UMA METODOLOGIA DE HW/SW CODESIGN DE PROTOCOLOS DE COMUNICAÇÃO BASEADA NA OTIMIZAÇÃO DE DESEMPENHO POR ALGORITMOS GENÉTICOS

Marcio Nunes de Miranda

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIAS EM ENGENHARIA ELÉTRICA.

Aprovada por:

Prof. Aloysio de Castro Pinto Pedroza, Dr.

Prof. Antônio Carneiro de Mesquita Filho, Dr. d'Etat

Prof. Luci Pirmez, D.Sc.

Prof. Edmundo Albuquerque de Souza e Silva, Ph.D.

Prof. Valmir Carneiro Barbosa, Ph.D.

Prof. José Ferreira de Rezende, Dr.

Prof. Marco Aurélio Cavalcanti Pacheco, Ph.D.

RIO DE JANEIRO, RJ - BRASIL

AGOSTO DE 2002

MIRANDA, MARCIO NUNES DE

Uma Metodologia de HW/SW CoDesign de Protocolos de Comunicação Baseada na Otimização de Desempenho por Algoritmos Genéticos [Rio de Janeiro] 2002

XVII, 140 p. 29,7 cm (COPPE/UFRJ, D.Sc., Engenharia Elétrica, 2002)

Tese - Universidade Federal do Rio de Janeiro, COPPE

1 Projeto de Protocolos

2. HW/SW Codesign

3. Algoritmos Genéticos

4. Avaliação de Desempenho

I. COPPE/UFRJ II. Título (série)

Aos meus pais e irmãos.

Agradecimentos

Aos meus pais Luiz e Maria José, pelo amor e apoio. Aos meus irmãos Marcelo e Marcos pelo incentivo.

Aos Profs. Aloysio de Castro Pinto Pedroza e Antônio Carneiro de Mesquita Filho pela orientação e amizade.

Aos Profs. José Ferreira de Rezende, Luci Pirmez, Julio Salek (in memorium) e Edmundo A. de Souza e Silva pelas sugestões dadas no Exame de Qualificação. Aos Profs. Valmir Carneiro Barbosa, Inês de Castro Dutra e Felipe França pelo apoio nos momentos mais críticos desta jornada.

Aos Profs. Valmir C. Barbosa, Marco Aurélio Pacheco, Luci Pirmez, José F. Rezende e Edmundo A. de Souza e Silva pela presença na banca e contribuição à tese.

Aos meus amigos Alejandro, José Maria, Célia e Octacília pelo apoio e incentivo constante. Aos amigos Paulo André, Kleber, Saulo, Eric, Belém, Pedro, Granato, e Artur pela ajuda na solução de diversos problemas e constante troca de idéias.

Aos Profs. Edmundo A. Souza e Silva e Rosa Maria Meri Leão pela infraestrutura do laboratório LAND/COPPE, sem a qual não teria sido possível a realização das simulações.

Aos Profs. Eduardo, Amit, Richard, Mauros, Otto e Leão; a todos os colegas das turmas de mestrado e doutorado de 1997, 1998, 1999, 2000, 2001 e 2002.

Aos amigos Ana Paula e Flávio, do laboratório LAND/COPPE, pela ajuda na solução de diversos problemas de simulação e suporte técnico em relação à ferramenta Tangram-II.

Ao GTA/PEE/COPPE pela infraestrutura fornecida.

O meu muito obrigado a todos vocês. Aqueles que, por falha da minha memória, não lembrei, por favor me desculpem.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Doutor em Ciências (D.Sc.)

UMA METODOLOGIA DE HW/SW CODESIGN DE PROTOCOLOS DE COMUNICAÇÃO BASEADA NA OTIMIZAÇÃO DE DESEMPENHO POR ALGORITMOS GENÉTICOS

Marcio Nunes de Miranda

AGOSTO/2002

Orientadores: Aloysio de Castro P. Pedroza
Antônio C. de Mesquita Filho

Programa: Engenharia Elétrica

Neste trabalho é apresentada uma metodologia de projeto de protocolos na qual o *hardware* (HW) e o *software* (SW) são desenvolvidos concorrentemente e de forma integrada, técnica conhecida como *HW/SW Codesign*. Essa técnica requer que se faça uma escolha das operações do protocolo que devem ser implementadas em HW e das operações que devem ser implementadas em SW, ou seja, a escolha de uma *partição HW/SW* que atenda a requisitos de projeto especificados pelo projetista. Nos trabalhos existentes na bibliografia, este processo de escolha é manual e muito dependente da experiência do projetista. A metodologia apresentada utiliza um algoritmo genético para otimizar o desempenho do protocolo a ser sintetizado, sob certas restrições de custo. O desempenho é avaliado por uma ferramenta de modelagem e análise de sistemas denominada Tangram-II e o custo do HW é determinado pelas ferramentas Synopsys e/ou Altera. A metodologia auxilia o projetista no processo de seleção da melhor partição HW/SW, através de critérios objetivos estabelecidos a partir dos resultados da otimização de uma função de erro. A análise dos resultados da aplicação da metodologia a protocolos com diferentes características indica que a mesma é mais eficiente quando o protocolo possui um alto grau de processamento nos nós da rede. Nesses casos a implementação em HW tem uma influência significativa no desempenho do protocolo.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Doctor of Science (D.Sc.)

HW/SW CODESIGN OF COMMUNICATION PROTOCOLS BASED ON PERFORMANCE OPTIMIZATION USING GENETIC ALGORITHMS

Marcio Nunes de Miranda

August/2002

Advisors: Aloysio de Castro P. Pedroza
Antônio C. de Mesquita Filho

Department: Electrical Engineering

This work presents a methodology to the design of protocols in which the hardware (HW) and the software (SW) are developed together and integrated. This technique, called HW/SW Codesign, demands a choice of the tasks that must be implemented in hardware and the tasks that must be implemented in software, that is, the choice of a HW/SW partition that meets some project constraints. In other works in bibliography this process is manual and depends on the designer experience. The methodology presented uses a genetic algorithm to optimize the protocol performance, under a restriction of cost of the protocol to be synthetized. The performance is evaluated by a modeling and analysis tool called Tangram-II and the cost of HW by the tools Synopsys and/or Altera. This work helps the designer to select the best HW/SW partition, through objective criterions established from the optimization results of an error function defined by the performance parameters required for the protocol. The analysis of the results, applying the methodology to different kinds of protocols, shows that it is more efficient when the protocol to be designed has a high processing at the nodes of the network. In these cases the implementation in hardware has more influence in the performance of the protocol.

Palavras-chave

Projeto de Protocolos

HW/SW Codesign

Algoritmos Genéticos

Avaliação de Desempenho

Lista de Acrônimos

ARM :	*Active Reliable Multicast*;
ASIC:	*Application Specific Integrated Circuit*;
CCS :	*Calculus of Communicating Systems*;
CMPP:	*Circulant Modulated Poisson Process*;
CDFG:	*Control Data Flow Graph*;
CSP :	*Communicating Sequential Processes*;
CSPL:	*C-based SPN Language*;
CSFM:	*Concurrent Finite State Machine*;
EMPA:	*Extended Markovian Process Algebra*;
ESG :	*Extended Syntax Graph*;
ISO :	*International Organization for Standardization*;
MMPP:	*Markov-Modulated Poisson Process*;
MRM :	*Markov Reward Model*;
OSI :	*Open Systems Interconnection*;
PLD :	*Programmable Logic Device*;
RTL :	*Register Transfer Level*;
RTT :	*Round Trip Time*;
RPC :	*Remote Procedure Call*;
SACK TCP:	*Selective Acknowledgements TCP*;
SC :	*Standard Cells*;
SLIF:	*Specification-Level Intermediate Format*;
SDF :	*Synchronous Data Flow*;
SOR :	*Sucessive Over Relaxation*;
SPN :	*Stochastic Petri Net*;
SRN :	*Stochastic Reward Net*;
TCP :	*Transmission Control Protocol*;
WATM :	*Wireless Asynchronous Transfer Mode*;

Sumário

Resumo **v**

Abstract **vi**

Lista de Acrônimos **ix**

Lista de Figuras **xv**

Lista de Tabelas **xvii**

1 Introdução **1**

1.1 O processo de partição HW/SW 2

1.2 Trabalhos Relacionados 4

1.3 Contribuições 5

1.4 Proposta e Estrutura do Trabalho 5

2 HW/SW Codesign de Protocolos de Comunicação **9**

2.1 Introdução 9

2.2 Síntese de Alto Nível 10

2.2.1 Escalonamento e Alocação 12

2.2.2 Aplicação da Síntese de alto nível ao Projeto de Protocolos . . 13

2.3 HW/SW Codesign 13

2.3.1 Conceitos Básicos 13

2.3.2 Ferramentas de partição HW/SW 17

COSMOS 17

PISH 18

SPECSYN 20

POLIS 21

PTOLEMY 22

LYCOS 24

COSYMA 25

2.3.3 HW/SW Codesign de Protocolos 26

2.3.4 A Partição HW/SW como um Problema de Otimização 28

2.4 Técnicas de Descrição Formal 29

2.4.1 Redes de Petri Estocásticas 30

ARP - Analisador de Redes de Petri 30

2.4.2 Outras TDFs 32

Estelle 32

CCS 32

LOTOS 34

2.5 Ferramentas de Análise de Desempenho 35

2.5.1 SHARPE 36

2.5.2 MARCA 37

2.5.3 SPNP 39

2.5.4 SMAQ 40

2.5.5 TWOTOWERS 41

2.6 Tangram-II 42

2.7 Comentários 43

3 Partição HW/SW de Protocolos **46**

3.1 Introdução 46

3.2 Avaliação de Desempenho de Protocolos 47

3.3 Metodologia de Auxílio à Partição HW/SW 49

3.3.1 Descrição Inicial e Verificação Formal 51

3.3.2 Construção do Modelo 52

3.3.3 Construção da Função Objetivo 53

3.3.4 Otimização da Função Objetivo 54

3.3.5 Critério de partição 55

3.3.6 Implementação em Hardware 56

3.4 Exemplo - Protocolo bit alternante 57

3.4.1 Metodologia Passo a Passo 57

3.4.2 Avaliação da Partição 62

3.4.3 Análise do Desempenho da Partição Escolhida 63

3.5 Comentários 64

4 Técnicas de Projeto **65**

4.1 Introdução 65

4.2 Sintaxe da ferramenta Tangram-II 65

4.2.1 Definições 66

4.2.2 Construção de Modelos 68

4.3 Algoritmos Genéticos 70

4.3.1 Conceitos Fundamentais 70

4.3.2 Operações básicas dos AGs 71

4.3.3 Parâmetros do Algoritmo Genético 74

4.3.4 Outros Tipos de AGs 75

4.3.5 Algoritmo utilizado na Metodologia Proposta 77

4.4 Comentários 80

5 Resultados **81**

5.1 Introdução 81

5.2 TCP "Congestion Avoidance" 82

5.2.1 TCP "Congestion Avoidance" com perda determinística 82

5.2.2 TCP "Congestion Avoidance" com perda aleatória 85

Avaliação da Partição 90

Análise do Desempenho da Partição Escolhida 91

5.3 Protocolo de Controle de *Handoff* para uma rede ATM sem fio 93

5.3.1 Avaliação da Partição 97

5.3.2 Análise do Desempenho da Partição Escolhida 98

5.4 Protocolo ARM 99

5.4.1 Avaliação da Partição 105

5.4.2 Análise do Desempenho da Partição Escolhida 106

5.5 Comentários 107

6 Conclusões **109**

Referências Bibliográficas **114**

A Método do Gradiente para a Determinação das Faixas dos Parâmetros **124**

B Modelo do Tangram-II para o Protocolo ARM **128**

Lista de Figuras

2.1 Diagrama Y 11

2.2 Abordagem hardware/software codesign. 14

2.3 Partição HW/SW e Refinamento 15

2.4 Codesign no contexto do projeto do sistema 16

2.5 A ferramenta COSMOS. 19

2.6 O sistema de *HW/SW codesign* PISH. 20

2.7 O ambiente SpecSyn. 22

2.8 O sistema POLIS. 23

2.9 Codesign utilizando o Ptolemy. 24

2.10 O sistema de *HW/SW codesign* LYCOS. 25

2.11 O sistema de partição COSYMA. 26

2.12 Exemplo de uma especificação em Estelle. 33

2.13 A ferramenta TwoTowers construída sobre o CWB-NC e MarCA . . . 42

2.14 O ambiente Tangram-II 43

3.1 Metodologia de projeto. 51

3.2 Rede de Petri do protocolo bit alternante. 59

3.3 Curva vazão versus carga do protocolo bit alternante. 60

3.4 Comparação entre as vazões para as diferentes implementações. . . . 63

4.1 Modelo de fila M/M/1/k 69

4.2 Diagrama de blocos de um algoritmo genético simples 72

4.3 Amostragem universal estocástica 73

4.4 Cruzamento de dois *indivíduos* num AG 74

4.5 Diagrama de blocos do algoritmo da metodologia 78

5.1 cwnd versus RTT com perda periódica. 83

5.2 Rede de Petri do mecanismo de controle de congestionamento "TCP Congestion Avoidance". 85

5.3 Curva Janela vs *perda* para o modelo matemático dado pela eq. 5.3. . 86

5.4 Evolução da Janela de congestionamento com probabilidade de perda com distribuição exponencial. 89

5.5 Rede de Petri condição-ação do Protocolo de controle de *Handoff*. . . 94

5.6 Probabilidade de Perda em função da taxa de transmissão. 95

5.7 Comparação entre as diversas implementações. 98

5.8 O protocolo ARM. 101

5.9 Implementação em SW versus implementação com a partição HW/SW.106

5.10 Implementação em HW versus implementação com a partição HW/SW.107

Lista de Tabelas

3.1 Taxas originais da Rede de Petri. 58

3.2 Taxas da Rede de Petri do protocolo bit alternante obtidas pelo AG. 59

3.3 Valores obtidos para o produto $\lambda_i * \pi_j$. 60

3.4 Medidas de atraso e área das transições do protocolo. 62

5.1 Taxas do TCP "Congestion Avoidance" obtidas pelo AG. 87

5.2 Valores obtidos para o produto $\lambda_i * \pi_j$. 87

5.3 Medidas de atraso e área. 91

5.4 Erros obtidos para cada implementação. 92

5.5 Taxas da rede de Petri do protocolo de *Handoff* obtidas pelo AG. . . 95

5.6 Valores obtidos para o produto $\lambda_i * \pi_j$. 96

5.7 Medidas de área e atraso. 97

5.8 Taxas da rede de Petri do protocolo ARM obtidas pelo AG. 104

5.9 Valores obtidos aplicando-se o critério de partição. 104

5.10 Medidas de área e atraso. 105

Capítulo 1

Introdução

A complexidade da nova geração de sistemas concorrentes que estão sendo atualmente projetados tornou muito importante o desenvolvimento de sofisticadas ferramentas e metodologias para a modelagem, o projeto e a análise de desempenho de tais sistemas. O comportamento desses sistemas geralmente é modelado por uma máquina de estados e, atualmente, devido ao recente desenvolvimento de novas tecnologias de *hardware*, modelos com milhares de estados podem ser resolvidos e uma classe bem maior de problemas pode ser solucionada com técnicas numéricas.

Os sistemas de comunicação, em particular, foram bastante beneficiados com essas novas tecnologias, pois protocolos de comunicação anteriormente implementados totalmente em *software* podem ser total ou parcialmente implementados em *hardware*, a um custo acessível, aumentando a velocidade de processamento das mensagens e conseguindo atender às exigências das redes de alta velocidade e com alto tráfego.

Essas redes exigem protocolos de comunicação específicos. O uso da otimização em *software* na elaboração de protocolos para as atuais redes nem sempre permite operações a altas taxas de velocidade. Um fator importante para o bom desempenho dos protocolos de alta velocidade é a utilização integrada de *hardware* e *software* durante sua implementação. Algumas operações do sistema, normalmente as menos

críticas em termos de velocidade, podem ser implementadas em *software*, enquanto as que necessitam de uma maior velocidade devem ser implementadas em *hardware*.

Devido à crescente demanda por protocolos com altas vazões, as soluções que utilizam *hardware* são cada vez mais investigadas. Num roteador, por exemplo, o encaminhamento de pacotes totalmente implementado em SCs (*Standard Cells*) pode fornecer uma vazão até dez vezes maior do que a implementação inteiramente baseada em microprocessadores de propósito geral [1].

Se no desenvolvimento do projeto as partes de *hardware* e *software* são integradas numa única especificação e desenvolvidas conjuntamente, o processo é conhecido como *HW/SW codesign* [2]. O princípio do *codesign* é a realização de um projeto cooperativo baseado em ambientes específicos, *hardware* e *software*, onde pode-se verificar e simular todo o sistema em cada etapa do projeto. Esse processo permite a detecção de erros nas etapas iniciais do projeto, reduzindo o seu tempo e o seu custo. No início, o projetista não tem conhecimento das partes que serão implementadas em *hardware* e das que serão implementadas em *software*, portanto a especificação inicial é construída utilizando-se diagramas de blocos. Após a formulação dos requisitos do sistema, a especificação deve ser refinada e validada.

1.1 O processo de partição HW/SW

Apesar dos tempos de execução em um *hardware* específico serem, normalmente, muito menores que os tempos de execução em uma implementação em *software*, o custo da primeira solução é maior. Portanto, deve-se estabelecer um compromisso entre o desempenho desejado e o custo de implementação para se decidir que partes deverão ser implementadas em *hardware* e que partes deverão ser implementadas em *software*. Essa etapa é denominada de **processo de partição HW/SW** e sua decisão tem impacto direto no desempenho do sistema de comunicação do qual o protocolo faz parte.

À medida que se avança no detalhamento da especificação, torna-se necessária a aplicação de técnicas que auxiliem o projetista a tomar decisões relativas à partição

HW/SW das operações do protocolo. Essas operações representam o comportamento do protocolo e podem ser descritas por um diagrama de estados, onde cada transição de estado representa um conjunto de cláusulas e expressões a serem avaliadas.

Deve-se considerar diversos aspectos na partição HW/SW do projeto de um protocolo de comunicação. Inicialmente deve-se definir a *granularidade* do sistema, ou seja, as menores partes indivisíveis usadas durante a partição, tais como: processos, subrotinas, blocos de instruções, instruções ou operações aritméticas. Quanto maior a granularidade menor o número de objetos e, consequentemente, menor o número de soluções possíveis, o que agiliza o processo mas diminui o número de partições HW/SW a serem avaliadas. Também devem ser definidos os parâmetros a serem usados para a realização da partição. Alguns parâmetros comumente utilizados incluem custo financeiro, desempenho, taxas de comunicação, área de silício do circuito integrado, confiabilidade, tamanho do programa, quantidade de dados e tamanho da memória. Este trabalho concentra-se nos parâmetros de desempenho e custo do *hardware*. O custo baseia-se na área de silício do circuito sintetizado e na sua velocidade.

Os diversos parâmetros utilizados devem ser combinados numa função de custo única, denominada *função objetivo*. Essa função fornece uma medida da qualidade da partição em questão e através de sua avaliação pode-se comparar duas partições e selecionar aquela que satisfaz melhor as especificações.

Atualmente existem diversas técnicas que procuram solucionar o problema da partição HW/SW: algoritmos de agrupamento (*clustering*), algoritmos de melhoria incremental (*iterative-improvement*), algoritmos customizados (*custom algorithms*) e algoritmos genéticos. Alguns são mais rápidos, como os de *clustering*, enquanto que outros são mais lentos porém encontram soluções melhores, como os algoritmos genéticos. Os algoritmos genéticos são particularmente úteis em problemas de otimização que envolvem um grande número de variáveis e, consequentemente, espaços de solução de dimensões elevadas, com múltiplos mínimos locais. Ao contrário dos métodos de otimização tradicionais, os AGs não utilizam derivadas na busca de

uma solução do problema e, portanto, se aplicam bem a problemas que possuem funções não-diferenciáveis.

1.2 Trabalhos Relacionados

Embora existam alguns esforços para se automatizar o processo de partição HW/SW, o projetista ainda dispõe de poucos recursos para auxiliá-lo no refinamento da especificação, no desenvolvimento do projeto e no processo de partição HW/SW de um dado protocolo.

Diversos grupos de pesquisa vêm desenvolvendo ambientes de projeto baseados na metodologia de *codesign*. Dentre eles, pode-se citar: COSMOS [3], SpecSyn [4], Ptolemy [5][6][7], LYCOS [8], Chinook [9] e PISH [10][11][12]. Esses ambientes de *codesign* diferem na linguagem de especificação, no método de particionamento, na arquitetura alvo e nos métodos de validação utilizados.

Fischer [13] ressalta a importância do uso combinado do *hardware* e do *software* para se alcançar um alto desempenho de sistemas distribuídos, como sistemas multimídias. Mais especificamente, é apresentado um ambiente de desenvolvimento para o suporte das etapas de projeto e implementação. O sistema é inicialmente especificado na linguagem Estelle e, posteriormente, dividido em várias especificações menores, sendo as partes de *software* implementadas na linguagem C e as partes de *hardware* descritas em VHDL, para posterior análise e desenvolvimento. Ao final, o projetista decide que partes serão implementadas em *hardware* e que partes serão implementadas em *software*.

Em Hidalgo [14], a partição HW/SW é realizada usando um algoritmo genético que utiliza, para avaliação da função objetivo, medidas de custo e desempenho. Nesse trabalho, as medidas de custo e desempenho são obtidas através de estimativas e medidas realizadas por Scott [15] em trabalho anterior. Foi construída uma tabela para os custos de *hardware* (assumiu-se que o custo do *software* é zero) e para os desempenhos de *hardware* e *software*. Cada bloco da especificação tem um determinado custo de *hardware* e um determinado desempenho. A partir dessas re-

strições a função objetivo é calculada e o algoritmo genético é utilizado na obtenção da melhor partição HW/SW.

1.3 Contribuições

A metodologia de projeto desenvolvida neste trabalho é uma evolução de técnicas de HW/SW Codesign anteriormente propostas e é aplicada especificamente ao projeto de protocolos de comunicação. A busca por essa evolução é motivada pela inexistência de técnicas que auxiliam objetivamente o processo de partição HW/SW, limitando a exploração do espaço de soluções. Nesse caso, é útil fornecer ao projetista subsídios que o auxiliem na escolha da melhor partição HW/SW. Essa metodologia foi inicialmente aplicada a um protocolo simples, o *bit alternante*, e posteriormente aplicada a protocolos mais complexos: ao *TCP Congestion Avoidance* [16][17], a um protocolo de controle de *Handoff* para uma rede ATM sem fio [18] e a um protocolo *multicast* confiável que utiliza roteadores ativos (ARM) [19].

O objetivo deste trabalho é apresentar uma nova metodologia que auxilie o processo de partição HW/SW de um protocolo de comunicação, estabelecendo critérios objetivos para que a mesma seja realizada sem necessidade do projetista utilizar técnicas manuais. Essa proposta utiliza de forma conjunta uma ferramenta de análise de desempenho, ferramentas de síntese de HW em SCs (*standard cells*) e PLD (*Programmable Logic Devices*) e um algoritmo genético.

1.4 Proposta e Estrutura do Trabalho

Os modelos de desempenho são, muitas vezes, modelos estocásticos, cujo comportamento é de natureza probabilística, ou seja, o tempo de serviço em um recurso (ou o tempo entre chegadas de mensagens) é aleatório. Muitos dos métodos e ferramentas utilizados na análise do desempenho de um sistema de comunicação são baseados em diagramas de estados e impõe-se a hipótese de que o tempo de permanência do sistema em cada estado até uma transição é exponencialmente distribuído. Portan-

to, é interessante que se possa aproveitar essa característica no projeto de protocolos de comunicação, pois facilita a modelagem do protocolo. Além disso, utilizando-se ferramentas apropriadas, é possível obter medidas que orientam o projetista a tomar decisões consistentes durante a implementação do protocolo.

Na metodologia desenvolvida neste trabalho, os protocolos são especificados, inicialmente, por um modelo de máquina de estados, onde cada transição de estado representa uma operação ou um conjunto de operações a serem avaliadas. Esta especificação serve como base para a construção do modelo da ferramenta de análise de desempenho Tangram-II [20][21][22][23] e para a descrição na linguagem VHDL [24][25] de maneira que a síntese dos circuitos de *hardware* possa ser realizada.

A ferramenta Tangram-II auxilia o projetista no cálculo de parâmetros de desempenho tais como retardo, vazão, probabilidade de perda de mensagens, entre outros. Um algoritmo genético é utilizado para otimizar uma função objetivo desses parâmetros. As ferramentas Synopsys [26] e Altera [27][28][29][30] fornecem informações de custo do HW a partir dos circuitos integrados sintetizados.

Inicialmente é especificado o desempenho desejado para o protocolo, em termos da vazão, banda passante, retardo ou qualquer outra medida de interesse que possa ser obtida a partir da ferramenta de análise de desempenho aqui utilizada. O protocolo é especificado com base nas redes de Petri estocásticas [31][32] para que possa ser realizada uma verificação e validação da especificação através da ferramenta ARP (analisador de redes de Petri), que verifica se a rede não possui *deadlocks* e/ou *livelocks*, além de fornecer outras características importantes, como, por exemplo, se a rede é *viva* e *limitada* [33]. Uma vez verificada a especificação, é construído o modelo apropriado à análise de desempenho do protocolo, desenvolvido no ambiente e na sintaxe da ferramenta de análise de desempenho Tangram-II. O modelo desenvolvido é paramétrico, sendo os valores dos parâmetros fornecidos por um método de otimização baseado em algoritmos genéticos (AGs) [34][35][36]. As medidas de desempenho desejadas são utilizadas na composição da função objetivo que será otimizada. Isso permite encontrar as taxas de transição da rede de Petri que melhor aproximam o desempenho e custo especificados. Uma vez determinadas as taxas de

transição da rede de Petri, são estabelecidos critérios objetivos para determinar a partição HW/SW.

Este trabalho está estruturado da seguinte forma: O capítulo 2 apresenta a teoria básica e principais características da síntese de alto nível e do processo de HW/SW *codesign*. São apresentadas as principais operações envolvidas na síntese, escalonamento e alocação, e como ela se aplica ao projeto de protocolos. Também são apresentadas algumas ferramentas de partição HW/SW existentes, a forma como o princípio de *codesign* se aplica ao projeto de protocolos e como o processo de HW/SW *codesign* pode ser abordado como um problema de otimização. Em seguida são apresentadas algumas ferramentas de verificação formal de protocolos e de análise de desempenho de sistemas.

O capítulo 3 mostra como medidas de interesse normalmente utilizadas para a avaliação de desempenho de sistemas podem ser utilizadas no projeto de protocolos para auxiliar na definição da melhor partição HW/SW. Em seguida, é apresentada e desenvolvida uma metodologia para a determinação da melhor partição HW/SW, através do estabelecimento de critérios de partição objetivos baseados nos resultados fornecidos pelas ferramentas de análise de desempenho e de síntese de *hardware* e pela otimização realizada pelo algoritmo genético. É descrita a forma como se constrói a função objetivo utilizada no algoritmo genético e o critério de partição utilizado, que gera uma solução de partição HW/SW cujo custo é analisado através dos resultados fornecidos pela síntese de *hardware* realizada por Lima [37], utilizando as ferramentas Synopsys e Altera.

O capítulo 4 apresenta uma descrição das principais ferramentas de projeto utilizadas na metodologia proposta. É apresentada uma descrição da sintaxe da ferramenta de análise de desempenho de sistemas Tangram-II, suas características e aplicações. Além disso, introduz os fundamentos básicos dos algoritmos genéticos, suas vantagens e aplicações, apresenta alguns tipos de algoritmos genéticos existentes na bibliografia e descreve o algoritmo utilizado na metodologia. A descrição detalhada das ferramentas de síntese de *hardware* está fora do escopo deste trabalho e pode ser encontrada em Lima [37].

No capítulo 5 são apresentados e analisados os resultados obtidos com a aplicação da metodologia a diversos tipos de protocolos, cada um com uma característica diferente. O critério de partição é aplicado e a partição HW/SW obtida é avaliada através dos resultados fornecidos por Lima [37]. A partir da partição sugerida é realizada uma análise do desempenho da mesma em relação às outras implementações possíveis.

No capítulo 6 são apresentadas as conclusões do trabalho desenvolvido e sugestões para trabalhos futuros.

Capítulo 2

HW/SW Codesign de Protocolos de Comunicação

2.1 Introdução

Usualmente protocolos de comunicação são implementados inteiramente em *software*. No entanto, para que os requisitos de velocidade das redes atuais sejam atendidos, pode ser necessário implementar parte das especificações de um protocolo em *hardware*, o que implica em promover a partição HW/SW de uma determinada especificação [38]. Dessa forma, é necessário realizar em alguma fase do projeto uma partição funcional da especificação para a definição dos componentes do sistema: processos, subrotinas, variáveis e canais de comunicação são alocados a um ou mais processadores padrão, *hardwares* específicos (ASICs , por exemplo), memórias e barramentos. Pode-se reduzir o tempo de execução de um protocolo totalmente implementado em *software* utilizando-se componentes de *hardware* específicos para determinadas funções. Equivalentemente, pode-se reduzir o custo de um sistema totalmente implementado em *hardware*, movendo-se algumas de suas funções, implementadas em *hardware* específico, para *software*, ou seja, para processadores padrão [39].

Assim, durante o projeto de um protocolo de comunicação, o projetista define, a partir de um modelo inicial, as tarefas do protocolo que serão implementadas em *hardware* e as que serão implementadas em *software*, procurando uma solução de compromisso entre o seu desempenho e o seu custo. Medidas típicas consideradas na síntese do protocolo incluem: os tempos de execução em *hardware* e *software*, o tempo de comunicação entre os módulos, tamanho do código objeto do *software* (incluindo as rotinas de comunicação) e área do circuito integrado.

2.2 Síntese de Alto Nível

A síntese de alto nível pode ser definida como um processo de mapeamento de uma descrição *comportamental* de um sistema em uma descrição RTL estrutural, análogo ao processo de compilação de linguagens como C ou Pascal em linguagem *assembly*. Para ilustrar esse processo utiliza-se o diagrama em **Y** da figura 2.1 [40]. Os eixos do diagrama representam três diferentes domínios (*comportamental*, estrutural e físico), enquanto que ao longo dos eixos existem diferentes níveis de abstração dentro cada domínio. O nível de abstração é tanto maior quanto mais se afasta do centro do diagrama. Esse diagrama pode ser estendido desenhando-se círculos concêntricos. Cada círculo cruza o eixo **Y** num nível de representação particular dentro de cada domínio.

No domínio *comportamental* o sistema é tratado como uma "caixa preta" com um conjunto de entradas, um conjunto de saídas e um conjunto de funções que descrevem o comportamento do sistema ao longo do tempo. No domínio estrutural a funcionalidade prevista no domínio *comportamental* é descrita por um conjunto de componentes e conexões sob restrições como custo, área e velocidade. Cada componente do domínio estrutural é definido por uma representação que algumas vezes pode ser utilizada como descrição funcional, nos níveis de abstração mais baixos do domínio *comportamental*. Dependendo do nível de abstração, pode-se ter, no domínio estrutural: transistores, barramentos, flip-flops, ALUs, RAMs, etc. O

domínio físico representa *layout* de células, módulos, circuitos impressos.

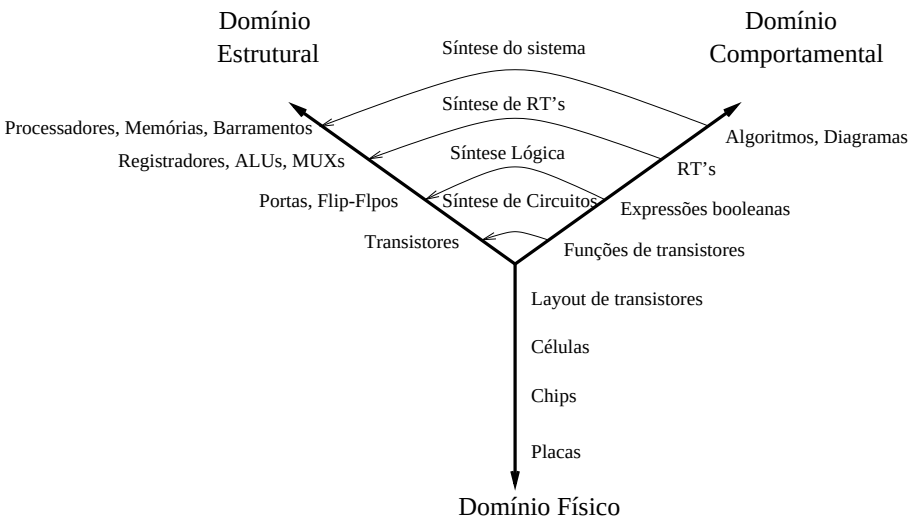

Figura 2.1: Diagrama Y

Em um mesmo domínio, a mudança de um nível de abstração mais alto para um mais baixo aumenta o grau de detalhamento do projeto, mantendo a consistência da especificação, isto é, as restrições impostas nos níveis superiores de descrição são mantidas nos níveis mais baixos.

Na síntese de alto nível utiliza-se formalismos baseados em diversas áreas, desde o paradigma de linguagem de programação, para a descrição *comportamental*, até modelos de *layout* para estimativa de custo e desempenho. Frequentemente utiliza-se como método de descrição formal para a síntese de alto nível as máquinas de estado finitas estendidas. Essa descrição é especialmente útil no projeto de protocolos de comunicação, como será visto mais adiante.

A partição HW/SW é apenas parte do problema de síntese de um sistema de comunicação. A partição produz um conjunto de subsistemas e primitivas de comunicação, onde cada subsistema se comunica com os outros subsistemas [41]. O ponto de partida da síntese de sistemas de comunicação é um modelo composto por um conjunto de processadores dentro de uma rede de comunicação composta por canais. O objetivo é transformar um sistema composto de processadores que se comunicam através de canais e primitivas de alto nível em um conjunto de processadores se comunicando através de barramentos e dividindo o controle da comunicação.

