Universidade Federal de Pernambuco
Centro de Informática

Pós-Graduação em Ciência da Computação

Uma Abordagem Híbrida para Estimação de Desempenho de Comunicação em Plataformas Baseadas em Barramentos

Guilherme Álvaro Rodrigues Maia Esmeraldo

Tese de Doutorado

Recife
09 de março de 2012

Universidade Federal de Pernambuco
Centro de Informática

Guilherme Álvaro Rodrigues Maia Esmeraldo

Uma Abordagem Híbrida para Estimação de Desempenho de Comunicação em Plataformas Baseadas em Barramentos

Trabalho apresentado ao Programa de Pós-Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Doutor em Ciência da Computação.

Orientadora: *Profa. Dra. Edna Barros*

Recife
09 de março de 2012

Catalogação na fonte
Bibliotecária Jane Souto Maior, CRB4-571

Esmeraldo, Guilherme Álvaro Rodrigues Maia
Uma abordagem híbrida para estimação de desempenho de comunicação em plataformas baseadas em barramentos / Guilherme Álvaro Rodrigues Maia Esmeraldo. - Recife: O Autor, 2012.
xix, 212 folhas: il., fig., tab.

Orientador: Edna Natividade da Silva Barros.
Tese (doutorado) - Universidade Federal de Pernambuco. CIn, Ciência da Computação, 2012.
Inclui bibliografia e apêndice.

1. Engenharia da computação. 2. Computação de alto desempenho. 3. Estatística aplicada. 4. Inteligência artificial. I. Barros, Edna Natividade da Silva Barros (orientador). II. Título.

621.39 CDD (23. ed.) MEI2012 – 055

Tese de Doutorado apresentada por **Guilherme Álvaro Rodrigues Maia Esmeraldo** à Pós- Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco, sob o título **"Uma Abordagem Híbrida para Estimação de Desempenho de Comunicação em Plataformas Baseadas em Barramentos"** orientada pela Profa. **Edna Natividade da Silva Barros** e aprovada pela Banca Examinadora formada pelos professores:

Profa. Renata Maria Cardoso Rodrigues de Souza
Centro de Informática / UFPE

Prof. Abel Guilhermino da Silva Filho
Centro de Informática / UFPE

Prof. Manoel Eusebio de Lima
Centro de Informática / UFPE

Prof. Elmar Uwe Kurt Melcher
Departamento de Sistemas e Computação / UFCG

Prof. Edson Barbosa Lisboa
Instituto Federal de Sergipe

Prof. Ivan Saraiva Silva
Universidade Federal do Piauí

Visto e permitida a impressão.
Recife, 9 de março de 2012.

Prof. Nelson Souto Rosa
Coordenador da Pós-Graduação em Ciência da Computação do
Centro de Informática da Universidade Federal de Pernambuco.

Dedico este trabalho a Deus e a pessoas muito especiais.

A minha companheira, Cristiana, por seu paradoxo motivacional. Sua paciência, me apoiando e suportando nesses anos todos, bem como impaciência, decidindo e me guiando nas horas certas, são responsáveis pelo que este trabalho se tornou. Devo tudo isto a você, meu amor.

A minha filhinha Maria Caroline, pequenina em tamanho mas infinita em amor e carinho. É e sempre será minha alegria de viver.

E aos meus pais, Dr. José Maria e Dra. Dilza, fontes de inspiração. Obrigado pela maior de todas as lições: "A caminhada é feita de fé, luta e descanso. Reze e mantenha-se vivo para lutar sempre".

Amo muito todos vocês.

Agradecimentos

Agradeço a Deus, por me mostrar as possibilidades e pelo discernimento na escolha dos caminhos corretos, mesmo que muitas vezes sacrificantes e dolorosos, para mim e minha família.

À professora doutora Edna Barros, pelas oportunidades, ensinamentos, confiança e discussões calorosamente construtivas. Nunca esquecerei os anos de muito trabalho, realizações e, principalmente, crescimento técnico, moral e pessoal. Obrigado, Edna.

À banca examinadora, professores doutores: Edson Lisboa, Abel Guilhermino, Manoel Eusebio, Elmar Melcher e Ivan Saraiva, por suas contribuições para melhorar a qualidade do documento final; e, em especial, a Renata Cardoso, pelos ensinamentos e disponibilidade para me ajudar a redescobrir a estatística e desvendar a fantástica teoria da análise de regressão.

Aos amigos do Instituto Federal do Ceará, Campus Crato: João Alberto, Robson Feitosa, Yuri Lacerda, Demetrius Tahim e Cícero Carlos. Nossas conversas e idéias levantadas acerca de pesquisa, moral e ética científica foram de muito valor.

E às pessoas do Centro de Informática da Universidade Federal de Pernambuco. Seu profissionalismo, simpatia e boa vontade fazem o trabalho ser mais agradável e eficiente.

Resumo

Com o aumento da complexidade e demanda por desempenho dos sistemas embarcados e redução do custo dos microprocessadores, projetistas de sistemas embarcados têm considerado sistemas multiprocessadores como as soluções para suas aplicações. Com o avanço nas tecnologias de integração tornou-se possível integrar em um chip bilhões de transistores. Desde que um microprocessador embarcado utiliza somente alguns poucos milhões de transistores, dez ou mais microprocessadores podem ser integrados em um único chip para formar um *Multi-Processor System-on-Chip* (MPSoC). No projeto desses sistemas, são necessárias a especificação e validação do comportamento funcional da aplicação do sistema antes da implementação final, através de modelos funcionais executáveis e estruturas de *testbenches*. Técnicas, como Projeto Baseado em Plataforma (PBP), procuram, através de reuso de componentes, bem como de modelos abstratos em nível de sistema, fornecer mecanismos para simplificar e tornar mais dinâmico o processo de desenvolvimento de MPSoCs, aumentando assim a produtividade dos projetistas. Nesta abordagem, o sistema a ser desenvolvido é, inicialmente, especificado através de uma descrição em alto nível, que sofrerá refinamentos até atingir a implementação final em hardware. As funções do sistema, contidas nessa especificação, são selecionadas para serem implementadas em software ou em hardware. Estes componentes fazem parte de uma arquitetura predefinida, conhecida como plataforma, que pode ser modificada para ser adaptada às restrições de projeto. MPSoCs são compostos por muitos componentes de processamento que executam processo concorrentes que se comunicam, portanto suas arquiteturas de comunicação *on-chip* devem atender às necessidades de comunicação das aplicações. Assim, enquanto existe uma grande quantidade de trabalhos que suportam as fases de particionamento/mapeamento, comparativamente, pouca pesquisa tem endereçado o problema de análise de comunicação para auxiliar o projeto de arquiteturas de comunicação dessas sistemas. As técnicas existentes para explorar as opções de configuração da estrutura de comunicação são imprecisas, pois fazem estimativas estáticas, descartando efeitos dinâmicos da arquitetura, como contenção de barramento, ou possuem baixa eficiência, pois têm que simular cada configuração do espaço de projeto. O objetivo deste trabalho é oferecer suporte de análise de comunicação nos processos de

seleção e refinamento das arquiteturas de comunicação, após a aplicação ter sido particionada e mapeada para uma plataforma, de acordo com o PBP. O uso da abordagem proposta permite que o projetista obtenha estimativas precisas de desempenho de comunicação para as configurações de barramento de todo o espaço de projeto, e, consequentemente, possa selecionar uma configuração que melhor atenda às restrições de comunicação do projeto.

Palavras-Chave: Sistemas Embarcados, Barramentos, Análise de Comunicação, Predição de Desempenho, Programação Genética, Modelos Lineares Generalizados.

Abstract

With the increasing of complexity and performance demand of embedded systems, as well as with the reduction of microprocessors cost, embedded systems designers have considered multiprocessors systems as the solutions for their applications. The improvement of the integration technologies made it possible to integrate billions of transistors onto a single chip. As an embedded microprocessor is composed by a few million transistors, ten or more microprocessors can be integrated into a single chip to form a Multi-Processor System-on-Chip (MPSoC). In the development of these systems, designers have to specify and validate the behavior of the system application prior to final implementation, by using executable functional models and *testbench* structures. Approaches, such as Platform Based Design (PBD), have considered platform components reuse and abstract models at the system level as good practices to simplify and turn more dynamic the process of developing MPSoCs, thereby increasing the designers productivity. In this approach, the system in development is initially specified using a high level description, which will gradually be refined down to the final implementation in hardware. The system functions described in the initial specification are selected to be implemented in software or in hardware components. These components compose an architecture known as a platform, which can be modified and adapted to meet the application constraints. MPSoCs are composed by many processing components that implement concurrent communicating processes, so the on-chip communication architecture must meet the applications communication requirements. Thus, while there are several studies focusing on the partitioning/mapping processes, comparatively few research projects have addressed the communication analysis problem to support the design of systems, including efficient communication architectures. Some existing techniques to explore the configuration options of the communication structure are inaccurate, since they perform static estimates and do not take into account the dynamic effects of architecture, such as bus contention, or they are inefficient, since they have to simulate each configuration of the design space. This work aims to support communication analysis in the selection and refinement of communication architectures in the design of multi-processors systems, considering that the application has been partitioned and

mapped to a platform, according to the PBD approach. By using the proposed approach, designer can have accurate estimates of the performance of the bus-based communication architecture for the entire design space, and, hence, can select a configuration that meets the communication constraints of the system.

Keywords: Embedded Systems, Buses, Communication Analysis, Performance Prediction, Genetic Programming, Generalized Linear Models.

Índice

Lista de Tabelas

Lista de Figuras

Principais Abreviaturas

ACG – Arquitetura de Comunicação Global

AE – Algoritmo Evolutivo

AEIQ – Algoritmo Evolutivo com Inspiração Quântica

AEMO – Algoritmo Evolutivo Multiobjetivos

AG – Algoritmo Genético

AHB – Advanced High-Performance Bus

AIC – Akaike Information Criterion

AMBA – Advanced Microcontroller Bus Archtecture

BMF – Bloco de Memória Física

BML – Bloco de Memória Lógica

CAD – Computer-Aided Design

CCATB – Cycle Count Accurate at Transaction Boundaries

CI – Circuito Integrado

DFDI – Diagrama de Fluxo de Dados Interprocesso

DSP – Digital Signal Processor

EP – Elemento de Processamento

GAC – Grafo de Análise de Comunicação

GC – Grafo de Comunicação

GPU – Graphics Processing Unit

ISS – Instruction Set Simulator

MED – Modelo Específico de Domínio

MEF – Máquina de Estados Finitos

MLG – Modelo Linear Generalizado

MPSoC – Multi-Processor System-on-Chip

MRL – Modelo de Regressão Linear

NoC – Network on Chip

PBP – Platform Based Design

PG – Programação Genética

PP – Pool de Processadores

RISC – Reduced Instruction Set Computer

RTL – Register-Transfer Level

SCSI – Small Computer System Interface

SGBDR – Sistema de Gerenciamento de Banco de Dados Relacional

SoC – System-on-Chip

SQR – Soma dos Quadrados dos Resíduos

TDMA – Time Division Multiple Access

TLM – Transaction Level Modeling

Capítulo 1

Introdução

Sistemas embarcados estão em todos os lugares, embora, frequentemente, passem despercebidos em nossas vidas cotidianas. Estão presentes em telefonia (celulares e centrais telefônicas), em eletrodomésticos (fornos micro-ondas, torradeiras, máquinas de lavar), eletroeletrônicos (aparelhos de TV, DVD players, home theaters, vídeo games), automóveis (controladores de injeção eletrônica, sistemas de ignição, freios ABS e sensores em geral), calculadoras, PDA's, equipamentos de rede (hubs, switches e firewalls), etc [99][100].

Um sistema embarcado consiste de um subsistema eletrônico específico para uma aplicação e é utilizado em uma entidade maior como um componente, um instrumento ou um veículo. O sistema embarcado pode encapsular a funcionalidade da aplicação em duas formas: executando software em CPUs ou em hardware especializado [93].

Devido tanto ao desenvolvimento das tecnologias de circuitos integrados (CIs), que hoje permitem integrar mais de um bilhão de transistores em um único chip [128], requisitos mais rígidos das aplicações e ciclos cada vez menores para lançamento de novos produtos [57], 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 [86]. A característica fundamental de um SoC é sua complexidade. Um chip de memória, por exemplo, pode ter muitos transistores, mas sua estrutura regular faz com que seja um componente e não um sistema. Os componentes de um SoC podem variar de acordo com a aplicação. Um SoC pode incluir memória on-chip (RAM e/ou ROM), microprocessador, interfaces com periféricos, lógica de controle de I/O, conversores de dados, módulos de hardware especificamente projetados e outros componentes que abrangem sistemas computacionais completos [69]. A utilização de SoCs pode trazer vários benefícios ao sistema, como a redução de custos, tamanhos físico e dissipação de potência, além de aumento de desempenho [12].

Com a redução do custo dos microprocessadores, projetistas de sistemas embarcados têm considerado sistemas distribuídos como a solução para suas aplicações [92]. Desde que um microprocessador embarcado utiliza somente alguns poucos milhões de portas lógicas, dez ou mais microprocessadores podem ser integrados em um único chip para formar um *Multi-Processor System-on-Chip* (MPSoC) [70], que é um SoC que inclui vários processadores homogêneos ou heterogêneos conectados por estruturas de interconexão. Na prática, muitos SoCs são MPSoCs, até mesmo porque é muito difícil projetar um SoC complexo sem utilizar muitos processadores[86]. Atualmente, os projetistas de sistemas embarcados têm utilizado comumente MPSoCs para desenvolver sistemas de alto desempenho [94].

No projeto de MPSoCs existem algumas restrições que devem ser observadas, entre elas: alto desempenho, capacidade de executar aplicações de tempo real, áreas físicas minimizadas e bem utilizadas, baixo consumo de energia ou mecanismos para, dinamicamente, diminuir esse consumo e interfaces com outros dispositivos [101][102].

Face a recente complexidade e aumento do tempo de projeto, em projetos de MPSoCs, tornou-se comum especificar e validar o comportamento funcional do sistema antes da implementação final, através de modelos funcionais executáveis que separam o comportamento da arquitetura [67]. Técnicas como Projeto Baseado em Plataforma (PBP) [76] e Abstrações em Nível de Sistema [23][95][96], procuram, através de reuso de componentes e plataformas, bem como modelos abstratos em nível de sistema, fornecer mecanismos para simplificar e tornar mais dinâmico o processo de desenvolvimento de MPSoCs, aumentando a produtividade dos projetistas.

Outro suporte importante para redução do tempo de projeto, é a utilização de ferramentas comerciais e/ou acadêmicas, como *ConvergenSC* [97] e *PDesigner* [98], que simplificam os processos de seleção e configuração de componentes, além de permitir edição e mapeamento de aplicações embarcadas, através de modelagem visual. Além disso, essas ferramentas automatizam a construção e execução de modelos simuláveis de MPSoCs.

1.1 Definição do Problema

MPSoCs são compostos por muitos componentes de processamento que executam processo concorrentes que se comunicam, portanto suas arquiteturas de comunicação on-chip devem atender às necessidades de comunicação das aplicações.

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 [86]. Estas interconexões podem variar em complexidade, podendo ser canais dedicados, barramento ou redes on-chip (NoCs) [103].

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 [111] [155][156].

Porém, os trabalhos publicados em [5],[59], [94] e [110] mostram que, por serem um recurso compartilhado, o barramento torna-se o fator limitante no desempenho da comunicação, bem como no consumo de potência, podendo ser comparado aos componentes do sistema (processadores e memórias), com quase 15% do consumo de energia de um sistema embarcado [108]. Além destes, outros estudos, como os apresentados em [104], [105], [106] e [107], mostram os impactos da estrutura de comunicação no sistema, como limitantes no consumo de energia e desempenho.

Desta forma, os projetistas de sistemas embarcados são, então, tentados a criar novos protocolos de barramento arbitrários, porém é comum utilizar os protocolos mais populares, uma vez que as companhias de semicondutores fabricam componentes que podem implementar diretamente esses protocolos [55].

Contudo, selecionar e reconfigurar arquiteturas de comunicação baseadas em barramentos padronizados, como por exemplo o barramento AMBA [48], para conseguir atender a todos os requisitos de comunicação da aplicação, é um processo que consome muito tempo [111]. Isto deve-se ao grande espaço de projeto 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 [87][111].

O processo de análise deve considerar todas estas características na configuração do barramento. Por exemplo, a utilização de barramentos de maior largura, alta frequência, presença de DMA e pipeline podem aumentar, significantemente, o desempenho da plataforma [61]. A utilização de algoritmos de arbitragem mais eficientes torna possível reduzir a contenção, no caso de vários dispositivos concorrendo pela utilização de um barramento compartilhado, aumentando também, consequentemente, o desempenho do sistema [25][106][109]. Vahid et al. [9] cita, também, que a utilização de topologias com hierarquia de barramento pode diminuir a latência na comunicação com periféricos.

Em contrapartida, Dandamudi explica em [10] que aumentar a largura do barramento torna o projeto mais caro, pois, tanto o barramento quanto suas interfaces, irão necessitar de mais espaço físico no *chip*. Isto também ocorre com adição de suporte à *pipeline*, que aumenta a complexidade do barramento e, consequentemente, o espaço físico ocupado[113]. Ainda em [10], é mostrado que aumentar a frequência de operação não é sinônimo de alto desempenho, pois frequências muito altas acarretam problemas de engenharia, tais como desvio de barramento (*bus skew*). O desvio é causado pelo fato de que sinais em linhas diferentes trafegam em velocidades levemente diferentes. A diferença torna-se um problema crítico quando a frequência do relógio é aumentada. Por exemplo, um processador com frequência de operação de 1GHz utiliza um período de 1ns enquanto um processador com frequência de 100MHz pode tolerar atrasos de propagação maiores com um período de 10ns. Além disso, em frequências maiores, o tamanho dos fios e espaçamento entre eles, que diminuem cada vez mais[112], induzem atrasos, que podem ser comparáveis ao período do relógio. Já Pasricha et. al., em [110], reafirma sobre o uso de DMA no aumento de desempenho de uma parte do sistema, mas destaca que pode degradar o desempenho de outro mestre acoplado ao mesmo barramento compartilhado, sendo necessária a modificação da estratégia de arbitragem ou topologia.

Observa-se então que existem configurações diferentes da estrutura de comunicação baseada em barramento que podem prejudicar o desempenho do sistema, aumentar o consumo de energia e tamanhos físicos ocupados, podendo inviabilizar o projeto de um MPSoC[87].

Shin et. al. destaca, em [51], que 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.

Em reconhecimento à dificuldade de seleção e configuração de arquiteturas de comunicação, torna-se evidente a necessidade de mecanismos de suporte aos projetistas no processo de exploração do espaço de projeto de arquitetura de comunicação.

Atualmente, no projeto de MPSoCs há uma clara tendência em se utilizar a abordagem de Projeto Baseado em Plataformas (PBP) [49]. 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)[56][114]. Estes componentes de hardware fazem parte de uma arquitetura predefinida, conhecida como plataforma, que pode ser modificada de forma 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 requisitos de comunicação em uma arquitetura de comunicação que melhor satisfaça 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.

Segundo estudo apresentando em [114], existe uma grande quantidade de trabalhos que focam nas fases de particionamento/mapeamento, e, comparativamente, pouca pesquisa tem endereçado o problema de análise de comunicação para auxiliar o projeto de arquiteturas de comunicação. As técnicas existentes que consideram os efeitos da arquitetura de comunicação podem ser divididas nas seguintes categorias:

1. Abordagens baseadas na simulação completa de sistema, cujos modelos podem incluir componentes e sua comunicação em diferentes níveis de abstração [25][49][50][52][54]. O uso de abstrações de comunicação permite estabelecer um compromisso entre precisão e desempenho. Entretanto, essas técnicas irão requerer uma simulação completa do sistema para avaliar cada ponto do espaço de projeto, implicando em alto custo computacional.

2. Técnicas que buscam, através de modelos matemático estatísticos [60][61][62], 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 [114]. Além disso, podem ser demasiadamente otimistas, por não considerarem os efeitos dinâmicos da aplicação, como latências de espera por contenção de barramento [58][59], ou pessimistas, assumindo o pior caso para contenção. Por não incluir detalhes suficientes de comunicação, a utilização deste tipo de técnica pode resultar em estimativas não tão precisas.

3. Abordagens híbridas[12][51][63][68][69][70][71], 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 e automatização do processo de análise e refinamento de comunicação.

1.2 Objetivos

Devido aos requisitos mais complexos e rígidos de comunicação das aplicações embarcadas emergentes, à dificuldade de selecionar suas arquiteturas de comunicação baseadas em barramento, e a ausência de mecanismos de suporte à análise de comunicação, eficientes e precisos. Este trabalho de tese teve como objetivo o desenvolvimento de uma nova técnica de análise híbrida capaz de estimar o desempenho de comunicação de uma plataforma para todas as configurações de barramento contidas no espaço de projeto. Essa técnica é baseada na simulação do sistema, em heurísticas e em estimativas estáticas,.

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. Espera-se que com o uso da abordagem proposta, o projetista possa obter estimativas precisas do desempenho da aplicação para as configurações de

barramento considerando todo o espaço de projeto, e, consequentemente, possa selecionar a configuração que atenda às restrições de desempenho do projeto. Adicionalmente, a abordagem proposta disponibiliza informações suficientes de comunicação para que o projetista possa, além de selecionar e configurar uma arquitetura de comunicação, identificar gargalos na aplicação.

Para tanto, foi desenvolvido um modelo de barramento genérico de comunicação que pode ser acoplado a qualquer plataforma especificada abstratamente em nível de sistema, com comunicações por transações (TLM), ou em RTL. Este componente realiza comunicação ponto a ponto e captura todas as transações de leitura e escrita.

Adicionalmente, foi proposta uma técnica que a partir de uma simulação da aplicação mapeada para a plataforma, retorna um modelo linear do barramento que permite calcular o desempenho para qualquer ponto do espaço de projeto. A técnica proposta se baseia em Programação Genética [18] para obtenção do modelo linear mais adequado e na técnica de Projeto de Experimentos [22] para definição do conjunto inicial de configurações a serem consideradas.

1.3 Contribuições

Este trabalho endereça o problema de selecionar e refinar a arquitetura de comunicação baseada em barramento de uma plataforma MPSoC, após o particionamento e mapeamento da aplicação. A principal contribuição deste trabalho é uma abordagem que combina as técnicas de simulação completa de sistema, algoritmos evolucionários e análise estática para dar suporte à análise de comunicação em plataformas baseadas em barramentos.

Com o uso da técnica de análise de comunicação proposta, neste trabalho de tese, 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 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.

Dessa forma, o processo de desenvolvimento de arquiteturas de comunicação de MPSoCs é simplificado, além de minimizar o tempo de projeto e trazer qualidade à implementação final do sistema.

1.4 Estrutura do Documento

Este documento é estruturado da seguinte maneira: O Capítulo 2 apresenta os conceitos fundamentais necessários para compreensão dos demais capítulos. Dentre os conceitos abordados, há noções de projeto baseado em plataformas, barramentos e suas características e algoritmos genéticos. No Capítulo 3, as principais abordagens de análise de comunicação são apresentadas. A abordagem proposta e resultados experimentais obtidos são apresentados nos Capítulos 4 e 5, respectivamente. A partir das técnicas propostas neste trabalho de tese foram desenvolvidas ferramentas para suporte à análise de comunicação. Essas ferramentas são apresentadas no Capítulo 6 e, por fim, as considerações finais e descrição de trabalhos futuros são apresentadas no Capítulo 7.

Capítulo 2

Conceitos Fundamentais

2.1 Introdução

Este capítulo aborda descrições de conceitos fundamentais teóricos necessários para compreensão dos próximos capítulos.

O capítulo se inicia com uma explanação da abordagem de Projeto Baseado em Plataforma (PBP). Desde que PBP necessita de mecanismos para análise, a abordagem proposta neste trabalho de tese consiste num mecanismo de suporte à exploração de estruturas de comunicação baseadas em barramentos em plataformas virtuais[1]. Na mesma seção é abordada uma breve descrição da linguagem de sistemas embarcados, SystemC [4], que é comumente utilizada em PBP e está presente em vários trabalhos relacionados, os quais serão apresentados no próximo capítulo, além de também ser utilizada na modelagem dos componentes apresentados neste trabalho de tese.

Na sequência, são apresentados conceitos e definições de barramento, tais como tipos de transferências, arbitragem e protocolos, para um melhor entendimento das possíveis configurações do espaço de projeto da arquitetura de comunicação.

Por fim, é apresentada a abordagem de otimização baseada em algoritmos evolutivos, também conhecida como algoritmos genéticos ou evolucionários, que foi utilizada na obtenção do modelo estatístico proposto o qual permite a estimativa do desempenho do sistema para diferentes configurações do espaço de projeto de arquitetura de comunicação.

[1] Neste trabalho de tese, o termo 'plataforma virtual' poderá ser utilizado onde se aplicar o termo 'plataforma abstrata descrita em nível de sistema'.

2.2 Projeto Baseado em Plataforma

De acordo com Sangiovanni-Vincentelli em [76], existem várias definições para Projeto Baseado em Plataforma (PBP), as quais dependem do domínio da aplicação.

No domínio dos circuitos integrados (CI), uma plataforma é considerada um CI flexível que pode ser adaptado para uma aplicação particular através de programação de um ou mais componentes do chip. Programação pode implicar em "adaptação do metal" (*gate arrays*), modificações elétricas, personalização do *Field-Programmable Gate Array* (FPGA), ou software para executar em um microprocessador ou DSP. Por exemplo, uma plataforma precisa ter uma microarquitetura fixa, para minimizar custo de criação e confecção da máscara, mas ser flexível o suficiente para permitir seu uso para uma classe de aplicações. O problema com essa abordagem é a falta potencial de otimização, que pode tornar o desempenho baixo e aumentar os tamanhos físicos das plataformas.

A abordagem PBP é definida em [76] como a criação de plataformas estáveis baseadas em microprocessadores que podem rapidamente ser estendidas, adaptadas para uma classe de aplicações, e podem ser utilizadas por projetistas para o desenvolvimento rápido de sistemas embarcados. A abordagem PBP tem sido utilizada com o objetivo de aumentar a produtividade e e melhorar a qualidade no desenvolvimento de sistemas dedicados complexos.

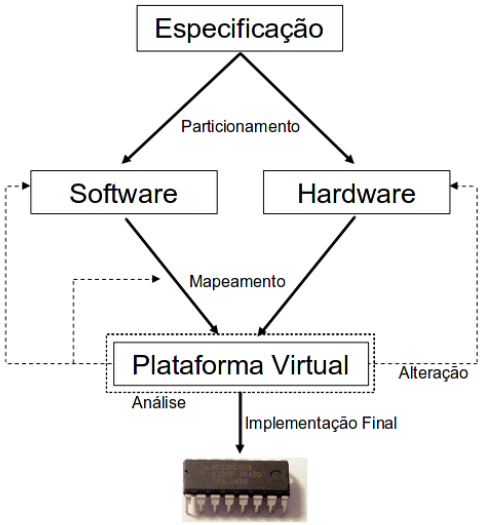


Figura 1 - Fluxo de projeto descrito na abordagem de projeto baseado em plataformas.

Segundo essa metodologia, o fluxo de projeto, que pode ser visto na Figura 1, começa com a especificação das funcionalidades do sistema que se deseja desenvolver. Em seguida, as funcionalidades são particionadas para serem executadas em hardware ou em software. Este processo é conhecido como particionamento. Logo após, será escolhida uma plataforma pré-definida que possa, através de seus componentes, atender aos critérios do particionamento. Com isto, as funcionalidades são mapeadas para os componentes da plataforma.

Após o processo de mapeamento, o sistema é analisado de acordo com determinadas métricas, consideradas como essenciais ao projeto. Atualmente, utilizam-se modelos de plataformas virtuais, cujas implementações são realizadas através de modelos de simulação, descritos em linguagens de descrição de hardware, como VHDL, e de alto nível, como SystemC. Com plataformas virtuais pode-se avaliar se a arquitetura satisfaz requisitos funcionais, ou até mesmo temporais, em fases iniciais do projeto, através de ajustes na aplicação, na plataforma ou até mesmo no próprio mapeamento entre ambas as partes. Durante estes ajustes, o espaço de projeto deve ser explorado em busca de melhores resultados, segundo os requisitos definidos para a aplicação.

No projeto de arquiteturas de comunicação, a fase de análise concentra-se em avaliar as métricas do sistema para diferentes configurações da estrutura de comunicação. O trabalho de tese apresentado neste documento propõe uma abordagem que permite ao projetista analisar e selecionar a configuração da estrutura de comunicação que atende às restrições impostas no projeto.

Do ponto de vista de processo de desenvolvimento, uma extensão da definição de plataforma pode ser considerada como uma camada de abstração no fluxo de projeto, a qual facilita os processos de refinamento para uma camada de abstração subsequente. Um ponto importante na aplicação deste princípio de projeto é a definição cuidadosa das camadas. Plataformas podem ser modeladas em diferentes níveis de abstração nas distintas fases do projeto, de forma que, dependendo da camada, uma instância de plataforma pode ser simplesmente um conceito ou uma abstração em software.

Sangiovanni-Vincentelli define em [76], que essa pilha de camadas é um modelo completo que permite visualizar a plataforma sob o ponto de vista da aplicação e da implementação. Sendo assim, é possível separar o processo de desenvolvimento da aplicação do processo de implementação da arquitetura. Além do mais, em cada camada é possível direcionar

o foco do projeto às métricas que devem ser avaliadas para a aplicação, como, por exemplo, análise funcional, temporal ou espacial.

No projeto de arquiteturas de comunicação baseadas em barramentos, quando considera-se a análise de desempenho, o fluxo de projeto pode ser dividido em seis camadas, como pode ser visto na Figura 2.

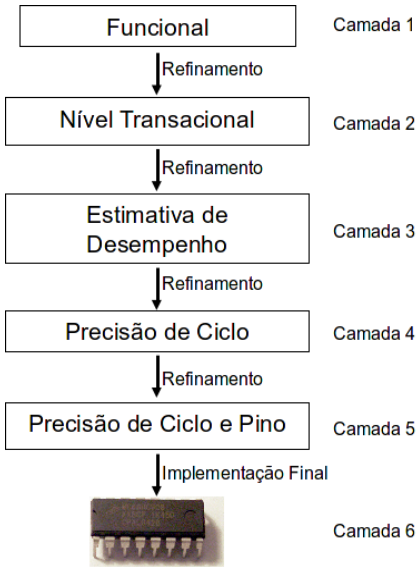


Figura 2 - Processo de refinamento de plataforma segundo a abordagem de projeto baseado em plataforma.

Nesse fluxo, a primeira camada consiste de modelagem funcional do barramento. Nesta camada, uma transferência de barramento consiste de chamadas de funções, as quais são utilizadas para configuração da transferência (*handshake*) e envio de dados, aos dispositivos de destino. O primeiro refinamento do modelo funcional, representado pela Camada 2 na Figura 2, consiste na conversão das funções de comunicação para transações de barramento. Uma transação é uma operação atômica que engloba uma transferência completa de barramento.

A camada seguinte consiste na inserção de mecanismo de estimativas de desempenho às transações. Com isso pode-se estimar o tempo de duração da cada transação e com isso tentar determinar o desempenho de comunicação da aplicação.

Na Camada 4, o modelo de estimativas de desempenho é substituído por um modelo mais preciso. Neste novo modelo, existe um relógio de sistema e cada transação é dividida em eventos. Os eventos de transação são disparados de acordo com os ciclos de relógio do sistema.

Assim, uma vez que se pode determinar o instante em que cada evento ocorrerá, é possível estimar, com precisão de ciclo de relógio, a duração de cada transação.

A última camada, que antecede a implementação final do sistema em hardware, é o refinamento do modelo de comunicação com precisão de ciclo para precisão de pino. Nesta ultima abstração, as transferências de barramentos são capturadas em termos de pinos, ou sinais, utilizados para interconectar os componentes da plataforma ao barramento [144]. Neste modelo os eventos de transferência, da Camada 4, são distribuídos entres os sinais de comunicação e continuam sendo determinados pelo ciclos de relógio.

Atualmente, na abordagem de PBP, são utilizadas linguagens de alto nível para modelar plataformas virtuais, permitindo assim avaliar restrições funcionais e temporais em fases iniciais do projeto. Uma das linguagens mais comumente utilizadas é a SystemC, por suportar a modelagem de componentes de plataformas em vários níveis de abstração, como funcional, com estimativas de desempenho, precisão de ciclo e precisão de pino. O Apêndice A apresenta descrições mais detalhadas das estruturas, definidas pela linguagem SystemC, utilizadas para modelagem de componentes de plataformas virtuais em alto nível.

2.3 Barramento

Um barramento é um recurso compartilhado para conectar múltiplos subsistemas [5]. Os barramentos são muito versáteis, pois permitem conectar, facilmente, novos componentes às plataformas. Em um dado momento, apenas um dispositivo (seja um registrador, ULA, memória, ou algum outro componente de plataforma) pode utilizá-lo.

Entretanto, este compartilhamento resulta, frequentemente, em atrasos nas comunicações. A taxa de transferência de um barramento é afetada por seu comprimento, assim como pelo número dos dispositivos que o compartilham [5].

Quanto à sua estrutura, barramentos podem ser utilizados em plataformas por consistirem de um caminho comum que conecta vários dispositivos (Figura 3).

Por causa do compartilhamento, o protocolo (conjunto de regras para uso do barramento) é muito importante. A Figura 3 mostra um barramento típico composto por linhas de dados, linhas de endereço, linhas de controle e linhas de energia. As linhas utilizadas para transferências de dados são chamadas, frequentemente, de barramento de dados. Estas linhas contêm a informação atual que deve ser transferida de um local para outro. Linhas de controle indicam

qual dispositivo tem permissão para utilizar o barramento, o tipo de transferência (escrita ou leitura, por exemplo), interrupções, sinais para sincronização por relógio de sistema, etc. Linhas de endereço indicam o local (na memória, por exemplo) de qual dado deve ser lido ou no qual deve ser escrito. As linhas de energia fornecem a alimentação elétrica necessária.

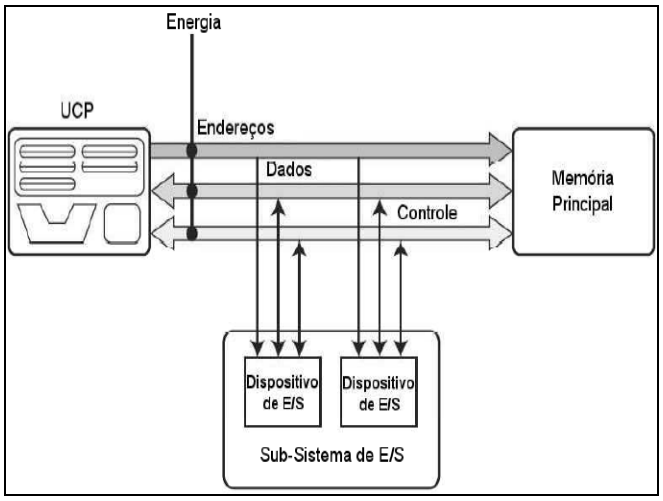


Figura 3 - Componentes de um barramento típico.

O uso de barramentos para dados e para endereços separadamente contribui para o aumento da eficiência, desde que pode-se enviar sinais de dados e endereços em paralelo aproveitando os ciclos de *pipeline* dos processadores. Porém, observe que o uso de barramentos separados aumenta o custo total do sistema [10]. Por exemplo, um processador de 32 bits com 32 linhas para dados e para endereços necessita de um total de 64 fios para estes dois barramentos. Para diminuir o custo destes sistemas, projetistas podem utilizar barramentos multiplexados, os quais não são dedicados a uma única função: informações de dados e endereçamento podem trafegar no mesmo barramento. Obviamente, barramentos multiplexados diminuem o custo, porém reduzem também o desempenho do sistema.

Quanto à temporização, barramentos podem ser classificados como síncronos e assíncronos. Síncronos são aqueles onde a sequência de eventos de uma transmissão é controlada pelo relógio de sistema. Em barramentos assíncronos, as linhas de controle coordenam a transmissão e um protocolo de *handshake* deve ser utilizado para garantir que a transferência seja efetuada.

Devido ao fato de se utilizar um protocolo de *handshake*, no lugar do relógio, para coordenar transações, é possível conectar, ao mesmo barramento, dispositivos com diferentes latências de relógio de forma mais eficiente, tornando os barramentos assíncronos mais escaláveis.

Na transmissão de informações, os dispositivos são classificados em mestre e escravo, onde o dispositivo mestre é o que inicia as ações e o escravo é o que responde aos pedidos do mestre.

Em sistemas com mais de um dispositivo que pode ser mestre, um mecanismo de arbitragem de barramento é necessário para coordenar o acesso ao barramento. Esquemas de arbitragem são utilizados para minimizar a contenção quando vários dispositivos mestres tentam utilizar o barramento no mesmo instante. Tempo de contenção é definido pelo intervalo entre o instante de solicitação do uso do barramento por um mestre e o instante que a permissão é concedida.

Visando aumentar a eficiência no uso de barramentos, que é a quantidade de transferências por tempo de uso do barramento por um mestre, alguns barramentos suportam diferentes tipos de transferências. Dentre estes tipos, podemos destacar os que seguem:

- **Transferência Simples:** este tipo caracteriza a transação mais fundamental, que consiste na transferência de uma única célula de informação.

- **Transferência em Rajada (*Burst*) ou Bloco (*Block*):** é caracterizada pela sequência de várias transferências simples e é bastante utilizada para transferência de blocos de informações. Este tipo de transferência é muito comum entre memória principal e memória cache, onde a unidade de transferência é dada em função do tamanho da linha da memória cache.

- **Transferência de Leitura-Modificação-Escrita (*Read-Modify-Write*):** consiste em uma transferência de leitura, onde os dados lidos são modificados e escritos novamente no mesmo endereço de leitura. São úteis em sistemas multiprocessadores concorrentes. Nestes sistemas, uma estrutura de dados compartilhada ou seção do código, chamada de seção crítica, deve ser acessada de forma mutuamente exclusiva (i.e. um mestre por vez no mesmo instante) para que os processos possam se comunicar. Para garantir esta

propriedade, utilizam-se estruturas de dados como mutex (*mutual exclusion*) ou semáforos [42]. Leitura-modificação-escrita dá suporte ao uso destas estruturas de dados, de forma que os processadores podem modificar os valores das variáveis de sinalização, sem permitir que outros processadores intervenham. Outros tipos de operações, como transferência de Testar-e-Configurar (*Test-and-Set*), são utilizadas para fornecer, também, este tipo de suporte[119].

- **Transferência de Leitura-Após-Escrita (Read-After-Write):** é um tipo transferência que consiste de uma de escrita seguida imediatamente por uma operação de leitura a partir do mesmo endereço de memória. Este tipo de operação é comumente utilizado para propósitos de checagem [119].

- **Divisão (*Split*):** é um tipo de transferência que pode ser interrompida por um periférico para ser realizada em outro momento. Normalmente, é utilizado por periféricos com estados de latência, ou seja, que não permitem atender imediatamente requisições dos mestres[48].

Os barramentos podem ser divididos de acordo com os tipos de informações transmitidas e os tipos de dispositivos que os utilizam. Barramentos locais de processador são curtos, de alta velocidade, pois seu desempenho deve aproximar-se do desempenho do sistema de memória para maximizar a largura de banda. Barramentos de I/O ou de periféricos são, tipicamente, maiores que os barramentos locais de processador e permitem muitos tipos de dispositivos com taxas de transferências variadas [5].

Para tentar minimizar áreas físicas, é comum desenvolver sistemas com apenas um barramento de alta velocidade, mas esta abordagem tem algumas desvantagens:

1. **Um barramento de alta velocidade requer dispositivos com interfaces de alta velocidade**: quando um periférico não permite comunicação de alta velocidade, uma interface deste tipo pode resultar na necessidade de portas e *buffers* adicionais para interface, o que significa um maior consumo de potência e custo.

2. **Barramentos de alta velocidade são específicos de processadores**: a integração de

periféricos com interfaces muito complexas pode tornar o projeto muito caro e impraticável.

3. **Presença de muitos periféricos no barramento:** a limitação no número de periféricos pode resultar em um barramento lento, devido às latências adicionais por trocas de mestres e contenção.

Consequentemente, é comum que projetos de sistemas incluam dois níveis de barramento: um barramento de alta velocidade para processadores e um de baixa velocidade para os periféricos. O barramento local do processador conecta, tipicamente, microprocessador, as memórias cache, os controladores de memória, os coprocessadores de alta velocidade e, certamente, os processadores de aplicação específica.

O barramento de periféricos conecta aqueles processadores que não possuem alta prioridade nos barramentos de processador. O barramento de periférico segue, normalmente, um padrão de barramento industrial, como AGP[6]e PCI[120], garantindo assim portabilidade aos periféricos. É, normalmente, um barramento mais estreito e mais lento do que um barramento local do processador, possuindo assim menos fios.

Para conectar dois barramentos, de taxas de transmissão diferentes, utiliza-se um componente chamado ponte (*bridge*). Uma ponte é um processador de propósito específico que permite comunicação entre componentes conectados em barramentos distintos. Por exemplo, um microprocessador pode gerar uma leitura no barramento local de processador com um endereço correspondente a um periférico. A ponte detecta que o endereço corresponde a um periférico, e assim pode gerar um sinal de leitura no barramento de periféricos. Depois de receber o dado, a ponte envia-o para o microprocessador. Este, assim, não necessita ter conhecimento da ponte - recebe o dado, depois de alguns ciclos, como se o periférico estivesse no barramento local do processador. A ponte se comporta como escravo do barramento local de processadores e mestre do barramento de periféricos.

Uma hierarquia de três níveis é também possível, como proposto na abordagem descrita em [8]. O primeiro nível é o barramento de processador, o segundo nível é o barramento do sistema e o terceiro é barramento de periféricos. O barramento do sistema seria um barramento de alta velocidade, assim como os barramentos locais de processador, que seria utilizado para

descarregar o tráfego do barramento de processadores. Isto pode beneficiar sistemas complexos com muitos coprocessadores[9].

2.3.1 Classificação dos Mecanismos de Arbitragem

Em sistemas multiprocessadores, ou em plataformas que possuam DMA, um barramento pode ser compartilhado entre vários mestres. Assim, para efetivar a comunicação entre componentes, deve-se garantir que o barramento não seja acessado por mestres diferentes no mesmo instante.

Para tanto, os barramentos incluem um mecanismo de arbitragem, que decide qual dispositivo terá prioridade na utilização de um barramento compartilhado [129].

Vários autores utilizam diferentes tipos de classificação de mecanismos de arbitragem, como as apresentadas em [5] e em [9]. Porém, Dandamudi apresenta em [10] um modelo de classificação bastante completo, onde os vários mecanismos de arbitragem podem ser subdivididos conforme taxonomia apresentada na Figura 4.

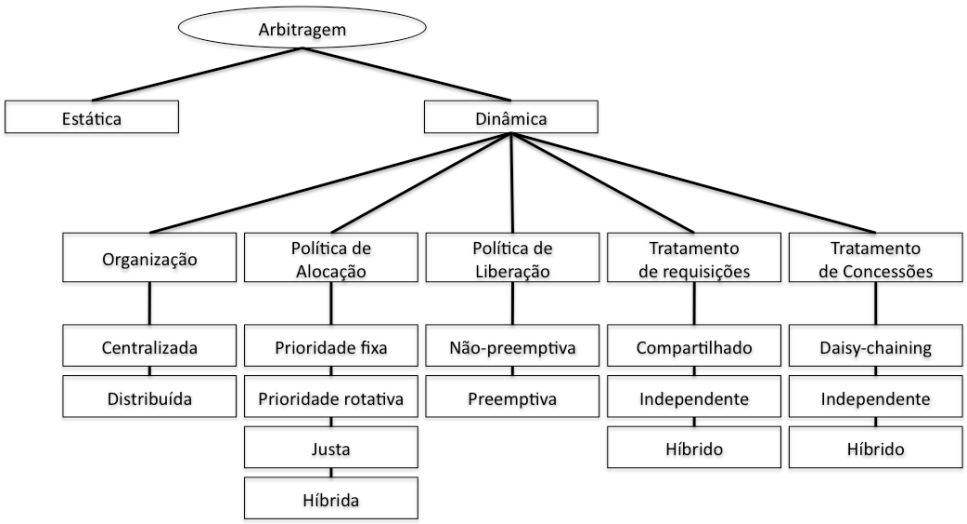


Figura 4 - Espaço de projeto de mecanismo de arbitragem.

O mecanismos são classificados como arbitragens estáticas e arbitragens dinâmicas, descritas a seguir:

1) **Arbitragem Estática:** a alocação do barramento entre os mestres é realizada de forma pré-determinada [130]. Por exemplo, podemos utilizar alocação *Round Robin*, onde os mestres que utilizam o barramento entram em uma fila de prioridades, na qual o primeiro elemento da fila tem permissão para efetuar transferências. Após efetuar as operações, o primeiro mestre da fila é movido para o fim da fila, o segundo, que passa a ser o primeiro, recebe permissão para efetuar transferências. A grande vantagem deste mecanismo é a simplicidade de implementação. Porém, o barramento poderá ser alocado para determinados mestres que não realizarão transferências, tornando este mecanismo ineficiente.

2) **Arbitragem Dinâmica:** a alocação do barramento é realizada em resposta a uma requisição de um mestre. Os mecanismos de arbitragem deste tipo podem ainda ser classificadas de acordo com as seguintes subdivisões:

   a) **Organização:** pode ser implementada de duas formas:

      i) **Centralizada:** um árbitro centralizado recebe requisições de todos os mestres do barramento. O árbitro, utilizando a política de alocação em efeito, determina para que mestre o barramento será concedido. Uma vez que a transferência é encerrada, o mestre atual deve liberar o barramento; a política de liberação determina o mecanismo de liberação do barramento.

      ii) **Distribuída:** o hardware de arbitragem é distribuído entre os mestres, de forma que um algoritmo distribuído irá determinar o mestre que irá usar o barramento. Assim, cada possível mestre irá participar do processo de arbitragem.

   b) **Política de Alocação:** quando vários mestres competem pelo barramento, o árbitro deve implementar uma política de alocação, que independe da organização (centralizada ou distribuída). De acordo com Dandamudi em [10], é possível identificar quatro tipos de políticas:

      i) **De Prioridade Fixa:** Nesta política, cada mestre possui uma prioridade fixa única. Quando muitos mestres requisitam o barramento, vence aquele de maior prioridade.

Desde que as prioridades são fixas, o processo de atribuição deve ser cuidadoso: senão é possível que um mestre não permita que outros mestres, de menor prioridade, utilizem o barramento (este bloqueio é conhecido como *starvation* [42]). Porém, este bloqueio pode não ser um problema desde que as requisições não sejam contínuas.

ii) **De Prioridade Rotativa:** nesta política a prioridade do mestre não é fixa. A prioridade do mestre pode ser dada, por exemplo, em função do tempo de espera para utilizar o barramento. Assim, quanto mais o mestre esperar, maior prioridade terá. Este tipo de política evita os bloqueios citados anteriormente.

iii) **Justas:** esta política é dada por justiça, que é um importante critério para uma política de alocação. Em sua forma básica, este tipo de política evita os tipos de bloqueios citados anteriormente. Como exemplos, pode-se utilizar (1) políticas de prioridades rotativas, (2) requisições predefinidas em janelas (são atendidas todas as requisições de uma janela para que a próxima seja atendida) e (3) requisições atendidas em função de tempos.

iv) **Híbridas:** podem ser utilizadas/combinadas políticas justas e de prioridades fixas ao mesmo tempo.

c) **Política de Liberação:** governa as condições nas quais o mestre atual do barramento libera o barramento para que os demais componentes do sistema possam utilizá-lo. Podemos classificá-las em:

i) **Não-Preemptiva:** neste modelo o mestre libera o barramento quando a transação for completada. Existem duas variações para implementação desta política:

(1) **Baseada em Transação:** o mestre libera o barramento quando sua transação é finalizada. Para realizar uma nova transação, o mestre deve requisitar o barramento novamente. Esta é a forma mais simples de implementar esta política.

(2) **Baseada em Demanda:** a desvantagem da abordagem anterior é que se há apenas um mestre, haverá ciclos adicionais de arbitragem para cada transação. Nas políticas baseadas em demanda, o mestre atual só irá liberar o barramento se houver uma requisição de outro mestre, senão continuará a utilizar o barramento.

ii) **Preemptiva:** Uma desvantagem potencial com as políticas não-preemptivas é que um mestre pode segurar o barramento por um longo tempo, dependendo do tipo de transação. Políticas preemptivas forçam o mestre a liberar o barramento sem completar a transação atual.

d) **Tratamento de Requisições:** estas políticas definem como são estruturadas as linhas de requisição do barramento e como o tratamento das requisições é realizado. Podem ser implementadas de três formas:

i) **Compartilhado:** todos os mestres compartilham um único sinal de requisição do barramento. Normalmente, este tipo de abordagem é utilizada com Tratamento de Concessões *Daisy-Chaining* (ver definição adiante). Uma vez que não se conhece os mestres que requisitaram o barramento, a concessão também deverá ser dada de forma aleatória.

ii) **Independente:** Nesta abordagem, todos os mestres são ligados ao barramento por linhas de requisição diferentes. Desde que as requisições ao barramento são recebidas em linhas diferentes, o árbitro pode implementar várias políticas de arbitragem, como prioridade rotativa ou política justa, ou até mesmo uma abordagem híbrida.

iii) **Híbrido:** combina as abordagens anteriores: os mestres são divididos em *N* classes. Linhas de requisição diferentes são utilizadas por cada classe como um esquema independente. Porém, dentro de cada classe, os mestres são conectados através de *Daisy-Chaining* (ver definição adiante).

e) **Tratamento de Concessões:** políticas que definem como são estruturadas as linhas de concessão do barramento e como o tratamento das concessões é realizado. Podem ser implementadas de três formas:

i) ***Daisy-Chaining*:** utiliza um sinal de concessão que é passado do mestre de maior prioridade para o de menor prioridade. Este esquema de arbitragem é simples mas não é justo, uma vez que existe a possibilidade de *starvation* (mestres de menor prioridade podem nunca receber permissão de uso do barramento).

ii) **Independente:** cada mestre possui um sinal de concessão. Uma vez que o arbitro seleciona o mestre que deverá utilizar o barramento, o respectivo sinal de concessão deverá ser habilitado.

iii) **Híbrido:** combina as abordagens anteriores: os mestres são divididos em *N* classes. Linhas de concessão diferentes são utilizadas por cada classe como um esquema independente. Porém, dentro de cada classe, os mestres são conectados através de *Daisy-Chaining*.

2.3.2 Fases dos Protocolos de Barramentos

Stallings destaca, em [119], que uma comunicação entre um mestre e um escravo de barramento é composta de vários eventos, conforme sequência é dada a seguir:

1. Mestre: envia uma requisição para uso do barramento
2. Mestre: o uso do barramento é concedido e o barramento é alocado para o mestre
3. Mestre: insere endereço/dados no barramento
4. Escravo: escravo é selecionado pelo barramento
5. Mestre: sinaliza transferência de dados
6. Escravo: envia/recebe dados
7. Mestre: libera o barramento

Analisando-os, podemos agrupá-los em fases iniciadas com eventos de mestres de barramento, descritas a seguir:

1. **Requisição de uso do barramento:** O mestre requisita ao barramento permissão para utilizá-lo. Esta permissão será concedida, ou não, pelo mecanismo de arbitragem.

2. **Configuração da transação:** O mestre envia ao barramento os sinais de controle. Nesta fase será configurado o modo de operação do barramento, como, por exemplo, escrita, leitura, operações de rajada, etc., e o endereço do dispositivo de onde o dado será lido ou escrito.

3. **Transmissão de dados:** Dependendo da operação, o mestre deverá enviar dados (operações de escrita) ou receber (operações de leitura).

4. **Liberação**: O barramento é liberado para que outros mestres possam realizar transferências. No caso de não ocorrer liberação, o mestre continua sendo o proprietário do barramento.

Como visto na seção anterior, em relação à temporização, existem dois tipos de barramentos: síncronos e assíncronos. Para cada um destes, as quatro fases estão presentes, porém são implementadas de forma diferente.

Em barramentos síncronos, a sequência dos eventos contidos nestas fases é gerenciada pelo relógio de sistema. Para barramentos assíncronos, outros eventos poderão ocorrer, além dos descritos anteriormente, que caracterizam a negociação da transação (*handshaking*).

Normalmente, os eventos embutidos na fase de Requisição são basicamente três: requisição (*request*), permissão (*grant*) e ocupado (*busy*). O primeiro, requisição, é sinalizado pelo mestre e significa que o mesmo deseja fazer transações. Se a permissão for concedida, o barramento sinaliza permissão para aquele mestre e sinaliza ocupado para os demais dispositivos.

Para a fase de Configuração, o mestre disponibiliza o endereço do dispositivo de/para onde se deseja fazer a transferência e, em paralelo, os sinais de controle. Os sinais de controle podem ser de vários tipos, entre eles:

- **Escrita ou leitura:** se a operação do mestre será de escrita ou de leitura. Este sinal é dirigido ao escravo para que ele possa se preparar para o tipo de transferência requerido pelo mestre.

- **Tipo de transferência:** dois tipos são, normalmente, suportados por barramentos:

    1. não-sequencial, para transferências simples e

    2. sequencial, para transferências por rajada ou bloco.

- **Tipo de rajada:** indica ao escravo o tipo de rajada. É utilizada com dispositivos que fazem endereçamento incremental (endereços de uma rajada calculados a partir do incremento de um endereço inicial) ou *wrapping* (endereços de uma rajada calculados a partir do incremento de um endereço inicial, porém há um limite de incremento que, quando atingido, o endereçamento volta ao endereço inicial).

- **Tamanho do dado:** tamanho, em bits, do dado a ser transferido. Este tipo de informação é importante para o mestre que deseja enviar um dado de tamanho menor que a largura do barramento [48].

- **Controle de proteção:** informações adicionais sobre acesso ao barramento. Pode ser utilizado para indicar, por exemplo, um modo de acesso privilegiado ou modo de usuário, para gerenciamento de memória por mestres, indicando se o endereço corrente é armazenável em memória cache ou não, etc [48].

Na fase de envio de dados, o barramento de dados é utilizado para essa transferência, pelo mestre ou escravo, de acordo com a operação configurada. Dependendo da política de liberação do barramento, o sinal de liberação pode ser ativado pelo mestre.

A próxima seção apresenta a técnica de otimização, conhecida como Algoritmos Genéticos. Esta técnica é utilizada em parte dos trabalhos relacionados e no trabalho de tese aqui proposto.

2.4 Algoritmos Genéticos

Propostos, inicialmente, por John Holland [72], Algoritmos Genéticos (AG), ou evolutivos, são abordagens de otimização por busca adaptativa. O princípio básico dos algoritmos genéticos, que possui analogia com evolução biológica por meio da seleção natural segundo Darwin[28], é, através de aplicação de transformações unárias (mutação) ou de ordem maior (cruzamento), evoluir uma população de indivíduos, que são potenciais soluções de um problema[47]. Estes indivíduos irão competir entre si: os mais aptos serão selecionados para sofrer as transformações para, em seguida, compor a próxima geração. Depois de um determinado número de gerações, o algoritmo de evolução, usualmente, convergirá e o melhor indivíduo representará a solução ótima ou aproximadamente ótima.

A técnica de algoritmos genéticos visa selecionar os pontos de maior aptidão, que podem minimizar ou maximizar a função objetivo. Apesar da aleatoriedade, algoritmos genéticos possuem os históricos das melhores soluções, dentre os candidatos, as quais podem ser utilizadas para encontrar soluções ainda melhores. Este processo é realizado através de processos iterativos, onde cada iteração é conhecida como geração. Um fluxo de projeto do algoritmo genético, proposto por Herreral et. al. em [73], pode ser visto na Figura 5.

De acordo com Herreral et. al., o primeiro passo em um algoritmo genético é a representação dos problemas através de soluções genotípicas, utilizando estruturas de dados de tamanho finito com conjunto de símbolos finito [73]. Os estudos de Gen e Cheng, em [17], e de Koza, em [121], afirmam que a representação da solução é muito importante, pois reflete diretamente na qualidade dos resultados. Cada pedaço indivisível desta representação, da estrutura de dados adotada, é chamado de *gene*, por analogia com as partes fundamentais que compõem o cromossomo biológico[17]. A representação desse elemento é arbitrária, sendo responsabilidade do projetista e deve ser adequado ao problema em questão. Cada um desses elementos, os genes, representa a presença ou não de uma característica, o seu genótipo, e podem ser combinados formando as características reais do indivíduo, ou o seu fenótipo[47]. É comum a representação binária de um gene, onde o valor 1 representa a presença de certa característica e 0 a ausência. Com esta representação, pode-se, por exemplo, utilizar vetores para representar um conjunto de características (genótipos). Outras estruturas de dados podem ser utilizadas, como árvores em Programação Genética[18][21].

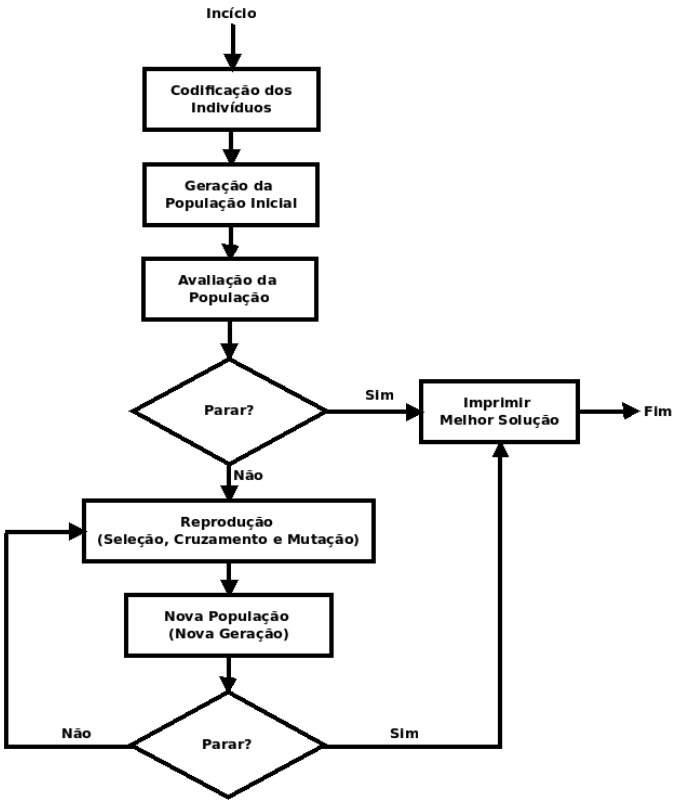


Figura 5 - Fluxo do algoritmo genético.

