HW/SW Codesign de um Protocolo Multicast Confiável Baseado na Otimização de Desempenho por Algoritmos Genéticos

Marcio N. Miranda[1] **, Ricardo N. B. Lima**[2] **,**
Aloysio C. P. Pedroza[2] **, Antônio C. Mesquita Filho**[2]

[1]Instituto de Informática – Universidade Federal de Goiás
Bloco IMF I - sala 227 Campus II - Samambaia 74001-970 Goiânia, GO

[2]Universidade Federal do Rio de Janeiro
Centro de Tecnologia - bloco H - sala 321
Caixa Postal 68504 - Ilha do Fundão 21945-970 Rio de Janeiro, RJ

miranda@inf.ufg.br,belem,aloysio@gta.ufrj.br,mesquita@coe.ufrj.br

Abstract. *This work presents a design methodology based on HW/SW Codesign and genetic algorithms. The design of a reliable multicast protocol is used to show how this methodology works. The main objective of this methodology is to find a faster implementation with lower costs from the HW/SW Codesign technique. The protocol performance is analyzed by a modelling environment called Tangram-II. The area of the hardware corresponding to a given implementation is calculated by the Synopsys tool. The methodology is intended as a tool to help protocol designers to select the best performance/cost compromise.*

Resumo. *Neste trabalho é apresentada uma metodologia de HW/SW Codesign de protocolos de comunicação baseada em técnicas de análise de desempenho e algoritmos genéticos. Um protocolo multicast confiável foi utilizado como estudo de caso para apresentar todo o ciclo de projeto. O principal objetivo é encontrar uma implementação que atenda a requisitos de desempenho especificados pelo projetista. O desempenho do protocolo é analisado pela ferramenta de modelagem Tangram-II. O custo do hardware associado a uma dada implementação é calculado pela ferramenta de síntese de hardware Synopsys. A metodologia pretende auxiliar o projetista de protocolos na seleção da melhor partição HW/SW.*

1. Introdução

O projeto de protocolos *multicast* confiáveis na Internet é um problema de difícil tratamento devido à capacidade limitada tanto do emissor quanto da rede de responder a perdas de pacotes. Requisições simultâneas de retransmissões relizadas por um grande número de receptores podem levar a uma sobrecarga da rede, causando o problema conhecido como *implosão de NACKs*. Além disso, receptores de um grupo *multicast* podem ter diferentes taxas de perda de pacotes, dependendo de sua localização na árvore *multicast*.

A retransmissão de dados para todo o grupo, quando somente uma parte dos receptores está sofrendo perda de dados, causa um desperdício de banda passante e degrada o desempenho da rede. Com o objetivo de tornar a recuperação de perdas escalável para grandes grupos são necessários mecanismos de otimização para controlar a implosão de NACKS, distribuir a carga das retransmissões e limitar o escopo dos pacotes retransmitidos somente aos receptores que estão experimentado perdas [1].

Um fator importante para melhorar o desempenho de protocolos com alto custo computacional, como é o caso dos protocolos *multicast* confiáveis, é a integração entre *software* (SW) e *hardware* (HW) durante a sua implementação. Em geral, as tarefas do protocolo que não exigem restrições de tempo podem ser implementadas em SW enquanto aquelas que requerem maior velocidade devem ser implementadas em HW. Devido à crescente demanda por protocolos com altas vazões, soluções que utilizam HW na implementação estão sendo cada vez mais investigadas. Por exemplo, roteadores totalmente implementados em circuitos integrados de aplicação específica (ASICs) fornecem vazões até 10 vezes maior do que aqueles implementados por microprocessadores de uso geral [2].

O princípio básico do *Codesign* é a realização de um projeto cooperativo baseado em dois ambientes de projeto específicos, HW e SW, onde pode-se verificar e simular todo o sistema em qualquer etapa do projeto. Durante o 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.