2.2.1 Escalonamento e Alocação

O modelo *comportamental* especifica as operações a serem executadas pelo *hardware*. Normalmente essas operações são representadas internamente através de um grafo com fluxos de controle e de dados (CDFG), que captura todas as dependências de controle e dados do modelo *comportamental*. Os algoritmos de escalonamento particionam esse grafo em subgrafos, compostos por um subconjunto de operações, onde cada subgrafo é executado num passo de controle. Cada passo de controle pode ser comparado a um estado, num diagrama de uma máquina de estado finita.

Cada operação a ser executada dentro de um passo requer uma unidade funcional. Portanto, o número total de unidades funcionais necessárias para a execução de um passo de controle corresponde ao número total de operações a serem executadas naquele passo. Os algoritmos de escalonamento otimizam o número de passos de controle e o número de operações a serem executadas em cada passo, além de definir qual o próximo conjunto de operações será executado ao final de um determinado passo. Essa otimização tem impacto direto no compromisso entre o custo do projeto e o seu desempenho.

Os algoritmos de alocação consistem na determinação do número e do tipo de unidades funcionais a serem utilizados no projeto, além de efetuar um mapeamento (*binding*) das variáveis e operações escalonadas no CDFG em unidades de armazenamento e interconexão, de maneira a assegurar que o sistema se comportará corretamente com o conjunto de componentes selecionados.

Os algoritmos de escalonamento e alocação estão relacionados. É difícil avaliar a qualidade de um determinado algoritmo de escalonamento sem considerar o algoritmo de alocação que está sendo utilizado. Dois escalonamentos diferentes com o mesmo número de unidades funcionais e passos de controle podem resultar em projetos com desempenhos completamente diferentes após o algoritmo de alocação ser executado.

2.2.2 Aplicação da Síntese de alto nível ao Projeto de Protocolos

Um protocolo de comunicação pode ser caracterizado por uma máquina de estado finita dominada por fluxo de controle. Seu projeto, normalmente, é uma tarefa difícil que tende a gerar muitos erros. Quanto mais complexo o protocolo maior a possibilidade de erros e, por esse motivo, a aplicação de métodos formais é essencial. Nesse sentido, a síntese de alto nível auxilia na determinação das unidades funcionais, escalonamento de tarefas e alocação do *hardware/software* para a execução das funções do protocolo. Uma das técnicas mais utilizadas para a especificação de protocolos é a modelagem em redes de Petri, pela facilidade de representar a interação entre as diversas partes do protocolo através de *transições* [42][43].

2.3 HW/SW Codesign

O *HW/SW Codesign* é uma metodologia de projeto concorrente e cooperativa na qual o desenvolvimento dos projetos de *hardware* e de *software* de um sistema é realizado de forma integrada, explorando tanto as vantagens de uma implementação em *hardware* quanto as vantagens de uma implementação em *software*. Essa metodologia tem por objetivo encontrar uma solução ótima da parte do sistema que deve ser implementada em *hardware* e da parte que deve ser implementada em *software*, a partir de uma especificação funcional.

2.3.1 Conceitos Básicos

A interação entre os projetos de *hardware* e de *software* assim como o processo de

partição *hardware/software* se estende pelos diversos níveis do ciclo de desenvolvimento e tem como objetivo encontrar um compromisso entre o desempenho e o custo do sistema. Essa técnica possibilita implementações mais eficientes, melhorando o desempenho e o custo-benefício do sistema. Como decisões de implementação em *software* têm influência no projeto de *hardware* (e vice-versa), os problemas decorrentes dessas decisões podem ser detectados logo no início do projeto e as respectivas alterações podem ser realizadas imediatamente, ainda num alto nível de abstração [44]. Deve existir um processo contínuo de refinamento do sistema para que seja possível realizar uma validação em cada um dos níveis do projeto. A figura 2.2 ilustra uma abordagem de *codesign* comumente empregada. O processo se inicia com uma representação funcional do sistema, independente do domínio do projeto, HW ou SW. Algumas representações do sistema são realizadas utilizando-se máquinas de estado finitas (MEF) e processos concorrentes. Em alguns ambientes de pesquisa, o sistema é descrito usando-se uma linguagem de programação, que é então compilada numa representação interna, tais como descrições de fluxo de dados e de controle.

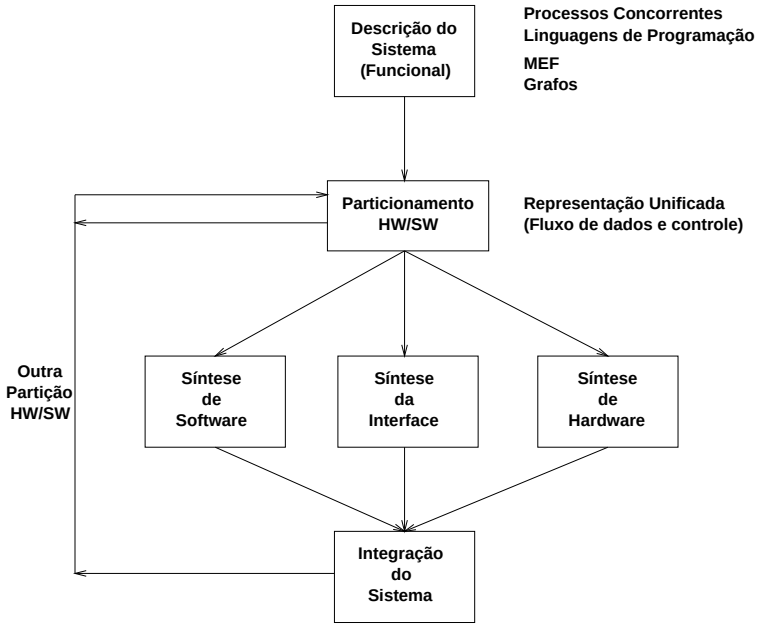

Figura 2.2: Abordagem hardware/software codesign.

O processo de partição determina quais funções serão implementadas em HW e quais serão implementadas em SW. Diversos grupos de pesquisa adotam, basica-

mente, a metodologia apresentada na figura 2.2 como solução para a implementação HW/SW de diversos sistemas.

Quanto maior a complexidade dos sistemas maior a necessidade de uma descrição detalhada. Por isso é extremamente importante o emprego de técnicas de decomposição para facilitar a análise do sistema. A metodologia *HW/SW Codesign* baseia-se fundamentalmente em três pontos: integração dos projetos de *hardware* e *software*, exploração de um compromisso custo versus desempenho entre *software* e *hardware*, através da avaliação das diversas alternativas possíveis de partição HW/SW e o contínuo refinamento da especificação e validação do projeto. A figura 2.3 ilustra esse processo. A metodologia de *HW/SW Codesign* pode ser vista como uma extensão da síntese de alto nível (HLS), como é ilustrado na figura 2.4. Ou seja, a síntese é uma parte do processo de concepção do sistema. As funções do sistema a serem implementadas são descritas como processos concorrentes que interagem entre si e na fase da partição podem ser mapeadas em diferentes componentes físicos.

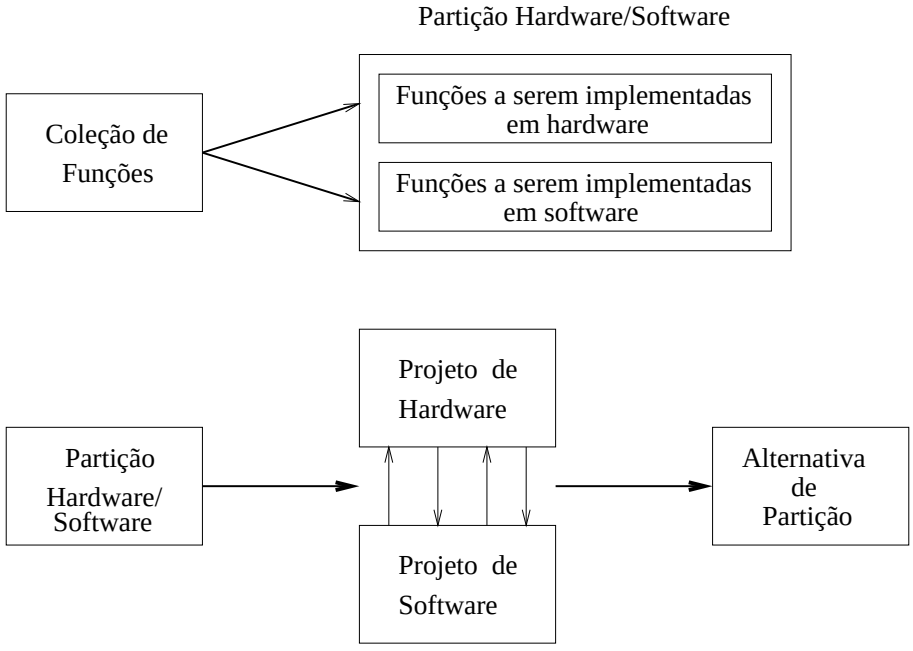

Figura 2.3: Partição HW/SW e Refinamento

Um dos critérios considerados na partição HW/SW é a minimização da comunicação entre as funções do sistema. No caso específico de protocolos de comunicação, esse é um fator importante para melhorar o desempenho, principalmente em termos

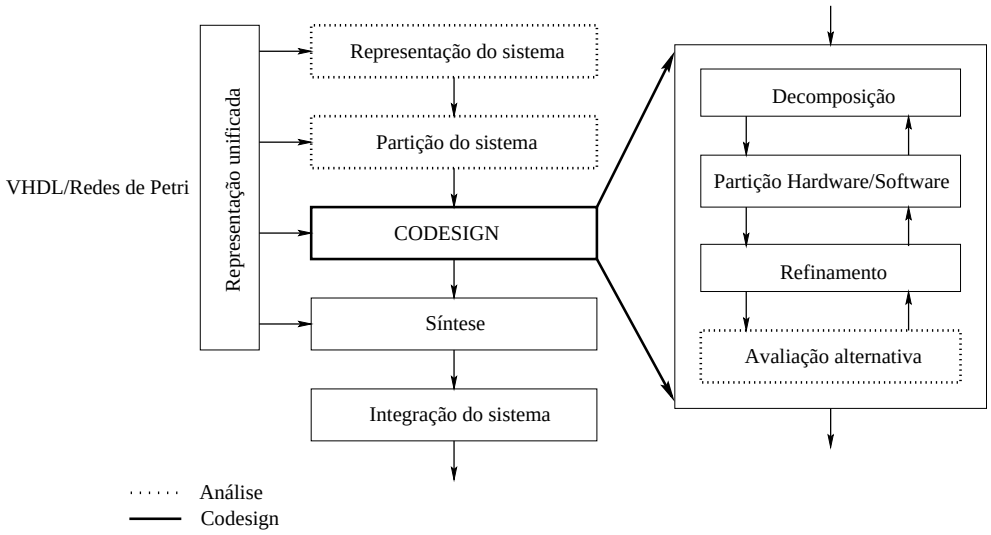

Figura 2.4: Codesign no contexto do projeto do sistema

de vazão e retardo. O processo de *codesign* consiste em realizar, iterativamente, os seguintes procedimentos: decomposição funcional do sistema, a partição HW/SW, o refinamento e a avaliação das alternativas possíveis de partição. Na decomposição, divide-se as funções do sistema em subfunções. A partição determina quais subfunções devem ser implementadas em *hardware* e quais devem ser implementadas em *software*. O refinamento produz novas alternativas de partição HW/SW. Na avaliação das alternativas de partição determina-se o desempenho do sistema e qual o custo associado, com o objetivo de selecionar uma determinada implementação HW/SW.

No processo de síntese do *software* define-se quais os processos concorrentes e realiza-se um escalonamento, ou seja, define-se quais processadores executam cada um dos processos. Deve-se considerar aspectos como minimização do tempo que o processador espera por eventos externos, restrições de tempo de execução para cada processo, sequência de execução de cada processo e garantia de que todos eles podem ser executados pelo menos uma vez. A síntese do *software* consiste em converter uma especificação complexa no nível de sistema em um programa tradicional numa linguagem de alto nível como, por exemplo, a linguagem C. A síntese de *hardware*, descrita detalhadamente na seção anterior, normalmente é realizada a partir de uma linguagem de descrição de *hardware* tal como VHDL e incorpora, além da síntese de

alto nível, a síntese de máquinas de estado finitas e a definição da tecnologia a ser utilizada.

Algumas vantagens da utilização do princípio de *codesign* são:

- Manter juntas as partes referentes à especificação do sistema durante todas as etapas do projeto;
- Possibilitar a detecção de erros nas fases iniciais do projeto;
- Possibilitar a verificação e a simulação de todo o sistema em qualquer etapa do projeto;
- Permitir um refinamento e uma reavaliação da partição em cada etapa do projeto, devido à sua característica iterativa;
- Diminuir os custos de projeto e permitir o teste de sistemas complexos;

2.3.2 Ferramentas de partição HW/SW

Diversos grupos de pesquisa têm desenvolvido ambientes de projeto baseados em *HW/SW Codesign*. Esses projetos diferem fundamentalmente na linguagem de especificação utilizada, no método de partição, na arquitetura alvo e nos métodos de validação utilizados. A seguir são apresentadas algumas características de alguns desses projetos.

COSMOS

O projeto COSMOS consiste de uma metodologia e um ambiente para especificação e síntese de sistemas distribuídos de *hardware* e *software* desenvolvido no Laboratório TIMA em Grenoble-França [3][45]. A primeira etapa de projeto consiste na especificação, utilizando uma das seguintes linguagens: SDL, Estelle, LOTOS, StateCharts, Esterel, CSP ou OCCAM. A descrição do sistema a partir de diversas linguagens é possível devido à utilização de um formato intermediário denominado

SOLAR [45], utilizado pelas diferentes ferramentas de projeto. Esse formato acomoda na mesma representação os principais conceitos de nível de sistemas: hierarquia, paralelismo, comunicação e sincronização.

Este ambiente utiliza uma técnica semi-automática de partição onde, partindo-se de uma descrição funcional do sistema, obtém-se um conjunto de processos a serem implementados em *software*, na linguagem C, que são compilados para algum processador de uso geral, e um conjunto de processos a serem implementados em *hardware*, na linguagem VHDL. A implementação em *hardware* pode utilizar componentes programáveis (PLDs) ou circuitos integrados dedicados (ASICs). Um conjunto de processos concorrentes define uma unidade de projeto (*design unit*) que pode conter outras unidades, formando uma hierarquia de unidades de projeto. A comunicação entre as unidades de projeto ocorre através de canais (*channel units*) que utilizam RPCs (*remote procedure calls*).

A partição HW/SW é realizada através da aplicação de transformações sobre a representação SOLAR, que podem ser de três tipos: decomposição funcional, reorganização estrutural e transformação de comunicação. A decomposição funcional refina as descrições *comportamentais*, a reorganização estrutural é aplicada no nível de unidades de projeto, enquanto que as transformações de comunicação permitem um refinamento das comunicações nos canais.

As transformações mencionadas são aplicadas pelo projetista de forma **manual**, ou seja, o projetista é responsável por determinar a melhor solução de compromisso entre o custo e o desempenho. A figura 2.5 ilustra o sistema COSMOS.

PISH

O sistema de HW/SW *Codesign* PISH (Projeto Integrado Software-Hardware) [10][11][12] foi desenvolvido no Centro de Informática da Universidade Federal de Pernambuco (UFPE). Esse sistema utiliza redes de Petri como um modelo intermediário, possibilitando tanto uma análise qualitativa das propriedades dessa es-

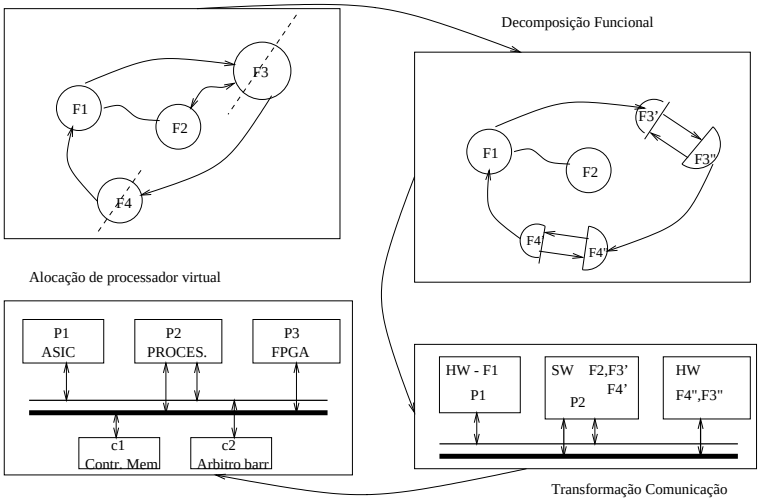

Figura 2.5: A ferramenta COSMOS.

pecificação quanto o cálculo de algumas métricas que são utilizadas no processo de partição.

O método de partição do sistema PISH considera uma arquitetura com múltiplos processadores e baseia-se no algoritmo de *clustering*, onde cada conjunto de funções deve ser implementado em *hardware* ou em *software*. A especificação do sistema é dada por um modelo *comportamental* na linguagem **occam**, que permite descrever de forma explícita a concorrência entre os processos. Essa descrição é traduzida para uma rede de Petri temporizada, que permite uma verificação formal do sistema após realizada a partição, ou seja, permite verificar se o sistema está semanticamente correto e se está de acordo com a especificação inicial. A partir da análise da rede de Petri, pode-se associar cada um dos processos a inúmeras possibilidades de implementação em *hardware* ou em *software*.

Antes da aplicação do algoritmo de *clustering* é necessário realizar um *splitting* da especificação, ou seja, separar todos os processos de maneira que se tenha uma maior flexibilidade quando for realizado o agrupamento desses processos em partições. O projetista define a *granularidade* desejada quando realiza o *splitting*. O algoritmo de *clustering* agrupa os processos de acordo com as métricas calculadas. A partir dessas métricas, da similaridade funcional, da concorrência entre os processos e do custo da

comunicação entre os mesmos é realizada a alocação inicial. O algoritmo de partição também indica os processos responsáveis pela comunicação entre os diversos grupos de processos.

Durante a fase de validação do protótipo pode-se optar pela realização de uma nova partição, caso as restrições de projeto não sejam satisfeitas. O sistema PISH enfatiza a correção do sistema ao qual foi aplicada a partição HW/SW. A figura 2.6 ilustra o sistema PISH.

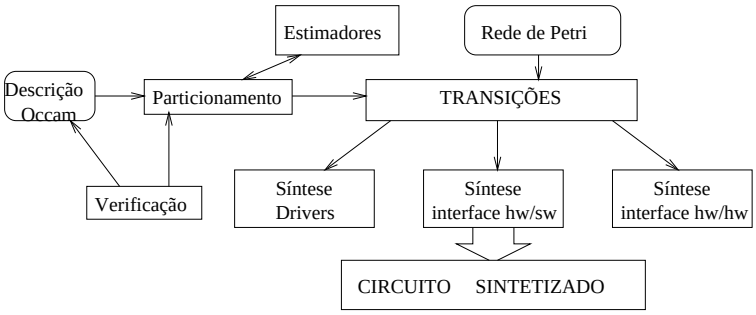

Figura 2.6: O sistema de *HW/SW codesign* PISH.

SPECSYN

O SpecSyn é um ambiente de projeto desenvolvido na Universidade de Irvine que se baseia no paradigma de projeto *especificação-exploração-refinamento* (SER) [4][46]. Essa abordagem especifica precisamente o sistema no nível funcional, explora inúmeras opções de projeto no nível de sistema e refina a especificação na opção selecionada. Uma opção selecionada consiste da alocação de componentes como processadores de uso geral ou dedicados (ASICs) e de uma partição funcional distribuída entre esses componentes.

A *exploração arquitetural* é definida como a tarefa de se encontrar um conjunto de potenciais arquiteturas que possam satisfazer as restrições impostas sobre algumas métricas e otimizar outras. Esse passo consiste nas tarefas de alocação, partição, transformação e estimativa. Normalmente várias iterações são realizadas para se chegar a uma solução.

Alocação é a tarefa de se adicionar componentes ao projeto. O alocador SpecSyn utiliza componentes armazenados numa biblioteca, caracterizados por suas restrições tecnológicas.

Durante o processo de partição, o projetista pode interagir com o sistema e realocar os objetos **manualmente**, controlando o peso das métricas envolvidas na função custo. Uma estimativa da qualidade do projeto é necessária para verificar se uma determinada partição de funções entre os componentes alocados satisfaz as restrições especificadas e também para que seja possível uma comparação entre as diversas alternativas de projeto.

O refinamento consiste na geração de uma nova especificação para cada componente do sistema após a etapa de exploração ter fornecido uma alocação e partição apropriadas. A especificação refinada serve como entrada para a verificação e para a síntese. O processo se encerra com a implementação dos componentes, ou seja, o *software* é compilado e o *hardware* sintetizado. A figura 2.7 ilustra o sistema SpecSyn.

POLIS

O POLIS é um sistema de *co-síntese* desenvolvido na Universidade da Califórnia, Berkeley, para apoiar o projeto de sistemas de tempo real, que respondem a eventos externos de forma assíncrona e possuem restrições de tempo. O sistema é especificado numa linguagem de alto nível (ESTEREL, através de máquinas de estado finitas (FSM's) ou em um subconjunto de Verilog ou VHDL) que pode ser traduzida diretamente em máquinas de estado finitas concorrentes (CFSM's). O fluxo de projeto do sistema POLIS pode ser visualizado na figura 2.8.

O sistema POLIS inclui um tradutor de CFSM para FSM. Dessa maneira, pode-se realizar uma verificação formal do sistema através de ferramentas apropriadas que indicam se a partição realizada preserva a funcionalidade estabelecida na especificação do sistema.

O processo de partição HW/SW é **fortemente baseado na experiência do**

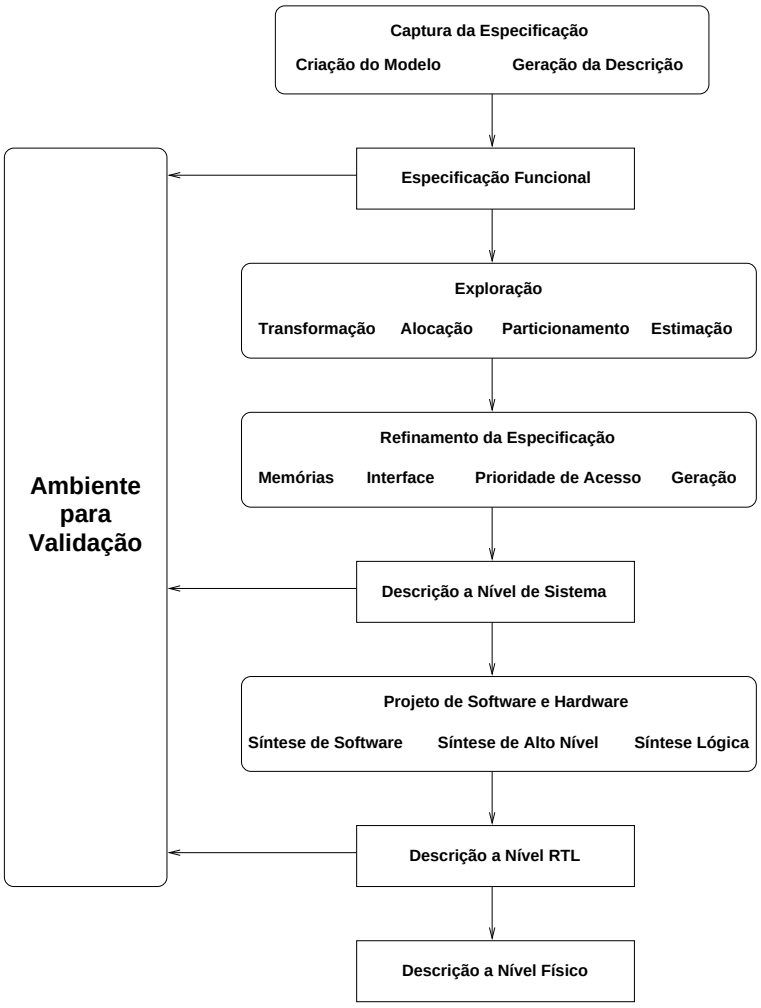

Figura 2.7: O ambiente SpecSyn.

projetista. O sistema POLIS fornece um ambiente para que o projetista possa avaliar rapidamente qualquer decisão tomada em relação à partição HW/SW realizada, à arquitetura adotada e ao escalonador utilizado, a partir da verificação formal ou do sistema de co-síntese.

PTOLEMY

O projeto Ptolemy [5][6][7] é um ambiente para modelagem, projeto e simulação de sistemas concorrentes desenvolvido na Universidade da Califórnia, Berkeley. Permite a simulação de redes de comunicação de dados e de vários outros tipos de

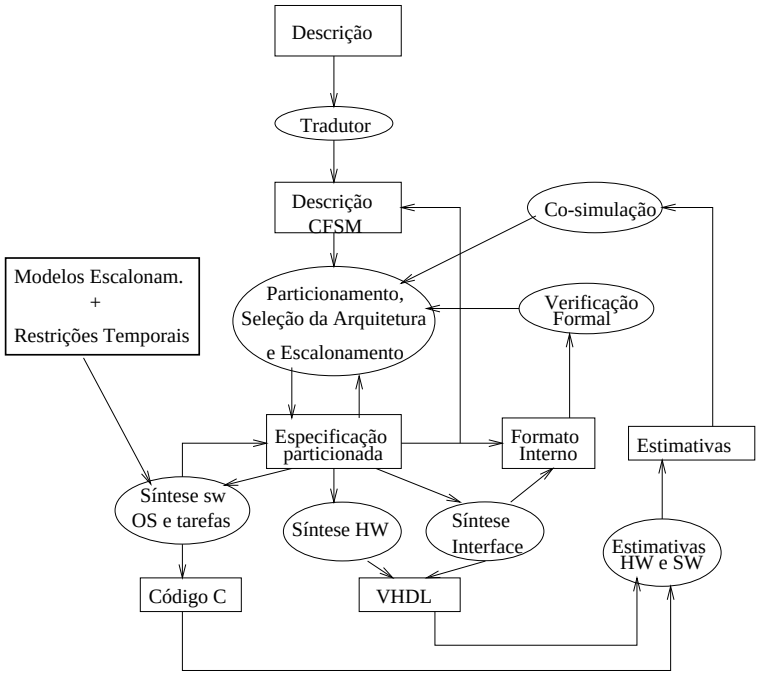

Figura 2.8: O sistema POLIS.

aplicações. Combina ambientes heterogêneos como simulação de eventos discretos e de fluxo de dados. Através da combinação desses diferentes tipos de domínio, o projetista pode modelar e simular sistemas heterogêneos. Essas características tornam o Ptolemy apropriado para o *HW/SW codesign*.

A figura 2.9 mostra como a ferramenta Ptolemy se aplica ao processo de *codesign*. A primeira fase do projeto é responsável pela obtenção das especificações para a aplicação. Em seguida é realizada uma simulação funcional utilizando SDF (*Synchronous Data Flow*) [47]. Posteriormente é realizada, manualmente, uma partição HW/SW do sistema. Nas fases 5, 6 e 7 (vide figura) é realizada uma simulação de *hardware* para a partição HW/SW escolhida. A partir dessa simulação e da partição HW/SW, o *software* é sintetizado (fase 4). Para finalizar o processo, o sistema completo (*hardware* e *software*) é simulado e verificado quanto à sua correção funcional e restrições temporais. O processo é repetido várias vezes para que o espaço de soluções seja explorado e a escolha final possa ser feita pelo projetista.

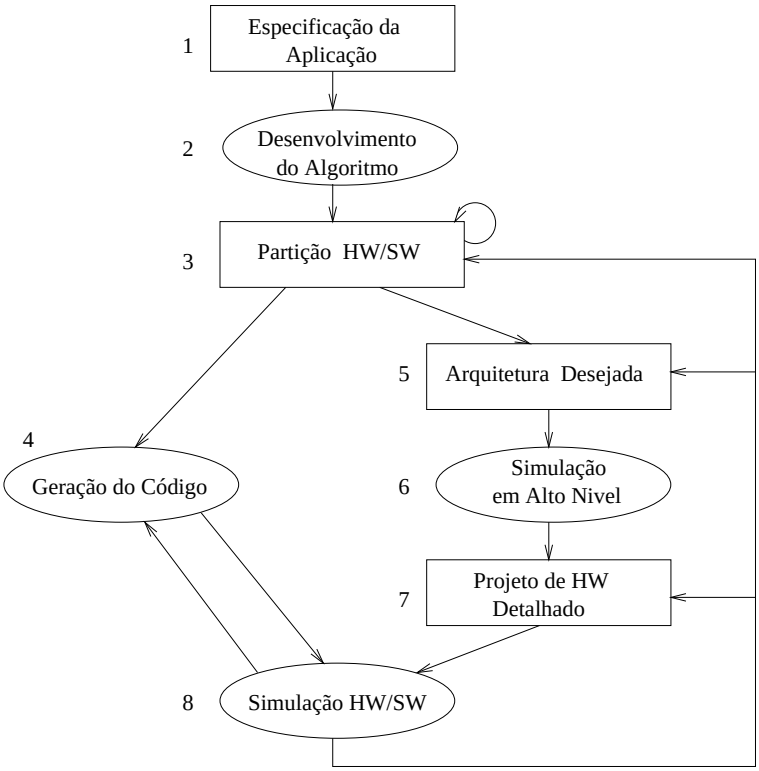

Figura 2.9: Codesign utilizando o Ptolemy.

LYCOS

O sistema LYCOS (LYngby CO-Synthesis) [8] é um sistema de *HW/SW codesign* desenvolvido na Universidade da Dinamarca. O sistema LYCOS é construído a partir de uma série de ferramentas centralizadas em torno de um modelo computacional denominado Quenya, baseado em grafos de fluxos de controle e de dados (CDFGs). O Quenya é uma representação interna que serve de interface entre as linguagens de especificação e a ferramenta de partição HW/SW.

Antes da operação de partição, o projetista seleciona a arquitetura alvo. A arquitetura alvo é composta de um processador de uso geral para o *software* e um componente de *hardware* (ASIC, PLD, etc.), conectados por um canal de comunicação.

A idéia básica do algoritmo de partição é encontrar o menor tempo de execução do sistema (desempenho), combinando sequências de alocação em *hardware* não

sobrepostas, tendo como restrição de custo a área disponível para o *hardware*. Esse processo é manual e depende da experiência do projetista. O algoritmo se baseia em técnicas de programação linear inteira e utiliza informações e estimativas de medidas típicas como: tempo de execução (*hardware* e *software*), tempo de comunicação, tamanho do código e área do *hardware*.

Na exploração das possíveis soluções, devem ser realizadas várias estimativas e partições diferentes. Dessa forma, a velocidade do processo de estimativa é crítica, necessitando de técnicas e heurísticas eficientes, que podem reduzir a complexidade do algoritmo para $O(n^3)$. A figura 2.10 ilustra o sistema LYCOS.

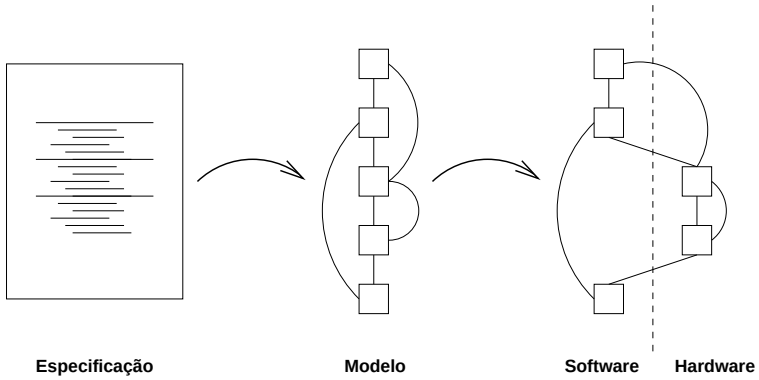

Figura 2.10: O sistema de *HW/SW codesign* LYCOS.

COSYMA

O sistema Cosyma [48] é um sistema de partição HW/SW desenvolvido na Universidade de Braunschweig cuja descrição de entrada consiste de vários processos que se comunicam e que possuem restrições de tempo.

A partição HW/SW é realizada no nível de blocos funcionais e requer uma análise da comunicação entre esses blocos e, posteriormente, a realização da síntese da comunicação. O processo de partição supõe, inicialmente, que todas as funções são implementadas em *software* e por isso diz-se que o mesmo é orientado ao *software*. A partição HW/SW é realizada transferindo-se funções para os componentes do *hardware*, iterativamente, até que todas as restrições de tempo sejam atingidas. Durante

a partição HW/SW procura-se, além de atingir as restrições de tempo real, minimizar os custos do hardware e o tempo de resposta do sistema. A minimização do tempo de resposta tem como objetivo verificar rapidamente a influência de possíveis modificações no sistema no que se refere ao custo e ao desempenho. A figura 2.11 ilustra o sistema COSYMA.

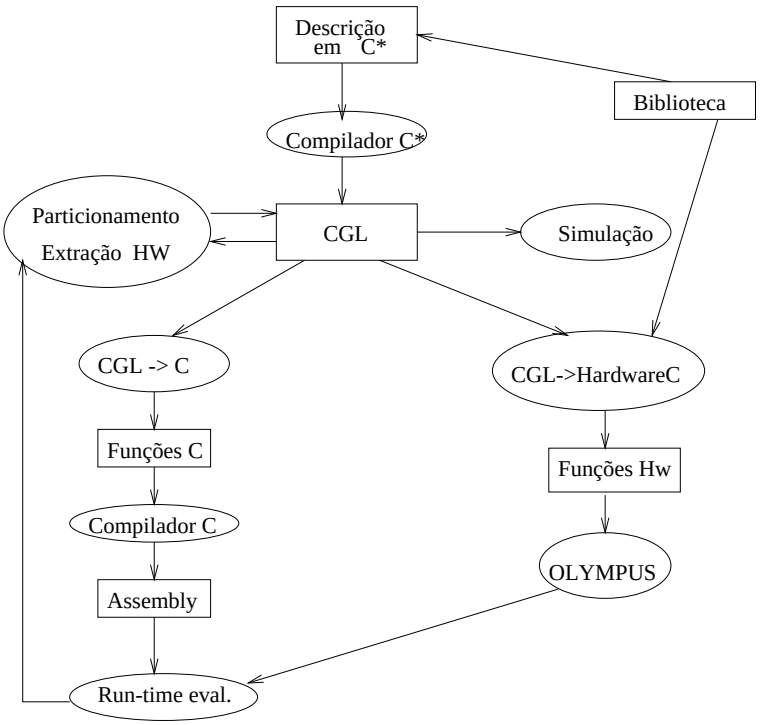

Figura 2.11: O sistema de partição COSYMA.