Após a codificação da estrutura dos indivíduos, é gerada uma população inicial. Este passo consiste em compor um conjunto de candidatos para a solução do problema, onde cada candidato é criado aleatoriamente. O tamanho da população para a formação desse conjunto inicial deve ser definido cuidadosamente, uma vez que o desempenho do algoritmo genético é extremamente sensível ao tamanho das populações. Caso o tamanho seja pequeno demais, não haverá espaço para que se tenha uma variedade genética suficientemente grande dentro desta população, o que faz com que o algoritmo seja de incapaz encontrar boas soluções. Caso o tamanho seja muito grande, o algoritmo irá consumir muito tempo se aproximando de uma busca exaustiva. O tamanho da população pode ser encontrado empiricamente, porém será específico para cada problema. Existem abordagens com tamanho de população variável, como é caso da abordagem baseada em idades, proposta inicialmente em [74], ou baseada em variabilidade genética. A primeira estabelece uma idade para cada indivíduo que aumenta a cada nova geração. O indivíduo só irá morrer após atingir um limite máximo de vida preestabelecido. Já na segunda abordagem, a população é amostrada e a diferença entre o genótipo dos indivíduos

selecionados é calculada. Se a diferença for pequena, a população pode ser aumentada através de inserção de indivíduos aleatórios.

De acordo com o fluxo proposto por Herrera et. al. em [73], a próxima fase consiste na avaliação da população inicial. A função de avaliação (ou função de ajuste ou função de custo) é a maneira de se determinar a qualidade de um indivíduo como solução do problema em questão. Essa função calcula um número que caracteriza a quão bons são os parâmetros representados no cromossomo que resolve o problema. Como AGs são técnicas de maximização, a função deve avaliar a qualidade do indivíduo de forma a estabelecer um parâmetro de comparação. Segundo Gen e Cheng, em [17], a função de avaliação deve refletir as necessidades do problema da forma mais direta possível e embutir todas as restrições do problema. A função deve punir os cromossomos que desrespeitam as restrições e permitir boas avaliações dos que atendam às restrições. Isto, somado ao objetivo de que a função tenha um mínimo de máximos locais possível.

Ao fim da avaliação, é iniciado o processo de evolução. Nesta fase são aplicadas as operações de seleção, mutação e evolução. A primeira é responsável pela seleção dos indivíduos que irão compor o conjunto de pais. Neste conjunto, a função de cruzamento genético irá atuar, de forma que o conteúdo genético de cada indivíduo irá ser transferido para outro, gerando novas soluções. O objetivo é agrupar as melhores características em certos indivíduos, formando soluções melhores. A função de mutação irá selecionar alguns dos indivíduos para ter seu conteúdo genético modificado de forma aleatória, de forma a inserir variabilidade genética às populações, evitando a convergência do algoritmo para um máximo local.

As funções de seleção, cruzamento e mutação podem ser implementadas de várias maneiras conforme descrito a seguir:

1. **Seleção:**
    a. **Roleta viciada**: os indivíduos são representados na roleta proporcionalmente ao seu índice de aptidão. Assim, os mais aptos ocupam uma proporção maior da roleta. Após a roleta girar um determinado número de vezes, dependendo do tamanho da população, os indivíduos sorteados são escolhidos para participar da próxima geração ou para o conjunto de pais.
    b. **Torneio**: consiste em selecionar aleatoriamente uma série de indivíduos da população e fazer com que entrem em competição direta pelo direito de

ser pai, usando como critério sua avaliação. Os vencedores dos torneios serão selecionados para próxima geração ou para o conjunto de pais. Como a escolha é aleatória, este método possibilita que indivíduos com baixa avaliação possam compor o conjunto de pais, permitindo a variabilidade genética.

c. **Estocástica uniforme**: os indivíduos são mapeados para segmentos contínuos de uma linha, sendo que o tamanho de cada segmento é proporcional ao valor da avaliação. Para selecionar o *n* indivíduos que serão pais, é sorteado um número *i* entre *0* e *1/n*, que serve como base do sorteio. Depois são atribuídos n ponteiros que passam a apontar para segmentos de reta, nas posições *i*, *i+1/n*, *i+2/n*, ..., *i+(n-1)/n*. Os indivíduos referentes aos segmentos apontados serão selecionados para aplicação dos operadores genéticos.

d. **Elitismo**: os indivíduos mais ajustados de cada geração são automaticamente selecionados para compor a geração seguinte - em pequena escala, pois tem como efeito colateral a convergência prematura do AG - o objetivo é garantir o aumento de desempenho com as gerações.

2. **Cruzamento**:

   a. **Ponto de corte**: constitui uma posição na representação vetorial entre dois genes de um cromossomo. Cada indivíduo de n genes contem *n-1* pontos de corte, e este ponto de corte é o ponto de separação entre cada um dos genes que compõem o material genético da cada pai. Depois de sorteado o ponto de corte, cada pai é separado em duas partes: uma à esquerda e outra à direita do ponto de corte. Observe que não necessariamente as duas partes devem possuir o mesmo tamanho. Os filhos serão então compostos da concatenação de uma parte de cada pai.

   b. **Dois pontos de corte**: similar ao anterior, porém com um acréscimo: em vez de sortear um só ponto de corte, são sorteados dois. O primeiro filho será então formado pela parte do primeiro pai fora dos pontos de corte e pela parte do segundo pai entre os pontos de corte e o segundo filho será formado pelas partes restantes. Com este modelo é possível combinar mais

esquemas cromossômicos do que o de um ponto de corte, tendo assim maior desempenho.

c. **Uniforme**: para cada gene de um indivíduo é sorteado um número zero ou um. Se o valor sorteado for igual a um, o filho número um recebe o gene da posição corrente do primeiro pai e o segundo filho o gene corrente do segundo pai. Por outro lado, se o valor sorteado for zero, as atribuições serão invertidas.

3. **Mutação**:
   a. **Simples**: a operação de mutação é associada a um valor de probabilidade normalmente bastante baixo, da ordem de 0,5% [17], e é sorteado um número entre 0 e 1. Se o número sorteado for menor que a probabilidade da operação de mutação predeterminada então o operador atua sobre o gene em questão, alterando-lhe o valor aleatoriamente. Repete-se então o processo para todos os genes componentes dos dois filhos.
   b. **Mutação dirigida**: inicia sua atuação após um certo número de gerações, pois quando ativada, busca as *n* melhores soluções dentro da população padrão e verifica qual é o padrão genético que têm em comum. Depois de descoberto o padrão que as melhores soluções têm em comum, o operador realiza as mutações (dentro desse esquema somente). Isso vai fazer com que surjam novas soluções com bagagem genética radicalmente diferente daquela que o domina a população naquele instante[17]. Com isso evita-se a convergência genética prematura e o algoritmo continua progredindo para encontrar soluções ainda melhores.

Após aplicação dos operadores de seleção, cruzamento e mutação, o algoritmo finalizará se: (1) foi encontrada uma solução satisfatória, desde que um dos candidatos atenda às restrições de solução do problema; (2) o melhor candidato perdura entre um número predeterminado de gerações; (3) o algoritmo atinge um limite máximo de gerações, que também é predeterminado.

2.5 Conclusão

Este capítulo apresentou descrições de conceitos fundamentais teóricos, necessários para entendimento dos demais capítulos. Inicialmente, foi apresentada a abordagem de Projeto Baseado em Plataformas, a qual define o conceito de plataformas reutilizáveis configuráveis, utilizadas para aumento no desempenho de projeto de arquiteturas de comunicação em sistemas embarcados. Em seguida, foram apresentados conceitos básicos de barramento, sendo abordadas taxonomias estruturais, temporais e funcionais, bem como protocolos de transferências de dados, hierarquia e classificação dos mecanismos de arbitragem. Por fim, o capítulo encerrou com uma apresentação da técnica de otimização baseada em algoritmos genéticos, componente essencial para o trabalho de tese aqui apresentado.

Capítulo 3

Estado da Arte

3.1 Introdução

Sistemas embarcados são projetados para aplicações específicas e arquiteturas podem diferir em funções dos seus componentes e das configurações dos mesmos. Customização é o processo de modificar uma plataforma genérica, e definir sua configuração para suportar uma aplicação específica. Este processo consiste na configuração otimizada dos componentes de hardware da plataforma (topologias de barramento, tamanho de buffer *on-chip*, largura do barramento de dados e endereços, configuração de cache, etc.) para aumentara eficiência do sistema, enquanto se reduz o custo, consumo de energia e/ou tamanho físico [84].

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 [12][85]. 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 [86].

Neste trabalho de tese, 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 [87].

Nas próximas seções, serão apresentados resumos de exemplos ilustrativos para os três tipos de técnicas citados.

3.2 Simulação Completa do Sistema

Nas técnicas baseadas em simulação, os efeitos das arquiteturas de comunicação são incorporados através do desenvolvimento de modelos de simulação, normalmente codificados em linguagens de descrição de hardware ou linguagens de alto nível[25][49][50][52][54]. Nestes modelos é possível especificar as características da comunicação, como, por exemplo, configurações de protocolos e topologias, e simular a execução do sistema.

Trabalhos científicos recentes têm dado grandes destaques a este tipo de abordagem e serão abordados nas próximas subseções.

3.2.1 Projeto baseado em interface

Rowson e Sangiovanni-Vincentelli propuseram em [49] uma abordagem de desenvolvimento baseada em interface, que busca abstrair a comunicação entre componentes, focando em suas interfaces. Assim, o modelo proposto para as interfaces permite que a estrutura de comunicação, que neste caso é baseada em barramento, possa ser desenvolvida de forma incremental, através de refinamentos, e hierarquicamente, sem necessidade de modificação dos componentes ligados a ela.

Para tanto a abordagem utiliza, no nível mais abstrato, um tipo de pacote de dados, chamado *token*, que representa a comunicação entre duas ou mais entidades que executam as funcionalidades do sistema em desenvolvimento. As funções que acessam o dado incluem operações de escrita, leitura e rajada. Simulações neste nível de abstração podem ser realizadas por meio da troca de *tokens*, codificada através da troca de apontadores, entre os componentes conectados ao barramento. O uso de *tokens* permite abstrair os processos de comunicação para suportar checagem de propriedades, como validação funcional, do sistema a ser desenvolvido.

Após a validação funcional, o modelo do sistema deve ser refinado. No modelo refinado, a criação dos pacotes deverá ser realizada via software e transmitida nas estruturas de hardware. O refinamento da estrutura de comunicação deve tratar os mesmos tipos de comunicação oferecidos pelos pacotes da camada de abstração anterior (leitura, escrita, rajada, etc), assim a troca de pacotes é refinada para um conjunto de transações de barramento, que agora pode ser compartilhado entre os mestres e incluir algum mecanismo de arbitragem. Naturalmente, neste nível de abstração ainda não é possível realizar análise de desempenho com precisão.

O próximo passo do refinamento inclui seleção de uma implementação de barramento, como uso do barramento AMBA[48] ou Avalon[133]. Agora, é possível ter informações com precisão de ciclo de cada transação de barramento que está ocorrendo. A arbitragem incluirá também informações temporais.

Ainda em [49], é proposto um simulador, chamado *Cheetah*, que dá suporte ao projeto baseado em interface. O simulador foi desenvolvido para suportar uma melhora no desempenho de simulação e o estilo de modelagem. O aumento do desempenho das simulações vem da abstração da interface. A exploração do estilo de modelagem inclui investigações em como simular uma interface abstratamente e configurar diferentes implementações de interface no simulador facilmente.

Este simulador é baseado em eventos, que são utilizados para ativar ações nos módulos e interfaces. Durante a simulação, a interface simplesmente repassa os *tokens* gerados pelos elementos funcionais, inserindo latências antes de entregar os *tokens* aos destinatários e informar que a transferência finalizou. Apenas com as latências adicionadas, ainda não é possível ter precisão de ciclo nas simulações. As sincronizações, como, por exemplo, arbitragem para liberar acesso ao barramento, também são implementadas através de eventos.

O simulador permite ainda avaliar as interfaces entre os módulos, através da captura de informações de comunicação. Os dados capturados podem conter eventos de inicio e fim de transações e sincronizações, necessários para reconstruir o comportamento das interfaces com precisão de ciclo. Assim, o dado capturado pode ainda ser expandido para incluir informações mais detalhadas das comunicações.

As interfaces consistem, então, de um tipo de especificação (cujos dados são transportados por uma transação simples através da interface), um cálculo de atraso para uma dada transação e um mecanismo que permite expandir, ou simular, uma transação capaz de anotar informação com precisão de ciclo. Com isso, é possível ter estimativas precisas acerca de cada transação e consequente aumento de desempenho nas simulações. Por outro lado, para suportar a exploração do espaço de projeto de arquiteturas de comunicação, ainda é necessário simular o sistema para cada configuração de barramento.

3.2.2 Uma abordagem prática para otimização de arquitetura de barramentos em nível transacional

Em [50] é proposta uma abordagem para projeto de arquiteturas de comunicação,

especificamente utilizando barramentos, baseada na simulação do sistema. O objetivo primário dessa metodologia é garantir a largura de banda para tratar vários fluxos de dados relacionados com aplicações distribuídas.

A proposta é baseada na fluxo de atividades descrito a seguir:

1. O particionamento de cada aplicação em hardware e software é realizado de forma semelhante à abordagem PBP;

2. Os projetistas criam um modelo de plataforma para simulação, através da conexão dos componentes. Simuladores com precisão de ciclo de processadores gerais ou DSP e alguns periféricos comuns, como temporizadores, controladores de interrupção, modelo de memória e outros, estão disponíveis em uma biblioteca. O processo de síntese de comunicação é realizado a partir de uma especificação das ligações, fornecida pelo projetista através de uma *netlist*, entre os componentes da plataforma. Em seguida o projetista seleciona, na biblioteca, os componentes responsáveis pela comunicação na plataforma, incluindo o mecanismo de arbitragem. A versão atual suporta todos os modelos do barramento AMBA [48]. Por fim, deve-se selecionar o mapeamento de memória para os mestres dos barramentos. Com isso, o mecanismo proposto de síntese gera uma descrição em CowareC (linguagem de descrição de hardware descontinuada em favor de SystemC)[64], de cada componente do subsistema de comunicação da plataforma, adicionando descrições para decodificadores de endereços, multiplexadores e pontes (caso haja hierarquia de barramentos). Com isto, a plataforma é gerada e, na sequência, o software da aplicação é compilado e mapeado para a plataforma;

3. Os projetistas avaliam então o desempenho da arquitetura baseados nos resultados quantitativos extraídos da execução do software no modelo de arquitetura. É possível utilizar ferramentas comerciais de extração para análise do barramento e da memória, em relação ao comportamento na simulação (tempos de transferências, latência, contenção, etc.), desde que a descrição para anexar o sistema de análise seja adicionada a cada modelo. O processo se repete nas etapas (2) e (3), para cada nova configuração de barramento, e com isso é possível explorar várias configurações até que a melhor arquitetura seja determinada.

Os autores dessa abordagem de análise endereçaram três métricas básicas de projeto: 1) Desempenho: os modelos de simulação são descritos em linguagem C, com modelos de comunicação transacionais; 2) Precisão: todos os elementos das plataformas são modelados com precisão de ciclo, de forma que as estimativas de desempenho das aplicações possam ter valores próximos aos valores dos modelos reais; e 3) Flexibilidade: suporte a modelos parametrizáveis

de barramentos, além de hierarquia de barramentos, com geração automática do modelo de simulação.

Com a geração automatizada da estrutura de comunicação e o uso de modelos transacionais de comunicação, o fluxo iterativo para exploração de comunicação, proposto em [50], pode obter aumento de desempenho. Contudo, a necessidade de simulação do sistema para cada configuração do barramento, utilizando modelos com precisão de ciclo, agregada com a possibilidade de construção de espaços de projeto maiores, dado o suporte a inclusão de novos modelos de barramento à biblioteca, podem conduzir em alto custo computacional para cobertura total do espaço de exploração da arquitetura de comunicação.

3.2.3 Análise de comunicação on-chip em uma plataforma multiprocessadora

Em [52] é apresentado um *framework* para simulação do sistema visando análise de comunicação. A plataforma de simulação para sistemas multiprocessadores gera tráfego na arquitetura de comunicação por meio de aplicações reais executando em uma plataforma incluindo processadores ARM.

O sistema inteiro é simulado (iniciadores da comunicação, memórias locais assim como compartilhadas, dispositivos de sincronização), com precisão de ciclo ou sinal (interfaces físicas de componentes de hardware, *pin accurate*). Isto permite uma análise realística do desempenho considerando as interconexões entre os componentes de hardware do sistema. Segundo os autores, o desempenho pode ser estimado precisamente para diferentes classes de aplicações (voltadas para comunicação ou computação), diferentes tipos de arquiteturas de software (sendo *standalone* ou com suporte de sistemas operacionais) e configurações distintas dos componentes de hardware (tamanho da cache, latência da memória externa, etc.).

A arquitetura da plataforma é composta por uma quantidade configurável de processadores ARM de 32 bits, memórias locais e compartilhadas, módulo gerenciador de interrupção, um semáforo e interconexões de 32 bits entre eles. Esta interconexão poder ser um barramento AMBA AHB (*Advanced High-Performance Bus*) ou uma topologia qualquer do barramento STBus (ST Microelectronics). Os núcleos processadores foram modelados a partir de uma versão adaptada de um simulador de instruções ARM, codificado em C++, chamado SWARM[66]. Todos os outros componentes foram codificados em SystemC. Para a plataforma foi portado o sistema operacional RTEMS[65]. Este foi escolhido por ser um SO bastante leve

para sistemas embarcados, oferecendo o suporte para multiprocessamento e permitindo chamadas nativas para comunicação e sincronização em ambientes multiprocessadores.

Segundos os autores, foram criadas ferramentas e bibliotecas para dar suporte ao desenvolvimento e análise eficientes de novas aplicações e *benchmarks* em seu framework, entre elas:

- as novas aplicações podem ser compiladas através de compiladores cruzados. Scripts são utilizados para gerar as plataformas multiprocessadoras automaticamente;

- foi criado um conjunto de *scripts* que permitem a compilação das aplicações;

- chamadas simples de sistema, providas pelos componentes da biblioteca, permitem a constituição de perfis de desempenho. Desta forma, estatísticas podem ser coletadas durante a execução do simulador, como, por exemplo, durante o processo de inicialização do sistema operacional RTEMS, ou ainda durante a execução de seções críticas de algoritmos.

O resultado da simulação pode ser configurado de forma a incluir: (i) estatísticas sobre desempenho de processadores e de estrutura de comunicação (entre elas: uso do barramento, eficiência de uso do barramento e tempo médio de leituras), (ii) formas de onda para todos os sinais e (iii) traces de acesso a memória.

O ambiente de simulação permite incluir pontos de parada de execuções no código fonte das aplicações, realizar a execução de instruções passo a passo; e inspecionar o conteúdo da memória. Para aplicações que necessitam do suporte de um sistema operacional, é possível ainda inspecionar trocas de mensagens e status dos processos.

Assim, nesse trabalho, a exploração do espaço de projeto de uma arquitetura de comunicação pode ser realizada através de simulação da plataforma e análise dos resultados para diferentes configurações suportadas pelo *framework*.

Para avaliação dessa abordagem, foram utilizadas dois tipos de análise quantitativas. A primeira consistiu na comparação de desempenho de cinco configurações diferentes de barramento, para quatro benchmarks, visando estabelecer suas diferenças. O segundo tipo de análise utilizou as técnicas propostas em [52] para análise de latências das estruturas de comunicação utilizadas no estudo, sob diferentes parâmetros do sistema, como tamanho da cache, latências de memória e otimizações do compilador. Os resultados mostraram que foi possível avaliar a melhor estrutura de comunicação, configuração de cache e otimizações de software, dentre as contidas no espaço de projeto, que implicaram em maior desempenho do sistema.

A grande vantagem dessa abordagem é gama de informações, para suporte à análise de

comunicação, que podem ser extraídas das simulações, como formas de onda dos sinais do barramento, *traces* de acesso à memória, além de métricas de análise estatística descritiva, como uso, eficiência ou tempos de leitura médios. No entanto, de acordo com os autores desse trabalho, o impacto para geração de todas dessas informações pode aumentar em até 25% o tempo de simulação.

3.2.4 Utilizando TLM para exploração de arquiteturas de comunicação baseadas em barramentos

O *gap* crescente de produtividade nos projetos de software e de hardware de sistemas embarcados motivou o desenvolvimento de novas técnicas para projetos de sistemas embarcados. Assim, diferente dos trabalhos citados anteriormente, em [25] é proposto um modelo de codificação de transações em alto nível de abstração que, segundo os autores, fornece uma abordagem rápida e eficiente para explorar arquiteturas de comunicação baseadas em barramento.

Em [25], os autores apresentam um modelo de fluxo de projeto que é baseado em reuso de componentes de hardware e modelagem em alto nível. Um esboço desse fluxo pode ser visto na Figura 6.

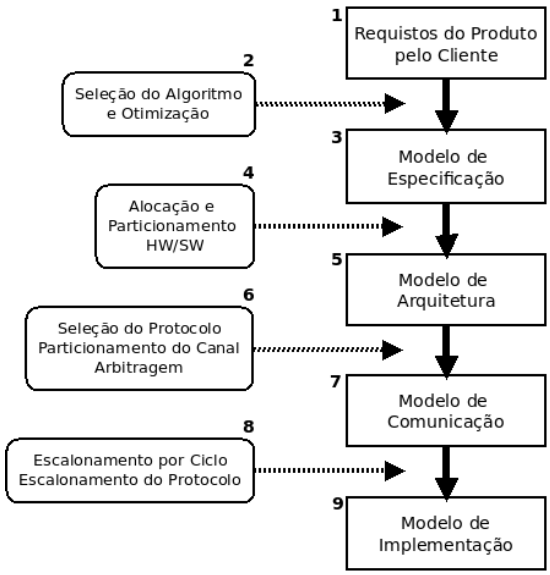


Figura 6 - Fluxo de projeto de sistemas embarcados.

Conforme mostrado no fluxo de projeto apresentado na Figura 6, o cliente fornece os

requisitos (1), os projetistas selecionam e executam otimizações baseadas nos requisitos fornecidos (2) e criam uma especificação funcional, que é geralmente um modelo de implementação em C/C++ que implementa apenas a funcionalidade do sistema (3). Em seguida, os projetistas fazem o particionamento da aplicação em hardware e software, alocando funcionalidades aos componentes de software em processadores de propósito geral e hardware (4) para criar um modelo arquitetural (5). Em seguida, o protocolo de comunicação a ser utilizado entre os componentes é selecionado, canais de comunicação são definidos e são selecionados os esquemas de arbitragem para resolver contenção em canais compartilhados (6), surgindo assim o modelo de comunicação (7). Finalmente, o comportamento dos componentes e dos canais são especificados considerando as restrições de desempenho (8), para se obter um modelo de implementação com precisão de ciclo (9). Este modelo pode ser especificado em nível abstração RTL, ou em outro nível de abstração que possa ser traduzido para este, para que, finalmente, seja realizada a síntese.

O foco do trabalho apresentado em [25] se concentra nas fases 5 à 7, onde os projetistas, geralmente, exploram o espaço de comunicação. A exploração envolve modelagem e análise dos fluxos de comunicação para se determinar a arquitetura de comunicação adequada (que neste caso é baseada em barramentos). A seleção da arquitetura de comunicação em um fluxo de projeto geralmente ocorre depois que o projetista fez o particionamento do hardware e software e o mapeamento da funcionalidade para plataforma. A seleção da arquitetura de comunicação, bem como as configurações de seus parâmetros, de forma a satisfazer os requisitos de projeto, e testes muitas vezes tornam-se impraticáveis, dado o enorme conjunto de possibilidades a ser considerado.

Assim, para minimizar este efeito foi proposta uma abordagem para modelagem de transações em alto nível de abstração, chamada Contagem de Ciclos com Precisão nos Limites de Transação (*Cycle Count Accurate at Transaction Boundaries* - CCATB). Este modelo é similar ao modelo TLM [54], exceto que as informações de tempo e controle, específicos do protocolo de um barramento, são incluídas na transação. Segundos os autores, a estimativa da quantidade de ciclos para uma transferência completa no barramento é equivalente aos ciclos decorridos em um modelo de comunicação com precisão de ciclos. A contagem de ciclos sendo exata, permite gerar estatísticas para a exploração exata do espaço de projeto, ao mesmo tempo em que se obtém um melhor desempenho das simulações.

SystemC 2.0 foi a linguagem de descrição escolhida para especificar o modelo CCATB. Barramentos baseados no modelo CCATB são especificados a partir da extensão dos canais de

comunicação SystemC TLM para incluir os detalhes de protocolo e estimativas de temporização específicos da arquitetura de comunicação. Os blocos computacionais (mestres e escravos do barramento) são modelados seguindo o padrão SystemC TLM.

Resultados experimentais mostraram que, utilizando o modelo transacional CCATB para especificar um barramento, foi possível duplicar o desempenho das simulações quando comparadas às com modelos de barramento com precisão de ciclo. Ainda segundo os autores, a precisão das estimativas é comparável a de um modelo de barramento com precisão de ciclo. Porém, os estudos de caso apresentados consideraram apenas espaços de projeto cujas configurações de barramento não oferecem impacto no processo proposto de estimação de latência de uma transação. Assim, para comprovação da precisão das estimativas seria necessário considerar, por exemplo, tipos diferentes de transferências, como por rajada e/ou divisão, ou até mesmo outros modelos de barramento, com diferentes protocolos de *handshake*.

3.2.5 Plataforma multiprocessadora flexível com exploração eficiente de intercomunicação para otimização do desempenho do sistema para uma aplicação específica

Na abordagem apresentada em [53] é proposto um modelo de plataforma MPSoC o qual, segundos os autores, suporta a avaliação rápida e precisa do desempenho para uma dada aplicação, com diferentes estruturas de comunicação. O modelo proposto suporta avaliar a melhor estrutura de comunicação, inicialmente, através da seleção da arquitetura e, em seguida, por variação da configuração dos respectivos parâmetros. Na versão atual, são suportados três arquiteturas: barramento Amba AHB, NoC Octagon e um servidor de memória distribuída [77]. Todos esses componentes são descritos em SystemC com precisão de ciclo.

Para simplificar a troca de estruturas de comunicação da plataforma, foi proposto um módulo de interface, entre componentes de plataforma e estrutura de comunicação, capaz de traduzir as primitivas de comunicação da aplicação em transações de comunicação definidas por cada protocolo. Um adaptador é um módulo capaz de traduzir endereços e formatar informação para corresponder ao canal de comunicação.

O fluxo de projeto inicia com a decomposição da aplicação em módulos de software executando em simuladores de instrução (*Instruction Set Simulators*- ISS) tradicionais e módulos de hardware, representados por modelos de geração de tráfego. Nesse trabalho, os módulos

geradores de tráfego, denominação dada aos mestres de barramento, também foram codificados em SystemC. O tráfego de comunicação é anotado a partir de instruções de sincronização de eventos, que em SystemC consistem de instruções do tipo *wait()* (ver Apêndice A).

A avaliação de desempenho da plataforma é realizada após co-simulação com registro de tráfego de comunicação. As métricas analisadas com esta abordagem são geradas a partir do tráfego anotado e consistem das latências de comunicação, taxas de transferências e tempo de execução da aplicação. Se a estrutura de interconexão atender a todas as restrições de projeto, o sistema será então implementado em hardware. De outra forma, os parâmetros serão alterados ou outra estrutura de comunicação poderá ser escolhida.

Assim como no trabalho apresentado em [50], esse trabalho endereçou três principais métricas para suporte à análise de comunicação, que são: desempenho de simulação, precisão de estimativas de desempenho e flexibilidade. Entre as vantagens do uso dessa abordagem, destaca-se a última, por suportar construção de espaços de projeto maiores, com três arquiteturas de comunicação diferentes, além de simplificar a troca e configuração da estrutura de comunicação.

Para aumentar o desempenho do processo de exploração, a abordagem apresentada em [53], se baseou na geração automática de interfaces para adaptação dos componentes da plataforma à estrutura de comunicação.

Por incluir modelos com precisão de ciclo para a comunicação, os simulações resultam em estimativas com precisão. Por outro lado, na prática, o processo de registro de eventos de comunicação, em modelos descritos com precisão de ciclo, pode implicar no aumento do tempo de simulação do sistema [11][52].

3.3 Estimativas Estáticas

A segunda categoria de trabalhos relacionados agrupa técnicas baseadas em estimação estática [60][61][62]. Tais trabalhos fazem uso de modelos estáticos da latência da comunicação entre componentes do sistema ou caracterizações estáticas do consumo de potência das tarefas do sistema. Estas técnicas frequentemente se aplicam aos cenários onde computações e comunicações podem ser estaticamente escalonadas.

A seguir serão apresentados trabalhos que abordam este tipo de técnica.

3.3.1 Estimação de desempenho da arquitetura de comunicação por agendamento de tarefas para exploração do espaço de projeto

Em [60] é proposta uma abordagem de exploração de espaços de projeto que utiliza como entrada um modelo de agendamento das tarefas para cada elemento de processamento, fornecido pelo projetista, e um trace de acesso à memória, obtido por simulação.

O fluxo de projeto da abordagem proposta é iterativo e inicia com uma única plataforma, que inclui os elementos de processamento, que podem ser processadores ou hardware especializado, com as respectivas tarefas previamente alocadas. Adicionalmente, a plataforma inclui uma memória compartilhada, que é dividida em blocos lógicos locais e compartilhados, e um modelo de barramento. Esta plataforma é, neste momento, a única candidata à solução e seria também a melhor arquitetura encontrada da iteração anterior, se houvesse.

Em seguida, é iniciada uma nova iteração do fluxo e, a partir da melhor arquitetura da iteração anterior, que é a plataforma inicial, o espaço de projeto é explorado de forma incremental através da seleção de um de seus elementos de processamento. O elemento selecionado deverá ser alocado para outro, ou um novo, barramento, gerando assim novas plataformas candidatas à melhor solução.

Às plataformas candidatas, é aplicada uma técnica de análise estática para selecionar as melhores, podando assim o espaço de projeto. Esta técnica utiliza como base o modelo de enfileiramento apresentado em [78], o qual cria uma fila de requisições a partir da atribuição de prioridades, através de agendamento de tarefas predeterminado, às requisições de uso do barramento SCSI pelos mestres. Para a abordagem proposta em [60], o modelo de enfileiramento, apresentado em [78], foi estendido, de forma a suportar mecanismos de arbitragem por prioridade fixa, *round robin* e TDMA.

Assim, uma vez que o escalonamento dos blocos funcionais foi predefinido, o padrão de requisições para uso do barramento pode agora ser determinado estaticamente. Isto é realizado através da divisão do agendamento em fatias de tempo de tal forma que todos os elementos de processamento mantêm seus padrões de requisição do barramento durante cada fatia de tempo. Então, é aplicada a técnica de enfileiramento em cada fatia de tempo, iniciando a partir do começo do agendamento. A Figura 7 mostra um exemplo desta abordagem.

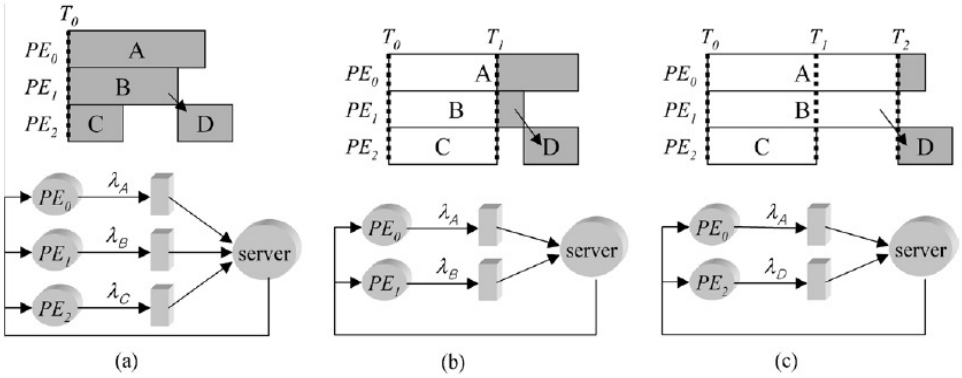


Figura 7 - Exemplo de enfileiramento de blocos funcionais.

Durante a primeira fatia de tempo, três blocos funcionais *A*, *B* e *C* são executados concorrentemente nos elementos de processamento PE_0, PE_1 e PE_2, respectivamente. Para avaliar o tempo esperado de atraso para acessar o barramento por cada elemento de processamento, um sistema de enfileiramento é construído, como pode ser visto na Figura 7(a). A partir do sistema de enfileiramento proposto, é computado o retardo no escalonamento inicial dos blocos funcionais devido à contenção de barramento. Supondo que o bloco *C*, em PE_2, é finalizado primeiro no instante T_1, então uma próxima fatia de tempo será considerada, onde os blocos A e B estão acessando o barramento, como é mostrado na Figura 7(b), e construído um novo modelo de enfileiramento com PE_0 e PE_1. Essa fatia de tempo será dimensionada até que *B* seja completada no instante T_2. Depois de T_2, outro modelo de enfileiramento com PE_0 e PE_2 é criado para avaliação do restante do agendamento. Este processo de avaliação é repetido até que o modelo de enfileiramento examine todos os escalonamentos dos blocos funcionais.

Para ilustrar o processo de exploração de comunicação com a técnica proposta, foi utilizado um estudo de caso que consistiu de uma aplicação de gravação de vídeo digital com quatro canais. Utilizou-se o algoritmo de codificação H.263 para cada canal. Esse algoritmo teve suas funções distribuídas entre um processador ARM720T, um componente de hardware para estimação de movimento e um componente de hardware para transformação cosseno-discreto. Os componentes de hardware dedicado foram compartilhados entre os quatro codificadores H.263. Assim, ao todos foram utilizados quatro processadores ARM720T e dois componentes de hardware dedicados. Com esses componentes, várias arquiteturas foram geradas a partir do aumento do número de barramentos, da alocação dos elementos de processamento para o barramentos, da alocação dos seguimentos de memória compartilhada e das conexões da pontes entre os barramentos. Para o modelo de agendamento de tarefas

utilizou-se três esquemas diferentes de arbitragem, que foram: por prioridade fixa, round robin e TDMA. Com essas configurações, foram exploradas mais de 200 arquiteturas para cada esquema de arbitragem.

Os resultados mostraram que as estimativas de desempenho fornecidas pelo modelo de agendamento, comparadas às de modelos de simulação, obtiveram erros variando entre 6% e 8,5%. Segundo os autores, a técnica proposta possui alto desempenho, levando em torno de 1,5 segundos para estimar o desempenho da aplicação para cada configuração de barramento. Esse tempo é pode ser considerado bastante significativo, uma vez que os modelos de simulação levam em torno de 5 minutos para cada simulação.

Apesar dos bons resultados apresentados, essa técnica é bastante limitada por considerar que o escalonamento dos blocos funcionais deve ser conhecido à priori. O fato de não considerar os efeitos dinâmicos na comunicação entre os componentes da plataforma, pode levar a estimativas com baixa precisão em cenários mais realísticos. Além disso, as arquiteturas utilizadas não incluem características presentes no estado da arte das arquiteturas de barramentos, como, por exemplo, tipos variados de transferências e de barramentos.

Desde que o objetivo da técnica de análise proposta em [60] é reduzir o espaço de projeto, os autores justificam que uma quantidade razoável de imprecisão nas estimativas pode ser tolerável para acelerar o processo de exploração.

3.3.2 Modelagem e análise da latência do barramento de sistema em plataformas SoC

Em [61], é apresentado um modelo estático de latência para um barramento compartilhado (AMBA AHB) conectado a vários componentes de plataforma. Utilizando este modelo é possível analisar as latências de comunicação no barramento on-chip para obter a taxa de transferência necessária. Este resultado é utilizado como critério para configurar a arquitetura de comunicação em um projeto de SoC.

Para tanto, foi proposto um modelo de latência de barramentos que considera uso de escravos ideais, ou seja, que durante as transferências consegue atender todas as requisições sem adicionar, à comunicação, estados de espera. Assim, o modelo de latência proposto é dado pela equação:

$$L_{bus} = 1 + N_D \tag{1}$$

onde N_D é o número de dados que serão transmitidos e *1* indica, de acordo com o protocolo, o ciclo necessário para o mestre solicitar e ter permissão para realizar operações no barramento. Essa permissão é dada pelo árbitro.

Para tratar transferências, de forma eficiente, o AMBA AHB possui dois tipos de transferências: simples e rajada. Estes tipos possuem latências diferentes para transferências de dados. O modo de transferência simples utiliza três ciclos de relógio, que inclui um ciclo para as fase de requisição, um ciclo para controle e dado, e um ciclo para transferir a informação, Já o modo de rajada necessita de, pelo menos, um ciclo a mais do que o modo simples, para realizar uma transferência. No entanto, a transferência em rajada possui o recurso de pipeline para tratar as fases de endereçamento e dado no mesmo ciclo, tornando a transferência mais eficiente.

Por estas razões a equação inicial de latência foi reescrita, considerando agora os efeitos do modo de transferência e arquitetura de pipeline:

$$L_{bus} = 3.N_D.S + \left\{ \left\lceil \frac{N_D.(1-S)}{B} + N_D.(1-S) \right\rceil \right\} \tag{2}$$

onde o valor *S*, que varia entre zero e um, sendo igual a zero para taxa de transferência no modo simples, e *B* representa a quantidade de transferências que formam a rajada. A constante *S* é parametrizável, pois a taxa de transferência depende das características do sistema. O primeiro termo, $3.N_D.S$ é a latência de uma transferência simples. O segundo termo, entre chaves, representa a latência da transferência em rajada, onde o número de ciclos de requisição é dividido pelo tamanho da rajada.

Até agora, os modelos apresentados só estimam a latência do barramento com um único mestre. Para estimar a latência quando se tem mais de um mestre em um único barramento, tem-se a equação dada por:

$$L_{SingleLayer} = N_M . L_{bus} \tag{3}$$

onde N_M representa o número de mestres. Com a arquitetura de *pipeline*, as fases de controle e dado, das transferências de cada mestre, irão ocupar o barramento simultaneamente. Assim, a equação, anterior foi reescrita como a seguir:

$$L_{bus} = (3 - 2.U).N_D.S + \left\{ \left\lceil \frac{N_D.(1-S)}{B} + N_D.(1-S) \right\rceil \right\} \quad (4)$$

onde o valor de U varia entre 0 e 1, e representa a probabilidade de uso do barramento, ou seja, a probabilidade das transferências simples continuarem.

As equações apresentadas anteriormente calculam as latências para uma arquitetura de comunicação com apenas um nível de barramento. Quando consideramos hierarquia de barramentos, é necessário utilizar o uso de componentes do tipo ponte para que os mestres possam ter acesso a escravos em níveis diferentes da hierarquia. Assim, para modelagem de hierarquia, mais parâmetros devem ser considerados. Inicialmente, com o aumento da quantidade de níveis em uma hierarquia, o número de componentes de plataformas em um único nível é reduzido. Em consequência, a taxa de transmissão de dados é aumentada. Segundo, a latência é aumentada quando há comunicação entre componentes de níveis diferentes, que é realizada através de pontes. Neste caso, um componente torna-se mestre nos dois níveis. Com isso, a taxa total de transferência de dados entre dois níveis consecutivos torna-se igual à taxa de apenas um nível.

A equação a seguir define a latência de um barramento compartilhado com mais de um nível de hierarquia, considerando as condições citadas anteriormente. A latência é reduzida com o aumento de N_L que é o número de camadas.

$$L_{MultiLayer} = \frac{N_D}{N_L}.L_{BusComplex}.(1-A) + \alpha.A \quad (5)$$

onde o valor de A varia entre 0 e 1, e representa a probabilidade de ocorrer um caminho de dados através de pontes. A equação é dividida em dois termos: o primeiro fornece a latência sem uso de ponte e, o segundo, a latência com uso de pontes. O fator α representa as latências causadas por utilizar pontes. Para definir o valor de α é necessário obter uma relação entre o número de pontes e o número de níveis utilizados, pois as latências irão depender do número de pontes ativas. O fator α é expresso pela equação a seguir:

$$\alpha = \sum_{i=0}^{N_L - 1} \left(\frac{\beta}{\chi}. \right) \frac{N_M}{N_L - i}.L_{bus} \quad (6)$$

Os fatores β e χ são definidos, respectivamente, pelas seguintes equações:

$$\beta = C_j^{N_L-1} \tag{7}$$

$$\chi = \sum_{j=1}^{N_L-1} C_j^{N_L-1} \tag{8}$$

onde β representa o número de caminhos de dados que utilizam o mesmo número de pontes e χ o total de caminhos de dados que utilizam pontes.

Ainda em [61], é apresentado um estudo comparativo com modelos de simulação com precisão de ciclo para avaliação da precisão das estimativas de desempenho fornecidas pelos modelos estáticos do barramento AMBA AHB. Esse estudo incluiu cenários com topologias contendo de um até três níveis de barramento e número de mestres variando entre um e oito. Para os modelos de simulação, foram considerados os mesmos parâmetros de configuração do barramento AMBA AHB oferecidos pelos modelos estáticos.

Os resultados mostraram que, para o cenário com apenas um nível de barramento, os modelos estáticos forneceram estimativas precisas, com erros relativos em torno de 4%. Com o aumento do número de níveis na hierarquia de barramento, os erros aumentaram para 14%. Segundo os autores, mesmo com esse aumento, esses índices são aceitáveis para estimação de desempenho de um barramento, podendo-se utilizar os modelos estáticos propostos para exploração de comunicação em estágios iniciais do projeto.

Entretanto, por não considerar os efeitos dinâmicos de comunicação, como latências de estados de espera, inseridas por escravos, e de contenção de barramento, haverá a redução da precisão das estimativas dadas pelos modelos propostos. De acordo com os resultados experimentais apresentados, a redução da precisão é acentuada com o uso de maior número de mestres e níveis de barramento nas plataformas, comprometendo a escalabilidade da técnica.

3.3.3 Estimativa de comunicação para hardware/software co-design

O trabalho publicado em [62] apresenta uma abordagem para estimação de taxas de transferência para a implementação de um determinado protocolo de comunicação utilizando o sistema de co-

síntese LYCOS [83]. Este sistema possui uma biblioteca de componentes de comunicação em alto nível de abstração que, para cada unidade de processamento e para cada protocolo suportado, captura desempenho, área, custo e outras características necessárias aos drivers e canal de comunicação.

O modelo de comunicação adotado suporta comunicação ponto a ponto e unidirecional [62]. A partir desse modelo de comunicação foram formulados modelos estáticos, com suas respectivas variáveis de configuração, com o objetivo de estimar as latências de transmissão. O primeiro elemento, o driver de software, recebe como entrada n_t palavras para transmissão e retorna n_c palavras de canal de comunicação. Com a frequência de transmissão do processador f_t e número de ciclos c_{tc} necessários para acionar o driver para transmissão e o número de ciclos necessários c_{tp} para processar (empacotar, dividir, etc.) cada palavra, a latência total de transmissão pode ser estimada por:

$$t_{td} = \frac{(c_{tc} + c_{tp} n_t)}{f_t} \tag{9}$$

A latência do canal de transmissão pode ser dada por:

$$t_{cd} = \frac{(c_{cs} + c_{ct} n_c)}{f_c} \tag{10}$$

onde c_{cs} representa a quantidade de ciclos necessários para sincronização, f_c a frequência de operação do canal e c_{ct} os ciclos necessários para transmitir uma palavra de canal.

Já para o driver de recepção, a equação de estimativas pode ser dada por:

$$t_{rd} = \frac{(c_{rs} + c_{rp} n_t)}{f_r} \tag{11}$$

onde c_{rs} é o número de chamadas do driver para recepção, c_{cs} o número de ciclos para processamento de cada palavra de canal recebida e f_r a frequência de operação do processador de recepção.

Para estimativa de latências totais para comunicação, assumiu-se que os elementos de comunicação descritos anteriormente atuam de forma paralela, de forma que a maior latência

dentre os três elementos irá determinar a latência total de comunicação t_t:

$$t_m = \max(t_{td}, t_{cd}, t_{rd}) \tag{12}$$

Assim, o cálculo da latência total de transmissão será dado por:

$$t_t = t_m + 2\frac{t_m}{n_t} \tag{13}$$

onde o último termo é uma aproximação das latências de início e conclusão do *pipeline*.

Knudsen e Madsen, ainda em [62], estenderam as equações anteriores para formalizar transferências por rajada. Para suportar transferências por rajada, o número de palavras de canal de comunicação é modelado como:

$$n_c = (n_b - 1)s_b + s_r \tag{14}$$

onde $(n_b\text{-}1)$ é número de rajadas, s_b é o tamanho da rajada e s_r é o tamanho restante da rajada, sendo $0 < s_r < s_b$. Denotando b_m como um dos três tipos de rajada suportados: *fixo* (com tamanho fixo), *max* (rajadas com tamanhos variados, mas existe um máximo) e *inf* (não há limite para o tamanho da rajada), pode-se calcular n_b e s_b conforme as equações a seguir:

$$n_b = \begin{cases} 1 & \because \quad b_m = \inf \\ \lceil n_{cd} / s_b \rceil & \because \quad b_m = fixo, \max \end{cases} \tag{15}$$

$$s_r = \begin{cases} s_b & \because \quad b_m = fixo \\ n_{cd} - (n_b - 1)s_b & \because \quad b_m = \max, \inf \end{cases} \tag{16}$$

onde n_{cd} é a quantidade atual de dados no canal correspondente às palavras de entrada do driver n_t.

Por fim, para calcular o número de ciclos necessários para sincronização de canal c_{cs}, utiliza-se os ciclos necessários para sincronização de uma rajada c_{sb} e o número de ciclos de sincronização por seção de transferência c_{ss} utilizando a equação a seguir:

$$c_{cs} = \lceil n_b c_{sb} \rceil + c_{ss} \tag{17}$$

Assim como nas abordagens apresentadas em [60] e [61], para estimação de desempenho

de comunicação, é necessário que o projetista conheça à priori a quantidade de transferências a serem realizadas na arquitetura de comunicação. Outro aspecto a ser considerado refere-se a ausência de suporte à arbitragem, uma vez que as estimativas são dadas apenas para um mestre de barramento com comunicação ponto a ponto. Como discutido anteriormente, por não permitir captar as efeitos dinâmicos da comunicação, o uso deste tipo de técnica em cenários reais pode trazer estimativas imprecisas.

3.4 Abordagens Híbridas

As técnicas consideradas híbridas [12][51][63][68][69][70][71] combinam simulação do sistema com alguma outra técnica, como por exemplo, análise estática ou heurísticas, para estimar o desempenho do sistema sob efeitos de configurações da comunicação. O objetivo é tornar, a partir dessa associação, os processos de exploração mais precisos e/ou mais eficientes, ou ainda estabelecer um compromisso entre precisão e eficiência.

Trabalhos da literatura científica combinam técnicas de simulação completa de sistema com técnicas de estimação estática, inteligência artificial, linguagens formais e, até mesmo, física quântica. Tais trabalhos são descritos a seguir.

3.4.1 Análise de desempenho ao nível de sistema para projeto de arquiteturas de comunicação on-chip

É uma metodologia de análise de desempenho, que utiliza modelos ao nível de sistema, para orientar o projeto da arquitetura de comunicação [12]. Esta técnica inclui fluxo de projeto que é baseado no uso de perfis de computação e comunicação da aplicação, que são gerados a partir de uma co-simulação inicial de hardware e software do sistema.

O modelo de co-simulação, que segue o modelo de projeto baseado em plataformas e com componentes descritos em nível de sistema, inclui um canal de comunicação abstrato, onde apenas eventos de comunicação e dados são transferidos entre os componentes da plataforma. O tempo necessário para gerar e consumir um evento de comunicação é dado apenas pelo tamanho da transferência e independe da quantidade de transferências concorrentes, de forma que o perfil de comunicação gerado ao fim da co-simulação não contém informações de ciclos de relógio.

A segunda fase, que se refere à análise de desempenho, envolve inicialmente a extração

de um grafo de análise de comunicação (GAC) do perfil de computação e comunicação obtido da co-simulação inicial. Em seguida, o projetista deve selecionar uma topologia para a estrutura de comunicação do sistema, que pode incluir canais de comunicação dedicados assim como canais de comunicação compartilhados (barramentos) conectados ou não por pontes. Além da topologia é possível configurar os parâmetros desses canais de comunicação, que incluem largura (em bits), frequência de operação (em MHz), tamanho da transferência de DMA (em número de palavras do canal) e latência (em ciclos de relógio).

Baseado nas configurações selecionadas pelo projetista, a ferramenta de análise manipula o GAC e gera um novo perfil de computação e comunicação do sistema, sendo que este novo incluirá informações de latências de transações com precisão de ciclo. O primeiro tipo de modificação do GAC adiciona latências às transações de forma que atendam aos requisitos temporais do protocolo de comunicação do barramento a ser estimado. Para tanto, é adicionado um novo vértice para cada vértice que representa uma transferência no GAC. Este novo vértice conterá uma estimava de tempo necessário para conclusão daquela transferência. Além disso, um vértice de comunicação, no GAC inicial, pode ser utilizado para representar mais de uma transferência, quando houver transferências seguidas de mesmo mestre no perfil de comunicação. Assim, na manipulação do GAC, serão adicionados novos vértices com as estimavas de latências referentes às transferências daquele grupo. Se forem consideradas transferências do tipo rajada, as estimativas de latência dos novos vértices terão de ser limitadas de acordo com o tamanho de uma rajada real especificado pelo projetista. Por exemplo, havendo um vértice de comunicação que representa 245 transferências, poderiam ser adicionados dois novos vértices com estimativas de latência: um com 128 ciclos (representando uma rajada completa de tamanho igual a 128, onde teríamos a transferência de uma palavra por ciclo) e outra com 117 ciclos (representa uma rajada incompleta com estimativas de latência para o restante das transferências).

Outros tipos de modificação no GAC incluem suporte à:

- Escalonamento de transações por arbitragem, neste caso, orientado por prioridade fixa. O mecanismo verifica o tempo de ativação dos vértices do GAC inicial. Caso haja vértices de mestres diferentes ativados no mesmo instante, o mestre de maior prioridade receberá logo atenção, adicionando um novo vértice com estimativa de latência para sua transferência. Já para o mestre de menor prioridade, o início da contagem de tempo para sua transação deverá incluir o tempo de atraso, dado a espera por contenção.

- Comunicação entre componentes em barramentos diferentes, através do uso de pontes. Serão adicionados novos vértices com as estimativas de latência necessárias para que uma transferência possa ser realizada entre componentes de barramentos diferentes em uma hierarquia. Será adicionado um novo vértice com latências para cada barramento envolvido na transferência.

Assim, após a manipulação do GAC, os resultados gerados podem ser:

- uma versão aumentada do GAC, que incorpora informações de latências, desde que o novo modelo de comunicação é mais detalhado;

- uma estimativa do desempenho de todo o sistema;

- os caminhos críticos de comunicação, como uma sequência de comunicações e computações;

- e estatísticas básicas de execução, como uso do barramento, conflitos e a proporção ocupada dos caminhos críticos por cada componente.

O processo de exploração de comunicação se dá pela extensão do GAC inicial para cada configuração do espaço de projeto. Estudos de caso mostraram que a técnica proposta em [12] é eficiente, com duas ou três ordens de magnitude mais rápida do que simulação do sistema. Porém, em espaços de projeto maiores, a aplicabilidade da técnica para exploração de comunicação pode ser comprometida, um vez ser necessário analisar cada configuração individualmente. Cada GAC estendido deverá ser analisado, através das métricas disponibilizadas ao fim do processo de incorporação das configurações de barramento, para verificar se as restrições de projeto foram atendidas. Resultados experimentais mostraram ainda que, por considerar os efeitos dinâmicos anotados no perfil de comunicação, foi possível obter estimativas precisas de desempenho, apresentando erros inferiores à 3%.

3.4.2 Exploração rápida de arquitetura de barramento parametrizável para projeto de SoC com comunicação centralizada

O trabalho em [51] propõe uma ferramenta, chamada Configurador de Barramento Automático (*Automatic Bus Configurator – ABC*), para exploração de espaços de projeto de comunicação, através do uso de algoritmos genéticos e computação distribuída.

O fluxo de projeto proposto inicia com a criação de uma população de indivíduos para a geração inicial do algoritmo genético. Cada característica genética representa um parâmetro de configuração da estrutura de comunicação. O AG cria os indivíduos através de aplicações das operações básicas de cruzamento e mutação. Uma vez que a população é formada, as configurações dos indivíduos serão utilizadas para compor modelos de simulação.

Para cada indivíduo, um arquivo de configuração é gerado baseado nos parâmetros utilizados. O arquivo então é utilizado para geração de códigos SystemC RTL. O modelo RTL é compilado e simulado para produzir arquivos com os resultados das simulações. O mecanismo de simulação proposto busca, através de computação distribuída, executar paralelamente os simuladores, com as respectivas configurações dos indivíduos genéticos.

Estes arquivos são então analisados para computar o valor da função de ajuste para cada indivíduo, cujo resultado é guardado em um novo arquivo. Foram utilizadas duas funções de ajuste: a primeira verifica a taxa de transferência de cada mestre e, a segunda, a largura de banda e latências totais.

Para aumentar o desempenho das simulações RTL, cada mestre é modelado como um modelo abstrato funcional, de forma que sua simulação é de várias ordens de magnitude mais rápida que um modelo RTL puro.

Caso os resultados da aplicação das funções de ajuste não sejam satisfatórios, uma nova população de configurações candidatas à solução será gerada a partir da população atual, através do mecanismo de evolução, e o algoritmo continuará executando até atingir os parâmetros de parada.

Essa abordagem buscou o aumento de desempenho do processo de exploração, que é baseado em simulações do sistema, através de algoritmos evolutivos e paralelização de simulações. O uso de algoritmo evolutivos permitiu automatizar os processos de seleção e avaliação de configurações de barramento. Essa avaliação é favorecida pelo uso de um sistema distribuído, de forma que os candidatos à solução final podem ser avaliados paralelamente. Contudo, a eficiência da paralelização é limitada ao número de candidatos à solução em cada geração do algoritmo evolutivo e ao número de nós de processamento no sistema distribuído. Além disso, o desempenho para avaliação dos candidatos de uma geração é limitado pelo maior tempo de simulação de uns dos indivíduos e pelo maior tempo de resposta dos nós do sistema distribuído.

3.4.3 Avaliação formal de desempenho de projetos de SoCs baseados no AMBA

Madl et al. propôs em [63] um método para análise de desempenho e avaliação de projetos de sistemas embarcados baseados no barramento AMBA AHB, através de métodos formais, o que permite, adicionalmente, garantir a corretude funcional do protocolo de comunicação.

Para tanto, um método de análise baseada em modelos formais foi desenvolvido. Este método utiliza simulação e verificação formal como técnicas complementares para análise de desempenho.

O fluxo de projeto da abordagem proposta inicia com um modelo específico de domínio (MED), que é uma especificação em alto nível das propriedades-chave do projeto, como sua estrutura, comportamento, ambiente e restrições que devem ser atendidas. De acordo com os autores, esse modelo pode ser expresso de várias formas, que incluem especificação textual, diagramas de tempo, meta modelos ou até mesmo outros tipos de modelos visuais.

Seguindo o fluxo de projeto proposto, o partir deste MED inicial, são gerados, em paralelo, modelos de simulação e modelos de análise formal do sistema. Os modelos formais foram adotados porque usualmente capturam propriedades importantes do sistema em níveis altos de abstração, comparados ao próprio MED. Da mesma forma acontece com os modelos de simulações, onde suas abstrações influenciam a complexidade da análise assim como sua precisão. Se o modelo é muito abstrato, os resultados podem ser imprecisos e se for muito complexo, o processo de análise cairá no problema da explosão do espaço de estados. A justificativa para a combinação de ambos os modelos (simulação e formais), é que o primeiro representa as características do modelo do sistema com precisão, enquanto o outro é adaptado para a verificação das propriedades.

Desta forma, as informações extraídas a partir dos modelos de simulação serão parâmetros importantes para checagem formal das propriedades do sistema. Esta checagem é dividida em duas partes: 1) verificação funcional da aplicação, para identificar situações de *deadlock* ou *livelock* [42] e 2) análise de desempenho, que identifica o limite dos piores tempos de execução do sistema.

O modelo de comunicação dos componentes de hardware do sistema proposto, na abordagem apresentada em [63], é baseado em máquinas de estados finitos (MEF), que representam o protocolo do barramento AMBA AHB. Este modelo foi escolhido principalmente porque é suportado por diversas ferramentas de checagem de propriedades formais.

Assim, temos como parâmetros para os modelos formais, as informações extraídas dos modelos de simulação que incluem os melhores e piores tempos de execução de cada mestre do barramento. Estes parâmetros de execução são obtidos independentemente para cada componente, e não incluem os tempos gastos na espera para utilizar o barramento ou durante a comunicação, referem-se exclusivamente ao tempo de computação. Outro aspecto importante, considerado na abordagem, é o tamanho de cada transação. Este tamanho corresponde ao tamanho das mensagens de leitura e escrita nos escravos.

O modelo de simulação captura o protocolo do barramento no nível transacional com precisão de ciclo, utilizando a biblioteca SystemC TLM. Este modelo permite alta precisão das estimativas com contagem de ciclo, pelo barramento, enquanto permite simulações e geração de perfil do comportamento aproximado dos componentes conectados ao barramento.

Por fim, na última etapa do fluxo de análise, o módulo de checagem formal explora exaustivamente todos os possíveis acessos ao barramento do sistema, através do MEF. Este método permite obter os limites do pior caso, que é o maior tempo de execução do sistema, sob aquele modelo de barramento como comunicação entre componentes do sistema, com alta precisão, e garantias formais, provando que o tempo de computação do sistema está entre as restrições predefinidas. Assim, o módulo de checagem formal generaliza os resultados das simulações para prever o comportamento do sistema no pior caso.

De acordo com os autores, essa abordagem de exploração tem alto custo computacional e o desempenho sofre degradação exponencial em relação ao espaço de estado do MEF do sistema analisado. Com isso pode apresentar problemas de escalabilidade quando aplicada ao projeto de sistemas mais complexos.

3.4.4 Exploração eficiente de arquiteturas de barramento on-chip e alocação de memória

Kim et al. propôs em [68] uma técnica de exploração de espaços de projeto de comunicação, que utiliza uma técnica de estimação estática, apresentada em [60], para podar rapidamente grande parte do espaço de projeto e, em seguida, simulações com precisão de ciclo baseadas em geração de perfis para também reduzir o conjunto de candidatos.

Inicialmente, o comportamento do sistema deve ser especificado em blocos funcionais. Estes blocos contêm partes do sistema que devem realizar algum tipo de computação. Além disso, é possível também especificar as dependências, ou precedências, de execução entre

módulos. Estas dependências podem ser interpretadas como comunicação ou sincronização entre os blocos funcionais.