Apesar dos tempos de execução em HW serem, normalmente, muito menores do que na implementação em SW, o custo deste tipo de solução é maior. Portanto, deve-se estabelecer um compromisso entre o desempenho desejado e o custo de implementação no momento de se decidir qual parte do protocolo deve ser implementada em HW e qual parte deve ser implementada em SW. Esta etapa é denominada de *particionamento* e sua decisão tem impacto direto no desempenho do sistema de comunicação do qual o protocolo faz parte.

Normalmente, os projetistas de protocolos não se preocupam com aspectos de desempenho *durante* o projeto e se baseiam na sua experiência pessoal e em técnicas informais para escolher a melhor partição HW/SW, o que limita consideravelmente a exploração do espaço de soluções. Neste caso, deve-se fornecer ao projetista ferramentas de análise de desempenho com o objetivo de obter medidas que indiquem a eficiência de uma determinada partição HW/SW. Embora existam alguns esforços para automatizar este processo, são poucos os recursos para auxiliar no refinamento da especificação, desenvolvimento e partição do protocolo [3][4].

Muitos grupos de pesquisa em renomadas universidades estão desenvolvendo ambientes de projeto baseados na metodologia de *Codesign*. Dentre eles pode-se citar: COSMOS [5], SpecSyn [6], Ptolemy [7][8][9], LYCOS [10], Chinook [11] e PISH [12]. Estes ambientes de *Codesign* diferem na linguagem de especificação, no método de particionamento, na arquitetura alvo e nos métodos de validação utilizados.

Fischer [4] ressalta a importância do uso combinado do HW e do SW para se alcançar um alto desempenho de sistemas distribuídos, como sistemas multimídias. É apresentado um ambiente de desenvolvimento para o suporte das etapas de projeto e

implementação. Hidalgo [13] propõe uma metodologia para a partição HW/SW baseada no uso de algoritmos genéticos. A divisão dos blocos funcionais em HW e SW é baseada na avaliação de uma função objetivo. Esta função utiliza uma tabela previamente estabelecida para os valores dos desempenhos, não calculando esses parâmetros durante o processo de busca da melhor partição, como é realizado no presente trabalho.

O projeto se baseia no uso da ferramenta de modelagem e análise de desempenho Tangram-II [14][15], na ferramenta de síntese Synopsys [16] e no uso de um algoritmo genético como método de otimização. O comportamento do protocolo é representado por um diagrama de estados, onde cada transição de estado representa uma operação ou um conjunto de operações a serem avaliadas, de maneira a selecionar a partição HW/SW que possua a melhor relação desempenho/custo. Em trabalho publicado anteriormente a metodologia é aplicada ao projeto de um protocolo dominado por fluxo de controle, como é o caso de um protocolo de *handoff* para redes sem fio [17]. Este trabalho implementa o protocolo ARM (*Active Reliable Multicast*) [1] usando a metodologia de projeto citada acima, ou seja, analisa a metodologia de projeto quando aplicada a um protocolo com grande carga de processamento nos nós, como é o caso do protocolo ARM.

A seção 2 descreve suscintamente o protocolo ARM. A seção 3 apresenta uma descrição das principais técnicas utilizadas na metodologia. Essas técnicas são descritas na seção 4. Na seção 5 são apresentados os resultados obtidos. A seção 6 apresenta os comentários finais.

2. O protocolo ARM

Active Reliable Multicast (ARM) [1] é um protocolo projetado para recuperação de perdas em transmissões *multicast* confiáveis em larga escala. Roteadores intermediários são usados para proteger o emissor e a rede de tráfego de pacotes de reparo desnecessários. Esse processo leva a uma queda significante no consumo de banda passante. Os roteadores ativos empregam três estratégias de recuperação de perdas: supressão de NACKs duplicados, recuperação local de perdas e *multicast* parcial. A supressão de NACKs duplicados reduz o número de NACKs trafegando em direção ao emissor e cruzando enlaces de gargalo. A recuperação local reduz a latência fim-a-fim e distribui a carga de retransmissão entre os roteadores. Roteadores ativos colocados em locais estratégicos da rede, armazenam pacotes 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* [18], 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 re-

ceptores 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 subseqüentes 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 trabalho, 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.