A função objetivo é composta pelo tempo de execução de cada bloco e pelo tempo de comunicação entre os mesmos. Dado um número máximo de componentes especificado pelo projetista, o objetivo é encontrar um conjunto mínimo de segmentos de código que leva a um ganho de velocidade quando implementado em *hardware*. O algoritmo utilizado para essa otimização é o *simulated annealing* [49].

2.3.3 HW/SW Codesign de Protocolos

Na maioria das ferramentas de partição apresentadas na seção 2.3.2 pode-se ob-

servar que a partição HW/SW é realizada de forma manual e/ou depende muito da experiência do projetista. O projetista não dispõe de critérios e medidas objetivas para auxiliá-lo no processo de partição. Este trabalho se diferencia das ferramentas existentes na medida em que fornece ao projetista dados concretos para a realização da partição. A técnica utilizada neste trabalho utiliza medidas de desempenho especificadas pelo projetista e calculadas por uma ferramenta apropriada. A otimização é realizada por um algoritmo genético. A metodologia de projeto é descrita com maiores detalhes no capítulo 3.

Particularmente no projeto de protocolos, a utilização da metodologia de *HW/SW codesign* tem assumido maior importância devido à crescente demanda por redes com alto desempenho, que suportem um alto tráfego. As implementações de parte do protocolo em *hardware* têm sido cada vez mais estudadas em virtude da constante evolução da tecnologia de *hardware*, dedicado a um baixo custo. Na partição da especificação de um dado protocolo, a parte mais sensível à variação do desempenho deve ser endereçada a um ambiente de implementação em *hardware*, enquanto a parte menos crítica deve ser endereçada a um ambiente de implementação em *software*.

O uso de técnicas de descrição formais é importante na fase de especificação do protocolo, pois permite que o modelo inicial seja verificado e validado em relação à sintaxe e em relação à semântica, ou seja, permite eliminar, por exemplo, *deadlocks*, e verificar a correção do modelo em relação à especificação inicial (ou "textual"). Neste trabalho são utilizadas as redes de Petri estocásticas para a modelagem inicial dos protocolos, devido à facilidade de visualização e validação da especificação em ferramentas apropriadas. Pode-se utilizar, por exemplo, um Analisador de Rede de Petri (ARP) [33] que fornece dados como: número de estados alcançáveis do protocolo, existência ou não de bloqueios (*deadlocks*) e *loops* infinitos (*livelocks*), além de características como os **invariantes** de transição, que definem sequências cíclicas de disparo de transições, tornando possível saber se o comportamento do modelo atende aos requisitos especificados.

Cada transição da rede de Petri representa um conjunto de funções a serem exe-

cutadas pelo protocolo, ou seja, funções a serem executadas por um componente de *hardware* ou por um processador padrão, em *software*, após a realização da partição. Cada *marca* da rede de Petri [42] representa um estado possível do protocolo. Uma *marca* de uma rede de Petri representa uma possível distribuição de *fichas* cada vez que uma transição é disparada. Nos exemplos apresentados no capítulo 5 as redes de Petri são binárias, isto é, cada *lugar* da rede de Petri possui uma ou zero fichas. Isso simplifica a análise do modelo e diminui o número de estados possíveis da cadeia de Markov associada. Também podem ser usadas linguagens de descrição de *software* ou de *hardware*. Como exemplo de linguagem de descrição de *software* pode-se citar as linguagens Estelle, Lotos e CCS. Entre as linguagens de descrição de *hardware*, a linguagem VHDL é uma das mais utilizadas.

2.3.4 A Partição HW/SW como um Problema de Otimização

Um dos maiores desafios dos algoritmos de partição HW/SW existentes é o grande número de alternativas possíveis a serem analisadas. Por esse motivo o problema de partição pode ser classificado como um problema de otimização, onde são combinados os diversos parâmetros de desempenho e custo do problema numa função única, denominada *função objetivo*. Genericamente, o principal objetivo de um algoritmo de partição é mapear um conjunto de objetos funcionais em um sistema composto de vários componentes, de maneira a otimizar essa função objetivo, ou seja, encontrar uma solução que minimize o custo e maximize o desempenho.

O problema da escolha da melhor partição HW/SW é um problema NP-difícil [50][51], ou seja, o algoritmo para a sua solução possui uma alta complexidade. Muitas metodologias de partição têm sido propostas: Gupta e Micheli [52][53] propõem uma metodologia que desloca paulatinamente as funções do sistema do *hardware* para o *software*, até que se alcance o limite mínimo do desempenho desejado, com o menor custo possível. Ernst [48] realiza o processo inverso, isto é, move as funções mais críticas do *software* para o *hardware* até que o desempenho deseja-

do seja alcançado. Woo [54] divide as funções em grupos de *hardware* e *software* e analisa posteriormente o desempenho da partição, de forma manual. Alomary [51] propõe um método de otimização combinatória, utilizando um algoritmo baseado no método *branch-and-bound* [55] e uma tabela que fornece valores de custo para os componentes de *hardware*.

Verifica-se em todos os trabalhos acima um processo iterativo de tentativa e erro, onde inicialmente faz-se a partição e posteriormente analisa-se o desempenho da mesma. Na metodologia proposta neste trabalho a experiência do projetista também é importante, principalmente na etapa de modelagem do protocolo, porém o desempenho é estabelecido *a priori* e os dados fornecidos após o processo de otimização e o critério de partição proposto tornam o **processo de partição HW/SW** menos dependente da experiência do projetista.

2.4 Técnicas de Descrição Formal

A seguir são descritas, sucintamente, algumas técnicas utilizadas para a especificação, verificação, e implementação de sistemas concorrentes e de protocolos, em particular. Essa especificação pode ser realizada de forma textual ou gráfica. Neste trabalho são utilizadas as redes de Petri estocásticas para especificar o protocolo a ser modelado. A ferramenta ARP [33] é utilizada para realizar a verificação dessas redes de Petri. Em seguida são apresentadas outras técnicas de descrição formal (TDFs) usualmente utilizadas na especificação e implementação de protocolos.

A validação da especificação pode ser vista sob os aspectos estático e dinâmico. A validação estática inclui a verificação de erros léxicos, sintáticos e relativos à semântica da especificação. A análise dinâmica é realizada simulando-se a especificação. A implementação pode ser submetida a um teste de conformidade. Nesse tipo de teste estimula-se o modelo com entradas programadas e as saídas emitidas são analisadas para se averiguar o comportamento da implementação em relação à especificação.

2.4.1 Redes de Petri Estocásticas

Uma das técnicas bastante utilizadas para a especificação e projeto de protocolos de comunicação é a descrição em redes de Petri [42]. Uma extensão dos modelos de rede de Petri tradicionais são as redes de Petri estocásticas (SPNs) [31][32]. Numa rede de Petri estocástica, uma transição, uma vez habilitada, é disparada após um período de tempo que obedece a uma distribuição de probabilidades exponencial. As redes de Petri estocásticas possibilitam tanto um estudo das propriedades qualitativas quanto uma análise quantitativa do desempenho de sistemas que envolvem concorrência e sincronização, como é o caso dos protocolos de comunicação. No entanto, a suposição de tempo de disparo exponencial é um fator que limita a análise para alguns tipos de sistemas cujos tempos não possuem distribuição exponencial. Para esses casos, foram propostas diversas extensões para as SPNs, tais como: as GSPNs, redes de Petri estocásticas generalizadas, que permitem a existência de transições com tempo de disparo igual a zero, as ESPNs, redes de Petri estocásticas extendidas, que permitem a existência de transições com outros tipos de distribuições e as DSPNs, redes de Petri estocásticas determinísticas, que permitem a existência de transições com tempo de disparo determinístico. Este trabalho utiliza apenas as SPNs tradicionais.

ARP - Analisador de Redes de Petri

Este trabalho utiliza essa ferramenta para validar a especificação do protocolo, elaborada utilizando-se as redes de Petri estocásticas. O ARP é uma ferramenta para análise e simulação de redes de Petri desenvolvida no laboratório de controle e microinformática da Universidade Federal de Santa Catarina. Foi desenvolvida em turbo Pascal e é executada nos sistemas operacionais MS-DOS ou Windows. Apesar da simplicidade dos ambientes no qual é executada, é uma ferramenta bastante poderosa e que fornece informações preciosas ao projetista quanto à correção do

protocolo. Atualmente, está sendo desenvolvida uma interface gráfica na linguagem Java.

A ferramenta ARP pode analisar redes de Petri dos seguintes tipos: redes lugar/transição com *marcas* numéricas e sem informação de tempo; redes temporizadas onde, uma vez habilitada, a transição é disparada dentro de um intervalo $[t_{min}, t_{max}]$ associado a ela; e redes temporizadas estendidas, nas quais uma função de probabilidade é associada a cada intervalo de disparo da transição. Neste trabalho as redes de Petri são estocásticas, ou seja, cada transição, uma vez habilitada, tem um tempo de disparo com distribuição de probabilidades exponencial.

As seguintes análises são realizadas pelo ARP:

- análise de *acessibilidade*: todos os estados possíveis são encontrados e é construído o *grafo de alcançabilidade* da rede. Algumas propriedades importantes são verificadas: presença de *deadlocks* ou *livelocks*, número máximo de fichas nos lugares e se existe alguma transição não disparável. Essas duas últimas propriedades nos informa se a rede é *viva*, *limitada* e *reiniciável* [42].

- análise de invariantes: fornece todas as invariantes de lugar e de transição existentes na rede. As invariantes de transição definem sequências cíclicas de disparo de transições e as invariantes de lugar estabelecem uma relação linear entre um subconjunto de lugares da rede de Petri. Uma invariante de lugar pode informar se a rede é *conservativa* [56] para um subconjunto de lugares da rede. As invariantes também podem ser utilizadas para demonstrar a correção parcial de um protocolo.

O ARP também pode fornecer, através de simulação, dados estatísticos como, por exemplo, o número médio de fichas em um determinado lugar, tempo médio do retardo no disparo das transições temporizadas e o tempo médio para se alcançar um determinado estado. A simulação também pode ser realizada manualmente, possibilitando ao usuário encontrar erros na rede, disparando as transições passo a passo.

O programa também possibilita o gerenciamento de arquivos, contém um editor

simples e um *browser* para os diretórios. As redes são descritas utilizando-se uma linguagem similar ao Pascal.

2.4.2 Outras TDFs

Estelle

A linguagem Estelle (Extended State Transition Language) [57] é uma técnica de descrição formal padronizada pela ISO (International Organization for Standardization). O objetivo foi produzir uma TDF para a especificação de sistemas distribuídos, protocolos de comunicação e, em particular, padrões relativos ao modelo de referência OSI. A linguagem Estelle é baseada numa Máquina de Estados Finita Estendida (MEFE), que combina dois tipos de notações: estados, interações e transições que são descritos através de uma MEF. Variáveis, parâmetros e prioridades são descritos através da linguagem de programação Pascal. Em Estelle uma especificação é composta de um conjunto de módulos onde cada módulo é representado por uma caixa preta com portas de entrada/saída, denominadas pontos de interação. Canais de comunicação bidirecionais podem conectar os módulos que se comunicam através de filas FIFO associadas a *pontos de interação*. Um módulo pode ser refinado em submódulos, definindo uma estrutura hierárquica entre os componentes da especificação. Canais internos podem conectar submódulos e as portas dos submódulos filhos podem ser vinculadas às portas do módulo pai. A figura 2.12 ilustra o exemplo de um sistema especificado em Estelle.

CCS

O CCS (*Calculus of Communicating Systems*) [58] é uma álgebra para especifi-

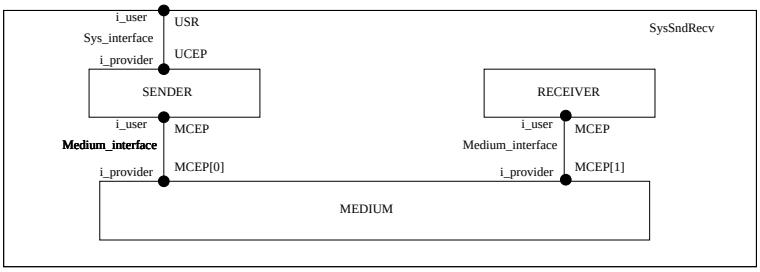

Figura 2.12: Exemplo de uma especificação em Estelle.

cação de sistemas concorrentes, representados por *agentes comunicantes*, que fornece um conjunto de termos, operadores e axiomas que podem ser usados para escrever e manipular expressões algébricas. As expressões definem os elementos do sistema e a forma como são manipuladas revelam o seu comportamento. Da mesma forma que uma teoria bem formulada sobre aspectos computacionais, como é o caso do estudo de máquinas de Turing [59], ajuda a entender as noções básicas da teoria de computação tradicional, o CCS ajuda a entender as noções da computação concorrente. Por exemplo, pode-se determinar se duas implementações diferentes de sistemas concorrentes são equivalentes e, consequentemente, se um sistema pode ser substituído pelo outro. O CCS também fornece subsídios para se determinar certas propriedades de um sistema, como ausência de *deadlocks* e *liveness*.

Os cinco operadores básicos do cálculo de processos são: a *ação*, a *rerrotulação*, a *seleção*, a *composição* e a *restrição*. O operador *ação* permite que o comportamento de um agente seja definido por uma sequência de eventos. A *rerrotulação* possibilita renomear as ações de um agente para que fiquem mais fáceis de serem entendidas.

O elemento básico no CCS é o *agente*, que possui um nome (único) e um determinado comportamento. O comportamento de um agente é definido por um conjunto de eventos ou ações que o agente é capaz de realizar. Por exemplo, seja um agente denominado **buffer** e um conjunto de eventos in, out. A sequência de eventos realizados pelo agente é descrita através do operador *ação*, denotado por um "·". O agente **buffer** pode ser definido para realizar a sequência de ações in, out, respectivamente, para armazenar um valor e depois recuperar o mesmo valor.

Os agentes são concorrentes no sentido de que podem atuar independentemente um do outro, porém pode existir uma interação entre eles. A possibilidade de interação entre os agentes é indicada por meio dos nomes de suas ações. Isso pode ser ilustrado através do exemplo abaixo:

$$F = a.x.F \tag{2.1}$$

$$G = \overline{x}.b.G \tag{2.2}$$

$$H = (F|G) \tag{2.3}$$

No exemplo acima, H é definido como a composição dos agentes F e G. Note que o agente F possui a ação x como um de seus eventos e o agente G possui a ação $\overline{x}$. Dois eventos com o mesmo nome base (x, no caso) indica que os mesmos podem interagir através daquela ação.

LOTOS

A linguagem LOTOS (Language of Temporal Ordering Specification) [60] é uma técnica de descrição formal para o projeto de sistemas concorrentes e distribuídos. Consiste de uma linguagem para a especificação de processos que se baseia em dois métodos de álgebra de processos, o CCS e o CSP (*Communicating Sequential Processes*). Entre as TDFs, a linguagem LOTOS mostra-se a de maior poder de expressão e abstração, mas, em contrapartida, a de maior complexidade.

A linguagem LOTOS se baseia na idéia de que provedores de serviço e protocolos de comunicação podem ser especificados pela definição das interações que constituem o *comportamento observável* desses sistemas. Utiliza os métodos algébricos introduzidos por Milner em CCS como também descrição de estruturas de dados e expressões de valores, baseada na teoria formal de tipos abstratos de dados. Não existe um mecanismo para a expressão de comportamentos dependentes do tempo, necessário, por exemplo, em aplicações multimídia.

Devido à falta de uma sintaxe gráfica, algumas propostas foram criadas tendo como propósito principal o mapeamento das estruturas LOTOS para uma representação visual mais compreensível. Uma dessas propostas é o G-LOTOS (LOTOS gráfico) [61].

Embora os serviços e os protocolos de comunicação sejam as principais áreas de utilização da linguagem LOTOS, ela também pode ser aplicada em outras áreas, como: especificação formal do processamento de chamadas telefônicas, especificação de mecanismos de segurança e especificação de algoritmos distribuídos.

2.5 Ferramentas de Análise de Desempenho

As ferramentas de análise de sistemas computacionais em geral podem ser caracterizadas de acordo com os tipos de modelos que utilizam para realizar a análise do sistema em questão. Esses modelos podem ser classificados da seguinte forma [62]: modelos para a análise de confiabilidade/disponibilidade (*dependability*), modelos para a análise de desempenho, que consideram que o sistema é perfeitamente confiável, e modelos para a análise de desempenho e confiabilidade (*performability*), onde as duas métricas são igualmente representadas. A análise de *dependability* representa as mudanças na estrutura do sistema causada por falhas em seus componentes. A análise de desempenho representa o comportamento do sistema mediante solicitações diversas a recursos que devem ser compartilhados. A análise de *performability* realiza uma previsão do desempenho do sistema considerando uma probabilidade de falha, ou seja, que a arquitetura do sistema muda dinamicamente.

A seguir são apresentadas algumas ferramentas existentes e suas características em relação ao tipo de análise que realizam e aos respectivos tipos de modelos que utilizam para realizar essa análise. A ferramenta utilizada neste trabalho é descrita com maiores detalhes no capítulo 4.

2.5.1 SHARPE

A ferramenta SHARPE (Symbolic Hierarchical Automated Reliability and Performance Evaluator) [63] é utilizada para análise de desempenho, de *dependability* e de *performability*. Essa ferramenta combina a flexibilidade de modelos markovianos e a eficiência de modelos *combinatoriais*.

Dadas as distribuições de probabilidades que descrevem o comportamento dos componentes de um sistema e a estrutura do mesmo, pode-se determinar qual será o comportamento de todo o sistema em função do tempo. Tempo entre falhas, tempo de execução de uma tarefa e a probabilidade de um componente estar funcionando convenientemente após um determinado tempo são algumas das medidas que descrevem o comportamento dos componentes. A estrutura do sistema pode ser dada, por exemplo, por uma cadeia de Markov.

Para análise de *dependability* são usados modelos como: digramas de blocos, árvore de falhas e grafos de confiabilidade. Para análise de desempenho utiliza-se grafos de tarefas e redes de filas. Grafos normalmente são utilizados para a análise de processos concorrentes e estocásticos. Os modelos de filas são úteis na análise de desempenho de sistemas, onde uma quantidade limitada de recursos deve ser compartilhada.

Modelos excessivamente grandes podem ser evitados devido à característica hierárquica da ferramenta SHARPE. Essa ferramenta possibilita ainda a combinação dos resultados de diferentes tipos de modelos. A interface gráfica da ferramenta permite, por exemplo, que o usuário defina uma cadeia de Markov através de uma matriz e posteriormente adicione novos nós (estados) e arcos (taxas). A interface também possui um banco de dados com o nome dos componentes do sistema e suas respectivas taxas de falha. Permite ainda desenhar gráficos com os resultados e criar planilhas no aplicativo Excel com esses resultados. A interface foi construída na linguagem Java.

2.5.2 MARCA

A ferramenta MARCA (MARkov Chain Analyzer) [64][65] é projetada para facilitar a geração e análise de cadeias de Markov com um grande número de estados. Essa ferramenta pode tratar cadeias de Markov a tempo discreto ou cadeias de Markov a tempo contínuo e utiliza a matriz de transição da cadeia para extrair propriedades como periodicidade, classificação dos seus estados e cálculo de medidas como distribuições em estado estacionário, distribuições em estado transiente e tempo médio até a absorção.

Para se desenvolver modelos baseados em cadeias de Markov é importante que se tenha um ambiente no qual o sistema a ser estudado possa ser representado de uma maneira fácil de se entender. Técnicas comumente utilizadas com essa finalidade são as redes de Petri estocásticas, redes de filas, entre outras. O estado de um sistema é formado pelo estado de cada um de seus componentes individuais, que podem ser vistos como os elementos de um vetor, denominado "vetor de descrição de estados". Com a finalidade de facilitar essa visualização do sistema a ferramenta MARCA utiliza o termo *balde* para indicar um componente e o termo *bola* para indicar o valor desse componente. A cada instante, o estado do modelo é representado pelo número de bolas em cada balde. Esse procedimento proporciona uma grande flexibilidade na modelagem de sistemas complexos. Por exemplo, o estado de um sistema onde vários processos compartilham recursos pode ser dado pela quantidade de recursos que cada processo do sistema está utilizando e a quantidade de recursos ainda disponíveis. Nesse caso, cada processo representa um balde e cada recurso representa uma bola. O movimento de uma bola de um balde para outro equivale a uma transição de estado à qual está associada uma *taxa de transição*.

Para se construir um modelo utilizando a ferramenta MARCA, o usuário deve fornecer as seguintes informações: o número de baldes, o número máximo de bolas que cada balde suporta, um estado inicial, ou seja, um valor inicial de bolas para cada balde, uma lista de todas as transições possíveis que podem acontecer, informações

necessárias para a determinação das taxas de transição entre os estados, o tipo de saída desejada para o espaço de estados, além de dados específicos para que algumas subrotinas escritas pelo usuário possam ser executadas. Esses dados são fornecidos através de um arquivo de entrada chamado **IN**.

A especificação da matriz de transição é o primeiro passo a ser dado pelo usuário, podendo ser realizada de quatro formas diferentes: utilizar uma facilidade da ferramenta gerando o espaço de estados a partir de um estado inicial e dos estados alcançáveis a partir desse, assim como as respectivas taxas de transição e probabilidade com que cada transição ocorre; digitação dos estados fonte, destino e das taxas de transição, interativamente; ler a matriz de um arquivo no formato (estado-fonte, estado-destino, valor), onde "valor" pode ser uma probabilidade de transição ou uma taxa de transição, dependendo do tipo de cadeia de Markov que o usuário está especificando. O quarto modo de especificar a matriz de transições é no próprio formato usado internamente pela ferramenta, ou seja, no formato Harwell-Boeing. Uma vez especificada a matriz de transições, o usuário pode salvá-la num arquivo denominado MATX_out.

As informações relativas às características da cadeia de Markov, como periodicidade e grau de decomposição, assim como informações sobre o comportamento dos métodos de solução escolhidos, são enviadas para um arquivo denominado **INFO**.

A ferramenta MARCA também possui uma interface gráfica denominada XMARCA que possibilita ao usuário manipular objetos de forma a construir uma rede de filas que é resolvida pelos métodos numéricos disponíveis. XMARCA é uma interface X-Windows para a ferramenta MARCA e foi projetada especificamente para a análise de modelos de filas.

Entre os métodos numéricos disponíveis para a solução da cadeia de Markov podem ser citados: eliminação gaussiana, SOR, Power, entre outros, para solução em estado estacionário; e Runge-Kutta, para solução em estado transiente. A saída dos resultados pode ser de forma gráfica ou em forma de tabelas.

2.5.3 SPNP

O SPNP (Stochastic Petri Net Package) [66] é uma ferramenta de modelagem para a solução analítica de SPNs. São gerados e resolvidos modelos de recompensa de Markov (MRMs) através da utilização de redes de Petri estocásticas de recompensa (SRNs) [67], que são uma extensão das redes de Petri. SPNs não-markovianas também podem ser descritas e resolvidas, assim como também podem ser realizadas simulações de eventos discretos. As SPNs são descritas na linguagem de entrada do SPNP, CSPL (C-based SPN Language), uma extensão da linguagem C que facilita a descrição das SPNs. Outra opção para a descrição da SPN é utilizar a interface gráfica do SPNP para especificar suas características. A mesma interface é usada para especificar os parâmetros do método de solução escolhido.

O SPNP também permite que seja realizada uma análise na rede de Petri, através da qual podem ser verificadas diversas propriedades do modelo construído, como *liveness* e invariantes.

Algumas das distribuições de probabilidade dos tempos de disparo das transições atualmente implementadas são: exponencial, uniforme, geométrica, Weibull, lognormal, Erlang, hiperexponencial, Pareto, Poisson, binomial, Gamma, entre outras.

O SPNP possui dois tipos de métodos de solução: os métodos numérico-analíticos e os métodos de simulação. Para análise em estado estacionário, os métodos disponíveis são: SOR, Gauss-Seidel e Power. Entre os métodos para análise em estado transiente da CMTC estão: uniformização padrão e uniformização usando o método Fox e Glynn para o cálculo das probabilidades de Poisson. O SPNP também possibilita uma análise da sensibilidade do sistema a pequenas mudanças na entrada. Essa análise é útil para a otimização do sistema.

Um novo ambiente para modelagem utilizando SPNs, denominado iSPN, foi desenvolvido para facilitar a criação de SPNs através de uma representação gráfica desses modelos. O iSPN é composto de um editor de redes de Petri, rotinas de visualização para a análise dos resultados e de um *debugger*. A versão atual dessa

interface foi desenvolvida na linguagem Java. Os resultados obtidos pela ferramenta SPNP podem ser usados como entrada da ferramenta SHARPE.

2.5.4 SMAQ

A ferramenta SMAQ (*Statistical Match And Queueing*) [68] é utilizada para modelagem de tráfego e análise de sistemas de filas, possibilitando extrair medidas para vários tipos de tráfego diferentes. Uma vantagem importante que diferencia essa ferramenta de outras é a possibilidade de modelar processos de chegada e de serviço complexos, que não se ajustam aos modelos estocásticos conhecidos. Por exemplo, é difícil descrever com um modelo genérico o processo de chegada de um tráfego multimídia em redes de alta velocidade, devido à sua diversidade de características. Normalmente, somente algumas medidas dos processos que caracterizam a chegada de mensagens e o tempo de serviço são mensuráveis. O objetivo da ferramenta SMAQ é desenvolver uma técnica de modelagem genérica que capture as medidas mais importantes desses processos e forneça uma solução para avaliar o comportamento do sistema em relação, por exemplo, ao tamanho do *buffer* e à taxa de perda.

Existem três componentes básicos no SMAQ: o primeiro se preocupa em identificar quais as medidas estatísticas mais importantes dos processos de chegada e de serviço para a análise do sistema de filas com *buffer* finito e construir uma estrutura para coletar essas medidas. Verifica-se que estatísticas de primeira e segunda ordem, como a função de distribuição acumulada e de autocorrelação, respectivamente, têm um impacto mais significativo no desempenho do sistema do que estatísticas de maior ordem.

O segundo componente é composto por uma biblioteca de modelos CMPP (*Circulant Modulated Poisson Process*), que incorpora as medidas coletadas no primeiro componente e que podem ser usados como geradores de tráfego de forma a permitir simulações e testes para análise de uma rede.

O terceiro componente fornece várias soluções numéricas para o cálculo de medidas de interesse do sistema de filas, como tamanho da fila e taxa de bloqueio, taxa de perda, etc, para um dado tipo de tráfego. Um método computacional denominado *folding algorithm* é utilizado para fornecer as soluções em estado estacionário e transiente. Esse algoritmo possibilita um cálculo preciso do tamanho da fila e da taxa de perda de sistemas com *buffer* finito enquanto que a maioria das outras técnicas utilizadas em teoria de filas têm que supor que o *buffer* é infinito.

A ferramenta SMAQ utiliza uma interface gráfica (GUI) desenvolvida na linguagem de programação Java e, portanto, pode ser disponibilizada para qualquer usuário da Internet cujo *browser* suporte Java.

2.5.5 TWOTOWERS

A ferramenta TWOTOWERS [69] é utilizada para a análise de desempenho e de propriedades funcionais de sistemas concorrentes especificados como cláusulas de sistemas de recompensa baseados em cálculo de processos e temporizados estocasticamente (EMPA_r), uma extensão do EMPA (*Extended Markovian Process Algebra*). As análises das propriedades funcionais e de desempenho se baseiam, respectivamente, em duas outras ferramentas existentes: CWB-NC e MARCA. O projetista desenvolve a representação algébrica do sistema de forma modular e, dessa forma, cada subsistema pode ser desenvolvido separadamente e reagrupados posteriormente através dos operadores da álgebra. A análise de cada termo, ou subsistema, é realizada pelo modelo semântico, composto por um sistema de transições rotulado com ações e seu respectivo tempo de duração. A figura 2.13 ilustra a ferramenta.

Uma vantagem dessa ferramenta sobre outras semelhantes (PEPA [70] e TIPP [71]) é aproveitar duas ferramentas conhecidas para realizar a análise funcional e de desempenho, pois dessa forma o projetista não necessita implementar novamente as rotinas de análise. Além disso, é fornecida ao usuário uma larga variedade de técnicas automatizadas para o estudo de sistemas concorrentes.

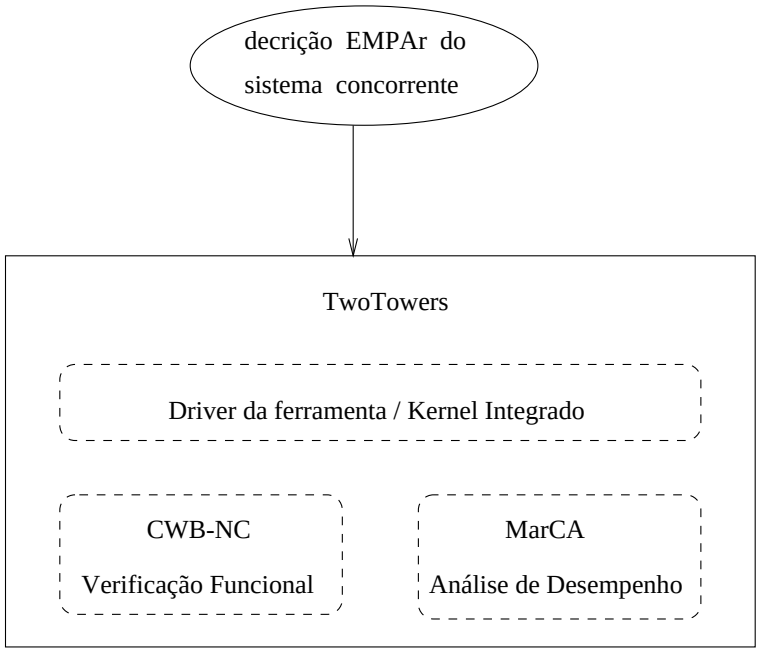

Figura 2.13: A ferramenta TwoTowers construída sobre o CWB-NC e MarCA

2.6 Tangram-II

O Tangram-II é uma ferramenta utilizada para especificação e análise de desempenho e confiabilidade de sistemas, principalmente sistemas de comunicação e computacionais. Foi desenvolvido com objetivos educacionais e de pesquisa no laboratório Land da COPPE/UFRJ [22][23].

Entre as principais características da ferramenta Tangram-II estão: possuir uma interface gráfica, o TGIF, baseada no paradigma de orientação a objetos, desenvolvida na linguagem Java, além de possuir uma variedade de métodos de solução para obtenção de medidas relacionadas ao modelo. Esses métodos possibilitam a análise em estado estacionário e em estado transiente.

O ambiente do Tangram-II também possui um simulador interativo e ferramentas multimídia que ajudam no processo de modelagem e trabalho cooperativo. Existem vários módulos para modelagem e análise de tráfego em redes de computadores.

A figura 2.14 mostra os principais componentes do ambiente Tangram-II. O

sistema a ser modelado é representado por uma coleção de objetos que interagem, enviando e recebendo *mensagens*. O estado de um objeto é representado por um conjunto de variáveis e o seu comportamento é definido por *eventos* e *mensagens*, assim como pelas *condições* que habilitam os *eventos* e as *ações* executadas quando um *evento* é disparado ou uma *mensagem* é recebida.

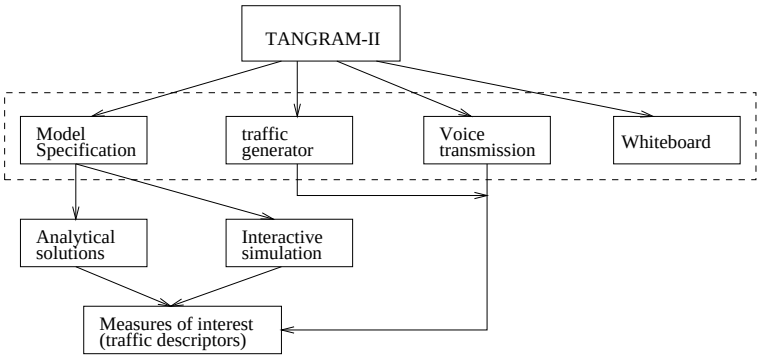

Figura 2.14: O ambiente Tangram-II

A ferramenta Tangram-II foi escolhida para modelar os protocolos da metodologia apresentada neste trabalho devido às características apresentadas, facilitando a implementação de modelos de diversos tipos e, particularmente, de protocolos de comunicação. Além disso, pode-se obter, com essa ferramenta, as medidas de desempenho especificadas pelo projetista e utilizadas pelo algoritmo genético na otimização e obtenção dos parâmetros necessários para se realizar a partição. Outro fator fundamental na escolha foi a disponibilidade da ferramenta no laboratório Land, acesso a todos os recursos materiais daquele laboratório e suporte técnico da equipe responsável pelo seu desenvolvimento.

2.7 Comentários

Nesse capítulo foram apresentados os conceitos básicos da síntese de protocolos de comunicação e da metodologia de projeto denominada *HW/SW codesign*.

Foi ressaltada a importância de uma correta escolha de uma partição *HW/SW* e, particularmente, sua influência no desempenho dos protocolos. Basicamente, a partição consiste em detalhar uma descrição funcional em vários módulos, definindo quais módulos devem ser implementados em *software* e quais módulos devem ser implementados em *hardware*.

