removido de M' e M_j passará a ser o próximo elemento de M'.
>
> v) Em seguida, remove-se o próximo vértice $v_{iu} \in M_i$, onde a prioridade p_i de M_i será a segunda maior entre as prioridades de P, e adiciona-se v_{iu} a V', desde a aresta $(v_{jz}, v_{iu}) \in E'$. Caso não existam mais vértices em M_i, este conjunto será removido de M' e o próximo elemento de M' passará a ser o M_i.
>
> vi) Repetem-se os passos (iv) e (v) até que não restem conjuntos a serem removidos de M'.

Outro aspecto importante considerado neste trabalho, relacionado à arbitragem, é o suporte a transferências com bloqueio (sem preempção) e transferências sem bloqueio (com preempção). Quando um mestre utiliza transferência sem bloqueio, ele pode ser desalocado do barramento, durante uma transferência, para que um mestre de maior prioridade possa utilizar o barramento. Ao receber novamente a concessão de uso do barramento, o mestre deverá reiniciar a transferência interrompida.

3. Agrupamento de transferências: ao convergir os ramos de um grafo de comunicação, os vértices referentes às transferências dos mestres do barramento farão parte de um único ramo, o qual representará toda a sequência de transferências no barramento. Neste ramo final é possível que haja sequências de vértices representando transferências seguidas de um mesmo mestre. De acordo com [LAHIRI, RAGHUNATHAN and DEY 2001], uma das formas de aumentar o desempenho do processo de estimação, utilizando GC se dá através do agrupamento dos vértices, reduzindo-se assim o tamanho do grafo. Neste trabalho, o processo de agrupamento será aplicado em conjuntos de vértices que seguem determinados padrões, dados por: 1) um conjunto de transferências realizadas sequencialmente e 2) transferências pertencentes a um mesmo mestre. Se a transferência for do tipo rajada, o agrupamento acontece em duas fases. Na primeira, será dado o agrupamento das transferências que compõem uma rajada. Por exemplo, uma transferência de rajada composta por quatro transferências simples, terá agrupamento dos quatro vértices em um único. Este último representará uma rajada completa. Na segunda fase, realiza-se o agrupamento dos vértices que representam sequências de rajadas completas. Os agrupamentos de transferências

simples e a primeira etapa do agrupamento de rajadas são bastante semelhantes e podem ser dadas da seguinte forma:

> Dado um grafo de comunicação $G=(V,E)$ e $v_1, v_2, v_3, .., v_n$ as transações tais que $v_1\ v_2\ ,..., v_n \in V$, $(v_1, v_2), (v_2\ , v_3), (v_3, ...), (..., v_n) \in E$ e $v_k=(m, e, t)$, para $k=1, 2, 3, ..., n$, onde n é o tamanho de uma sequência de vértices analisada, m representa o mestre que iniciou a transferência, e define o conjunto de eventos da transferência e t o número de ordem na sequência de transações, o processo de agrupamento será dado por:
>
> i) Inicialmente, deve-se verificar se os valores de m em v_k, para $k=1,2,3,...,n$, são iguais, ou seja, se as transações da sequência analisada pertencem ao mesmo mestre.
>
> ii) Caso pertençam ao mesmo mestre, deve-se verificar, em seguida, se a sequência de eventos de comunicação dado por e dos vértices v_k, para $k=1,2,3,...,n$, contêm as mesmas quantidades de eventos q, c, d, e l, ou seja, se as transações são do mesmo tipo.
>
> iii) Caso o tipo de transferência configurada para o mestre seja do tipo simples, será adicionado um novo campo T ao vértice v_1 . O novo campo T receberá valor n, que é a quantidade de vértices da sequência analisada e que contabilizará a quantidade de transferências que o vértice v_1 representará. Os demais vértices $v_2\ , v_3\ , ...,\ v_n$ serão descartados.
>
> iv) Caso o tipo de transferência configurada para o mestre seja do tipo rajada, na primeira etapa do agrupamento será adicionado um novo campo T ao vértice v_1. Esse novo campo receberá valor referente ao tamanho da rajada, que é predeterminado no perfil do barramento. Os campos d e c, de e, em v_1, serão também modificados para incluir a quantidade de eventos de endereçamento e envio de dados para as transferências internas da rajada. Os demais vértices da sequência, $v_2\ , v_3\ , ...,\ v_n$, serão descartados.