Em seguida, esses blocos são mapeados em elementos físicos de processamento, que podem ser, por exemplo, processadores ou DSPs. A comunicação dos blocos funcionais é mapeada para uma arquitetura de comunicação composta de apenas um barramento e uma memória. Esta memória pode ser dividida em segmentos lógicos do tipo local ou do tipo compartilhado.

A partir da arquitetura inicial são gerados novos candidatos através de alocação de elementos de processamento bem como suas memórias locais e memórias compartilhadas em novos barramentos. Com isto, os novos candidatos terão diferentes topologias de barramentos e disposições de componentes de plataforma. Em seguida, obtidas as novas alternativas do espaço de projeto, o passo seguinte é definir as prioridades dos mestres do barramento.

Através de busca exaustiva é possível encontrar uma configuração ótima, mas é muito dispendioso. Assim, uma heurística gulosa de atribuição de prioridade é utilizada, onde a maior prioridade é atribuída ao elemento de processamento mais crítico e com mais acessos à memória. A equação de atribuição de prioridade é dada a seguir:

$$R(PE) = \sum_{fb_i \in FB_{PE}} R(fb_i) = \sum_{fb_i \in FB_{PE}} \{BW(fb_i).C(fb_i)\} \quad (18)$$

onde $BW(fb_i)$ representa a quantidade de dados transferidos por unidade de tempo (taxa de transferência) do bloco funcional fb_i, $C(fb_i)$ a criticalidade de fb_i, que é a soma do tamanho do escalonamento dos blocos funcionais no maior caminho de execução daqueles que podem ser executados somente depois de fb_i. A taxa de transferência e a criticalidade são obtidos a partir de traces de memória, que são obtidos a partir de ISS, e de agendamento dos blocos funcionais, respectivamente. $R(fb_i)$ representa uma medida de posição (*rank*) do bloco funcional fb_i como o produto da criticalidade e largura de banda. $R(PE)$ é a soma das posições dos blocos funcionais mapeados no elemento de processamento PE. Com isso a maior posição dos elementos de processamento é eleita para ter a maior prioridade. É possível, em nível de investigação, trocar as prioridades de elementos de processamento no mesmo barramento, mas isso pode exigir bastante tempo de exploração.

A etapa seguinte consiste na aplicação do algoritmo de estimação de desempenho estático. Esta técnica é baseada no modelo de enfileiramento do sistema, apresentado em [60], onde os elementos de processamento são os consumidores e o barramento com sua memória

associada é um servidor simples. A taxa de requisição de serviços de cada elemento de processamento é extraída dos traces de memória como uma função do tempo de execução. O método considera o efeito da contenção do barramento, dando resultados de estimação com precisão suficiente para serem utilizados numa primeira redução do espaço de configurações. São selecionadas para a próxima etapa apenas 10% das arquiteturas do espaço (visto que a técnica de estimação estática tem 10% de erro em relação ao resultado de uma simulação precisa).

Em seguida, as arquiteturas restantes são simuladas com extração de traces. Agora é possível avaliar precisamente o desempenho do espaço de configurações restantes e identificar a melhor configuração. Se o desempenho da melhor arquitetura não é melhorado em relação a interação anterior, o loop de exploração é finalizado. Senão, uma nova iteração inicia-se.

A melhor configuração irá sofrer novas alocações de elementos de processamento em novos barramentos. Com as novas arquiteturas serão atribuídas as prioridades, aplicada a técnica de estimação estática de desempenho, simulações com extração de traces e seleção de melhor arquitetura e assim por diante.

Duas principais vantagens no uso dessa abordagem para exploração de comunicação podem ser destacadas. A primeira consiste na possibilidade de se utilizar espaços de projeto maiores, desde que pode considerar diferentes números de barramentos, topologias, alocação de componentes, atribuições de prioridades aos mestres, além de outras condições de operação [68]. Os autores afirmam que o método é flexível o suficiente para que novos parâmetros de configuração de barramento possam ser adicionados para compor espaços ainda maiores. A outra vantagem refere-se ao desempenho de exploração. Nos estudos de caso apresentados em [68], os resultados mostraram que utilizando as heurísticas propostas pode-se reduzir, em média, o espaço de projeto em 181 vezes. Contudo, observa-se que, por considerar arquiteturas que apresentaram erros de precisão em até 10% para geração de novas arquiteturas derivadas, a busca pela melhor configuração pode cair em um máximo local.

3.4.5 Avaliação de desempenho para arquiteturas SoC utilizando simulação em nível transacional baseada em traces

A abordagem, apresentada em [69], utiliza um modelo de simulação em nível de sistema de SoCs, onde seus componentes (também chamados de recursos) são especificados em SystemC TLM e podem variar entre: processadores RISC, hardware para tarefas de processamento

específicas e blocos de memória (SRAM ou controladores de memória para RAM externa ao SoC). Além destes, um gerenciador de *buffer* responsável por armazenar e recuperar pacotes de tamanhos variados na memória e um gerenciador de fila que administra as filas de saída que são anexadas ao barramento.

O princípio básico da técnica de avaliação de desempenho baseia-se na abstração das funcionalidades e cobertura apenas da interação das subfunções associados na arquitetura, representadas por transações entre componentes do SoC. Esta abstração permite aumentar o desempenho das simulações. Cada subfunção é capturada como uma sequência de transações, também referenciada como *trace*. Os *traces* são gerados e armazenados pelos componentes do SoC, que podem conter vários *traces*, um por cada subfunção. Estes *traces* serão então utilizados de forma conjunta para estimar o desempenho total da aplicação, sob as configurações predeterminadas para componentes da plataforma.

Cada *trace* é formado por uma sequência de tarefas de execução que são intercaladas com transações, representando a comunicação com outros *traces* de diferentes recursos. Para aumentar o desempenho da simulação, uma tarefa de processamento é denotada apenas pela sua latência de execução, em cada um dos recursos. Em recursos compartilhados, os *traces* são combinados, sobrepondo as transações e coletando informações sobre utilização, de forma que é possível ter uma avaliação precisa do desempenho da arquitetura. A Figura 8 mostra um exemplo básico de *trace* para uma CPU que executa algum processamento de pacotes.

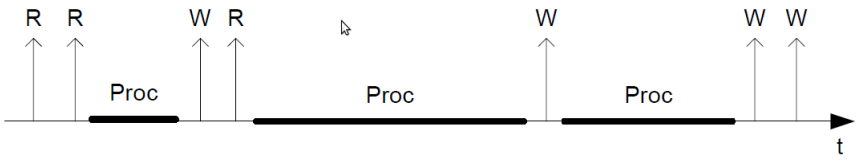


Figura 8 - *Trace* obtido de CPU a partir de simulação de SoC em nível de sistema.

O *trace* da Figura 8 inicia com duas operações de leitura da memória, representadas por duas entradas “R”, para obter a referência e o cabeçalho do pacote. Depois de certo tempo de processamento do pacote, uma pesquisa é realizada, representada por uma operação de escrita seguida de leitura (entrada “W” seguida de “R”). Então o processamento do pacote é continuado com uma operação de escrita intermediária e, finalmente, as partes modificadas do pacote são escritas de volta na memória. O pacote é então submetido ao seu destino, como, por exemplo, o gerenciador de fila. Essas transações (leituras e escritas), os destinos bem como a quantidade de dados a serem transferidos são anotados no *trace* e o pacote é enviado para a interface da CPU.

O barramento, por sua vez, mapeia o volume de dados para as quantidades respectivas de palavras e – se o modelo cobrir este nível de detalhes – o número de transferências por rajada e arbitragem para administrar o uso do barramento.

Se o modelo de barramento estiver disponível, as requisições da CPU são encaminhadas como uma sequência de transações de barramento para a interface do escravo. Os escravos, que no exemplo apresentado em [69] consistem de memórias SRAM e SDRAM, registram nos *traces* as operações de leitura e escrita, que neste caso podem ser por transferências simples ou de rajada, incluindo suas próprias latências.

Ao fim, serão combinadas as latências dos recursos do SoC, presentes nos *traces*, para compor o desempenho da aplicação, como é exemplificado na Figura 9. Nela, podemos ver que são adicionadas, ao *trace* da CPU, as latências do protocolo do barramento e de acesso às memórias do SOC.

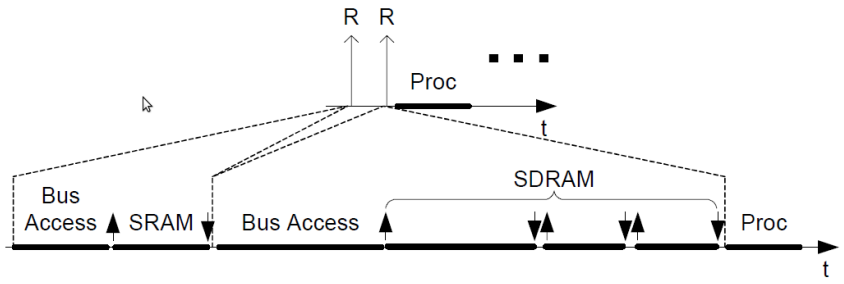


Figura 9 - Trace modificado para estimar o desempenho da aplicação SoC.

Assim como na abordagem apresentada em [12], o uso de perfil de comunicação - também chamado de *trace* nessas abordagens - permite considerar os efeitos dinâmicos de comunicação de uma aplicação, além de tornar mais eficiente o processo de estimação de desempenho, em relação às abordagens baseadas em simulações. Entretanto, os processo de combinação de *traces* e de adição de latências devem ser realizados para cada configuração distinta de barramento contida no espaço de projeto. Assim, em espaços maiores, o processo de exploração pode tornar-se computacionalmente impraticável.

3.4.6 Exploração de arquitetura on-chip para MPSoC baseado em pool de processadores

Um conjunto de elementos de processamento fortemente acoplados, juntamente com memórias *on-chip* e elementos de entrada/saída, formando subsistemas, e a partir destes, construindo um

sistema MPSoC completo, é chamado de pool de processadores (PP). O uso de PP em projetos de sistemas embarcados tem seus benefícios, que vão desde modularidade, reuso do subsistema e escalabilidade. Exemplos de uso desses subsistemas podem ser encontrados no projeto SHAPES em [79], em placas aceleradoras (GPUs) da NVidia, AMD, Intel e no projeto Cell BE da IBM [70].

A arquitetura SHAPES, também chamada de *tile* em [79], consiste de um conjunto de elementos de processamentos (EPs) diferentes, tais como RISC, DSP, memória *on-chip* e interfaces para comunicação externa ao *tile*. A estrutura de comunicação interna é uma matriz de barramentos multicamadas, com trocas de dados baseadas em envio de pacotes. A comunicação externa é realizada a partir de trocas de *streams* de dados. Estes dois modelos de comunicação implicam em estruturas de comunicação diferentes.

A organização dos componentes e a necessidade de exploração da arquitetura de comunicação da arquitetura SHAPES e de outros projetos baseados em PP motivaram a abordagem apresentada em [70]. Esta abordagem utiliza análise estática para podar o espaço de projeto, através do cálculo de largura de banda mínima de uso de memória a partir de informações de agendamento de tarefas. O espaço de projeto é composto por arquiteturas formadas a partir dos seguintes parâmetros da estrutura de comunicação: frequência de relógio, prioridades dos mestres (mecanismo de arbitragem por prioridade fixa), topologia da matriz de barramentos e alocação da memória *on-chip*, tanto para comunicação interna como comunicação externa ao chip. As arquiteturas baseada em PP consistem, para a abordagem proposta em [70], de vários PP e uma arquitetura de comunicação global (ACG) que interconecta os PPs. Cada PP contém EPs, memórias SRAM *on-chip* e uma interface para a ACG. A ACG contém memórias SRAM e uma interface para uma DRAM externa ao chip. Denota-se uma memória como bloco de memória física (BMF). Um bloco de memória lógico (BML), definido no modelo de tarefas da aplicação é mapeado para um BMF. As estruturas para comunicação interna e externa ao chip são baseadas em matrizes de barramentos. Uma matriz de barramento consiste de um modelo de barramento com múltiplos de pontos de arbitragem (PA), como pode ser visto na Figura 10.

O fluxo da abordagem de exploração tem como entrada o modelo de agendamento de tarefas, os traces de memória de cada tarefa e os requisitos mínimos de largura de banda para acesso aos BMLs pelas tarefas. Com isto é gerado um modelo de arquitetura, com uma memória SRAM *on-chip* para cada PP e ACG. Os demais parâmetros da arquitetura permanecem indeterminados.

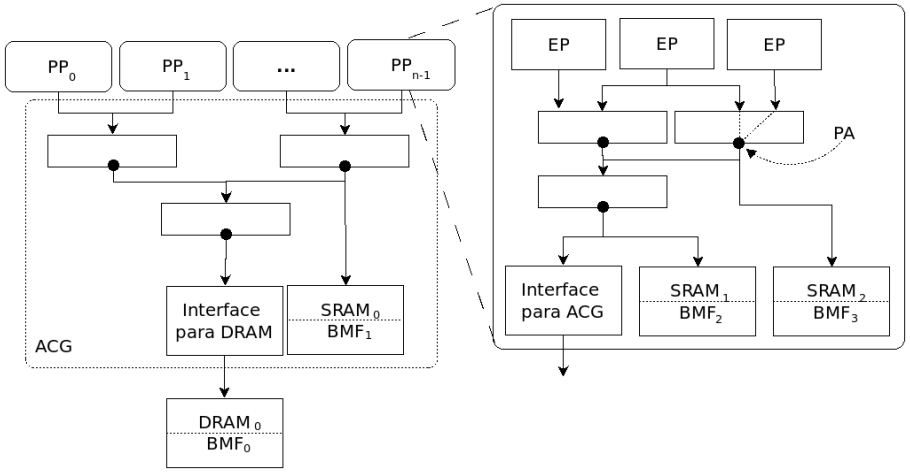


Figura 10 - Arquitetura MPSoC baseada em pool de processadores.

A próxima etapa do fluxo de projeto consiste no uso de técnicas de algoritmos evolucionários com inspiração quântica (AEIQ) para gerar arquiteturas candidatas, por alocação de BMLs para BMFs, determinar as topologias de matriz de barramentos, frequência de relógio e prioridades da arbitragem.

Assim como os algoritmos evolucionários (AE), AEIQ também consiste na representação de indivíduos, função de avaliação (ou ajuste) e dinamismo de população[80]. A diferença é que a abordagem AEIQ utiliza bits quânticos (Q-bits) como representação probabilística para os indivíduos, ao contrário dos AE tradicionais que utilizam representação binária dos genes.

A representação probabilística de Q-bits é descrita com dois valores α e β, onde $|\alpha|^2$e$|\beta|^2$referem-se as probabilidades para que os Q-bits tenham valores 0 ou 1 ($|\alpha|^2+|\beta|^2=1$), respectivamente. Assim, os genes que formam o cromossomo de um indivíduo quântico são representados por Q-bits, ou mais formalmente:

$$q_j^t = \begin{bmatrix} \alpha_{j_1}^t & \alpha_{j_2}^t & \dots & \alpha_{j_m}^t \\ \beta_{j_1}^t & \beta_{j_2}^t & \dots & \beta_{j_m}^t \end{bmatrix} \tag{19}$$

onde $j=1,2, \ldots,n$, $i=1,2, \ldots,m$, m é o número de Q-bits em cada cadeia e q_j^t é o indivíduo j da geração t.

Outra diferença entre as abordagens evolutivas é que AEIQ não requer operação de cruzamento entre múltiplas soluções, o que torna a implementação de AEIQ mais simples do que os AG tradicionais.

Para a diversidade das gerações, são disponibilizados vários grupos de soluções de uma única vez, cada um dos quais tendo sua própria representação de cadeia de Q-bits. Esta redundância resulta em soluções melhores, pela prevenção de queda em máximo local, apesar de duplicações excessivas poderem consumir muito tempo, tornando a técnica lenta. Definindo como *m* o tamanho da cadeia de Q-bits de um indivíduo, *n* a quantidade de grupos e *p* a quantidade de indivíduos em um grupo, o procedimento por AEIQ pode ser resumido como:

1. Para cada grupo, *p* indivíduos são gerados baseados na probabilidade dos Q-bits que foram definidas para o grupo. Então, um conjunto de indivíduos de todos os grupos da geração *t*, denotada por *P(t)*, é gerado. Cada um dos indivíduos em *P(t)* é reparado por regras dependentes da aplicação para tornar-se uma solução válida.
2. Então, todos os indivíduos em *P(t)* são avaliados, através da função de avaliação, e as melhores soluções são agrupadas e guardadas, em *B(t)*, a partir de *P(t)*.
3. Baseado em *B(t)*, a cadeia de Q-bits de cada grupo é ajustada de forma a gerar novos indivíduos próximos, em probabilidade, da melhor solução. Este processo é conhecido como rotação de Q-bits.
4. Entre as melhores soluções de todos os grupos, a melhor é guardada.
5. Se a condição de parada do algoritmo é alcançada, o processo de evolução finaliza e retorna a melhor solução encontrada. Assim, como nos AE tradicionais, uma condição de parada normalmente é definida como um número máximo de gerações ou até mesmo a convergência genética para a melhor solução. Caso as condições ainda não tenham sido alcançadas, o procedimento retorna ao passo 1.

Para podar o espaço de soluções candidatas geradas com o AEIQ, é aplicada técnica de análise estática que verifica se a arquitetura gerada atende às restrições mínimas de largura de banda (alocação de BLMs para BFMs e se PAs suportam o tráfego paralelo das tarefas do sistema). Os candidatos que não atendem a essas restrições são então descartados.

Para os que permanecem, é selecionado o melhor, que é aquele com maior largura de banda, para compor o conjunto *B(t)*, além daqueles que obtiveram as maiores pontuações a partir da função de avaliação (dada a partir da largura de banda máxima, áreas de memórias e matriz de barramento, e frequência de relógio). Com isso evita-se assim a necessidade de aplicar simulações para avaliar todas as arquiteturas candidatas.

Depois que o conjunto *B(t)* é construído, escolhe-se o melhor dessa geração *t*. Da mesma forma, é selecionado o pior para ser simulado. Se este último atende às restrições de projeto, é

atualizada a melhor solução global com a melhor solução da geração *t* e as demais soluções candidatas são descartadas, partindo-se assim para uma nova geração. O algoritmo de AEIQ continuará até que uma das condições de parada seja satisfeita.

Ao ser finalizado o AEIQ, a melhor arquitetura selecionada terá o número de BMFs, para cada PPs e ACG, incrementado em um, gerando assim um novo modelo de arquitetura que será novamente evoluída pelo algoritmo de AEIQ, para explorar soluções com desempenhos ainda maiores.

O número de BMFs será incrementado até atingir o número de MBLs. Depois que todas as iterações finalizam, é obtido um conjunto de arquiteturas que satisfazem as restrições de projeto. Baseado na área e na frequência de relógio das estruturas de comunicação, é estabelecido o conjunto de soluções Pareto-ótimo.

Utilizando as heurísticas propostas, estudos experimentais mostraram, em [70], que foi possível podar os espaços de projeto em até 99,9%, e, consequentemente, reduzir a necessidade de simulação para cada uma das configurações desse espaço. Além disso, o uso de AEs é flexível o suficiente para permitir que novos parâmetros sejam adicionados aos modelos evolutivos, desde que a função de avaliação seja adaptada para considerar as novas características, para favorecer a escalabilidade da técnica de análise.

3.4.7 Síntese de sistema combinada com exploração de arquitetura de comunicação para MPSoCs

Um nova abordagem de exploração de arquiteturas de comunicação é apresentada em [71]. Essa abordagem utiliza duas especificações distintas: uma para a aplicação e outra para a arquitetura. A partir destas duas especificações, várias implementações podem ser derivadas pela alocação da arquitetura e mapeamento da aplicação.

As especificações consistem de dois grafos (definidos a seguir):

- A arquitetura é dada por um grafo orientado $G_R (R, E_R)$. Os vértices R representam os recursos, tais como processadores, memórias e barramentos. Os arcos direcionados E_R indicam as conexões de comunicação entre dois recursos.
- A aplicação é dada por um grafo bipartido orientado $G_T (T, E_T)$ com $T=P \cup C$. Os vértices em T podem ser tarefa de processamento $p \in P$ ou tarefa de comunicação $c \in C$. Cada arco em E_T conecta um vértice em P para um em C, ou vice versa. Cada tarefa de

processamento pode ter vários arcos de entrada, que indicam dependências de dados para informação de comunicação da tarefa de comunicação antecessora. Cada tarefa de comunicação possui exatamente uma tarefa de processamento antecessora, como elemento de envio de dados, mas uma tarefa de processamento pode ter várias tarefas de comunicação sucessoras. É possível ainda que cada tarefa de comunicação tenha mais de um sucessor, caracterizando assim uma comunicação em *multicast*. Cada tarefa p em P pode ser implementada em um recurso a partir de R_p com $R_p \subseteq R$. Cada tarefa de comunicação $c \in C$ pode ser roteada como um subconjunto de recursos a partir R_c com $R_c \subseteq R$.

Uma implementação consiste do grafo de alocação G_A que é deduzido a partir do grafo da arquitetura e a função i que mapeia a aplicação em G_A. Assim:

- A alocação é um grafo orientado $G_A(A, E_A)$, que é um subgrafo induzido de G_R. A alocação contém todos os recursos que são disponibilizados na implementação atual e os arcos são induzidos a partir de G_R tal que G_A é notificado das conexões de comunicação.
- Cada tarefa de processamento $p \in P$ é limitada exatamente a um recurso alocado $i(p)$ tal que $i(p) \in (A \cap R_p)$. Cada tarefa de comunicação em $c \in C$ é roteada para uma árvore que é um subgrafo de alocação, tal que $i(c) \subseteq G_A$ com todos os vértices em R_C. Todas estas ligações e roteamentos têm de ser executadas de forma que todas as dependências de dados dadas pelas duas seguintes condições sejam satisfeitas:
  1. Para cada tarefa de comunicação $c \in C$, a raiz do roteamento tem que ser igual à ligação da tarefa de processamento antecessora $p \in P$. Que detém:

$$\forall (p, c) \in E_T : root(i(c)) = i(p) \tag{20}$$

  2. Para cada tarefa de processamento $p \in P$ os roteamentos da tarefa de comunicação antecessora $c \in C$ têm que ser para o mesmo recurso como a ligação do processo p. Assim:

$$\forall (p, c) \in E_T : i(p) \in i(c) \tag{21}$$

Uma implementação é classificada como praticável quando todos os requisitos relativos ao processo de mapeamento de processamento e comunicação bem como dependências de dados são atendidos. Conforme definição, o processo de exploração de espaços de projeto pode ser formulado como um problema de otimização de uma função multi objetivos *f(x)*, desde que *x* seja uma implementação praticável.

Para o processo de busca de melhor solução, foi necessário definir as seguintes variáveis binárias:

- $\boldsymbol{r}$ - uma variável para cada recurso $r \in R$ indicando se o recurso foi alocado (1) ou não (0);
- $\boldsymbol{p_r}$- um conjunto de variáveis para cada tarefa de processamento $p \in P$ e recursos $r \in R_p$ disponíveis, indicando se a tarefa de processamento é limitada no recurso (1) ou não (0);
- $\boldsymbol{c_r}$ - Um conjunto de variáveis para cada tarefa de comunicação $c \in C$ e cada recurso disponível $r \in R_c$, indicando se a tarefa de comunicação é roteada no recurso (1) ou não (0);
- $\boldsymbol{c_{r,n}}$ - Variáveis adicionais para cada par de recurso e comunicação indicando em qual passo de comunicação $n \in N$ (tarefas de comunicação são roteadas passo a passo) uma comunicação é roteada sobre um recurso.

A partir dessas definições, pode-se formular como restrições lineares as restrições apresentadas nas definições dos modelos de especificação de aplicação G_T e de arquitetura de comunicação G_R (ver fórmulas das restrições lineares em [71]). Com isto, o problema de encontrar uma solução praticável para *x*, torna-se um problema de busca binária. Assim, para uma solução *x*, sua implementação será dada pela alocação a partir das variáveis *r*, a ligação para cada tarefa de processamento a partir das variáveis p_r, e o roteamento da tarefa de comunicação a partir das variáveis c_r e $c_{r,n}$.

Assim, o processo de otimização de *f(x)*, para exploração dos espaços de projeto, é dado através da abordagem híbrida de otimização *SAT Decoding* [81], que utiliza algoritmos evolucionários multiobjetivos (AEMO) e um solucionador pseudo booleano[82] para solucionar problemas de satisfatibilidade booleana.

O algoritmo evolucionário utiliza modelos de simulação, codificados em SystemC TLM, para avaliação de cada indivíduo genético. Para tanto, os modelos de simulação são gerados e transformados em grafos de aplicação através dos seguintes passos:

- Utilizando as restrições de mapeamento, *threads* são obrigatoriamente mapeadas para recursos computacionais, como processadores, enquanto os canais de comunicação são mapeados para recursos de memória, como memórias *on-chip*.
- E cada ligação de *socket*[2] é representada exatamente por uma tarefa de comunicação que conecta uma tarefa de processo de envio a uma tarefa de recebimento.

A **Erro! Fonte de referência não encontrada.** ilustra o processo de transformação de um grafo de fluxo de dados de uma aplicação bastante simples.

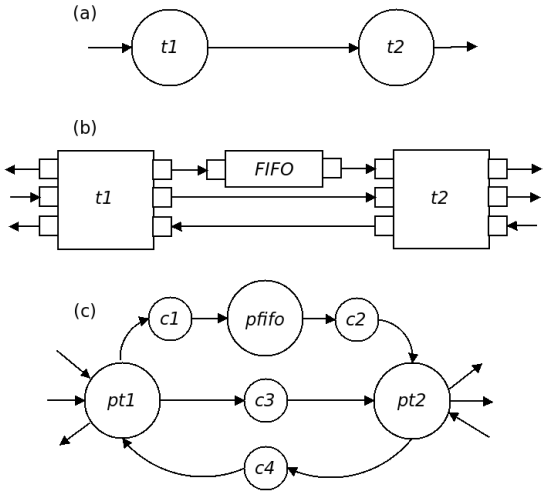


Figura 11 - Exemplo de transformação de grafo de fluxo de dados da aplicação (a) para respectivos modelo de simulação (b) e modelo de exploração (c).

Na **Erro! Fonte de referência não encontrada.** (a) temos um grafo de fluxo de dados para uma aplicação com apenas duas tarefas (vértices *t1* e *t2*). Este grafo é mapeado então para um modelo de simulação, ilustrado na **Erro! Fonte de referência não encontrada.** (b), com as tarefas mapeadas para threads de módulos SystemC (t1 e t2) e o arco *(t1,t2)* mapeado para um canal SystemC (*fifo*) e *sockets* SystemC (para eventos de sincronização). Por fim, o modelo de

[2] Em SystemC, um *socket* nada mais é do que uma abstração para um conjunto de pinos na interface de um componente.

simulação é mapeado para um grafo de aplicação, como é mostrado na **Erro! Fonte de referência não encontrada.** (c). Nesta, é ilustrado que os módulos SystemC foram mapeados para recursos computacionais e o canal SystemC para recurso de memória (vértices pt1, pt2 pfifo). As ligações entre módulos e canais SystemC, **Erro! Fonte de referência não encontrada.** (b), foram mapeados para tarefas de comunicação, ilustrado pelos vértices c1, c2, c3 e c4 na **Erro! Fonte de referência não encontrada.** (c).

Assim como as abordagens apresentadas em [51] e [70], o trabalho apresentado em [71] utilizou algoritmos evolutivos para gerar arquiteturas candidatas à solução, através de alocação de tarefas em recursos de computação e de comunicação. Contudo, diferentemente das outras abordagens, esse trabalho utilizou uma função de avaliação multiobjetivos de forma a considerar menor área ocupada, maior largura de banda e menor latência, como métricas de projeto.

De acordo com estudos experimentais, apresentados em [71], foi possível construir um conjunto pareto-ótmo, considerando as métricas de largura de banda e área, com 27 soluções, selecionadas a partir da avaliação de 7600 configurações distintas. Quando um dos objetivos se torna a latência, a avaliação do desempenho das arquiteturas candidatas é realizada por simulação. Assim, todos os candidatos, de todas as gerações, têm que ser simulados, reduzindo-se assim a eficiência do algoritmo evolutivo. De acordo com os autores, 99,9% do tempo de exploração é gasto com as simulações das arquiteturas candidatas.

3.5 Análise das Abordagens

Nas seções anteriores, foi apresentado um resumo do estado da arte em análise de comunicação em plataformas multiprocessadoras. Considerando o principal objetivo deste trabalho de tese e as principais características dos trabalhos abordados da literatura, pode-se listar um conjunto de métricas essenciais ao processo de exploração de espaços de projeto de arquiteturas de comunicação baseadas em barramentos. Seguem descrições de cada uma:

- **Precisão**: é necessário que as estimativas de desempenho sejam precisas, captando efeitos dinâmicos de comunicação, como, por exemplo, contenção por arbitragem, para permitir identificar as soluções ótimas (melhores e/ou piores desempenhos da comunicação da aplicação, a partir das configurações de barramento do espaço de projeto).

- **Desempenho**: para identificar as melhores configurações, a técnica de análise deve ser capaz de explorar todo o espaço de projeto. Assim, a análise de configurações de barramento em espaços de projeto maiores, deve-se utilizar abordagem eficiente.

- **Compromisso**: flexibilidade da abordagem de exploração para permitir estabelecer um compromisso entre métricas de projeto, como, por exemplo, desempenho e precisão, de forma a beneficiar uma das duas. Por exemplo, em fases iniciais, o projeto de um sistema embarcado concentra-se na validação funcional dos elementos do sistema a ser desenvolvimento. Assim, pode-se reduzir a precisão das estimativas de desempenho para tornar o processo de avaliação funcionais mais eficiente.

- **Concorrência**: capacidade de analisar paralelamente várias configurações do espaço de projeto. Com isto evita-se ter de analisar cada configuração individualmente, demandando várias iterações até que o espaço de projeto seja coberto e a configuração ótima seja encontrada.

- **Detalhamento**: disponibilizar informações sobre processos envolvidos na comunicação, como, por exemplo, latências de transações, endereços acessados e eventos de comunicação. Com estas informações, pode-se, por exemplo, gerar estatísticas de uso e eficiência de uso do barramento, ou até mesmo, identificar caminhos críticos e porcentagem de utilização de caminhos críticos por cada mestre.

- **Flexibilidade**: possibilidade de construção de espaços de projeto maiores, através do uso de variados modelos de barramento, bem como respectivos parâmetros de configuração.

- **Escalabilidade**: permite estender o uso da técnica de exploração em sistemas mais complexos, com maior número de componentes de plataformas e níveis de barramento.

- **Automatização**: a abordagem de exploração deve ser capaz de suportar mecanismos para automatizar os processos envolvidos na estimação de desempenho, analise e seleção de configurações, como, por exemplo, as melhores e piores, ou alguma que atenda as restrições de projeto. Porém, deve permitir interação entre o projetista e o processo de análise de comunicação.

A Tabela 1 faz um mapeamento entre as abordagens analisadas neste capítulo (colunas) e as características descritas (linhas). Os trabalhos foram agrupados por seus respectivos tipos de abordagens de análise: simulação de sistema, estimação estática e técnicas híbridas (Seções 3.2, 3.3 e 3.4, respectivamente). Logo abaixo dos rótulos de grupo, cada um dos trabalhos analisados é rotulado pela numeração de subseção em que foi descrito. Para as células que cruzam as abordagens com as características observadas, as legendas são: 'x' significa que a abordagem contempla totalmente a característica; 'p' significa que atende parcialmente e '-' implica que não dá suporte.

Tabela 1 - Comparativo entre abordagens de análise de comunicação

Abordagem / Caracter.	Simulação de sistema (3.2)					Estimação Estática (3.3)			Híbrida (3.4)						
	3.2.1	3.2.2	3.2.3	3.2.4	3.2.5	3.3.1	3.3.2	3.3.3	3.4.1	3.4.2	3.4.3	3.4.4	3.4.5	3.4.6	3.4.7
Precisão	x	x	x	x	x	p	p	p	p	x	-	p	x	x	x
Desempenho	-	-	-	-	-	p	x	x	p	-	-	p	p	-	-
Compromisso	x	x	x	x	x	-	-	-	-	x	-	-	-	-	-
Concorrência	-	-	-	-	-	-	-	-	-	x	-	-	-	x	x
Detalhamento	x	p	x	x	x	p	-	-	x	-	-	p	x	-	-
Flexibilidade	x	x	p	x	x	p	-	-	x	x	-	p	x	x	x
Escalabilidade	x	x	x	x	x	x	x	-	x	x	-	x	x	x	x
Automatização	-	p	p	-	p	-	-	-	-	x	x	x	-	x	x

Legenda:
- = não suportada
x = suportada
p = parcialmente suportada

A análise da tabela indica que as abordagens de análise de comunicação apresentadas neste capítulo são limitadas, considerando as características descritas anteriormente.

Tomando cada uma das características, observa-se que dentre todas as abordagens analisadas conjuntamente, destacam-se as baseadas em simulação como as mais precisas. Desde que os simuladores captam características reais do sistema, como, por exemplo, funcionalidades, latências e efeitos dinâmicos, todas puderam obter, segundos os autores, estimavas precisas. As abordagens de estimativas estáticas suportaram parcialmente esta característica, pois resultados precisos dependem de especificações prévias dos modelos de agendamento de tarefas ou transações. Além disso, a precisão das estimativas é também prejudicada, neste tipo de abordagem, por não considerar os efeitos dinâmicos de comunicação. Para o último grupo, técnicas híbridas, a abordagem da subseção 3.4.3 destaca-se por utilizar modelos formais para verificação de propriedades do sistema, identificando apenas o pior desempenho do espaço de projeto, com valores de estimativas superiores aos reais, assim como é apontado em [63].

Ao contrário dos resultados anteriores, as técnicas baseadas em simulação apresentaram menores desempenhos para análise de pontos dos espaços de projeto, mesmo utilizando modelos em nível de sistema e comunicações em nível transacional. Isto se justifica pela necessidade de simulação completa do sistema, que pode consumir muito tempo, dependendo da aplicação, para que as estimativas possam ser obtidas. Em suma, as abordagens de estimativas estáticas são bastante eficientes. Porém, a abordagem da subseção 3.3.1, que modifica o agendamento de tarefas para embutir latências de comunicação através de técnica de enfileiramento, tem seu desempenho reduzido linearmente, segundos os autores, na medida em que a complexidade da aplicação aumenta [60]. Já as abordagens híbridas incluem simulações em seus fluxos de projeto, herdando assim os baixos desempenhos. Uma parte das abordagens híbridas utiliza técnicas complementares para 1) buscar formas de minimizar a quantidade de simulações necessárias para exploração, ou 2) para estimação de desempenho, complementando assim os resultados das simulações. Essas últimas são descritas nas subseções 3.4.1, 3.4.4 e 3.4.5.

Observe que as técnicas de simulação de sistema permitem estabelecer um compromisso entre precisão e desempenho, pois os modelos de comunicação envolvidos permitem aumentar a abstração das transferências, através de transações em nível de sistema, e com isso aumentar o desempenho em favor da precisão. Da mesma forma, pode-se penalizar o desempenho através da inserção de maior quantidade de características de comunicação. A abordagem híbrida da subseção 3.4.2, que utiliza modelos de simulação codificados em SystemC RTL, também permite que estes modelos possam ser substituídos por outros mais abstratos para consequente aumento de desempenho.

Pode-se, ainda, otimizar o desempenho da abordagem de análise sem comprometer a precisão das estimativas através de exploração concorrente. As abordagens híbridas apresentadas nas subseções 3.4.2, 3.4.6 e 3.4.7 exploram esta característica através de algoritmos evolutivos. Observe que as três não dão suporte à análise eficiente de configurações individuais, mas, por utilizarem paralelismos, conseguem explorar os espaços de projeto com alto desempenho e obter resultados com precisão.

Ainda, por utilizar modelos evolucionários para realizar estimativas, essas três abordagens não permitem detalhar a comunicação para as respectivas configurações do espaço de projeto, que é característica dominante entre as abordagens que se baseiam em simulação do sistema. Assim, para análise mais detalhada da comunicação de uma determinada aplicação, deve-se utilizar modelos de simulação de sistema ou abordagens baseadas em traces de execução (abordagens híbridas apresentadas nas subseções 3.4.1 e 3.4.5).

Flexibilidade e escalabilidade são duas características importantes, pois ambas estão relacionadas com a construção de espaços de projeto maiores. Observando os grupos de técnicas, destacam-se as baseadas em simulações de sistema e híbridas, que suportam totalmente ou parcialmente essas duas características. A exceção é a abordagem apresentada na subseção 3.4.3, que assim como as abordagens do grupo de estimativas estáticas, inclui um modelo de estimativas de desempenho específico para um tipo de barramento. Madl et. al. também destaca, em [63], que essa abordagem não é escalar porque seu uso com aplicações mais complexas pode desencadear explosão de estados para análise, e com isso tornar-se computacionalmente impraticável. Em termos gerais, pode-se então induzir que as abordagens que apresentaram flexibilidade para configuração de seus modelos de comunicação são também escaláveis para aplicações embarcadas maiores.

A observação da última característica nos trabalhos científicos analisados mostrou que, em grande parte das abordagens, os mecanismos de exploração são parcialmente automatizados. Algumas abordagens baseadas em simulação apresentaram mecanismos simples para automatização, como, por exemplo, a configuração de comunicação da plataforma, compilação da aplicação e geração do simulador, mas o processo de exploração ainda é condicionado à ciclo iterativos ditados pela experiência do projetista. As abordagens híbridas que automatizaram a exploração, em geral, utilizaram heurísticas e algoritmos evolutivos para tomadas de decisões.

3.6 Conclusões

Foram apresentadas nas seções anteriores diversas abordagens para exploração de espaços de projeto de comunicação de sistemas embarcados multiprocessados. Neste trabalho de tese, estas abordagens puderam ser agrupadas em técnicas que utilizam simulação do sistema, baseadas em estimativas estáticas e híbridas.

Observou-se nessas abordagens um conjunto de características importantes e necessárias para otimizar o projeto da comunicação. Desta observação, concluiu-se que nenhuma das abordagens suporta conjuntamente todas essas características.

Desta forma, este trabalho de tese propõe uma nova abordagem para exploração de espaços de projeto de comunicação através da combinação de técnicas de simulação, análise estática e algoritmos evolutivos, e que tem como requisitos as características apresentadas na seção anterior, observadas nos trabalhos da literatura.

Capítulo 4
Análise de Comunicação em Plataformas Multiprocessadoras

4.1 Introdução

Como visto nas seções anteriores, MPSoCs são sistemas que contém vários componentes heterogêneos de hardware. Novas abordagens, como a de projeto baseado em plataformas, vêm sendo utilizadas visando aumentar o desempenho do projeto desses sistemas e a qualidade do produto final através de reuso de plataformas virtuais predefinidas.

Na abordagem PBP, as funções do sistema a ser desenvolvido são escolhidas para serem implementadas por hardware ou software, que é a fase de particionamento, e, em seguida, são mapeadas para os componentes de uma plataforma virtual preconcebida, capaz de tratar aquela classe de aplicações. Desse mapeamento, seguem processos de análise para possíveis refinamentos na aplicação, plataforma ou até mesmo do modelo de mapeamento. Uma vez que o protótipo do sistema é validado, de acordo com as restrições de projeto, segue-se a implementação final do sistema.

Para satisfazer restrições de comunicação entre componentes de plataforma, é necessário que a plataforma inclua uma estrutura de comunicação eficiente. Para o projeto de estruturas de comunicação baseadas em barramentos, deve-se considerar as diferentes combinações de valores dos parâmetros de configuração para que os requisitos de comunicação possam ser atendidos. Entre esses parâmetros, temos: largura dos barramentos de dados e endereços, disponibilidade de DMA e *pipeline*, frequência de operação, hierarquia de barramentos, disposição dos componentes entre os barramentos, entre outros já discutidos.

Novas abordagens vêm sendo desenvolvidas para auxiliar o processo de exploração dos espaços de projeto de estruturas de comunicação, porém, essas técnicas não preenchem todos os requisitos analisados. Vimos que para haver um processo de exploração eficaz, é necessário

obter estimativas precisas de desempenho para todas as configurações de barramento do espaço de projeto. Além disso, em espaços maiores, construídos a partir da inclusão de mais características de configuração de barramentos, o mecanismo de exploração deverá ser escalavel e atuar de forma eficiente, para que seja possível analisar todas as possíveis soluções até encontrar a melhor. Contudo, muitas vezes, o processo de customização da comunicação não consiste apenas em explorar as configurações do barramento. Deve-se analisar também o comportamento da aplicação, através da análise de perfis de comunicação e/ou traces de acesso à memória, para identificar problemas de comunicação, como *starvation*.

Nesse contexto, a proposta deste trabalho de tese concentra-se no desenvolvimento de uma nova técnica de análise de arquiteturas de comunicação baseadas em barramentos. A técnica proposta se baseia uma simulação completa do sistema combinada com algoritmos evolutivos e análise estática, com objetivo de dar suporte ao projetista identificar, através de uma exploração eficiente, uma configuração da estrutura de comunicação que minimiza as latências de comunicação entre os componentes da plataforma, ou que pelo menos atenda às restrições temporais definidas no projeto.

O restante do capítulo apresenta com maiores detalhes o fluxo proposto para análise de estruturas de comunicação, bem como descreve as técnicas adotadas para sua implementação.

4.2 Abordagem Proposta

O fluxo de projeto da abordagem proposta para análise de comunicação, em termos gerais, contém as seguintes fases, dispostas em ordem de execução: (1) extração do perfil de comunicação, entre os componentes da plataforma, que será utilizado para (2) estimação de desempenho do sistema considerando as configurações de barramento de um subconjunto do espaço de projeto. As estimativas serão utilizadas para (3) geração de modelo de comunicação e, por fim, (4) estimação de espaço de projeto. O fluxo da abordagem de exploração proposta, com detalhamento das fases, pode ser visto na Figura 12.

A primeira fase tem início com o mapeamento da aplicação para uma plataforma virtual pré-concebida, para que o sistema possa ser simulado. Para modelar a comunicação entre os componentes dessa plataforma, é proposto neste trabalho um modelo de barramento genérico, desenvolvido a partir do trabalho apresentado em [11]. A vantagem deste modelo de barramento é permitir a simulação considerando um modelo de estrutura de comunicação ideal, o qual não

introduz latências nos processos de comunicação e que permita transferências em paralelo, desde que os periféricos forneçam suporte para este tipo de acesso. Adicionalmente, este modelo permite o registro das informações de todas as transferências, possibilitando a extração de um perfil de comunicação da aplicação.

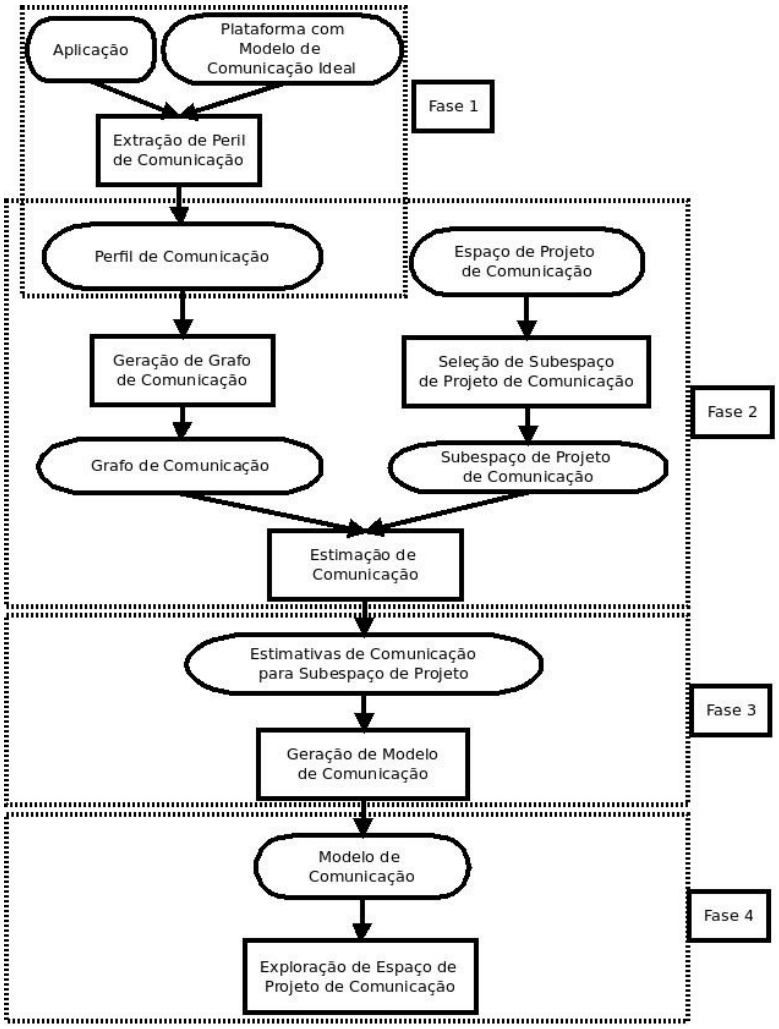


Figura 12 - Fluxo principal proposto para abordagem de exploração de espaço de projeto de comunicação

A segunda fase começa com a representação do perfil de comunicação na memória principal, através de uma estrutura de dados, com objetivo de otimização do acesso aos dados registrados no perfil. 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 sob os efeitos de barramentos reais. Com isso pode-se aumentar o desempenho de exploração do espaço de projeto de arquiteturas de comunicação, através da redução do número de simulações.

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 os perfis de comunicação é uma extensão do modelo proposto por Lahiri et al. em [12], que utilizou grafos - chamado de Grafos de Comunicação (GC) - para representar o perfil. No trabalho de Lahiri et. al. os vértices dos grafos 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 deste trabalho de tese é a extensão do trabalho proposto por Lahiri et. al., 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, de forma a tornar as estimativas mais precisas.

Com as duas fases iniciais do fluxo de projeto proposto neste trabalho de tese, assim como no trabalho de Lahiri et. al., é possível obter estimativas de desempenho da aplicação para cada configuração de barramento do espaço de projeto. Porém, a abordagem de estimação por grafos de comunicação não inclui automatização do mecanismo de análise, sendo necessário que o projetista selecione as configurações de barramento para estimar seu desempenho. Como discutido no capítulo anterior, em aplicações mais complexas, as plataformas terão maior número de componentes e, consequentemente, haverá maior número de parâmetros de configuração a serem ajustados. Assim, os mecanismos de suporte à análise de comunicação

devem dispor de mecanismos que automatizem o processo de exploração dos espaços de projeto de comunicação.

Com isso, as demais atividades da segunda fase e fases posteriores foram propostas objetivando o aumento do desempenho e automatização da abordagem de exploração com grafos de comunicação.

Ainda na fase 2 do fluxo de projeto proposto, ilustrado na Figura 12, após representar o perfil de comunicação em memória, através de grafos de comunicação, é selecionado um subconjunto do espaço de projeto, através do uso de uma técnica de projeto de experimentos [22], cujas configurações de barramento possam ser utilizadas para estimar os desempenhos do sistema, através da modificação dos grafos de comunicação.

Com isso, as configurações de barramento e respectivos desempenhos, estimados por grafos de comunicação, poderão ser utilizados para obter, na terceira fase do fluxo de projeto proposto, um modelo de comunicação da aplicação, capaz de estimar, na última fase, o desempenho de comunicação para todas as configurações do espaço de projeto. Desta forma, busca-se aumentar a eficiência da exploração do espaço de projeto através da minimização da quantidade de estimativas obtidas pela abordagem de grafos de comunicação. Para o processo de exploração de comunicação, com este modelo de comunicação, busca-se estimar todo o espaço de projeto de forma automática, com resultados precisos, além da possibilidade de identificar configurações bem específicas, como as melhores ou piores.

A seção seguinte detalha, por fases, os modelos propostos e técnicas desenvolvidas neste trabalho de tese, que fazem parte do fluxo mostrado na Figura 12.

4.3 Descrição das Fases do Fluxo de Projeto

O fluxo de projeto proposto, ilustrado na Figura 12, é dividido em fases, que incluem o uso de técnicas distintas, e que, ao fim de cada uma delas, são gerados artefatos. Alguns destes artefatos constituem entradas para fases subsequentes, seguindo o fluxo de projeto proposto para exploração. Até a obtenção de um artefato final, o modelo de comunicação que pode ser utilizado para estimar o desempenho de todo o espaço de projeto.

Estas técnicas e artefatos compõem parte essencial do presente trabalho. Esta seção detalha, para cada fase do fluxo proposto, a descrição desses elementos, utilizados para

composição da abordagem de exploração do espaço de projeto da arquitetura de comunicação para uma determinada aplicação.

4.3.1 Fase 1 – Extração de Perfil de Comunicação

A primeira fase da abordagem de exploração proposta neste trabalho de tese consiste na extração de um perfil de comunicação da aplicação (ver fluxo de projeto proposto para esta fase na Figura 13) .

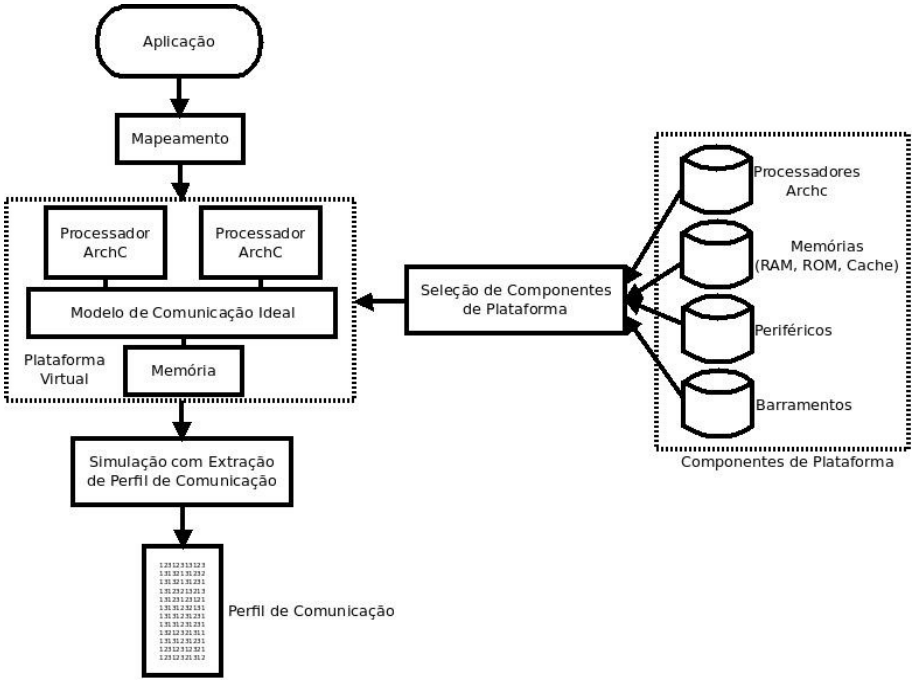


Figura 13 - Fluxo de projeto para extração de perfil de comunicação de uma aplicação.

O fluxo inicia com a especificação de um modelo de simulação do sistema, composto pela aplicação mapeada na plataforma virtual. A plataforma virtual inclui modelos de simulação de componentes de hardware, como, por exemplo, processadores, barramentos, e memórias.

Nessa plataforma virtual deve ser utilizado, como estrutura de comunicação, um modelo de barramento genérico, que permite transações em paralelo e não introduz latências na comunicação. Este modelo de barramento suporta o registro das informações de comunicação entre os componentes da plataforma, durante a simulação do sistema. A partir destas informações é gerado o perfil de comunicação.

A seguir serão detalhados os modelos propostos, neste trabalho de tese, para plataformas virtuais, estrutura de comunicação ideal e perfil de comunicação, bem como a técnica de extração do perfil de comunicação.

4.3.1.1 Modelos de simulação de plataformas

O modelo de plataforma virtual utilizado neste trabalho de tese foi inicialmente proposto em [11], e inclui componentes codificados em SystemC, com interfaces de comunicação padronizadas com SystemC TLM [25]. Esses componentes consistem basicamente de:

- processadores, descritos em ArchC [24], que é uma linguagem de descrição de processadores, que por sua vez, gera um modelo simulável codificado em SystemC com interface SystemC TLM;
- memórias RAM, com capacidade, tamanho de palavra e latências parametrizáveis e
- conectores (*wrappers*) para suportar inclusão de tipos diferentes de barramentos, como definido inicialmente em[11] e publicado em [115][116].

Para o software embarcado, o modelo de aplicações multiprocessadas, adotado para este trabalho de tese, define que cada processo deve ser executado em um mestre de barramento diferente, uma vez que não é utilizado sistema operacional com escalonador de processos para gerenciar a troca de contexto. Essas aplicações foram codificadas seguindo o padrão ANSI C e compiladas com compiladores cruzados, disponíveis na página do projeto ArchC[43].

O modelo de estrutura de comunicação, que é utilizado para extração de perfil de comunicação, será descrito com maiores detalhes na subseção a seguir.

4.3.1.2 Modelos de Estrutura de Comunicação Ideal e do Perfil de Comunicação

O mecanismo de exploração proposto é baseado em uma simulação inicial da aplicação mapeada na plataforma utilizando o modelo de barramento genérico proposto, como arquitetura de comunicação. Este modelo dá suporte ao registro de informações da comunicação entre componentes da plataforma virtual e contém as seguintes características:

- permite transferências de dados com tamanhos variáveis. Como não existe a restrição de largura de barramento de dados, é possível que haja transferências de dados com tamanhos de palavras variados.

- permite transferências de vários mestres distintos em paralelo: uma vez que não existem prioridades de uso do barramento. É possível que dois ou mais mestres possam acessar, em paralelo, algum dispositivo, desde que este possua suporte a transferências paralelas.

- inclui um árbitro de barramento com política de alocação híbrida com classes de prioridades. Assim, é possível atribuir aos mestres prioridades de uso do barramento, caso seja necessário. Este tipo de característica é importante em casos onde devem existir processos com maior prioridade sobre outros. Nos casos onde não existe prioridade, como dois processos de mesma prioridade realizando transferências, o árbitro pode conceder acesso paralelo ao barramento. Novamente, é importante ressaltar que o dispositivo escravo deve permitir acessos paralelos. Em casos onde há necessidade de atribuir prioridade a algum mestre e a outros não, o árbitro atuará de forma híbrida: haverá classes de prioridades, onde mestres em classes de maior prioridade terão acesso ao barramento concedido, e mestres pertencentes à mesma classe de prioridade acessarão o barramento em paralelo.

- foi codificado em SystemC, com interfaces que seguem o padrão SystemC TLM, em alto nível de abstração e funcional (não insere latências de comunicação nas transferências).

- foi desenvolvido com base no modelo proposto em [11], que define um padrão de interface para modelos de simulação de barramento. Este padrão disponibiliza um modelo de componente de interface, chamado de *wrapper*, que permite conectar componentes de plataforma e barramentos. Este componente permite que componentes mestres e escravos possam ser plugados facilmente, além de permitir que o modelo de barramento da plataforma possa ser trocado sem necessidade de modificação da interface de comunicação dos componentes da plataforma. Nestes *wrappers*, a interface com o barramento considera que uma transação de barramento é composta de quatro fases

fundamentais, como definido na Subseção 2.3.2, que são: requisição de uso do barramento, configuração da transação, transmissão de dados e liberação do barramento.

- possui suporte para registrar informações de comunicação, suportando assim a geração de um perfil de comunicação da aplicação. Durante uma simulação, o modelo de comunicação proposto permite capturar diversos tipos de informações, como eventos embutidos nas transferências, dados transferidos, endereços acessados, tempos entre transferências (como latências de processamento por mestres e estados de espera inseridos na comunicação por dispositivos escravos).

A partir das informações capturadas é gerado um perfil de comunicação da aplicação. Um perfil de comunicação poder conter, assim como definido inicialmente em [11], diversos tipos de informações, como endereços de acesso e dados transferidos, além de dados estatísticos, como uso e eficiência de utilização do barramento por cada mestre.

Para a abordagem proposta neste trabalho de tese, o perfil de comunicação é definido formalmente a seguir:

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 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.

As Figura 14 e 15 apresentam diagramas que exemplificam perfis de comunicação extraídos de quatro aplicações distintas quaisquer.

Como podemos observar, as Figura 14 e 15 mostram, em suas partes superiores, Diagramas de Fluxo de Dados Interprocessos (DFDI) e, logo abaixo, um trecho de seus respectivos perfis de comunicação. O DFDI é definido como um diagrama que utiliza grafos para representar processos, e respectivos sentidos de comunicação. Nestes grafos, os processos são representados como vértices e o sentido das transferências de dados (comunicação ou

sincronização) como arcos direcionados. Já nos perfis de comunicação, os elementos de cada linha representam, nesta ordem: nome do mestre, tipo de evento de transação e número da transação. O nome do mestre refere-se ao identificador do componente mestre que realiza a transferência no barramento. Tipo de evento representa uma as fases do protocolo de barramento internas a uma transação e pode ser de quatro tipos: requisição (*q*), controle (*c*), dado (*d*) e liberação (*l*). Por fim, o número de sequência da transação é a contagem do número total de transações (transferências completas com as quatro fases do protocolo) realizadas no barramento. Para transações realizadas em paralelo, no modelo de barramento ideal proposto, a contagem de transações será acrescida apenas uma única vez, contabilizando como se apenas uma transação tivesse sido realizada, representando, assim, no perfil, o paralelismo de transações.

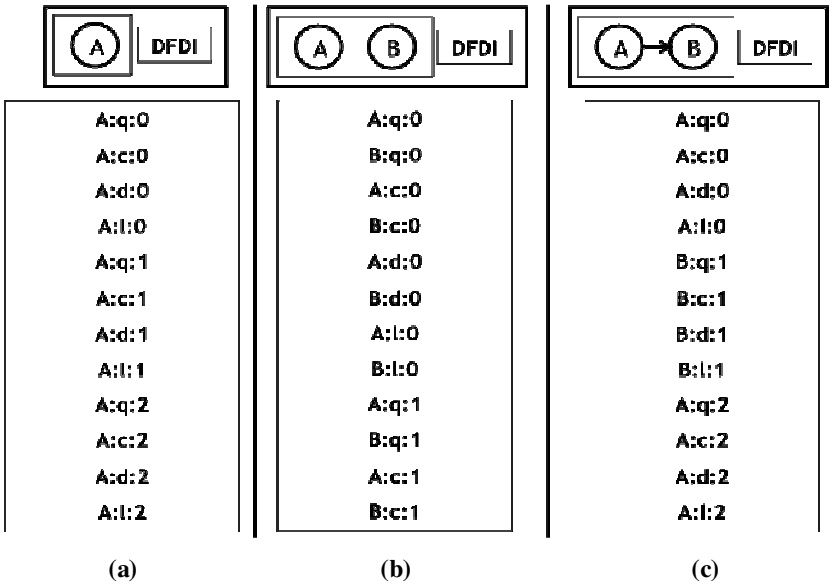


Figura 14 - Perfis de comunicação para aplicações com(a) apenas um mestre, (b) dois mestres sem comunicação e (c) dois mestres com comunicação.