O algoritmo de partição é outra característica importante e a maioria deles é baseada em heurísticas porque o problema de partição HW/SW é um problema NP-completo. Foram apresentados vários sistemas de partição HW/SW que existem atualmente e que diferem na capacidade de encontrar a solução ótima e no custo computacional. Também foi mostrado como o processo de *HW/SW codesign* pode ser formulado como um problema de otimização.

Foi ressaltada a importância do desenvolvimento de um processo de partição HW/SW que forneça dados concretos ao projetista, tornando o processo de decisão objetivo e mais rápido que os processos manuais de tentativa e erro.

Em seguida foram apresentadas algumas técnicas de descrição formal normalmente utilizadas no projeto de sistemas concorrentes. Em particular, a ferramenta ARP, que é utilizada neste trabalho para a verificação das redes de Petri estocásticas utilizadas na especificação de protocolos.

Algumas ferramentas de análise de desempenho de diversos tipos de sistemas também foram apresentadas com o objetivo de mostrar como elas podem ser utilizadas na fase de projeto de um protocolo e não somente quando o mesmo já se encontra implementado, o que aumenta o custo do projeto caso o desempenho não seja o desejado. Foi apresentada a ferramenta Tangram-II e a justificativa para a escolha dessa ferramenta na metodologia de projeto. A sintaxe dessa ferramenta é descrita com mais detalhes no capítulo 4, assim como o algoritmo utilizado para a determinação dos parâmetros necessários para a realização da partição.

Portanto, esse capítulo apresentou a motivação para a necessidade do desenvolvimento de uma metodologia de projeto de protocolos que auxilie o projetista no processo de partição HW/SW e a justificativa para algumas ferramentas utilizadas.

No capítulo a seguir o problema é formalizado e apresenta-se uma metodologia para o processo de partição.

Capítulo 3

Partição HW/SW de Protocolos

3.1 Introdução

A determinação da partição HW/SW ótima a ser utilizada no projeto de um protocolo de comunicação tem sido realizada manualmente pelo projetista, que se baseia para isso na própria experiência. Esse fator torna essa decisão bastante subjetiva e sujeita a erros. Por questões de custo, muitos algoritmos de partição [72] consideram que o projetista deseja implementar um protocolo com o menor número possível de componentes de *hardware* dedicado, capaz de atender a um determinado desempenho. Porém o custo desse tipo de *hardware* tem baixado continuamente. Nesse contexto, a maior necessidade do projetista passa a ser dispor de uma metodologia que considere medidas de desempenho durante o projeto do protocolo, de maneira a atender aos requisitos de velocidade que as redes atuais exigem.

A seção 3.2 apresenta alguns conceitos sobre a utilização de medidas de desempenho no projeto de um protocolo. Em seguida, na seção 3.3, descreve-se detalhadamente a metodologia de auxílio ao projetista para o processo de partição HW/SW. Em particular, discute-se a função objetivo utilizada no processo de otimização dos parâmetros do modelo do protocolo. A seção 3.4 mostra uma aplicação simples, que tem como objetivo principal apresentar a metodologia de forma didática, além de in-

tegrar as diversas ferramentas utilizadas. A seção 3.5 apresenta alguns comentários sobre a metodologia apresentada.

3.2 Avaliação de Desempenho de Protocolos

O desempenho de uma rede está fortemente relacionado com o protocolo utilizado e com o tempo de processamento, em cada nó, dos pacotes que transitam pela rede. Essa situação se torna mais crítica em redes de alta velocidade. Portanto, no projeto de um protocolo, devem ser considerados alguns aspectos que influenciam diretamente o seu desempenho e, consequentemente, o desempenho da rede como um todo. Entre os fatores que influenciam o desempenho de um protocolo pode-se citar, por exemplo, o *timeout*. Em determinados protocolos, quando um pacote é enviado um temporizador é disparado, para que o pacote seja reenviado caso uma confirmação de recebimento (ACK) não seja recebida dentro de um determinado período de tempo. Se, em protocolos como o **ARQ**, o *timeout* for ajustado com um valor muito pequeno, serão realizadas várias retransmissões desnecessárias. Caso ele seja ajustado com um valor muito alto ocorrem atrasos muito grandes até a detecção da perda e retransmissão do pacote. A ocorrência de *timeouts* normalmente degrada o desempenho da rede em relação à vazão, portanto o protocolo deve ser projetado de forma a evitá-los.

Outros aspectos importantes relacionados ao desempenho de um protocolo incluem a largura de faixa consumida, o número de pacotes perdidos e o seu comportamento quando uma perda é detectada. Esse comportamento determina o número de retransmissões que o emissor terá que realizar. Algumas vezes ocorre apenas uma inversão na ordem de entrega desses pacotes no receptor, não sendo necessária a retransmissão. Alguns protocolos conseguem distinguir se ocorreu uma perda ou se apenas houve uma inversão dos números de sequência dos pacotes [73][74][75].

A probabilidade de perda de pacotes é outra medida de desempenho importante, porém está mais relacionada com a rede em si e com o tamanho dos *buffers* em

cada nó da rede do que com o protocolo usado. O retardo, outra medida que influencia de forma significativa o desempenho de um protocolo, está relacionado com o *software*. Por isso é importante que a CPU utilizada seja veloz. No entanto, como o tratamento de cada pacote gasta um tempo adicional relativo à interrupção gerada na CPU, deve-se reduzir tanto quanto possível o número de transmissões, concentrando em cada uma a maior quantidade de informação possível. Dessa forma, diminui-se o número de interrupções e, consequentemente, o retardo do protocolo. Pelo mesmo motivo deve-se reduzir o número de nós intermediários de uma rede.

Certos protocolos baseiam-se no controle do tamanho da janela do emissor e do receptor para determinar a quantidade de pacotes colocados na rede, influenciando diretamente a sua vazão. O RTT (*ROUND TRIP TIME*), ou seja, o tempo decorrido entre o emissor enviar um pacote e receber o seu ACK, é um fator importante quando associado à largura de faixa (BW) disponível. O produto $BW * RTT$ fornece uma medida da quantidade de tráfego (em bits) que deve transitar na rede para que a eficiência máxima seja atingida.

Como já foi mencionado, nas redes de alta velocidade o desempenho de um protocolo é muito determinante. Existem modelos utilizados para a análise do desempenho dessas redes, *markovianos* e *não-markovianos*. Entre os modelos de fontes mais utilizados pode-se citar: modelo de fluidos, MMPP (*Markov-Modulated Poisson Process*) [76], processos duplamente estocásticos [76], modelos auto-similares [77][78], entre outros. Na metodologia apresentada neste trabalho utiliza-se um modelo *markoviano*, uma rede de Petri estocástica, para especificar o comportamento do protocolo a ser modelado e, posteriormente, particionado.

A otimização realizada neste trabalho determina os parâmetros do protocolo que atendem ao desempenho especificado pelas medidas de QoS desejadas para a rede e ao custo especificado para a implementação. Uma vez determinados os parâmetros ótimos, neste caso representados pelas taxas da rede de Petri especificada, calcula-se o produto $\lambda_i * \pi$, definido como "vazão" da transição, onde λ_i é a taxa determinada no processo de otimização e π é a soma das probabilidades em estado estacionário dos estados nos quais a transição i está habilitada. Note que, em geral, considera-se

vazão uma medida de desempenho do protocolo. Nesse caso, a medida foi estendida às transições, aumentando sua granularidade.

O processo de partição considera que as transições com vazão elevada têm maior probabilidade de serem implementadas em *hardware*. A decisão final é tomada em função das medidas de atraso e área dos circuitos sintetizados, fornecidas pelas ferramentas Synopsys e Altera. Este trabalho utiliza os parâmetros determinados pelo método de otimização na formulação de um critério de partição HW/SW de um protocolo. O objetivo é que ao final da implementação do protocolo o desempenho atenda a uma determinada especificação em relação às medidas de QoS desejadas para a rede e que o custo da implementação esteja dentro dos limites desejados. A especificação desejada é relacionada aos parâmetros do protocolo através de uma função de erro, discutida na seção 3.3.3. Essa função é então minimizada para a determinação dos parâmetros do protocolo a partir dos quais pode-se estabeler um critério objetivo para a partição HW/SW.

3.3 Metodologia de Auxílio à Partição HW/SW

Nas metodologias de partição HW/SW mencionadas nas seções 1.2 e 2.3.4, assim como nas ferramentas descritas na seção 2.3.2, não existe um procedimento padrão que auxilie no processo de *codesign*. A melhor partição é escolhida manualmente e depois de serem investigadas muitas alternativas, tornando o processo demorado e custoso. Todas as metodologias dependem muito da experiência do projetista. Mesmo em Alomary [51], onde já existe uma proposta de automatização, o desempenho do sistema só é determinado após a escolha da partição.

Em Hidalgo [14] é proposta uma metodologia para a partição HW/SW que também utiliza algoritmos genéticos, assim como na metodologia desenvolvida aqui, mas a função objetivo utiliza uma tabela previamente estabelecida para os valores dos desempenhos de *hardware* e de *software*, não calculando os parâmetros de desempenho durante o processo, isto é, esses parâmetros não são considerados no processo

de otimização, ao contrário da metodologia apresentada neste trabalho.

A metodologia aqui apresentada congrega uma série de técnicas em um único processo e propõe uma solução que auxilia o projetista na escolha da melhor partição HW/SW, de maneira a atender às medidas de desempenho e custo estabelecidas na especificação. Essas técnicas são as redes de Petri, para a especificação do protocolo em máquina de estados, uma ferramenta para a verificação e correção dessa especificação, uma ferramenta de modelagem e análise de desempenho, um algoritmo genético (AG) para a otimização de uma função objetivo e ferramentas de síntese de *hardware*. Cada partição HW/SW é definida em função da otimização realizada pelo algoritmo genético e dos dados fornecidos pelas ferramentas de síntese de *hardware*.

Inicialmente, o projetista especifica os valores de desempenho e de custo do *hardware* desejados. O custo do *software*, em princípio, é considerado baixo. Nessa fase, o protocolo é descrito por um modelo de máquina de estados, onde cada transição de estado representa uma operação ou um conjunto de operações a serem avaliadas. A partir dessa descrição, o fluxo de projeto segue dois caminhos paralelos. O primeiro caminho é responsável pela determinação dos parâmetros que atendem aos requisitos de desempenho do protocolo, através do uso de uma ferramenta de análise de desempenho e de um AG. O segundo, desenvolvido por Lima [37], é responsável pela extração das medidas relativas ao custo, representado pelo atraso e pela área de silício de cada circuito sintetizado. Essas medidas são obtidas através da análise das implementações em HW de todas as transições do protocolo, tanto em *standard cells*(SCs) quanto em PLDs. Como resultado, obtém-se uma tabela com a área do circuito sintetizado e seu respectivo atraso, para cada uma das transições do protocolo, no caso de implementação em SCs, e somente dos atrasos, no caso de implementação em PLDs. A figura 3.1 apresenta o diagrama de blocos da metodologia.

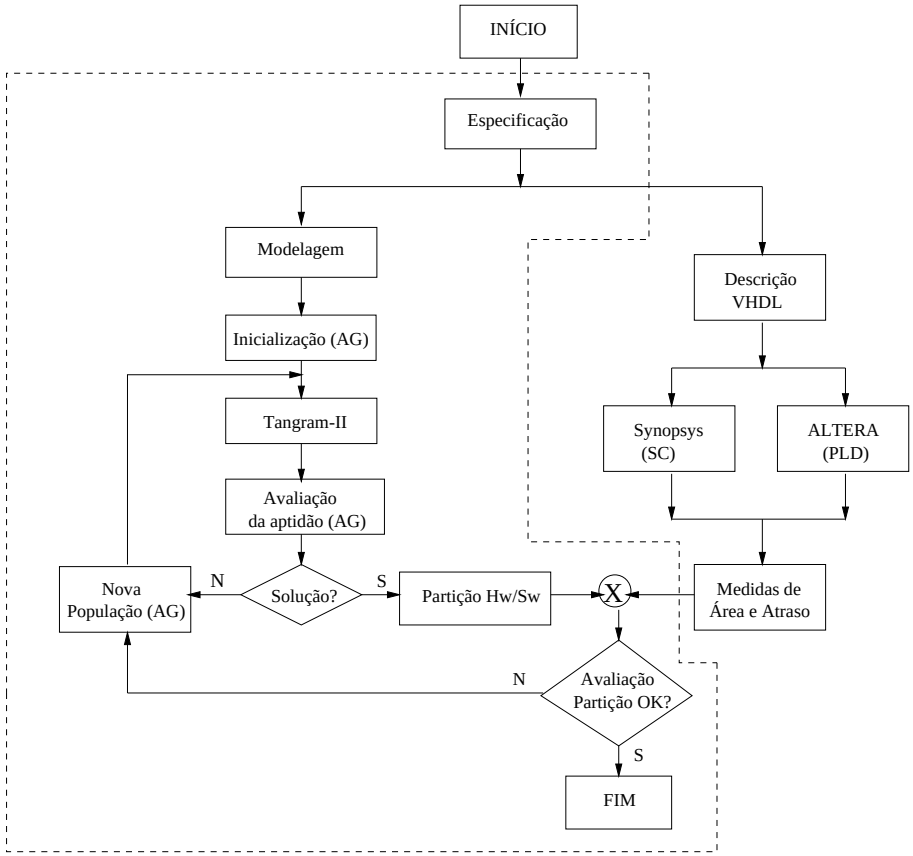

Figura 3.1: Metodologia de projeto.

3.3.1 Descrição Inicial e Verificação Formal

Neste trabalho os protocolos são descritos por redes de Petri estocásticas (*Stochastic Petri Nets* - **SPNs**), que são uma extensão do modelo de rede de Petri tradicional. Nas SPNs o tempo de disparo de uma transição, após a mesma ser habilitada, possui distribuição exponencial com taxa λ_i, onde λ_i é a taxa associada à transição i do modelo. Em Molloy [79] é mostrado que as SPNs são equivalentes a uma cadeia de Markov a parâmetro contínuo finita (CMTC) [80], onde cada *marca* da rede de Petri corresponde a um estado da cadeia e as taxas da SPN correspondem às taxas de transição entre os estados da cadeia. A disponibilidade de ferramentas para a verificação formal da especificação determinou a opção pelas SPNs para descrever o comportamento protocolo.

Uma vez descrito o comportamento do protocolo, verifica-se a correção dessa descrição através da aplicação da ferramenta **ARP** (Analisador de Redes de Petri), que realiza uma análise sintática e semântica. A análise sintática verifica se a rede é *viva*, *limitada* e *reiniciável*. A análise semântica é realizada através da verificação de propriedades específicas, como *invariantes de transição* e *invariantes de lugar*. Dessa forma, pode-se verificar se a descrição do protocolo está livre de *deadlocks* e *livelocks*.

3.3.2 Construção do Modelo

Uma vez verificada a correção da descrição inicial através da ferramenta ARP, é construído um modelo na linguagem da ferramenta de análise de desempenho Tangram-II, utilizando-se uma interface gráfica, o TGIF (Tangram Graphical Interface) [81]. Cada transição da rede de Petri é associada a um *evento* e cada *lugar* é representado por uma *variável de estado* no modelo do Tangram-II. As variáveis de estado auxiliam na implementação do comportamento do protocolo, por exemplo, elas podem corresponder ao número de *fichas* em cada lugar numa determinada *marca*. A *condição* para que um *evento* seja habilitado é similar à condição de disparo de uma transição da SPN, mas também pode depender de outras variáveis de estado, no caso de protocolos mais complexos. Uma *condição* está associada a uma expressão, semelhante à linguagem de programação C. Se a expressão é verdadeira, o *evento* é habilitado e disparado após um intervalo de tempo com distribuição exponencial. O disparo do *evento* provoca a execução de uma ou mais *ações* que descrevem o comportamento do protocolo. No capítulo 4 são fornecidos maiores detalhes sobre a sintaxe da ferramenta Tangram-II.

Os parâmetros do modelo construído no Tangram-II são as taxas dos *eventos*. Como cada transição da SPN, quando habilitada, possui um tempo de disparo com distribuição exponencial, os *eventos* também têm distribuição exponencial e, portanto, cada modelo tem uma Cadeia de Markov a Tempo Contínuo (CMTC) associada, gerada pela ferramenta Tangram-II. Deve ser observado que a cadeia de Markov gerada é uma cadeia **literal** que somente é resolvida quando são atribuídos valo-

res numéricos aos parâmetros, uma vez que os métodos de solução disponíveis na ferramenta Tangram-II não são literais.

3.3.3 Construção da Função Objetivo

De uma forma geral, a *função objetivo* ou função de erro do problema de partição HW/SW pode ser dada pela diferença entre os valores das medidas de desempenho desejadas, especificadas pelo projetista, e os valores das medidas de desempenho obtidas para um dado conjunto de parâmetros. Como os parâmetros especificados normalmente "competem entre si", é útil atribuir-se pesos a cada um dos parâmetros que compõem a função objetivo para equalizar as possíveis diferenças de sensibilidade entre eles [14]. Essa função fornece uma medida da qualidade de uma determinada solução e através de sua avaliação pode-se selecionar aquela que satisfaz melhor os requisitos de desempenho. A forma geral da função objetivo é a seguinte:

$$\varepsilon(\lambda) = \sum_{i=1}^{n} w_i * |f_{d_i}(\lambda) - f_{o_i}(\lambda)| \quad \mathbb{R}^n \rightarrow \mathbb{R}^1 \tag{3.1}$$

onde:

$\lambda = (\lambda_1, ..., \lambda_n)$ - é o vetor de parâmetros da otimização;

f_{d_i} - é a i-ésima medida de desempenho desejada;

f_{o_i} - é a i-ésima medida de desempenho obtida;

w_i - são os fatores de ponderação ou normalização;

n - número de parâmetros de desempenho a serem otimizados;

A função objetivo utilizada neste trabalho é definida a partir de medidas de desempenho do protocolo, tais como vazão, retardo, probabilidade de perda, entre outras, obtidas a partir do Tangram-II. No entanto, foi utilizada apenas uma medida

de desempenho em cada uma das otimizações realizadas nos exemplos apresentados e, consequentemente, o peso é igual a 1. As medidas de desempenho utilizadas dependem dos requisitos de qualidade de serviço desejados. O problema de otimização consiste em determinar λ^* tal que:

$$\varepsilon(\lambda^*) = \min_{\lambda} \varepsilon(\lambda) \tag{3.2}$$

3.3.4 Otimização da Função Objetivo

O problema de otimização inicial consiste em determinar as taxas de transição de estado da CMTC, isto é, as taxas dos *eventos* do modelo, que satisfazem restrições de desempenho impostas pelo projetista. Com base nessas restrições, o melhor conjunto de parâmetros é determinado pelo algoritmo genético. A determinação dessas taxas fornece subsídios para orientar a escolha de um critério de partição.

A cadeia de Markov gerada pelo Tangram-II só pode ser resolvida quando os parâmetros forem substituídos por valores numéricos. Esses valores são fornecidos por um método de otimização baseado em AGs que, inicialmente, gera conjuntos de valores para os parâmetros de forma aleatória, porém obedecendo certas restrições de projeto ou de análise de sensibilidades (vide apêndice A). Após a substituição dos parâmetros pelos valores numéricos, a cadeia de Markov é resolvida através de algum dos métodos de solução analítica que a ferramenta Tangram-II oferece. Dessa forma são calculadas as probabilidades em estado estacionário de cada um dos estados da CMTC. Essas probabilidades são utilizadas para o cálculo de medidas de interesse relacionadas com o desempenho do protocolo. Essas medidas são os *valores obtidos*.

A especificação a ser atendida pode ser dada, por exemplo, por uma curva relativa a uma determinada medida de desempenho e por valores máximos para a área de silício e atraso do componente de *hardware* a ser sintetizado. Essa curva fornece os *valores desejados* da função objetivo. De posse dos valores desejados e dos valores obtidos pode-se calcular o valor da função objetivo para cada conjunto de taxas gerado pelo AG. Cada conjunto é denominado um *indivíduo* e o número de indiví-

duos corresponde ao tamanho da *população* gerada pelo AG. O projetista também especifica o número de pontos da curva que ele deseja aproximar. O valor da função objetivo, dado pela soma dos erros entre os valores desejados e os valores obtidos, fornece uma medida da *aptidão* do indivíduo para resolver o problema.

O processo de otimização é iniciado calculando-se a *aptidão* de cada *indivíduo* da população e verificando-se se algum desses *indivíduos* representa a solução do problema. Caso a solução não tenha sido encontrada, realiza-se uma seleção dos *indivíduos*, com maior probabilidade para os *indivíduos* mais aptos, faz-se o cruzamento de cada um com seu *parceiro* e gera-se uma nova população. O Tangram-II é realimentado com a nova *geração* de indivíduos e repete-se o processo até que se encontre um conjunto de taxas que atenda a um erro máximo especificado pelo projetista ou se alcance um determinado número de gerações.

3.3.5 Critério de partição

Cada transição da SPN está relacionada a um conjunto de operações do protocolo a serem implementadas em *hardware* ou em *software*. O conjunto de taxas calculadas pelo AG que atende às medidas de desempenho especificadas deve determinar uma partição HW/SW possível para o protocolo. Com essa finalidade é estabelecido um critério para relacionar um conjunto de taxas com uma partição.

A taxa de um determinado *evento* do modelo está relacionada ao número de vezes, por segundo, que um determinado conjunto de operações do protocolo é executado. Isso poderia levar à uma conclusão (precipitada) de que essas operações são as que devem ser implementadas em *hardware*. No entanto, deve-se considerar as probabilidades em estado estacionário de cada estado do sistema no momento de decidir sobre uma partição HW/SW. Isso significa que um *evento* pode ter uma alta taxa de disparo, mas a probabilidade do sistema estar num estado em que o mesmo está habilitado pode ser muito baixa.

A partir dessas considerações, foi estabelecido como critério de partição HW/SW

dos *eventos* o produto $\lambda_i * \pi$ onde:

λ_i - taxa de disparo do *evento* i calculada pelo AG.

π - soma das probabilidades em estado estacionário dos estados nos quais o *evento* i está habilitado.

Cada *evento* tem uma "vazão" associada a ele. Dessa forma, as operações do protocolo relacionadas com os eventos de maior "vazão" são implementadas em *hardware* e aquelas associadas aos eventos de menor "vazão" são implementadas em *software*. Esse critério possibilita relacionar as taxas calculadas pelo AG a uma partição HW/SW. Essa é a primeira partição a ser avaliada em relação ao custo do *hardware*.

3.3.6 Implementação em Hardware

A partição HW/SW obtida é avaliada a partir dos dados de área de silício e atraso dos circuitos sintetizados em Lima [37]. Esses dados são fornecidos pelas ferramentas Synopsys e Altera.

A especificação do protocolo em máquina de estados serve como referência para a entrada das descrições *comportamentais* em VHDL. Cada transição de estado (ou *evento*, no Tangram-II) é sintetizada de modo a se obter as medidas de atraso e área de cada uma delas.

A ferramenta Synopsys sintetiza um circuito lógico a partir da descrição *comportamental* de cada transição em VHDL. Essa implementação é realizada utilizando-se uma biblioteca de células padrão (*standard cells* - SCs) numa dada tecnologia. As etapas de alocação, roteamento de células e otimização são automáticas e se tornam transparentes para o projetista.

PLDs são circuitos integrados digitais que podem ser programados a fim de implementar funções booleanas definidas através de programação. A ferramenta Altera implementa as funções lógicas de cada transição da SPN a partir de suas

descrições em VHDL.

Ao final do processo de síntese, obtém-se uma tabela contendo todas as medidas de atraso e área para cada uma das transições do protocolo. Essas medidas são utilizadas para avaliar se a partição HW/SW atende à especificação de área do *hardware*, isto é, se as áreas dos circuitos equivalentes às transições escolhidas para serem implementadas em *hardware* obedecem aos limites especificados e se os respectivos atrasos atendem às taxas calculadas pelo AG. Se a avaliação atender aos requisitos, as referidas transições são implementadas em SCs e o processo se encerra. Se alguma área exceder o limite especificado, verifica-se se a implementação em PLD atende ao atraso especificado. Nesse caso, essas transições são implementadas em PLDs. Se o atraso não for atendido, uma nova população é gerada e uma nova otimização realizada. O processo de síntese em *hardware* do protocolo é parte de um trabalho complementar a esse, desenvolvido por Lima [37], onde podem ser obtidos maiores detalhes.

3.4 Exemplo - Protocolo bit alternante

O protocolo bit alternante é simples de ser modelado, Molloy [79] fornece a curva da vazão em função da taxa de novas mensagens e o objetivo principal desse exemplo é apresentar a metodologia com um exemplo prático, simples e didático.

3.4.1 Metodologia Passo a Passo

O protocolo bit alternante, descrito em Molloy [79], é especificado por uma rede de Petri, como é ilustrado na figura 3.2 e, posteriormente, modelado na ferramenta Tangram-II. O modelo construído na ferramenta Tangram-II é omitido. O apêndice B apresenta um exemplo de um modelo do Tangram-II para o protocolo ARM [82], que é estudado na seção 5.4.

A correção da rede de Petri é verificada utilizando-se a ferramenta ARP. A correção do modelo do Tangram-II, nesse caso, pode ser realizada através dos valores das taxas dos eventos, mostradas na tabela 3.1, utilizadas em Molloy [79]. A cadeia de Markov gerada pelo Tangram-II é resolvida para diversos valores de taxa de disparo da transição NEW_MSG (chegada de novas mensagens), que indica a *carga* do protocolo. Especificamente nesse caso, o produto dessa taxa pela probabilidade do sistema estar em um estado no qual a transição associada a ela está habilitada fornece a vazão do protocolo. Esse processo permite que seja encontrada a curva vazão versus carga para o modelo do Tangram-II. Obteve-se a curva da figura 3.3, que é a mesma curva obtida por Molloy [79].

Verificada a correção do modelo do Tangram-II, implementa-se o algoritmo genético. Nessa etapa, as taxas das transições da rede de Petri são parâmetros ou, analogamente, o modelo do Tangram-II é um modelo com parâmetros. Os valores aleatórios gerados para cada taxa pelo AG podem ter que atender a restrições de projeto, que dependem do protocolo que está sendo modelado. Nesse exemplo, a restrição estipulada é que a taxa de perda de mensagens e de ACKs seja igual a 5% da taxa de envio de mensagens e de ACKs, respectivamente. Além disso, a taxa de disparo da transição **Timeout** foi fixada em aproximadamente um quarto da taxa de perda para que o *timeout* não seja disparado sem necessidade, diminuindo a vazão do protocolo.

Transição	**Taxa (1/seg.)**
Send, Send Ack	9,375
Msg Drop, Ack Drop	3,91
Crc ok, Ack ok	74,22
Timeout	1,00

Tabela 3.1: Taxas originais da Rede de Petri.

Como a curva da figura 3.3 é monotonamente crescente, 10 pontos são suficientes para aproximá-la e obter os parâmetros desejados. Foram realizados experimentos com uma população de até 5000 indivíduos, 10 gerações e o valor 0,001 para a soma

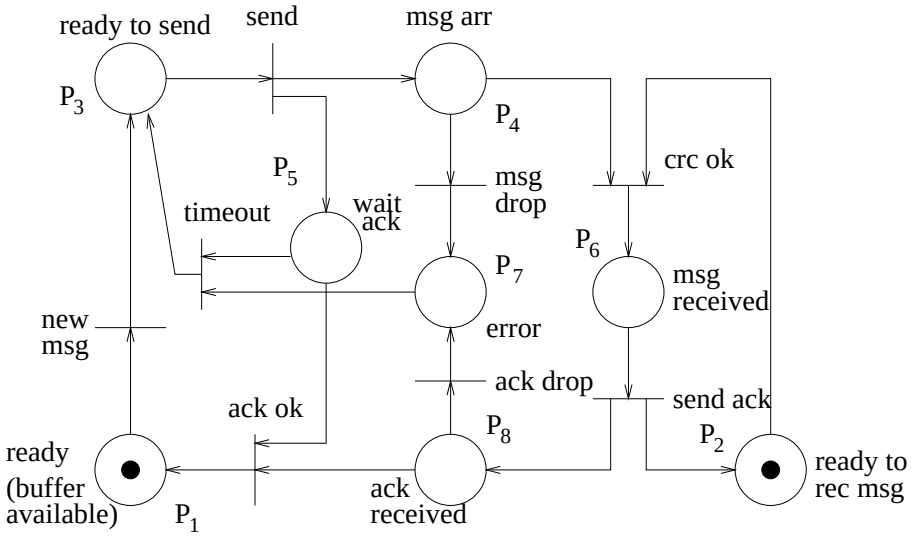

Figura 3.2: Rede de Petri do protocolo bit alternante.

total dos erros dos 10 valores de vazão desejados. Posteriormente, verificou-se que devido à simplicidade da curva a ser aproximada, uma população com 100 indivíduos é suficiente para o algoritmo convergir para os valores de vazão desejados. A função objetivo é a soma das diferenças entre as vazões obtidas e as vazões desejadas, que são dadas pelos pontos escolhidos. As transições λ_3, λ_4 e λ_5 (*timeout*, perda de ACK e perda de mensagem, respectivamente) são fixadas de acordo com o valor das outras transições de forma a atender as restrições estabelecidas. Os resultados obtidos são apresentados na tabela 3.2.

Transição	**Taxa (1/seg.)**
Send (λ_1)	25,9
Send Ack (λ_7)	30,0
Msg Drop (λ_5)	1,8
Ack Drop (λ_4)	1,5
Ack OK (λ_2)	27,1
CRC OK (λ_6)	36,5
Timeout (λ_3)	0,5

Tabela 3.2: Taxas da Rede de Petri do protocolo bit alternante obtidas pelo AG.

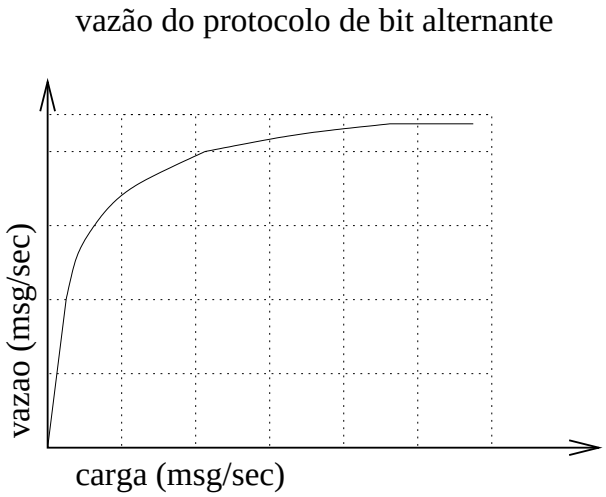

Figura 3.3: Curva vazão versus carga do protocolo bit alternante.

Utilizando as taxas da tabela 3.2, aplica-se o critério de partição descrito na seção 3.3.5 e calcula-se o produto $\lambda_i * \pi_j$ para cada transição do protocolo, onde λ_i é cada uma das taxas de transição da rede de Petri (vide tabela 3.2), calculadas pelo AG, e π_j é a soma das probabilidades em estado estacionário dos estados da cadeia de Markov onde a transição i está habilitada. Os resultados obtidos são apresentados na tabela 3.3:

Evento	**Produto**
$\lambda_1 * \pi_2$	2,46
$\lambda_7 * \pi_5$	2,34
$\lambda_2 * \pi_6$	2,22
$\lambda_6 * \pi_3$	2,34
$\lambda_4 * \pi_6$	0,11
$\lambda_5 * \pi_3$	0,12
$\lambda_3 * \pi_4$	0,21

Tabela 3.3: Valores obtidos para o produto $\lambda_i * \pi_j$.

Os estados possíveis do protocolo, uma *marca* da rede de Petri, são dados abaixo:

E_1 - 1 ficha em P1 e outra em P2;

E_2 - 1 ficha em P2 e outra em P3;

E_3 - 1 ficha em P2, outra em P4 e outra em P5;

E_4 - 1 ficha em P5 e outra em P6;

E_5 - 1 ficha em P2, outra em P5 e outra em P7;

E_6 - 1 ficha em P2, outra em P5 e outra em P8;

Observando-se a tabela acima, verifica-se que a "vazão" da transição 3 (Timeout) é aproximadamente igual à soma das "vazões" das transições 4 e 5, que correspondem a perdas. Esse resultado já era esperado, pois pode-se observar na rede de Petri da figura 3.2 que o *timeout* é habilitado sempre que ocorre uma perda. Verifica-se também que a soma das "vazões" das transições 4 e 5 é aproximadamente igual a 5% da soma das "vazões" das transições 1, 2, 6, 7, que são transições de transmissão e recepção de mensagens. Dessa forma, a probabilidade de erro especificada em Molloy [79] é atendida.

Aplicando-se o critério de partição descrito na seção 3.3.5, as operações do protocolo correspondentes às transições 1, 2, 6 e 7 devem ser implementadas em *hardware* enquanto que aquelas correspondentes às transições 3, 4 e 5 podem ser implementadas em *software*. Como as tarefas de recepção de mensagens costumam consumir mais tempo que as tarefas para enviar uma mensagem, realiza-se uma partição inicial colocando-se as transições 2 e 6 (recepção) em *hardware* e as transições 1 e 7 em *software*. Essa é a primeira partição a ser avaliada em relação ao custo do *hardware*, em termos da área de silício e atraso dos circuitos integrados sintetizados pelas ferramentas Synopsys e Altera. Caso o custo do HW esteja dentro das especificações e as taxas encontradas possam ser atendidas pelos circuitos sintetizados, uma solução foi encontrada. Caso contrário, uma nova população é gerada para que o processo de otimização continue.

3.4.2 Avaliação da Partição

As descrições *comportamentais* em VHDL de cada transição de estado do protocolo são utilizadas como entrada para a ferramenta Synopsys. A síntese lógica dos circuitos no Synopsys foi realizada utilizando-se uma biblioteca de células padrão (*standard cells*) ES2 com tecnologia 0,7μm. A tabela 3.4 mostra as medidas de atraso e área obtidas para cada uma das transições de estado do protocolo. Os valores de área são fornecidos automaticamente pelo Synopsys.