A figura 1 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*. A especificação do protocolo ARM é utilizada como base para a construção do modelo paramétrico do Tangram-II. Como os nós da rede são ativos, o modelo do Tangram-II é bem detalhado. O comportamento do roteador ativo, descrito nos parágrafos anteriores, cujos detalhes podem ser encontrados em Lehman [1], está representado no modelo do Tangram-II que pode ser visto em Miranda [19].

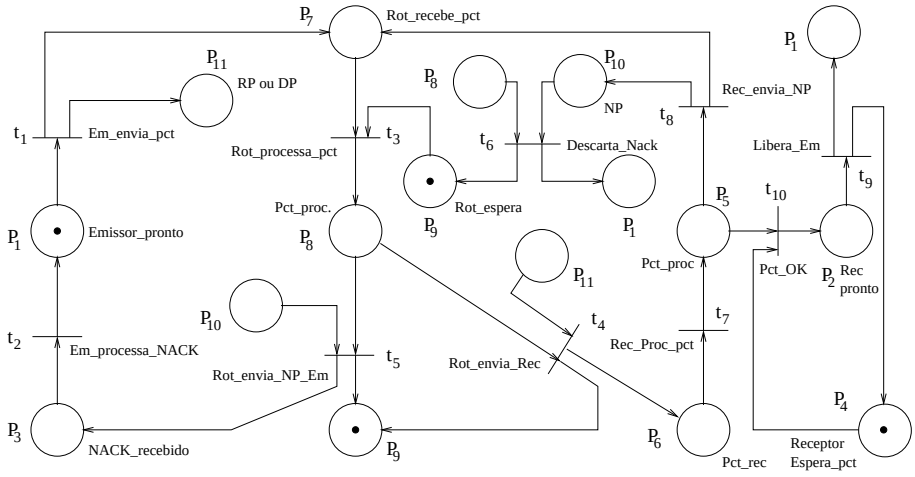

Figura 1: O protocolo ARM.

3. Técnicas de projeto

A metodologia apresentada neste trabalho, congrega uma série de técnicas em um único processo e propõe uma solução para o problema da automatização da partição HW/SW. O ponto de partida consiste na adoção de um projeto cooperativo, HW/SW *Codesign*, baseado em dois ambientes de projeto específicos.

A modelagem e a análise de desempenho de um determinado protocolo é realizada através do uso da ferramenta Tangram-II. O Tangram-II é uma ferramenta utilizada para

especificação e análise de desempenho de sistemas. Entre suas características principais estão: possuir uma interface gráfica baseada no paradigma de orientação a objetos e uma variedade de métodos de solução para obtenção de medidas de interesse relacionadas ao modelo. O sistema a ser modelado é representado por uma coleção de objetos, cujo estado 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. Maiores detalhes sobre a sintaxe da ferramenta Tangram-II podem ser obtidos em Silva [14].

Um algoritmo genético é utilizado visando a otimização de uma função objetivo composta por parâmetros de desempenho da partição. Algoritmos genéticos (AGs) são usados como ferramenta de otimização para minimizar uma função objetivo composta dos parâmetros de desempenho. Os AGs [20][21] vêm tendo larga aceitação devido à simplificação que eles permitem na formulação e solução de problemas de otimização. Esta característica é particularmente útil em problemas de otimização complexos, envolvendo um grande número de variáveis e, conseqüentemente, espaços de solução de dimensões elevadas. Além disso, em muitos casos onde outras estratégias falham na busca de uma solução, os AGs convergem.