Na Figura 14(a), o DFDI possui apenas um vértice, ilustrado pelo círculo com a letra A, representando assim uma aplicação com apenas um processo. Em seu perfil de comunicação, vemos que a sequência dos eventos *q*, *c*, *d* e *l* sempre se repetem e a cada repetição há um incremento no contador das transações, caracterizando, assim, transações compostas pelos quatro eventos.

A Figura 14(b) contém um DFDI com dois vértices e nenhum arco. Este diagrama representa uma aplicação com dois processos que não se comunicam. Observando o perfil de

comunicação para esta aplicação, verifica-se que a sequência dos eventos das transações se repete para os dois mestres: a cada evento para o mestre A, logo em seguida, no perfil de comunicação, ocorre o mesmo evento para o mestre B. É possível ver ainda que a contagem de transações também segue a mesma sequência para ambos os mestres e sofre incremento a cada par de transações, uma para cada um dos mestres. Este comportamento expressa a ideia de paralelismo na uso do barramento, onde as transferências do mestre A são realizadas ao mesmo tempo que as do mestre B.

Na Figura 14(c), o DFDI mostra dois processos, com comunicação entre A e B (arco direcionado saindo do vértice A para B). Esta aplicação poderia ser, por exemplo, um sistema produtor-consumidor, sendo A o produtor e B o consumidor. Porém, para garantir este comportamento, o mestre A deverá ter maior prioridade de uso do barramento sobre B. Neste caso, os processos não mais farão transferências em paralelo, mas sim sequenciais. Deve-se então aplicar ao barramento um mecanismo de arbitragem por prioridades, onde o mestre A terá maior prioridade. Observando o perfil de comunicação da Figura 14(c), verifica-se que a contagem de transações é incrementada a cada transação, quer seja do mestre A ou do mestre B.

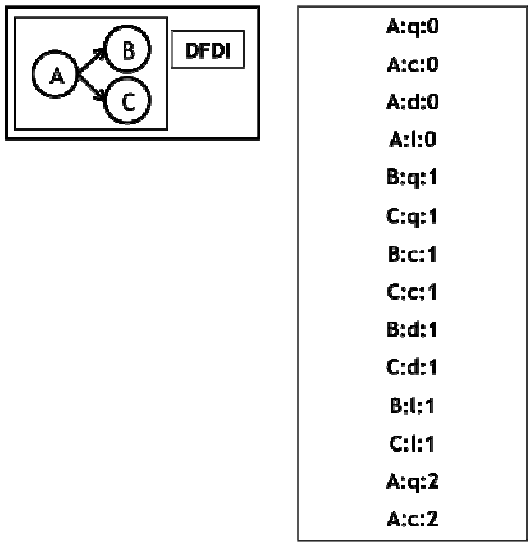


Figura 15 - Perfil de comunicação para aplicação com três mestres de barramento com comunicação.

Já a Figura 15 mostra um DFDI para uma aplicação com três processos, sendo que existem transferências de dados (ou sincronização) entre o processo A e os processos B e C. O perfil de comunicação referente a esse DFDI mostra que a contagem de transações é

incrementada a cada transação de A e a cada par de transações, uma para cada mestre, de B e C. Percebe-se, assim, com este perfil que A tem maior prioridade sobre os mestres B e C (uma vez que a contagem de transações inicia com A) e estes dois últimos estão contidos na mesma classe de prioridades de uso do barramento realizando transações em paralelo (contagem de transações de mesmo valor).

A próxima subseção detalha o processo de extração de perfil de comunicação de uma aplicação.

4.3.1.3 Algoritmo para Extração de Perfil de Comunicação

Uma vez que a aplicação é mapeada no modelo de plataforma virtual, que inclui o modelo de barramento genérico como estrutura de comunicação, o barramento irá capturar informações de comunicação entre os componentes da plataforma durante a simulação do sistema. Ao fim da simulação, todas essas informações estarão armazenadas em um arquivo, resultando assim no perfil de comunicação da aplicação. Um pseudo-algoritmo para extração de perfil de comunicação pode ser visto na Figura 16.

```
1:  While true:
2:  wait(event);
3:  if event is request:
4:     registerRequest(profile, master, t_counter(master));
5:     arbiter.doArbitration(master);
6:  else if event is control:
7:     if is master:
8:        registerControl(profile, master, t_counter(master));
9:        slave = decoder.decodeAddress(control.address);
10:       sendControlToSlave(slave, control);
11: else if event is data:
12:    if is master:
13:       registerData(profile, master, t_counter(master));
14:       if is control.write:
15:          sendDataToSlave(slave, data);
16:       else readDataFromSlave(slave, data);
17: else:
18:    if is master:
19:       registerRelease(profile, master, t_counter(master));
20:       releaseBus(self);
21: arbiter.incrementCounter(t_counter(master));
```

Figura 16 - Fluxo principal do algoritmo de extração de perfil de comunicação.

O algoritmo consiste de um laço de iteração (linha 1), cujas instruções iniciam com o bloqueio do barramento, até que ocorra um evento de comunicação (linha 2). Como visto em seções anteriores, existem quatro tipos de eventos de comunicação, e dependendo do tipo, o barramento irá atuar de forma diferente. Assim, o próximo passo é identificar o tipo de evento (linhas 3, 6, 11 e 17).

Caso o evento seja uma requisição de uso do barramento (linha 3), na linha 4, será registrado no perfil de comunicação o identificador do mestre, o evento de requisição e o número de sequência da transação em andamento para o mestre que requisitou uso (cada mestre conterá seu próprio contador de transações). Após o registro, o arbitro do barramento inicia o processo de arbitragem, bloqueando o mestre até que tenha permissão de uso do barramento (linha 5).

Caso o evento seja de envio de sinais de controle (linha 6), o barramento verifica se os sinais foram enviados pelo mestre atual do barramento e, em caso positivo, registra o identificador do mestre, o evento de envio de sinais de controle e o número de sequência da transação em andamento. Em seguida, o barramento aciona o decodificador, que realiza o processo de decodificação de endereço (linha 9) para selecionar o escravo destino da transação e, após selecionado o escravo, serão enviados para ele os sinais de controle da transação (linha 10).

Já para o evento de envio de dados (linha 11), novamente o barramento verifica se o evento foi disparado pelo mestre do barramento (linha 12). Se confirmado, além de registrar o evento no perfil de comunicação (linha 13), o barramento irá verificar o tipo de transferência, se é de escrita ou leitura de dados (linhas 14 e 16 respectivamente), realizando assim escrita (linha 15) ou leitura (linha 16) de dados no escravo selecionado.

Para o último tipo de evento, que é de liberação do barramento, é verificado se a solicitação de liberação é dada pelo mestre atual (linha 18). Uma vez confirmada, é registrado no perfil de comunicação o evento de liberação do barramento e a liberação do barramento é realizada (linhas 19 e 20 respectivamente).

Por fim, na linha 21, a iteração finaliza com o incremento do contador de transação do mestre que finalizou a transação, pelo arbitro do barramento. O árbitro verifica o esquema de prioridades definido para mestres, de forma que os mestres com prioridades iguais terão o mesmo valor de contagem de transações, e mestres com valores diferentes terão valores de contagem alternados.

A Figura 17 contém um diagrama que ilustra a estrutura de conexão de mestre e escravo

ao barramento genérico para extração do perfil de comunicação. Percebe-se, nessa figura, que os componentes mestre e escravo comunicam-se através de transações TLM com componentes de interface mestre (*wrapper* mestre) e escravo (*wrapper* escravo), respectivamente. O *wrapper* mestre receberá um pacote TLM do mestre, contendo informações de comunicação, como por exemplo, endereço de acesso e dados a serem transferidos, e deverá encaminhá-lo ao *wrapper* escravo, para que este repasse os dados de comunicação ao escravo.

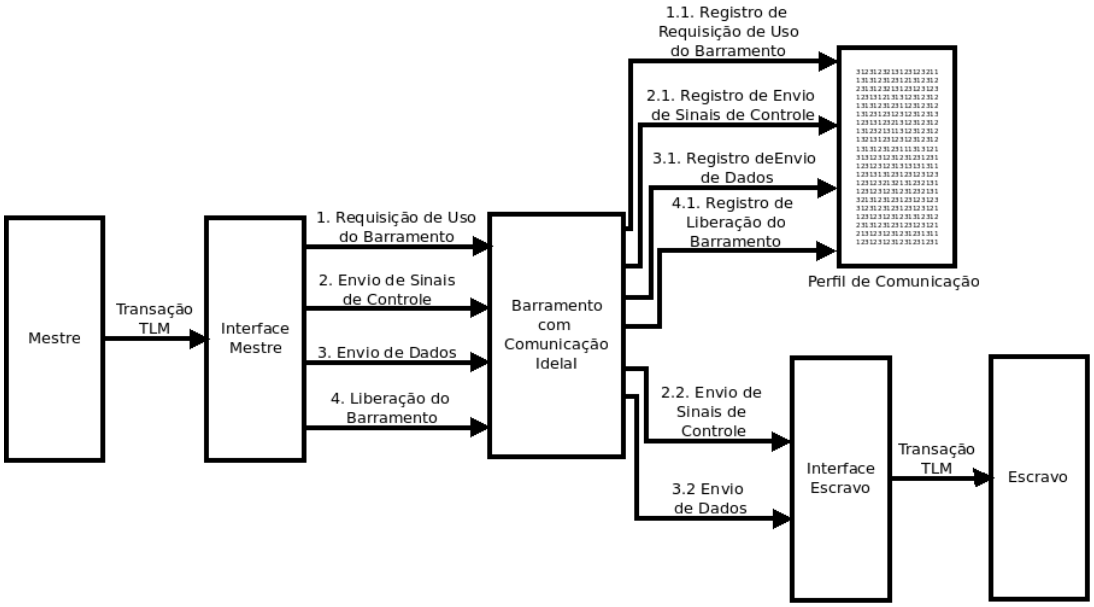


Figura 17 - Procedimento de registro de informações de comunicação em perfil de comunicação por barramento genérico proposto.

Ainda na Figura 17, pode-se observar que a transação TLM no *wrapper* mestre, será distribuída em eventos de comunicação, que representam as quatro fases de uma transação de barramento. Esses eventos serão repassados ao barramento genérico, que deverá registrá-los no perfil de comunicação, antes de repassá-los ao *wrapper* escravo. Por fim, o *wrapper* escravo agrupará os sinais de controle e dados a serem enviados em um novo pacote TLM, repassando-o ao escravo, através de nova transação TLM.

4.3.2 Fase 2 – Estimação do Desempenho de Comunicação do Sistema para um Subconjunto do Espaço de Projeto

O processo de estimação de desempenho de comunicação para um subespaço de projeto (ver fluxo na **Erro! Fonte de referência não encontrada.**) inicia com a representação do perfil de comunicação em memória, através de grafos de comunicação. Um grafo de comunicação é uma estrutura de dados que captura as informações do perfil de comunicação e que pode ter sua estrutura modificada de forma automática para embutir os efeitos de diferentes configurações do barramento e, com isso, suportar a estimação do desempenho da aplicação para um barramento real.

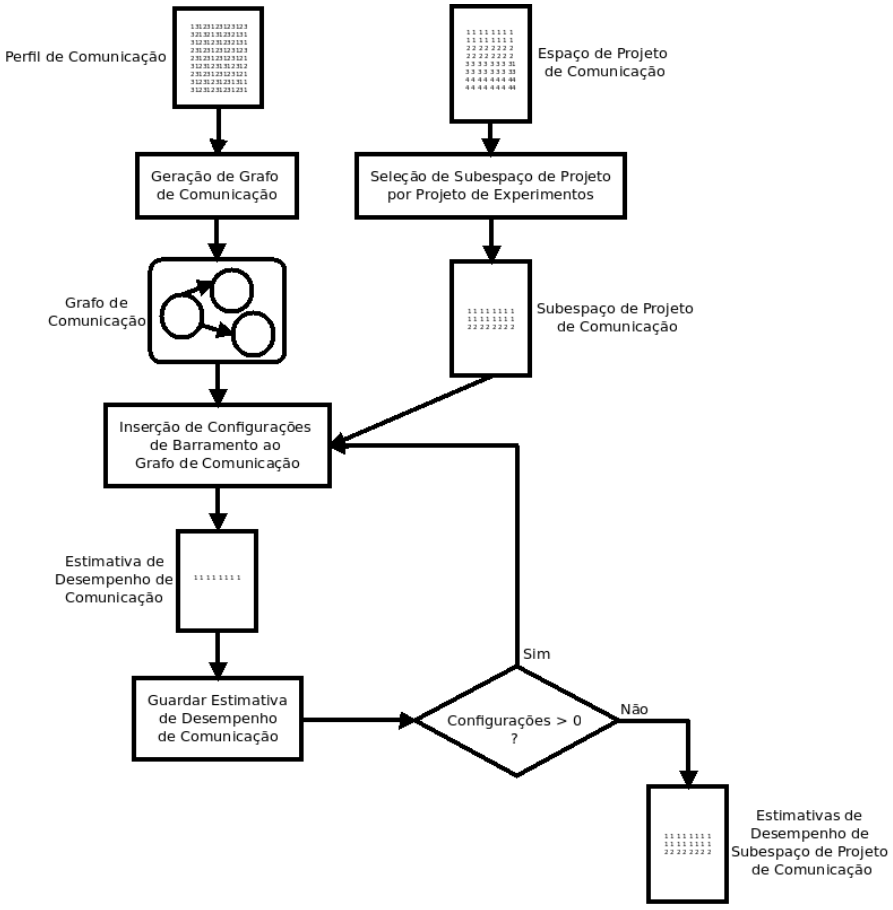


Figura 18 - Fluxo de projeto para estimação de desempenho de comunicação do sistema para subconjunto do espaço de projeto.

Em seguida, deverão ser selecionados os pontos do espaço de projeto que terão seus desempenhos estimados pela técnica baseada em grafos de comunicação. Para a seleção dos pontos, neste trabalho de tese é utilizada a técnica de Projeto de Experimentos [22], que visa obter um conjunto de pontos cujas estimativas sejam significantes para se obter um modelo de comunicação capaz de estimar o desempenho de comunicação para todos os pontos do espaço de projeto.

Assim, o grafo de comunicação inicial é modificado para embutir características de cada ponto do subespaço selecionado, disponibilizando as estimativas de desempenho de comunicação para aquele ponto do subespaço de projeto.

As subseções a seguir incluem uma descrição do modelo de grafo de comunicação, da técnica de projeto de experimentos utilizada para selecionar os pontos do conjunto de treinamento e o processo de extensão do grafo de comunicação para estimação de desempenho.

4.3.2.1 Modelo do Grafo de Comunicação

Uma vez que o perfil de comunicação foi capturado a partir de simulação com modelo de barramento genérico, o processo de exploração seguirá realizando alterações na estrutura deste perfil para estimar o desempenho do sistema sob determinadas configurações do barramento, contidas no espaço de projeto e selecionadas por técnica de projeto de experimentos (ver subseção anterior).

Um perfil de comunicação poderá conter, dependendo da aplicação, milhões de registros de comunicações entre os componentes de uma plataforma. Para inserir as características de barramentos ao perfil e estimar o desempenho da aplicação, o perfil deve ser representado através de alguma estrutura de dados, de forma que as modificações em sua estrutura sejam realizadas de forma mais eficiente.

Como discutido anteriormente, a estrutura de dados proposta é um tipo de grafo direcionado, chamada de grafo de comunicação. O modelo de grafo de comunicação proposto neste trabalho, que é baseado na abordagem proposta em [12], é definido formalmente 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 $v \in V$, e 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, representada no vértice, 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 $(u,v) \in E$, representará uma ligação que 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).

Com este modelo pode-se capturar a comunicação da aplicação na plataforma, isolando-a do processamento, realizado nos componentes da plataforma, para exploração de estruturas de comunicação.

Ao contrário do modelo proposto em [12], onde as transações são atômicas, no modelo proposto neste trabalho de tese, as transações são subdivididas em quatro eventos de transações de barramento: requisição de uso do barramento, configuração da transação, dados e liberação. Esses eventos estão diretamente relacionados às fases dos protocolos de barramentos e contidos no perfil de comunicação, como visto na seção anterior. O objetivo desta subdivisão é permitir que a abordagem de exploração possa suportar transferências com preempção e, consequentemente, abranger um espaço de projeto maior, bem como obter estimativas mais precisas, desde que se pode atribuir latências individuais para cada fase de uma transação.

Estruturalmente, o grafo conterá um fluxo de vértices (ou ramo) para cada mestre, de forma que as transferências de cada mestre serão distribuídas sequencialmente e isoladas das transferências dos demais mestres.

Para os perfis de comunicação mostrados nas Figura 14 e 15, pode-se ver os respectivos grafos de comunicação nas Figura 19 e 20, respectivamente.

Podemos ver na parte superior da Figura 19, os DFDIs para as três aplicações ilustradas na Figura 14. Logo abaixo dos DFDIs, ainda na Figura 19, vemos os respectivos grafos de comunicação que poderão ser gerados a partir dos perfis de comunicação apresentados na Figura 14. Cada grafo de comunicação possui em seus vértices três campos: identificador de mestre (*m*), número de ordem da transação (campo *t*) e respectivos eventos (campo *e*).

A disposição dos valores do campo *t* ao longo dos vértices do grafo dependerá de como as prioridades de arbitragem foram alocadas para cada mestre. Na aplicação com apenas um mestre de barramento, grafo da Figura 19(a), o campo de contagem de transação inicia em zero e, a cada nova transação, esse valor sofre incremento.

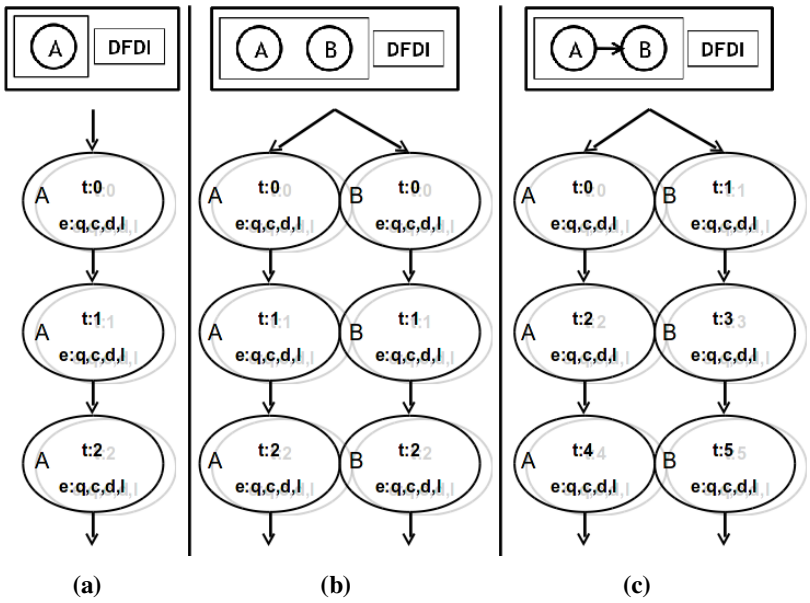


Figura 19 - Grafos de comunicação para os perfis de comunicação apresentados na Figura 14.

Já para uma aplicação com dois mestres de mesma prioridade, grafo central da Figura 19, o valor do campo *t* será igual para cada transação dos mestres A e B, como é mostrado nos dois fluxos de vértices do grafo, sinalizando, assim, transações em paralelo. Observe que o grafo representa as sequências de transações de cada mestre com ramos diferentes: cada mestre terá seu próprio ramo de vértices.

Por fim, o grafo da Figura 19(c), mostra a aplicação em que o mestre A se comunica com o B (representado por um arco dirigido com orientação saindo do vértice A e dirigindo-se para B), onde, consequentemente, o mestre A deverá ter maior prioridade de uso do barramento sobre B. Neste grafo, os campos *t* dos vértices, que representam as sequências de transações dos mestres A e B, terão seus valores incrementados seguindo uma contagem contínua entre eles, iniciando em A, que é o mestre de maior prioridade, como pode ser visto no ramo à esquerda do grafo na Figura 19(c). Nesta figura vemos ainda que o campo *t* possui valor zero no primeiro vértice do ramo de A e segue incrementando nos fluxos de B e A, alternadamente. A contagem alternada de transações entre vértices de ramos diferentes modela o conceito de arbitragem no barramento, onde um mestre ao finalizar sua transferência concede o barramento para que outros possam utilizá-lo.

A Figura 20 apresenta exemplo de uma parte de um grafo de comunicação para o perfil de comunicação da Figura 15. Seguindo o modelo proposto, teremos três fluxos de vértices no

grafo, um para cada mestre. Verifica-se que a contagem da sequência de transações, representada no grafo da Figura 20, segue a mesma sequência do perfil de comunicação: inicia no primeiro vértice do ramo de A e sofre incremento a cada novo vértice do ramo de A e a cada vértice de B e C, conjuntamente. Observe que, seguindo o modelo proposto para grafos de comunicação, a disposição dos vértices e a contagem da sequência de transações caracterizam diferenças de prioridade de uso do barramento entre o mestre A e os o mestres B e C, sendo que o mestre A possui maior prioridade sobre os demais.

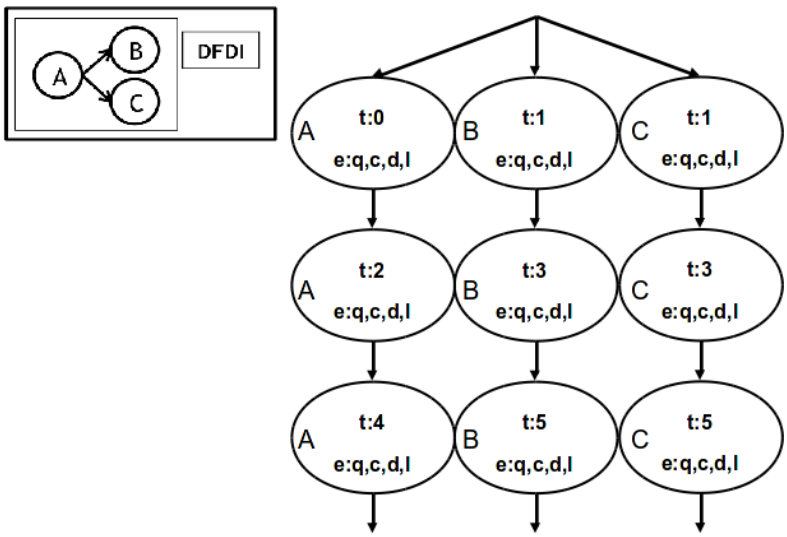


Figura 20 - Grafos de comunicação para os perfis de comunicação apresentados na Figura 15.

A subseção a seguir apresenta o algoritmo de geração dos grafos de comunicação a partir de um perfil de comunicação.

4.3.2.2 Algoritmo para Geração de Grafo de Comunicação

O pseudo-algoritmo para geração do grafo de comunicação (ver Figura 21) é iniciado com a criação de um grafo vazio (linha 1), e segue, na linha 2, com a leitura dos registros de eventos de transações contidos no perfil de comunicação (linha 2).

Para cada registro, verifica-se, inicialmente, na linha 3, se existe um ramo para o mestre que originou esse evento de comunicação. Caso o ramo não tenha sido criado, na linha 4 será criado para o mestre do registro corrente.

Na linha 5 verifica-se se o evento de comunicação é tipo de requisição, e se confirmado, será criado, na linha 6, um novo vértice de comunicação no ramo referente ao mestre que requisitou o uso do barramento. Os registros que contêm os demais eventos de comunicação serão adicionados ao último vértice do ramo do respectivo mestre (linha 8).

As iterações continuam até que não haja mais registros no perfil.

```
1:  graph = new Graph();
2:  for each register in profile:
3:     if register.master has not branch:
4:  graph.newBranch(register.master);
5:     if register.event is request:
6:        graph.addNewVertice(register);
7:     else:
8:        graph.addNewEvent(register);
```

Figura 21 - Fluxo principal do algoritmo de geração de grafos de comunicação.

Para dar início à estimação de comunicação, é selecionada uma configuração de barramento, contida no espaço de projeto, e o grafo de comunicação é modificado para considerar esta configuração no perfil de comunicação. Esse processo deve ser repetido para cada configuração de barramento do subespaço de projeto, de forma que essas configurações e respectivas estimativas de desempenho sejam utilizados na obtenção do modelo de comunicação da aplicação.

A subseção a seguir detalha o processo de seleção das configurações de barramento que definem o subespaço de projeto.

4.3.2.3 Seleção do Conjunto de Treinamento por Projeto de Experimentos

A seleção dos elementos que irão compor o subespaço de projeto pode ser realizada de várias maneiras, porém técnicas como amostragem aleatória não garantem uma amostra distribuída e amostragem baseada em variância não permite coletar o conjunto completo dos dados da amostra, de forma que os conjuntos selecionados podem não ser suficientes para se obter um modelo de comunicação capaz de estimar a comunicação da aplicação.

Assim, como sugerido em [14], utiliza-se, nesta tese de doutorado, a técnica de Projeto de Experimentos [22]. Projeto de experimentos, também conhecido na estatística como Experimento Controlado, refere-se ao processo de planejar, projetar e analisar um experimento de forma que conclusões válidas e objetivas podem ser extraídas efetivamente e eficientemente. De uma forma geral, o objetivo do uso dessas técnicas é coletar a quantidade máxima de informações relevantes com consumo mínimo de tempo e recursos, e obter soluções ótimas mesmo quando é impossível ter um modelo matemático funcional (determinístico) [22][31][32][33]. Busca-se assim, através da aplicação da técnica de projeto de experimentos, selecionar um subconjunto do espaço de projeto, cujos pontos terão desempenhos de comunicação estimados, utilizando os grafos de comunicação. Essas configurações, bem como respectivos desempenhos estimados, são utilizadas para obter um modelo de comunicação capaz de estimar o desempenho da aplicação para todas as configurações do espaço de projeto.

A técnica de projeto de experimentos adotada neste trabalho de tese é conhecida como Hipercubo Latino Uniforme de Audze-Eglais [35][36].

O método de Audze-Eglais é baseado na seguinte analogia com a física:

> Suponha um sistema composto por pontos de unidade de massa que exercem forças repulsivas entre si fazendo com que o sistema tenha energia potencial. Quando os pontos são liberados a partir de um estado inicial, eles se movem. Esses pontos irão atingir o equilíbrio quando a energia potencial das forças repulsivas das massas é mínimo. Se a magnitude das forças repulsivas é inversamente proporcional ao quadrado da distância entre os pontos, então a minimização da equação abaixo irá produzir um sistema de pontos distribuídos tão uniforme quanto possível.

$$U = \sum_{p=1}^{n} \sum_{q=p+1}^{n} \frac{1}{L_{pq}^2} \tag{22}$$

onde U é a energia potencial e L_{pq} é a distância entre os pontos p e q, sendo que $p \neq q$, e n é o número de pontos do espaço de projeto.

Os pontos do espaço de projeto são dados por valores dos parâmetros de configuração do barramento, de forma que cada ponto será uma combinação dos valores que esses parâmetros

podem receber. O método de Audze-Eglais pode ser aplicado a este espaço de projeto, desde que sejam considerados os intervalos (as distâncias) entre os valores de cada parâmetro de configuração de barramento, e que esses intervalos sejam tomados conjuntamente, de forma a minimizar a função objetivo.

Por exemplo, para selecionar três pontos de um dado espaço de projeto, mostrado na Tabela 2, poderíamos, inicialmente, representá-lo graficamente, através de um gráfico de dispersão tridimensional, como pode ser visto na Figura 22. E, em seguida, tenta-se traçar um triângulo, de forma que seu perímetro seja máximo. Ainda na Figura 22, o leitor pode notar que o triângulo, com linhas pontilhadas, tem perímetro máximo em relação aos outros triângulos que podem ser esboçados no gráfico. Neste caso, a função objetivo seria inversamente proporcional ao perímetro do triângulo.

Tabela 2 - Exemplo de espaço de projeto de estrutura de comunicação contendo três parâmetros de configuração: Request Cycle, Pipeline Cycle e Bus Width.

Request Cycle (rc)	Pipeline Cycle (pc)	Bus Width (bw)
1	1	8
1	1	16
1	1	32
1	2	8
1	2	16
1	2	32
2	1	8
2	1	16
2	1	32
2	2	8
2	2	16
2	2	32
3	1	8
3	1	16
3	1	32
3	2	8
3	2	16
3	2	32

A minimização da equação (22) pode ser realizada por alguma técnica de otimização ou por verificação de todas as combinações possíveis, como foi apresentado anteriormente através do gráfico da Figura 22. Utilizar a segunda abordagem pode ser inviável, pois a busca por cada combinação possível em espaços de projeto com muitos pontos tem um alto custo computacional. Assim, neste trabalho de tese, foi utilizada a ferramenta GPRSKit [41], que utiliza técnicas de programação genética para minimizar a equação, e disponibiliza, como saída, as configurações de barramento encontradas no processo de otimização da equação.

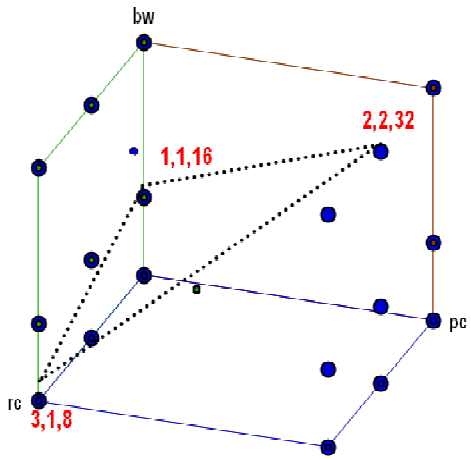


Figura 22 - Esboço gráfico de espaço de projeto, contendo três parâmetros de configuração de barramento, com três pontos com distância

Uma vez que as configurações de barramento foram selecionadas, através de método de Audze-Eglais, compondo o conjunto de treinamento, o grafo de comunicação deverá ser estendido para cada uma dessas configurações, de forma a incorporar seus efeitos de comunicação, como por exemplo, latências e contenção, na estimativa do desempenho da aplicação. A subseção a seguir detalha o processo de extensão do grafo de comunicação.

4.3.2.4 Estimação de Desempenho de Comunicação para um Subespaço de Projeto

Como mencionado, a estimação do desempenho para um subespaço de projeto é realizado utilizando grafos de comunicação, que são estruturas de dados criadas para alocar em memória principal o perfil de comunicação de uma aplicação. O grafo de comunicação, assim como o perfil de comunicação, representa eventos e sequências das transações dos mestres, para uma dada aplicação, em um cenário com modelo de comunicação ideal.

Para que os grafos de comunicação possam ser utilizados na exploração de arquiteturas de comunicação, para diferentes configurações de barramento, é necessário que os efeitos dessas características sejam adicionados ao grafo, como, por exemplo, latências provindas de protocolos de negociação de transferência, mecanismo de arbitragem, entre outros.

As características do barramento, que terão seus efeitos adicionados ao grafo de comunicação para estimação de desempenho das aplicações, 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 de relógio, suporte a transferências com *pipeline*, informações de latências das transferências (quantidade de ciclos necessários para completar as fases de uma transferência), o tipo 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, sendo um perfil para cada modelo de barramento. A **Erro! Fonte de referência não encontrada.** apresenta um exemplo de um perfil de barramento.

```
1:  PERIOD=5,6,10:NS
2:  WIDTH=1,2,4
3:  REQUEST=0
4:  CONTROL=1
5:  DATA=1
6:  RELEASE=0
7:  PIPELINE=YES
8:  SIZE=1
9:  TYPE=LOCKED,UNLOCKED
10: MASTERS=4
11: PRIORITIES=M1:1,M2:2:3:4,M3:2:3:4,M4:2:3:4
12: WIDTHS=M1:4,M2:4,M3:4,M4:4
```

Figura 23 - Exemplo de Perfil de Barramento.

Seguem descrições do tipo de configuração que cada linha da **Erro! Fonte de referência não encontrada.** representa e dos respectivos valores que estão assumindo:

1) Período entre ciclos de relógio: os valores 5,6 e 10 referem-se ao tempo medido em nanossegundos (diretiva NS);
2) Largura de barramento de dados: os valores 1, 2 e 4 referem-se ao número de palavras de 8 bits que o barramento de dados assumirá. Neste caso, será 8, 16 e 32 bits.
3) Número de ciclos de relógio entre a requisição de uso do barramento e a resposta;
4) Número de ciclos de relógio para envio de sinais de controle;
5) Número de ciclos de relógio para realizar uma transferência de leitura ou escrita;
6) Número de ciclos de relógio para solicitar a liberação de uso do barramento;
7) Configuração de suporte à pipeline: os valores que essa configuração pode assumir são: YES e/ou NO;
8) Tamanho das transferências: quando a forem consideradas transferências simples, esse parâmetro assumirá o valor 1. Caso sejam consideradas transferências do tipo rajada, a configuração será dada em função do tamanho da rajada (e.g. 4, 8 ou 16).
9) Tipo de transferência: serão consideradas transferências sem preempção (diretiva LOCKED) ou com preempção (diretiva UNLOCKED);
10) A quantidade de mestres no barramento;
11) As prioridades de arbitragem que cada mestre assumirá: a definição da(s) prioridade(s) que um mestre pode assumir é dada pelo nome do mestre seguido pela(s) prioridade(s). Desta forma, a notação M2:2:3:4 diz que o mestre cujo nome é M2 poderá assumir a segunda, terceira e quarta maiores prioridades; e
12) Tamanho de palavra de cada mestre: a definição do(s) tamanho(s) de palavra(s) de um mestre é dada pelo nome do mestre, seguido pelo(s) tamanho(s) de palavra(s) que o mestre poderá assumir, indicado(s) pelo número de palavras de 8 bits. Por exemplo, a definição M1:4 indica que o mestre M1 terá tamanho de palavra igual a quatro palavras de 8bits.

Após a incorporação das características do barramento, o novo grafo, chamado de grafo de comunicação estendido, permitirá estimar o desempenho do sistema considerando as respectivas características da comunicação retiradas do perfil de comunicação, bem como as diferentes características do barramento.

Em termos gerais, o processo de extensão de um grafo de comunicação envolve quatro etapas (ver fluxo na **Erro! Fonte de referência não encontrada.**):

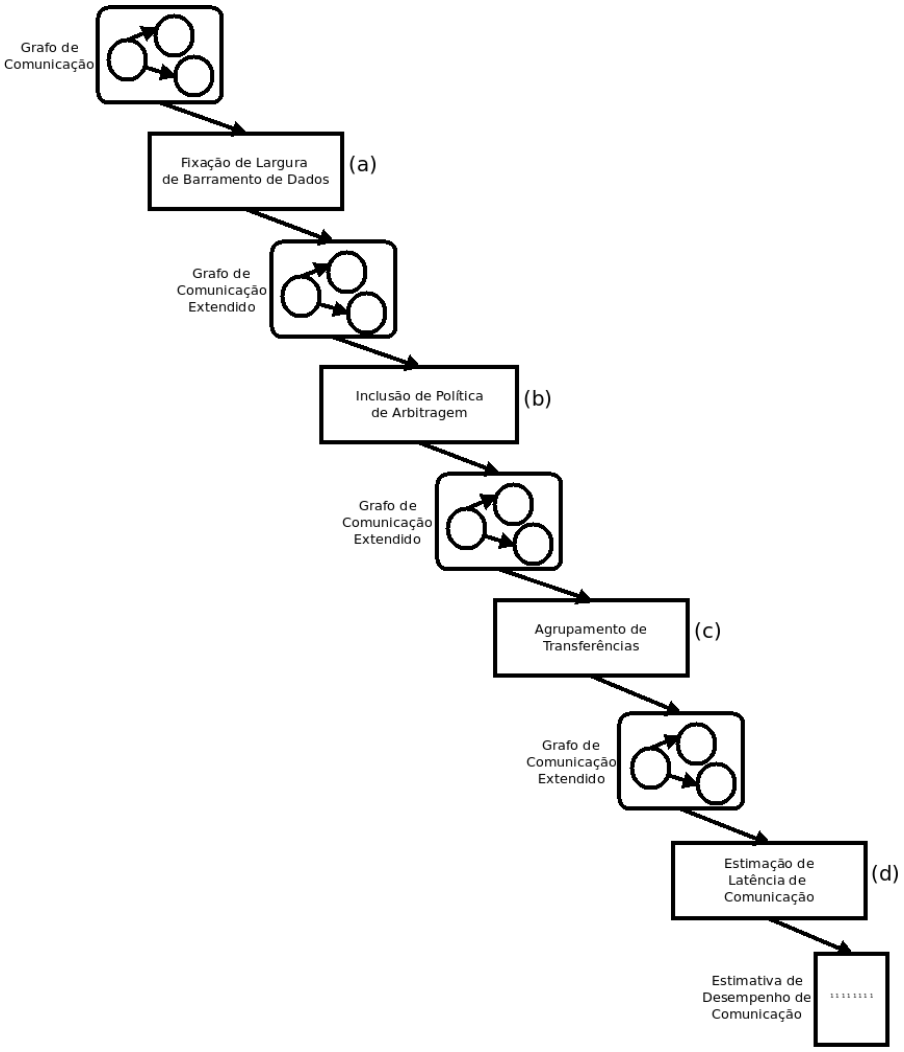


Figura 24 - Fluxo com sequência de etapas para extensão de um grafo de comunicação.

a) **Fixação de largura do barramento de dados**: modifica o número de vértices para que a quantidade de transferências seja restringida de acordo com uma largura fixa do barramento de dados.

b) **Inclusão de política de arbitragem**: os ramos do grafo convergirão para que seja embutida uma política de arbitragem, caso haja mais de um mestre de barramento.

c) **Agrupamento de transferências**: agrupamento de transferências seguidas de um mesmo

mestre em um único vértice, para minimizar o tamanho do grafo.

d) **Estimação de latências**: cálculo das latências das transferências 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.

Estas etapas serão descritas com mais detalhes a seguir.

a) Fixação de largura do barramento de dados

O grafo de comunicação gerado a partir do perfil de comunicação, representa as transferências de um barramento ideal, onde os dados transferidos podem ser de tamanhos variados, dependendo apenas da capacidade de transferência de dados de cada mestre. Por exemplo, no mesmo barramento poderíamos ter transferências de números inteiros (32 bits), inteiros longos (64 bits) ou até mesmo uma *string* completa.

Em barramentos reais este comportamento não é permitido, desde que a largura do barramento de dados é definida, impondo assim transferências de uma única quantidade de bits. Em barramentos reais, mestres com tamanhos de palavra maiores que a largura do barramento têm a transferência dividida em duas ou mais transferências com larguras de dados menores (iguais ou menores à largura do barramento de dados) [45]. Por exemplo, um mestre com tamanho de palavra de 64 bits para transferir em um barramento de 32 bits teria que dividir uma palavra através de duas transferências de 32 bits.

Já para mestres com tamanho de palavra inferior à largura do barramento, a transferência de uma palavra pode ser realizada com apenas uma transferência de barramento, sendo que apenas os bits do barramento de dados que contém a palavra transmitida serão aproveitados. Um mestre deste tipo pode ainda agrupar várias transferências em uma única transferência de barramento, através de multiplexação nos bits do barramento de dados, como é o caso do barramento PCI [7], o qual permite transferência de 64 bits com duas palavras 32 bits embutidas. Assim, um mestre com palavra de 8 bits, por exemplo, poderia transmitir oito palavras com apenas uma transferência em um barramento de 64 bits. Naturalmente, uma transferência de 64 bits, com agrupamento de oito palavras de 8 bits, teria maior desempenho em relação a oito

transferências de 8 bits, visto as latências adicionais dadas pela configuração de cada transferência e pelos dispositivos escravos.

Para incorporar uma largura fixa do barramento de dados, o grafo de comunicação deve ser modificado para suportar a largura de barramento definida no perfil do barramento. Para um mestre de 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.

b) 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. Foi mencionado 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 acontece em um modelo de barramento real.

Para um mecanismo de arbitragem por prioridade fixa, por exemplo, a convergência é iniciada a partir dos vértices dos ramos de maior prioridade. O grafo de comunicação da Figura 19(b), por definição da própria aplicação, não inclui prioridades de uso do barramento entre os mestres, o que pode ser visto pelos pares de vértices com mesma contagem de transações (transferências em paralelo). Assim, ao definir as prioridades dos dois mestres, os ramos do grafo terão que convergir de forma que o mestre de maior prioridade iniciará a disposição de vértices no ramo final. As Figura 25(a) e (b) apresentam dois possíveis fluxos, gerados após a convergência dos fluxos de vértices entre os mestres A e B. Observe que a Figura 25(a) ilustra um novo fluxo no qual é priorizado mestre A enquanto que na Figura 25(b), o mestre B é priorizado. Para o exemplo apresentado na Figura 19(c), onde existe uma definição implícita de prioridades, onde o mestre A possui maior prioridade. Neste caso, os fluxos convergem de acordo com a contagem da sequência de vértices, como pode ser visto na Figura 25(c). Nas Figuras 25 (a), (b) e (c), os vértices dos mestres A e B permaneceram à esquerda e à direita, respectivamente, após a convergência para ilustrar que não sofrem modificações. Na convergência, o conjunto de arcos, que é o único elemento que sofre modificação, deverá incluir apenas os arcos relacionados ao fluxo gerado.

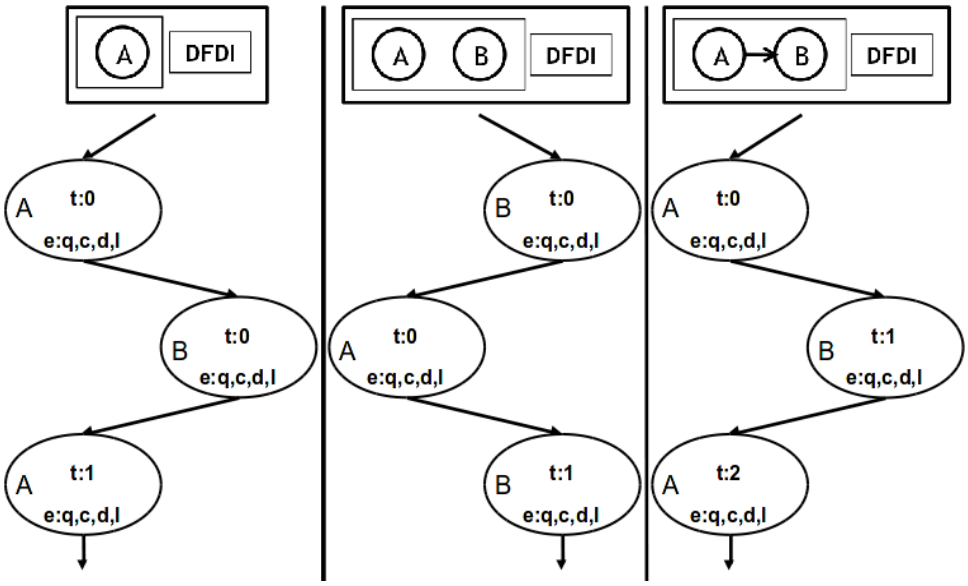


Figura 25 - Grafos de comunicação estendidos, com suporte de arbitragem, para os grafos de comunicação da Figura 19.

Formalmente, o processo de convergência dos ramos de um grafo de comunicação é descrito a seguir:

Seja $G=(V,E)$ um grafo de comunicação extraído da simulação 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 seguindo de acordo com as prioridades definida por P.
iv) Remove-se o próximo vértice $v_{jz} \in M_j$, onde $M_j \in M'$ e M_j o é conjunto de vértices do mestre de maior prioridade, e adiciona-se v_{jz} a V_1. Caso não existam vértices em M_j, este conjunto será removido de M' e o próximo elemento de M' passará a ser o M_j.
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_1, desde a aresta $(v_{jz}, v_{iu}) \in E'$. Caso não existam mais vértices em M_i, este conjunto será removido de M' e 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 de tese, 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. Alguns barramentos comerciais, como o AMBA, oferecem transferências com e sem bloqueio.

O algoritmo apresentado anteriormente suporta transferências com bloqueio. Para suportar transferências sem bloqueio na convergência de ramos, mestres que iniciam transferências e são desalocados do barramento por terem menor prioridade, terão vértices adicionais representando as transferências interrompidas. Estes novos vértices, no entanto, terão apenas os eventos que foram concluídos na transação. Em termos práticos, o grafo incluirá no ramo final um novo vértice com os eventos concluídos para a transferência que sofre preempção, e em seguida o vértice para a transferência do mestre que recebeu concessão.

A Figura 26 ilustra o suporte a transferências sem bloqueio para os exemplos das Figura 19(b) e (c).

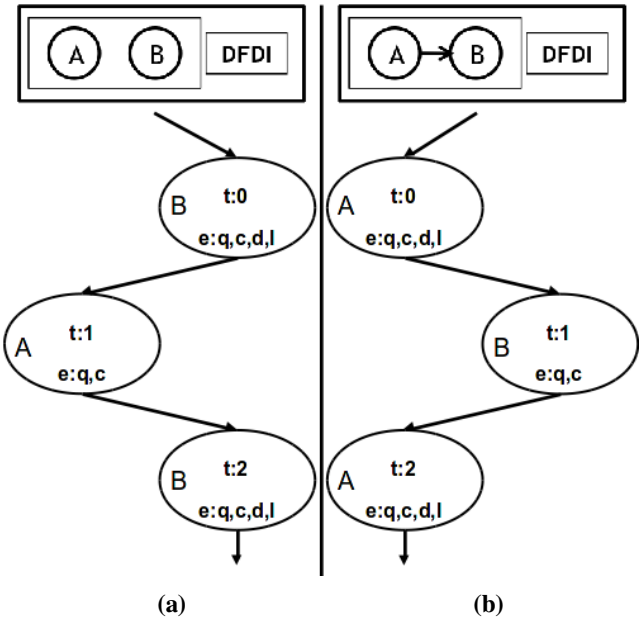


Figura 26 - Grafos de comunicação estendidos com transferências sem bloqueio para os grafos de comunicação das Figura 19(b) e (c).

Na Figura 26(a), verifica-se que o mestre B possui maior prioridade de uso do barramento, pois realiza transferência em *t=0*. Em *t=1*, o mestre A inicia uma transferência, porém, sofre preempção, como mostra a lista de eventos do vértice - apenas foram concluídos os eventos de requisição (*q*) e controle (*c*). Em *t=2*, o mestre B realiza outra transferência. Para o exemplo da Figura 26(b), as prioridades de uso do barramento são implícitas à aplicação, como mostra o DFDI, sendo que A possui maior prioridade sobre B. Assim, em *t=0*, o mestre A realiza

uma transferência. Em *t=1*, o mestre B inicia uma transferência, porém é desalocado do barramento para o mestre A possa realizar outra transferência (*t=2*).

c) 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. Os vértices nestas sequências poderão ainda ter a mesma disposição dos eventos que representam as fases de protocolos de barramentos. As diferenças surgem apenas no valor na contagem da sequência de transferências (campo t).

De acordo com Lahiri et al. em [12], uma das formas de aumentar o desempenho do processo de estimação, utilizando de grafos de comunicação se dá através do agrupamento dos vértices, reduzindo-se assim o tamanho do grafo. 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, terá dois tipos de agrupamentos. O agrupamento acontece em duas fases. Na primeira fase, 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. A Figura 27 ilustra estas duas fases no agrupamento de transferências.

A Figura 27(a) mostra um grafo de comunicação com um conjunto de vértices representando uma sequência de transferências para um mestre A. Todos os vértices desta sequência possuem a mesma disposição de eventos das fases de uma transferência: requisição, controle, dados e liberação. Como se trata de um conjunto transferências simples, o agrupamento para este grafo se dá pela contagem do total de vértices da sequência, que neste caso são 1232 vértices. E em seguida, tem-se a criação de um novo vértice que conterá a informação obtida a partir da contagem, como pode ser visto na Figura 27 (b). Nesta figura, observa-se a existência de apenas um vértice, com os seguintes campos: número da transferência inicial (campo *t*), total de transferências do agrupamento (campo *T*) e o conjunto de eventos de cada transferência do

agrupamento. Além disto, o agrupamento informará também o total de ocorrências de cada evento em cada transferência que representa, como pode ser visto nos campos *q*, *c*, *d*, *l* com valores iguais a 1. O grafo da Figura 27 (b) descreve a realização de 1232 transferências, e em cada uma delas, os eventos *q*, *c*, *d* e *l* ocorreram uma única vez

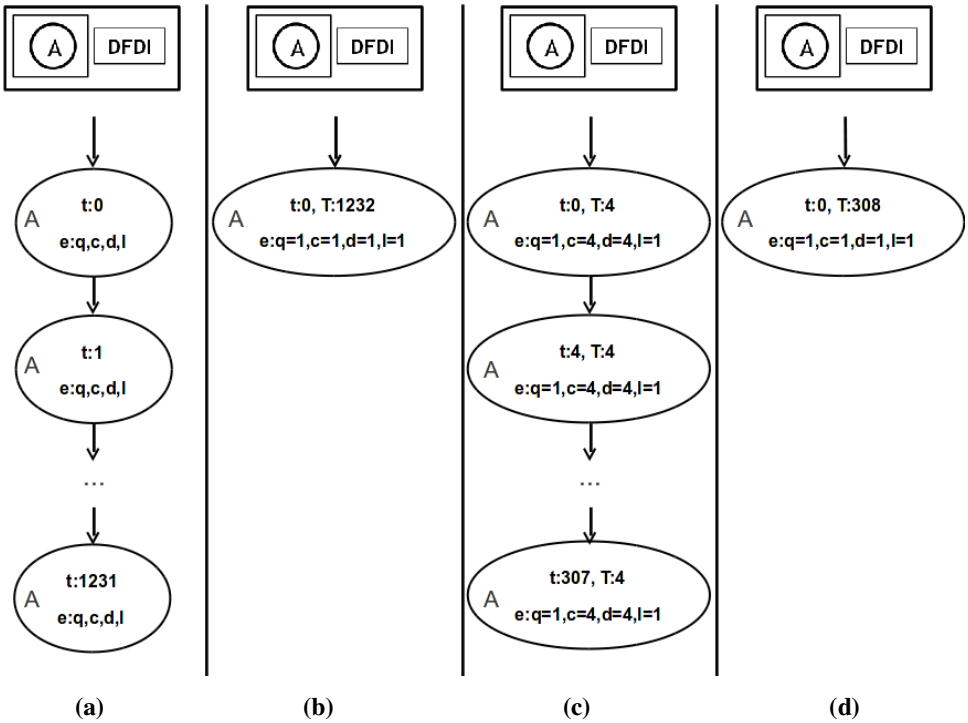


Figura 27 - Grafo de comunicação e grafo de comunicação estendido para aplicação com único mestre de barramento.

Para transferências em rajada, no primeiro nível haverá o agrupamento das transferências para caracterizar as rajadas, como pode ser visto na Figura 27 (c). Nesta figura, cada vértice do grafo conterá os mesmos campos de um agrupamento de transferências simples, com as seguintes diferenças:

- campo *T*: contará a quantidade de transferências simples que compõem a rajada, ou seja, o tamanho da rajada;
- campos *q*, *c*, *d*, *l*: conterão a quantidade total de ocorrências das transferências que o vértice representa.

A interpretação de cada um dos vértices do grafo de comunicação da Figura 27 (c) sugere rajada de tamanho igual a 4 ($T=4$), com uma requisição de uso do barramento ($q=1$), quatro eventos de controle ($c=4$), quatro envios de dados ($d=4$) e uma liberação ($l=1$).

Na segunda etapa, os vértices que representam sequências de rajadas serão agrupados. A Figura 27 (d) apresenta um grafo de comunicação com apenas um vértice, que é o resultado do agrupamento dos 308 vértices do grafo da Figura 27 (c).

A descrição formal do agrupamento de transferências simples e a primeira etapa do agrupamento de rajadas, que são bastante semelhantes, são apresentados a seguir:

> Seja $G=(V,E)$ um grafo de comunicação e v_1, v_2, v_3 .., v_n as transações tal 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 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 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:

Seja $G'=(V',E')$ um grafo de comunicação e $v_{1'}, v_{2'}, v_{3'}$ .., $v_{n'}$ transações do tipo rajada tal 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 de *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 $k=1,2,3,...,n$, contêm 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, 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.

d) Estimação de Latências

Por fim, após o agrupamento de vértices, pode-se estimar a latência das transferências e, consequentemente, da aplicação considerando determinadas características de um barramento.

O processo de estimação baseia-se na contagem do tempo necessário para realização das transferências representadas pelos vértices do grafo de comunicação estendido final.

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 abaixo:

$$P = \frac{1}{F} \tag{23}$$

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.

Em seguida, contabiliza-se o tempo necessário para realizar as transferências representadas por cada vértice, de acordo com seu número de ciclos. Supondo um barramento com frequência de operação de 100MHz, teríamos:

$$P = \frac{1}{(1.10^8)}s \quad \therefore \quad P = 1.10^{-8} \quad \therefore \quad P = 10ns$$

onde *10 ns* é o tempo necessário para ocorrência de cada ciclo de barramento. Com essa informação, e as quantidades de ciclos necessárias 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:

$$t_{(vértice)} = (q + c + d + l).T.P \tag{24}$$

onde q, c, d e l são as quantidades de ciclos para que os eventos de transferência de barramento possam ser executados, T é total de transferências representadas pelo vértice e P é período entre ciclos. Para estimar o tempo necessário para as transferências do grafo da Figura 27(b), por exemplo, o calculo é dado por:

$$t_{(vértice)} = (1+1+1+1).1232.10ns \quad \therefore \quad t_{(vértice)} = 49280ns$$

onde 49280 ns é tempo necessário para que as transferências do vértice possam ser executadas. Observe que como o grafo possui apenas um vértice, dado pelo processo de agrupamento de vértices, o tempo estimado para este vértice também é o tempo necessário para todas as transferências daquela aplicação, uma vez que esta possui apenas um mestre.

Em grafos de comunicação com mais de um vértice, a estimativa de desempenho da aplicação é realizada através do cálculo do tempo para 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. Podemos ver na Tabela 3 a disposição dos eventos de transferência para rajadas de tamanho 4 e 8, respectivamente. Observe que em ambas as rajadas, o pipeline inicia a partir do terceiro ciclo da transação (colunas 3 e 11), quando há o alinhamento entre as fases de dados e controle da primeira e segunda transferência, respectivamente. As fases de dados e controle, para as transferências que compõem uma rajada, sempre serão alinhadas exatamente *n - 1* vezes, onde *n* é número de transferências simples de uma rajada. Para uma rajada de tamanho 4, por exemplo, teremos 3 alinhamentos, entre as fases de dados e controle, e para uma rajada de tamanho 8, teremos 7 alinhamentos, como pode ser visto na Tabela 3.

Tabela 3 - Sequências de eventos de barramento para transferências por rajada, com suporte a *pipeline*, de tamanhos 4 e 8, respectivamente.

Transferência	**Ciclo**																		
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**
1	q	c	d						q	c	d								
2			c	d							c	d							
3				c	d							c	d						
4					c	d	l						c	d					
5														c	d				
6															c	d			
7																c	d		
8																	c	d	l

Observando ainda a Tabela 3, podemos perceber outro padrão relacionado aos eventos das transferências em rajadas. As fases de requisição e controle da primeira transferência (colunas 1, 2, 9 e 10) e as fases de dado e liberação da última transferência (colunas 6, 7, 18 e 19), que compõem as rajadas, não são alinhadas com outras transferências dentro de um *pipeline*.

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

$$t_{(vértice)} = (q + c + d + l + (n-1).c).T.P \tag{25}$$

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 período entre ciclos. O termo $(q+c+d+l)$ contabiliza as fases não alinhadas, e $(n-1).c$ aos alinhamentos entre fases de dados e controle. 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*. Desta forma, podemos deduzir uma nova equação para estimar a latência de vértices com transferências em rajada e suporte à pipeline:

$$T_{(vértice)} = (q + c + d + l + (n-1).c).T.P \tag{25}$$

$$\therefore$$

$$T_{(vértice)} = (q + 2.c + l + n.c - c).T.P$$

$$\therefore$$

$$T_{(vértice)} = (q + l + (n+1).c).T.P \tag{26}$$

Para finalizar, a latência total estimada para todas as transferências representadas pelo vértice contido no grafo exposto na Figura 27(d) é dada por:

$$T_{(vértice)} = (1+1+5).208.10ns \quad \therefore \quad T_{(vértice)} = 21550ns$$

O processo de estimação de desempenho de comunicação através da técnica de grafos de comunicação deve ser realizado para cada configuração do subespaço de projeto. Com estas

estimativas, o processo de busca de um modelo de comunicação capaz de estimar o desempenho de comunicação para todas as configurações de barramento do espaço de projeto é iniciado. A próxima subseção apresenta com maiores detalhes o modelo proposto de comunicação, bem como as técnicas de busca desenvolvidas neste trabalho de tese.

4.3.3 Fase 3 – Geração do Modelo de Comunicação

Neste trabalho de tese, um modelo de comunicação deve ser capaz de estimar com precisão, e de forma automática, o desempenho da comunicação de uma aplicação para todos os pontos do espaço de projeto de comunicação.

Tradicionalmente, abordagens, como as apresentadas em [61][62], utilizam modelos matemáticos determinísticos para relacionar as configurações de barramento com o desempenho de comunicação da aplicação. Porém, observa-se que nem sempre é possível obter uma função determinística que determine precisamente essas relações, apresentando assim diferenças entre as estimativas e os valores reais do desempenho da comunicação.

Considerando assim os erros de estimativas, para este trabalho de tese foi adotado o uso da técnica de análise de regressão na definição do modelo de comunicação, que é uma técnica estatística que representa a dependência entre duas ou mais variáveis [117], de forma a estabelecer uma relação probabilística. A grande vantagem do uso de modelos de regressão lineares (MRL) é a possibilidade de controlar os erros de estimativas, além de permitir a validação formal do modelo através de inferência estatística. Este tipo de análise também permite um melhor entendimento da relação entre variáveis, através de análise das variáveis do modelo que têm maior influência nas estimativas, além de quantificar e qualificar essa influência. Modelos de regressão linear já foram utilizados em diversas abordagens para estimação de desempenho de processadores e memórias cache, como, por exemplo, nas técnicas apresentadas em [88][89][90].

As seções a seguir apresentam a definição do modelo de comunicação como um modelo de regressão linear e a abordagem para geração deste modelo para estimar o desempenho de comunicação de uma aplicação.

4.3.3.1 Modelo de Comunicação

Assim como a grande maioria das análises estatísticas, o objetivo de regressão é resumir, através de um modelo estatístico, as relações entre as variáveis de forma simples e útil [40]. Em alguns problemas, pode também ser utilizada para especificar como uma das variáveis, neste caso, chamada de variável resposta ou dependente, varia em função da mudança dos valores das demais variáveis da relação, chamadas de variáveis preditoras, regressoras ou sistemáticas.