Já a segunda etapa do processo de agrupamento para transferências em rajada pode ser formalizado de acordo com a definição a seguir:

> Dado $G'=(V',E')$ um grafo de comunicação e $v_1', v_2', v_3' .., v_n'$ transações do tipo rajada tais que $v_1'\ , v_2'\ ,..., v_n' \in V'\ , (\ v_1'\ , v_2'\ ), (\ v_2'\ v_3'\ ), (\ v_3'\ ,...), (..., v_n'\ ) \in E'$ e $v'_k =(m, e, t, T)$, para $k=1, 2, 3, ..., n$, onde n é tamanho de uma sequência de vértices analisada, m representa o mestre que iniciou a transferência, e consiste no conjunto de eventos da transferência, t representa o número de ordem na sequência de transações e T define o tamanho da rajada, a segunda etapa do processo de agrupamento será dada por:
>
> i) Verifica-se se os valores dos campos m dos vértices v_k', para $k=1,2,3,...,n$, são iguais, ou seja, se as transações da sequência analisada pertencem ao mesmo mestre.
>
> ii) Caso pertençam ao mesmo mestre, deve-se verificar, em seguida, se a sequência de eventos de comunicação contida no evento e dos vértices v_k , para $k=1,2,3,...,n$, contém as mesmas quantidades de eventos q, c, d e l, ou seja, se as transações são do mesmo tipo, e se são do tipo rajada, ou seja, se os campos $c>1$ e $d>1$, de e, bem como campo $T>1$.
>
> iii) Caso a sequência de transferências seja do tipo rajada, será atribuído ao campo T do vértice v_1' valor referente ao número de transferências por rajada que irão compor o agrupamento. Os demais vértices $v_2', v_3', ..., v_n'$ serão descartados.

4. Estimação de latências: o cálculo das latências das transferências é dado através da contagem de ciclos necessários para conclusão das transferências nos vértices. Nesta contagem, são considerados: frequência de operação do barramento e número de ciclos necessários para cada fase de uma transação de barramento.

O processo de estimação é iniciado com o cálculo do tempo necessário para a ocorrência de um ciclo de relógio do barramento. Isto pode ser realizado a partir da frequência de operação do barramento, obtida no perfil de barramento, e é dado pela equação a seguir:

$$P = \frac{1}{F} \quad (1)$$

onde P é o tempo necessário para a ocorrência de cada ciclo de barramento (período) e F é a frequência de operação informada no perfil de barramento.

Com essa informação, e as quantidades de ciclos necessários para completar as fases de uma transação, calcula-se o tempo necessário para que as transferências representadas pelos vértices possam ser executadas. O cálculo para um vértice é dado pela equação:

$$T_{(vértice)} = (q + c + d + l) . T . P \quad (2)$$

onde q, c, d e l são as quantidades de ciclos para que os eventos de transferência de barramento possam ser executados, T é o total de transferências representadas pelo vértice e P é o período entre ciclos.

Em grafos de comunicação com mais de um vértice, mesmo após realizado o agrupamento de transferências, a estimativa de desempenho da aplicação é realizada através do cálculo das latências de cada vértice. Ao fim, contabiliza-se o total, o qual irá representar o tempo total da comunicação da aplicação.

Para grafos com vértices representando transferências por rajada, do tipo simples, no cálculo do tempo podemos considerar o suporte a *pipeline*, caso este parâmetro de configuração esteja presente no perfil do barramento. Em modo de *pipeline*, as fases de configurações da transação (controle) e envio de dados são executadas em paralelo.