Transição	**Standard Cells Atraso**	**Standard Cells Área (mm^2)**
Send (λ_1)	44 ns	0,202
Ack OK (λ_2)	4,7 ns	0,017
Timeout (λ_3)	5 ns	0,015
Ack Drop (λ_4)	4,7 ns	0,015
Msg Drop (λ_5)	4,7 ns	0,015
CRC OK (λ_6)	4,7 ns	0,017
Send Ack (λ_7)	44 ns	0,202

Tabela 3.4: Medidas de atraso e área das transições do protocolo.

Tendo em vista os resultados apresentados na tabela 3.4, a partição proposta é avaliada. Levando-se em conta, por exemplo, um custo de área especificado pelo projetista de 1 mm^2, pode-se notar que as transições indicadas pelo processo de particionamento para implementação em *hardware* (t_2 e t_6) podem ser implementadas utilizando-se *standard cells* pois todas as medidas de área são menores do que 1 mm^2 e os atrasos associados satisfazem as taxas calculadas pelo AG. Caso ficasse constatado que alguma transição excedeu o limite de área estipulado pelo projetista, a síntese com PLDs deveria ser realizada para implementar as tarefas dessas transições.

3.4.3 Análise do Desempenho da Partição Escolhida

Uma vez determinada a partição HW/SW, utiliza-se os atrasos fornecidos pela ferramenta Synopsys para se analisar o desempenho da partição escolhida. Calcula-se a taxa da i-ésima transição (λ_i) invertendo-se o atraso, ou seja, $\lambda_i = 1/A_i$, onde A_i é o atraso dado pelo Synopsys no caso da transição i ser implementada em *standard cells* ou o atraso dado pelo Altera, caso a transição fosse implementada com PLDs, ou o atraso do SW, caso a transição fosse implementada em SW. A ferramenta Tangram-II utiliza essas taxas como entrada para obter o desempenho final da partição (nesse caso, representado pela vazão do protocolo).

A figura 3.4 compara as vazões do protocolo para o caso onde o a implementação é totalmente realizada em SW, totalmente realizada em HW e utilizando-se a partição HW/SW escolhida.

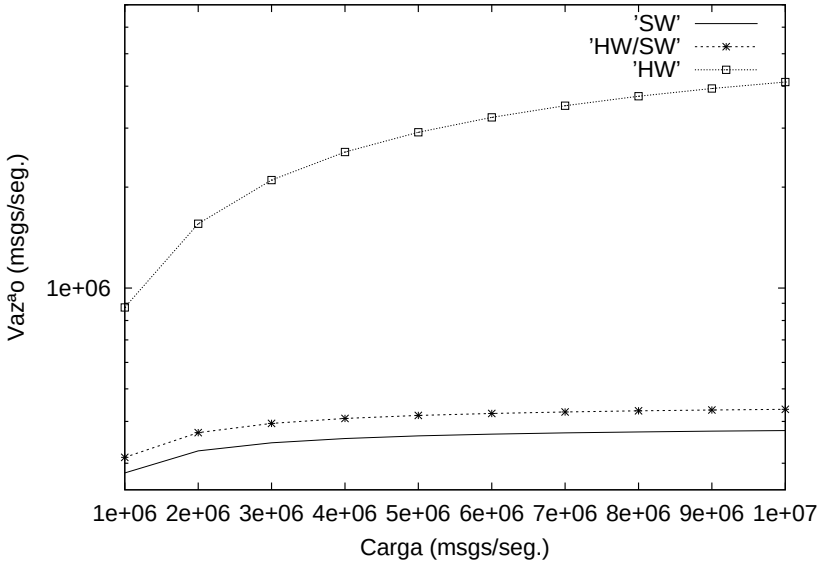

Figura 3.4: Comparação entre as vazões para as diferentes implementações.

As taxas de perda foram fixadas de maneira que a perda fosse de 5% para os três casos e a taxa do *timeout* fixada em função da perda. Pode-se perceber que o desempenho da implementação em *hardware* é cerca de três vezes maior que o desempenho da implementação em *software* quando a carga é baixa, chegando a ser 10 vezes maior para altas cargas. A melhora do desempenho da partição HW/SW em

relação à implementação em *software* é apenas de 11%, indicando que esse protocolo não é um bom candidato para se aplicar a metodologia descrita.

Devido à simplicidade do protocolo, esse exemplo teve como finalidade principal apresentar a metodologia de forma prática e didática. No capítulo 5 protocolos mais complexos serão implementados e será possível avaliar melhor o potencial da metodologia, assim como para quais tipos de protocolos a metodologia de projeto é mais adequada.

3.5 Comentários

Neste capítulo foram apresentados conceitos básicos relativos à utilização da análise de desempenho no projeto de um protocolo de comunicação. A metodologia de otimização e o critério de partição apresentados fornecem subsídios ao projetista para uma escolha objetiva da melhor partição HW/SW, mantendo a análise em alto nível, ou seja, sem a necessidade de um conhecimento detalhado no nível de circuitos. A metodologia entra imediatamente no ciclo do projeto, pois cada iteração do AG analisa uma solução possível para o problema. No capítulo a seguir são apresentadas as características e a sintaxe da ferramenta Tangram-II, além de conceitos básicos sobre algoritmos genéticos e o algoritmo utilizado na metodologia proposta.

Capítulo 4

Técnicas de Projeto

4.1 Introdução

A metodologia apresentada neste trabalho, congrega uma série de técnicas em um único processo que auxilia na escolha de uma partição HW/SW. O ponto de partida consiste na adoção de um projeto cooperativo, HW/SW *codesign*, discutido no capítulo 2. Este capítulo descreve a sintaxe da ferramenta Tangram-II, usada na modelagem do protocolo e no processo de avaliação da função objetivo para que o algoritmo genético possa escolher a melhor solução para o problema. Também são apresentados os conceitos básicos dos algoritmos genéticos, método de otimização utilizado no capítulo 3 e é discutido o algoritmo genético utilizado neste trabalho. As ferramentas de síntese de *hardware* Synopsys e Altera são descritas em Lima [37]. A seção 4.4 realiza alguns comentários sobre as técnicas descritas.

4.2 Sintaxe da ferramenta Tangram-II

Nesta seção é apresentada a sintaxe da linguagem utilizada para construção

de modelos na ferramenta Tangram-II. Para isto é necessário introduzir algumas definições básicas.

4.2.1 Definições

Eventos são gerados espontâneamente por um objeto. Caso as *condições* associadas àquele *evento* sejam satisfeitas, a *ação* relacionada a ele é executada.

Condições são expressões booleanas avaliadas a partir do estado corrente do objeto e usadas para indicar quando um *evento* está habilitado. Uma *condição* deve vir entre parênteses e pode ser composta por variáveis, valores numéricos e operadores aritméticos e lógicos.

Ações são usadas para modificar o estado de um objeto e para enviar *mensagens* a outros objetos. Uma *ação* é executada quando um *evento* é disparado ou uma *mensagem* é recebida. *Ações* são descritas em código C-like e podem usar variáveis e constantes declaradas no objeto. Uma *ação* também pode conter variáveis locais, declaradas dentro do escopo da *ação*. As variáveis de estado não podem ter o seu valor alterado durante uma *ação*. Somente ao final da *ação* é usada a função **set_st(<var_name>, <value>)** para alterar o valor da variável de estado. Várias *ações* podem ser associadas a um *evento* ou a uma *mensagem*, cada uma com uma probabilidade diferente de ocorrência (soma = 1). As probabilidades são especificadas usando-se a palavra-chave PROB seguida de uma expressão aritmética.

Exemplo de múltiplas *ações*:

```
action= {
/* chegou um pacote correto */
int q;
q = q + 1;
```

```
set_st("queue",q);
} : prob = p;
/* chegou um pacote incorreto , não faz nada */
{
;
} : prob = 1 - p;
```

Mensagens são usadas para representar a interação entre os objetos e são enviadas num tempo igual a zero. Quando um *evento* ocorre ou uma *mensagem* é recebida, o conjunto de *ações* especificadas pelo usuário é executado com uma dada distribuição de probabilidade. Como resultado da execução de uma *ação*, o estado do objeto pode mudar e *mensagens* podem ser enviadas para outros objetos do modelo.

O Tangram-II possui diversos métodos de solução da cadeia de Markov gerada a partir da especificação do modelo. Existem também métodos de solução para cadeias com *eventos* determinísticos (modelos não-Markovianos). O simulador interativo possibilita a geração de *traces* de variáveis que o projetista deseja monitorar. Essa possibilidade é extremamente útil quando a inclusão de uma variável de estado no modelo provoca a explosão do espaço de estados, tornando a cadeia de Markov associada de difícil solução analítica. Pode-se também realizar a simulação de *eventos* raros e animação.

Os objetos são especificados utilizando-se uma representação gráfica. A interface gráfica TGIF mostra um *template* que auxilia o usuário na construção de cada objeto do modelo. Após a construção do modelo, gera-se a cadeia de Markov associada. O usuário, então, seleciona o método de solução a ser adotado para a solução da cadeia de Markov e obtenção das probabilidades em estado estacionário que serão utilizadas no cálculo das medidas de desempenho.

4.2.2 Construção de Modelos

Define-se um objeto no Tangram-II através da especificação de seis atributos:

- "Declaration attribute"

  Nesse atributo são declaradas as variáveis, constantes e os parâmetros utilizados no objeto. A declaração é realizada em três subseções diferentes, uma para cada tipo de declaração (variáveis, constantes e parâmetros).

- "Initialization attribute"

  Nesse atributo as constantes são definidas e as variáveis de estado recebem seus valores iniciais. Os valores dos parâmetros são fornecidos posteriormente, quando da solução analítica da cadeia de Markov associada ao modelo.

- "The State variables attribute"

  Esse atributo é utilizado pelo simulador. As variáveis de estado do modelo devem ser colocadas nessa seção, caso o simulador esteja sendo utilizado.

- "The Events attribute"

  Nesse atributo são declarados todos os *eventos*. Um *evento* é definido por um nome, uma distribuição de probabilidades e seus parâmetros. Um *evento* deve ter uma *condição* e pelo menos uma *ação* associada a essa *condição*.

- "The Messages attribute"

  Nesse atributo são especificadas todas as *mensagens* que são recebidas por um objeto. Quando uma *mensagem* é recebida, é realizada uma *ação* associada a ela. A recepção de uma *mensagem* é definida pela constante denominada "Message_port", que indica a porta a ser usada na recepção da *mensagem*.

- "The Rewards attribute"

  Nesse atributo são declaradas as *recompensas* de taxa, associadas com um determinado estado do sistema e as *recompensas* de impulso, associadas a uma

determinada transição de estado. *Recompensas* são acumuladas ao longo de uma simulação e servem para avaliar medidas difíceis de se obter. São utilizadas quando não se deseja criar uma variável de estado que faria o tamanho da cadeia de Markov crescer indevidamente, criando problemas para a sua solução analítica.

Na figura 4.1 é apresentado um exemplo da sintaxe usada no Tangram-II para a construção de um modelo de fila M/M/1/k, ou seja, um modelo cujo tempo entre chegadas de pacotes possui distribuição exponencial, o tempo de serviço de cada pacote possui distribuição exponencial, existe apenas um servidor e o tamanho da fila é limitado a k pacotes.

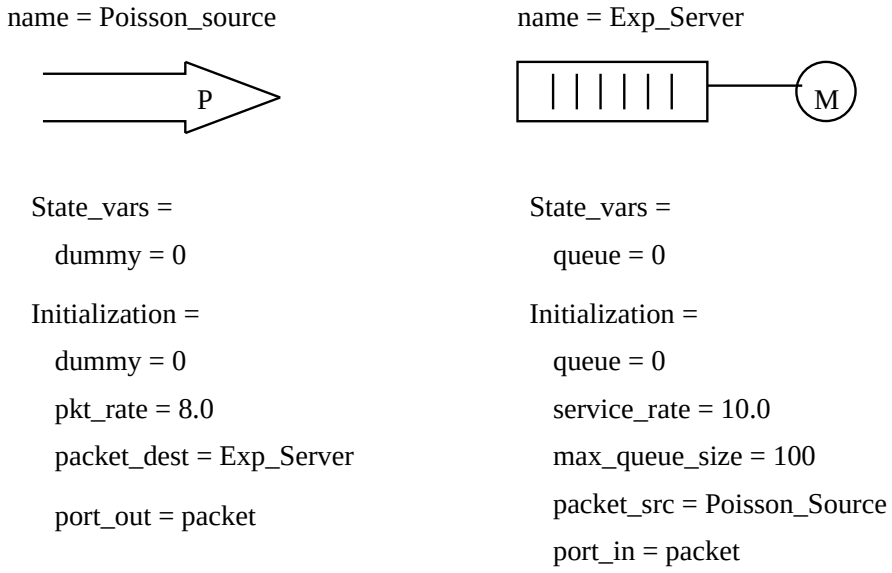

Figura 4.1: Modelo de fila M/M/1/k

Neste trabalho, o modelo construído no Tangram-II é um modelo parametrizado onde os parâmetros representam as taxas de transição do protocolo modelado. A cadeia de Markov associada ao modelo é gerada uma única vez. Como os métodos de solução do Tangram-II não são literais, é necessário fornecer valores numéricos ao Tangram-II para que seja possível resolver a cadeia de Markov. Esses valores numéricos são fornecidos por um algoritmo genético, como é apresentado na seção 4.3.5.

Maiores detalhes sobre a sintaxe da ferramenta Tangram-II podem ser obtidos em Silva [23].

4.3 Algoritmos Genéticos

Os algoritmos genéticos (AGs) vêm tendo larga aceitação devido à simplificação que eles introduzem na formulação e solução de problemas de otimização. Essa característica é particularmente útil em problemas de otimização complexos, NP-Completos, que envolvem um grande número de variáveis e, consequentemente, espaços de solução de dimensões elevadas, com múltiplos mínimos locais. Além disso, em muitos casos onde outras estratégias falham na busca de uma solução, os AGs convergem. Outra vantagem da utilização dos AGs em problemas de otimização é que, ao contrário dos métodos tradicionais, os AGs não utilizam derivadas na busca de uma solução do problema. Portanto, os AGs se aplicam bem a problemas que possuem funções não-diferenciáveis.

4.3.1 Conceitos Fundamentais

Algoritmos genéticos simples normalmente trabalham com descrições de entrada formadas por cadeias de bits de tamanho fixo, denominadas *cromossomos*, onde um ou mais parâmetros do problema de otimização são codificados, cada parâmetro correspondendo a um *gene*. Para *cromossomos* de tamanho variável, as implementações mais eficientes são baseadas na Programação Genética [35]. Os AGs possuem um paralelismo implícito decorrente da avaliação independente de cada uma dessas cadeias de bits. Existem três tipos de representação possíveis para os *cromossomos*: binária, inteira ou real. A essa representação se dá o nome de *alfabeto* do AG. Pode-se usar qualquer um dos três tipos, de acordo com a classe de problema que se deseja resolver. Neste trabalho, cada *cromossomo* é formado por um conjunto

de taxas dos *eventos* do modelo, onde o valor de cada taxa é representado por uma variável real. Consequentemente, utilizou-se um vetor de variáveis reais de dimensão m para representar o *cromossomo*, onde m é o número de parâmetros do modelo. Deve ser notado que cada *cromossomo*, chamado de *indivíduo*, corresponde a um ponto no espaço de soluções do problema de otimização. O processo de solução adotado nos AGs consiste em gerar, através de regras específicas, um grande número de *indivíduos*, uma *população*, de forma a promover uma varredura tão extensa quanto necessária do espaço de soluções.

A estrutura básica do AG é mostrada na figura 4.2. Pode-se identificar quatro operações básicas na figura: **avaliação** ou **cálculo da aptidão**, a **seleção** dos *indivíduos*, a **reprodução (ou cruzamento)** e a **mutação**. A *aptidão* de um *indivíduo* está relacionada diretamente com a capacidade daquele *indivíduo* em resolver o problema. Quanto mais próximo da solução ótima, maior é a sua *aptidão*. Neste trabalho, o cálculo da *aptidão* é realizado com base nos dados fornecidos pela ferramenta Tangram-II. A probabilidade de um *indivíduo* ser selecionado para reprodução é diretamente proporcional à sua *aptidão*. Na reprodução, os *indivíduos* selecionados são cruzados com seus *parceiros* e na mutação alguns *genes* são alterados aleatoriamente. Ao final de cada iteração cria-se uma nova população, chamada de *geração* que, espera-se, represente uma melhor aproximação da solução do problema de otimização que a população anterior. Essas operações são descritas detalhadamente na seção a seguir.

4.3.2 Operações básicas dos AGs

Com referência ao diagrama da figura 4.2, observa-se que cada iteração do AG corresponde à aplicação de um conjunto de quatro operações básicas: cálculo da *aptidão*, seleção, reprodução (ou cruzamento) e mutação.

A população inicial é gerada atribuindo-se aleatoriamente valores aos *genes* de

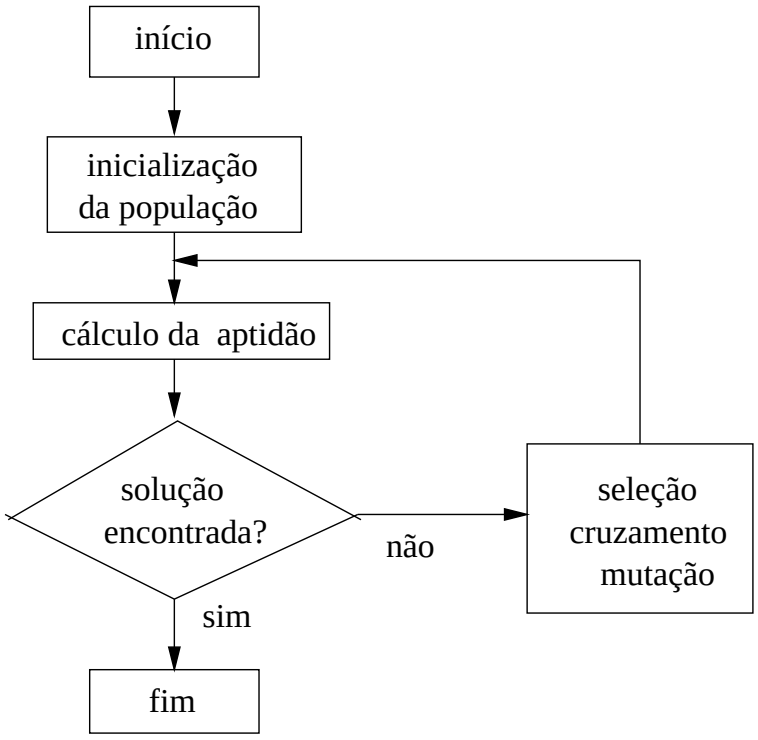

Figura 4.2: Diagrama de blocos de um algoritmo genético simples

cada *cromossomo*, considerando-se as restrições de domínio de cada gene. Para cada *indivíduo* da população é atribuída uma *aptidão* bruta a partir do resultado do cálculo de uma função de erro, também chamada de *função objetivo* do problema de otimização. A *aptidão* bruta é em seguida normalizada (*aptidão* normalizada) para permitir um melhor controle do processo de seleção. Em geral, a *aptidão* do melhor *indivíduo* em conjunto com a limitação do número de gerações são usados como critérios de terminação do algoritmo.

- Inicialização

  Uma população de n *indivíduos* é gerada aleatoriamente, onde cada um dos *indivíduos* representa um ponto do espaço de soluções.

- Cálculo da aptidão

  Geralmente a *aptidão* do *indivíduo* é determinada através do cálculo da função objetivo, que é definida pelas especificações de projeto. Neste trabalho, cada *indivíduo* é uma entrada para a ferramenta Tangram-II, cuja saída fornece

medidas que permitem o cálculo da *aptidão* do *indivíduo*. Ainda nessa fase os *indivíduos* são ordenados conforme a sua *aptidão*.

- Seleção

  Nessa fase, cada *indivíduo* tem uma probabilidade de ser selecionado proporcional à sua *aptidão*. Esses *indivíduos* são utilizados para gerar uma nova população por cruzamento. Um dos processos de seleção mais utilizados é denominado *amostragem universal estocástica* [34]. Para visualizar esse método considere um círculo dividido em n setores (tamanho da população), onde a área de cada setor é proporcional à *aptidão* do *indivíduo* (figura 4.3). Coloca-se sobre esse círculo uma "roleta" com n cursores, igualmente espaçados. Após um giro da roleta a posição dos cursores indica os *indivíduos* selecionados. Evidentemente, os *indivíduos* cujos setores possuem maior área terão maior probabilidade de serem selecionados várias vezes. Como consequência, a seleção de *indivíduos* pode conter várias cópias de um mesmo *indivíduo* enquanto outros podem desaparecer.

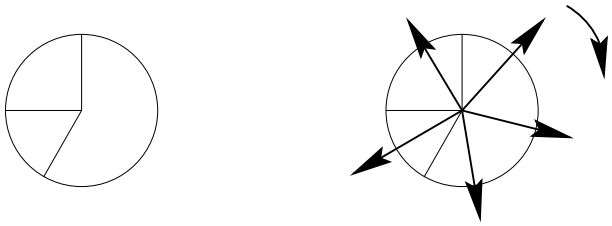

Figura 4.3: Amostragem universal estocástica

- Reprodução (*Cross-Over*)

  Os *indivíduos* selecionados na etapa anterior são *cruzados* da seguinte forma: a lista de *indivíduos* selecionados é embaralhada aleatoriamente criando-se, dessa forma, uma segunda lista, chamada lista de *parceiros*. Cada *indivíduo* selecionado é então cruzado, segundo uma probabilidade denominada de *probabilidade de cross-over*, com o *indivíduo* que ocupa a mesma posição na lista de parceiros. A forma como se realiza esse cruzamento é ilustrada na figura 4.4.

Os *cromossomos* de cada par de *indivíduos* a serem cruzados são partidos em um ponto, chamado ponto de corte (*one-point cross-over*), sorteado aleatoriamente. Um novo *cromossomo* é gerado permutando-se a metade inicial de um *cromossomo* com a metade final do outro. Deve-se notar que se o *cromossomo* for representado por uma cadeia de bits, como na figura 4.4, o ponto de corte pode incidir em qualquer posição (bit) no interior de um *gene*, não importando os limites do *gene*. No entanto, no caso de *genes* representados por números reais ou inteiros, a menor unidade do *cromossomo* que pode ser permutada é o *gene*.

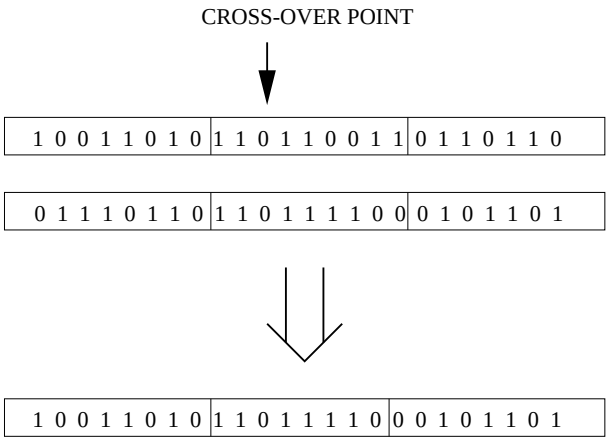

Figura 4.4: Cruzamento de dois *indivíduos* num AG

- Mutação

  A operação de *mutação* é utilizada principalmente para evitar que o AG convirja prematuramente para mínimos locais. A mutação é efetuada alterando-se o valor de um *gene* de um *indivíduo* sorteado aleatoriamente com uma determinada probabilidade, denominada probabilidade de mutação. Vários *indivíduos* da nova população podem ter um de seus *genes* alterado aleatoriamente.

4.3.3 Parâmetros do Algoritmo Genético

Além da forma como o *cromossomo* é codificado, existem vários parâmetros do

AG que podem ser escolhidos de forma a melhorar o seu desempenho, adaptando-o às características particulares de determinadas classes de problemas. Os mais importantes são: o tamanho da população, o número de gerações, a probabilidade de cross-over e a probabilidade de mutação. A influência de cada parâmetro no desempenho do algoritmo depende da classe de problemas que está sendo tratada. Assim, a determinação de um conjunto de valores otimizado para esses parâmetros dependerá da realização de um grande número de experimentos e testes. Na maioria da literatura os valores encontrados estão na faixa de 60 a 65% para a probabilidade de cross-over e entre 0,1 e 5% para a probabilidade de mutação. O tamanho da população e o número de gerações dependem da complexidade do problema de otimização e devem ser determinados experimentalmente. No entanto, deve ser observado que o tamanho da população e o número de gerações influenciam diretamente no tamanho do espaço de busca.

4.3.4 Outros Tipos de AGs

A teoria apresentada até o momento refere-se às características de um AG simples. Nesta seção são apresentadas algumas variações desse algoritmo.

- GENITOR

  O Genitor [83] é um algoritmo cujos melhores pontos encontrados são preservados na população. Esse procedimento é denominado de *elitismo*. O *elitismo* provoca uma busca mais agressiva, que na prática é geralmente bastante efetiva. No entanto existe o perigo de uma convergência prematura para mínimos locais.

  Existem três diferenças básicas entre o Genitor e os AGs simples. A etapa de reprodução produz um "descendente" de cada vez, que é colocado novamente na população atual. Esse "descendente" é colocado no lugar do pior *indivíduo* (menor *aptidão*) da população atual ao invés de substituir seus "pais". Ainda,

a *aptidão* é atribuída de acordo com um *ranking* e assume valores discretos, ao contrário da maioria dos outros tipos de AGs.

- CHC

  Outro AG baseado na coleção dos melhores *indivíduos* é o CHC (*Cross generational elitist selection, Heterogeneous recombination and Cataclysmic mutation*). Após o cruzamento, os N melhores *indivíduos* são coletados levando-se em consideração a população atual e a população gerada após o cruzamento. Remove-se os *indivíduos* duplicados. Esse método impõe uma busca mais agressiva, assim como no Genitor. Repare que a seleção está implícita no algoritmo, a partir do momento que se escolhe os melhores *indivíduos* de cada população (anterior e atual). No entanto, para se manter uma diversidade de *indivíduos* na população, somente são habilitados para o cruzamento os *indivíduos* que mantém uma determinada distância entre si, baseada no código de Hamming.

  Normalmente, esse algoritmo utiliza populações pequenas, com cerca de 50 *indivíduos*. Para solucionar o problema de convergência prematura para mínimos locais, provocada por uma busca agressiva, é utilizada uma alta taxa de mutação, preservando-se, no entanto, o melhor *indivíduo* da população. A partir da primeira seleção aleatória, utiliza-se o *cross-over* diretamente nas populações subsequentes.

- ALGORITMOS HÍBRIDOS

  Nem sempre os AGs são a melhor solução para problemas de otimização específicos. Os algoritmos híbridos utilizam os AGs como ponto de partida para métodos de otimização tradicionais, como o *simulated annealing* [49] e métodos de gradiente [84], entre outros. Os AGs introduzem um *overhead* computacional devido à busca baseada em populações. A mistura das técnicas tradicionais com os AGs introduzem uma espécie de aprendizado no AG. Após a população inicial ser gerada, é aplicada a cada *indivíduo* a técnica denominada *hill-climbing*, que utiliza derivadas. Posteriormente, após a nova geração ser criada, a técnica de *hill-climbing* é aplicada novamente a cada "descendente".

Essa técnica tem a desvantagem de degradar a habilidade de busca do AG, dado que a probabilidade de cada *indivíduo* ser selecionado é alterada. Apesar dessa desvantagem, os algoritmos híbridos possuem um desempenho satisfatório em problemas de otimização e algumas vantagens em relação aos AGs tradicionais. Como o método de *hill-climbing* é aplicado a muitos pontos do espaço de busca, a probabilidade de algum desses pontos "caminhar" na direção de um mínimo global é alta e, nesse caso, métodos que utilizam derivadas são rápidos e efetivos [85].

Utilizar uma otimização local para melhorar a população inicial, como é feito no algoritmo desenvolvido neste trabalho, que será descrito a seguir, somente degrada a amostragem inicial mas não interfere nas gerações subsequentes. Além disso, o método de *hill-climbing* conjugado com o AG pode determinar um pequeno número de alterações nas probabilidades, mas não altera bruscamente a geração subsequente.

4.3.5 Algoritmo utilizado na Metodologia Proposta

Nesta seção é discutido o algoritmo desenvolvido neste trabalho para implementar a metodologia proposta. O algoritmo utilizado é híbrido pois realiza-se uma otimização local para gerar a população inicial, antes do processo de busca de uma solução ser iniciado. Essa otimização é realizada para se determinar as faixas de variação permitidas para cada taxa do modelo, de acordo com a sensibilidade da função objetivo em relação a cada uma dessas taxas e restrições de projeto porventura utilizadas. Essas faixas são determinadas utilizado-se um método de otimização tradicional, que faz uso de derivadas, descrito no apêndice A. A figura 4.5 apresenta o diagrama de blocos do algoritmo. Uma vez gerada a população inicial, a busca de uma solução é realizada por um AG simples porque o processo de cálculo das sensibilidades já restringe o espaço de busca e, caso fosse utilizado um algoritmo com elitismo, esse espaço ficaria ainda mais restrito, prejudicando a busca de uma solução.

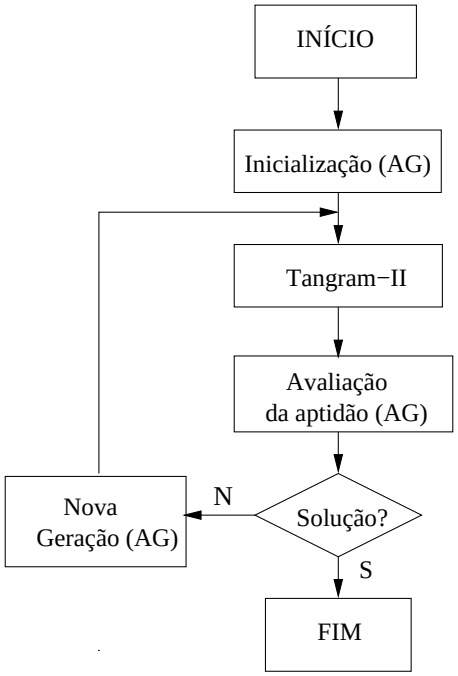

Figura 4.5: Diagrama de blocos do algoritmo da metodologia

O algoritmo desenvolvido foi adaptado para o problema de encontrar as taxas dos eventos do modelo do protocolo. A codificação do *cromossomo* foi realizada como um vetor de números reais:

$$V = (\lambda_1, \lambda_2, ..., \lambda_m) \tag{4.1}$$

onde:
λ_i - são os parâmetros (taxas) do modelo literal a serem determinados;
m - quantidade de parâmetros do modelo.

O número de taxas depende do modelo construído para o protocolo em questão e é passado como parâmetro para o programa, assim como o tamanho utilizado para a população inicial e o nome do modelo do Tangram-II.

A população inicial do AG, cujo tamanho é estabelecido pelo projetista, é obtida por uma função que gera números reais aleatoriamente, utilizando um banco de sementes do gerador de números aleatórios, o relógio do computador e as faixas iniciais determinadas pela otimização local. Os valores numéricos correspondentes a

cada *indivíduo* são transferidos pelo AG ao modelo literal do Tangram-II através de uma função denominada *trans_param*, desenvolvida por Figureiredo [21] e alterada para que os valores numéricos possam ser lidos de um arquivo. Com os parâmetros do modelo substituídos por valores numéricos torna-se possível executar um dos métodos existentes para a solução da cadeia de Markov, através do programa **solv** do Tangram-II, que é chamado pelo AG.

Os métodos de solução no Tangram-II são divididos em *métodos diretos* e *métodos iterativos*. O método é *direto* quando a solução exata é obtida após um número finito de etapas. O método é *iterativo* quando produz uma sequência de soluções aproximadas que convergem para o valor exato. Geralmente, os métodos *diretos* são apropriados quando o espaço de estados do modelo não é muito grande e a matriz de transição de estados correspondente não é esparsa. Os métodos iterativos são apropriados quando a matriz é de grande dimensão e esparsa [62].

Os métodos diretos implementados no Tangram-II são o Grassman Taksar and Heyman (**GTH**) e **GTH** por blocos. Os métodos iterativos são: **SOR** (*Sucessive Over Relaxation*), Jacobi, Gauss-Siedel e Power. Nos resultados apresentados no capítulo 5 são utilizados os métodos **GTH** e **SOR**. Como o número de estados da cadeia de Markov gerada em cada um dos exemplos é relativamente pequeno, não foram detectadas diferenças significativas entre os diversos métodos de solução testados, tanto em relação aos valores das probabilidades em estado estacionário encontradas quanto em relação ao tempo de execução do AG.

Calculadas as probabilidades em estado estacionário para cada *indivíduo* da população, obtém-se a *aptidão* bruta de cada *indivíduo* através da avaliação da função objetivo. Tendo sido determinada a *aptidão* bruta de cada *indivíduo*, calcula-se a sua *aptidão* normalizada, realiza-se o processo de seleção, cruzamento, mutação e uma nova população é gerada, caso nenhum dos *indivíduos* seja uma solução para o problema. A saída do algoritmo é um conjunto de taxas dos eventos do modelo.

A estrutura do algoritmo tem uma complexidade polinomial (n^3), proporcional ao número de pontos que se deseja aproximar, ao tamanho da população e ao número de gerações especificadas pelo projetista.

4.4 Comentários

Nesse capítulo foram apresentadas as principais características da ferramenta Tangram-II, seus fundamentos básicos, sua sintaxe e um exemplo de implementação. A integração dessa ferramenta com o algoritmo genético utilizado é fundamental para a avaliação da função objetivo e para o funcionamento da metodologia de forma geral.

Os AGs são apropriados para problemas de otimização complexos, que envolvem muitas variáveis e um espaço de soluções de dimensão elevada. Nesse contexto os algoritmos genéticos abrangem um grande número de aplicações. Os AGs possuem um custo computacional elevado, devido ao grande número de variáveis envolvidas, às populações elevadas e ao número de gerações utilizado para a cobertura do espaço de soluções.