Neste trabalho de tese, um modelo linear irá relacionar o desempenho de comunicação da plataforma, que é a variável resposta, com as configurações de barramento do espaço de projeto de comunicação, que são as variáveis preditoras.

O modelo estatístico de regressão linear consiste de duas funções, uma para média e uma para variância, definidas, respectivamente, pelas equações a seguir:

$$E(Y \mid X = x) = \beta_0 + \beta_1 x \tag{27}$$

$$Var(Y \mid X = x) = \sigma^2 \tag{28}$$

onde os parâmetros na função da média são o intercepto β_0, que é o valor da média $E(Y|X=x)$ quando x é igual a zero, e a inclinação da reta β_1, que é a taxa de mudança em $E(Y|X=x)$ para uma mudança de valores de X, como ser visto na Figura 28.

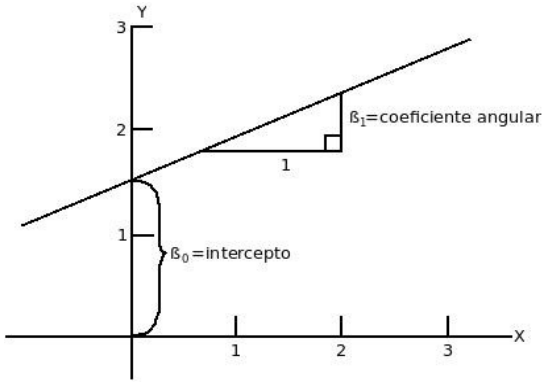


Figura 28 - Gráfico da equação de reta $E(Y|X=x)=\beta_0 + \beta_1 x$

Variando esses parâmetros, é possível obter todas as equações de reta. Na maioria das aplicações, os parâmetros são desconhecidos e devem ser estimados utilizando os dados do problema. Desta forma, assume-se que a função de variância é constante, com um valor positivo σ^2 que normalmente é desconhecido.

Diferentemente de modelos matemáticos, que são determinísticos, modelos de regressão linear consideram os erros entre os valores observados e os valores estimados pela equação da reta. Assim, por causa da variância $\sigma^2>0$, os valores observados para a i-ésima resposta y_i serão tipicamente diferentes dos valores esperados $E(Y|X=x_i)$. Para considerar a diferença entre os dados observados e os esperados, tem-se o conceito de *erro estatístico*, ou e_i, para o caso i definido implicitamente pela equação:

$$y_i = E(Y \mid X = x_i) + e_i \tag{29}$$

, ou explicitamente, por:

$$e_i = y_i - E(Y \mid X = x_i) \tag{30}$$

Os erros e_i dependem dos parâmetros desconhecidos na função da média e são variáveis aleatórias, correspondendo à distância vertical entre o ponto y_i e a função da média $E(Y|X=x_i)$.

São feitas duas suposições importantes sobre a natureza dos erros. Primeiro, assume-se que $E(e_i|x_i)=0$. A segunda suposição é que os erros devem ser independentes, significando que o valor do erro para um caso não gera informação sobre o valor do erro para outro caso. De uma forma geral, assume-se que os erros são normalmente distribuídos (distribuição estatística gaussiana), com média igual a zero e variância σ^2, desconhecida.

Muitos métodos estatísticos são sugeridos para obter estimativas dos parâmetros de um modelo, entre eles destacam-se os métodos dos Mínimos Quadrados e de Máxima Verossimilhança.

Supondo n pares de observações (x_1, y_1), (x_2, y_2), ..., (x_n, y_n), as estimativas $\hat{\beta}_0$ e $\hat{\beta}_1$ de β_0 e β_1, respectivamente, devem resultar em uma linha que melhor se ajusta aos pontos. Assim, o método dos mínimos quadrados busca minimizar a soma dos quadrados dos resíduos e_i, que serão definidos a seguir, onde os estimadores são dados pelas equações abaixo:

$$\hat{\beta}_1 = \frac{\sum_{i=1}^{n} y_i x_i - \frac{(\sum_{i=1}^{n} y_i)(\sum_{i=1}^{n} x_i)}{n}}{\sum_{i=1}^{n} x_i^2 - \frac{(\sum_{i=1}^{n} x_i)^2}{n}} \tag{31}$$

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x} \tag{32}$$

onde $\bar{x}$ e $\bar{y}$ são dados por:

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n} \tag{33}$$

$$\bar{y} = \frac{\sum_{i=1}^{n} y_i}{n} \tag{34}$$

Com os estimadores, a reta (ou modelo) de regressão é dada, por:

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x \tag{35}$$

onde cada par de observações satisfaz a relação:

$$y_i = \hat{\beta}_0 + \hat{\beta}_1 x_i + e_i, \qquad para \quad i = 1,2,..,n \tag{36}$$

A partir da equação acima, podemos então definir resíduo como:

$$r = \hat{e}_i = y_i - \hat{y}_i \tag{37}$$

onde $\hat{e}_i$ será o erro no ajuste do modelo para a i-ésima observação y_i.

Os resíduos $\hat{e}_i$ são utilizados para obter uma estimativa da variância σ^2 através da soma dos quadrados de $\hat{e}_i$:

$$\hat{\sigma}^2 = \frac{\sum_{i=1}^{n} \hat{e}_i^2}{n-2} \tag{38}$$

Para utilizar modelos de regressão linear de forma a estabelecer as relações entre parâmetros de configuração de barramento e desempenho de comunicação das aplicações embarcadas, é necessário determinar formalmente se as suposições de normalidade sobre a natureza dos erros podem ser atendidas. Assim, neste trabalho de tese, foi adotado o formalismo de Modelos Lineares Generalizados (MLG)[1][122], por serem mais flexíveis quanto às distribuições de probabilidade estabelecidas para os erros de estimativas, como modelos de comunicação.

4.3.3.1.1 Modelos Lineares Generalizados

Modelo Linear Generalizado é um método baseado na técnica de Máxima Verossimilhança [122] na qual os parâmetros do modelo de regressão podem ser estimados quando o modelo estatístico da distribuição dos erros pertence à família exponencial de distribuições [2]. Os parâmetros são, então, utilizados para calcular os valores esperados da variável resposta e, assim, os resíduos. A homogeneidade dos resíduos é avaliada e utilizada como critério para determinar se o modelo dos erros é apropriado [2].

Verossimilhança é definida como o produto das probabilidades da observação de cada valor da variável resposta. Para distribuições contínuas, como a Gaussiana e a gama, as funções de densidade são utilizadas no lugar da probabilidade. É usual considerar o logaritmo da verossimilhança para se ter valores mais tratáveis (e qualquer máximo da verossimilhança é também máximo da log-verossimilhança). Estimação de máxima verossimilhança, na prática, procura identificar os valores dos parâmetros que maximizem esta (log-) verossimilhança[3].

Um modelo linear generalizado possui três componentes. O primeiro é o componente aleatório $\boldsymbol{Y}$, que é um vetor de observações de y, tendo n elementos que são independentemente distribuídos e pertencentes a uma das distribuições da família exponencial.

Formalmente, a família exponencial é uma família com dois parâmetros [3], definida como:

$$f_i(y_i, \theta_i, \varphi) = \exp\left(\frac{y_i\theta_i - b(\theta_i)}{a_i(\varphi)} + c(y_i, \varphi) \right) \tag{39}$$

onde $a_i(\varphi)$, $b(\theta_i)$ e $c(y_i,\varphi)$ são funções conhecidas; θ_i é um parâmetro relacionado à média; e φ é um parâmetro relacionado à variância. Para fins práticos, é útil saber que um membro da família exponencial possui as seguintes propriedades:

- a distribuição é completamente especificada em termos de sua média e variância,
- a variância de Y_i é uma função de sua média.

A segunda propriedade é obtida expressando-se a variância como

$$V(y_i) = \frac{\varphi . V(\mu_i)}{\omega_i} \tag{40}$$

onde $V(.)$, chamada de função de variância, é uma função especificada; o parâmetro φ refere-se à variância; e ω_i é uma constante que atribui peso, ou credibilidade, à observação i.

Várias distribuições conhecidas pertencem à família exponencial, como exemplo: Gaussiana (Normal), Poisson, binomial, gama, e Gaussiana inversa. Os valores correspondentes das funções de variância são ilustrados na Tabela 4.

Tabela 4 - Distribuições e respectivas funções de variância.

Distribuição	V(x)
Normal	1
Poisson	x
Gama	x^2
Binomial	x(1-x)
Gaussiana Inversa	x^3

O segundo é o componente preditor, que é a especificação para o vetor $\boldsymbol{Y}$ em termos de parâmetros desconhecidos $\beta_1, \beta_2,..., \beta_p$. Um preditor linear $\boldsymbol{\eta}$ é dado por:

$$\eta = \sum_{j=1}^{p} X_j \beta_j + \xi_j \tag{41}$$

onde X é o vetor de variáveis preditoras para as observações Y e ξ é o vetor de erros, associados às observações.

Algumas vezes é necessário atribuir categorias ou classificações a variabilidade evidente nos dados. Por exemplo, neste estudo, as simulações podem ser classificadas pela presença ou ausência de operações com pipeline. Em outro exemplo, os processadores podem executar suas aplicações em três níveis de prioridade diferentes. Estas duas variáveis preditoras são chamadas de fatores [131]. As variáveis preditoras podem ser quantitativas ou qualitativas. As qualitativas são divididas em classes. As classes individuais de uma classificação são chamadas de níveis ou classes de um fator. Na classificação de dados em termos de fatores e seus níveis, a característica importante observada é a extensão de quais níveis diferentes de um fator podem influenciar a variável de interesse [131]. Fatores são frequentemente representados por variáveis *dummy*[132].

Seja D um fator com cinco níveis, a j-ésima variável *dummy* U_j, para o fator D, com $j=1,...,5$, possui o i-ésimo valor u_{ij}, para $i=1,...,n$, dado por

$$u_{ij} = \begin{cases} 1, & se \quad D_i = j-ésima \quad categoria \quad de \quad D \\ 0, & do \quad contrário. \end{cases} \tag{42}$$

Para exemplificar, consideremos o parâmetro de configuração de um barramento com *suporte a pipeline* como um fator D de dois níveis (suporte e ausência de pipeline). Tomando uma amostra, disposta na Tabela 5, com cinco configurações diferentes, podemos representar o fator D com as variáveis *dummy* da Tabela 6.

Tabela 5 - Amostra de tamanho 5 com diferentes configurações de suporte a pipeline.

	Suporte a pipeline
1	Sim
2	Não
3	Sim
4	Não
5	Não

Tabela 6 - Representação da amostra da Tabela 5 através de variáveis dummy.

	u1	u2
1	1	0
2	0	1
3	1	0
4	0	1
5	0	1

Podemos observar na Tabela 6 que as configurações com suporte à pipeline tiveram valores *u1=1* e *u2=0*, e as configurações sem suporte tiveram valores *u1=0* e *u2=1*.

Neste trabalho de tese, verificou-se que os valores assumidos pelos parâmetros abordados de configuração de barramento são valores discretos, ou classificáveis, caracterizando assim a necessidade de considerá-los como fatores para a abordagem de estimação por MLG.

MLGs podem também considerar a combinação entre dois ou mais fatores. Quando o MLG possui mais de um fator, o efeito da combinação de dois ou mais fatores é chamado efeito de interação. Interações ocorrem quando o efeito de um fator varia de acordo com o nível de outro fator. Em contraste o efeito de um fator simples, ou seja, sem interação, é chamado de efeito principal. O conceito de interação é dado como segue: se a mudança na média da variável resposta entre dois níveis de um fator A é a mesma para níveis diferentes de um fator B, pode-se dizer que não há interação; mas se a mudança é diferente para diferentes níveis de B, pode-se dizer que há interação [7]. Interações relatam o efeito que fatores possuem sobre o risco do modelo, e que não são relatados na análise de correlação entre os fatores.

Por fim, o terceiro componente é a ligação entre o componente aleatório e os componentes sistemáticos. Essa ligação é frequentemente escrita como $\eta=g(Y)$ ou $Y=g^{-1}(\eta)$, onde g é a função de ligação [2]. A tabela a seguir ilustra as funções de ligação mais comuns:

Tabela 7 - Funções de ligação

Nome	g(x)	$g^{-1}(x)$
Identidade	x	x
Log	ln(x)	e^x
Logit	ln(x/(1 - x))	$e^x/(1 + e^x)$
Recíproca	1/x	1/x

Modelos de regressão linear clássicos têm uma distribuição normal no primeiro componente e a função de ligação identidade para o terceiro componente.

Cada estrutura de erro é associada a uma função de ligação chamada "canônica", que simplifica a matemática para solução analítica de MLGs. A Tabela 8 ilustra a função de ligação canônica para algumas distribuições de erro.

Tabela 8 - Distribuições e respectivas funções de ligação canônica.

Distribuição	Função de Ligação Canônica
Normal	$\eta = Y$
Lognormal	$\eta = Y$
Gama	$\eta = 1/Y$
Poisson	$\eta = \log(Y)$

De acordo com Nelder e Wedderburn em [20], o fluxo tradicional de projeto para modelagem por MLGs pode ser dividido em três etapas: (i) formulação dos modelos; (ii) ajuste e (iii) inferência.

Os MLGs formam um ferramental de grande utilidade prática, pois apresentam grande flexibilidade na etapa (i), computação simples em (ii) e critérios razoáveis em (iii). Essas etapas são realizadas sequencialmente. Na análise de dados complexos, após a realização da etapa de inferência, pode-se voltar à etapa (i) e escolher outros modelos, a partir de informações mais detalhadas obtidas do estudo feito em (iii).

A primeira etapa, formulação de modelos, compreende a escolha de opções para a distribuição de probabilidades da variável resposta (componente aleatório), variáveis preditoras e a função de ligação entre estes dois componentes. A variável resposta utilizada neste trabalho de tese consiste na estimativa de desempenho da estrutura de comunicação da plataforma. As variáveis preditoras são os parâmetros de configurações dos barramentos contidos no espaço de projeto de comunicação. Para o estudo foram analisadas diversas funções de ligação, sendo empiricamente escolhida a função identidade, pois representa o mapeamento direto entre configurações de barramento e respectivos desempenhos estimados.

A etapa de ajuste consiste no processo de estimação dos parâmetros lineares dos modelos lineares generalizados. Vários métodos podem ser usados para estimar os parâmetros dos MLG. Como o método de máxima verossimilhança é o procedimento de estimação mais utilizado atualmente [118], e que é bastante simples, esse método foi adotado neste trabalho de tese.

Por fim, a etapa de inferência tem como objetivo principal verificar a adequação do modelo e realizar um estudo detalhado quanto às discrepâncias de desempenho obtido por simulação completa do sistema e desempenho estimado com o modelo linear generalizado. Essas

discrepâncias, quando significativas, podem implicar na escolha de outro modelo linear, ou em aceitar a existência de dados aberrantes. Em qualquer caso, toda a metodologia de trabalho deverá ser repetida. O analista deve, nessa etapa, verificar a precisão e a interdependência das estimativas de desempenho, construir regiões de confiança e testes sobre os parâmetros de interesse, analisar estatisticamente os resíduos e realizar previsões. Para tanto pode dispor de métodos mais formais, como testes de hipótese, ou de análise gráfica residual, que é um método de avaliação mais subjetivo e, consequentemente, mais flexível.

A subseção a seguir descreve os procedimentos estatísticos tradicionais para formulação de um modelo linear generalizado com mais detalhes.

4.3.3.2 Abordagem Estatística para Formulação de um Modelo Linear Generalizado

A construção dos modelos lineares generalizados, em abordagens tradicionais, se inicia com a definição da variável aleatória e identificação das variáveis sistemáticas.

O componente aleatório (Vetor $\boldsymbol{Y}$), neste trabalho de tese foi definido como o desempenho de comunicação da aplicação e as variáveis preditoras (vetor $\boldsymbol{X}$) são as configurações de barramento contidas no espaço de projeto.

Em seguida, o primeiro estudo a ser realizado sobre as variáveis preditoras foi a relação de multicolinearidade. O termo multicolinearidade (ou colinearidade) refere-se a uma relação aproximadamente linear entre variáveis preditoras, em outras palavras. De acordo com Dobson [135], esta condição tem várias conseqüências indesejáveis e Hocking cita, em [136], algumas delas: as estimativas dos parâmetros β podem ser maiores do que previsto, uma estimativa pode ser negativa, contradizendo a regra prevista para o preditor, e variáveis preditoras podem tornar-se insignificantes para o modelo. Existem muitas técnicas para detecção de multicolinearidade, sendo as mais comuns o cálculo da tolerância e o fator de aumento de variância, também conhecido como *Variance Inflation Factor* (VIF)[140]. De acordo com O'Brien, em [140], quando o resultado do cálculo da tolerância, ou do VIF, é inferior a 0,25, ou superior à 4, respectivamente, é necessário remover uma ou mais variáveis preditoras do modelo para reduzir a colinearidade.

Uma vez verificado a inexistência de multicolinearidade, o passo seguinte consiste da

seleção de variáveis, dentre as opções, que melhor ajustam o modelo. O objetivo é dividir o conjunto de variáveis preditoras $\boldsymbol{X}$ em um conjunto de termos ativos $\boldsymbol{X_A}$ e um conjunto de termos inativos $\boldsymbol{X_I}$. Para tanto, existem dois aspectos que devem ser considerados. Primeiro, dado um candidato particular $\boldsymbol{X_C}$ para ser termo ativo, é necessário utilizar um critério para comparar $\boldsymbol{X_C}$ com outras possíveis escolhas de $\boldsymbol{X_A}$. O segundo aspecto, que é computacional, refere-se ao grande número de comparações que precisam ser realizadas.

Segundo Weisberg [40], critérios para comparar vários subconjuntos de candidatos podem ser o grau de ajuste do modelo e sua complexidade. Grau de ajuste do modelo é medido comumente pela soma dos quadrados dos resíduos (SQR) e a complexidade é medida pelo número de termos *tc* em $\boldsymbol{X_C}$, incluindo o intercepto no modelo linear. Entre os critérios mais utilizados estão *Akaike Information Criterion* (AIC) [137], *Bayes Information Criterion* (BIC) [138] e *Malow's* C_p [139].

Segundo o estudo apresentando em [145], o critério mais popular atualmente é o AIC, o qual pode ser calculado pela equação a seguir:

$$AIC = -2.M + 2.tc \tag{43}$$

onde *M* é a Máxima Log-Verossimilhança e *tc* o número de termos do modelo.

Para comparação de modelos, é comum utilizar a técnica computacional de Regressão *Stepwise* [123], que é uma técnica de escolha do conjunto de variáveis preditoras, que irão compor um modelo linear, através da inserção, ou remoção, gradual (a cada passo) de uma variável ao modelo. Esse processo é controlado pelo teste estatístico F (*Fisher*) [124], que compara modelos lineares verificando os que mais se ajustam a um conjunto de dados.

Apesar de muito utilizada, a técnica de regressão *stepwise* pode apresentar problemas quando há correlação linear entre variáveis preditoras [44], além de sobrevalorizar as estimativas dos parâmetros do modelo linear [125]. Outro aspecto importante a ser considerado é que a utilização de conjuntos de treinamento maiores não favorece a busca de um modelo linear mais ajustado [126].

Em estudos experimentais, que serão apresentadas no Capítulo 5, para formulação de um modelo linear generalizado para modelar a comunicação de aplicações embarcadas utilizou-se a técnica computacional de regressão *stepwise* para automatizar a formulação dos modelos e o critério *AIC* para compará-los. Os resultados mostraram que somente foi possível obter um MLG

bem ajustado aos dados após utilizar conjuntos de treinamentos com tamanhos iguais ou superiores a 75% do espaço de projeto.

A próxima subseção apresenta a abordagem proposta para geração automatizada de modelos lineares generalizados capazes de estimar o desempenho de comunicação de uma aplicação.

4.3.3.3 Abordagem Proposta para Geração de Modelo de Comunicação

Utilizando as estimativas obtidas a partir do uso da abordagem de grafos de comunicação como conjunto de treinamento, é proposto um método baseado em Programação Genética (PG) [18][21] para buscar um modelo linear generalizado que possa ser utilizado para estimar com precisão todo o espaço de projeto de arquiteturas de comunicação. O fluxo da abordagem de PG pode ser visto na Figura 29.

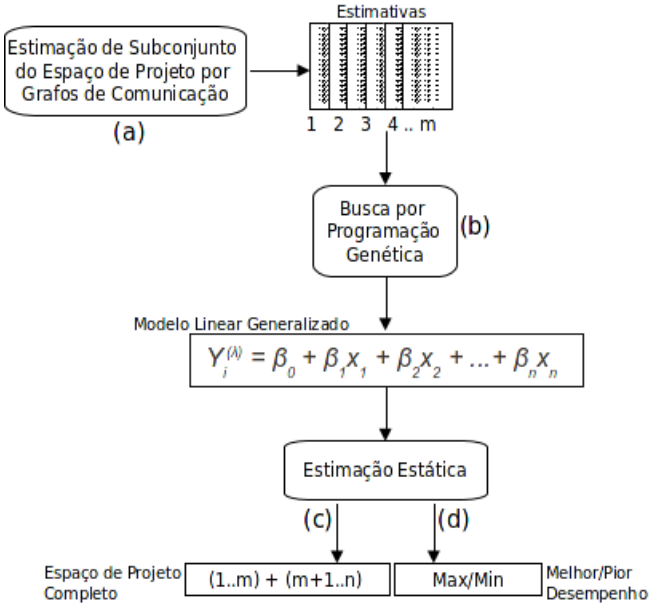


Figura 29 - Processos de criação e ajuste de melhor MLG por PG.

O processo de estimação por grafos de comunicação, mostrado na Figura 29(a), é realizado somente para um subconjunto do espaço de projeto, obtido a partir da aplicação de técnicas de projetos de experimentos aos pontos do espaço de projeto de configurações de

barramentos. As configurações do subconjunto de configurações do barramento bem como suas respectivas estimativas de desempenho serão utilizadas como conjunto de treinamento para estimação dos parâmetros do MLG. Diferente das abordagens tradicionais para estimar os parâmetros de um MLG [1], que têm como requisitos o conhecimento do conjunto de dados e de análises estatísticas, a abordagem proposta baseada em PG permite automatizar o processo de busca do MLG que se ajusta bem à aplicação, como pode ser visto na Figura 29 (b).

Para tanto, é gerado aleatoriamente um conjunto inicial de MLGs candidatos à solução. Esse conjunto passa por um processo iterativo de evolução, através da aplicação dos operadores genéticos de seleção, mutação e cruzamento. O conjunto de treinamento é utilizado para avaliar o ajuste às estimativas de desempenho e os parâmetros de configuração do barramento obtidos com a abordagem de grafos de comunicação.

Uma vez que um MLG foi selecionado depois de um número máximo predefinido de gerações, é possível utilizá-lo para estimar o desempenho do restante do espaço de projeto, como mostra a Figura 29(c), ou, ainda, para identificar as configurações de barramento que resultam os melhores e piores desempenhos da aplicação, Figura 29(d).

De acordo com Michalewicz em [19], Programação Genética consiste na aplicação dos algoritmos genéticos a estruturas de dados, que representem programas de computador, de forma a resolver problemas específicos. Em [18], Koza et al. cita que programação genética pode criar automaticamente uma solução geral na forma de uma estrutura de dados genética cujos parâmetros sejam expressões matemáticas contendo variáveis livres e que podem ser aplicáveis em diversas áreas.

Gen e Chen afirmam, em [17], que o problema fundamental atacado pela programação genética consiste de uma sequência de dados de entrada e de saída e da busca de uma função ou um programa que realize o melhor mapeamento entre eles.

Em termos gerais, no mecanismo proposto de exploração de comunicação, o algoritmo de programação genética busca um mapeamento, através de um MLG, entre o desempenho da estrutura de comunicação de uma determinada plataforma e as configurações de barramento presentes no espaço de projeto de comunicação.

Programação genética funciona semelhantemente aos algoritmos genéticos tradicionais: identificar uma estrutura de dados para codificar os programas/expressões e então aplicar operadores genéticos (cruzamento e mutação) de forma a evoluir tal estrutura, identificando a melhor solução para o problema em questão. Em PG, normalmente utiliza-se árvores, como

estrutura de dados [26], pois como as soluções são comumente expressões matemáticas, é necessário manter sua estrutura sintática (árvores são muito utilizadas para representar estruturas sintáticas, definidas de acordo com alguma gramática formal [75]). Os operadores de seleção, cruzamento e mutação são semelhantes aos dos algoritmos genéticos tradicionais, porém adequados às árvores de expressões. A técnica de PG foi aplicada anteriormente em outros projetos de sistemas embarcados e obteve bons resultados [13][14][15][16].

A subseção a seguir apresenta a abordagem de programação genética para busca de modelos lineares generalizados. Serão detalhados o algoritmo de programação genética proposto, a forma como os modelos lineares generalizados são codificados como soluções genéticas, o processo de criação da geração inicial e os operadores de seleção, cruzamento e mutação.

Algoritmo Proposto para Programação Genética

Programação Genética é uma técnica de busca e otimização, onde o objetivo é encontrar o melhor programa de computador que soluciona um problema a partir de algumas gerações de uma população de programas. A estrutura do pseudo-algoritmo proposto pode ser vista na Figura 30.

```
1:  genInitialPopulation(population,term,func);
2:  assesPopulation(population, data);
3:  while generationCount is lesser then MAX_GEN_C:
4:          parents = selectParents(population);
5:          population = genNewPopulation(population, parents,term,func);
6:          assesPopulation(population,data);
7:          if there is not bestOne:
8:                  bestOne = getBestOne(population);
9:          else:
10:                 if bestOne is better then getBestOne(population):
11:                         increase(bestOneCount);
12:                 else:
13:                         reset(bestOneCount);
14:                         bestOne = getBestOne(population);
15:         if bestOneCount is greater then MAX_BO_C:
16:                 break;
17:         increase(generationCount);
```

Figura 30 - Fluxo principal do algoritmo de programação genética.

Na primeira linha do algoritmo, a função *getInitialPopulation* gera uma população inicial de modelos lineares generalizados candidatos. Esta função recebe como parâmetros o conjunto de variáveis preditoras (*term*), que são os parâmetros de configuração do barramento, e o conjunto de operações (*func*) que podem conectar as variáveis preditoras no modelo linear generalizado. A linha 2 mostra a função *assesPopulation*, a qual avalia a população para verificar as soluções que melhor se ajustam aos dados do problema (configurações do barramento e respectivas latências estimadas com o modelo de grafos de comunicação). Esta avaliação é realizada através do uso da técnica de Máxima Verossimilhança para estimar os parâmetros dos MLGs candidatos. A linha 3 inicia um bloco de ciclos de iteração, o qual é executado até que o número máximo de gerações seja alcançado, sendo este o primeiro critério de parada do algoritmo.

As linhas 4 e 5 mostram, respectivamente, a função *selectParents,* a qual seleciona o conjunto de pais através de campeonatos, que têm seus participantes aleatoriamente selecionados, e a função *genNewPopulation*, que cria novas gerações de MLGs candidatos através de operações de cruzamento e mutação. Basicamente, as linhas 4 e 5 irão realizar o processo de evolução da população dos MLGs candidatos. Na sexta linha, as novas gerações são avaliadas pelo mesmo processo da geração inicial. Uma vez que cada indivíduo foi avaliado, a linha 7 verifica se existe um candidato a melhor solução e, se houver, na linha 8, a variável *bestOne* guarda uma referência para o melhor candidato. A linha 10 verifica se o melhor candidato permanece a melhor solução comparada à geração atual e, se sim, na linha 11, é incrementado o contador com o número de gerações as quais o melhor candidato permaneceu como a melhor solução. Caso o melhor candidato, que é o MLG mais ajustado às configurações de barramento e aos respectivos desempenhos de comunicação estimados pelos grafos de comunicação, não seja a melhor solução, comparado às soluções da geração atual, na linha 13, o contador é reiniciado e na linha 14 é selecionada a melhor solução da geração atual para ser o melhor candidato.

A linha 15 verifica se o melhor candidato excedeu o contador de tempo de vida (número máximo de gerações que uma solução pode ser o melhor candidato). Com isso o melhor candidato será também, provavelmente, o melhor candidato para as próximas gerações, e, na Linha 16, o algoritmo interrompe as iterações. Finalmente, o contador de gerações é incrementado (linha 17), inicializando, assim, uma iteração para uma nova geração.

As subseções a seguir detalham as partes principais do algoritmo de PG proposto.

Modelando MLGs como indivíduos genéticos

Modelos lineares generalizados são modelos estatísticos compostos por três elementos: uma variável dependente, variáveis sistemáticas (ou independentes) e uma função de ligação, cujo objetivo é o ajustamento linear entre a variável dependente e as sistemáticas.

Estes modelos são estruturados, na abordagem de programação genética proposta, como árvores, chamadas de árvores de expressões, onde os nós internos são operadores de ligação (representados pelo operador aritmético de adição) ou de iteração (representados pelo operador aritmético de multiplicação) entre variáveis preditoras, que são os parâmetros de configuração do barramento, e estão localizadas nas folhas da árvore, como pode ser visto na Figura 31.

Vemos na parte superior da Figura 31, um modelo linear generalizado, e logo abaixo, o mesmo modelo em forma de árvore, que é a estrutura de um indivíduo genético. Neste indivíduo, temos na raiz da árvore, e da subárvore esquerda, o operador de ligação, e nas folhas temos as variáveis preditoras *rc*, *m1p* e *m2p*, que representam o número de ciclos para requisição de uso do barramento, a prioridade de uso do barramento do mestre 1 e prioridades de uso do barramento do mestre 2, respectivamente.

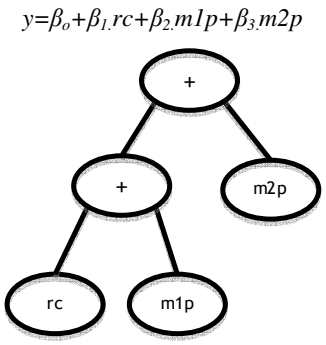


Figura 31 - Exemplo de modelo linear generalizado modelado como um indivíduo genético.

Formalmente, um MLG modelado como um indivíduo genético pode ser definido como uma árvore que contém um conjunto finito de um ou mais nós, onde:

i) existe um nó especial denominado raiz;
ii) os demais nós formam:
(1) dois conjuntos disjuntos, onde

(2) cada um desses conjuntos também é uma árvore, que neste caso, é também chamada de subárvore. As subárvores podem ser esquerda ou direita.

iii) a raiz da árvore, e das subárvores adjacentes, será um operador de ligação ou de iteração;

iv) a folhas serão as variáveis preditoras. Neste caso, cada folha representará um parâmetro de configuração do barramento.

A função de ligação irá conectar o componente preditor, que são as variáveis preditoras e operadores de ligação e/ou interação entre fatores, e que conjuntamente formam o indivíduo genético. Para identificar a melhor função de ligação, é utilizada, neste trabalho de tese, a técnica de transformação Box-Cox [27][3], que é dada por:

$$y_i^{(\lambda)} = \begin{cases} \dfrac{y_i^{\lambda}}{\lambda} & \text{Q } \lambda \neq 0 \\ log(\ y_i\ ) & \text{Q } \lambda = 0 \end{cases} \tag{44}$$

onde y_i é variável dependente e λ é o respectivo parâmetro de transformação. Ajustando o MLG para os dados através de valores diferentes de λ, é possível estimar o valor de λ que otimiza a correlação entre variáveis dependentes e sistemáticas.

A função de ligação encontrada deverá ser aplicada à variável resposta, que contém os desempenhos estimados com a abordagem de grafos de comunicação, para modificar a escala de seus valores, visando assim normalização. Com isso, mudando-se a escala dos valores das estimativas de desempenho, obtidas com a abordagem de grafos de comunicação, busca-se aproximá-los de uma forma normal, e consequentemente, minimiza-se a variabilidade dos dados, de forma que, ao utilizar o mecanismo de PG, pode-se identificar um MLG mais preciso.

Geração Inicial

Para que o algoritmo de evolução possa atuar, através da aplicação dos operadores de seleção, cruzamento e evolução, é necessário que haja uma geração inicial. Para tanto, visando

[3] A técnica de Box-Cox é bastante utilizada em estatística por permitir estabilizar variância entre estimativas, aproximar distribuições de dados para uma distribuição normal e aumentar correlação entre variáveis.

variabilidade de indivíduos e, consequentemente, uma melhora na precisão de resultados, adotou-se a técnica *Ramped Half-and-Half*[46].

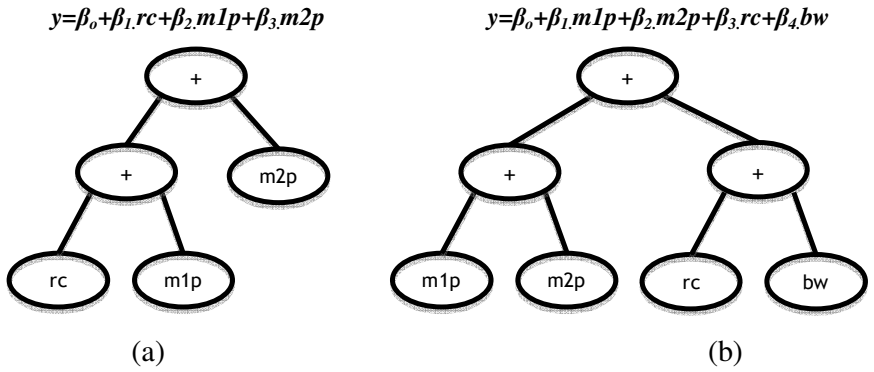


Figura 32 - Exemplos de árvores criadas a partir de (a) método de geração completo e (b) método de geração por crescimento.

Esta técnica seleciona, inicialmente, um valor aleatório para ser a profundidade máxima da árvore a ser gerada. Em seguida, é selecionado aleatoriamente o método de geração da nova árvore, que pode ser obtido por:

- **Crescimento**: este método cria árvores de vários tamanhos e formas, respeitando o limite de profundidade selecionado inicialmente. A Figura 32 (a) mostra um exemplo de uma árvore criada a partir da aplicação deste método. Nela, vemos que as folhas possuem diferentes profundidades.
- **Completo**: uma árvore criada com este método tem suas folhas de mesma profundidade, que é selecionada também aleatoriamente, porém respeita o limite de profundidade selecionado inicialmente. A Figura 32 (b) mostra uma árvore criada com este método. Observe que todas as folhas possuem mesma profundidade.

Seleção de pais

O método para seleção de pais deve simular o mecanismo de seleção natural que age sobre as espécies biológicas: pais mais aptos, aqueles que são mais ajustados aos dados do problema, geram um grande número de filhos, enquanto os menos aptos também podem gerar descendentes, evitando, assim, convergência genética prematura. Consequentemente, foca-se em

indivíduos com grande ajuste sem descartar completamente aqueles indivíduos com grau de ajuste extremamente baixo.

Sendo assim, para compor o conjunto de MLGs pais, é utilizado o método de seleção por campeonato [17]. Nesta abordagem alguns MLGs candidatos à solução são escolhidos aleatoriamente para competir entre si. Com esta técnica de seleção, os melhores MLGs da população somente terão vantagem sobre os piores, ou seja, só vencerão campeonatos, se forem selecionados.

A abordagem proposta para programação genética também utiliza a técnica de Seleção por Elitismo [19][29]. Nesta técnica, um pequeno subconjunto dos MLGs mais ajustados às configurações de barramento do espaço de projeto e estimativas de desempenho, obtidas pela abordagem com grafos de comunicação, de cada geração, serão automaticamente selecionados para compor a próxima geração. Com isso, garante-se que os resultados da abordagem de PG sempre terão aumento progressivo.

Ajuste de Indivíduos e Evolução

Para explicar o processo de evolução, é necessário entender o modelo de avaliação de um modelo linear generalizado candidato e mostrar como o processo de evolução é dado.

O ajuste de um MLG candidato é avaliado baseado na qualidade dos valores estimados que ele gera em comparação com os dados obtidos a partir das estimativas de desempenho obtidas com grafos de comunicação. A qualidade do modelo é quantificada a partir do cálculo da máxima verossimilhança, que é um método geral para estimação de parâmetros β_i de um MLG.

De forma a encontrar o MLG que melhor se ajusta aos dados obtidos com grafos de comunicação, os operadores de cruzamento e mutação são aplicados aos indivíduos genéticos, as árvores MLGs, como mostrado na Figura 33, partes (a) e (b), respectivamente. Os operadores de cruzamento e mutação, em programação genética, são similares aos presentes em algoritmos genéticos convencionais. No primeiro operador, os candidatos são selecionados para reprodução de acordo com seu ajuste (candidatos com maior ajuste aos dados do problema possuem probabilidades maiores de serem selecionados) e, em seguida, trocam conteúdos genéticos (subárvores), escolhidos aleatoriamente, entre si.

A Figura 33 (a) ilustra o cruzamento dos pais $y=\beta_0+ \beta_1.rc +\beta_2.m1p + \beta_3.m2p$ e $y=\beta_0+ \beta_1.m1p +\beta_2.m2p + \beta_3.m3p$, gerando os filhos $y=\beta_0+ \beta_1.m1p +\beta_2.m2p$ e $y=\beta_0+ \beta_1.rc +\beta_2.m1p + \beta_3.m2p+ \beta_4.m3p$.

Com mutação, uma subárvore é selecionada aleatoriamente em um MLG, também escolhido aleatoriamente, e mutada em uma nova subárvore diferente. A Figura 33 (b) ilustra a mutação do modelo $y=\beta_0+ \beta_1.m1p +\beta_2.m2p + \beta_3.m3p$ para $y=\beta_0+ \beta_1.pc + \beta_2.m3p$, onde se percebe que foi mutado o conteúdo genético $m1p + m2p$ para pc.

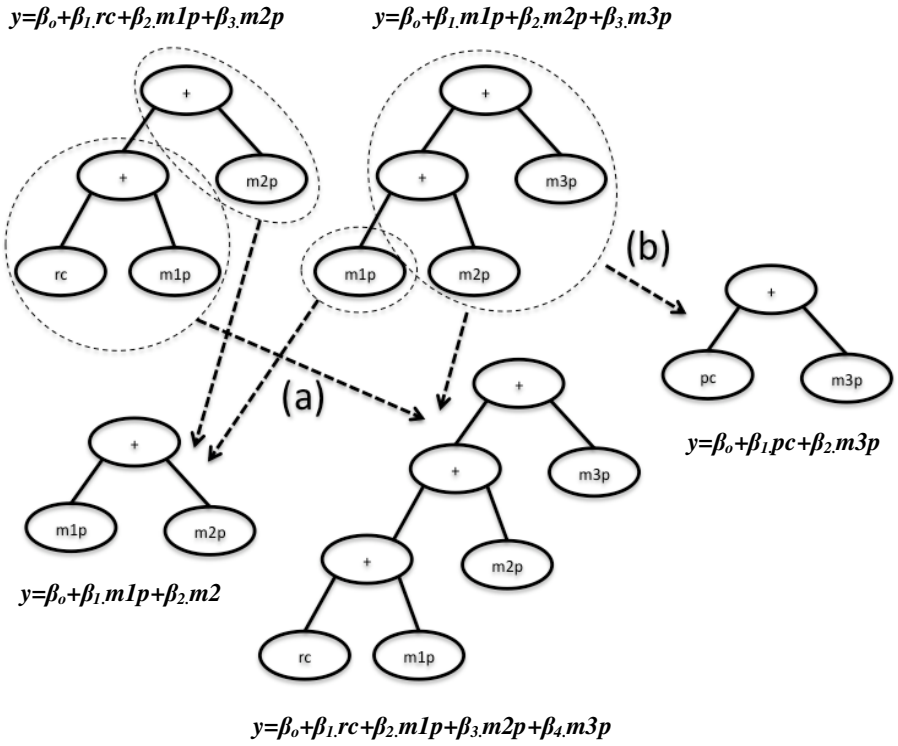


Figura 33 - Árvores de expressões representado MLGs sob operações de (a) cruzamento e (b) mutação.

4.4 Conclusão

Neste capítulo foi apresentada uma nova abordagem de análise de comunicação em plataformas multiprocessadoras baseadas em barramentos. A abordagem proposta permite estimar o desempenho da aplicação sob os efeitos de configurações de barramento para todo o espaço de projeto.

O fluxo de projeto proposto inclui inicialmente uma simulação do sistema com modelo de barramento ideal, o qual não insere latências de comunicação no desempenho da aplicação e permite transferências em paralelo com dados de diversos tamanhos. Com essa simulação, é extraído o perfil de comunicação da aplicação, que é utilizando, em seguida, por mecanismo de análise estática, que inclui grafos de comunicação, programação genética e modelos lineares generalizados, para estimar o desempenho da aplicação para as configurações de barramento de todo o espaço de projeto.

O próximo capítulo apresenta resultados experimentais da aplicação da abordagem proposta.

Capítulo 5
Resultados Experimentais

5.1 Introdução

Para fins de validação da abordagem de análise de comunicação proposta neste trabalho de tese, foram utilizados dois estudos de caso contendo aplicações do benchmark SPLASH [37]. Estas aplicações, que são concorrentes, requerem alto desempenho na comunicação entre processos.

Como o principal objetivos do trabalho proposto é capturar um modelo de comunicação que seja capaz de estimar o desempenho de comunicação da aplicação de forma precisa e eficiente, a validação da técnica proposta será baseada na avaliação da redução do tempo de exploração do espaço de projeto com o modelo linear generalizado obtido, comparando este tempo com o tempo necessário quando se utiliza as abordagens tradicionais

No entanto, além de preocupar-se com o aumento de desempenho pelo uso das técnicas propostas, a validação consistiu em verificar também a precisão dos resultados obtidos. Todo o processo de validação é realizado em três cenários distintos:

Cenário 1: Avaliação do Espaço de Projeto usando simulações e técnicas estatísticas para formulação do MLG: realização de simulações de todo o espaço de projeto de configurações de barramento, utilizando plataformas virtuais de hardware descritas em linguagem de alto nível de abstração. Em seguida, são aplicadas técnicas estatísticas para selecionar um subespaço de configurações de barramento e para formular modelos lineares generalizados para representar a comunicação das aplicações na plataforma. Os demais pontos do espaço de projeto serão comparados com as estimativas dos MLGs para validação. Objetiva-se que os modelos sejam capazes de estimar estaticamente os desempenhos da comunicação no barramento para todos os pontos dos espaços de projeto. Ao fim, serão comparados:

a. O tempo necessário para formulação de um MLG para estimação do espaço de projeto com simulação do sistema: espera-se que o tempo necessário para formular um MLG, e consequentemente estimar o desempenho de comunicação de todo o espaço de projeto, seja inferior ao necessário para realizar simulações para cada configuração do espaço de projeto.

b. A precisão das estimativas: simulações resultam em estimativas bastante precisas das latências das comunicações entre módulos do barramento[12]. Assim, serão calculados os erros entre as estimativas fornecidas pelo conjunto de simulações e pelo uso do modelo linear generalizado, este sendo formulado por técnicas estatísticas tradicionais.

Cenário 2: Uso de simulações e obtenção do Modelo (MLG) através de Programação Genética: Neste cenário são realizadas simulações, uma para cada configuração de barramento do espaço de projeto, utilizando plataformas de hardware descritas em linguagem de alto nível de abstração. Após as simulações é aplicada a técnica de projeto de experimentos para selecionar um subconjunto desses pontos do espaço de projeto. Para estes pontos é aplicado o algoritmo proposto, baseado em programação genética, para geração de um modelo linear generalizado. Este modelo será utilizado para estimar estaticamente o desempenho de comunicação para todos os pontos do espaço de projeto. Para os resultados das estimativas obtidas neste cenário serão comparados:

c. O tempo necessário para geração de um MLG: espera-se que ao utilizar o método por programação genética seja possível gerar um MLG capaz de estimar todo o espaço de projeto, com tempo inferior ao do MLG formulado por técnicas estatísticas tradicionais.

d. Serão calculados os erros entre as estimativas fornecidas pelo conjunto de simulações e pelo uso do modelo linear generalizado, este sendo ajustado por programação genética.

Cenário 3: Uso de uma simulação para geração de grafos de comunicação, estimativa baseada em grafos de comunicação e programação genética para a captura do MLG de comunicação: esta etapa visa contemplar os resultados do modelo de grafo de comunicação em conjunção com a técnica proposta baseada em programação genética. A abordagem utiliza uma única simulação, com um modelo de barramento ideal, para extração do perfil de comunicação. Utilizando a técnica de projeto de experimentos serão selecionados os pontos do espaço de projeto que, juntamente com o perfil de comunicação, serão utilizados para geração de grafos de comunicação para estimação das latências de comunicação. As estimativas de desempenho obtidas através dos grafos serão utilizados como entrada para o algoritmo de programação genética, gerando assim um modelo linear generalizado. Com esta técnica que está sendo proposta neste trabalho espera-se obter uma redução significativa do tempo de exploração, em relação aos outros cenários, além de estimativas precisas do desempenho da aplicação em comparação com resultados obtidos por simulação.

As próximas seções descrevem com maiores detalhes, para cada uma destes cenários, o método de avaliação, as técnicas e ferramentas utilizadas, bem como os respectivos resultados.

5.2 Descrição dos estudos de caso e resultados das simulações

As duas aplicações escolhidas, para o processo de validação dos modelos e técnicas propostos neste trabalho de tese, foram ordenação de um conjunto de número inteiros por radix [39] e multiplicação de duas matrizes de números inteiros.

A primeira aplicação, cujo código fonte utilizado faz parte do pacote de *benchmarks* SPLASH [37][4], 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, algumas *flags* de controle e um *mutex* (para gerenciar o acesso

[4] 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.

mutuamente exclusivo). Em seguida, os inteiros na estrutura de dados serão ordenados pelos dois processos concorrentes.

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 serem multiplicadas. Para gerenciar o acesso às estruturas de dados em memória compartilhada, foi utilizado o algoritmo de Lamport (também conhecido como Bakery Algorithm) [38], o qual utiliza espera ocupada (semelhante ao *mutex),* e que permite acesso mutuamente exclusivo a mais de dois processos.

Para execução das duas aplicações, foram projetados dois modelos de plataformas de hardware em alto nível, descritas em SystemC, compondo, assim, os modelos de simulação para cada aplicação. Para analisar a eficiência e precisão da abordagem proposta neste trabalho de tese, foram utilizadas diferentes plataformas multiprocessadora baseadas no processador MIPS (modelo 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 de alto nível do ARM Amba AHB[48] com precisão de ciclo.

Os parâmetros de configuração do barramento utilizados para definir o espaço de projeto, estão apresentados na Tabela 9.

Tabela 9 - Parâmetros de configuração do barramento Amba AHB nas plataformas dos modelos simulação

Parâmetro de Configuração	Ordenação por radix	Multiplicação de matrizes
Largura do barramento	8, 16, 32 bits	8, 16, 32 bits
Mecanismo de arbitragem	Prioridade fixa	Prioridade fixa, sendo que o primeiro mestre sempre terá maior prioridade que os demais.
Frequência de operação	100, 166, 200 (MHz)	100, 166,200 (MHz)
Tipo de transferência	Simples com preempção, Simples sem preempção.	Simples com preempção, Simples sem preempção.

Com os parâmetros apresentados na Tabela 9, foi possível construir espaços de projeto totalizando 72 configurações distintas 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 3 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 memória RAM de 4 GB. Os tempos de simulação do espaço de projeto para as duas aplicações são mostrados na tabela a seguir.

Tabela 10 - Desempenho da abordagem de estimação por simulação do sistema para os estudos de caso.

	Ordenação por radix	**Multiplicação de matrizes**
Tempo médio de simulação para cada configuração	31 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 projeto, apresentados na Tabela 10, incluem: tempo para configuração da plataforma, compilação e execução do simulador.

5.3 Resultados obtidos no Cenário 1: Usando métodos estatísticos para formulação de modelos de comunicação a partir de simulações

Para os estudos de caso, utilizou-se a técnica de validação cruzada de um ponto [34] para formulação e validação dos modelos lineares generalizados. Este técnica consiste em dividir o espaço de projeto em dois subconjuntos menores, sendo que o primeiro, que é chamado de conjunto de treinamento, é utilizado para estimar os parâmetros lineares do MLG, e o outro, chamado de conjunto de teste, é utilizado para validação através de testes de aderência e/ou análise residual.

Sobre seleção dos conjuntos de treinamento para formulação dos MLGs, considerou-se dois aspectos: 1) os pontos do espaço de projeto que foram utilizados para compor o conjunto e 2) e quantidade de pontos. Para seleção dos pontos, foi utilizada amostragem aleatória, de forma

que o processo de estimação dos parâmetros lineares dos MLG não fosse influenciado e, consequentemente, gerasse modelos enviesados, ou seja, modelos com estimativas erradas dos parâmetros. Os conjuntos de treinamento tiveram seus tamanhos escolhidos de forma incremental e empírica, observando o processo de validação para cada conjunto.

Para a fase de formulação dos modelos, utilizou-se a técnica computacional de regressão *stepwise* para a seleção dos componentes do modelo, bem como o critério AIC[5] para comparação de ajuste aos dados entre modelos. Os parâmetros do MLG foram estimados através da técnica de máxima verossimilhança utilizando a distribuição Gaussiana como distribuição teórica dos erros. Foram selecionados empiricamente vários conjuntos de treinamento, porém, dentre eles, somente a partir daqueles com tamanho igual ou superior a 75% do espaço de projeto foi possível ter estimativas precisas de desempenho.

Para avaliar a eficiência na obtenção dos MLGs, observou-se que a técnica de regressão *stepwise* não insere *overhead* no método, de forma que o desempenho total é limitado pelos tempos de simulação dos conjuntos de treinamento. Assim, para obter um MLG para a aplicação de ordenação por radix, considera-se o tempo de construção do conjunto de treinamento com 75% do espaço de projeto, totalizando-se assim 27 minutos e 54 segundos. Já para a aplicação de multiplicação de matrizes, o tempo para estimação do conjunto de treinamento por simulação foi de 4 horas e 33 minutos.

As Tabela 11 e 12 mostram os MLG selecionados por regressão *stepwise* para conjuntos de treinamento com tamanho de 75% das configurações do espaço de projeto para as aplicações de ordenação por radix e multiplicação de matrizes, respectivamente.

Tabela 11 - MLG selecionado por regressão *stepwise* para a aplicação de ordenação por radix.

Termo do MLG	Parâmetro β_i
p12	-6,011e-06
bw2	-0,405
bw4	-1,099
ty2	0,288
pe6	0,182
pe10	0,693
bw2:ty2	-0,288

[5] *Akaike Information Criterion*. (para descrição ver Subseção 4.3.3.2).

bw4:ty2	-0,288
p12:p22	6,011e-06
p12:ty2	6,412e-06
p12:bw4	4,008e-07
p22:bw4	-4,008e-07
p12:p22:ty2	-6,412e-06
Intercepto	16,740

Tabela 12 - MLG selecionado por regressão *stepwise* para a aplicação de multiplicação de matrizes.

Termo do MLG	Parâmetro β_i
p23	0,100
p24	0,100
p33	0,093
p34	0,093
p43	-0,028
p44	0,006
bw2	-0,693
bw4	-1,324
ty2	0,076
pe6	0,182
pe10	0,693
p24:p33	-0,008
p23:p34	-0,307
p24:p34	-0,148
p33:p43	-0,073
p34:p43	0,106
p33:p44	-0,099
p34:p44	-0,072
p43:bw2	-0,0001
p44:bw2	0,0004
p43:bw4	-0,003
p44:bw4	-0,015
p23:p43	-0,069
p24:p43	0,094
p23:p44	0,206
p24:p44	-0,133
p43:ty2	-0,004

p44:ty2	0,002
bw2:ty2	0,0004
bw4:ty2	0,042
p24:p33:p43	-0,094
p24:p34:p43	-0,188
p24:p33:p44	0,014
p24:p34:p44	0,172
p43:bw2:ty2	-0,0006
p44:bw2:ty2	-0,002
p43:bw4:ty2	0,114
p44:bw4:ty2	-0,080
Intercepto	18,131

Neste trabalho de tese, os MLGs formulados por regressão *stepwise* foram avaliados por:

- Teste de aderência, para verificar se os dados do conjunto de treinamento e os dados ajustados pelos MLGs provem da mesma população. Foi escolhido o teste de aderência não-paramétrico [124] de *Mann-Whitney-Wilcoxon* [127], o qual é utilizado para verificar se os dados de dois conjuntos independentes tendem a ser iguais (a hipótese nula) ou diferentes (hipótese alternativa). Neste trabalho de tese, definiu-se o nível de significância em 5%. Os resultados do teste de *Mann-Whitney-Wilcoxon* com *p-value* superiores a essa margem indicam a não rejeição da hipótese nula.
- Cálculo do erro relativo médio global, como medida de localização central do conjunto de resíduos, os quais são erros apresentados entre as estimativas dadas pelo MLG para o conjunto de treinamento, e erros relativos máximos e mínimos para verificar possível presença de pontos aberrantes (*outliers*[6]) e precisão das

[6] *Outlier* é um ponto que diverge dos demais pontos da amostra, possuindo maior valor residual [151]. Existem várias causas para ocorrência de *outliers*, entre elas: erros de amostragem, erro de execução (inclusão na amostra de pontos que não pertencem à população) e variabilidade natural da população [152]. **Erro! Fonte de referência não encontrada.**Um *outlier* quando removido da amostra e, dessa forma, influencia significantemente o ajuste de um modelo de regressão é chamado de ponto de influência [153]. A decisão de manter ou remover tais pontos da amostra deve se basear em análise dos dados e da população, bem como deve satisfazer uma suposição na análise

estimativas. Além desses, é calculado também o desvio padrão, como medida de dispersão dos erros em torno da média.

- Diagnóstico gráfico residual, onde foram construídos os seguintes diagramas:
  - Diagrama de distribuição de erros acumulados, para verificar a distribuição dos erros, dados pelo MLG sobre o espaço de projeto. Esse diagrama será utilizado para quantificar a dispersão das estimativas em relação aos conjuntos de treinamento e teste;
  - Q-Q Plots e Histogramas, para verificar suposições sobre a distribuição de probabilidade do erro;
  - Diagrama de dispersão dos resíduos contra os valores ajustados, para verificar linearidade e de variância constante (homocedasticidade);
  - Diagrama de dispersão dos resíduos contra o tempo ou ordem de coleta, para verificar se existem correlação entre os erros;
  - Diagrama *Boxplot*, para verificar a presença de pontos aberrantes (*outliers*).

Os resultados do teste de aderência, bem como a média global, valores máximos e mínimos dos erros relativos de ajuste ao conjunto de treinamento, para os modelos apresentados nas Tabela 11 e 12 são apresentados na Tabela 13.

Como pode ser visto Tabela 13, os altos índices do teste de *Mann-Whitney-Wilcoxon,* para ambas as aplicações, indicam uma forte tendência de que as estimativas se ajustam aos dados dos conjuntos de treinamento.

Tabela 13 - Teste de ajuste aos dados dos conjuntos de treinamento, erros médio global, máximo e mínimo, bem como desvios padrões para as aplicações de ordenação por radix e multiplicação de matrizes.

	Ordenação por radix	**Multiplicação de matrizes**
Teste de *Mann-Whitney-Wilcoxon* (*p-value*)	99,75%	89,12%
Erro médio global	1,36e-15%	7,41%
Erro máximo	2,l2e-14%	13,64%
Erro mínimo	0%	8,53e-3%
Desvio Padrão	5e-15%	2,74%

dos dados ou prover algum resultado desejado [154]. Neste trabalho de tese, devido a natureza de formulação automatizada dos MLGs por PG, os *outliers* serão considerados como parte das amostras.

Para a aplicação de ordenação por radix, os erros médio global, máximo e mínimo tiveram índices muito baixos e próximos, indicando boa precisão dos dados. Já para a aplicação de multiplicação de matrizes, houve grande disparidade entre os valores dessas métricas, indicando a possível presença de *outliers*. Os desvios padrões mostram que há pouca variabilidade dos erros em torno das médias para ambas as aplicações.

As Figura 34(a) e (b) apresentam gráficos de erro acumulado para as duas aplicações. Nelas, pode ser visto que, para ordenação por radix, todos os pontos do conjunto de treinamento ficaram com erros acumulados muito próximos de zero; e, para multiplicação de matrizes, quase 100% do conjunto de treinamento apresentou erros acumulados inferiores a 15%. Estes resultados mostram que os MLG estão bem ajustados aos conjuntos de treinamento.

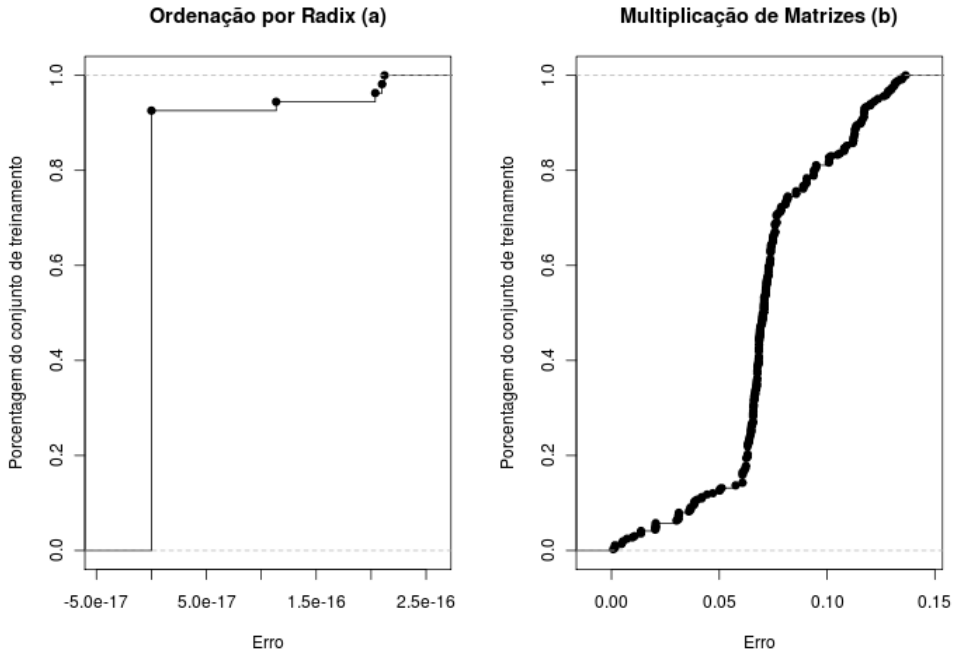


Figura 34 - Gráficos de erros acumulados para as aplicações de (a) ordenação por radix e (b) multiplicação de matrizes.

As Figura 35 e 36 apresentam gráficos para verificação da suposição de normalidade dos erros, uma vez que foi escolhida a distribuição Gaussiana para estimação dos parâmetros via máxima verossimilhança. As Figura 35(a) e (b) apresentam, respectivamente, um gráfico *q-qplot* e um histograma, e comprovam normalidade dos erros dos dados ajustados para a aplicação de ordenação por radix. É possível ainda verificar a presença de pontos distantes dos demais,

candidatos a *outliers*. Essas mesmas características podem ser observadas nas Figura 36(a) e (b), referentes aos dados ajustados para a aplicação de multiplicação de matrizes.