Para cálculo do tempo de uma transferência em rajada, utiliza-se a equação a seguir, que inclui as fases livres e paralelas:

$$T_{(vértice)} = ((q + c + d + l) + (n - 1) . c) . T . P \quad (3)$$

onde q, c, d e l são as quantidades de ciclos para que os eventos de transferência de barramento possam ser executados, n é o tamanho da rajada, T é o número total de transferências representadas pelo vértice e P é o período entre ciclos. O termo *(q+c+d+l)* contabiliza as fases não paralelas, e *(n-1).c* as fases de dados e controle em paralelo. Para o cálculo dos ciclos alinhados, a equação considera que a quantidade de ciclos necessária para a fase de controle *(c)* é igual à quantidade de ciclos da fase de dados *(d)*, desde que são executadas em paralelo dentro de um *pipeline*.

Assim como no trabalho de [LAHIRI, RAGHUNATHAN and DEY 2001], onde é possível estimar os tempos da comunicação da aplicação para cada configuração de barramento do espaço de projeto, a abordagem proposta neste artigo obtém também maior desempenho em relação às técnicas de simulação, contudo, por incluir as fases internas inerentes às transações de barramentos, busca atingir uma maior precisão nas estimativas.

4. Resultados Experimentais

Para avaliação da abordagem proposta, foram escolhidas duas aplicações que geram bastante tráfego de comunicação no barramento: uma de ordenação de um conjunto de números por *radix* [CORMEN et al. 2009] e outra de multiplicação de matrizes de números inteiros. A primeira aplicação, cujo código fonte utilizado faz parte do pacote

de *benchmarks SPLASH*[1] [WOO et al. 1995], consiste de dois processos, onde o primeiro aloca, em uma memória compartilhada, uma estrutura de dados, composta por um conjunto de números inteiros escolhidos aleatoriamente e algumas *flags* de controle. Em seguida, os inteiros na estrutura de dados serão ordenados pelos dois processos paralelamente. A segunda aplicação possui quatro processos, sendo que o primeiro aloca em memória compartilhada duas matrizes de números inteiros, escolhidos de forma aleatória, bem como respectivos *flags* de controle, e, juntamente com os demais processos, seleciona determinadas linhas e colunas das matrizes para serem multiplicadas. Para sincronizar o acesso às estruturas de dados em memória compartilhada, foi utilizado o algoritmo de *Lamport* (também conhecido como *Lamport's Bakery Algorithm*) [CORMEN et al. 2009].

Para analisar a eficiência e precisão da abordagem proposta para exploração de espaços de projeto de comunicação, foram utilizadas plataformas multiprocessadoras baseadas no processador *MIPS* (descritos em *ArchC*): uma plataforma com dois processadores para a aplicação *radix* e uma plataforma com quatro processadores para a aplicação de multiplicação de matrizes. Os demais componentes das plataformas consistiram de uma memória compartilhada, para armazenar programas e dados das aplicações, bem como dados compartilhados, e um modelo em alto nível, com precisão de ciclo, do barramento *ARM AMBA AHB* [ARM AMBA 1999]. Os parâmetros de configuração do barramento utilizados para definir o espaço de projeto, são apresentados na tabela a seguir.

Tabela 1. Perfil do barramento AMBA AHB utilizado nas aplicações do estudo de caso.

Parâmetro de Configuração	Ordenação por *Radix*	Multiplicação de Matrizes
Largura do barramento de dados	8, 16 e 32 *bits*	8, 16 e 32 *bits*
Mecanismo de arbitragem	Prioridade Fixa	Prioridade Fixa, sendo que o primeiro mestre sempre terá maior prioridade em relação aos demais mestres.
Frequência	100, 166 e 200 (MHz)	100, 166 e 200 (MHz)
Tipo de transferência	Simples com preempção, Simples sem preempção.	Simples com preempção, Simples sem preempção.

Combinando os parâmetros apresentados na Tabela 1, foi possível construir espaços de projeto totalizando 72 configurações distintas de barramento para a aplicação de ordenação por *radix* e 486 para a aplicação de multiplicação de matrizes. A diferença do tamanho dos espaços de projeto se dá pela quantidade de mestres em cada plataforma, sendo necessário combinar prioridades para dois mestres para a aplicação de ordenação por *radix* e para três mestres para a de multiplicação de matrizes.