O controle sobre os parâmetros do algoritmo é essencial para uma convergência rápida para uma solução e para evitar convergências prematuras para mínimos locais. Para problemas específicos é aconselhável a utilização de algoritmos híbridos, que misturam as técnicas dos AGs com os métodos de otimização tradicionais, que utilizam derivadas. Foi discutido o algoritmo utilizado na metodologia de partição proposta no capítulo 3 e como ele se utiliza da ferramenta Tangram-II. No próximo capítulo esse algoritmo é aplicado a alguns protocolos para a avaliação da metodologia de partição.

Capítulo 5

Resultados

5.1 Introdução

Neste capítulo são apresentados e analisados os resultados obtidos através da implementação computacional da metodologia proposta. Aplica-se a metodologia a protocolos com diferentes características. Dessa maneira é possível analisar a sua eficiência e praticidade, além da viabilidade do critério de partição HW/SW utilizado.

Na seção 5.2 é modelado um mecanismo de controle de congestionamento do protocolo TCP, onde se pretende analisar o potencial da metodologia aplicada a um protocolo bastante utilizado. Na seção 5.3 é modelado um protocolo de controle de *handoff* aplicado a uma rede sem fio baseado na tecnologia ATM (WATM) com estações base que controlam o roteamento à medida que as unidades móveis trocam de células. Nesse exemplo pretende-se analisar a metodologia quando aplicada ao projeto de um protocolo dominado por fluxo de controle. Na seção 5.4 é modelado um protocolo *multicast* confiável que realiza processamento nos nós (*nós ativos*). Nesse exemplo, pretende-se analisar a metodologia no caso de um protocolo com um alto grau de processamento nos nós. Em todos os exemplos apresentados a seguir é utilizada a hipótese de que os atrasos da implementação em SW são aproximadamente

10 vezes maiores do que os atrasos obtidos para as implementações em *standard cells*.

5.2 TCP "Congestion Avoidance"

5.2.1 TCP "Congestion Avoidance" com perda determinística

Neste exemplo é analisado o desempenho do protocolo TCP quando em fase de controle de congestionamento, o "TCP Congestion Avoidance", utilizando-se o modelo matemático proposto em Mathis [86]. A janela de congestionamento é incrementada de uma parcela constante para cada ACK recebido pelo emissor e dividida pela metade quando é detectado um congestionamento. O modelo é simplificado e supõe que somente o mecanismo "Congestion Avoidance" influencia o desempenho do TCP. As seguintes suposições são realizadas:

- tráfego moderado (com poucas perdas), permitindo que o TCP se recupere das perdas sem a necessidade de *timeout*;
- não existem retransmissões por *timeout*;
- a janela do receptor possui um tamanho "suficientemente grande", ou seja, a janela de congestionamento é controlada pela janela do emissor;
- múltiplas perdas dentro de um *round trip time* (RTT) indicam sinal de congestionamento;
- o emissor tem sempre dados a enviar;
- o receptor envia um ACK para cada *segmento* de dados (certa quantidade de dados em bytes de tamanho fixo) recebido;
- o RTT é constante porque a banda passante é suficiente para que não ocorra a formação de filas;

- as conexões são longas o suficiente para que o algoritmo de "Congestion Avoidance" alcance estado estacionário;
- perda de pacotes com uma probabilidade constante p. Isso significa que aproximadamente $1/p$ pacotes são enviados corretamente, seguidos da ocorrência de uma perda;
- detalhes referentes a recuperação e retransmissão dos pacotes são desprezados, embora a recuperação de perdas seja completada dentro de um RTT.

Com as suposições acima, pode-se aproximar a curva *cwnd versus RTT* por um gráfico com a forma de um *dente de serra*, como é mostrado na figura 5.1, onde *cwnd* é o tamanho da janela de congestionamento do emissor em número de pacotes e *RTT* é o *round trip time*. O modelo se aplica a quase todas as implementações SACK TCP (TCP com *Selective Acknowledgements*).

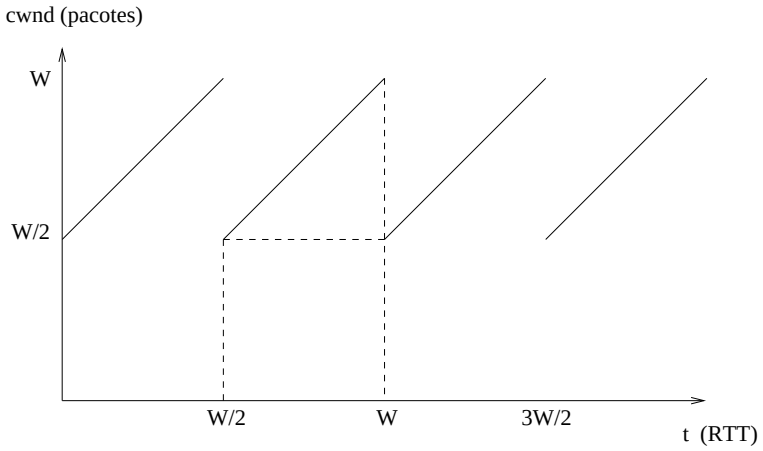

Figura 5.1: cwnd versus RTT com perda periódica.

Analisando-se a figura 5.1, encontra-se a equação para o tamanho máximo da janela de congestionamento em função da probabilidade de perda, conforme demonstrado em Mathis [86], considerando-se a quantidade de pacotes enviada em um ciclo (área sob um ciclo do gráfico) e a informação de que $1/p$ pacotes são enviados em cada ciclo:

$$W = \sqrt{8/(3 * p)} \tag{5.1}$$

onde W é o tamanho máximo da janela de congestionamento, em pacotes. Mas a largura de faixa em bytes, BW, é dada pela quantidade de dados transmitidos por ciclo (em bytes) dividida pelo tempo do ciclo (em segundos), ou seja:

$$BW = (MSS * 3/8 * W^2)/(RTT * W/2) \tag{5.2}$$

onde MSS é o tamanho máximo de um segmento de dados em bytes. Substituindo-se W na equação 5.2 obtém-se:

$$BW * RTT/MSS = C/\sqrt{p} \tag{5.3}$$

onde C é uma constante de proporcionalidade, nesse caso igual a $\sqrt{3/2}$ e p é a probabilidade de perda. A expressão $BW * RTT/MSS$ fornece o tamanho da janela de congestionamento em número de pacotes. A constante C engloba uma série de termos constantes que dependem do tipo de implementação do TCP, da estratégia de envio de ACK's utilizada e do mecanismo de perda (periódica ou aleatória).

O mecanismo de controle de congestionamento "TCP Congestion Avoidance" é especificado por uma rede de Petri, como é ilustrado na figura 5.2 [16][17].

Os valores para as taxas da rede de Petri do modelo determinístico são estipulados considerando-se que para cada unidade de tempo (no caso, um RTT), que corresponde à recepção de um ACK, o valor da janela é incrementado de um segmento, a partir de um valor inicial que, em estado estacionário, corresponde à metade da janela máxima W, onde W é calculada a partir da equação 5.1. Como exemplo, considerou-se um valor para a probabilidade de perda constante igual a 0,01, ou seja, aproximadamente a cada 100 pacotes enviados corretamente um é perdido. Dessa forma, no modelo do Tangram-II, atribui-se o valor 4 para as taxas dos eventos (transições) Envia_pct (λ_2), Msg_ok (λ_7), Envia_ack (λ_5) e Ack_ok (λ_3) e o valor 1 para o evento Aumenta_jan (λ_1). Neste caso, os eventos são determinísticos. Para garantir que o disparo da transição PERDA seja realizado após a transmissão de exatamente $1/p$ pacotes, criou-se uma variável de estado denominada **janela**, cujo valor máximo, calculado pela equação 5.1, é uma das condições para que o evento

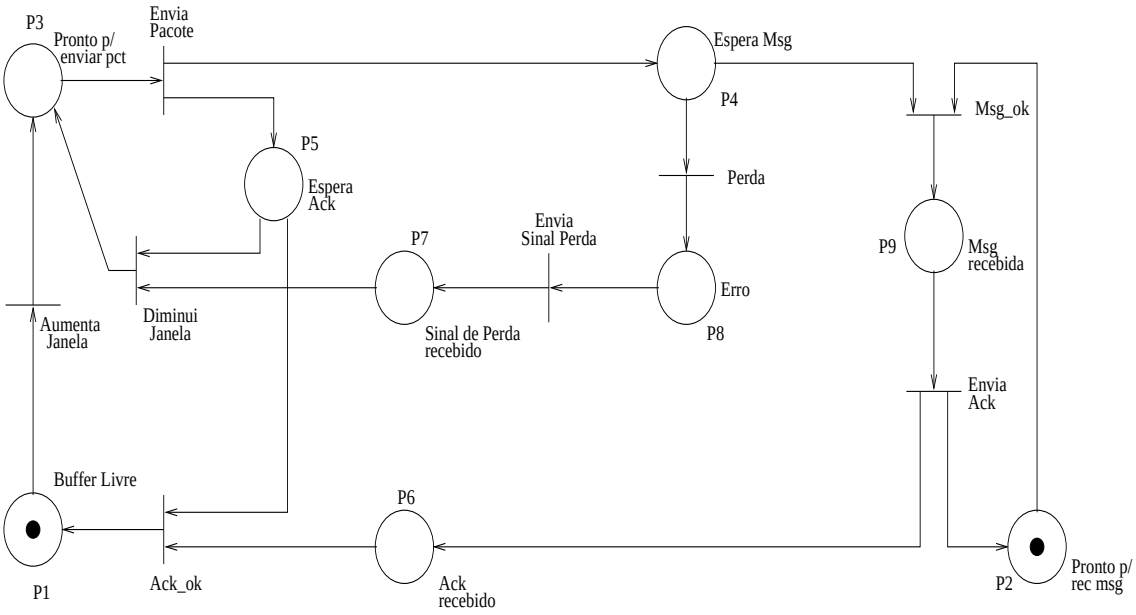

Figura 5.2: Rede de Petri do mecanismo de controle de congestionamento "TCP Congestion Avoidance".

Perda seja disparado.

Dessa forma, a cada RTT $(1/4+1/4+1/4+1/4)$ a janela de congestionamento é aumentada de um segmento. Note que, segundo o gráfico da figura 5.1, a diminuição da janela devido a um congestionamento ocorre quase que instantaneamente (1 RTT, segundo a suposição do modelo), por isso são fixados valores altos para as taxas dos eventos **Perda** (λ_8), Sinal_perda (λ_6) e Diminui_jan (λ_4). A partir do modelo do Tangram-II, é realizada uma simulação para se analisar o comportamento da janela de congestionamento. Verifica-se que o comportamento da janela de congestionamento fornece exatamente um gráfico no formato de uma dente de serra, mostrado na figura 5.1.

5.2.2 TCP "Congestion Avoidance" com perda aleatória

Nesta seção é apresentada uma versão onde a perda de pacotes é aleatória. Dessa forma, a taxa da transição PERDA da rede de Petri mostrada na figura 5.2 possui uma distribuição de probabilidades exponencial, segundo a metodologia apresentada.

A hipótese de que não se formam filas continua valendo para este caso.

A determinação dos parâmetros do protocolo é realizada pelo AG aproximando a curva $(BW * RTT)/MSS$ *versus perda*, traçada utilizando-se a equação 5.3, com $C = 1$, que pode ser visualizada na figura 5.3. A expressão $BW * RTT/MSS$ é uma estimativa do tamanho médio da janela de congestionamento. Deve ser observado que o gráfico está em escala logarítmica.

A cadeia de Markov gerada foi resolvida para diversos valores de probabilidade de perda (transição PERDA), de forma que o algoritmo genético pudesse escolher o conjunto de valores para as taxas de transição que melhor aproxima a curva.

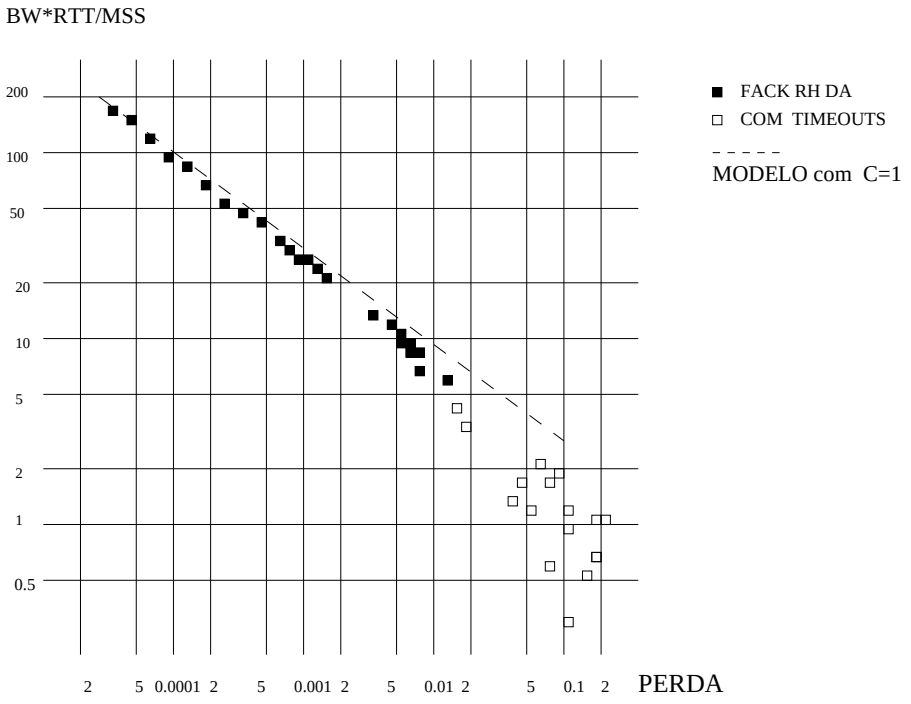

Figura 5.3: Curva Janela vs *perda* para o modelo matemático dado pela eq. 5.3.

Foram utilizados 10 pontos para aproximar a curva do modelo e obter os parâmetros desejados. Foram realizados experimentos com uma população de 100 indivíduos e utilizou-se como critério de terminação do algoritmo genético o limite de 10 gerações. A taxa da transição PERDA (λ_8) foi variada (abscissa da curva) e a função objetivo é definida como a soma das diferenças entre o tamanhos de janela obtidos e os tamanhos de janela desejados, dados pelos pontos escolhidos. Obteve-se um erro mínimo para a função objetivo igual a aproximadamente 1,67. Esse valor indica um

erro aproximado de 1,7% para cada um dos pontos escolhidos. Os resultados obtidos são apresentados na tabela 5.1. O pequeno erro no valor das janelas fornecidas pelo AG se deve ao valor da constante C: com perdas aleatórias seu valor é um pouco menor do que 1 [86].

Transição	**Taxa (1/s)**
Aumenta_jan (λ_1)	446,94
Envia_pct (λ_2)	1078,67
Envia_ack (λ_5)	984,02
Diminui_jan (λ_4)	46,17
Ack_ok (λ_3)	803,90
Sinal_perda (λ_6)	116,41
Msg_ok (λ_7)	1

Tabela 5.1: Taxas do TCP "Congestion Avoidance" obtidas pelo AG.

Observe que a transição MSG_OK (λ_7) foi fixada em 1 porque o modelo original assume que $1/p$ pacotes são enviados corretamente antes da ocorrência de uma perda, portanto, em média, $1/p$ pacotes são enviados antes de uma perda. Aplicando-se o mesmo critério de partição utilizado na seção 3.4 às taxas da tabela 5.1, obtém-se os resultados apresentados na tabela 5.2:

Evento	**Produto**
$\lambda_1 * \pi_1$	1,07
$\lambda_2 * \pi_2$	1,76
$\lambda_3 * \pi_7$	1,04
$\lambda_4 * \pi_6$	0,013
$\lambda_5 * \pi_5$	1,71
$\lambda_6 * \pi_4$	0,23
$\lambda_7 * \pi_3$	0,99

Tabela 5.2: Valores obtidos para o produto $\lambda_i * \pi_j$.

onde cada π_j é dado por:

E_1 - 1 ficha em P1 e outra em P2;

E_2 - 1 ficha em P2 e outra em P3;

E_3 - 1 ficha em P2, outra em P4 e outra em P5;

E_4 - 1 ficha em P2, outra em P5 e outra em P8;

E_5 - 1 ficha em P5 e outra em P9;

E_6 - 1 ficha em P2, outra em P5 e outra em P7;

E_7 - 1 ficha em P2, outra em P5 e outra em P6;

De acordo com os resultados da tabela 5.2, as operações do protocolo correspondentes aos eventos 2 e 5, por terem maior "vazão", devem ser implementadas em HW enquanto as correspondentes aos eventos 4 e 6, por terem baixa "vazão", podem ser implementadas em SW. Com relação aos eventos 1, 3, 7, como os valores são próximos, deve-se analisar o modelo. Observa-se que os eventos 2, 3, 5 e 7 são aqueles que correspondem à transmissão de pacotes e ao recebimento de ACK's. É como se esses eventos correspondessem ao *caminho crítico do protocolo*, isto é, os eventos pelos quais o protocolo passa o maior número de vezes quando em estado estacionário. De acordo com essa observação, as operações do protocolo correspondentes aos eventos 2, 3, 5 e 7 devem ser implementadas em HW enquanto que as correspondentes aos eventos 1, 4 e 6 podem ser implementadas em SW. Essa partição é avaliada em conjunto com as medidas de atraso e custo de área fornecidas pelas ferramentas Synopsys e Altera. Caso o custo do HW esteja dentro das especificações e as taxas encontradas possam ser atendidas pelos circuitos sintetizados, uma solução foi encontrada. Caso contrário, uma nova população é gerada para que o processo de otimização continue.

Não foi possível encontrar uma solução analítica que considerasse a variação no tamanho da janela no modelo do Tangram-II, pois neste caso o número de estados da cadeia de Markov gerada excede o limite da ferramenta. Foi necessário criar uma *recompensa de impulso* (vide seção 4.2) denominada **tam_jan** para que o tamanho da janela ao longo do tempo pudesse ser analisado. Esse é um recurso que a ferramenta Tangram-II possui para auxiliar os processos de simulação. No modelo desse

exemplo, essa recompensa é incrementada cada vez que o evento **Aumenta_jan** é disparado ou decrementada cada vez que o evento **Diminui_jan** é disparado. Após a determinação das taxas de transição, realizada pelo algoritmo genético, a taxa do evento **Perda** (λ_8), que tem distribuição exponencial, foi fixada em 0,01 para que se pudesse fazer uma comparação do comportamento do protocolo com perda perió-dica (caso anterior) com o comportamento do protocolo com perda aleatória. Foi realizada uma simulação onde foi feito um *trace* da recompensa **tam_jan**. O resultado pode ser visualizado na figura 5.4. O modelo do Tangram-II foi construído para que o tamanho da janela volte ao valor original cada vez que ocorre uma perda, ao contrário do modelo original, onde o tamanho da janela cai à metade. Essa alteração foi realizada para se visualizar melhor o comportamento do crescimento do tamanho da janela em relação à variação da probabilidade de perda. Não foi fixado uma valor máximo para a janela como condição para o evento **Perda** ser disparado, como no caso anterior, pois neste caso a probabilidade de perda tem distribuição exponencial.

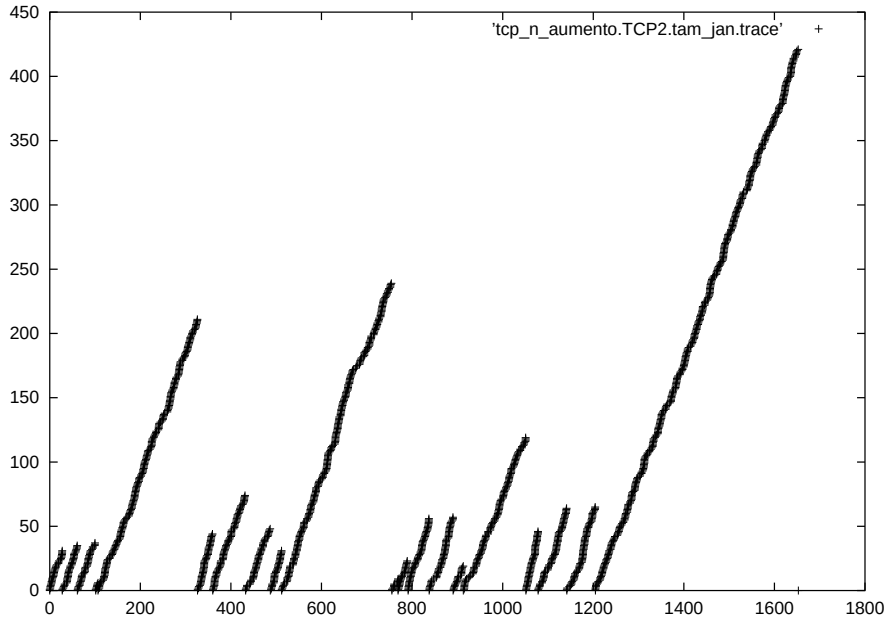

Figura 5.4: Evolução da Janela de congestionamento com probabilidade de perda com distribuição exponencial.

Observa-se através do gráfico da figura 5.4 e dos valores fornecidos pela simu-

lação realizada no Tangram-II que a média dos valores máximos do tamanho que a janela de congestionamento alcança em cada ciclo é aproximadamente igual ao valor máximo da janela do caso determinístico, para uma mesma probabilidade de perda. Quanto menor a probabilidade de perda, melhor é o resultado fornecido pelo algoritmo genético. É justamente nesses casos (poucas perdas) que o modelo da equação 5.3 se ajusta melhor aos casos reais. Esse resultado indica que o modelo foi bem construído, os valores das taxas encontradas estão coerentes e o seu comportamento se aproxima mais de um comportamento **real** do mecanismo "Congestion avoidance" encontrado nas diversas implementações do protocolo TCP. O modelo **ideal** da seção anterior faz muitas suposições que não se verificam na prática.

Avaliação da Partição

As descrições *comportamentais* em VHDL de cada transição de estado do protocolo são utilizadas como entrada para as ferramentas Synopsys e Altera. A síntese lógica dos circuitos no Synopsys foi realizada utilizando-se uma biblioteca de células padrão (*standard cells*) ES2 com tecnologia 0,7μm. A implementação dos circuitos lógicos no Altera foi realizada utilizando-se o compilador do ambiente no modo "auto" para a escolha do componente mais adequado ao projeto. Essa síntese foi realizada por Lima [16].

A tabela 5.3 mostra as medidas de atraso e custo de área obtidos para cada uma das transições de estado do protocolo. Os valores de área são fornecidos automaticamente pelo Synopsys. Medidas de área não fazem sentido no caso de utilização da ferramenta Altera, pois o componente escolhido tem área fixa.

Essas medidas mostram uma vantagem do projeto utilizando *standard cells* em relação à implementação utilizando PLDs, no que diz respeito à velocidade de processamento. Isso acontece porque o Synopsys oferece um maior grau de liberdade na alocação e roteamento de células, implicando num melhor compromisso entre área e velocidade. Por outro lado, a implementação em PLDs é imediata e o sistema pode ser reprogramado sem a necessidade de uma nova rodada de fabricação, como no caso das *standard cells*.

Transição	PLD Atraso	Standard Cells Área (mm^2)	Standard Cells Atraso
Aumenta_jan (λ_1)	17 ns	0,213	5,5 ns
Envia_pct (λ_2)	50 ns	8,456	10 ns
Ack_ok (λ_3)	7 ns	0,017	5 ns
Diminui_jan (λ_4)	7 ns	0,194	5 ns
Envia_ack (λ_5)	7 ns	0,017	5 ns
Sinal_perda (λ_6)	7 ns	0,017	5 ns
Msg_ok (λ_7)	50 ns	8,456	10 ns

Tabela 5.3: Medidas de atraso e área.

Tendo em vista os resultados apresentados na tabela 5.3, a partição proposta é avaliada. Levando-se em conta, por exemplo, um custo de área especificado pelo projetista de 4 mm^2, as transições 2 e 7 ultrapassam os requisitos desejados. Sugere-se, então, que essas transições sejam implementadas em PLDs. Apesar das medidas de atraso da implementação em PLD serem maiores do que as da implementação em *standard cells*, elas atendem às taxas calculadas pelo AG (vide tabela 5.1). As transições 3 e 5 serão implementadas em *standard cells*, pois as medidas de área atendem ao requisito exigido e as medidas de atraso atendem às taxas calculadas pelo AG.

Análise do Desempenho da Partição Escolhida

Uma vez determinada a partição HW/SW, utiliza-se os atrasos fornecidos pelas ferramentas Synopsys e Altera para se analisar o desempenho da partição. Calcula-se a taxa da i-ésima transição (λ_i) da mesma maneira realizada na seção 3.4.3. A tabela 5.4 mostra a soma dos erros obtidos para cada ponto da curva usada na otimização, para os casos onde o protocolo é totalmente implementado em SW, totalmente implementado em PLD, totalmente implementado em *standard cells* e implementado utilizando-se a partição HW/SW com PLDs e com *standard cells* (SCs). Pôde-se verificar que não existe um ganho de desempenho quando as tran-

sições são implementadas em HW. Somente o erro em relação à curva utilizada para a otimização diminui à medida que a velocidade de processamento aumenta. Isso se deve ao fato do modelo utilizado supor que a janela de congestionamento é completamente dependente da probabilidade de perda de pacotes no meio (vide Eq. 5.3). Os resultados obtidos estão coerentes com os resultados esperados para cada tipo de implementação: o erro maior para a implementação em SW, o menor para a implementação em *standard cells* e valores intermediários para as partições HW/SW e para a implementação em PLD.

Implementação	**ERRO**
SW	0,000360
HW (SC)	0,000019
HW (PLD)	0,000104
HW/SW (SC/SW)	0,000036
HW/SW (PLD/SW)	0,000117

Tabela 5.4: Erros obtidos para cada implementação.

Sob as suposições realizadas pelo modelo, a equação 5.3 pode ser vista como um limite superior para o desempenho do protocolo, ou seja:

$$BW * RTT/MSS < C/\sqrt{p} \tag{5.4}$$

O **ERRO** obtido, mostrado na tabela 5.4 é calculado em relação a esse limite. Pode-se observar que os resultados obtidos, ou seja, os valores dos erros obtidos para cada tipo de implementação, estão coerentes com os resultados esperados: a implementação somente em SW fornece o maior erro e a implementação somente em *standard cells* fornece o menor erro. As implementações com PLDs e as partições PLD/SW e SC/SW fornecem valores intermediários.

Os resultados das simulações realizadas em Mathis [86] comprovam que a equação 5.3 é um limite superior para a largura de faixa de implementações do protocolo TCP baseadas no mecanismo "Congestion Avoidance".

5.3 Protocolo de Controle de *Handoff* para uma rede ATM sem fio

Em redes onde existem terminais móveis, o processo de migração de um terminal entre duas estações base exige a existência de um controle das conexões ativas. Esse processo de transição é denominado de *handoff*. Neste exemplo, a metodologia apresentada é aplicada a um protocolo de controle de *handoff* para redes ATM sem fio e o desempenho do protocolo é analisado utilizando-se a sequência de primitivas fornecidas em Acharya [87]. Nesse tipo de protocolo, a localização de um terminal em relação à rede não pode ser determinada apenas através de seu endereço. Esquemas adicionais de endereçamento são necessários para localizar e rastrear os terminais móveis, juntamente com modificações apropriadas no processo de iniciação da conexão. Um protocolo de controle de *handoff* eficiente estabelece novas rotas dinamicamente, ao invés de estabelecer toda a conexão novamente. Os esquemas de *handoff* podem ser classificados de duas maneiras: aqueles que estabelecem novas rotas e aqueles que realizam uma extensão da rota existente. O primeiro esquema é baseado na remoção de parte da conexão existente e na adição de um novo caminho a partir do ponto de separação, também chamado de ponto de *cross-over*. A seleção do ponto de *cross-over* influencia diretamente o desempenho do protocolo em termos de latência, probabilidade de perda e utilização. O segundo esquema é baseado na extensão da conexão inicial, a partir do ponto de acesso anterior, para o novo ponto de acesso [88]. A figura 5.5 apresenta a especificação do protocolo dado por uma rede de Petri condição-ação.

Neste exemplo, a medida de desempenho utilizada é a probabilidade de perda de pacotes em função da taxa de transmissão. O atraso ocasionado pelo *handoff* (T_d) fornece o tempo mínimo para se alterar a tabela de roteamento para múltiplos VCs. Esse tempo é diretamente proporcional à taxa de perda causada pelo procedimento de *handoff*. No modelo utilizado em Yuan [89], uma perda ocorre quando uma mudança na tabela de roteamento é realizada durante a transmissão de um fluxo

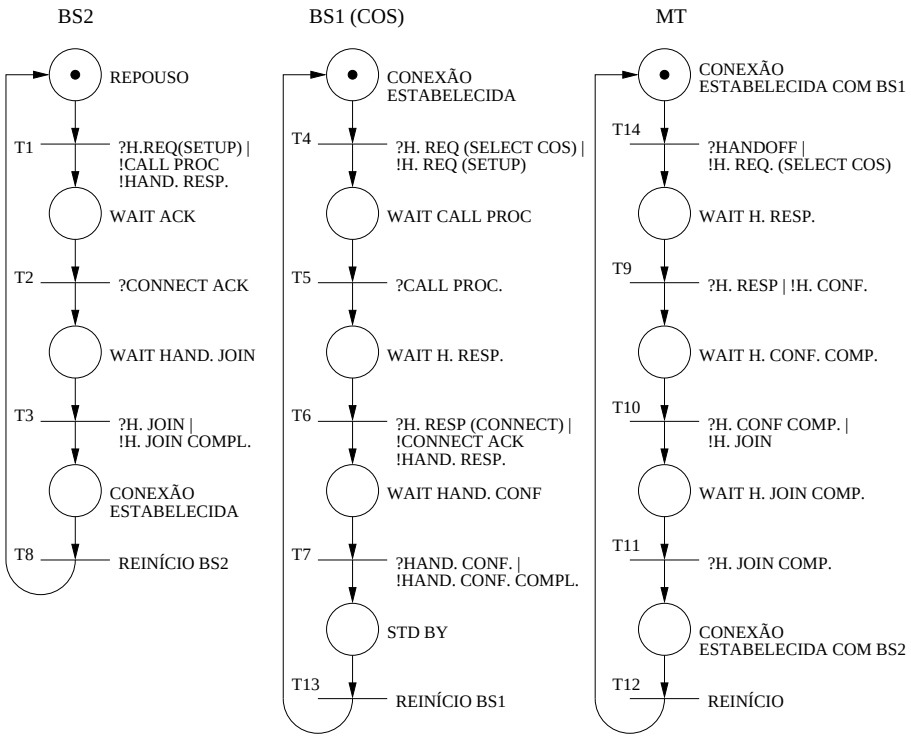

Figura 5.5: Rede de Petri condição-ação do Protocolo de controle de *Handoff*.

de células dentro do mesmo pacote. Considere que o evento de *handoff* ocorre aleatoriamente dentro do intervalo $T = 1/R$, onde T é o tempo de transmissão de um pacote e R é a taxa de transmissão de pacotes. Portanto, a probabilidade de perda de pacotes P é dada por $P = T_d * R$. A figura 5.6 mostra o gráfico P versus R.

Foram escolhidos 6 pontos da curva para se determinar os parâmetros do modelo, o tamanho da população é de 100 indivíduos e o número de gerações utilizado como critério de terminação é igual a 10. A função objetivo é a diferença entre a probabilidade de perda obtida e a probabilidade de perda desejada, ou seja, os parâmetros do protocolo são extraídos aproximando-se o gráfico da figura 5.6. Os resultados são apresentados na tabela 5.5. O erro encontrado para a soma das diferenças entre as probabilidades desejadas e obtidas foi de 0,006.

Utilizando as taxas da tabela 5.5 e as probabilidades em estado estacionário calculadas pelo Tangram-II, o mesmo critério de partição da seção 3.3.5 pode ser aplicado e o produto $\lambda_i * \pi_j$ calculado para cada um dos eventos (transições) do

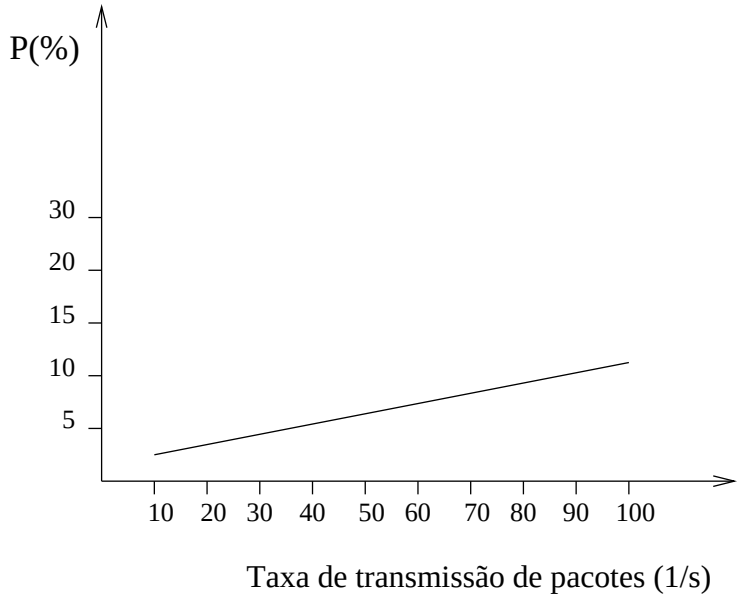

Figura 5.6: Probabilidade de Perda em função da taxa de transmissão.