As Figura 35(c) e 36(c) mostram gráficos de dispersão para resíduos contra valores ajustados para comprovar a suposição de variância constante dos erros (homocedasticidade).

Os diagramas das Figura 35(d) e 36(d) mostram os resíduos por ordem de coleta. Nesses diagramas verifica-se que não existe um padrão entre a distribuição dos resíduos, comprovando a independência entre eles.

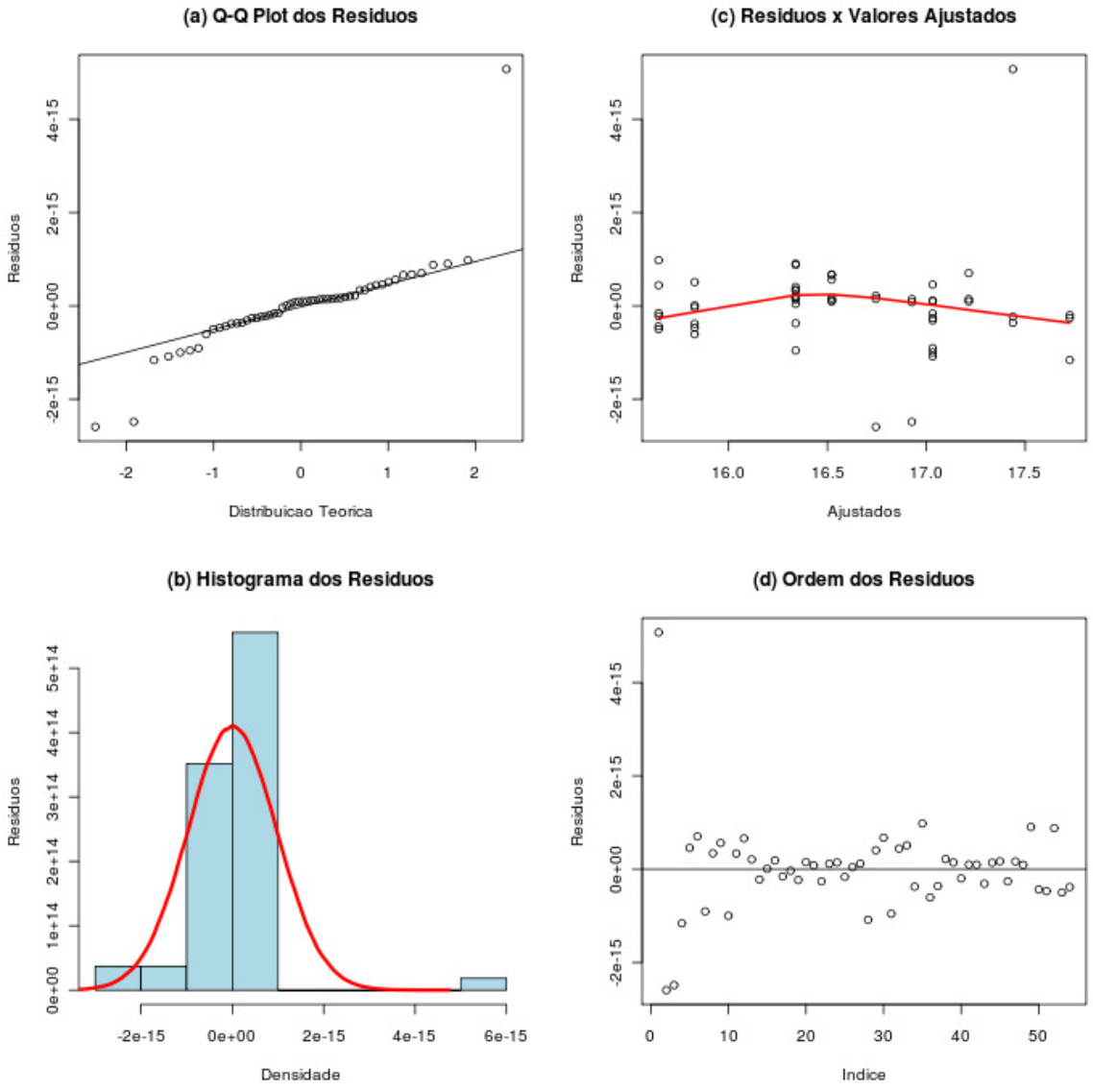


Figura 35 - Grafos para análise de suposições sobre distribuição dos erros para a aplicação de ordenação por radix.

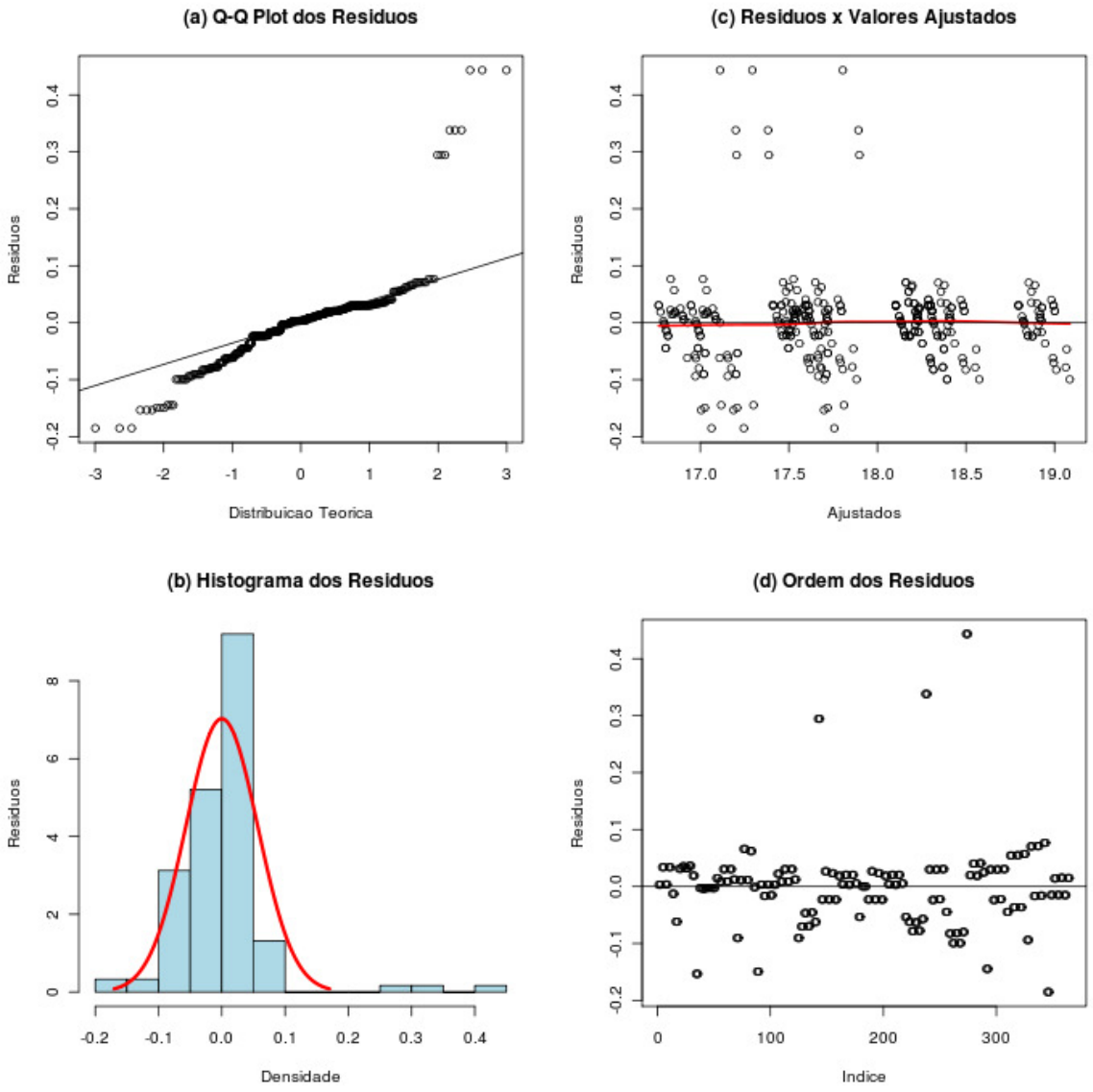


Figura 36 - Gráficos para análise de suposições sobre distribuição dos erros para a aplicação de multiplicação de matrizes.

A Tabela 14 apresenta os resultados do teste *Mann-Whitney-Wilcoxon* para os conjuntos de teste contra predições dadas pelos MLGs. Para a aplicação de ordenação por radix os dados se ajustaram muito bem, como é indicado pelo valor de 80,46%, porém para a aplicação de multiplicação de matrizes, o baixo valor obtido induz à rejeição da hipótese nula. É possível ver ainda, na Tabela 14, que os erros médios globais, erros máximos e mínimos para ambas as aplicações. Para a aplicação de ordenação por radix, esses valores ficaram bastante próximos,

indicando que as estimativas ficaram distribuídas entre a média. Esse fenômeno também pode ser comprovado pelo desvio padrão apresentado na Tabela 14. Já para a aplicação de multiplicação de matrizes, existe uma grande diferença entre essas métricas, tendo valor muito superior ao desvio padrão, indicando possível presença de *outliers*.

Tabela 14 - Teste de ajuste aos dados do conjunto de teste e erros médio global, máximo e mínimo, bem como desvios padrões para as aplicações de ordenação por radix e multiplicação de matrizes.

	Ordenação por radix	**Multiplicação de matrizes**
Teste de *Mann-Whitney-Wilcoxon* (*p-value*)	80,46%	1,62e-07%
Erro médio global	5%	2,9%
Erro máximo	8,8%	13,69%
Erro mínimo	2,3%	$3{,}23.10^{-3}$%
Desvio padrão	2,51%	2,91%

Para comprovar a existência de outliers, os diagramas boxplot, das Figura 37(a) e (b), apresentam a distribuição dos resíduos das aplicações de ordenação por radix e multiplicação de matrizes, respectivamente. Observa-se nesses dois diagramas que existem pontos distantes das cercas superiores e inferiores, indicando a presença de *outliers*.

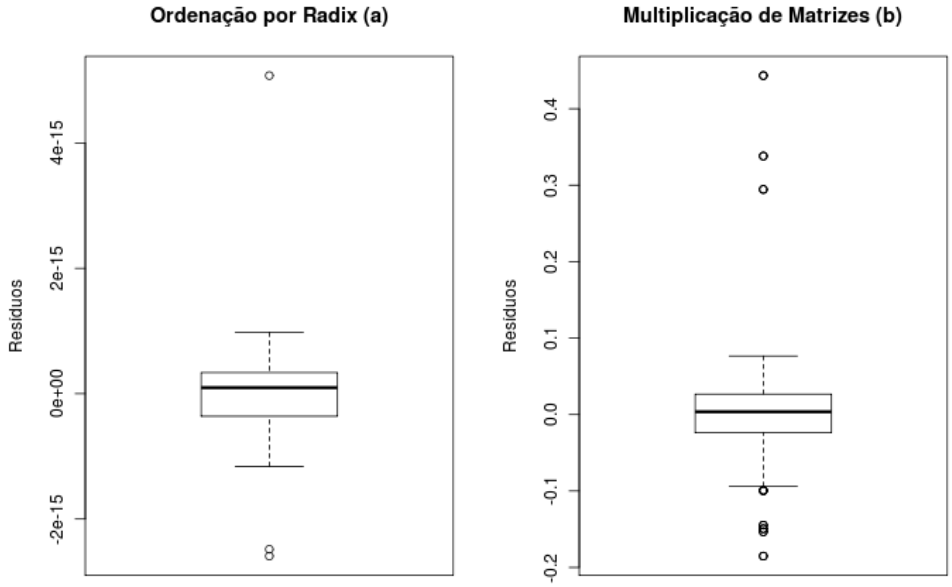


Figura 37 - Gráficos *Boxplot* para os resíduos das aplicações de (a) ordenação por radix e (b) multiplicação de matrizes.

Os passos seguintes para se ter um modelo bem ajustado consistem em analisar os *outliers* e, quando for o caso, remover os pontos de influência; realizar novamente a estimação dos parâmetros lineares do modelo; verificar, através de testes de aderência e análise residual, as suposições sobre as distribuições dos erros; e, por fim, realizar predições. Pode-se ainda alterar a distribuição teórica dos erros por alguma da família de distribuições simétricas (e.g. *t-student*). Este processo é iterativo até que o modelo não apresente novos pontos de influência.

Como foi visto nesta seção, com a utilização de regressão a *stepwise* é possível obter um MLG bem ajustado, porém fez-se necessário utilizar um conjunto de treinamento muito grande, sendo no mínimo de 75% de configurações do espaço de projeto. Assim, para minimizar o tamanho do conjunto de treinamento, é proposto, neste trabalho de tese, o uso de Programação Genética. A próxima seção apresenta alguns resultados experimentais com o uso dessa técnica.

5.4 Resultados do Cenário 2: Obtenção do MLG para estimação através de programação genética.

Para a seleção dos pontos que compõem o conjunto de treinamento para aplicação da técnica de programação genética, a técnica proposta para obtenção de um MLG bem ajustado, utilizou a técnica de projeto de experimentos implementada na ferramenta GPRSKit[41]. A técnica de projeto de experimentos é baseada na otimização da função objetivo de Audze-Eglais [35]. Com o uso dessa técnica é possível ter uma amostra mais representativa do conjunto de treinamento. Assim, para avaliar essa suposição, utilizaram-se conjuntos de treinamento menores para os dois estudos de caso, ordenação por radix e multiplicação de matrizes, compostos por 10% e 20% do tamanho do espaço de projeto, respectivamente. Os pontos selecionados podem ser vistos no Apêndice B.

Aplicando a técnica de Box-Cox [27] aos conjuntos de treinamento identificou-se a função logaritmo como sendo a melhor função de ligação para os MLG candidatos. Com isso, aplicamos a função logaritmo aos tempos estimados pelas simulações, transformando assim suas escalas.

O parâmetros do algoritmo de programação genética foram selecionados empiricamente, e consistiram de: 1000 candidatos para cada geração de árvores de MLGs; limitando em 50 o número máximo de gerações; e condição de parada do algoritmo consistindo de um MLG como

o candidato mais ajustado por 30 gerações consecutivas (o algoritmo, em média, finaliza em 40 gerações). [7]

Para cada geração foram realizados 900 campeonatos onde foram escolhidos aleatoriamente 50 MLGs participantes. Durante o campeonato foi calculado o índice AIC para avaliar cada um dos participantes. Assim, os vencedores, aqueles com melhores índices AIC, foram selecionados para realizar cruzamento. Para mutação, um fator de mutação foi calculado aleatoriamente em todas as árvores de MLGs. Se o valor do fator calculado para cada árvore foi inferior a 0,05 – índice demonstrado em [134] como apto a encontrar soluções boas em vários tipos de problemas – então esta árvore sofrerá mutação e, em seguida, será selecionada para compor a próxima geração. Finalmente, para completar o número de indivíduos da próxima geração, as árvores MLGs mais ajustadas da geração atual foram selecionadas (técnica de elitismo).

Uma vez que o processamento do algoritmo de PG finalizou, foi selecionado o indivíduo genético mais ajustado aos dados da geração final como o MLG final.

Os tempos para seleção dos pontos para construção dos conjuntos de treinamento, através da ferramenta GPRSKit, e seleção do MLG final são expostos na Tabela 15. Observe que para as duas métricas, os tempos ficaram muitos próximos para os conjuntos de treinamento com 10% e 20% do espaço de projeto, com exceção do tempo médio para construção do conjunto de treinamento da aplicação de multiplicação de matrizes.

Tabela 15 - Desempenho para seleção de pontos e do modelo linear generalizado final para as aplicações de ordenação por radix e multiplicação de matrizes.

	Ordenação por radix		**Multiplicação de matrizes**	
Tamanho do conjunto de treinamento	**10%**	**20%**	**10%**	**20%**
Tempo médio para construção do conjunto de treinamento	13s	15s	43s	2m21s
Tempo médio de execução do algoritmo de programação genética	1m38s	1m44s	2m32s	2m32s

[7]Em [121], Koza afirma que, em média, 50 gerações são suficientes para se encontrar uma solução aceitável; e quanto maior o número de indivíduos na população, maior a probabilidade de se encontrar uma solução válida.

A Tabela 16 apresenta o MLG selecionado por PG como o mais ajustado entre os dois conjuntos de treinamento da aplicação de ordenação por radix.

Tabela 16 - MLG selecionado por PG para a aplicação de ordenação por radix.

Termo do MLG	Parâmetro β_i
p1	-0,083
p2	0,589
ty	0,183
bw	-0,386
fr	0,290
fr:p2	-0,078
fr:ty	-0,070
p2:ty	-0,092
fr:p2:ty	0,033
Intercepto	16,165

Já para a aplicação de multiplicação de matrizes, a Tabela 17 apresenta o MLG final selecionado pelo algoritmo de PG.

Tabela 17 - MLG selecionado por PG para a aplicação de multiplicação de matrizes.

Termo do MLG	Parâmetro β_i
p2	-2,404
p3	-2,667
p4	-2,886
bw	-1,490
ty	-3,918
fr	-0,001
p3:p4	0,949
p3:p2	0,803
p2:p4	0,923
bw:p2	0,266
bw:p3	0,329
bw:fr	0,045
p3:ty	1,331

ty:p4	1,335
fr:p3	0,030
bw:p4	0,264
bw:ty	0,457
fr:ty	0,061
ty:p2	1,169
bw:p2:p4	-0,085
p3:p2:p4	-0,303
p3:ty:p2	-0,419
p3:ty:p4	-0,479
bw:p3:p4	-0,081
bw:fr:ty	-0,015
bw:p3:p2	-0,087
bw:fr:p3	-0,011
bw:ty:p4	-0,122
bw:ty:p2	-0,127
bw:p3:ty	-0,156
fr:p3:ty	-0,014
ty:p2:p4	-0,458
bw:p3:ty:p4	0,044
bw:p3:p2:p4	0,027
bw:fr:p3:ty	0,002
bw:ty:p2:p4	0,044
bw:p3:ty:p2	0,053
p3:ty:p2:p4	0,610
bw:p3:ty:p2:p4	-0,017
Intercepto	26,603

Para verificar a aderência dos MLGs aos dados dos conjuntos de treinamento foram aplicados o teste de *Mann-Whitney-Wilcoxon*, com 5% de significância, além do cálculo dos erros relativos globais médios, máximos, mínimos e desvios padrões. Os resultados podem ser vistos na Tabela 18.

Tabela 18 - Teste de ajuste aos dados do conjunto de treinamento e erros médio global, máximo e mínimo, bem como desvios padrões, para as aplicações de ordenação por radix e multiplicação de matrizes.

	Aplicação			
	Ordenação por Radix		**Multiplicação de Matrizes**	
	10%	**20%**	**10%**	**20%**
Teste de *Mann-Whitney-Wilcoxon* (*p-value*)	100%	98,17%	100%	100%
Erro médio global	7,81-06%	3,44%	0%	3,91%
Erro máximo	1,44e-05%	10,16%	0%	12,68%
Erro mínimo	0%	2,35e-06%	0%	6,06e-03
Desvio padrão	7,31e-6%	3,81%	0%	2,89%

Como se verifica na Tabela 18, de acordo com os resultados do teste de aderência, as estimativas dadas pelos MLGs selecionados por PG tendem a ser iguais aos dados dos conjuntos de treinamento. Para os conjuntos de treinamento com tamanho de 10% dos espaços de projetos, os erros médio global, máximo e mínimo, bem como desvios padrões, obtiveram índices muito baixos, e relativamente próximos, indicando assim precisão dos dados ajustados. Já para os conjuntos de 20%, os índices obtiveram valores distantes entre si, indicando a possibilidade de existência de *outliers*. Esse comportamento também é indicado pelos valores dados pelos desvios padrões, os quais são inferiores aos dos erros máximos.

Os diagramas de erros acumulados, de análise residual e *boxplots* dos modelos lineares generalizados encontrados por programação genética para os dois conjuntos de treinamento de cada aplicação podem ser encontrados no Apêndice C.

Da análise dos erros acumulados (ver Figura 50) percebe-se que quase todos os pontos dos conjuntos de treinamento para ambas as aplicações obtiveram erros totais inferiores a 10%. Nos diagramas de análise residual, apresentados nas Figuras 51 e 52, foi possível verificar que as suposições – normalidade, homocedasticidade e independência dos erros – sobre a distribuição teórica dos erros foram atendidas para os conjuntos de treinamento de ambas as aplicações. Por fim, os diagramas *boxplot*, apresentados na Figura 53, comprovam a presença de *outliers* para os conjuntos de treinamento com 20% do espaço de projeto para ambas as aplicações.

A Tabela 19 apresenta os resultados da aplicação do teste de *Mann-Whitney-Wilcoxon* para verificar o ajuste das predições, dadas pelos MLGs encontrados por programação genética, aos conjuntos de teste das aplicações de ordenação por radix e multiplicação de matrizes.

Percebe-se que os índices indicaram a não rejeição da hipótese nula, que é a igualdade entre distribuições, o que caracteriza bom ajuste aos conjuntos de teste. Ainda na Tabela 19, são mostrados os erros relativos médios globais, máximos e mínimos, bem como respectivos desvios padrões, entre as predições e conjuntos de teste. Da análise dos erros, pode-se verificar que os MLGS permitiram predições muito precisas, uma vez que erros médios e mínimos obtiveram índices baixos. Porém, percebe-se uma possível presença de *outliers*, visto pelos erros máximos com índices muito altos, superando os desvios padrões.

Tabela 19 - Teste de ajuste aos dados do conjunto de teste e erros médios globais, máximos e mínimos, bem como desvios padrões, para as aplicações de ordenação por radix e multiplicação de matrizes.

	Aplicação			
	Ordenação por Radix		**Multiplicação de Matrizes**	
	10%	**20%**	**10%**	**20%**
Teste de *Mann-Whitney-Wilcoxon*	53,05%	69,11%	81,84%	56,49%
Erro médio global	4,12%	4,15%	4,14%	3,94%
Erro máximo	11,11%	14,21%	14,44%	12,10%
Erro mínimo	4.905e-05%	9,171e-2%	3,708e-2%	5.518e-03
Desvio Padrão	2,71%	3,12%	2,92%	2,89%

De forma geral, o uso de programação genética para formulação do MLG permitiu obter a mesma precisão das estimativas quando comparada ao método de regressão *stepwise*. A vantagem do uso da técnica de programação genética consiste no uso de conjuntos de treinamento menores, necessitando, assim, um menor número de simulações e, por conseguinte, um aumento de desempenho na obtenção do modelo de comunicação final.

5.5 Resultados Cenário 3: Estimação com o uso de grafos de comunicação, e programação genética para formulação do MLG.

A seção anterior mostrou, através de resultados experimentais, ser possível reduzir o número de simulações necessárias para obter-se um modelo de comunicação, que permite estimativas de comunicação precisas para uma determinada aplicação. Para tanto, utilizou-se espaços de

treinamento com tamanhos de até 10% da quantidade de simulações dos pontos dos espaços de projeto, justamente com a técnica de programação genética proposta.

Para um aumento do desempenho da abordagem de análise de comunicação proposta neste trabalho de tese, foi proposta ainda a integração das abordagens de estimação por grafos de comunicação e da técnica de programação genética para formulação do modelo de comunicação. O uso da técnica de grafos de comunicação visa substituir as simulações, de forma a reduzir o tempo necessário para estimação do desempenho de comunicação para cada configuração de barramento do espaço de projeto.

Novamente, utilizou-se a ferramenta GPRSKit para selecionar os pontos do espaço de projeto que farão parte do conjunto de treinamento. A diferença nesta nova abordagem é que as configurações selecionadas tiveram desempenhos estimados através da abordagem com grafos de comunicação (ver conjuntos de treinamento com respectivos desempenhos estimados no Apêndice D).

Para a geração dos grafos de comunicação substituiu-se o modelo de alto nível do barramento AMBA das plataformas de simulação pelo barramento com modelo de comunicação ideal proposto neste trabalho de tese. A Tabela 20 apresenta os resultados das simulações para cada aplicação, integrando informações de desempenho e dos perfis de comunicação gerados.

Tabela 20 - Resultados das simulações e dados do perfil de comunicação para as aplicações de ordenação por radix e multiplicação de matrizes.

	Ordenação por radix		**Multiplicação de matrizes**			
Total de transferências de leitura	441.346		3.114.601			
Total de transferências de escrita	182.422		504.980			
Total de transferências	623.768		3.619.581			
Total de 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 de perfil de comunicação	9 segundos		25 segundos			

Observa-se da análise da Tabela 20, que a aplicação de multiplicação de matrizes inclui um maior número de transferências, implicando assim em maior tempo de simulação e maior espaço em disco para o perfil de comunicação.

Uma vez que os perfis de comunicação foram extraídos, seguiu-se o processo de geração de grafos de comunicação. Para cada perfil de comunicação foi gerado um grafo de comunicação inicial com nós representando as transações de cada mestre de barramento. Para aumentar a eficiência deste processo, os perfis de comunicação foram carregados no Sistema Gerenciamento de Banco de Dados Relacional (SGBDR) PostgreSQL[91]. Cada aplicação foi criado um banco de dados no SGBDR. Os bancos de dados das aplicações do estudos de caso consistiam de uma única tabela, em que cada linha representou uma entrada do perfil de comunicação. As linhas são divididas em campos, que representam os campos de uma entrada do perfil de comunicação. Com o PostgreSQL foi possível otimizar o uso da memória através de mecanismo de cache, *buffers* compartilhados, memória virtual e paginação, obtendo-se assim maior desempenho ao acessar os elementos do perfil e geração do grafo de comunicação.

Após o grafo de comunicação ser construído, foram selecionadas as configurações dos conjuntos de treinamento para serem incorporadas ao grafo. A Tabela 21 apresenta alguns resultados de desempenho para estimação dos espaços de treinamento com a abordagem de grafos de comunicação.

Tabela 21 - Desempenhos para estimação de espaços de treinamento por grafos de comunicação para as aplicações de ordenação por radix e multiplicação de matrizes.

Métrica	**Ordenação por radix**		**Multiplicação de matrizes**	
Tempo médio para gerar o grafo de comunicação inicial	0,86 s		14,66 s	
Tempo médio para estimação de cada configuração do espaço de projeto	0,56 s		6,67 s	
Tamanho do conjunto de treinamento	10%	20%	10%	20%
Tempo para estimação do conjunto de treinamento	4s	7,8 s	5 min e 21 s	10 min e 40s

Observa-se na Tabela 21 que os tempos para estimação de cada configuração do espaço de treinamento é muito menor que o tempo obtido com a simulação do sistema. Consequentemente, os desempenhos para estimação dos conjuntos de treinamento serão também melhores.

A Tabela 22 apresenta os resultados dos cálculos do teste de *Mann-Whitney-Wilcoxon*, erros relativos médios globais, máximos e mínimos, bem como desvios padrões, calculados em relação aos respectivos desempenhos estimados por simulações, para os conjuntos de treinamento das aplicações de ordenação por radix e multiplicação de matrizes.

Tabela 22 - Precisão das estimativas por grafos de comunicação para os conjuntos de treinamento das aplicações de ordenação por radix e multiplicação de matrizes.

	Aplicação			
	Ordenação por Radix		**Multiplicação de Matrizes**	
	10%	**20%**	**10%**	**20%**
Teste de *Mann-Whitney-Wilcoxon* (*p-value*)	60,89%	56,64%	69,5%	69,01%
Erro médio global	0%	3,43%	0%	3,53%
Erro máximo	0%	10%%	0%	11,342%
Erro mínimo	0%	0%	0%	0%
Desvio padrão	0%	3,81%	0%	2,76%

Analisando a Tabela 22 percebe-se a obtenção de bons resultados da aplicação dos testes de *Mann-Whitney-Wilcoxon* aos conjuntos de treinamento, pois mostraram que as estimativas dadas por grafos de comunicação se ajustaram bem às dadas por simulações. A análise dos erros mostra que as estimativas dos conjuntos de treinamento para ambas as aplicações foram bastante precisas em relação às das simulações. Percebe-se que existem indícios da existências de outliers, dados os valores dos erros máximos superiores aos desvios padrões.

Os diagramas de erros acumulados, de análise residual e *boxplots* dos modelos lineares generalizados encontrados por programação genética para os dois conjuntos de treinamento de cada aplicação podem ser encontrados no Apêndice E.

Da análise dos erros acumulados percebe-se que quase 100% dos pontos dos conjuntos de treinamento para a aplicação de ordenação por radix apresentam erros inferiores a 10% e, para a aplicação de multiplicação de matrizes, inferiores a 12%, nos conjuntos com 20% do espaço de projeto. Nos diagramas de análise residual foi possível verificar que as suposições – normalidade, homocedasticidade e independência dos erros – sobre a distribuição teórica dos erros foram atendidas para os conjuntos de treinamento de ambas as aplicações. Por fim, os diagramas *boxplot* comprovam a presença de *outliers* para os conjuntos de treinamento com 20% do espaço de projeto para a aplicação de multiplicação de matrizes.

Seguindo o fluxo de projeto, proposto no capítulo anterior, foi utilizada novamente a técnica de Box-Cox para identificar a melhor função de ligação dos MLGs buscados por PG. Assim, como nos experimentos apresentados na seção anterior, também foi identificada a função logaritmo como sendo a melhor função de ligação. Com esta função, os desempenhos estimados pela técnica de grafos de comunicação, dos conjuntos de treinamento, foram transformados através da aplicação da função logaritmo aos dados estimados por grafos de comunicação. Assim, os conjuntos de treinamento foram utilizados juntamente com a abordagem de programação genética para busca de MLGs. Tabela 23 apresenta o melhor MLG encontrado para os conjuntos de treinamento para a aplicação de ordenação por radix.

Tabela 23 - MLG selecionado por GC e PG para a aplicação de ordenação por radix.

Termo do MLG	Parâmetro β_i
p1	1.075
p2	0.120
ty	1.454
bw	-0.325
fr	0.303
bw:fr	-0.006
fr:p1	-0.131
fr:ty	-0.126
ty:p2	-0.181
p1:ty	-0.908
fr:p1:ty	0.106
Intercepto	14.813

Já para a aplicação de multiplicação de matrizes, a Tabela 24 apresenta o melhor MLG encontrado por programação genética.

Tabela 24 - MLG selecionado por GC e PG para a aplicação de multiplicação de matrizes.

Termo do MLG	Parâmetro β_i
p2	0,343
p3	1,055
p4	1,666
ty	1,216

fr	0,445
bw	-1,636
p2:ty	0,387
p4:p2	-0,426
p3:p2	0,039
p3:p4	-0,553
bw:ty	-0,179
p3:ty	-0,063
p4:ty	-0,769
fr:p4	-0,146
fr:p3	-0,162
fr:p2	-0,182
bw:p4	0,053
bw:p2	1,225
bw:p3	0,958
p3:p4:p2	0,098
p3:p4:ty	0,151
p4:p2:ty	0,042
p3:p2:ty	-0,326
bw:p3:p4	-0,162
fr:p3:p2	0,084
fr:p3:p4	0,060
bw:p3:p2	-0,679
fr:p4:p2	0,074
bw:p4:ty	0,283
bw:p3:ty	-0,274
bw:p2:ty	-0,408
bw:p4:p2	-0,241
bw:p4:p2:ty	0,034
fr:p3:p4:p2	-0,030
bw:p3:p4:p2	0,150
p3:p4:p2:ty	0,041
bw:p3:p4:ty	0,003
bw:p3:p2:ty	0,293
bw:p3:p4:p2:ty	-0,054
Intercepto	14,788

Os resultados dos cálculos do teste de *Mann-Whitney-Wilcoxon*, erros relativos médios globais, máximos e mínimos, bem como desvios padrões, para as predições de desempenho de comunicação dadas pelos MLGs encontrados, por programação genética, para as aplicações de ordenação por radix e multiplicação de matrizes pode ser vistos na Tabela 25. Para o cálculo dessas métricas, foram considerados como conjuntos de teste os desempenhos de comunicação estimados por simulação.

Dos resultados dos testes de aderência, apresentados na Tabela 25, percebe-se que os índices indicaram a não rejeição da hipótese nula, o que caracteriza bom ajuste aos conjuntos de teste.

Ainda na Tabela 25, são encontradas os erros médios globais, máximos e mínimos entre as predições e conjuntos de teste. Da análise dos erros, pode-se verificar que os MLGs permitiram predições precisas, uma vez que se obtiveram erros mínimos muito baixos e erros médios globais variando entre 3,41% e 6,86, com variância variando entre e,57% e 5,71%. Porém, percebe-se a possível presença de *outliers*, visto pelos erros máximos com índices muito altos, com destaque para o conjunto com 10% para a aplicação de multiplicação de matrizes, o qual supera em quase sete vezes o valor do desvio padrão.

Tabela 25 - Precisão das predições dadas pelos MLGs das aplicações de ordenação por radix e multiplicação de matrizes.

	Aplicação			
	Ordenação por Radix		**Multiplicação de Matrizes**	
	10%	**20%**	**10%**	**20%**
Teste de *Mann-Whitney-Wilcoxon* (*p-value*)	56,58%	71,42%	78,04%	80,53%
Erro médio global	4%	3,77%	6,86%	3,41%
Erro máximo	11,11%	12,37%	35,41%	11,71%
Erro mínimo	0%	0%	1,68e-2	4,26e-3
Desvio padrão	2,76%	2,81%	5,71%	2,57%

Com os resultados dos estudos dos erros, pode-se considerar que as estimativas de desempenho obtidas pela técnica de grafos de comunicação, podem ser utilizadas com a técnica de programação genética proposta para a formulação do modelo de comunicação.

5.6 Conclusões

As abordagens utilizadas nos três cenários, propostos neste capítulo, foram avaliadas comparativamente. Para tanto, foram utilizados os erros relativos médios globais e tempo total, para avaliar a precisão das estimativas e o tempo total necessário para se formular o MLG final, respectivamente.

A Figura 38 mostra um gráfico comparativo entre os erros relativos médios globais das estimativas fornecidas pelas abordagens estatística com simulações (Est./Sim), considerando conjunto de treinamento com 75% do espaço de projeto, programação genética com simulações (PG/Sim) e programação genética com grafos de comunicação (PG/GC), com conjuntos de treinamento de 10% e 20% do espaço de projeto. Observa-se que mesmo com conjuntos de treinamento menores, os erros médios globais foram menores para as abordagens de programação genética e grafos de comunicação, sendo que a precisão decaiu para o conjunto de treinamento com apenas 10% do espaço de projeto, possuindo menor representatividade da população.

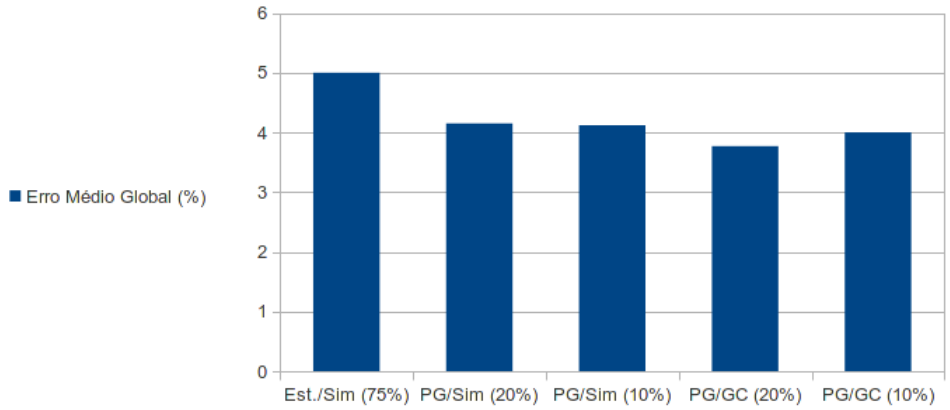


Figura 38 - Gráfico comparativo entre precisão das estimativas de desempenho de comunicação entre as abordagens de busca de MLG propostas para a aplicação de ordenação por radix.

Já para a aplicação de multiplicação de matrizes, os erros médios globais aumentaram sensivelmente, uma vez que foram utilizados conjuntos de treinamento menores, para as abordagens de que utilizam programação genética, como pode ser visto na Figura 39. A abordagem estatística, que utilizou regressão *stepwise*, por utilizar um conjunto de treinamento

maior obteve o melhor resultado. A abordagem de programação genética com grafos de comunicação destaca-se por obter, comparativamente, o pior resultado. Este efeito dá-se pela propagação de erros das estimativas de desempenho dadas pela abordagem de grafos de comunicação para os MLGs. Contudo, devido à magnitude do erro médio global ser inferior à 7%, esse resultado não interfere na aplicabilidade da abordagem.

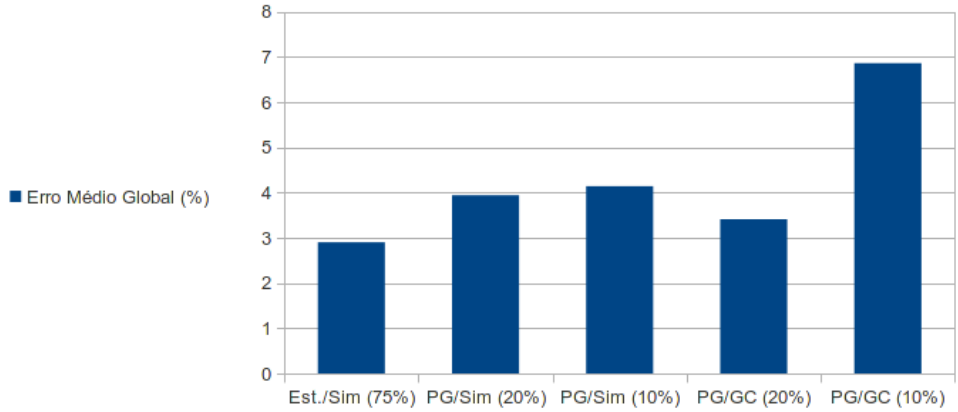


Figura 39 - Gráfico comparativo entre precisão das estimativas de desempenho de comunicação entre as abordagens de busca de MLG propostas para a aplicação de multiplicação de matrizes.

Analisando o desempenho das abordagens apresentadas neste capítulo, as Figura 40 e 41 mostram gráficos comparativos do tempo, em segundos, para obter um MLG para as aplicações de ordenação por radix e multiplicação de matrizes, respectivamente.

Observa-se, na Figura 40 que o desempenho aumenta com as abordagens de programação genética. Este aumento se dá pela redução do tamanho dos conjuntos de treinamento, necessários para obter-se os MLGs. A abordagem de programação genética com grafos de comunicação destaca-se como a de melhor desempenho, uma vez que as estimativas de desempenho de comunicação dos pontos dos conjuntos de treinamento são dadas pela técnica de grafos de comunicação. Esses efeitos são mais acentuados para a aplicação de multiplicação de matrizes, cujos resultados de desempenho podem ser vistos na Figura 41.

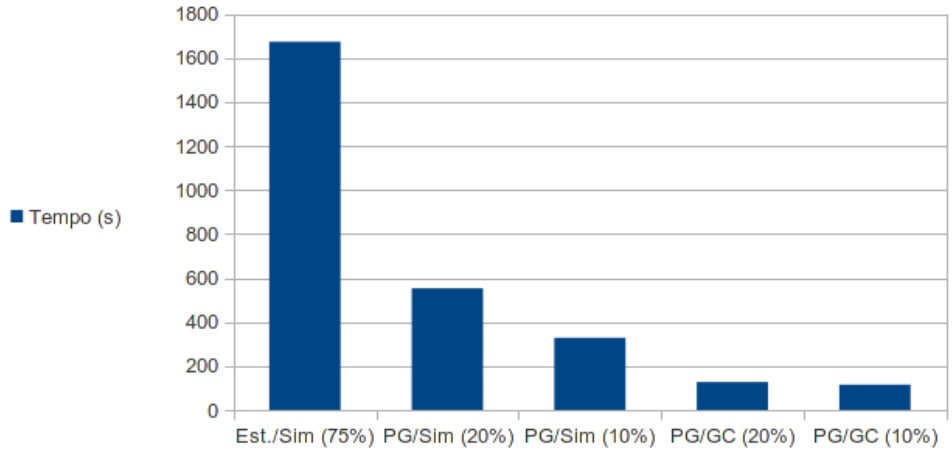


Figura 40 - Gráfico comparativo de desempenho entre as abordagens de busca de MLG propostas para a aplicação de ordenação por radix.

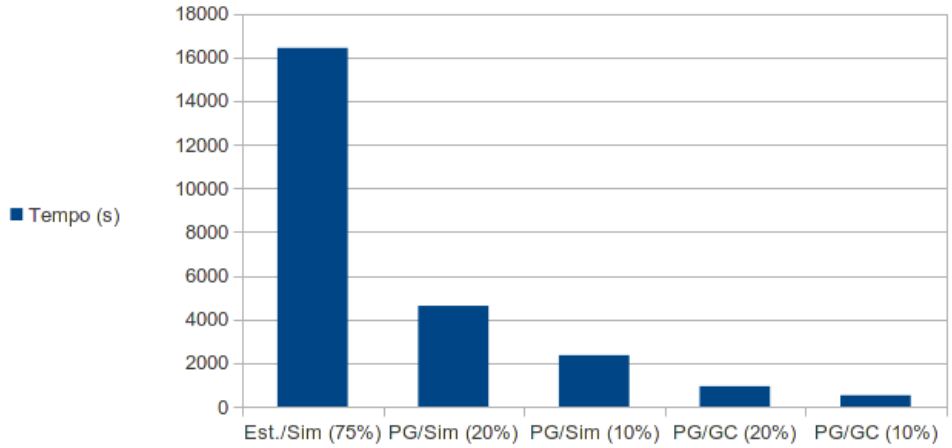


Figura 41 - Gráfico comparativo de desempenho entre as abordagens de busca de MLG propostas para a aplicação de multiplicação de matrizes.

A Tabela 26 mostra um comparativo entre as abordagens propostas com a abordagem de formulação de MLG por regressão *stepwise,* , apresentando os ganhos relativos de desempenho e precisão. Observa-se que em ambas as abordagens propostas, para as duas aplicações, houve ganho de desempenho considerável, variando entre 71,91% e 96,86%.

Analisando ainda a Tabela 26, o uso da abordagem de programação genética com simulações aumentou a precisão dos resultados, mostrando que o método de programação genética é mais efetivo para obtenção de MLGs mais ajustados. Já o uso da abordagem que

agrega programação genética com grafos de comunicação resultou em uma pequena perda da precisão das estimativas, em relação à abordagem de regressão *stepwise*.

Tabela 26 - Ganho relativo de desempenho e precisão das abordagens propostas em relação a abordagem que utiliza regressão *stepwise*.

Métrica	**PG/Sim**				**PG/GC**			
Aplicação	**OR**		**MM**		**OR**		**MM**	
Conjunto	**10%**	**20%**	**10%**	**20%**	**10%**	**20%**	**10%**	**20%**
Desempenho	80,40%	66,96%	85,66%	71,91%	93,14%	92,41%	96,86%	94,31%
Precisão	0,88%	0,85%	1%	1,23%	-1,24%	-1,04%	-3,96%	-0,51%

Legenda:
OR = Ordenação por Radix
MM = Multiplicação de Matrizes

Em termos gerais, qualificando as abordagens propostas neste trabalho de tese, considerando os estudos de caso apresentados neste capítulo, pode-se concluir que o uso de conjuntos de treinamento maiores permite obtenção de estimativas mais precisas. Porém, utilizando-se as técnicas com programação genética pode-se obter resultados equivalentes com redução desses conjuntos. Além disso, a utilização das técnicas de programação genética permitiu ainda aumentar consideravelmente o desempenho na obtenção dos MLGs como modelos de comunicação das aplicações estudados. Destaca-se a técnica de programação genética com grafos de comunicação como a técnica que apresenta o melhor desempenho para obtenção do MLG, levando a uma pequena redução de precisão de estimativas de desempenho de comunicação.

A Tabela 27 mostra um comparativo entre a abordagem proposta neste trabalho de tese com os trabalhas da literatura estudados, sob as características observadas nas abordagens para análise de comunicação. Nela vemos que a abordagem proposta neste trabalho de tese (coluna mais à direita com rótulo “AP”) suporta totalmente ou parcialmente todas as características levantadas.

Com os resultados apresentados na Tabela 27, vimos que, com uso das abordagens propostas neste trabalho de tese, foi possível obter uma técnica para exploração eficiente com estimativas bastante precisas dos desempenhos de uma determinada aplicação para as configurações de barramento contidas no espaço de projeto.

Tabela 27 - Comparativo entre abordagens de análise de comunicação da literatura com a abordagem proposta

Abordagem / Caracter.	Simulação de sistema (3.2)					Estimação Estática (3.3)			Híbrida (3.4)							AP
	3.2.1	3.2.2	3.2.3	3.2.4	3.2.5	3.3.1	3.3.2	3.3.3	3.4.1	3.4.2	3.4.3	3.4.4	3.4.5	3.4.6	3.4.7	
Precisão	x	x	x	x	x	p	p	p	p	x	-	p	x	x	x	x
Desempenho	-	-	-	-	-	p	x	x	p	-	-	p	p	-	-	x
Compromisso	x	x	x	x	x	-	-	-	-	x	-	-	-	-	-	x
Concorrência	-	-	-	-	-	-	-	-	-	x	-	-	-	x	x	x
Detalhamento	x	p	x	x	x	p	-	-	x	-	-	p	x	-	-	x
Flexibilidade	x	x	p	x	x	p	-	-	x	x	-	p	x	x	x	p
Escalabilidade	x	x	x	x	x	x	x	-	x	x	-	x	x	x	x	p
Automatização	-	p	p	-	p	-	-	-	-	x	x	x	-	x	x	p

Legenda:
- = não suportada
x = suportada
p = parcialmente suportada

O uso da abordagem de estimativas por grafos de comunicação permitiu reduzir o tempo necessário para estimação de desempenho de comunicação para uma configuração do espaço de projeto, removendo-se assim a necessidade de mais de uma simulação completa do sistema. Além disso, com o perfil de comunicação gerado na primeira fase do fluxo de projeto proposto, é possível analisar a comunicação da aplicação antes da seleção da estrutura de comunicação.

A abordagem de programação genética com simulações permitiu reduzir os tamanhos dos conjuntos de treinamento, necessários para obter um MLG, aumentando mais ainda o desempenho de exploração de comunicação. A natureza da técnica de GP permitiu avaliar paralelamente diversos MLGs e automatizar a seleção do melhor. Os MLG encontrados puderam ser utilizados estimar os desempenhos de comunicação para todo o espaço de projeto e com isso obter as melhores configurações de barramento.

Dois estudos de caso foram utilizados para validar o uso da abordagem proposta. Porém, os resultados da análise mostraram que são necessários estudos de caso com aplicações mais complexas, maior quantidade de parâmetros de configuração de barramentos e de elementos de plataforma. Só assim poderão ser efetivadas as análises das características de escalabilidade e flexibilidade para a abordagem proposta.

As técnicas propostas neste trabalho de tese foram codificadas como ferramentas de apoio à análise de comunicação em plataformas virtuais e estão em fase de integração ao Framework PDesigner. O próximo capítulo fornece descrições dessas ferramentas.

Capítulo 6
Implementação da Abordagem Proposta

6.1 Introdução

A abordagem híbrida para análise de comunicação, proposta neste trabalho de tese, foi implementada como um conjunto de ferramentas que suportam exploração dos espaços de configurações de barramentos em projetos baseados em plataforma. Estas ferramentas estão em fase de integração ao framework PDesigner [98], com objetivo de dar suporte ao projeto de comunicação de sistemas embarcados. Elas permitem explorar diversas configurações de barramentos considerando uma única simulação da aplicação em um modelo de plataforma executável.

Como um suporte para simplificar o processo de exploração de espaços de projeto de comunicação, as ferramentas permitem ao projetista definir as configurações que irão compor o espaço de projeto, estimar os desempenhos de comunicação para cada uma dessas configurações, além de gerar o perfil de comunicação para uma determinada configuração de barramento.

6.2 Seleção de Barramento Ideal e Simulação do Sistema

O framework PDesigner possui uma paleta de componentes gráficos utilizados para construção de modelos simuláveis de plataformas virtuais. Essa paleta possui elementos de processamento, memórias e periféricos, bem como barramentos, incluindo a implementação do barramento com comunicação ideal (ver Subseção 4.3.1.2), chamado de *ZeroBus*. A Figura 42 ilustra o projeto de um sistema embarcado utilizando o Framework PDesigner.

Nessa ferramenta, o projeto de um sistema embarcado é iniciado com a seleção dos componentes de plataforma desejados, através da paleta de componentes, ver Figura 42 (a), inserção na área de projeto, ver Figura 42 (b), configuração do componente, Figura 42 (c),

conexão entre as interfaces desses componentes, para compor a plataforma virtual, e, por fim, simulação.

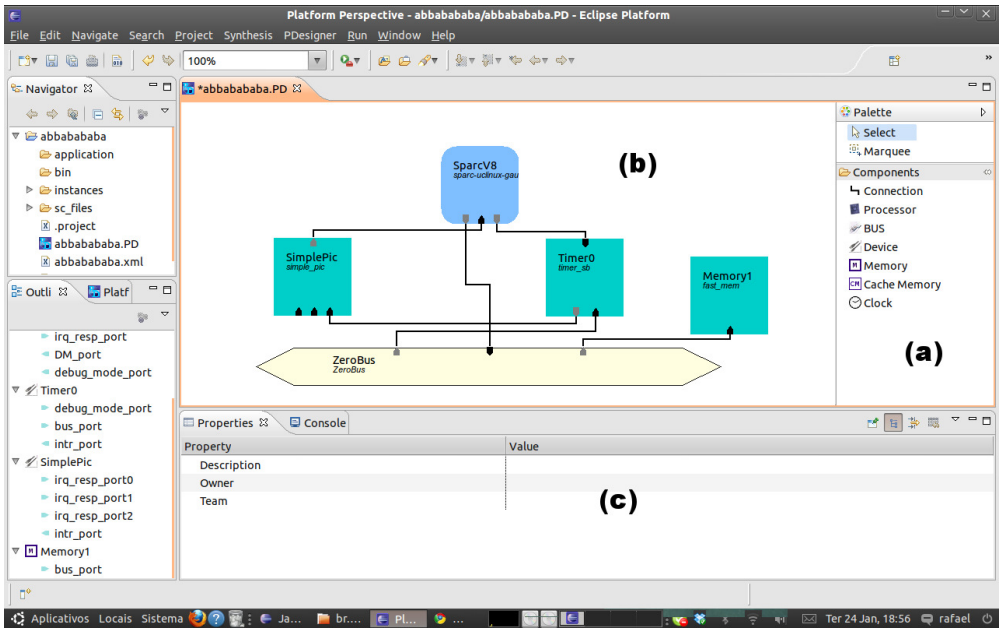


Figura 42 - Framework PDesigner para suporte gráfico no projeto de estruturas de comunicação de sistemas embarcados.

Para o projeto de comunicação, adiciona-se às plataformas o barramento *ZeroBus* para extração, durante a simulação do sistema, do perfil de comunicação da aplicação, que será gravado no disco rígido.

O perfil de comunicação pode ser utilizado para estimar a comunicação de uma configuração de barramento específica, ou ainda para estimação de desempenho de comunicação para espaço de projeto.

6.3 Definição de Perfil de Barramento

Além do perfil de comunicação, obtido a partir de simulação com o barramento *ZeroBus*, é necessário definir as configurações de barramento do espaço de projeto que terão seus respectivos desempenhos estimados. Essas configurações deverão estar inseridas no perfil de barramento.

Para a criação de um perfil de barramento, foi desenvolvida uma ferramenta que consiste de um simples formulário, onde os campos a serem preenchidos referem-se às configurações de barramento que irão compor o espaço de projeto. Ao fim do preenchimento do formulário, o perfil de barramento é armazenado em disco. As Figura 43 e 44 mostram exemplos de formulários preenchidos com uma única configuração e várias configurações diferentes de barramento, respectivamente.

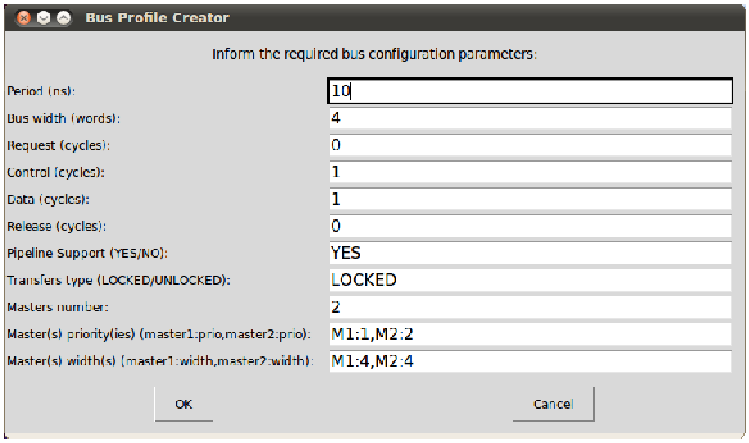


Figura 43 - Exemplo de perfil de barramento com uma única configuração

A Figura 43 mostra a criação de um perfil de barramento com as seguintes configurações: período de 10ns; barramento de dados com largura de 32 bits (4 palavras de 8 bits); número de ciclos para requisição de uso do barramento, envio de sinais de controle, envio de dados e liberação do barramento de 0, 1, 1 e 0, respectivamente; suporte a operações com pipeline; transferências bloqueadas (sem preempção); dois mestres de barramento; primeiro mestre (M1) com prioridade máxima (valor 1) de uso do barramento e segundo mestre (M2) com menor prioridade; e, por fim, tamanho da palavra de transferência de 32 bits (4 palavras) para transferência de dados nos dois mestres (M1 e M2).

Já a Figura 44 ilustra a criação de um perfil de barramento com um espaço de configurações maior, o qual inclui as seguintes configurações: períodos de 5, 6 e 10ns; larguras do barramento de 8, 16 e 32 bits (1, 2 e 4 palavras, respectivamente); 0 e 2 ciclos para requisição de uso do barramento, 2 ciclos para envio de sinais de controle, 2 ciclos para envio de dados e liberação de barramento sem consumo de ciclos; com e sem suporte a operações com pipeline; suporte a transferências bloqueadas (sem preempção) e desbloqueadas (com preempção); 3

mestres de barramento, onde o primeiro mestre (M1) tem prioridade máxima (valor 1) e os dois outros mestres (M2 e M3) possuirão prioridades intermediária (valor 2) e mínima (valor 3); e, por fim, tamanho da palavra de transferência dos dois mestres, 32 bits (4 palavras) para os dois mestres (M1 e M2).

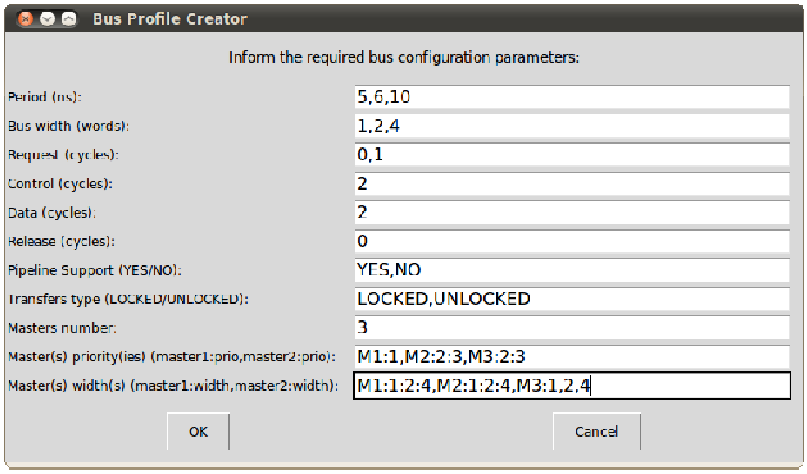


Figura 44 - Exemplo de perfil de barramento com várias configurações

6.4 Estimação de Desempenho de Comunicação para Espaço de Projeto de Comunicação

O processo de estimação de comunicação inicia com a seleção dos arquivos em disco que contém os perfis de comunicação e de barramento. Em seguida deve ser informado o tamanho do conjunto de treinamento, como pode ser visto na Figura 45, para que seja aplicada a técnica de projeto de experimento para selecionar as configurações do perfil de barramento, as quais terão os desempenhos estimados por grafos de comunicação.

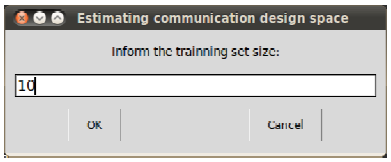
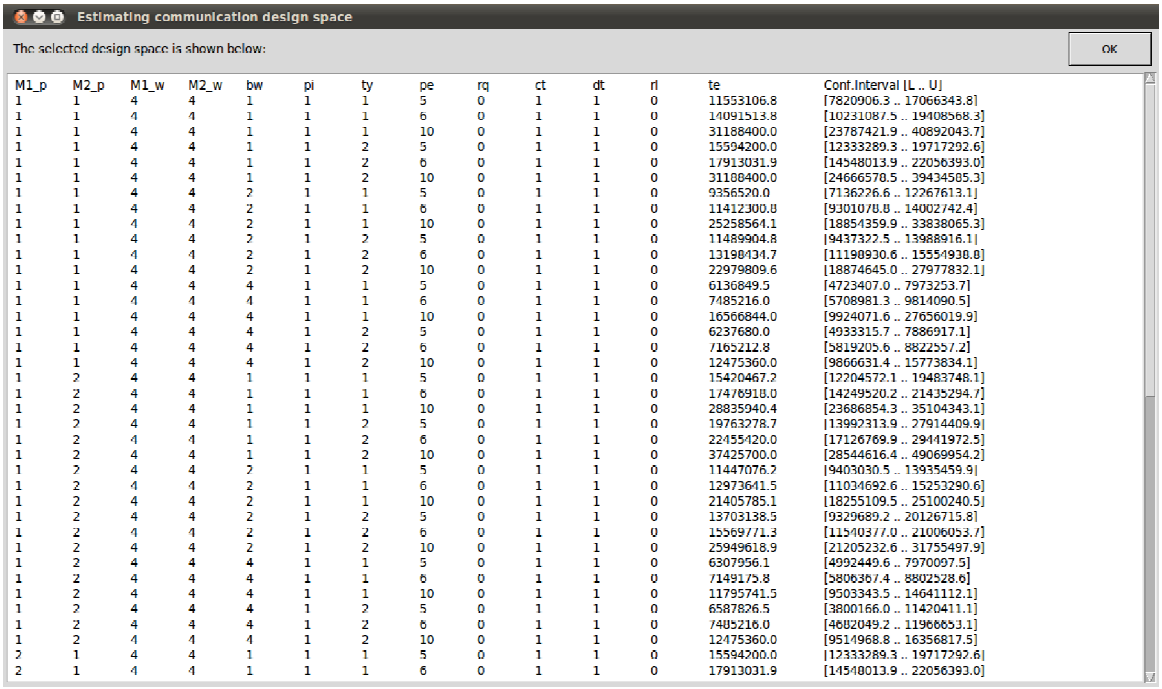


Figura 45 - Definição do tamanho do conjunto de treinamento para estimação de desempenho de subespaço de projeto.