Para as simulações, foi utilizado um microcomputador com processador Intel Core 2 Duo de 2,4 GHz com 4 GB de RAM. Os tempos de simulação do espaço de

1 Conjunto de aplicações multiprocessadas utilizadas para estudo das seguintes propriedades: balanceamento de carga computacional, taxas de computação e **requisitos de tráfego na comunicação**; além de questões relacionadas à localidade espacial e como estas propriedades podem ser escaláveis com o tamanho dos problemas e a quantidade de processadores.

projeto para as duas aplicações são mostrados na tabela a seguir.

Tabela 2. Resultados de desempenho para a simulação dos espaços de projetos das aplicações do estudo de caso.

Métrica	Ordenação por *Radix*	Multiplicação de Matrizes
Tempo médio de simulação para cada configuração	32 segundos*	45 segundos*
Tempo total de simulação de espaço de projeto	36 minutos e 10 segundos*	6 horas e 31 minutos*

* Os tempos de simulação para cada configuração do espaço de projeto incluem: tempo para reconfiguração dos parâmetros do barramento, compilação e execução do simulador.

Para a geração dos grafos de comunicação, substituiu-se o modelo de alto nível do barramento ARM AMBA das plataformas de simulação pelo barramento com modelo de comunicação genérico proposto neste trabalho. A Tabela 3 apresenta os resultados das simulações para cada aplicação, integrando informações de desempenho e dos perfis de comunicação gerados.

Tabela 3. Resultados das simulações e respectivos perfis de comunicação para as aplicações do estudo de caso.

Métrica	Ordenação por *Radix*		Multiplicação de Matrizes			
Transferências de leitura	441.346		3.114.601			
Transferências de escrita	182.422		504.980			
Total de transferências	623.768		3.619.581			
Transferências por mestre	Mestre		Mestre			
	1	2	1	2	3	4
	311.866	311.902	96.411	837.370	1.089.817	1.595.983
Tamanho do perfil de comunicação em disco	5,1 MB		62 MB			
Tempo médio de simulação para extração do perfil de comunicação	9 segundos		25 segundos			

Da análise da Tabela 3, observa-se que a aplicação de multiplicação de matrizes inclui um maior número de transferências, implicando assim em um maior tempo de simulação e um maior espaço em disco para o perfil de comunicação.

Após o grafo de comunicação ser construído, foram selecionadas as configurações do perfil de barramento, apresentado na Tabela 1, para serem incorporadas ao grafo. A Tabela 4 apresenta os resultados de desempenho para estimação dos espaços de projeto com a abordagem de grafos de comunicação proposta. Observa-se na Tabela 4 que os tempos para estimação de cada configuração são muito menores que os tempos obtidos por simulação do sistema. Consequentemente, os desempenhos para estimação dos espaços de projeto também serão maiores, com ganhos de 98,2% e 86,2% para as aplicações de ordenação por *radix* e multiplicação de matrizes, respectivamente.

Tabela 4. Desempenhos para estimação de espaços de projeto por grafos de comunicação para as aplicações do estudo de caso.

Métrica	Ordenação por *Radix*	Multiplicação de Matrizes
Tempo médio para geração de grafo de comunicação	0,9 segundos	14,7 segundos
Tempo médio para estimação de desempenho para uma configuração de barramento	0,6 segundos	6,7 segundos
Ganho médio de desempenho sobre a simulação de uma configuração de barramento	98,2%	85,1%
Tempo médio para estimação do desempenho para o espaço de projeto	41 segundos	54 minutos
Ganho médio de desempenho sobre a simulação do espaço de projeto.	98,2%	86,2%

Já a Tabela 5 apresenta os erros máximos, mínimos e médios entre as estimativas obtidas pela abordagem de grafos de comunicação e simulação completa do sistema para os espaços de projeto para as duas aplicações do estudo de caso.