Transição	**Taxa (1/s)**
t_1 (λ_1)	42,34
t_2 (λ_2)	60,12
t_3 (λ_3)	421,63
t_4 (λ_4)	344,91
t_5 (λ_5)	477,38
t_6 (λ_6)	397,72
t_7 (λ_7)	14,31
t_8 (λ_8)	156,63
t_9 (λ_9)	20,76
t_{10} (λ_{10})	8,43
t_{11} (λ_{11})	114,90
t_{12} (λ_{12})	338,55
t_{13} (λ_{13})	474,19
t_{14} (λ_{14})	68,03

Tabela 5.5: Taxas da rede de Petri do protocolo de *Handoff* obtidas pelo AG.

protocolo. Os resultados são mostrados na tabela 5.6:

Evento	**Produto**
$\lambda_1 * \pi_1$	1.52
$\lambda_2 * \pi_2$	3.05
$\lambda_3 * \pi_3$	5.22
$\lambda_4 * \pi_4$	97.61
$\lambda_5 * \pi_5$	14.31
$\lambda_6 * \pi_6$	27.75
$\lambda_7 * \pi_7$	70.32
$\lambda_8 * \pi_8$	0.20
$\lambda_9 * \pi_9$	1.17
$\lambda_{10} * \pi_{10}$	0.17
$\lambda_{11} * \pi_{11}$	1.88
$\lambda_{12} * \pi_{12}$	80.21
$\lambda_{13} * \pi_{13}$	13.60
$\lambda_{14} * \pi_{14}$	3.67

Tabela 5.6: Valores obtidos para o produto $\lambda_i * \pi_j$.

As tarefas do protocolo relacionadas às transições t_4, t_5, t_6, t_7, t_{12} e t_{13} devem ser implementadas em *hardware* porque as respectivas transições possuem um produto $\lambda_i * \pi_j$ maior, enquanto aquelas relacionadas às demais transições, com um produto menor, podem ser implementadas em *software*. Como nos exemplos anteriores, essa é a primeira partição a ser avaliada em relação ao custo de implementação em *hardware*. De acordo com os resultados obtidos e analisando-se o modelo da figura 5.5, as transições a serem implementadas em *hardware* aceleram o processo de *handoff* porque são aquelas associadas às tarefas de mudança da tabela de roteamento na *cross-over switch* e ao restabelecimento da conexão do terminal móvel com a nova estação base. Isso indica que o comportamento do modelo aproxima o mecanismo real de protocolos de controle de *handoff* encontrados em diversas implementações de redes ATM sem fio.

5.3.1 Avaliação da Partição

As descrições VHDL de cada transição do protocolo são usadas tanto para a implementação em *standard cells* quanto em PLDs. No primeiro caso, a síntese realizada por Lima [88] utiliza a biblioteca padrão ES2 com tecnologia 0.7μm. A tabela 5.7 mostra as medidas de área e atraso obtidas para cada uma das transições de estado do protocolo.

Transição	**Standard Cells Área (mm^2)**	**Standard Cells Atraso**	**PLD Atraso**
1	0.150	20 ns	60 ns
2	0.015	5 ns	30 ns
3	0.023	5 ns	30 ns
4	0.223	16 ns	60 ns
5	0.015	5 ns	30 ns
6	0.130	20 ns	60 ns
7	0.023	5 ns	30 ns
8	0.015	5 ns	30 ns
9	0.024	5 ns	30 ns
10	0.015	5 ns	30 ns
11	0.015	5 ns	30 ns
12	0.015	5 ns	30 ns
13	0.015	5 ns	30 ns
14	0.223	16 ns	60 ns

Tabela 5.7: Medidas de área e atraso.

A partição proposta é avaliada tomando os valores apresentados e considerando uma área especificada para o custo de, por exemplo, 1 mm^2. Dessa forma, pode-se notar que todas as transições indicadas pelo processo de particionamento (t_4, t_5, t_6, t_7, t_{12} e t_{13}) podem ser implementadas em *hardware* utilizando-se *standard cells* ou PLDs pois todas as medidas de área são menores do que 1 mm^2 e os atrasos associados satisfazem as taxas calculadas pelo AG.

5.3.2 Análise do Desempenho da Partição Escolhida

Utilizando o mesmo procedimento da seção 3.4.3 realiza-se a análise final da partição escolhida. A figura 5.7 compara o desempenho do protocolo de *handoff* totalmente implementado em *software*, em *standard cells* e em PLDs com o desempenho da partição HW/SW escolhida. Para isso, duas implementações da partição escolhida foram realizadas: a primeira, usando parte das transições implementadas em PLDs (aquelas designadas para *hardware*) e parte implementada em *software*. A segunda, usando parte das transições implementadas em *standard cells* e parte em *software*. O parâmetro de desempenho utilizado para comparar essas implementações é a probabilidade de perda em função da taxa de transmissão.

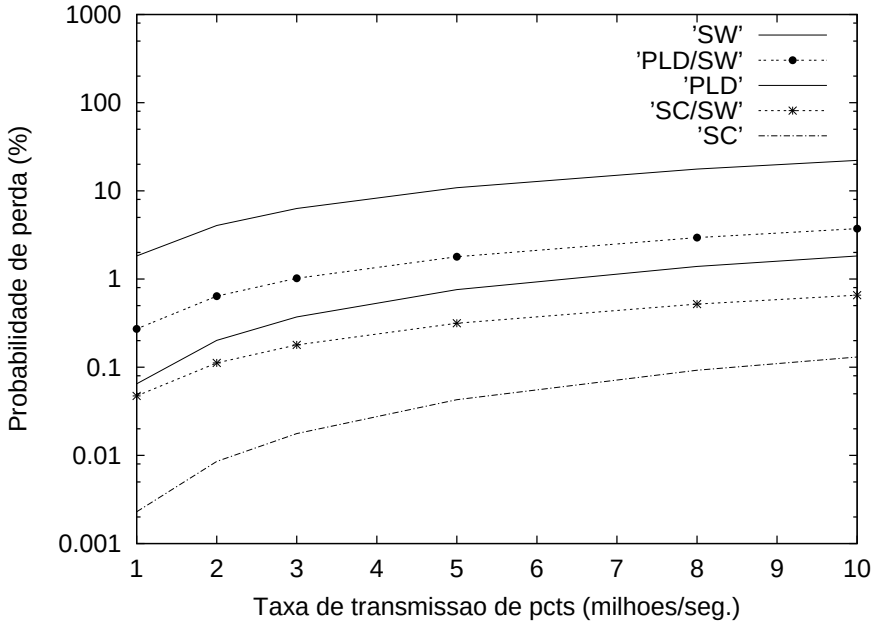

Figura 5.7: Comparação entre as diversas implementações.

Pode-se observar que a probabilidade de perda na implementação utilizando *standard cells* é aproximadamente 900 vezes menor do que a probabilidade de perda encontrada para a implementação em *software*. A implementação em PLDs fornece perdas trinta vezes maiores que a implementação em *standard cells*. Não obstante, implementações em PLDs são ferramentas importantes para prototipagem rápida de sistemas de comunicação a um custo baixo. A partição HW/SW implementada usando PLDs fornece resultados seis vezes melhor do que a implementação em *soft-*

ware. A partição HW/SW implementada usando *standard cells* fornece resultados quarenta vezes melhor do que a implementação em *software*. Portanto, neste caso, a solução utilizando PLDs satisfaz os requisitos de desempenho com um custo menor do que a implementação em *standard cells*. Observe também que os resultados estão coerentes com aqueles obtidos no exemplo anterior.

5.4 Protocolo ARM

O projeto de protocolos *multicast* confiáveis [90][91] para a Internet é um problema de difícil tratamento devido à capacidade limitada da rede e do próprio emissor em responder a sinalizações de perdas. A utilização apenas da otimização em *software* no projeto desses protocolos nem sempre permite uma operação em alta velocidade, causando sobrecarga na rede e o problema conhecido como **implosão de NACKs**. Daí a necessidade da utilização de *hardware* para aumentar o desempenho desses protocolos.

O protocolo ARM (*Active Reliable Multicast*) [82] tem como objetivo a recuperação de pacotes perdidos em protocolos *multicast* confiáveis, utilizando roteadores intermediários, denominados roteadores **ativos**, que protegem o emissor e a rede de tráfego desnecessário de NACKS e de pacotes de reparo. Esses roteadores realizam um processamento local que permite a recuperação de pacotes de dados perdidos sem a necessidade de retransmissão desse pacote para todo o grupo. Dessa forma, o problema de implosão de NACKs é evitado, a carga de retransmissões é distribuída e as retransmissões são restritas a um escopo local da rede, levando a uma queda significativa no consumo de banda passante.

O protocolo ARM emprega três tipos de estratégias de recuperação de perdas: supressão de NACKs duplicados, um esquema de recuperação local de perdas baseado nos roteadores ativos e transmissão *multicast* parcial, ou seja, o pacote de reparo é transmitido para um escopo reduzido e não para todo o grupo. A supressão de NACKs duplicados reduz o número de NACKs navegando em direção ao emissor e

o tráfego que cruza os enlaces de gargalo da rede. A recuperação local de perdas reduz a latência fim-a-fim e distribui a carga de retransmissões.

Roteadores ativos colocados em posições estratégicas armazenam dados de forma *best effort* para possíveis futuras retransmissões. Normalmente esses roteadores são colocados imediatamente antes de enlaces onde ocorrem muitas perdas. Esse esquema permite aos nós-destino recuperarem-se rapidamente de perdas de pacotes de dados.

Considera-se que a rede fornece o endereço IP-multicast de acordo com o estilo de roteamento *multicast* [92], no qual uma árvore com raiz no emissor é construída para distribuir os pacotes de dados. O protocolo ARM é do tipo *receiver-reliable*, ou seja, os receptores são responsáveis pela detecção da perda e por requisitar os pacotes perdidos, através de seu número de sequência. Os receptores detectam perdas através do recebimento de um pacote com número de sequência errado. Considera-se um cenário onde existe apenas um emissor e vários receptores no grupo *multicast*. O receptor envia um NACK ao emissor no momento em que é detectada uma perda. Múltiplos NACKs de diferentes receptores são armazenados e "fundidos" nos nós ativos ao longo da árvore *multicast*. Nós inativos simplesmente retransmitem o pacote em direção ao emissor. O emissor responde ao primeiro NACK recebido transmitindo um pacote de reparo para todos os participantes do grupo *multicast* e ignora NACKs subsequentes para esse pacote por um determinado tempo.

Quando um roteador ativo recebe um NACK, indicando que um receptor detectou uma perda, ele retransmite o pacote solicitado, caso o mesmo esteja armazenado. Caso contrário o NACK é processado para se saber se será descartado, no caso de ser um NACK duplicado, ou enviado em direção ao emissor. Além disso, conforme mencionado anteriormente, os roteadores ativos retransmitem pacotes somente para aqueles receptores que, previamente, os tenham requisitado (*multicast* parcial).

Neste exemplo, a análise do desempenho do protocolo é focalizada no tempo de processamento de um pacote em um roteador ativo como função do número de receptores por grupo para cada partição HW/SW determinada pelo algoritmo genético. Outra medida de interesse comumente utilizada é a banda passante da rede. No

entanto, com a evolução da tecnologia para redes de alta velocidade e o crescimento dos protocolos *multicast* como um importante paradigma de comunicação, essa medida vem deixando de ser um fator crítico e, neste caso, a velocidade de processamento nos nós torna-se um fator preponderante para o desempenho dos protocolos *multicast* confiáveis [93].

A figura 5.8 apresenta uma especificação em rede de Petri de um protocolo ARM. Essa especificação simplificada é composta de um receptor, um roteador ativo e K receptores por grupo *multicast*. Seguindo o procedimento da metodologia, a especificação do protocolo ARM é utilizada como base para a construção do modelo paramétrico do Tangram-II. Nesse caso, no entanto, como os nós da rede são ativos, o modelo do Tangram-II é bem mais detalhado do que nos casos anteriores. O comportamento do roteador ativo, descrito nos parágrafos anteriores, cujos detalhes podem ser encontrados em Lehman [82], está representado no modelo do Tangram-II. A implementação desse modelo é encontrada no apêndice B.

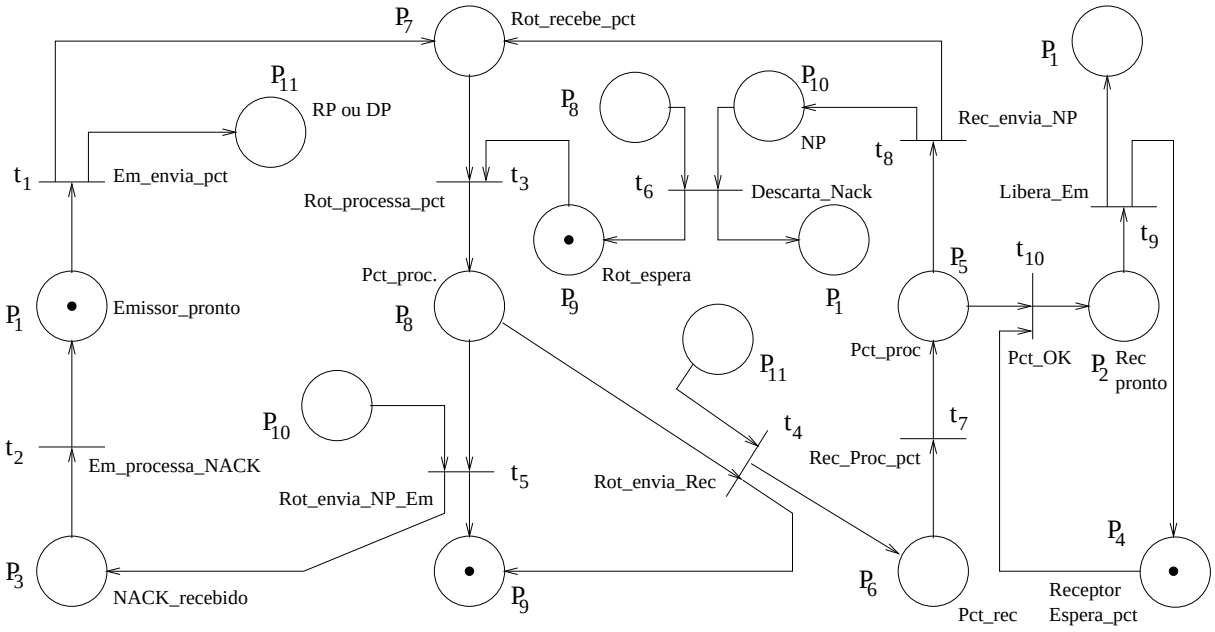

Figura 5.8: O protocolo ARM.

Como, neste caso, não é especificada pelas referências uma curva de desempenho, deve ser encontrada uma expressão que relacione o tempo de processamento com o número de receptores de um grupo *multicast*. Considere o número de pacotes de

dados e pacotes de reparo processados por unidade de tempo num roteador ativo, além do número de NACKs processados quando esse roteador possui o pacote requisitado armazenado. Suponha ainda que o roteador envia o pacote processado para todos os receptores do grupo. Se a soma das probabilidades em estado estacionário dos estados que habilitam o evento t_3 (e cujas variáveis de estado correspondem ao comportamento descrito) é multiplicada pela taxa da transição t_3 (λ_3), obtém-se a parte da "vazão" da transição t_3 que deve ser igual à "vazão" da transição t_4, isto é:

$$\pi_i * \lambda_3 = \pi_j * K * \lambda_4 \tag{5.5}$$

onde:

π_i - soma das probabilidades em estado estacionário dos estados nos quais t_3 está habilitada e cujas variáveis de estado obedecem ao comportamento descrito para o protocolo;

π_j - soma das probabilidades em estado estacionário dos estados nos quais t_4 está habilitada;

K - número de receptores por grupo *multicast*;

Dada a equação 5.5, pode-se encontrar uma relação entre o tempo de processamento no roteador e o número de receptores. O tempo de processamento por pacote no roteador, t_{proc}, é igual a $1/\lambda_3$. Simula-se o aumento do número de receptores multiplicando-se a taxa λ_4 pelo número de receptores desejado, ou seja, o valor da taxa passado ao Tangram-II é igual a $K * \lambda_4$. O mesmo critério se aplica às taxas λ_7, λ_8, λ_9 e λ_{10}, referentes ao receptor. Dessa forma, o tempo de processamento esperado, por pacote, em função do número de receptores pode ser dado por:

$$t_{proc} = \pi_i/(\pi_j * K * \lambda_4) \tag{5.6}$$

Segundo Towsley [93], de forma geral, em um protocolo *multicast*, o tempo de processamento em um roteador aumenta com o número de receptores. A expressão acima mostra que, para se manter esse tempo de processamento constante, o tempo

de processamento por pacote e por receptor deve diminuir à medida que se aumenta o número de receptores. Isso implica em um aumento da velocidade de processamento e, consequentemente, da parcela de tarefas do protocolo selecionadas para implementação em *hardware* à medida que se aumenta o número de receptores, até o limite em que todo o protocolo tem que ser implementado em *hardware*, para uma dada tecnologia. Cada tipo de tecnologia de *hardware* utilizada fornece um limite superior para o desempenho do protocolo. A equação 5.6 é utilizada na função objetivo do algoritmo genético para que sejam encontrados os parâmetros do modelo.

Para se iniciar o processo de otimização são calculados 6 valores para o tempo de processamento desejado. Esses valores são obtidos a partir dos valores de atraso do evento t_3 obtidos por Lima [94] através da utilização da ferramenta Synopsys. Dependendo do tipo de pacote e da quantidade de acessos necessários à memória, esse atraso varia entre 40 ns (4 ciclos) e 330 ns (33 ciclos). O projeto deve considerar o pior caso, ou seja, 330 ns. Esse valor é obtido para o processamento de um pacote e considerando-se apenas 1 receptor. Nas simulações realizadas por Lehman [82] foram considerados grupos *multicast* com até 100 receptores. Se for considerado que o grupo pode conter até 300 receptores, obtém-se um valor máximo de aproximadamente 0.0001 seg. para o tempo de processamento no roteador ativo. Esse será o valor utilizado como limite para o tempo de processamento. Esse tempo deve permanecer constante à medida que se aumenta o número de receptores. Dessa maneira obtém-se os outros 5 valores necessários para iniciar o processo de otimização.

Neste exemplo, devido à maior complexidade do protocolo e de seu respectivo modelo, a convergência foi mais demorada e foi necessário utilizar um tamanho de população igual a 1000 indivíduos. O número de gerações utilizado como critério de terminação do algoritmo foi igual a 10. A otimização foi realizada baseada na função objetivo formada pela diferença entre os tempos de processamento obtidos para cada número de receptores utilizado (1, 5, 10, 20, 50 e 100) e os tempos de processamento desejados. Os parâmetros encontrados são apresentados na tabela 5.8.

Utilizando as taxas da tabela 5.8 e as probabilidades calculadas pelo Tangram-

Transição	Taxa (1/s)
t_1 (λ_1)	42,9
t_2 (λ_2)	2070,0
t_3 (λ_3)	9904,7
t_4 (λ_4)	3075,6
t_5 (λ_5)	2430,4
t_6 (λ_6)	7464,5
t_7 (λ_7)	5144,3
t_8 (λ_8)	9427,5
t_9 (λ_9)	8117,0
t_{10} (λ_{10})	2438,5

Tabela 5.8: Taxas da rede de Petri do protocolo ARM obtidas pelo AG.

II, pode-se aplicar o critério de partição da seção 3.3.5. Os resultados encontrados podem ser vistos na tabela 5.9.

Evento	Produto
$\lambda_1 * \pi_1$	41.50
$\lambda_2 * \pi_2$	1.55
$\lambda_3 * \pi_3$	271.33
$\lambda_4 * \pi_4$	14.12
$\lambda_5 * \pi_5$	0.50
$\lambda_6 * \pi_6$	1.55
$\lambda_7 * \pi_7$	0.41
$\lambda_8 * \pi_8$	1.54
$\lambda_9 * \pi_9$	0.80
$\lambda_{10} * \pi_{10}$	0.24

Tabela 5.9: Valores obtidos aplicando-se o critério de partição.

As tarefas do protocolo relacionadas às transições t_1, t_3 e t_4 devem ser implementadas em *hardware*, enquanto as demais transições podem ser implementadas

em *software*. Pode-se perceber, analisando os resultados, que as transições a serem implementadas em *hardware* são aquelas que tornam mais rápido o processamento de um pacote no roteador ativo.

5.4.1 Avaliação da Partição

A síntese realizada por Lima [94] utiliza a biblioteca padrão ES2 com tecnologia 0.7μm. A tabela 5.10 mostra as medidas de área e atraso obtidas para cada transição do protocolo. A partição HW/SW é avaliada considerando uma área máxima de 4 mm^2 para o *hardware*. Dessa forma, todas as transições indicadas para implementação em *hardware* podem ser implementadas em *standard cells* porque as respectivas medidas de área estão abaixo de 4 mm^2 e as medidas de atraso associadas satisfazem as taxas calculadas de AG.

Transição	**Standard Cells Área (mm^2)**	**Standard Cells Atraso**
1	0.202	44 ns
2	0.202	44 ns
3	1.668	330 ns
4	0.331	21 ns
5	0.331	21 ns
6	0.202	44 ns
7	0.202	44 ns
8	0.015	5 ns
9	0.015	5 ns
10	0.015	5 ns

Tabela 5.10: Medidas de área e atraso.

5.4.2 Análise do Desempenho da Partição Escolhida

A figura 5.9 compara os tempos de processamento em um roteador ativo em função do número de receptores para o caso onde o protocolo é totalmente implementado em SW com os tempos de processamento no caso onde o protocolo é implementado utilizando-se a partição HW/SW escolhida.

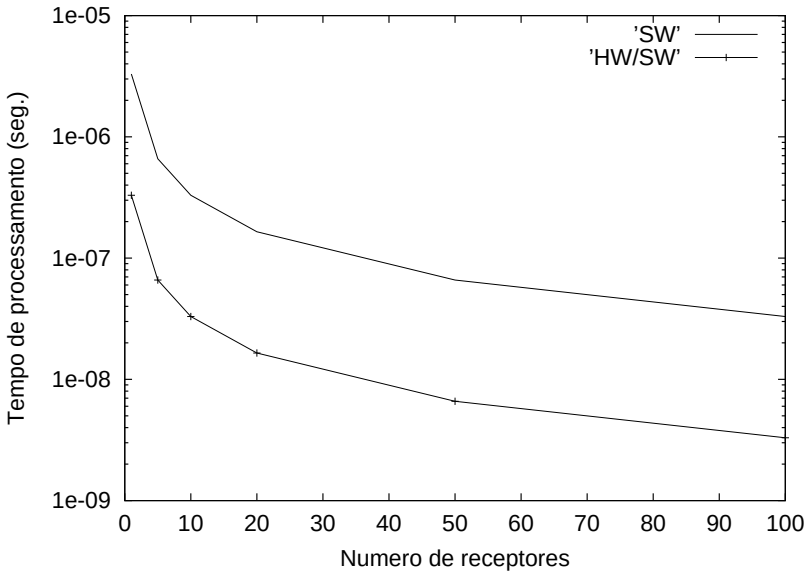

Figura 5.9: Implementação em SW versus implementação com a partição HW/SW.

Pode-se perceber que os tempos de processamento da implementação utilizando a partição HW/SW é aproximadamente 10 vezes menor que os tempos de processamento da implementação em SW, ou seja, o desempenho da partição é quase 10 vezes melhor.

A figura 5.10 compara os tempos de processamento para o caso onde o protocolo é totalmente implementado em HW com os tempos de processamento no caso onde o protocolo é implementado utilizando-se a partição HW/SW. Nesse caso, os tempos de processamento são praticamente idênticos. Esse resultado já era esperado, uma vez que as "vazões" das transições implementadas em HW são muito maiores que as "vazões" das outras transições, principalmente a "vazão" da transição t_3 (vide tabela 5.9). No entanto, a implementação que utiliza a partição HW/SW permite uma economia de aproximadamente 31% na área de HW, conforme pode ser comprovado

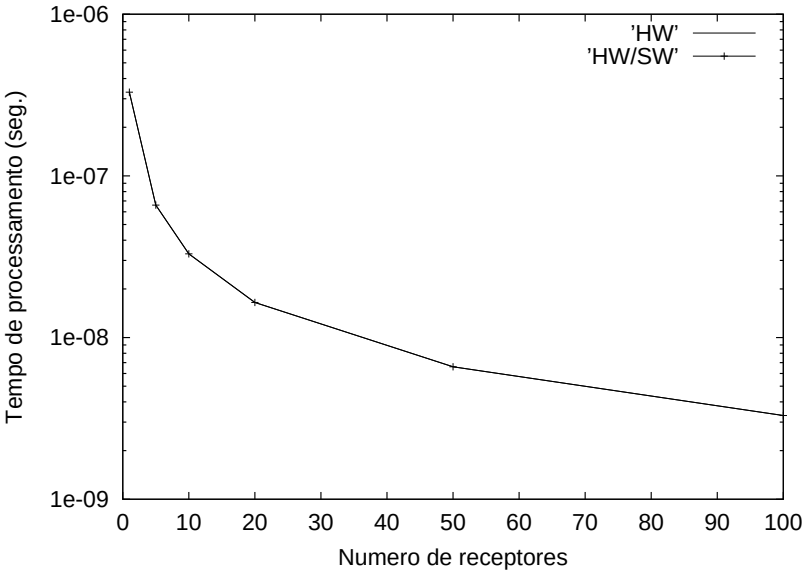

Figura 5.10: Implementação em HW versus implementação com a partição HW/SW.

analisando-se a tabela 5.10.

Foi analisado também um outro caso: a transição t_3 implementada em SW e as transições 1 e 4 implementadas em HW. Observa-se que os tempos de processamento são quase iguais aos tempos obtidos para a implementação em SW. Esse fato se deve à maior "vazão" da transição t_3 em relação às demais e indica que a maior parte do processamento se concentra nessa transição, o que já era esperado pois é a transição que verifica o tipo de pacote recebido pelo roteador ativo e realiza o processamento adequado do mesmo de acordo com os valores das variáveis de estado (vide apêndice B).

Um processo subsequente de refinamento poderia ser realizado e uma nova partição HW/SW gerada, onde somente a transição t_3 seria implementada em HW. Nesse caso, a economia de HW seria de 48%, de acordo com a tabela 5.10.

5.5 Comentários

Nesse capítulo foram apresentados e analisados os resultados obtidos com a aplicação da metodologia ao projeto de protocolos com características diferentes. Os

resultados mostram que o protocolo que mais se beneficiou da aplicação da metodologia foi o protocolo ARM. Esse resultado indica que protocolos com alto grau de processamento nos nós são aqueles mais apropriados para implementação em HW.

Para a obtenção dos resultados numéricos foi desenvolvido um programa na linguagem de programação C, que implementa a metodologia. Esse programa foi executado em um microcomputador Pentium-III 866 Mhz com 128MB de memória RAM, no sistema operacional Linux RedHat 7.2.

O algoritmo convergiu em cerca de alguns minutos para o número de pontos, populações de até 300 indivíduos e 10 gerações. Para experimentos realizados com populações de 5000 indivíduos, 10 pontos, 10 gerações e/ou um erro total de 0,00001 para a função objetivo, o algoritmo convergiu em cerca de 3 horas.

O tempo de convergência indica que o programa da ferramenta Tangram-II (**solv**) que executa o método de solução da cadeia de Markov é rápido e o algoritmo é executado, no máximo, em algumas horas no ambiente mencionado, apesar do Tangram-II poder ser chamado até centenas de milhares de vezes durante a execução do algoritmo genético, dependendo do tamanho da população, do número de pontos a serem aproximados e do número de gerações. Esse tempo é curto se considerarmos todo o processo de projeto e síntese do protocolo, principalmente pelo tempo economizado pelo projetista na escolha da melhor melhor partição HW/SW.

Capítulo 6

Conclusões

O crescente tráfego das redes atuais causado por aplicações multimídia e *multicast*, entre outras, gerou uma demanda por redes de alta velocidade e por protocolos com altas vazões. Por outro lado, verifica-se uma tendência, nos próximos anos, de uma taxa de crescimento bem maior da largura de faixa dos enlaces do que das velocidades de processamento nos nós dessas redes. Isso gera a necessidade de se investigar soluções que utilizam *hardware* na implementação dos protocolos executados nesses nós. Nesse contexto, o processo de *HW/SW codesign* têm grande importância, permitindo a detecção de erros nas etapas iniciais do projeto. No entanto, esse processo introduz um problema de alta complexidade: decidir quais tarefas do protocolo devem ser implementadas em *hardware* e quais devem ser implementadas em *software*, ou seja, definir a partição HW/SW.

A necessidade da constante interferência do projetista no processo de escolha da melhor partição HW/SW, utilizando apenas a sua experiência, torna essa escolha muito subjetiva e o processo de implementação muito lento e sujeito a erros. É importante que o projetista disponha de medidas e critérios objetivos que o auxiliem a tomar uma decisão objetiva e precisa. Com essa finalidade, a contribuição principal deste trabalho é o desenvolvimento de uma metodologia que auxilia o projetista no processo de escolha da partição HW/SW. Essa metodologia baseia-se

nos seguintes conceitos e ferramentas: *HW/SW codesign*, redes de Petri estocásticas, na ferramenta de análise de desempenho Tangram-II, nos algoritmos genéticos e em duas ferramentas de síntese de *hardware*, o Synopsys e o Altera. A integração dessas ferramentas em um único processo é um importante resultado desse trabalho. Outro fator importante que diferencia a metodologia apresentada das metodologias encontradas na bibliografia é a utilização de medidas de desempenho no processo de otimização realizado pelo algoritmo genético desde o início do ciclo de projeto. A metodologia também estabelece um critério que relaciona uma saída do AG com uma determinada partição HW/SW. Esse critério se mostra coerente ao longo dos diversos exemplos apresentados.

Na maioria dos experimentos realizados o algoritmo convergiu em algumas horas no máximo, mesmo para populações de milhares de indivíduos. Devido aos diferentes graus de sensibilidade de cada transição do modelo, ou seja, devido a alguns parâmetros influenciarem mais o valor da função objetivo do que outros, optou-se pela utilização de um algoritmo genético híbrido que varia a faixa de valores possíveis de cada parâmetro conforme a sua sensibilidade. Esse algoritmo associa técnicas tradicionais, que fazem uso de derivadas para determinar as faixas de variação permitidas para cada taxa de transição do protocolo, a um algoritmo genético simples. Isso possibilita uma convergência mais rápida do algoritmo genético para o erro especificado para a função objetivo e não provoca uma convergência prematura para um mínimo local. Apesar do algoritmo seguir uma estrutura básica de um algoritmo genético simples, foi necessário um grande desenvolvimento do mesmo para adaptá-lo à metodologia em virtude da necessidade de integração com o programa que implementa o método de solução da ferramenta Tangram-II e com o programa que substitui os parâmetros do modelo pelos valores numéricos da população gerada pelo AG.

O custo da implementação em *hardware* pode ser significativamente reduzido caso sejam utilizados dispositivos lógicos programáveis (PLDs), que aproximam o custo do *hardware* do custo do *software*, pois o *hardware* pode ser desenvolvido e testado em um programa comercial desenvolvido com essa finalidade (Altera) e sintetizado em pastilhas de baixo custo, ao contrário da implementação em *standard cells*.

Apesar da metodologia apresentada ter sido direcionada para auxiliar o processo de partição HW/SW de protocolos de comunicação, ela é totalmente genérica, de alto nível de abstração e pode ser aplicada a qualquer sistema especificado em máquinas de estado. Nesse caso pode-se realizar a verificação do modelo utilizando a ferramenta ARP. No entanto, a modelagem pode ser realizada diretamente na ferramenta Tangram-II. A metodologia pode ser utilizada no nível de rede, no nível do protocolo ou mesmo no nível de uma transição. Particularmente no último caso, pode haver necessidade de reparticionar um conjunto de operações representado por uma transição, dividindo-a em várias outras. Isso é necessário caso seja verificado na etapa do projeto de *hardware* que uma operação consome, por exemplo, 90% do tempo da transição. Nesse caso, mesmo que uma transição tenha sido selecionada para implementação em *hardware* não significa que todas as operações relativas àquela transição serão implementadas em *hardware* se for realizada uma nova partição. O exemplo da seção 5.4 mostrou a necessidade desse refinamento do modelo.

A literatura disponível sobre os esforços realizados para auxiliar o processo de partição HW/SW ainda é muito pobre. Apenas Hidalgo [14] faz uma tentativa semelhante ao processo descrito neste trabalho sem, no entanto, utilizar uma ferramenta de análise de desempenho que forneça medidas **durante** o processo de otimização.

O método de otimização baseado em algoritmos genéticos mostrou ser eficiente em relação à convergência e fácil de ser implementado e modificado. Além disso, não é necessária muita precisão no cálculo da função objetivo porque o critério de seleção adotado (SUS) é um critério aproximado. Apesar de demandar um custo computacional elevado, o algoritmo genético não é tendencioso e evita funções de erro mais complexas como, por exemplo, a função exponencial utilizada no método de otimização *simulated annealing* [49].

O exemplo do protocolo bit alternante serviu para depurar o algoritmo genético, integrá-lo à ferramenta Tangram-II e apresentar a metodologia de forma prática e didática. Já nos exemplos do capítulo 5 é avaliada a viabilidade, o potencial e a praticidade da metodologia no auxílio ao projeto de protocolos, particularmente à partição HW/SW. Foram escolhidos três tipos de protocolos, com características

diferentes, para avaliar a metodologia. No primeiro, um protocolo de controle de congestionamento do TCP, bastante utilizado nas redes atuais, são obtidos resultados coerentes com o comportamento esperado para o protocolo. No entanto, o desempenho do protocolo TCP está diretamente ligado ao número de pacotes perdidos na rede. Portanto, numa rede na qual ocorrem muitas perdas, mesmo que o protocolo seja totalmente implementado em *hardware*, não é possível garantir um alto desempenho devido às limitações impostas pelo tráfego. Mas essa situação tem mudado ultimamente devido ao crescimento acentuado das larguras de faixa dos enlaces [93] e pode-se ter, num futuro próximo, um protocolo TCP com poucas perdas decorrentes do tráfego. Nesse caso é interessante que o processamento nos nós da rede seja rápido e, portanto, a metodologia apresentada é de grande utilidade.