Em seguida, essas estimativas serão utilizadas, juntamente com o algoritmo de programação genética, para gerar um modelo linear generalizado. O modelo final será utilizado para estimar o desempenho de comunicação da aplicação, bem como o intervalo de confiança, para todo o espaço de projeto de configurações de barramento, como pode ser visto na Figura 46.



M1_p	M2_p	M1_w	M2_w	bw	pi	ty	pe	rq	ct	dt	rl	te	Conf.Interval [L .. U]
1	1	4	4	1	1	1	5	0	1	1	0	11553106.8	[7820906.3 .. 17066343.8]
1	1	4	4	1	1	1	6	0	1	1	0	14091513.8	[10231087.5 .. 19408568.3]
1	1	4	4	1	1	1	10	0	1	1	0	31188400.0	[23787421.9 .. 40892043.7]
1	1	4	4	1	1	2	5	0	1	1	0	15594200.0	[12333289.3 .. 19717292.6]
1	1	4	4	1	1	2	6	0	1	1	0	17913031.9	[14548013.9 .. 22056393.0]
1	1	4	4	1	1	2	10	0	1	1	0	31188400.0	[24666578.5 .. 39434585.3]
1	1	4	4	2	1	1	5	0	1	1	0	9356520.0	[7136226.6 .. 12267613.1]
1	1	4	4	2	1	1	6	0	1	1	0	11412300.8	[9301078.8 .. 14002742.4]
1	1	4	4	2	1	1	10	0	1	1	0	25258564.1	[18854359.9 .. 33838065.3]
1	1	4	4	2	1	2	5	0	1	1	0	11489904.8	[9437322.5 .. 13988916.1]
1	1	4	4	2	1	2	6	0	1	1	0	13198134.7	[11190930.6 .. 15554930.8]
1	1	4	4	2	1	2	10	0	1	1	0	22979809.6	[18874645.0 .. 27977832.1]
1	1	4	4	4	1	1	5	0	1	1	0	6136849.5	[4723407.0 .. 7973253.7]
1	1	4	4	4	1	1	6	0	1	1	0	7485216.0	[5708981.3 .. 9814090.5]
1	1	4	4	4	1	1	10	0	1	1	0	16566844.0	[9924071.6 .. 27656019.9]
1	1	4	4	4	1	2	5	0	1	1	0	6237680.0	[4933315.7 .. 7886917.1]
1	1	4	4	4	1	2	6	0	1	1	0	7165212.8	[5819205.6 .. 8822557.2]
1	1	4	4	4	1	2	10	0	1	1	0	12475360.0	[9866631.4 .. 15773834.1]
1	2	4	4	1	1	1	5	0	1	1	0	15420467.2	[12204572.1 .. 19483748.1]
1	2	4	4	1	1	1	6	0	1	1	0	17476918.0	[14249520.2 .. 21435294.7]
1	2	4	4	1	1	1	10	0	1	1	0	28835940.4	[23686854.3 .. 35104343.1]
1	2	4	4	1	1	2	5	0	1	1	0	19763278.7	[13992313.9 .. 27914409.9]
1	2	4	4	1	1	2	6	0	1	1	0	22455420.0	[17126769.9 .. 29441972.5]
1	2	4	4	1	1	2	10	0	1	1	0	37425700.0	[28544616.4 .. 49069954.2]
1	2	4	4	2	1	1	5	0	1	1	0	11447076.2	[9403030.5 .. 13935459.9]
1	2	4	4	2	1	1	6	0	1	1	0	12973641.5	[11034692.6 .. 15253290.6]
1	2	4	4	2	1	1	10	0	1	1	0	21405785.1	[18255109.5 .. 25100240.5]
1	2	4	4	2	1	2	5	0	1	1	0	13703138.5	[9329689.2 .. 20126715.8]
1	2	4	4	2	1	2	6	0	1	1	0	15569771.3	[11540377.0 .. 21006053.7]
1	2	4	4	2	1	2	10	0	1	1	0	25949618.9	[21205232.6 .. 31755497.9]
1	2	4	4	4	1	1	5	0	1	1	0	6307956.1	[4992449.6 .. 7970097.5]
1	2	4	4	4	1	1	6	0	1	1	0	7149179.8	[5806367.4 .. 8802528.6]
1	2	4	4	4	1	1	10	0	1	1	0	11795741.5	[9503343.5 .. 14641112.1]
1	2	4	4	4	1	2	5	0	1	1	0	6587826.5	[3800166.0 .. 11420411.1]
1	2	4	4	4	1	2	6	0	1	1	0	7485216.0	[4682049.2 .. 11968653.1]
1	2	4	4	4	1	2	10	0	1	1	0	12475360.0	[9514968.8 .. 16356817.5]
2	1	4	4	1	1	1	5	0	1	1	0	15594200.0	[12333289.3 .. 19717292.6]
2	1	4	4	1	1	1	6	0	1	1	0	17913031.9	[14548013.9 .. 22056393.0]

Figura 46 - Estimativas de desempenho de comunicação para espaço de projeto de comunicação.

Na Figura 46, as três colunas mais à direita apresentam os desempenhos estimados, bem como os limites inferiores e superiores do intervalo de confiança para essas estimativas, respectivamente, medidos em nanossegundos. As demais colunas representam as respectivas configurações de barramento do espaço de projeto, como, por exemplo, largura de barramento (coluna *bw*) ou período entre ciclos de operação (coluna *pe*).

6.5 Geração de Perfil de Comunicação para Determinada Configuração

Por fim, é proposta uma ferramenta capaz de gerar o perfil de comunicação para uma determinada configuração de barramento. Essa ferramenta tem como entrada arquivos com perfil de comunicação inicial, gerado com o barramento *ZeroBus*, e com perfil de barramento, sendo que este último deverá conter apenas a configuração de barramento desejada. A Figura 47 apresenta parte de um perfil de comunicação gerado pela ferramenta aqui proposta.

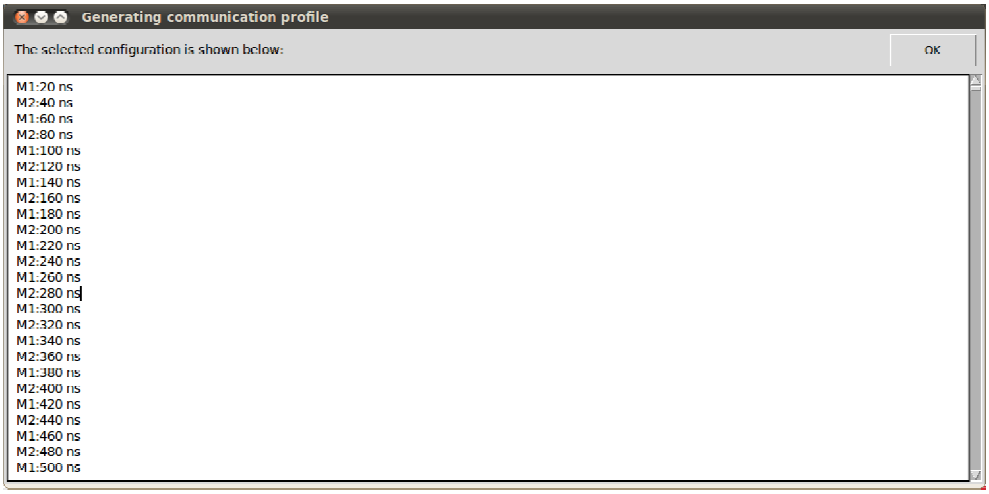


Figura 47 - Perfil de comunicação gerado através de grafos de comunicação para uma configuração de barramento.

A Figura 47 contém sequencias de transferências, e respectivos tempos de finalização, em nanossegundos, para os mestres barramento M1 e M2.

Essa ferramenta será útil para verificar as sequências de transações de barramento afim de encontrar gargalos de comunicação, como *starvation*.

6.6 Conclusão

As ferramentas propostas neste trabalho de tese foram codificadas as seguintes tecnologias:

- o *ZeroBus*, que é o modelo de barramento genérico capaz de capturar comunicação e gerar perfil de comunicação, foi codificado utilizando a linguagem de programação C++,

com as bibliotecas de modelagem de sistemas SystemC, e de suporte a transações SystemC TLM, disponíveis em [146].

- Para a ferramenta de definição do perfil de barramento, basicamente utilizou-se a linguagem de programação Python [148] e a biblioteca Python EasyGUI [150], para a interface gráfica de formulário.

- Para o desenvolvimento da ferramenta de estimação de desempenho de comunicação para o espaço de projeto de comunicação foram agregadas as seguintes tecnologias:

    - utilizou-se a ferramenta GPRSKit [41] para a seleção dos pontos do espaço de projeto para compor os conjuntos de treinamento. GPRSKit implementa, através de programação genética, um modelo de otimização da função objetivo definida na técnica de projeto de experimentos proposta por Audze-Eglais [35].

    - a linguagem de programação Python para codificar os mecanismos de geração e modificação de grafos de comunicação.

    - a linguagem de programação Python, com a biblioteca RPy [147], e a ferramenta R-Project [149] para codificar a ferramenta de busca de MLGs, por programação genética, e estimação de desempenho para os pontos do espaço de projeto.

    - A linguagem de programação Python com a biblioteca EasyGUI para as interfaces gráficas de janelas.

- A ferramenta de geração de perfil de comunicação a partir de grafos de comunicação foi codificada com linguagem de programação Python com a biblioteca EasyGUI [150].

A Tabela 28 apresenta o total de arquivos e de linhas de código utilizados para implementar cada uma das ferramentas propostas neste trabalho de tese.

Tabela 28 - Relação de arquivos e linhas para código para as ferramentas propostas

Ferramenta	Arquivos	Linhas de Código
Barramento ZeroBus	40	1984
Definição de Perfil de Barramento	1	33
Estimação de Desempenho do Comunicação do Espaço de Projeto	13	2930
Geração de perfil de comunicação por GC	2	506

Capítulo 7
Conclusões

7.1 Introdução

A abordagem de projeto baseado em plataformas permite aumento de desempenho, através do uso de modelos de simulação de plataformas de hardware preconcebidas, no desenvolvimento de novos sistemas embarcados. Essas plataformas podem conter diversos tipos de componentes, como processadores, memórias e periféricos. Para a troca de dados entre esses componentes é comum a utilização de barramentos como estruturas de comunicação.

No projeto dessas estruturas de comunicação, muitos parâmetros de configuração devem ser considerados, uma vez que podem influenciar o desempenho final do sistema. De forma a otimizar as configurações do barramento para uma aplicação, pode-se utilizar plataformas virtuais simuláveis contendo modelos configuráveis de barramentos. A vantagem deste tipo de técnica é que se pode obter estimativas precisas do desempenho de comunicação. Porém para identificação das melhores configurações, ou alguma que atenda as restrições de projeto, é necessário simular todo o espaço de projeto, o que pode ser um processo computacionalmente inviável. Por outro lado, foram propostos modelos estáticos de barramentos, que utilizam funções determinísticas, com objetivo de estimar espaços de projeto com maior desempenho, porém obtendo precisão reduzida nas estimativas. Trabalhos recentes agregam outros tipos de técnicas, como, por exemplo, heurísticas, às simulações do sistema e estimativas estáticas, buscando aumento de desempenho e estimativas de comunicação mais precisas, porém pecam em outros aspectos, como automatização e flexibilidade da abordagem de exploração.

Neste trabalho de tese, uma nova técnica híbrida foi proposta para, através de uma única simulação do sistema, obter um modelo estático, baseado em modelos estatísticos não-determinísticos, para modelar a comunicação de uma aplicação. A técnica proposta busca reduzir o tempo de exploração do espaço de projeto, além de disponibilizar estimativas precisas do desempenho de comunicação para uma determinada aplicação.

A técnica proposta é baseada em uma única simulação da aplicação mapeada para uma plataforma virtual. Esta plataforma contém um modelo de barramento genérico capaz de capturar a comunicação entre componentes e gerar um perfil de comunicação da aplicação. Com o perfil comunicação é possível estimar o desempenho de comunicação para as configurações de barramento do espaço de projeto, através de estimativas obtidas por grafos de comunicação. Usando a técnica proposta baseada em programação genética foi possível obter um modelo da comunicação (baseado em modelos lineares generalizados).

Entre as contribuições deste trabalho de tese, podemos destacar:

- uso de modelos lineares generalizados para modelar a estrutura de comunicação de uma aplicação. Com esta técnica foi possível controlar os erros de estimativas, através do uso de distribuições estatísticas teóricas, e, com isso, obter maior precisão nos desempenhos estimados de comunicação. Outras vantagens incluem: estudo das influências de cada parâmetro de configuração de barramento no desempenho final do sistema e validação formal do modelo de comunicação, através de testes de aderência;
- uso de abordagem de programação genética para busca paralelizada de modelos lineares generalizados, caracterizando-se assim como uma nova abordagem automatizada para formulação de obtenção desses modelos. Além disso, foi possível, juntamente com técnicas de projeto de experimentos para seleção de pontos dos conjuntos de treinamento, e estimativas de desempenho por grafos de comunicação, reduzir potencialmente os tempos para formulação dos modelos lineares generalizados finais;
- uma nova abordagem de suporte à análise de comunicação, a qual permite ao projetista explorar de forma eficiente as configurações do barramento para uma determinada plataforma. Assim, através do ajuste da comunicação, pode-se, consequentemente,
- aumentar o desempenho do sistema e trazer qualidade à implementação final.

7.2 Trabalhos Futuros

Atualmente, a abordagem proposta para estimação de desempenho de comunicação por grafos de comunicação suporta os seguintes parâmetros de configuração: prioridade de uso do barramento, largura do barramento de dados, *pipeline*, transações com bloqueio e sem bloqueio, transações

simples e por rajada, número de ciclos para as fases de requisição de uso do barramento, envio de sinais de controle, envio de dados e liberação do barramento, bem como período entre ciclos. Para aumentar o espaço de projeto, disponibilizando mais opções de configurações de barramento, sugere-se como trabalho futuro a inclusão de novos parâmetros de configuração, como por exemplo:

- tamanhos de palavra diferentes para os mestres do barramento;
- outros tipos de transferências, como lê-modifica-escreve e divisão;
- novos mecanismos de arbitragem.

O presente trabalho focou na customização de estruturas de comunicação compostas por apenas um barramento. Em projetos de sistemas embarcados mais complexos, se faz necessário o uso de mais de um barramento, constituindo-se assim uma hierarquia. Assim, as técnicas propostas deverão suportar a exploração de configurações de hierarquias de barramentos, bem como parâmetros de configuração das respectivas pontes de comunicação.

Outro trabalho a ser realizado refere-se à customização do algoritmo de programação genética, visando aumento de desempenho e soluções mais refinadas. Propõe-se o estudo e experimentação de outras técnicas, como operadores de seleção *steady-state* [30], de cruzamento e mutação adaptativos [142].

Além de exploração para otimização de desempenho de comunicação, a técnica proposta poderia ser adaptada para customizar a estrutura de comunicação para redução do consumo de energia. A proposta seria poder realizar estimação de consumo de energia para cada configuração de barramento, contida no espaço de projeto, utilizando modelos de energia de barramentos, como, por exemplo, o apresentado em [143].

Por fim, a integração total das técnicas propostas e ferramentas desenvolvidas, as quais foram apresentadas no Capítulo 6, ao Framework PDesigner. Com o PDesigner é possível, graficamente, a construção de plataformas virtuais e configuração de seus componentes. Assim, com a integração, as ferramentas propostas poderão oferecer suporte à exploração rápida de estruturas de comunicação, além de disponibilizar mecanismos para análise de configurações individuais.

Referências Bibliográficas

[1] MCCULLAGH, P. and NELDER, J. A. ***Generalized Linear Models***. Chapman and Hall. 1989.

[2] JIAO, Y.; CHEN,Y.; SHNEIDER, D. and WROBLEWSKI, J.***A simulation study of impacts of error structure on modeling stock-recruitment data using generalized linear models***. NRC Research Press Website: http://www.nrcresearchpress.com/, 2004.

[3] ANDERSON, D.; FELDBLUM, S.; MODLIN, C.; SCHIRMACHER, D.; SCHIRMACHER,E. and THANDI, N. ***A Practitioner's Guide to Generalized Linear Models***. Watson Wyatt Worldwide, 2007.

[4] BLACK, D. C. and DONOVAN, J. ***SystemC: From the Groung Up***. In Kluwer Academic Publishers., 2004.

[5] NULL, L. and LOBUR, J. ***The Essencials of Computer Organization and Architecture***. In Jones and Bartlett Publishers, 2003.

[6] AGP. ***Intel AGP V3.0 Interface Specification***. Rev. 1.0. Disponível em: http://download.intel.com/support/motherboards/desktop/sb/agp30.pdf. Data do último acesso: 19 de janeiro de 2012.

[7] S. I. G. PCI. ***PCI Local Bus Specification, Production Version. Rev. 2.1***. disponível em: http://www.pcisig.com/members/downloads/specifications/conventional/conventional_pci_2_3.pdf. Data do último acesso: 19 de janeiro de 2012.

[8] VSI Alliance. ***Virtual Component Interface Standard***, Version 2 (OCB 2 2.0), April 2001.

[9] VAHID, F. and GIVARGIS, T. ***Embedded System Design: A Unified***

Hardware/Software Introduction. In John Wiley & Sons, Inc., 2002.

[10] DANDAMUDI, S. P. ***Fundamentals of Computer Organization and Design***. Springer, 2003.

[11] ESMERALDO, G. A. R. M. ***Mecanismos de Suporte à Modelagem e Análise de Comunicação em Plataformas Multiprocessadoras***. Dissertação de Mestrado. Programa de Pós-Graduação em Ciência da Computação, CIn – UFPE. Março, 2007.

[12] LAHIRI, K., RAGHUNATHAN, A., DEY S., ***System-Level Performance Analysis for Designing On-Chip Communication Architectures***. In IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, vol. 20, no.6, pp.768-783, June 2001.

[13] IPEK, E.; McKEE, S. A.; SUPINSKI, B. R.; CARUAMA, R. ***Efficiently Exploring Architectural Design Spaces via Predictive Modeling***. *In* 12th International Conference on Architectural Support for Programming Languages and Operating Systems, 2006.

[14] COOK, H. and SKADRON, K. ***Predictive Design Space Exploration Using Genetically Programmed Response Surfaces***. In Proceedings of the Design Automation Conf. (DAC), 2008.

[15] ARAUJO, S.G.; MESQUITA, A.C.; PEDROZA, A.C.P. ***Optimized datapath design by evolutionary computation***. In Proceedings of System-on-Chip for Real-Time Applications, 2003.

[16] DENIZIAK, S. and GORSKI, A. ***Hardware/Software Co-synthesis of Distributed Embedded Systems Using Genetic Programming***. Springer Berlin / Heidelberg, 2008.

[17] GEN, M. and CHENG, R. ***Genetic algorithms and engineering optimization.***Wiley, 2000.

[18] KOZA, J. R.; KEANE, M. A.; STREETER, M. J.; MYDLOWEC, W.; YU, J.; LANZA,

G. ***Genetic Programming IV: Routine Human Competitive Machine Inteligence***. Springer, 2005.

[19] MICHALEWICZ, Z. ***Genetic Algorithms + Data Structures = Evolution Programs***. Springer, 1998.

[20] NELDER, J. A. and WEDDERBURN, R. W. M. ***Generalized linear models***. Journal of the Royal Statistical Society, A, 135, 370{384.25, 1972.

[21] ESMERALDO, G. A. R. M. and BARROS, E. N. S. ***A Genetic Programming Based Approach for Efficiently Exploring Architectural Communication Design Space of MPSOCS***. In: Proceedings of IEEE VI Southern Programmable Logic Conference, 2010.

[22] ANTONY, J. ***Design of Experiments for Engineers and Scientists.*** Butterworth-Heinemann, 2003.

[23] BAILEY, B.; MARTIN, G.; PIZIALI, A. ***ESL Design and Verification: A Prescription for Electronic System Level Methodology***. Morgan Kaufmann/Elsevier, 2007.

[24] AZEVEDO, R.; RIGO, S.; BARTHOLOMEU, M.; ARAUJO, G.; ARAUJO, C.; BARROS, E. ***The ArchC architecture description language and tools***. Int. J. Parallel Program.,33(5):453–484, 2005.

[25] PASRICHA, S.; DUTT, N.; BEN-ROMDHANE, M. ***Using TLM for Exploring Bus-based SoC Communication Architectures***. International Conference on Application-specific Systems, Architectures and Processors, Samos, Greece, 2005.

[26] CHELLAPILLA, K. ***Evolving Computer Programs Without Subtree Crossover.*** IEEE. Transactions on Evolutionary Computation, 1(3):209–216, September 1997.

[27] BOX, G. E. P. and COX, D. R. ***An analysis of transformations (with discussion).***Journal of the Royal Statistical Society, B, 26, 211–252, 1964.

[28] MICHALEWICZ, Z. ***Heuristic methods for evolutionary computation techniques.*** Journal of Heuristics, 1:177–206, 1995.

[29] AHN, C. W. and RAMAKRISHNA, R. S. ***Elitism-Based Compact Genetic Algorithms***. IEEE Transactions On Evolutionary Computation, Vol. 7, No. 4, 2003.

[30] MITCHELL, M. ***An Introduction to Genetic Algorithms.*** MIT Press, 1999.

[31] DEAN, A. and VOSS, D. ***Design and Analysis of Experiements***. Springer, 1999.

[32] COX, D. E. ***The Theory of the Design of Experiments***. Chapman and Hall/CRC, 2000.

[33] LASIC, R. Ž. ***Design of Experiments in Chemical Engineering, A Pratical Guide***. Wiley, 2004.

[34] ARLOT, S. and CELISSE, A. ***A survey of cross-validation procedures for model selection***. Statistics Surveys, Volume 4, pp. 40-79, 2010.

[35] AUDZE, P. and EGLAIS, V. ***A new approach to the planning out of experiments***. Problems of dynamics and strength, volume 35, 1977.

[36] BATES, S. J., SIENZ, J. and LANGLEY, D. S. ***Formulation of the Audze-Eglais Uniform Latin Hypercube Design of Experiments***. Adv. Eng. Software, 34(8):493{506, Jun. 2003.

[37] WOO, S. C., OHARA, M., TORRIE, E., SINGH, J. P. and GUPTA, A. ***The SPLASH-2 Programs: Characterization and Methodological Considerations***. In Proceedings of the 22nd International Symposium on Computer Architecture, pages 24-36, Santa Margherita Ligure, Italy, June 1995.

[38] LAMPORT, L. A ***New Solution of Dijkstra's Concurrent Programming Problem***. Communications of the ACM 17, 8 (August 1974), 453-455.

[39] CORMEN, T.H., LEISERSON, C.E. RIVEST, R.L. and STEIN, C. ***Introduction to Algorithms***. McGraw-Hill and The Mit Press, 2001.

[40] WEISBERG, S. ***Applied Linear Regression, Willey and Sons, Third Edition***.Wiley, 2005.

[41] GPRSKit. ***Genetically Programmed Respone Surfaces Kit***. Homepage: http://www.cs.berkeley.edu/~hcook/gprs.html. Data do último acesso: 19 de janeiro de 2012.

[42] TANENBAUM, A. S., ***Sistemas Operacionais Modernos***, Prentice-Hall, 2003.

[43] ArchC. ***The Architecture Description Language***. Homepage: http://archc.sourceforge.net/. Data do ultimo acesso: 19 de janeiro de 2012.

[44] BISHOP, C. M. ***Pattern Recognition and Machine Learning, 2nd. Edition***. Springer, 2007.

[45] BURD, S.D. ***Systems Architecture, 6th Edition***. Cengage Learning, 2011.

[46] KOZA, J. R. ***Genetic Programming On the Programming of Computers by Means of Natural Selection***. MIT Press, 1998.

[47] FEITOSA, R. F. F. ***Laconibot: Um Agente para atuar em leilões do tipo CDA do TAC***. Dissertação de Mestrado. Programa de Pós-Graduação em Ciência da Computação, UECE. Agosto, 2009.

[48] ARM AMBA. ***AMBA Specification rev. 2.0, IHI-0011A***, May 1999. Disponível em: http://www.arm.com/products/system-ip/amba/amba-open-specifications.php. Data do último acesso: 19 de janeiro de 2012.

[49] ROWSON, J. A. and SANGIOVANNI-VINCENTELLI. A. ***Interface based design***. In Proceedings of the Design Automation Conference (DAC), pages 178–183, June 1997.

[50] OGAWA , O.; NOYER , S. B.; CHAUVET,P.; SHINOHARA, K.; WATANABE, NIIZUMA,Y. H.,SASAKI, T. and TAKEI, Y. ***A practical approach for bus architecture optimization at transaction level***. In Proceedings of the Conference on Design, Automation and Test in Europe (DATE), 2003.

[51] SHIN, C.;KIM, Y.-T.;CHUNG, E.-Y.; CHOI, K.-M.;KONG, J.-T. and EO, S.-K. ***Fast Exploration of Parameterized Bus Architecture for Communication-Centric SoC design***. In Proceedings of the conference on Design, automation and test in Europe (DATE), 2004.

[52] LOGHI,M.;ANGIOLINI, F.;BERTOZZI, D.; BENINI, L. and ZAFALON, R. ***Analyzing On-Chip Communication in a MPSoC Environment***. In Proceedings of the conference on Design, automation and test in Europe (DATE), 2004.

[53] DUMITRASCU, F.; BACIVAROV, I.; PIERALISI, L.; BONACIU, M. and JERRAYA, A. A. ***Flexible MPSoC Platform with Fast Interconnect Exploration for Optimal System Performance for a Specific Application***. In Proceedings of the conference on Design, automation and test in Europe (DATE), 2006.

[54] PASRICHA, S. ***Transaction Level Modeling of SoC with SystemC 2.0***. In Synopsys User Group Conference (SNUG), 2002.

[55] YEN, T. and WOLF, W. ***Communication synthesis for distributed embedded systems***. In Proceedings of the Int. Conf. Computer-Aided Design, pages 288–294, November 1995.

[56] DAVEAU, J.; ISMAIL,T. B. and JERRAYA, A. A. ***Synthesis of system-level communication by an allocation based approach***. In Proceedings of the Int. Symp. System Level Synthesis, pages 150–155, September 1995.

[57] DEY, S. and BOMMU, S. ***Performance analysis of a system of communication processes***. In Proceedings of the Int. Conf. Computer-Aided Design, pages 590–597, November, 1997.

[58] KNUDSEN, P. and MADSEN, J. ***Integrating communication protocol selection with partitioning in hardware/software codesign***. In Proceedings of the Int. Symp. System Level Synthesis, pages 111–116, December, 1998.

[59] GASTEIER, M. and GLESNER, M. ***Bus-based communication synthesis on system level***. In ACM Trans. Design Automation Electronic Systems, pages 1–11, January 1999.

[60] KIM, S.; IM, C. and HA, S. ***Schedule-aware performance estimation of communication architecture for efficient design space exploration***. In Proceedings of the Intl. Conf. on Hardware/Software Codesign and System Synthesis, pages 195–200, 2003.

[61] CHO, Y.-S. ; CHOI,E.-J. and CHO, K.-R. ***Modeling and analysis of the system bus latency on the SoC platform***. In Proceedings of the International Workshop on System-Level Interconnect Prediction, pages 67–74, 2006.

[62] KNUDSEN, P.V. and MADSEN, J. ***Communication Estimation for Hardware/Software Codesign.*** Sixth International Workshop on Hardware/Software Co-Design (CODES'98), 1998.

[63] MADL, G.; PASRICHA, S.; ZHU, Q.; BATHEN, L. A. D. and DUTT, N. ***Formal performance evaluation of AMBA-based system-on-chip designs***. In Proceedings of the 6th ACM and IEEE International Conference on Embedded Software, pages 22–25, 2006.

[64] WIEFERINK, A.; MEYR, H. and LEUPERS, R. ***Retargetable Processor System Integration Into Multi-Processor System-on-Chip Platforms***. Springer, 2008.

[65] RTEMS. Real Time for Multiprocessor Systems. Homepage: http://www.rtems.com/. Data do último acesso: 19 de janeiro de 2012.

[66] SWARM. ***Software ARM***. Homepage: http://www.cl.cam.ac.uk/~mwd24/phd/swarm.html. Data do último acesso: 19 de janeiro de 2012.

[67] LIEVERSE, P.; WOLF,P. V. D. and DEPRETTERE, E. ***A trace transformation technique for communication refinement***. In Proceedings of the Symposium on Hardware/Software Codesign (CODES), 2001.

[68] KIM, S.; IM, C. and HA, S. ***Efficient exploration of on-chip bus architectures and memory allocation***. In Proceedings of the 2nd IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis, pages 248–253, 2004.

[69] WILD, T.; HEKERSDORF, A. and OHLENDORF, R. ***Performance evaluation for system on-chip architectures using trace-based transaction level simulation***. In Proceedings of the conference on Design, automation and test in Europe (DATE), pages 248–253, 2006.

[70] JOO, Y.-P.; KIM, S. and HA, S. ***On-chip communication architecture exploration for processor-pool-based MPSoC***. DATE, 2009.

[71] LUKASIEWYCZ, M.; STREUBÜHR, M.; GLAß, M.; HAUBELT, C. and TEICH, J. ***Combined System Synthesis and Communication Architecture Exploration for MPSoCs***. In Proceedings of Design, Automation and Test in Europe (DATE 2009), pp. 472-477, 2009.

[72] HOLLAND, J. H. ***Adaptation in Natural and Artificial Systems***. University of Michigan Press, Ann Arbor, 1975.

[73] HERRERA, F.; LOZANO, M.; VERDEGAY, J. L.***Tackling Real-Coded Genetic Algorithms – Operator and Tools for Behavioural Analysis***. Artificial Intelligence Review 12, pag. 265-319, Kluwer Academic Publishers, Amsterdã, Holanda, 1998.

[74] ARABAS, J.; MICHALEWICZ, Z. MULAWKA, J. J. ***GAVaPS - A Genetic Algorithm with Varying Population Size***. International Conference on Evolutionary Computation 1994: 73-78.

[75] ULLMAN, J. D. ***Compiladores: Princípios, Técnicas e Ferramentas***. Prentice Hall, 2008.

[76] SANGIOVANNI-VINCENTELLI, A. ***Defining platform-based design***. In EDesign of EETimes, 2002.

[77] HAN, S.-I.;BAGHDADI, A.;BONACIU, M.;CHAE,S.-I. and JERRAYA, A.A. ***An Efficient Scalable and Flexible Data Transfer Architecture for Multiprocessor SoC with Massive Distributed Memory.*** Design Automation Conference (DAC), 2004.

[78] BRANDWAJN,A. ***A note on SCSI bus waits***. ACM SIGMETRICS Perf. Eval. Rev., vol. 30, pp. 41–47, Sep. 2002.

[79] PAOLUCCI, P.S.; JERRAYA, A.A.; LEUPERS, R.; THIELE, L. and VICINI, P. ***SHAPES: a tiled scalable software hardware architecture platform for embedded systems***. In Proc. CODES+ISSS, pp.176-172, October 2006.

[80] HAN, K.-H. and KIM,J.-H. ***Quantum-inspired evolutionary algorithm for a class of combinatorial optimization***. IEEE TEC, vol.6, no.6, pp.580-593, December 2002.

[81] LUKASIEWYCZ, M.; GLAß, M.; HAUBELT, C. and TEICH, J. ***SATDecoding in Evolutionary Algorithms for Discrete Constrained Optimization Problems***. In Proc. of CEC '07, pages 935–942, 2007.

[82] CHAI, D. and KUEHLMANN, A. ***A fast pseudo-Boolean constraint solver***. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2005.

[83] MADSEN, I.; GRODE, J.;KNUDSEN, P. V.; PETERSEN, M. E. and HAXTHAUSEN, A. ***LYCOS: the Lyngby Co-Synthesis System***. Design Automation for Embedded Systems, 2(2):195 - 235,1997.

[84] VIANA, P. ***A methodology to explore memory hierarchy architectures for embeddedSystems.*** Tese de Doutorado. Programa de Pós-Graduação em Ciência da Computação, CIn – UFPE., Agosto, 2006.

[85] WIEFERINK, A.; KOGEL, T.; LEUPERS, R.; ASCHEID, G.; MEYR, H.; BRAUN, G. and NOHL, A. ***A system level processor/communication co-exploration methodology formulti-processor system-on-chip platforms***. In Proceedings of the Int. Conf. on Design, Automation and Test in Europe (DATE), February 2004.

[86] JERRAYA, A. A. and WOLF, W. ***Multiprocessor systems-on-chips***. In Morgan Kaufmann, September 2004.

[87] LAHIRI, K.; RAGHUNATHAN, A. and DEY, S. ***Efficient exploration of the SoC communication architecture design space***. In Proceedings of the Intl. Conf. on Computer Aided Design, pages 424–430, 2000.

[88] MOSELEY, T.;KIM, J.; CONNORS. D. and GRUNWALD, D. ***Methods for Modeling Resource Contention on Simultaneous Multithreaded Processors***. Proc. 23rd Int'l Conf. Computer Design (ICCD), 2005.

[89] MARIANI, G.; PALERMO, G.; ZACCARIA, V.; BRANKOVIC, A.; JOVIC, J. and SILVANO, C. ***A Correlation-Based Design Space Exploration Methodology for Multi-Processor Systems-on-Chip***. Design Automation Conference (DAC), 2010, pp. 120-125.

[90] PASRICHA, S.; PARK, Y.; KURDAHI, F.J. and DUTT, N.D. ***CAPPS: A Framework for Power-Performance Tradeoffs in Bus-Matrix-Based On-Chip Communication Architecture Synthesis***. IEEE Trans. VLSI Syst.(2010) 209-221.

[91] PostgreSQL. ***Sistema de Banco de Dados Relacional PostgreSQL***. Homepage:http://www.postgresql.org/. Data do último acesso: 19 de janeiro de 2012.

[92] ORTEGA, R.B. and BORRIELLO, G. ***Communication synthesis for distributed embedded systems***. In Proceedings of ICCAD. 1998, 437-444.

[93] JERRAYA, A. A. ***Long term trends for embedded system design***. In Proceedings of the CEPA 2 Workshop - Digital Platforms for Defence, pages 15–16, 2005.

[94] PASRICHA, S.; DUTT, N. and BEN-ROMDHANE, M. ***Constraint-Driven Bus Matrix Synthesis for MPSoC***. In Proceedings of the Asia and South Pacific Design Automation Conference, 2006.

[95] KEUTZER, K.; MALIK, S.; NEWTON, A. R.; RABAEY, J. M. and SANGIOVANNI-VINCENTELLI, A.***System-level design: Orthogonalization of concerns and platform-based design***. In IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2000.

[96] BAILEY, B.; MARTIN, G. and PIZIALI, A.***ESL Design and Verification: A Prescription for Electronic System Level Methodology***. Morgan Kaufmann/Elsevier, 2007.

[97] CADENCE and COWARE. ***ConvergenSC / Incisive Design Flow***. White Paper. Rev. 2., 2004. Disponível em: http://www.cadence.com/. Data do último acesso: 19 de janeiro de 2012.

[98] AZEVEDO, R. J.; ARAÚJO, C.; Gomes, M.; BARROS, E.; RIGO, Sandro ; ARAÚJO, G. C. S. de. ***Platform Designer: An Approach for Modeling Multiprocessor Platforms based on SystemC***. Design Automation for Embedded Systems, v. 10, p. 253-283, 2005.

[99] PIMENTEL, A. D.; ERBAS, C. and POLSTRA, S. ***A Systematic Approach to Exploring Embedded System Architectures at Multiple Abstraction Levels***. In IEEE Transactions On Computers, 2004.

[100] NOERGAARD, T. ***Embedded Systems Architecture: A Comprehensive Guide for Engineers and Programmers (Embedded Technology)***. Newnes, 2005.

[101] WOLF, F. ***The future of multiprocessor systems-on-chips***. In Proceedings of 41st Design Automation Conference, 2004.

[102] MARTIN, G. ***Overview of the MPSoC design challenge***. In Proceedings of 43RD Design Automation Conference, 2006.

[103] XU, J.; WOLF, W.; HENKEL, J. and CHAKRADHAR, S. ***A methodology for design, modeling, and analysis of networks-on-chip***. In IEEE International Symposium on Circuits and Systems ISCAS., pages 1778–1781, 2005.

[104] BENINI, L. and MICHELI, G. D. ***Powering networks on chips***. In Proceedings of the Int. Symp. Syst. Level Synthesis, pages 33–38, 2001.

[105] HO, R.; MAI, K. W. and HOROWITZ, M. A. ***The future of wires***. In Proceedings of the IEEE, vol. 89, pages 490–504, April 2001.

[106] GUIBALY, F. E. ***Design and analysis of arbitration protocols***. In IEEE Transactions on Computers, pages 161–171, 1989.

[107] RAGHUNATHAN, V.; SRIVASTAVA, M. B. and GUPTA, R. K. ***A survey of techniques for energy-efficient on-chip communication***. In Proceedings of the Design Automation Conf. (DAC), pages 900–905, 2003.

[108] LAHIRI, K. and RAGHUNATHAN, A. ***Power analysis of system-level on-chip communication architectures***. In Proceedings of the Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS), pages 236–241, September 2004.

[109] EL-REWINI, H. and ABD-EL-BARR, M. ***Advanced computer architecture and parallel processing***. In John Wiley & Sons, Inc., 2005.

[110] PASRICHA, S.; DUTT, N. and BEN-ROMDHANE, M. ***Automated throughput-driven synthesis of bus-based communication architectures***. In Proceedings of the Asia and South Pacific Design Automation Conference, 2005.

[111] PASRICHA, S.; DUTT, N.; BOZORGZADEH, E. and BEN-ROMDHANE, M. ***Floorplan-aware Bus Architecture Synthesis*****.** CECS Technical Report 04-27, Oct 2004.

[112] AKL, C. J. and BAYOUMI, M.A. ***Transition Skew Coding: A Power and Area Efficient Encoding Technique for Global On-Chip Interconnects***. In Proceedings of the Asia and South Pacific Design Automation Conference, 2007.

[113] LI, Y. ***Physically Constrained Architecture for Chip Multiprocessors***. Doctor of Philosophy Thesis. School of Engineering and Applied Science, University of Virginia, 2006.

[114] LAHIRI, K.; RAGHUNATHAN, A. and DEY, S. ***System-Level Performance Analysis for Designing On-Chip Communication Architectures***. In IEEE Trans. on Computer Aided-Design of Integrated Circuits and Systems, 2001.

[115] SILVA, A. L. M.; ESMERALDO, G. A. R. M.; VIANA, P. da S.; BARROS, E. ***Cache-Analyzer: Design Space Evaluation of Configurable-Caches in a Single-Pass***. In 18th IEEE/IFIP International Workshop on Rapid System Prototyping, 2007.

[116] ESMERALDO, G. A. R. M. ; BARROS, E. ***Using SystemC for Flexible Bus Protocol Modeling and Communication Profiling Generation***. In Latin American SystemC User's Group Meeting, Natal/RN, 2009.

[117] MONTGOMERY, D. C. and RUNGER, G. C. ***Applied Statistics and Probability for Engineers, 3rd, edition***. Wiley, 2003.

[118] CHUN , Y.H. ***Monte Carlo analysis of estimation methods for the prediction of customer response patterns in direct marketing***. In European Journal of Operational Research, pp.673-678, 2012.

[119] STALLINGS, W. ***Computer Organization and Archtecture, 8th Ed***. Prentice Hall, 2009.

[120] S. I. G. PCI. ***PCI Express Base 3.0 Specification***. Disponível em: http://www.pcisig.com/specifications/pciexpress/base3. Data do último acesso: 19 de janeiro de 2012.

[121] KOZA, J.R. ***Genetic Programming***. The MIT Press, Cambridge, 1992.

[122] FAHRMEIR, L. and TUTZ, G. ***Multivariate Statistical Modelling Based on Generalized Linear Models***. Springer, 2001.

[123] HOCKING, R. R. ***The Analysis and Selection of Variables in Linear Regression***. Biometrics, 1976.

[124] SPRENT, N. and SMEETON, N. C. ***Applied Nonparametric Statistical Methods, Fourth Edition***. Chapman and Hall/CRC, 2007.

[125] BURNHAM, K. P. and ANDERSON, D. ***Model Selection and Multi-Model Inference, 2nd Ed***. Springer, 2002.

[126] DERKSEN, S. and KESELMAN, H. J. ***Backward, forward and stepwise automated subset selection algorithms: frequency of obtaining authentic and noise variables***. British Journal of Mathematical and Statistical Psychology 45: 265–282. 1992.

[127] FAY, M. P. and PROSCHAN, M. A. ***Wilcoxon-Mann-Whitney or t-test? On assumptions for hypothesis tests and multiple interpretations of decision rules***. Statistics Survey, Volume 4, 1-39, 2010.

[128] BOUAJILA, A.; ZEPPENFELD, J.; STECHELE, W.; BERNAUER, A.; BRINGMANN, O.; ROSENSTIEL, W. and HERKERSDORF, A. ***Autonomic System on Chip Platform***. In Organic Computing - A Paradigm Shift for Complex Systems. Springer, 2011.

[129] SAINI, P.; SINGH, M. and SINGH, B. ***VHDL Implementation of PCI Bus Arbiter Using Arbitration Algorithms***. In Communications in Computer and Information Science, Springer, 2011.

[130] ROSÉN, J.; NEIKTER, C.; ELES, P.; PENG, Z.; BURGIO, P. and BENINI, L. ***Bus Access Design for Combined Worst and Average Case Execution Time Optimization of Predictable Real-Time Applications on Multiprocessor Systems-on-Chip***. IIn IEEE Trans. on Real-Time and Embedded Technology and Applications Symposium (RTAS), 2011.

[131] McCULLOCH, C. E. and SEARLE, S. R. ***Generalized, Linear and Mixed Models***. Willey and Sons, Inc, 2001.

[132] ANDERSON, D.; FELDBLUM, S.; MODLIN, C.; SCHIRMACHER, D.; SCHIRMACHER, E. and THANDI, E. ***A Practitioner's Guide to Generalized Linear Models***. Watson Wyatt Worldwide, 2007.

[133] ALTERA AVALON. ***Avalon Interface Specification, Version 11. May, 2011***. Disponível em:http://www.altera.com/literature/manual/mnl_avalon_spec.pdf. Data do último acesso: 19 de janeiro de 2012.

[134] MADAR, J.; ABONYI, J. and SZEIFERT, F. ***Genetic Programming for the Identification of Nonlinear Input –Output Models***. InIndustrial and Engineering Chemistry Research, vol. 44, pp. 3178 – 3186, 2005.

[135] DOBSON, A. J. ***An Introduction to Generalized Linear Models***. ***Second Edition***. Chapman and Hall/CRC, 2002.

[136] HOCKING, R. R. ***Methods and Applications of Linear Models Regression and The Analysis of Variance***. ***Second Edition***. Wiley and Sons, inc, 2003.

[137] SAKAMOTO, Y., ISHIGURO, M., and KITAGAWA, G. ***Akaike Information Criterion Statistics***. Reidel, 1987.

[138] SCHWARZ, G. ***Estimating the dimension of a model***. Annals of Statistics, 6, 461–464, 1978.

[139] MALLOWS, C. ***Some comments on Cp***. Technometrics, 15, 661–676, 1973.

[140] O'BRIEN, R. M. ***A Caution Regarding Rules of Thumb for Variance Inflation Factors***. Quality and Quantity 41(5)673-690. Springer, 2007.

[141] ROSNER, B. ***Fundamentals of Biostatistics, Seventh Edition***. Brooks/Cole, 2010.

[142] SIVANANDAM, S. N. and DEEPA, S. N. ***Introduction to Genetic Algorithms***. Springer, 2008.

[143] CALDARI, M.; CONTI, M.; COPPOLA, M.; CRIPPA, P.; ORCIONI, S.; PIERALISI, L. and TURCHETTI, C. ***System-Level Power Analysis Methodology Applied to the***

AMBA AHB Bus. In Proceedings of the conference on Design, Automation and Test in Europe, 2003.

[144] PASRICHA, S. ***On-chip communication architectures: system on chip interconnect***. Morgan Kaufmann, 2008.

[145] NADLER, B. ***Nonparametric Detection of Signals by Information Theoretic Criteria: Performance Analysis and an Improved Estimator***. In IEEE Transactions On Signal Processing, VOL. 58, NO. 5, May, 2010.

[146] ACCELLERA. ***Accellera Systems Initiative***. Homepage: http://www.accellera.org/home/. Data do último acesso: 19 de janeiro de 2012.

[147] RPY. ***R for Python: A simple and efficient access to R from Python***. Disponível em: http://rpy.sourceforge.net/. Data do último acesso: 19 de janeiro de 2012.

[148] PYTHON. ***Python Programming Language***. Homepage: http://python.org/. Data do último acesso: 19 de janeiro de 2012.

[149] R-PROJECT. ***The R Project for Statistical Computing***. Homepage: http://www.r-project.org/. Data do último acesso: 19 de janeiro de 2012.

[150] EASYGUI. ***EasyGUI Homepage***. Homepage: http://easygui.sourceforge.net/. Data do último acesso: 19 de janeiro de 2012.

[151] HAWKINS, D. ***Identification of Outliers***. Chapman and Hall, 1980.

[152] BARNETT, V. and LEWIS, T. ***Handbook of Outliers***. New York: Wiley, 1994.

[153] REDDY, T. A. ***Applied Data Analysis and Modeling for Energy Engineers and Scientists***. Springer, 2011.

[154] SCHUENEMEYER, J.and DREW, L. ***Statistics for Earth and Environmental Scientists***. Wiley, 2011.

[155] SHANTI, D. and AMUTHA, R. ***Simulation of Inter Processor Communication Architecture in MPSoC***. In European Journal of Scientific Research, Vol.72 No.1, pp. 74-83, 2012.

[156] SIALA, M. A. and SAOUD, S. B. ***A Survey on Existing MPSOCs Architectures***. In The International Journal of Computer Applications,Vol. 19, No.3, 2011.

[157] WESTERBERG, C. H. and LEVINE, J. ***GenPlan: Combining Genetic Programming and Planning***. In Proceedings of The 19th Workshop of the UK Planning and Scheduling Special Interest Group, The Open University, Milton Keynes, UK, 14-15pp, 2000.

Apêndice A

SystemC

Vários trabalhos na literatura de projeto de sistemas embarcados utilizam plataformas virtuais, com seus componentes descritos em SystemC. Além desses, o modelo de plataforma virtual proposto neste trabalho de tese também contém seus componentes descritos em SystemC.

SystemC é uma linguagem de projeto de sistemas embarcados que surgiu da necessidade pervasiva por uma linguagem que aumentasse a produtividade do projetista de sistemas eletrônicos [4]. Na realidade, SystemC não é uma linguagem, mas sim uma biblioteca de classes C++ que permite modelagem de software e hardware, criando, assim, uma abstração do sistema através de um modelo simulável.

Suas principais características incluem:

• **Modelo temporal:** SystemC define um modelo de tempo com 64 bits de resolução utilizando uma classe conhecida como *sc_time*. Possui um micro núcleo de simulação que permite modelar e avançar o tempo global, através da geração de eventos que simulam um relógio de sistema. Os componentes modelados com SystemC podem sincronizar suas ações com o relógio do sistema, implementado pelo micro núcleo, através de diretivas *wait()*. Essas diretivas são instruções que interrompem as ações de componente até que um novo evento seja disparado.

• **Tipos de dados de hardware:** não é possível ter-se a variedade de tipos de dados necessários para um hardware digital dentro dos limites dos tipos de dados nativos de C++. Assim, SystemC provê tipos de dados compatíveis com hardware que dão suporte à configuração em bits da largura do dado. Além disto, permite representar aos quatro estados da lógica digital *four-state* (0,1,X,Z) e todos os métodos necessários para utilizar estes tipos de dados, incluindo conversão entre os tipos de dados de hardware e conversão de tipos de dados de hardware para software.

• **Hierarquia e estrutura:** para modelar hierarquia de hardware, SystemC utiliza uma entidade chamada módulo. Estes módulos são interconectados através de canais de comunicação. A hierarquia surge da instanciação de classes de módulos dentro de outros módulos.

• **Gerenciamento de comunicação:** canais de comunicação representam um poderoso mecanismo para modelagem de comunicações. Conceitualmente, um canal é mais do que um simples sinal ou fio. Canais podem representar esquemas de comunicação mais complexos que eventualmente podem ser mapeados para hardware.

• **Concorrência:** simulação de execução paralela é efetuada pela simulação de cada unidade concorrente. Cada unidade pode ser executada até que a simulação de outras unidades seja necessária para manter os comportamentos alinhados no tempo.

Para modelar todas essas características, SystemC define vários tipos de componentes. A Figura 48 ilustra um sistema com os principais tipos.

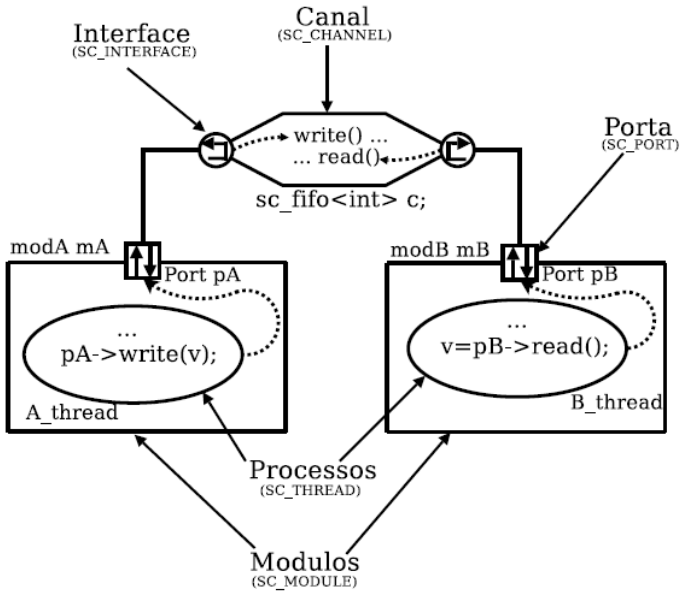


Figura 48 - Principais componentes de SystemC.

Como indicado anteriormente, o micro núcleo permite escalonar os processos de simulação, que são os processos da aplicação embarcada. Processos de simulação são simplesmente funções membro, que na Figura 48 é representado por uma *SC_THREAD*, de módulos, que, por sua vez, são definidos pela diretiva *SC_MODULE*. Para comunicarem-se, módulos utilizam portas (*SC_PORTS*). As portas são ligadas aos canais de comunicação, que são definidos por *SC_CHANNEL* e *SC_INTERFACE*.

Em SystemC, além de porta e canal, outra estrutura utilizada para comunicação entre módulos é o *SC_EXPORT*. Este, conceitualmente, é bastante semelhante a *SC_PORT*, apenas se diferencia pelo fato de não necessitar de um canal para comunicação. Módulos que se comunicam através de *SC_PORT* e *SC_EXPORT*, têm suas interfaces ligadas diretamente, como se estivem acessando um canal de comunicação. A Figura 49 ilustra a conexão de dois módulos, A e B, através de *SC_PORT* e *SC_EXPORT*.

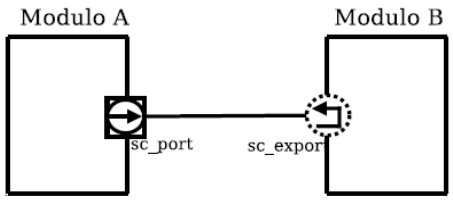


Figura 49 - Comunicação de dois módulos através de SC_PORT e SC_EXPORT.

SystemC, por ser uma biblioteca de classes C++, permite modelar componentes de hardware em diferentes níveis de abstração. Estes níveis podem ser divididos, basicamente, de acordo com as informações temporais que fornecem: precisão de ciclo (*cycle accurate*), tempo estimado (*time estimated*) e comportamental (*behavioral*). No primeiro nível, que possui o menor desempenho de simulação em relação aos demais modelos de alto nível, sua funcionalidade é regida por um relógio (oscilador) de sistema (também conhecido por *clock*). Por causa desta estrutura, os modelos possuem informações bastante precisas de temporização. O segundo nível não possui relógio de sistema, porém os modelos incluem estimativas dos tempos necessários para cada operação. O último nível não possui informações temporais. Este último, geralmente, é utilizado para validar as funções do componente e possui o maior desempenho de simulação.

Apêndice B

Conjuntos de Construção selecionados por Técnica de Projeto de Experimentos para Abordagem de Programação Genética com Modelos Lineares Generalizados

B.1 Ordenação por Radix

10% do espaço de projeto (7 pontos)

p1	p2	bw	ty	fr	te
1	1	4	2	6	7485708
2	1	1	2	6	22456896
1	1	1	1	5	24952110
2	1	4	1	10	12476080
2	2	2	1	6	14971272
1	2	4	2	10	12476180
2	2	1	2	10	37428500

20% do espaço de projeto (14 pontos)

p1	p2	bw	ty	fr	te
1	2	4	2	5	6238090
1	1	1	2	6	22457100
2	1	4	1	5	6238040
2	2	2	2	6	14971272
2	1	2	1	6	14971278
2	1	1	1	10	49904230
2	1	2	2	10	24951900
1	2	1	2	10	37428500
1	1	1	1	5	24952110
1	2	2	1	10	24952120
1	2	4	1	6	7485642
2	2	4	1	10	12476070
2	2	1	2	5	18714250
1	1	4	2	6	7485708

B.2 Multiplicação de Matrizes

10% do espaço de projeto (48 pontos)

p1	p2	p3	p4	bw	ty	fr	te
1	3	4	4	4	2	10	53204350
1	2	3	2	4	2	10	65655070
1	2	4	4	1	2	5	85527080
1	4	2	2	2	2	6	79001568

1	2	2	2	4	1	5	32066050
1	4	3	4	2	1	6	87184968
1	4	2	4	4	1	6	43927062
1	2	3	4	1	1	10	217534480
1	3	2	2	1	2	6	119021052
1	4	2	3	1	2	10	198864120
1	4	3	2	2	2	10	131669280
1	3	2	4	4	2	10	53182820
1	2	2	3	4	1	10	52545860
1	4	2	3	4	2	10	65957730
1	4	4	2	2	1	10	159227630
1	4	3	2	4	1	6	43591902
1	2	2	3	1	1	5	86377255
1	2	2	3	4	2	10	53182820
1	2	4	4	1	1	5	122150360
1	4	4	4	2	1	10	124226870
1	4	2	2	2	1	10	129514600
1	2	4	2	4	2	6	39393042
1	4	4	3	1	2	10	198368420
1	2	4	2	4	1	5	34436260
1	4	4	2	1	2	6	119021052
1	3	2	3	1	1	6	143863452
1	4	4	4	4	1	5	32066050
1	4	3	4	4	2	6	31909692
1	3	3	2	4	1	6	47313876
1	4	2	3	1	1	6	144272826
1	2	2	4	4	2	6	31909692
1	2	2	2	2	1	10	124226870
1	2	3	3	1	1	10	244300720
1	4	4	4	4	2	10	65815770
1	3	3	2	1	2	10	198368420
1	3	4	2	1	1	5	106240305
1	4	3	4	1	1	6	143863452
1	2	3	3	1	2	6	102632496
1	2	2	4	1	2	5	85521090
1	4	3	4	1	2	5	85521090
1	2	4	2	1	2	6	118726452
1	2	4	3	2	1	6	93781278
1	3	4	3	4	2	6	39393042
1	3	4	2	4	1	10	65566620
1	4	4	4	2	2	6	79001568
1	2	3	4	2	2	10	106401860
1	2	3	4	4	1	6	36380022
1	3	2	2	1	1	5	98201190

20% do espaço de projeto (96 pontos)

p1	**p2**	**p3**	**p4**	**bw**	**ty**	**fr**	**ts**	**p1**	**p2**	**p3**	**p4**	**bw**	**ty**	**fr**	**ts**
1	2	2	4	2	2	10	106361000	1	3	3	4	2	1	10	105013770
1	3	3	4	1	2	6	102625308	1	4	3	4	1	1	6	143863452
1	4	4	3	1	2	10	198368420	1	2	3	2	2	1	6	82740090
1	4	2	4	2	1	6	87184968	1	4	2	4	1	2	5	85521090
1	3	3	4	2	2	10	106361000	1	2	4	2	1	1	5	102664535
1	4	3	2	4	1	6	43591902	1	4	4	4	4	1	6	38479260
1	4	2	2	4	1	5	32438605	1	2	2	4	1	2	6	102625308
1	3	2	2	1	1	5	98201190	1	4	4	4	4	2	6	39489462
1	4	4	3	1	1	5	126332705	1	4	3	2	2	1	6	87240774
1	4	4	3	4	1	5	39428230	1	4	3	4	1	2	10	171042180
1	3	4	4	4	2	10	53204350	1	3	3	4	4	1	6	31527516
1	2	2	2	2	1	6	74536122	1	3	4	2	1	2	6	118726452
1	2	3	2	1	2	6	118726452	1	3	2	2	1	2	5	99184210
1	3	2	3	4	2	6	31909692	1	2	3	4	2	1	6	72681414
1	4	4	2	2	2	10	131669280	1	4	2	3	1	1	10	240454710
1	3	3	3	2	2	6	79001568	1	2	4	4	4	1	10	76755900
1	4	4	2	2	2	10	131669280	1	3	4	3	4	1	6	41323512
1	3	4	2	2	1	5	64521810	1	3	2	4	1	2	10	171042180
1	3	4	2	4	1	6	39339972	1	3	4	2	2	2	6	78785352
1	4	4	2	1	2	5	99184210	1	4	2	2	4	2	5	32907885
1	2	2	4	2	2	5	53180500	1	4	3	3	4	1	10	64877210
1	3	4	4	4	2	6	31922610	1	4	4	3	4	1	6	47313876

1	2	4	4	2	2	10	106401860	1	2	2	3	4	1	5	26272930
1	2	4	3	1	2	10	198267160	1	2	4	3	4	2	10	65848410
1	3	2	3	4	2	10	53182820	1	3	2	3	1	2	6	102625308
1	4	2	2	2	2	6	79001568	1	2	4	4	4	2	5	26602175
1	3	3	3	4	1	10	64132100	1	4	3	4	4	2	6	31909692
1	2	4	4	1	1	6	146580432	1	4	3	4	2	1	6	87184968
1	4	3	3	2	2	10	131669280	1	2	2	2	4	2	6	39489462
1	3	3	4	2	2	5	53180500	1	3	4	4	1	2	5	85527080
1	3	4	3	4	1	10	68872520	1	2	4	2	4	2	5	32827535
1	2	2	3	4	2	10	53182820	1	3	3	4	1	1	5	86377255
1	2	2	3	4	1	10	52545860	1	4	2	2	2	1	5	64757300
1	3	2	2	1	1	10	196402380	1	3	4	2	2	2	10	131308920
1	2	2	3	1	2	5	85521090	1	4	2	2	2	1	10	129514600
1	2	3	4	4	1	5	30316685	1	4	4	2	2	1	5	79613815
1	4	2	4	2	2	6	63816600	1	4	2	4	4	2	10	53182820
1	4	2	3	4	2	10	65957730	1	3	4	4	2	1	6	92674152
1	2	4	4	2	1	10	154456920	1	3	3	4	4	1	10	52545860
1	2	3	2	1	1	6	123197442	1	4	4	4	2	1	10	124226870
1	3	3	2	1	2	10	198368420	1	2	3	4	1	2	5	85527080
1	2	4	2	2	1	10	137900150	1	2	3	3	1	1	10	244300720
1	2	3	2	4	1	6	41323512	1	2	2	3	4	2	6	31909692
1	2	2	2	2	1	10	124226870	1	2	2	4	2	1	10	105013770
1	4	4	3	1	2	6	119021052	1	4	3	2	1	1	10	242337550
1	2	4	3	2	2	6	79022040	1	3	2	2	2	2	10	131669280
1	4	2	2	1	2	5	99184210								

Apêndice C

Diagramas de para Análise Residual Gráfica de Modelos Lineares Generalizados Encontrados por Programação Genética

C.1 Ordenação por Radix

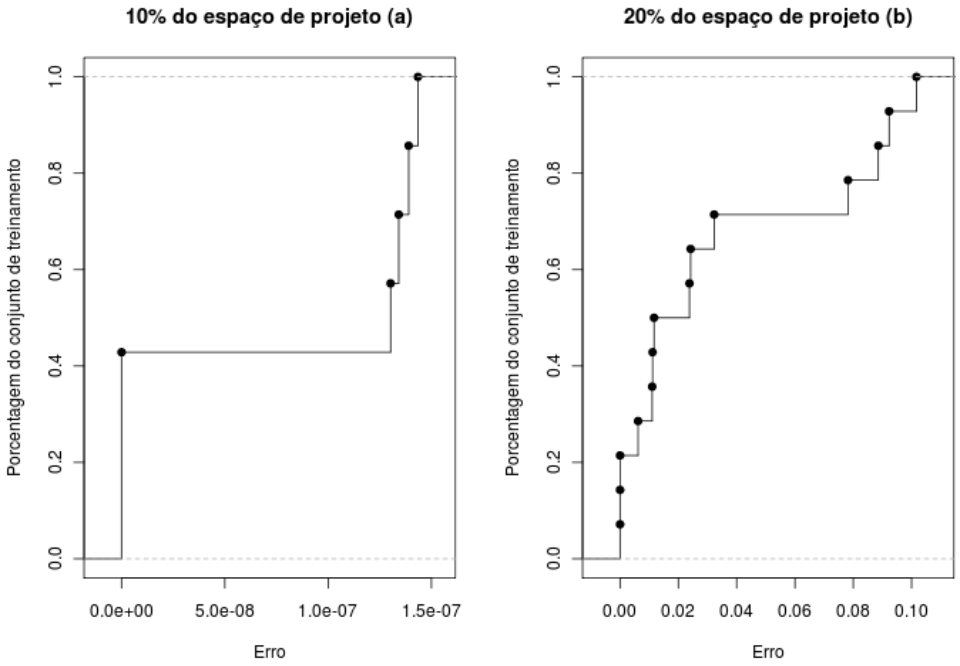


Figura 50 - Gráficos de erros acumulados para MLGs ajustados a partir dos conjuntos de treinamento com (a) 10% e (b) 20% do espaço de projeto da aplicação de ordenação por radix.