Tabela 5. Erros médio, máximo e mínimo das estimativas de desempenho obtidas por grafos de comunicação para as aplicações do estudo de caso.

Métrica	Ordenação por *Radix*	Multiplicação de Matrizes
Erro máximo	4,2e-08 %	3,3 %
Erro mínimo	2,8e-08 %	3,1e-07 %
Erro médio	3,5e-08 %	0,6 %

De acordo com os resultados expostos na Tabela 5, percebe-se que as estimativas de desempenho obtidas pela abordagem de grafos de comunicação são bastante próximas às obtidas por simulação do sistema, com erro máximo, entre as duas aplicações, de 3,3% para a multiplicação de matrizes.

Os gráficos de erro acumulado, mostrados na Figura 2, reforçam a precisão das estimativas obtidas com a abordagem proposta, onde o erro acumulado de todo o espaço de projeto da aplicação de ordenação por *radix* fica bem próximo de zero (total de quase 5,0e-06) e da aplicação de multiplicação de matrizes fica em torno de 3%.

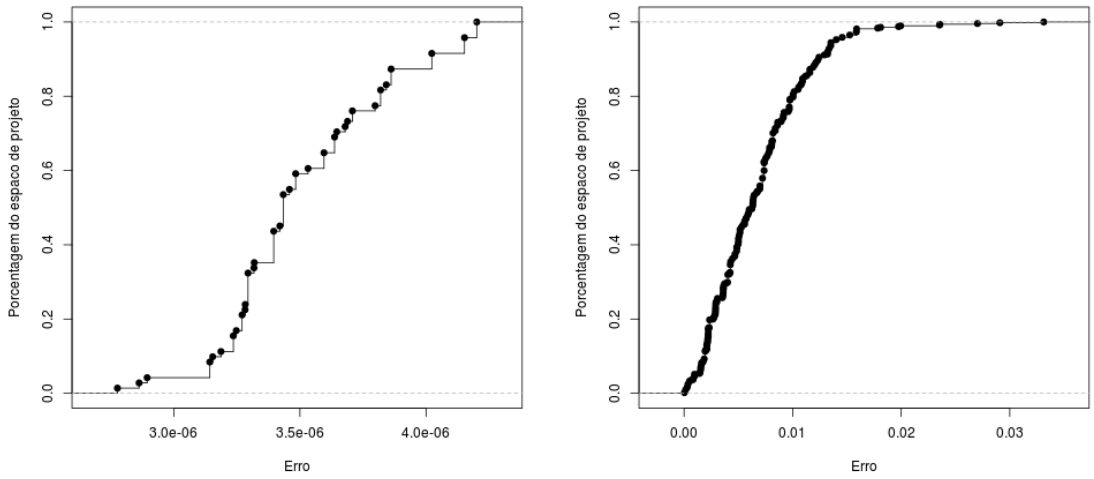

Figura 2. Gráficos de erros acumulados para as aplicações de (a) Ordenação por *Radix* e (b) Multiplicação de Matrizes.

5. Conclusões

Projetar a estrutura de comunicação de plataformas multiprocessadoras é uma tarefa difícil e que pode consumir muito tempo e esforço. Quando se considera o uso de barramentos compartilhados, muitos parâmetros devem ser considerados para que se possa atender aos requisitos de comunicação dos componentes da plataforma.

Este artigo apresentou uma nova abordagem que alia uma única simulação do sistema com uma abordagem de estimativas de desempenho de comunicação por grafos de comunicação para exploração de espaços de projeto de configurações de barramento.

Por considerar apenas uma simulação de sistema e os eventos internos às transações de barramento, a abordagem aqui proposta apresentou, nos experimentos, um ganho máximo de 98,2% no desempenho da tarefa de exploração e um erro máximo de 3,3 % nas estimativas de desempenho do sistema, sendo assim bastante otimistas em relação à abordagem de simulação completa de sistema.