O processo de solução adotado nos AGs consiste em gerar, aleatoriamente, um grande número de indivíduos, uma *população*, de forma a promover uma varredura tão extensa quanto necessária do espaço de soluções. Após a inicialização da população, cada iteração do AG corresponde à aplicação de um conjunto de quatro operações básicas: cálculo de aptidão, seleção, cruzamento e mutação. Ao fim destas operações 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. O tamanho da população e o número de gerações definem diretamente o tamanho do espaço de busca a ser coberto.

O projeto de circuitos integrados se inicia a partir de uma descrição comportamental do sistema na linguagem VHDL, usada como entrada da ferramenta Synopsys. O Synopsys produz como resultado um circuito lógico do qual são extraídas informações referentes ao custo de implementação e medidas de atraso.

4. Particionamento HW/SW do protocolo ARM

A figura 2 apresenta o diagrama de blocos da metodologia. Inicialmente, 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. Nesta fase, o diagrama de estados é analisado com o objetivo de se detectar *live-locks* e *dead-locks*. A partir desta especificação, o fluxo de projeto segue dois caminhos paralelos. No primeiro, a especificação é detalhada e o modelo é construído utilizando-se a ferramenta Tangram-II. Os parâmetros do protocolo que atendem aos requisitos de desempenho especificados são determinados pelo Tangram-II e pelo algoritmo genético. No segundo caminho são obtidas as medidas de atraso e custo através da ferramenta Synopsys, que fornece como saída um circuito lógico a partir do qual pode-se extrair as medidas de área e atraso.

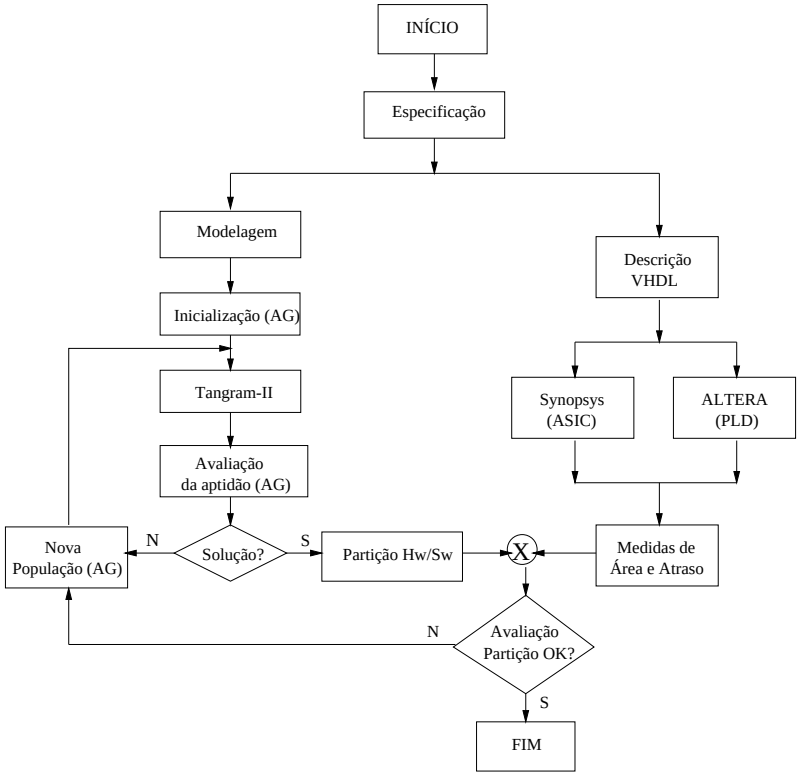

Figura 2: Metodologia de projeto.

4.1. Otimização do Desempenho

A partir da especificação do protocolo em máquina de estados é realizada a sua modelagem na ferramenta Tangram-II, através de um modelo com parâmetros, que são as taxas dos *eventos*. O problema inicial consiste em determinar as taxas de transição de estado que fornecem os requisitos de desempenho especificados. A determinação dessas taxas fornecem subsídios para o estabelecimento de um critério de partição.