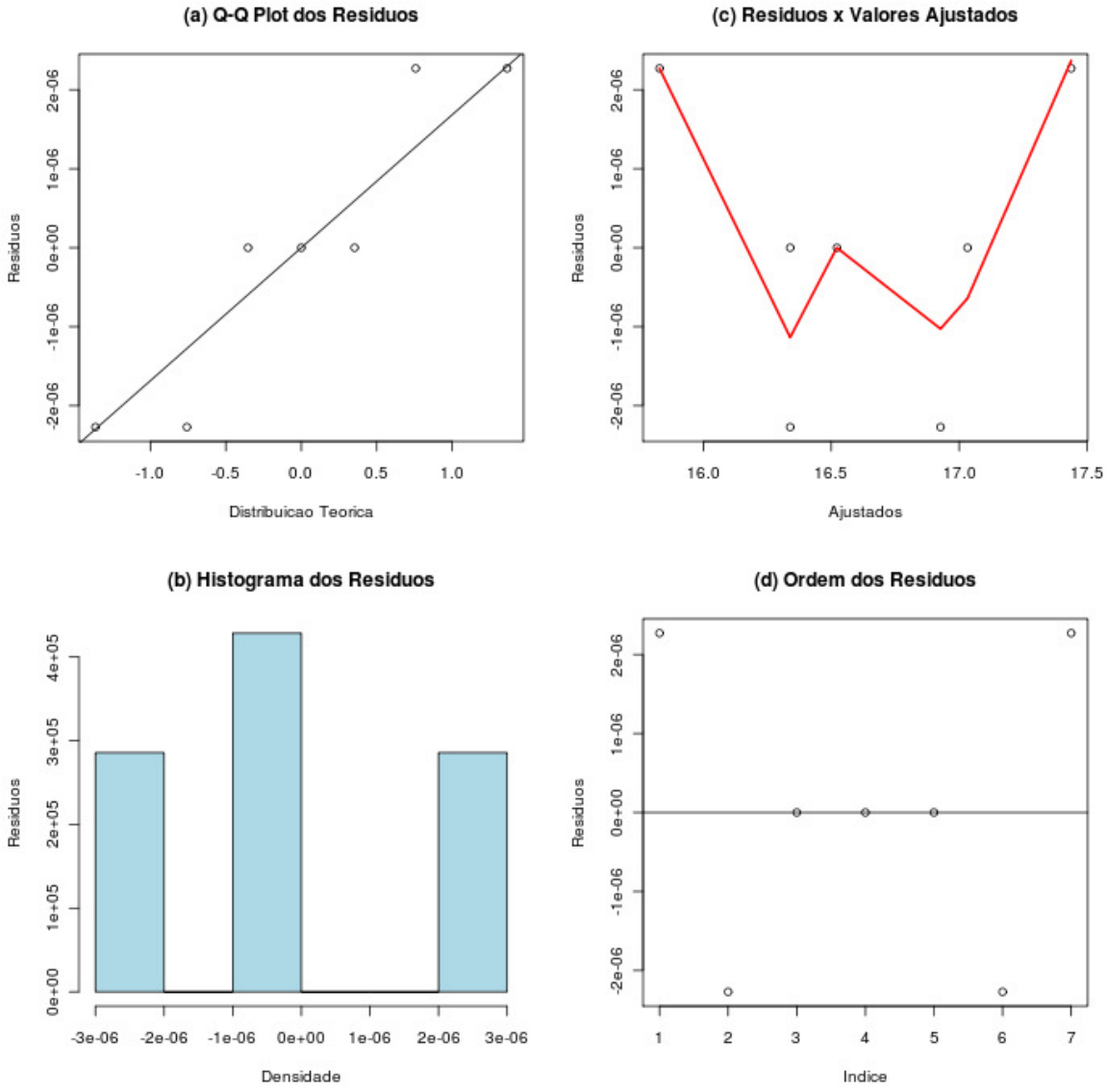


Figura 51 - Gráficos para análise de suposições sobre distribuição dos erros para a conjunto de treinamento com 10% do espaço de projeto da aplicação de ordenação por radix.

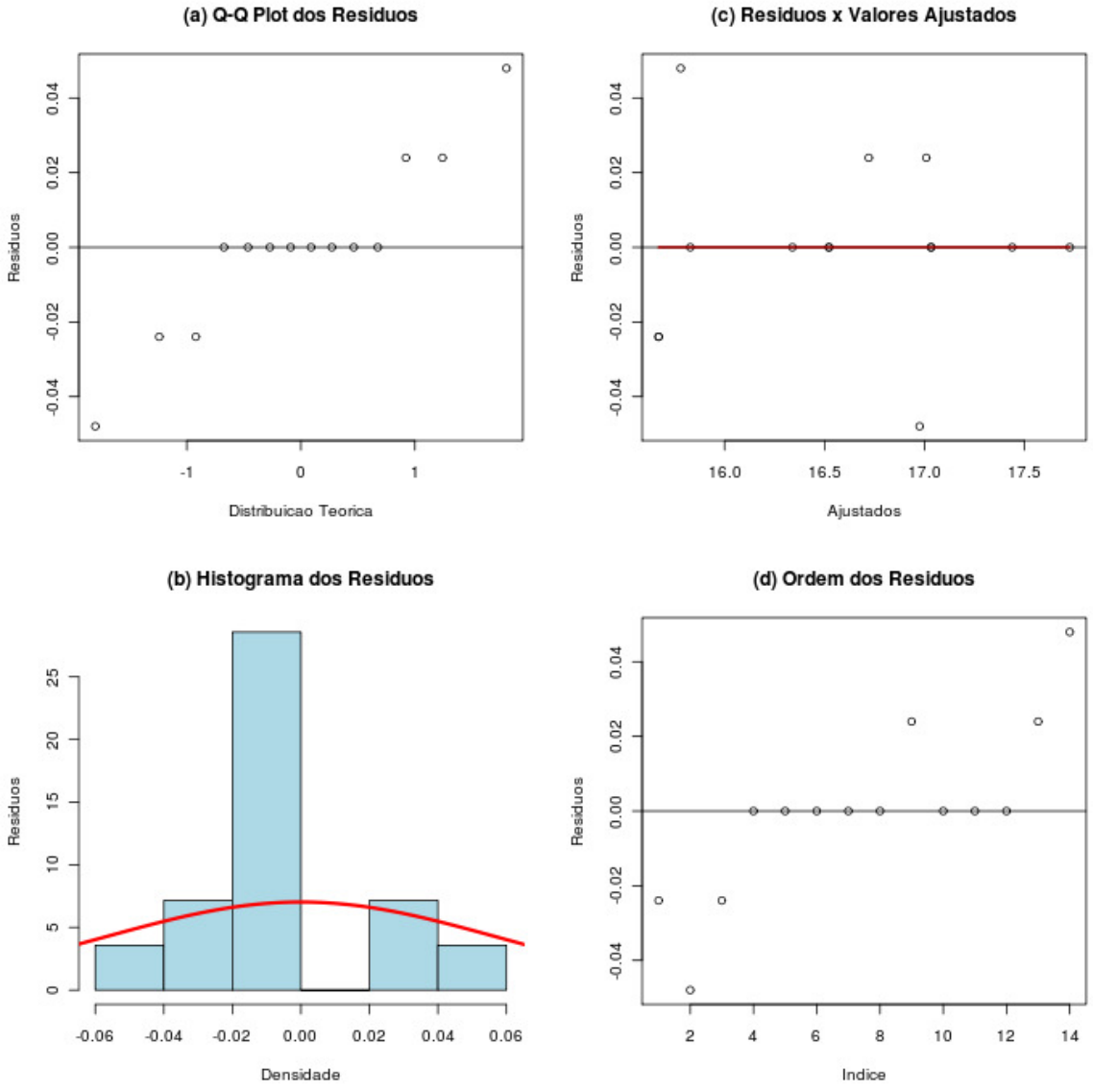


Figura 52 - Gráficos para análise de suposições sobre distribuição dos erros para a conjunto de treinamento com 20% do espaço de projeto da aplicação de ordenação por radix.

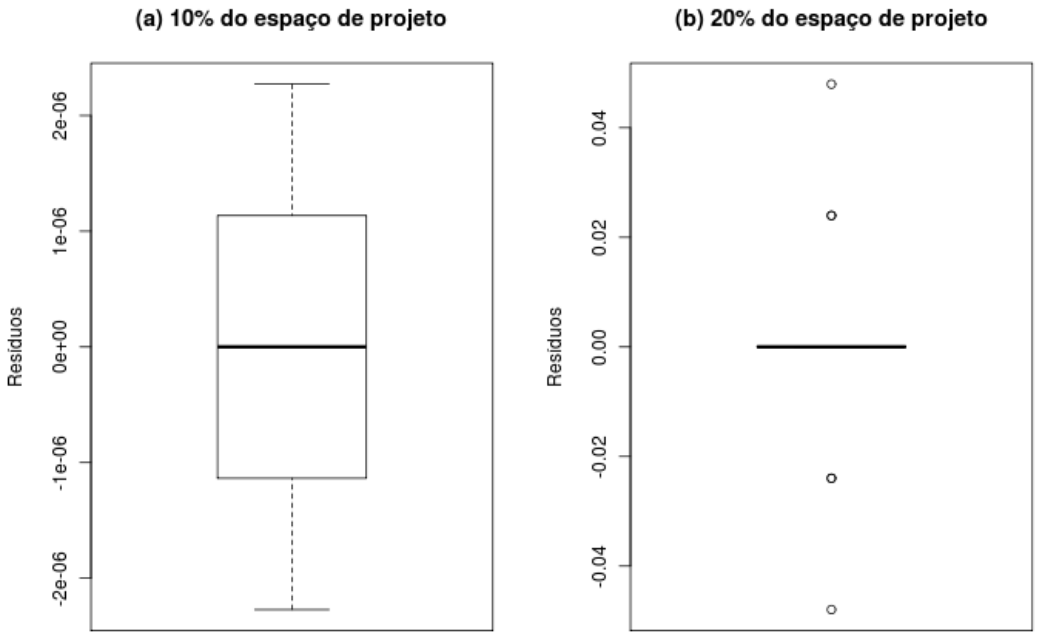


Figura 53 - Gráficos Boxplot para os resíduos dos MLGs formulados a partir dos conjuntos de treinamento com tamanhos de (a) 10% e (b) 20% do espaço de projeto da aplicação de ordenação por radix.

C.2 Multiplicação de Matrizes

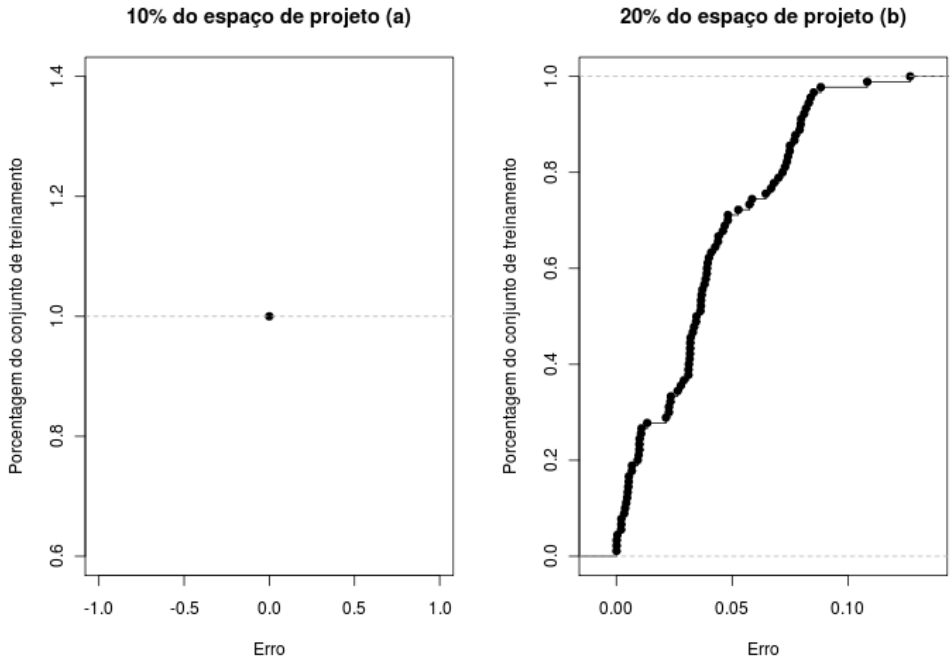


Figura 54 - Gráficos de erros acumulados para MLGs ajustados a partir dos conjuntos de treinamento com (a) 10% e (b) 20% do espaço de projeto da aplicação de multiplicação de matrizes.

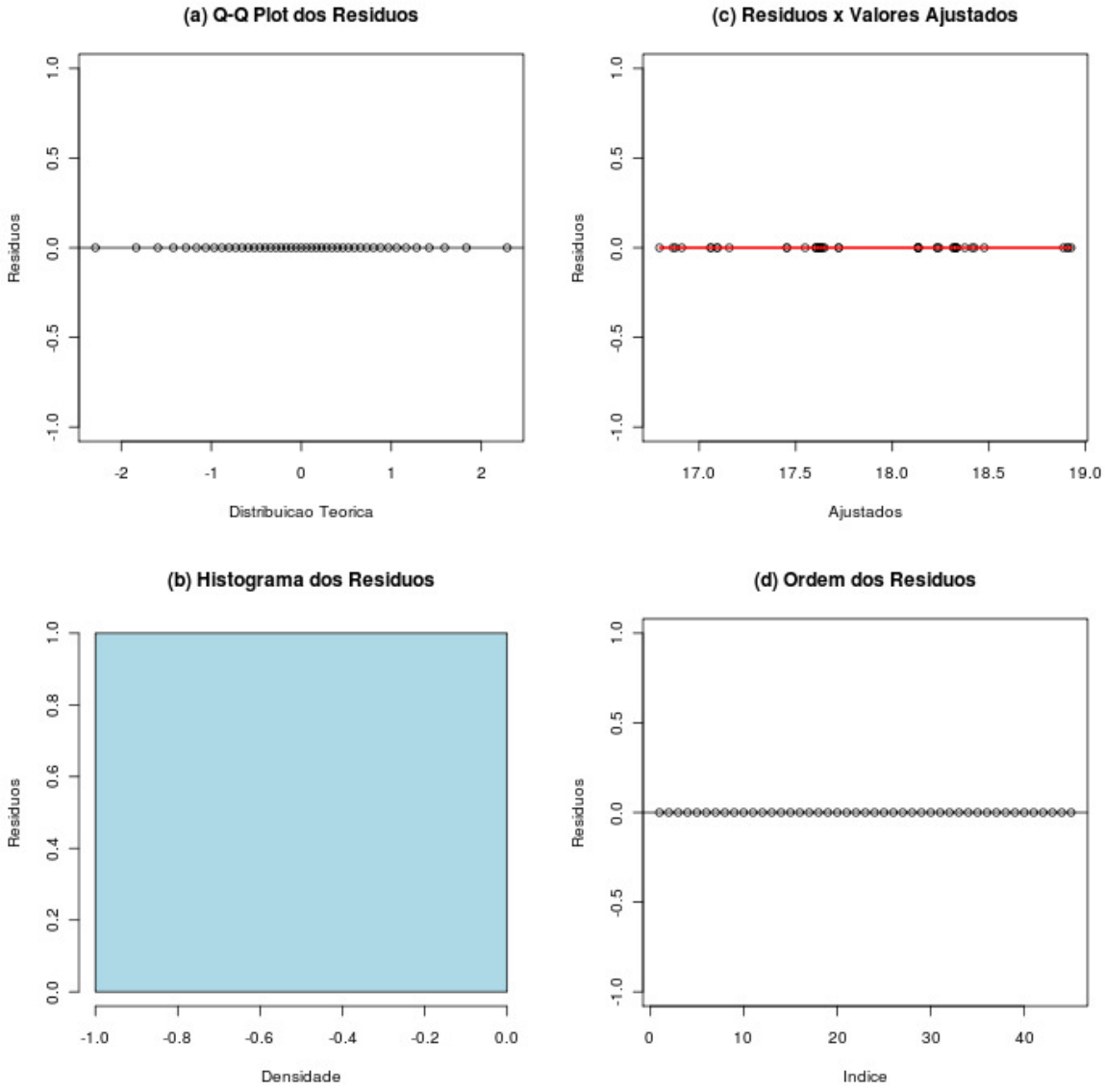


Figura 55 - Gráficos para análise de suposições sobre distribuição dos erros para a conjunto de treinamento com 10% do espaço de projeto da aplicação de multiplicação de matrizes.

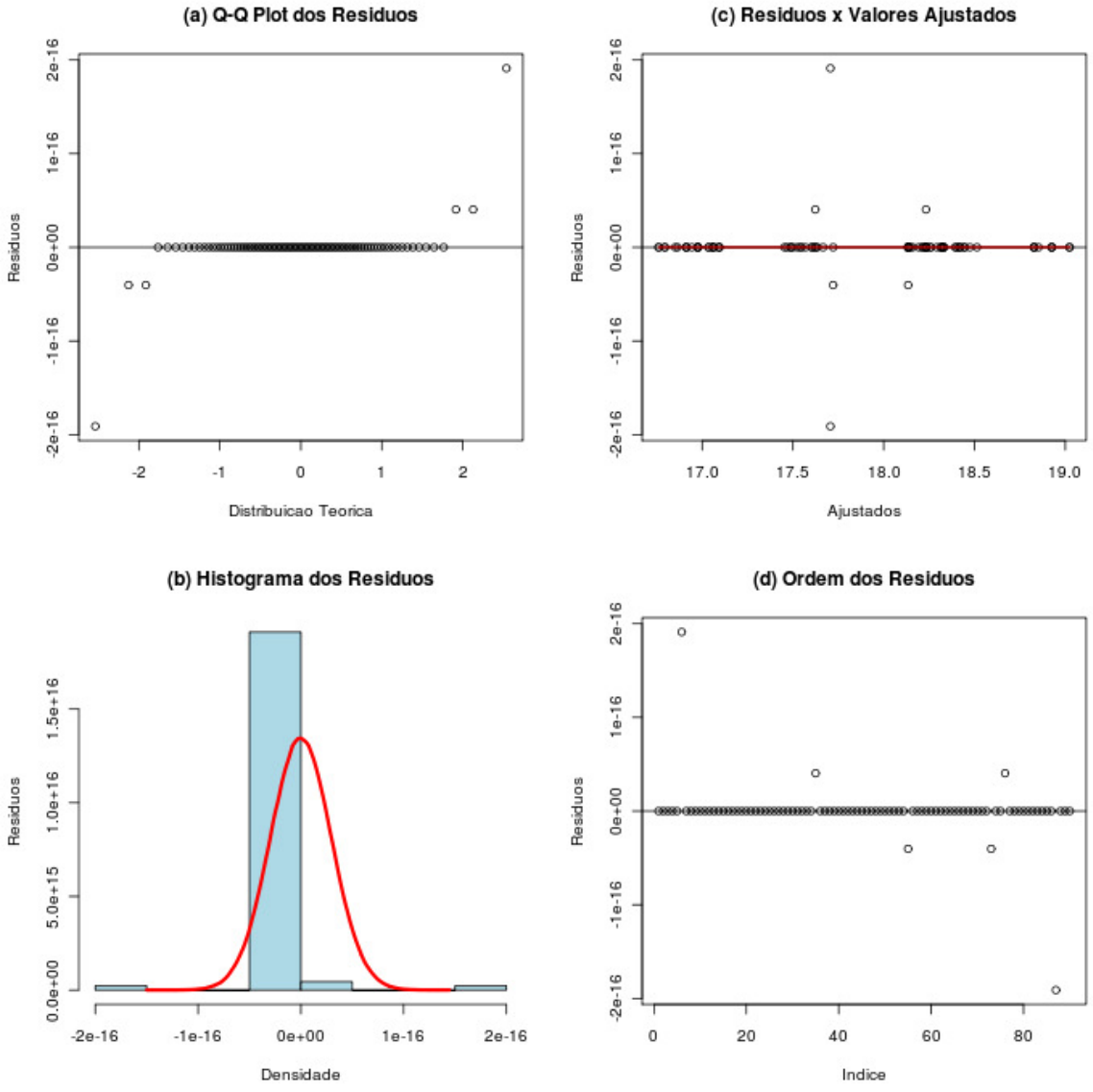


Figura 56 - Gráficos para análise de suposições sobre distribuição dos erros para a conjunto de treinamento com 20% do espaço de projeto da aplicação de multiplicação de matrizes.

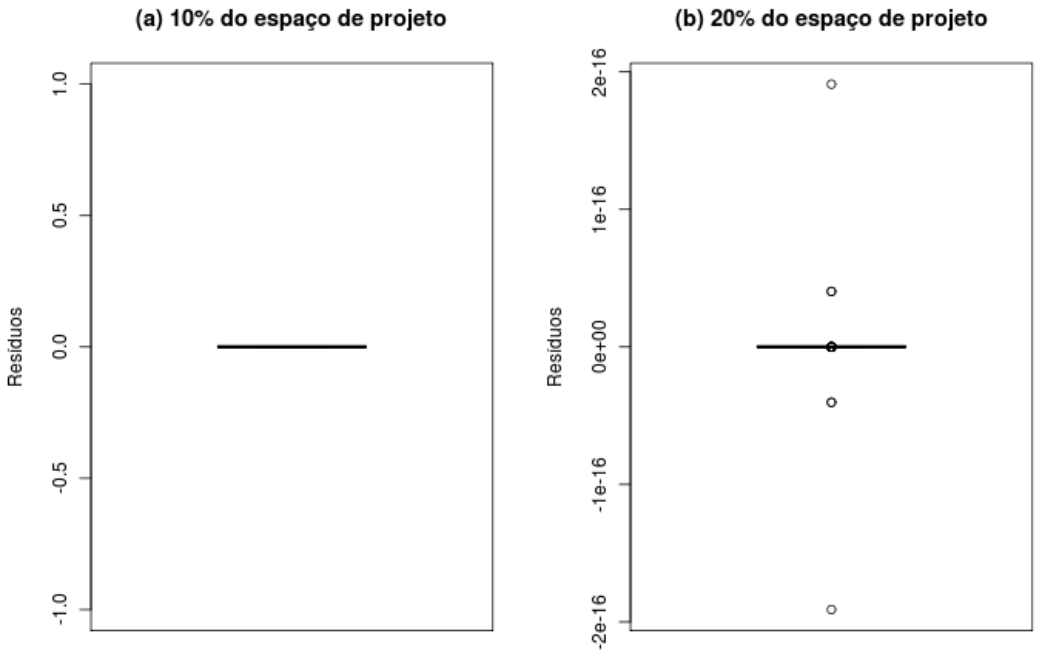


Figura 57 - Gráficos Boxplot para os resíduos dos MLGs formulados a partir dos conjuntos de treinamento com tamanhos de (a) 10% e (b) 20% do espaço de projeto da aplicação de multiplicação de matrizes.

Apêndice D

Conjuntos de Construção selecionados por Técnica de Projeto de Experimentos para Abordagem de Grafos de Comunicação e Programação Genética com Modelos Lineares Generalizados

D.1 Ordenação por Radix

10% do espaço de projeto (7 pontos)

p1	p2	bw	ty	fr	te
1	1	4	1	6	7485216
2	1	1	1	6	22455648
1	1	1	2	5	24950720
2	1	4	2	10	12475360
2	2	2	2	6	14970432
1	2	4	1	10	12475360
2	2	1	1	10	3742608

20% do espaço de projeto (14 pontos)

p1	p2	bw	ty	fr	te
1	2	4	1	5	6237680
1	1	1	1	6	22455648
2	1	4	2	5	6237680
2	2	2	1	6	14970432
2	1	2	2	6	14970432
2	1	1	2	10	49901440
2	1	2	1	10	24950720
1	2	1	1	10	37426080
1	1	1	2	5	24950720
1	2	2	2	10	24950720
1	2	4	2	6	7485216
2	2	4	2	10	12475360
2	2	1	1	5	18713040
1	1	4	1	6	7485216

D.2 Multiplicação de Matrizes

10% do espaço de projeto (48 pontos)

1	p1	p2	p3	p4	bw	ty	fr	te
2	1	3	4	4	4	1	10	38016641
3	1	2	3	2	4	1	10	51514401
4	1	2	4	4	1	1	5	65695316
5	1	4	2	2	2	1	6	57155757
6	1	2	2	2	4	2	5	23229701
7	1	4	3	4	2	2	6	41293989
8	1	4	2	4	4	2	6	24242697
9	1	2	3	4	1	2	10	132848681
10	1	3	2	2	1	1	6	107927553
11	1	4	2	3	1	1	10	175623301
12	1	4	3	2	2	1	10	86183071
13	1	3	2	4	4	1	10	35445711
14	1	2	2	3	4	2	10	40075471
15	1	4	2	3	4	1	10	51514401
16	1	4	4	2	2	2	10	74690721
17	1	4	3	2	4	2	6	30139077
18	1	2	2	3	1	2	5	69616306
19	1	2	2	3	4	1	10	38016641
20	1	2	4	4	1	2	5	69616306
21	1	4	4	4	2	2	10	69616321
22	1	4	2	2	2	2	10	94911921
23	1	2	4	2	4	1	6	33462213
24	1	4	4	3	1	1	10	165982211
25	1	2	4	2	4	2	5	29134971
26	1	4	4	2	1	1	6	102142899
27	1	3	2	3	1	2	6	76204029
28	1	4	4	4	4	2	5	23229701
29	1	4	3	4	4	1	6	23820999
30	1	3	3	2	4	2	6	29177589
31	1	4	2	3	1	2	6	112617501
32	1	2	2	4	4	1	6	25363557
33	1	2	2	2	2	2	10	69616321
34	1	2	3	3	1	2	10	132848681
35	1	4	4	4	4	1	10	38016641
36	1	3	3	2	1	1	10	165982211
37	1	3	4	2	1	2	5	74690706
38	1	4	3	4	1	2	6	76204029
39	1	2	3	3	1	1	6	77557593
40	1	2	2	4	1	1	5	65695316
41	1	4	3	4	1	1	5	60874771
42	1	2	4	2	1	1	6	107927553
43	1	2	4	3	2	2	6	56471349
44	1	3	4	3	4	1	6	33462213
45	1	3	4	2	4	2	10	44373361
46	1	4	4	4	2	1	6	41970777
47	1	2	3	4	2	1	10	65695341
48	1	2	3	4	4	2	6	26598855
49	1	3	2	2	1	2	5	94911906

20% do espaço de projeto (96 pontos)

p1	p2	p3	p4	bw	ty	fr	te	p1	p2	p3	p4	bw	ty	fr	te
1	2	2	4	2	1	10	65695341	1	4	2	2	1	1	5	91003616
1	3	3	4	1	1	6	77557593	1	3	3	4	2	2	10	69616321
1	4	4	3	1	1	10	165982211	1	4	3	4	1	2	6	76204029
1	4	2	4	2	2	6	41293989	1	2	3	2	2	2	6	56471349
1	3	3	4	2	1	10	65695341	1	4	2	4	1	1	5	60874771
1	4	3	2	4	2	6	30139077	1	2	4	2	1	2	5	89862946
1	4	2	2	4	2	5	30116721	1	4	4	4	4	2	6	26598855
1	3	2	2	1	2	5	94911906	1	2	2	4	1	1	6	77557593
1	4	4	3	1	2	5	74690706	1	4	4	4	4	1	6	25363557

1	4	4	3	4	2	5	25378646	1	4	3	2	2	2	6	53426709
1	3	4	4	4	1	10	38016641	1	4	3	4	1	1	10	115365611
1	2	2	2	2	2	6	44323365	1	3	3	4	4	2	6	26598855
1	2	3	2	1	1	6	107927553	1	3	4	2	1	1	6	102142899
1	3	2	3	4	1	6	23820999	1	3	2	2	1	1	5	91003616
1	4	4	2	2	1	10	86183071	1	2	3	4	2	2	6	44323365
1	3	3	3	2	1	6	41970777	1	4	2	3	1	2	10	183439881
1	4	4	2	2	1	10	86183071	1	2	4	4	4	2	10	40075471
1	3	4	2	2	2	5	40537326	1	3	4	3	4	2	6	33685179
1	2	2	2	1	2	10	132848681	1	3	2	4	1	1	10	115365611
1	3	4	2	4	2	6	29177589	1	3	4	2	2	1	6	54263415
1	4	4	2	1	1	5	86183071	1	4	2	2	4	1	5	28949166
1	2	2	4	2	1	5	36039636	1	4	3	3	4	2	10	53849511
1	3	4	4	4	1	6	25363557	1	4	4	3	4	2	6	29177589
1	2	4	4	2	1	10	65695341	1	2	2	3	4	2	5	23229701
1	2	4	3	1	1	10	175623301	1	2	4	3	4	1	10	51514401
1	3	2	3	4	1	10	35445711	1	3	2	3	1	1	6	71772939
1	4	2	2	2	1	6	57155757	1	2	4	4	4	1	5	22200286
1	3	3	3	4	2	10	40075471	1	2	2	2	1	2	6	82262781
1	2	4	4	1	2	6	82262781	1	4	3	4	4	1	6	23820999
1	4	3	3	2	1	10	91003641	1	4	3	4	2	2	6	41293989
1	3	3	4	2	1	5	36039636	1	2	2	2	4	1	6	25363557
1	3	4	3	4	2	10	51886011	1	3	4	4	1	1	5	65695316
1	2	2	3	4	1	10	38016641	1	2	4	2	4	1	5	28949166
1	3	3	3	1	2	6	82262781	1	3	3	4	1	2	5	69616306
1	2	2	3	4	2	10	40075471	1	4	2	2	2	2	5	50647926
1	3	2	2	1	2	10	183439881	1	3	4	2	2	1	10	86183071
1	2	2	3	1	1	5	65695316	1	4	2	2	2	2	10	94911921
1	2	3	4	4	2	5	23229701	1	4	4	2	2	2	5	40537326
1	4	2	4	2	1	6	39078435	1	4	2	4	4	1	10	35445711
1	4	2	3	4	1	10	51514401	1	3	4	4	2	2	6	44323365
1	2	4	4	2	2	10	69616321	1	3	3	4	4	2	10	40075471
1	2	3	2	1	2	6	106558749	1	4	4	4	2	2	10	69616321
1	3	3	2	1	1	10	165982211	1	2	3	4	1	1	5	65695316
1	2	4	2	2	2	10	89862961	1	2	3	3	1	2	10	132848681
1	2	3	2	4	2	6	33685179	1	2	2	3	4	1	6	25363557
1	2	2	2	2	2	10	69616321	1	2	2	4	2	2	10	69616321
1	4	4	3	1	1	6	102142899	1	4	3	2	1	2	10	163193161
1	2	4	3	2	1	6	57155757	1	3	2	2	2	1	10	91003641

Apêndice E

Diagramas de para Análise Residual Gráfica de Modelos Lineares Generalizados Encontrados por Grafos de Comunicação e Programação Genética

E.1 Ordenação por Radix

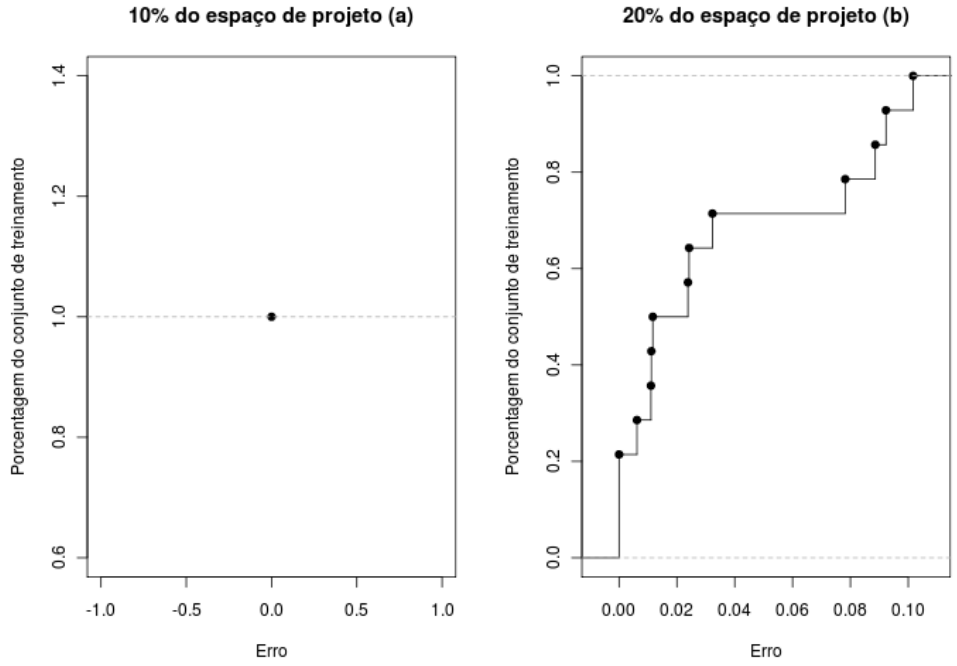


Figura 58 - Gráficos de erros acumulados para MLGs ajustados a partir dos conjuntos de treinamento com (a) 10% e (b) 20% do espaço de projeto da aplicação de ordenação por radix.

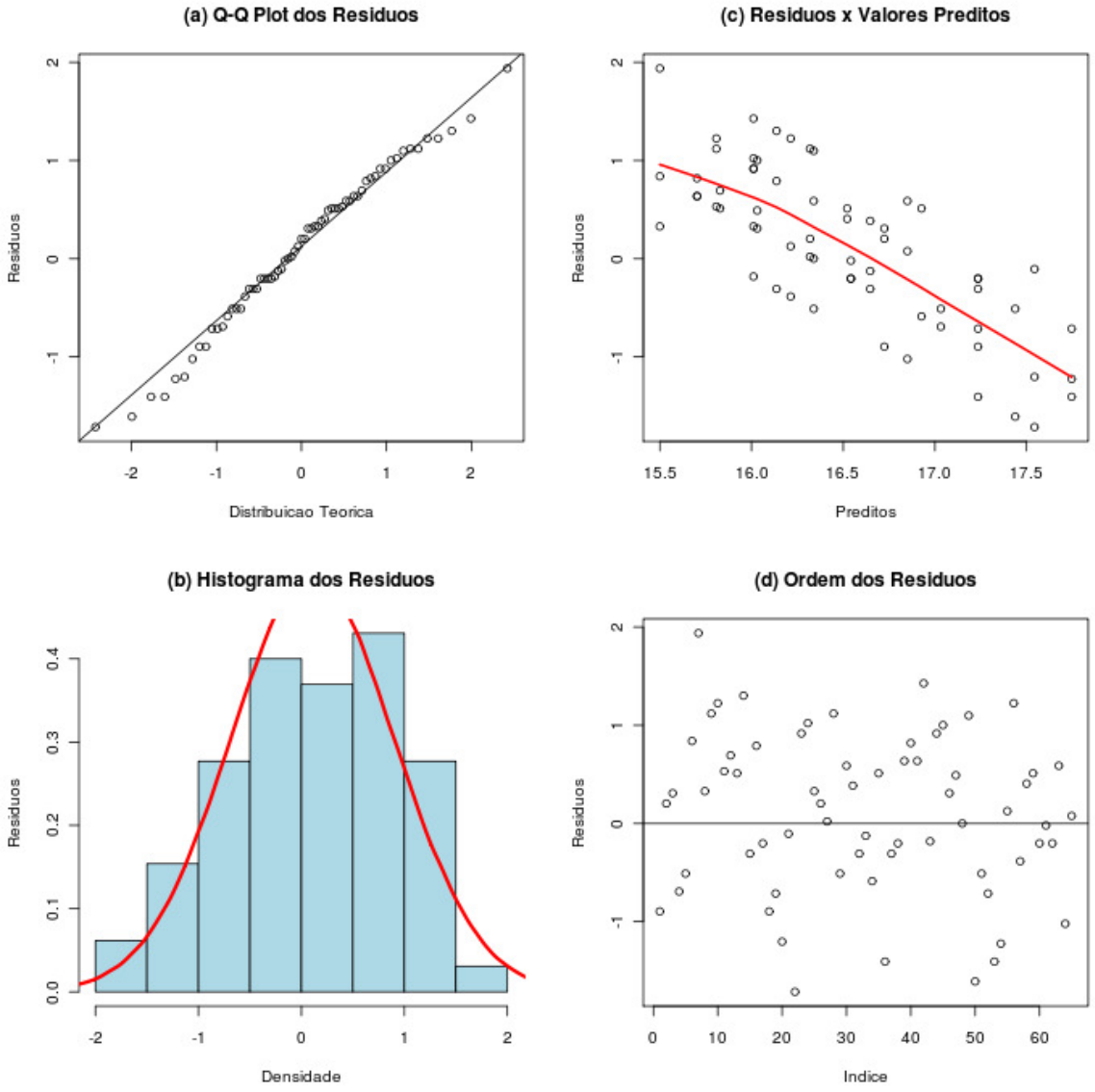


Figura 59 - Gráficos para análise de suposições sobre distribuição dos erros para a conjunto de treinamento com 10% do espaço de projeto da aplicação de ordenação por radix.

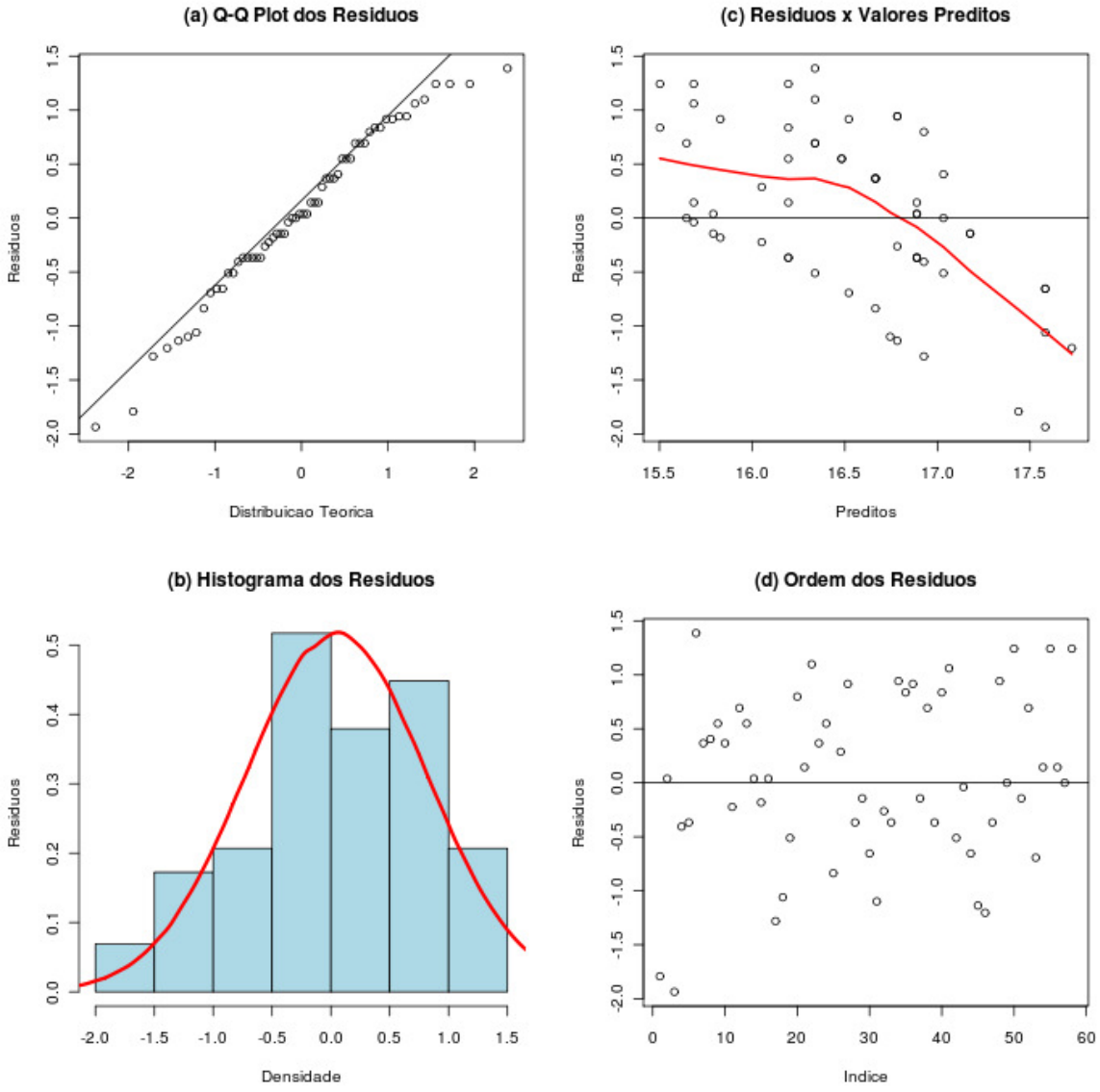


Figura 60 - Gráficos para análise de suposições sobre distribuição dos erros para a conjunto de treinamento com 20% do espaço de projeto da aplicação de ordenação por radix.

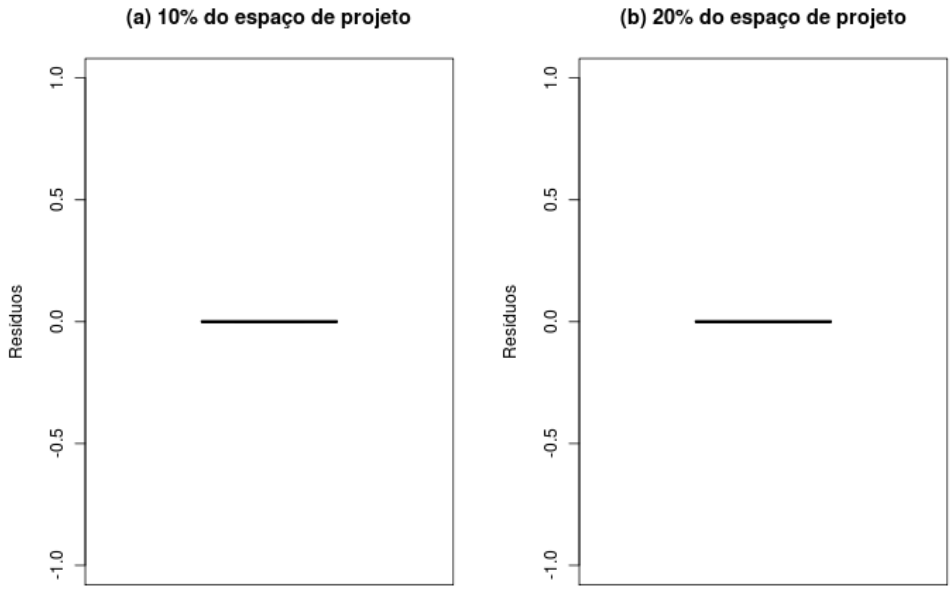


Figura 61 - Gráficos Boxplot para os resíduos dos MLGs formulados a partir dos conjuntos de treinamento com tamanhos de (a) 10% e (b) 20% do espaço de projeto da aplicação de ordenação por radix.

E.2 Multiplicação de Matrizes

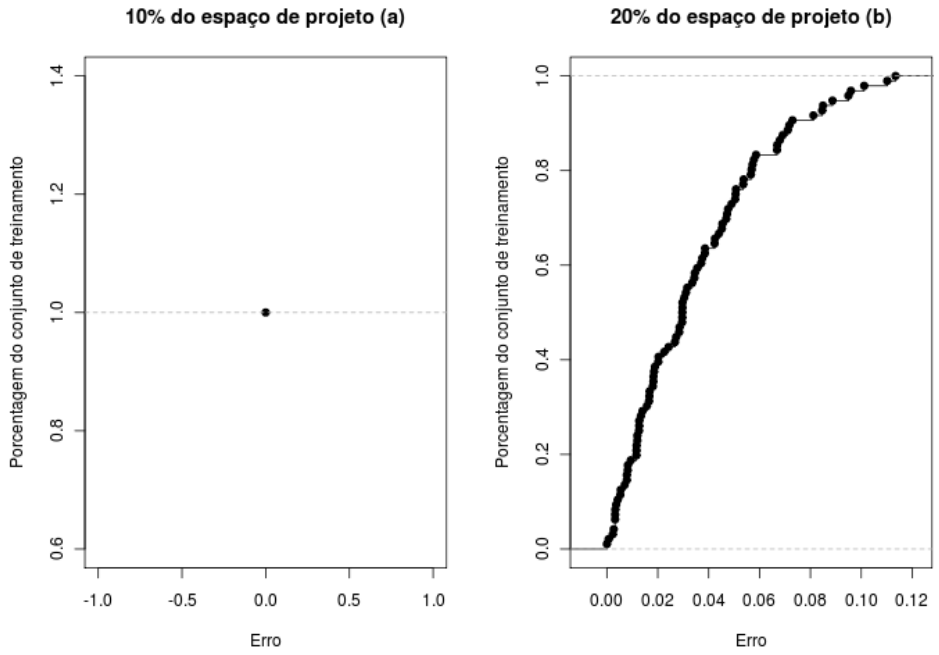


Figura 62 - Gráficos de erros acumulados para MLGs ajustados a partir dos conjuntos de treinamento com (a) 10% e (b) 20% do espaço de projeto da aplicação de multiplicação de matrizes.

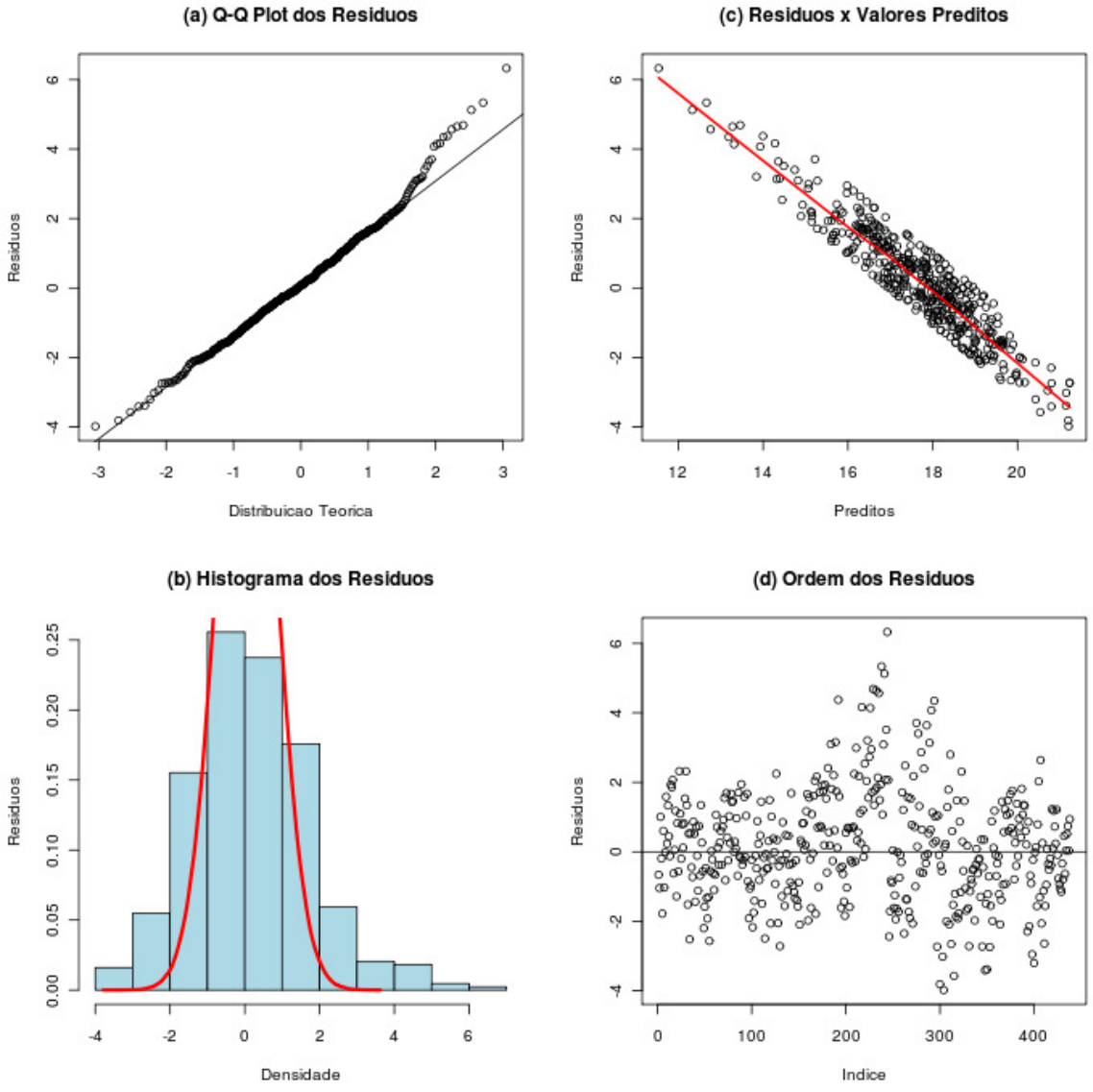


Figura 63 - Gráficos para análise de suposições sobre distribuição dos erros para a conjunto de 'treinamento com 10% do espaço de projeto da aplicação de multiplicação de matrizes.

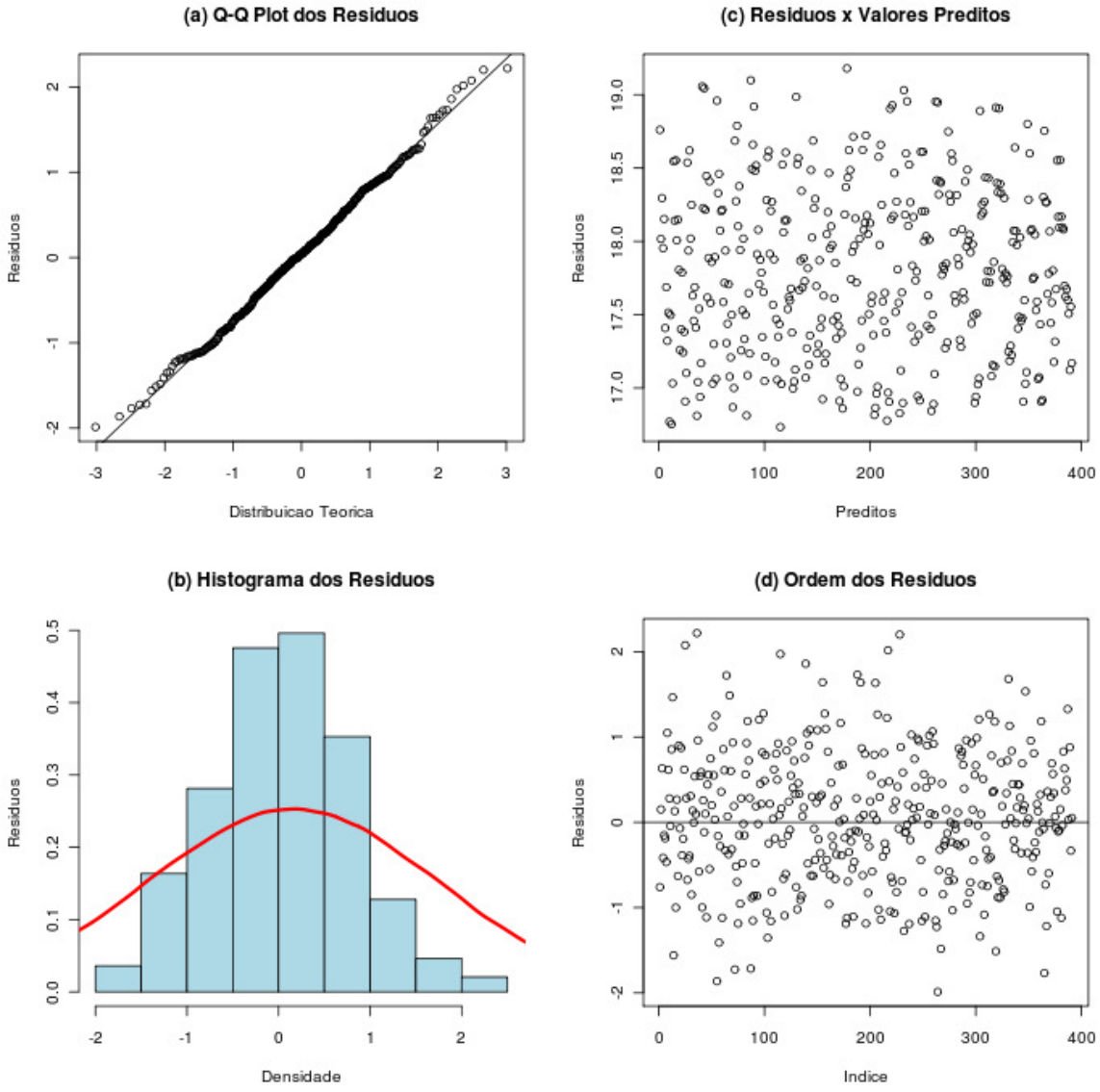


Figura 64 - Gráficos para análise de suposições sobre distribuição dos erros para a conjunto de treinamento com 20% do espaço de projeto da aplicação de multiplicação de matrizes.

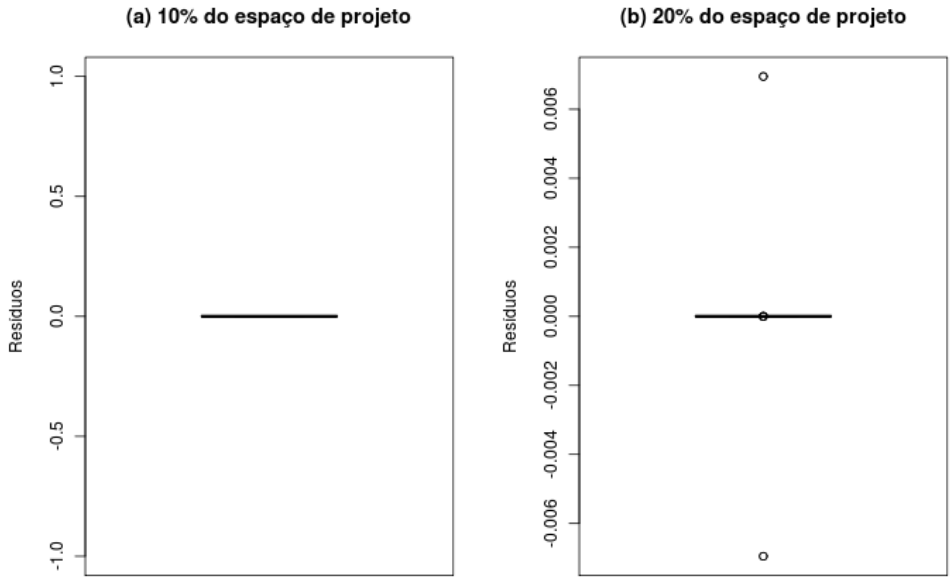


Figura 65 - Gráficos Boxplot para os resíduos dos MLGs formulados a partir dos conjuntos de treinamento com tamanhos de (a) 10% e (b) 20% do espaço de projeto da aplicação de multiplicação de matrizes.