6. Referências

ArchC Team (2007) "The ArchC Architecture Description Language v2.0 Reference Manual", http://ufpr.dl.sourceforge.net/project/archc/ac_lrm/2.0/ac_lrm-v2.0.pdf.

ARM AMBA. (1999) "AMBA Specification rev. 2.0", http://www-micro.deis.unibo.it/~magagni/amba99.pdf, May.

Bouajila, A.; Zeppenfeld, J.; Stechele, W.; Bernauer, A.; Bringmann, O.; Rosenstiel, W. and Herkersdorf, A. (2011) "Autonomic System on Chip Platform", Organic Computing - A Paradigm Shift for Complex Systems. Springer.

Cho, Y.-S. ; Choi, E.-J. and Cho, K.-R. (2006) "Modeling and analysis of the system bus latency on the SoC platform", Proceedings of the International Workshop on

System- Level Interconnect Prediction, pages 67–74.

Cormen, T.H., Leiserson, C.E. Rivest, R.L. and Stein, C. (2009) “Introduction to Algorithms, 3rd. Ed.”, The Mit Press.

Gasteier, M. and Glesner, M. (1999) “Bus-based communication synthesis on system level”, ACM Trans. Design Automation Electronic Systems, pages 1–11.

Jerraya, A. A.; Wolf, W. (2004) “Multiprocessor systems-on-chips”, Morgan Kaufmann.

Joo, Y.-P.; Kim, S. and Ha, S. (2009) “On-chip communication architecture exploration for processor-pool-based MPSoC”, Proc. of Design, Automation and Test in Europe.

Kim, S.; Im, C. and Ha, S. (2003) “Schedule-aware Performance Estimation of Communication Architecture for Efficient Design Space Exploration”, Proc. of the Intl. Conf. onHardware/Software Codesign and System Synthesis, pages 195–200.

Lahiri, K. and Raghunathan, A. (2004) “Power Analysis of System-level On-chip Communication Architectures”, Proc. of the Conference on Hardware/Software Codesignand System Synthesis, pages 236–241.

Lahiri, K., Raghunathan, A., Dey, S. (2001) “System-Level Performance Analysis for Designing On-Chip Communication Architectures”, IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, vol. 20, no.6, pp.768-783.

Lukasiewycz, M.; Streubühr, M.; Glaß, M.; Haubelt, C. and Teich, J. (2009) “Combined System Synthesis and Communication Architecture Exploration for MPSoCs”, Proceedings of Design, Automation and Test in Europe, pp. 472-477.

Pasricha, S.; Dutt, N.; Bozorgzadeh, E. and Ben-Romdhane, M. (2004) “Floorplan-aware Bus Architecture Synthesis”, CECS Technical Report 04-27.

Pasricha, S.; Dutt, N.; Ben-Romdhane, M. (2005) “Using TLM for Exploring Bus-based SoC Communication Architectures”, Int. Conference on Application- specific Systems, Architectures and Processors, Samos, Greece.

Rowson, J. A. and Sangiovanni-Vincentelli. A. (1997) “Interface based design”, Proceedings of the Design Automation Conference, pages 178–183.

Shanti, D. and Amutha, R. (2012) “Simulation of Inter Processor Communication Architecture in MPSoC”, European J. of Scientific Research, Vol.72 No.1, pp. 74-83.

Shin, C.; Kim, Y.-T.; Chung, E.-Y.; Choi, K.-M.; Kong, J.-T. and Eo, S.-K. (2004) “Fast Exploration of Parameterized Bus Architecture for Communication-Centric SoC design”, Proc. of the conference on Design, automation and test in Europe.

Wild, T.; Hekerdof, A.. Ohlendorf, R. (2006) “Performance evaluation for system on-chip architectures using trace-based transaction level simulation”, Proceedings of the conference on Design, automation and test in Europe, pages 248–253.

Woo, S. C., Ohara, M., Torrie, E., Singh, J. P. and Gupta, A. (1995) “The SPLASH-2 Programs: Characterization and Methodological Considerations”, Proc. of the 22nd International Symposium on Computer Architecture, pages 24-36, Ligure, Italy.