Em seguida, procura-se avaliar a metodologia quando aplicada a um protocolo de controle de *handoff* para redes ATM sem fio. Esse tipo de protocolo é dominado por fluxo de controle. Os resultados obtidos mostram uma nítida vantagem da implementação em *standard cells* sobre a implementação em PLDs ou *software*. Dessa maneira, reduz-se a probabilidade de perda de pacotes de dados durante a passagem de uma célula para outra, pois o tempo gasto no processo é bem menor do que se a tarefa fosse implementada em *software*. No entanto, percebe-se a importância da utilização de PLDs como uma ferramenta para prototipagem rápida, aumentando o desempenho do protocolo e mantendo os custos próximos aos de uma implementação em *software*.

Posteriormente, aplica-se a metodologia a um protocolo *Multicast* confiável onde os nós são ativos. No protocolo ARM os nós ativos possuem um processamento bastante significativo, apesar desse processamento ser distribuído. Os resultados obtidos mostram que as tarefas de processamento dos pacotes nos roteadores ativos são as que mais influenciam o desempenho do protocolo e que o critério de partição adotado fornece resultados coerentes com os esperados, apesar de ser utilizado um modelo com algumas simplificações. Esse exemplo mostra que os protocolos que mais se aproveitam da metodologia apresentada são aqueles que possuem um alto grau de processamento nos nós e cujo desempenho está diretamente relacionado à velocidade desse processamento.

A metodologia apresentada é de fácil utilização e bastante genérica, podendo ser aplicada a qualquer tipo de protocolo. Sugere-se como continuação deste trabalho o projeto de protocolos destinados às aplicações multimídia [95] e à criptografia de dados [96][97] devido à alta carga de processamento exigida por esses protocolos. Outra sugestão é o aumento do grau de automatização da metodologia, integrando ao programa algumas tarefas que ainda são realizadas manualmente como o cálculo das sensibilidades dos parâmetros do modelo (apêndice A), cujo objetivo é a determinação das faixas de variação de cada um desses parâmetros para a geração da população inicial, e o cálculo da "vazão" (produto $\lambda_i * \pi_j$) de cada transição (evento). Dessa maneira, se os dados da síntese de HW forem armazenados em um arquivo de uma forma padronizada, o programa do AG poderia ler esses dados e já fornecer a solução definitiva para a partição HW/SW ou, se for o caso, gerar uma nova população automaticamente.

Referências Bibliográficas

[1] EXTREMENETWORKS. Technology - Hardware Architecture. *Internet Draft* (2000). `http://www.extremenetworks.com`.

[2] GAJSKI, D., AND VAHID, F. Specification and Design of Embedded Software/Hardware Systems. *IEEE Design and Tests of Computer 12*, 1 (janeiro de 1995), 53–67.

[3] ISMAIL, T. B., ABID, M., AND JERRAYA, A. COSMOS: A Codesign Approach for Communication Systems. *3th International Workshop on Hardware/Software Codesign* (1994), 17–24.

[4] GAJSKI, D., VAHID, F., NARAYAN, S., AND GONG, J. System-Level Exploration with SpecSyn. *Design Automation Conference* (1998), 812–817.

[5] KALAVADE, A., AND LEE, E. A. Hardware/Software Codesign using Ptolemy - A Case Study. *Proceedings of IEEE International Workshop on Hardware/Software Codesign* (1992).

[6] KALAVADE, A., AND LEE, E. A. Hardware/Software Codesign using Ptolemy - A Case Study. In *Proceedings of the First International Workshop on Hardware/Software Codesign* (1992).

[7] Overview of the Ptolemy Project - Computer Science Department - University of California. `http://ptolemy.eecs.berkeley.edu`.

[8] MADSEN, J., GRODE, J., KNUDSEN, P. V., PETERSEN, M. E., AND HAXTHAUSEN, A. LYCOS: the Lyngby Co-Synthesis System. *Design Automation for Embedded Systems 2*, 2 (fevereiro de 1997), 1–43.

[9] Chou, P. H., Ortega, R. B., and Borriello, G. The Chinook Hardware/Software Co-Synthesis System. *8th International Symposium on System Synthesis* (setembro de 1995).

[10] Maciel, P., Barros, E., and Rosenstiel, W. Estimating Functional Unit Number in the PISH Codesign System by Using Petri Nets. In *XII Symposium on Integrated Circuits and Systems Design* (setembro de 1999), pp. 32–35.

[11] Maciel, P., Barros, E., and Rosenstiel, W. A Petri Net Based Approach for Performing the Initial Allocation in Hardware/Software Codesign. In *IEEE International Conference on Systems, Man, and Sybernetics* (outubro de 1998).

[12] Maciel, P., and Barros, E. Capturing Time Constraints by Using Petri Nets in the Context of Hardware/Software Codesign. In *7th IEEE International Workshop on Rapid System Prototyping* (junho de 1996).

[13] Fischer, S., Wytrebowicz, J., and Budkowski, S. Hardware/Software Co-design of Communication Protocols. In *Proceedinds of IEEE 22nd Euromicro Conference* (1996).

[14] Hidalgo, J. I., and Lanchares, J. Functional Partitioning for Hardware/Software Codesign using Genetic Algorithm. In *Proceedings of the 23rd Euromicro Conference* (1997).

[15] Scott, S. D., Samal, A., and Seth, S. HGA: A Hardware-Based Genetic Algorithm. In *Proceedinds of ACM/SIGDA Third International Symposium on Field Programmable Gate Arrays* (1995).

[16] Miranda, M. N. and Lima, R. N. and Pedroza, A. C. P. and Mesquita Filho, A. C. HW/SW Codesign de Protocolos Baseado na Otimização de Desempenho por Algoritmos Genéticos. In *XIX Simpósio Brasileiro de Redes de Computadores - SBRC'2001* (maio de 2001).

[17] Miranda, M. N. and Lima, R. N. and Pedroza, A. C. P. and Mesquita Filho, A. C. HW/SW Codesign of Protocols Based on Performance Opti-

mization Using Genetic Algorithms. In *17th International Teletraffic Congress - ITC'2001* (dezembro de 2001).

[18] Lima, R. N. and Miranda, M. N. and Pedroza, A. C. P. and Mesquita Filho A. C. HW/SW Codesign of Handoff Protocol for Wireless ATM Networks based on Performance Optimization using Genetic Algorithm. In *SBCCI 2002 - 15th Symposium on Integrated Circuits and System Design* (setembro de 2002).

[19] Miranda, M. N. and Lima, R. N. and Pedroza, A. C. P. and Mesquita Filho, A. C. Design of a Reliable Multicast Protocol Using HW/SW Codesign Based on Performance Optimization with Genetic Algorithms. In *IEEE International Telecommunications Symposium (ITS2002)* (setembro de 2002).

[20] Carmo, R., Carvalho, L., Sousa e Silva, E., Diniz, M., and Muntz, R. Performance/availability modeling with the Tangram-II modeling environment. *Performance Evaluation 33*, 1 (junho de 1998), 45–65.

[21] Figueiredo, D. R. Um módulo de simulação da ferramenta Tangram-II: suporte para medidas com recompensa, recursos de eventos raros e aplicações a modelos de rede multimídia. Tese de Mestrado, Programa de Engenharia de Sistemas - COPPE/UFRJ, julho de 1999.

[22] Souza e Silva, E. and Leão, R. M. M. The Tangram-II Environment. In *Computer Peformance Evaluation - Modelling Techniques and Tools - 11th International Conference (TOOLS2000)* (março de 2000).

[23] Silva, A. P. C. Tangram-II User's Manual. Relatório técnico, Universidade Federal do Rio de Janeiro, outubro de 2000. `http://www.land.ufrj.br`.

[24] IEEE Std 1076-1987. *IEEE Standard VHDL Language Reference Manual*, março de 1988.

[25] Ashenden, P. J. *The VHDL Cookbook*. Computer Science Dept. of University of Adelaide, 1990.

[26] Synopsys, Inc. *Synopsys Online Documentation, v1998.02*, 1998.

[27] ALTERA, CORP. *Data Book and Max + PlusII Getting Started*, 1997.

[28] ALTERA, CORP. *Data Book*, 1996.

[29] ALTERA, CORP. *Max + PlusII VHDL*, 1996.

[30] ALTERA, CORP. *Max + PlusII Getting Started*, 1997.

[31] CHOI, H., KULKARNI, V., AND TRIVEDI, K. Markov regenerative stochastic Petri nets. *Performance Evaluation 20*, 1 (maio de 1994), 337–357.

[32] GERMAN, R., AND LINDEMANN, C. Analysis of stochastic Petri nets by the method of supplementary variables. *Performance Evaluation 20*, 1 (maio de 1994), 317–335.

[33] ARP - Analisador/Simulador de Redes de Petri - LCMI - Departamento de Engenharia Elétrica da Universidade Federal de Santa Catarina. `http://www.ppgia.pucpr.br/ maziero/petri/manual-pt.html`.

[34] MITCHELL, M. *An Introduction to Genetic Algorithms*. MIT Press, 1996.

[35] HORNER, H. A C++ Class Library for Genetic Programming: The Vienna University of Economics - Genetic Programming Kernel. *Internet Draft* (maio de 1996). `http://www.wu-wien.ac.at/usr/h88/h8850092`.

[36] KOZA, J. *Genetic Programming II*. MIT Press, 1994.

[37] LIMA, R. N. B. Uma Metodologia para Implementação de Protocolos de Comunicação Utilizando Hardware/Software Codesign. Relatório técnico, Universidade Federal do Rio de Janeiro, dezembro de 2000. `http://www.gta.ufrj.br`.

[38] PIRMEZ, L. *Uma metodologia para síntese de alto nível de protocolos a partir de uma descrição formal*. Tese de Doutorado, Programa de Engenharia Elétrica - COPPE/UFRJ, 1996.

[39] VAHID, F., AND LEE, T. D. Towards a model for hardware and software functional partitioning. In *International Workshop on Hardware/Software Codesign* (1996).

[40] Gajski, D., Dutt, N., Wu, A., and Lin, S. *High-Level Synthesis - Introduction to Chip and System Design.* Kluwer Academic Publishers, 1991.

[41] Daveau, J., et al. COSMOS: An SDL based Hardware/Software Codesign Environment. *Hardware/Software Codesign and Coverification, Current Issues in Electronic Modelling 8* (1997).

[42] Peterson, J. L. Petri Nets*. *Computing Surveys 9*, 3 (setembro de 1977), 223–252.

[43] Diaz, M. Modeling and Analysis of Communication and Cooperation Protocols Using Petri Net Based Models. *Computer Networks 10*, 6 (junho de 1982), 419–441.

[44] Kumar, S., et al. A Framework for Hardware/Software Codesign. *IEEE Computer 26*, 12 (dezembro de 1993), 39–45.

[45] Jerraya, A., and O'Brien, K. SOLAR: An Intermediate Format for System-Level Modeling and Synthesis. *in Computer Aided Software-Hardware Engineering, J. Rozenblit, K. Buchenrieder (eds), IEEE Press* (1994).

[46] Vahid, F., Lee, T. D., and Hsu, Y.-C. A Comparison of Functional and Structural Partitioning. *International Symposium on System Synthesis* (Nov 1996), 121–126.

[47] Lee, E. A., and Messerschmitt, D. G. Synchronous Data Flow. In *IEEE Proceedings* (setembro de 1987).

[48] Ernst, R., Henkel, J., and Benner, T. Hardware-Software Cosynthesis for Microcontrollers. *IEEE Design and Tests of Computer* (dezembro de 1993).

[49] Laarhoven, A. *Simulated Annealing: theory and applications.* D. Reidel, 1987.

[50] Black, P. E. NP-hard - definition. *Internet Draft* (2001). `http://www.nist.gov/dads/HTML/nphard.html`.

[51] Alomary, A. Y. A Hardware/Software Codesign Partitioner For ASIP Design. In *ICECS'96* (1996).

[52] Gupta, R. K., and Micheli, G. Hardware-Software Cosynthesis for Digital Systems. *IEEE Design and Tests of Computer* (setembro de 1993).

[53] Gupta, R. K., Coelho, C. N., and Micheli, G. Program Implementation Schemes for Hardware-Software Systems. *IEEE Computer* (janeiro de 1994).

[54] Woo, N. S., Dunlop, A. E., and Wolf, W. Codesign from Cospecification. *IEEE Computer* (janeiro de 1994).

[55] Wirth, N. *Algorithms and Data Structures*, 1996.

[56] Pedroza, A. C. Redes de Computadores: Arquitetura e Projeto de Protocolos. Relatório técnico, Universidade Federal do Rio de Janeiro, maio de 1999.

[57] Budkowski, S. Estelle development toolset. *Computer Networks and ISDN Systems, Special Issue on FDT Concepts and Tools 25(1)* (1992).

[58] Milner, R. *Communication and Concurrency.* Prentice-Hall, 1989.

[59] Barwise, J., and Etchemindy, J. *Turing's World 3.0.* Cambridge: Cambridge University Press, 1993.

[60] Bolognesi, T., and Brinksma, E. Introduction to the ISO Specification Language LOTOS. *Computer Networks and ISDN Systems 14*, 1 (janeiro de 1988), 25–29.

[61] Bolognesi, T. and Najm, E. and Tilanus, P. A. J. G-LOTOS: A Graphical Language for Concurrent Systems. *Computer Networks and ISDN Systems 26*, 9 (1994), 1101–1127.

[62] Souza e Silva, E., and Muntz, R. *Métodos computacionais de solução de cadeias de Markov: aplicações a sistemas de computação e comunicação.* Escola de Computação 1992 - Gramado, 1992.

[63] Hirel, C and Zang, X. and Trivedi, K. S. Reliability and Performability Modeling Using SHARPE 2000. In *Computer Peformance Evaluation - Modelling Techniques and Tools - 11th International Conference (TOOLS2000)* (março de 2000).

[64] Stewart, W. J. *MARCA: Markov Chain Analyzer, a software package for Markov Chains. in: Numerical Solution of Markov Chains.* Marcel Dekker, 1991.

[65] Stewart, W. J., and Dayar, T. Comparison of Partitioning Techniques for Two-level Iterative Solvers on Large, Sparse Markov Chains. *SIAM Journal on Scientific Computing 21*, 5 (2000), 1691–1705.

[66] Hirel, C and Tuffin, B. and Trivedi, K. S. SPNP: Stochastic Petri Nets. Version 6.0. In *Computer Peformance Evaluation - Modelling Techniques and Tools - 11th International Conference (TOOLS2000)* (março de 2000).

[67] Mupala, J. K., and Ciardo, Trivedi, K. S. Stochastic Reward Nets for Reliability Prediction. *Communications in Reliability, Maintenability and Serviceability 1*, 2 (1994), 9–20.

[68] Li, S., Park, S., and Arifler, D. SMAQ: A Measurement-Based Tool for Traffic Modeling and Queuing Analysis Part I: Design Methodologies and Software Architecture. *IEEE Communications Magazine*, 36 (agosto de 1998), 56–65.

[69] Bernardo, M. Implementing Symbolic Models for Value Passing in TwoTowers. In *Computer Peformance Evaluation - Modelling Techniques and Tools - 11th International Conference (TOOLS2000)* (março de 2000).

[70] Gilmore, S. and Hillston, J. The PEPA Workbench: A Tool to Support a Process Algebra-based Approach to Performance Modelling. *Proceedings of PERFORMANCE TOOLS'94 - LNCS 794* (1994), 353–368.

[71] Hermanns, H. and Mertsiotakis, V. and Rettelbach, M. A Construction and Analysis Tool Based on the Stochastic Process Algebra TIPP. *Proceedings of TACAS'96 - LNCS 1055* (1996), 427–430.

[72] Vahid, F., Le, T. D., and Hsu, Y. A Comparison of Functional and Structural Partitioning. In *International Symposium on System Synthesis* (1996).

[73] Stevens, W. TCP Slow Start, Congestion Avoidance, Fast Retransmit and Fast Recovery Algorithms. *Internet RFC 2001* (janeiro de 1997).

[74] Comer, D. E. *Internetworking with TCP/IP - vol.1.* Prentice Hall, 1995.

[75] Peterson, L. L., and Davie, B. S. *Computer Networks: A Systems Approach.* Morgan Kaufmann Publishers, 1996.

[76] Schwartz, M. *Broadband Integrated Networks.* Prentice Hall, 1996.

[77] Leland, W., et al. On the Self-Similar Nature of Ethernet Traffic (Extended Version). *IEEE/ACM Transactions on Networking 2*, 1 (fevereiro de 1994), 1–15.

[78] Willinger, W., et al. Self-Similarity Through High-Variability: Statistical Analysis of Ethernet LAN Traffic at the Source Level. *IEEE/ACM Transactions on Networking 5*, 1 (fevereiro de 1997), 71–86.

[79] Molloy, M. K. Performance Analisys Using Stochastic Petri Nets. *IEEE Transactions on Computers 31*, 9 (setembro de 1982), 913–917.

[80] Ross, S. M. *Stochastic Processes.* John Wiley & Sons, 1983.

[81] Cheng, W. C.-W. The Tangram Graphical Interface Facility (TGIF) manual. *Internet Draft* (2000). `http://bourbon.cs.ucla.edu:8801/tgif/`.

[82] Lehman, L. H. and Garland, S. J. and Tennenhouse, D. L. Active Reliable Multicast. In *Proceedings of INFOCOM'98* (abril de 1998).

[83] Whitley, D. The Genitor Algorithm and Selective Pressure. In *Proceedings of 3rd International Conference on Genetic Algorithms* (1989).

[84] Miranda, M. N. Aproximação de Distribuições de Probabilidades não Exponenciais por Distribuições do Tipo Fase. Tese de Mestrado, Instituto Nacional de Pesquisas Espaciais - INPE, outubro de 1996.

[85] Davis, L. D. *Handbook of Genetic Algorithms*, 1991.

[86] Mathis, M., Semke, J., and Mahdavi, J. The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm. *Computer Communication Review 27*, 3 (julho de 1997), 1–16.

[87] Acharya, A., Li, J., and Bakre, A. Design and Prototyping of Location Management and Handoff Protocols for Wireless ATM networks. In *ICUPC'97* (1997).

[88] Miranda, M. N., Lima, R. N., Pedroza, A. C. P., Mesquita Filho, A. C., and Valentim Filho, J. Design of Handoff Protocols for Wireless ATM Networks. Relatório técnico, Universidade Federal do Rio de Janeiro, setembro de 2001. `http://www.gta.ufrj.br`.

[89] Yuan, R., Biswas, K., French, L. J., Li, J., and Raychaudhuri, D. A Signaling and Control Architecture for Mobility Support in Wireless ATM Networks. *MONET 1*, 3 (março de 1996), 287–298.

[90] Levine, B. N. and Garcia-Luna-Aceves, J. J. A Comparison of Reliable Multicast Protocols. *ACM Multimedia Systems Journal 6*, 5 (agosto de 1998), 334–348.

[91] Sanjoy, P. and Sabnani, K. K. and Lin, J. C. and Bhattacharyya, S. Reliable Multicast Transport Protocol. *IEEE Journal on Selected Areas in Communications 15*, 3 (abril de 1997), 407–421.

[92] Deering, S. E. Multicast Routing in Internetworks and Extended LANs. In *Proceedings of ACM SIGCOMM'88* (agosto de 1988).

[93] Towsley, D. and Kurose, J. F. and Pingali, S. A Comparison of Sender-Initiated and Receiver-Initiated Reliable Multicast Protocols. *IEEE Journal on Selected Areas in Communications 15*, 3 (1997), 398–406.

[94] Miranda, M. N., Lima, R. N., Pedroza, A. C. P., and Mesquita Filho, A. C. Design of an Active Reliable Multicast Protocol Based on Performance Optimization Using Genetic Algorithms. Relatório técnico, Universidade Federal do Rio de Janeiro, dezembro de 2001. `http://www.gta.ufrj.br`.

[95] Gonçalves, P. A. S. Um serviço ativo de distribuição de vídeo multiponto. Tese de Mestrado, Programa de Engenharia Elétrica - COPPE/UFRJ, março de 2000.

[96] Abadi, M. Two facets of authentication. In *Proceedings of the 11th IEEE Computer Security Foundations Workshop* (1998).

[97] Abadi, M. Security protocols and specifications. In *In Foundations of Software Science and Computation Structures: Second International Conference, FOSSACS '99* (1999).

Apêndice A

Método do Gradiente para a Determinação das Faixas dos Parâmetros

Dada uma função escalar de um vetor:

$$f(x) = 0 \tag{A.1}$$

onde $x \in R^n$ e:

$$f : R^n \longrightarrow R \tag{A.2}$$

Seja x^k o valor de x na iteração k. Utilizando o desenvolvimento da função em série de Taylor em torno de x^k, obtém-se:

$$f(x^{k+1}) = f(x^k) + \nabla^t f(x^k)(x^{k+1} - x^k) + ... \tag{A.3}$$

onde:

$$\Delta x = x^{k+1} - x^k \tag{A.4}$$

e

$$\nabla^t f = [\frac{\partial f}{\partial x_1} \quad \frac{\partial f}{\partial x_2} \quad ... \quad \frac{\partial f}{\partial x_n}] \tag{A.5}$$

Truncando-se a série nos termos de primeira ordem e igualando a zero, o valor de x^{k+1} pode ser obtido por:

$$\nabla^t f(x^k).(x^{k+1} - x^k) = -f(x^k) \tag{A.6}$$

e

$$\nabla^t f(x^k).\nabla f(x^k).(x^{k+1} - x^k) = -f(x^k).\nabla f(x^k) \tag{A.7}$$

$$x^{k+1} = x^k - \frac{\nabla f(x^k)}{< \nabla f(x^k).\nabla f(x^k) >}.f(x^k) \tag{A.8}$$

onde

$$\frac{\nabla f(x^k)}{< \nabla f(x^k).\nabla f(x^k) >} \tag{A.9}$$

é a inversa generalizada de $\nabla f(x^k)$. Nos problemas de otimização, o que se deseja determinar é o mínimo (máximo) de uma função, ou seja:

$$\min_x f(x)$$

. No caso particular deste trabalho, a função f é a função objetivo a ser otimizada pelo algoritmo genético, chamada de $\varepsilon(\lambda)$, onde λ é o vetor das taxas da rede de Petri que modela o protocolo. Neste caso, o valor de $\varepsilon(\lambda)$ no ponto ótimo não é necessariamente nulo, logo a equação A.8 deve ser modificada para:

$$\lambda^{k+1} = \lambda^k - \alpha.\frac{\nabla \varepsilon(\lambda^k)}{< \nabla \varepsilon(\lambda^k).\nabla \varepsilon(\lambda^k) >}.\varepsilon(\lambda^k) \tag{A.10}$$

onde $\alpha \in (0, 1]$ é um fator de amortecimento. Dado que no processo de otimização não se conhece o valor analítico de $\nabla\varepsilon$ utiliza-se uma aproximação discreta. No contexto deste trabalho, as componentes do gradiente são denominadas de *sensibilidades*, sendo definidas, no caso geral, por:

$$S^{\varepsilon}_{\lambda_i} = \frac{\varepsilon(\lambda_i + \Delta\lambda_i) - \varepsilon(\lambda_i)}{\Delta\lambda_i} \tag{A.11}$$

O vetor de sensibilidades $S^{\varepsilon}_{\lambda}$, dado por:

$$S^{\varepsilon}_{\lambda} = \begin{bmatrix} S^{\varepsilon}_{\lambda_1} \\ S^{\varepsilon}_{\lambda_2} \\ \cdots \\ S^{\varepsilon}_{\lambda_n} \end{bmatrix} \tag{A.12}$$

é utilizado em substituição ao gradiente. O módulo do vetor de sensibilidades é dado por:

$$\|S_\lambda^\varepsilon\|^2 = (S_\lambda^\varepsilon)^t . S_\lambda^\varepsilon \tag{A.13}$$

Pela série de Taylor, uma variação de $\delta\lambda_i$ no parâmetro λ_i produzirá uma variação no erro dada por:

$$\varepsilon({\lambda_i}^k + \delta{\lambda_i}^k) \simeq \varepsilon({\lambda_i}^k) + S_{{\lambda_i}^k}^\varepsilon . \delta{\lambda_i}^k \tag{A.14}$$

Se $S_{\lambda_i}^\varepsilon < 0$ isto implica que $\varepsilon(\lambda_i^k + \delta\lambda_i^k)$ pode ser feito nulo para um valor $\delta\lambda_i^k$ positivo dado pela equação A.10 com $\alpha = 1$, ou seja:

$$\lambda_i^{k+1} = \lambda_i^k - \frac{S_{\lambda_i^k}^\varepsilon}{\|S_{\lambda^k}^\varepsilon\|^2} \cdot \varepsilon(\lambda_i^k) \tag{A.15}$$

Supondo que o limite de variação de λ_i na iteração k é dado por FA_i^k, adotou-se a heurística de estender a faixa de variação de λ_i na iteração $k+1$ até o limite:

$$FA_i^{k+1} = FA_i^k - \frac{\varepsilon(\lambda_i)}{\|S_{\lambda^k}^\varepsilon\|^2} \cdot S_{\lambda_i}^\varepsilon \tag{A.16}$$

onde FA_i^k é a faixa de variação dos parâmetros na iteração k e FA_i^{k+1} é a faixa de variação na iteração $k+1$. Note que no caso de $S_{\lambda_i}^\varepsilon > 0$ o limite será reduzido. Após a determinação das faixas de variação ótimas para os parâmetros executa-se o programa de AG da seção 4.3.5.

Resumindo o algoritmo:

- Estabelecer uma faixa inicial única e pequena para todos os parâmetros;
- Executar o AG e ao fim de 10 gerações verificar o erro;
- Variar cada uma das taxas de 1%, uma de cada vez, e avaliar o valor de erro produzido por cada parâmetro;

- A partir dos valores obtidos, calcular as sensibilidades em relação a cada parâmetro, segundo a equação A.11;
- Calcular o módulo do vetor de sensibilidades a partir da equação A.13;
- Calcular a nova faixa de variação de cada parâmetro a partir da equação A.16
- Repetir as operações anteriores até que todos os valores calculados para as sensibilidades não alterem mais a faixa corrente

Apêndice B

Modelo do Tangram-II para o Protocolo ARM

Declaration=

Var

State: fp1,fp2,fp3,fp4,fp5,fp6,fp7,fp8,fp9,fp10,fp11,n_links;

State: tipo_pct;

State: RR_found;

State: Cache_a,DP_found;

State: NR_found;

Param

Float: lambda1,lambda2,lambda3,lambda4,lambda5;

Float: lambda6,lambda7,lambda8,lambda9,lambda10;

name=ARM

Events=

event= Em_envia_pct(EXP,lambda1)

condition= (fp1 > 0)

```
action= {
int p1,p7,p11,tipo,NACK_count_NP,Em_count;
p1=0;
p7=0;
p11=0;
tipo=1;
NACK_count_NP=0;
Em_count=0;
tipo=tipo_pct;
if (tipo == 3)
    tipo=2;
p1=fp1;
p7=fp7;
p1=p1-1;
p7=p7+1;
p11=p11+1;
set_st("fp1",p1);
set_st("fp7",p7);
set_st("fp11",p11);
set_st("tipo_pct",tipo);
};

    event= Em_processa_NP(EXP,lambda2)
condition= (fp3 > 0)
action= {
int p1,p3,tipo;
p1=0;
p3=0;
tipo=3;
p1=fp1;
p3=fp3;
```

```
p3=p3-1;
p1=p1+1;
set_st("fp1",p1);
set_st("fp3",p3);
set_st("tipo_pct",tipo);
};


    event= Rot_processa_pct(EXP,lambda3)
condition= ((fp7 > 0) && (fp9 > 0))
action= {
int p7,p8,p9,p10,p11, tipo,cache,STO_SN,subsc_links,temp;
int found_NR,found_RR,found_DP,NACK_count_RR,NACK_count_NR;
int NACK_count_DP,NACK_count_NP,NACK_count_RP;


    temp=0;
p7=0;
p8=0;
p9=0;
p10=0;
p11=0;
tipo=0;
subsc_links = 1;
cache=1;
STO_SN=0;
found_NR=0;
found_RR=0;
found_DP=0;
NACK_count_RR=0;
NACK_count_NR=0;
NACK_count_DP=0;
NACK_count_NP=0;
```

```
NACK_count_RP=0;
cache=Cache_a;
p7=fp7;
p8=fp8;
p9=fp9;
p10=fp10;
p11=fp11;
tipo=tipo_pct;
found_NR = NR_found;
found_RR = RR_found;


    switch(tipo)
{
/** ROTEADOR PROCESSA UM DP **/
    case 1:
if (cache)
{
    found_DP = 1;
cache = 0;
    }
p7=p7-1;
p9=p9-1;
p8=p8+1;
break;


    /** ROTEADOR PROCESSA UM RP **/
    case 2:


    if (cache)
```

```
{
    found_DP=1;
cache = 0;
if (found_NR)
found_NR=0;
    }
p7=p7-1;
p9=p9-1;
p8=p8+1;
if (!found_RR)
    found_RR=1;
    break;

    /** ROTEADOR PROCESSA UM NP **/
case 3:
if (DP_found)
    {
tipo = 2;
p7=p7-1;
p9=p9-1;
p10=p10-1;
p8=p8+1;
p11=p11+1;

    if (!found_RR)
    found_RR=1;
    }
else
```

```
{
    if (NR_found)
    subsc_links = subsc_links + 1;
    else
{
    if (!NR_found)
found_NR = 1;
    subsc_links = subsc_links + 1;
p7=p7-1;
p9=p9-1;
p8=p8+1;
}
}
break;
default: temp=temp+1;
}
set_st("fp7",p7);
set_st("fp8",p8);
set_st("fp9",p9);
set_st("fp10",p10);
set_st("fp11",p11);
set_st("Cache_a",cache);
set_st("tipo_pct",tipo);
set_st("NR_found",found_NR);
set_st("RR_found",found_RR);
set_st("DP_found",found_DP);
set_st("n_links",subsc_links);
};
```

```
    event= Rot_Envia_Rec(EXP,lambda4)
condition= ((fp8 > 0) && (fp11 > 0))
action= {
int p6,p8,p9,p11;
p6=0;
p8=0;
p9=0;
p11=0;
p6=fp6;
p8=fp8;
p9=fp9;
p11=fp11;
p8=p8-1;
p11=p11-1;
p6=p6+1;
p9=p9+1;
set_st("fp6",p6);
set_st("fp9",p9);
set_st("fp8",p8);
set_st("fp11",p11);
};


    event= Rot_env_NP_Em(EXP,lambda5)
condition= ((fp8 > 0) && (fp10 > 0) && (tipo_pct == 3))
action= {
int p8,p3,p9,p10;


    p8=0;
p3=0;
p9=0;
p10=0;
```

```
p3=fp3;
p8=fp8;
p9=fp9;
p10=fp10;
p8=p8-1;
p10=p10-1;
p3=p3+1;
p9=p9+1;
set_st("fp3",p3);
set_st("fp8",p8);
set_st("fp9",p9);
set_st("fp10",p10);
};

    event= Descarta_NP(EXP,lambda6)
condition= ((tipo_pct == 3) && (fp8 > 0) && (fp10 > 0) && (RR_found))
action= {
int p1,p8,p9,p10,tipo;

    /* DESCARTA NACK */
p1=0;
p8=0;
p9=0;
p10=0;
tipo=1;
p1=fp1;
p8=fp8;
p9=fp9;
p10=fp10;
p8=p8-1;
p10=p10-1;
```

```
p1=p1+1;
p9=p9+1;
set_st("fp1",p1);
set_st("fp8",p8);
set_st("fp9",p9);
set_st("fp10",p10);
set_st("tipo_pct",tipo);
};

    event= Rec_processa_pct(EXP,lambda7)
condition= (fp6 > 0)
action= {
int p5,p6;

    p5=0;
p6=0;
p6=fp6;
p5=fp5;
p6=p6-1;
p5=p5+1;
set_st("fp6",p6);
set_st("fp5",p5);
};

    event= Rec_envia_NP(EXP,lambda8)
condition= (fp5 > 0)
action= {
int p5,p7,p10,tipo,NACK_count_NP,N_SN;

    NACK_count_NP=0;
```

```
N_SN=0;
p5=0;
p7=0;
p10=0;
tipo=3;
p5=fp5;
p7=fp7;
p10=fp10;
p5=p5-1;
p7=p7+1;
p10=p10+1;
set_st("fp5",p5);
set_st("fp7",p7);
set_st("fp10",p10);
set_st("tipo_pct",tipo);
};

    event= Libera_Em(EXP,lambda9)
condition= (fp2 > 0)
action= {
int p1,p2,p4;
p1 = 0;
p2 = 0;
p4 = 0;
p1 = fp1;
p2 = fp2;
p4 = fp4;
p2 = p2 - 1;
p4 = p4 + 1;
p1 = p1 + 1;
set_st("fp1",p1);
```

```
set_st("fp2",p2);
set_st("fp4",p4);
};

    event= Rec_pct_OK(EXP,lambda10)
condition= ((fp5 > 0) && (fp4 > 0))
action= {
int p2,p4,p5,tipo;

    p2=0;
p4=0;
p5=0;
tipo=1;
p2=fp2;
p4=fp4;
p5=fp5;
p5=p5-1;
p4=p4-1;
p2=p2+1;
set_st("fp2",p2);
set_st("fp4",p4);
set_st("fp5",p5);
set_st("tipo_pct",tipo);
};

    event= Libera_cache(EXP,10)
condition= (Cache_a == 0)
action= {
int cache,subsc_links,STO_SN,found_NR,found_RR;
int found_DP,NACK_count_RR;
```

```
    cache=1;
subsc_links = 1;
STO_SN=0;
found_NR=0;
found_RR=0;
found_DP=0;
set_st("Cache_a",cache);
set_st("NR_found",found_NR);
set_st("RR_found",found_RR);
set_st("DP_found",found_DP);
set_st("n_links",subsc_links);


    };


    Rewards=


    Initialization=
fp1=1
fp4=1
fp2=0
fp3=0
fp5=0
fp6=0
fp7=0
fp8=0
fp9=1
fp10=0
fp11=0
n_links=1
```

tipo_pct=1

RR_found=0

Cache_a=1

DP_found=0

NR_found=0

State_vars=