Os valores numéricos necessários para resolver o modelo do Tangram-II são fornecidos por um método de otimização baseado em AGs que, inicialmente, gera conjuntos de valores para os parâmetros de forma aleatória. Após a substituição dos parâmetros por esses valores numéricos, o Tangram-II resolve a cadeia de Markov associada ao modelo. Estas probabilidades são utilizadas para estimar o desempenho do protocolo através do valor de uma função objetivo que é avaliada para cada conjunto de parâmetros gerado pelo AG, ou seja, para cada *indivíduo* gerado pelo AG. O valor da função objetivo fornece a *aptidão* de um determinado indivíduo para resolver o problema.

Nesta etapa inicia-se o processo de otimização, onde uma nova *população* é gerada para ser avaliada pelo Tangram-II, até que uma solução seja encontrada. As taxas encontradas como solução são utilizadas para obter uma partição HW/SW que é avaliada pelas medidas de área e atraso fornecidas pelo Synopsys. Caso a área e o atraso da partição atendam à especificação, a solução foi encontrada, caso contrário uma nova população é gerada.

4.2. Implementação em HW

A especificação do protocolo em máquina de estados serve como referência para a entrada das descrições comportamentais em VHDL no Synopsys. Cada transição de estado, contendo uma ou mais operações do protocolo, é associada a uma *entidade* em VHDL. Cada *entidade* é sintetizada, de modo a se obter as medidas de atraso e área referentes às transições de estado quando implementadas em HW. A ferramenta Synopsys sintetiza um circuito lógico a partir da descrição comportamental de cada transição em VHDL. Esta implementação é realizada utilizando-se uma biblioteca de células padrão (*standard cells*) numa dada tecnologia. As etapas de alocação e roteamento de células e otimização são automáticas e se tornam "transparentes" para o projetista.

4.3. A 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 [13]. 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:

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

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. As medidas de desempenho utilizadas dependem dos requisitos de qualidade de serviço desejados. O problema de otimização consiste em determinar λ^* tal que:

$$\varepsilon(\lambda^*) = \min_{\lambda} \varepsilon(\lambda) \tag{2}$$

4.4. O Critério de Partição

Com o objetivo de auxiliar o projetista no particionamento é fundamental estabelecer um critério que relacione as taxas calculadas pelo AG com uma determinada partição HW/SW. Quanto maior a taxa de um determinado evento maior o número de vezes, por segundo, que um determinado conjunto de operações do protocolo é executado. Deve-se considerar, a partir dos resultados obtidos resolvendo-se a cadeia de Markov associada, as probabilidades em estado estacionário de cada estado do sistema. Ou seja, mesmo que um evento tenha uma alta taxa, a probabilidade do sistema estar num estado em que o mesmo está habilitado pode ser muito baixa.

A partir das considerações realizadas acima foi estabelecido como critério de partição HW/SW dos eventos, onde cada evento representa um conjunto de operações do protocolo, o produto $\lambda_i * \pi$, onde:

λ_i - taxa do evento i determinada pelo AG.

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

O critério de partição apresentado é usado como uma estimativa inicial da partição HW/SW e fornece subsídios ao projetista para uma escolha objetiva da melhor partição. Essa partição é avaliada através das medidas de área e atraso pela ferramenta Synopsys.

4.5. Algoritmo Genético Implementado

O algoritmo utilizado neste trabalho é *híbrido*, pois mistura AGs com métodos de otimização tradicionais, que utilizam derivadas na determinação das faixas iniciais.

O algoritmo desenvolvido é um AG simples, onde o cromossomo foi codificado como um vetor de variáveis reais que representam os parâmetros (taxas dos eventos) a serem determinados. A dimensão do vetor depende do número de taxas que se deseja determinar, conforme o modelo construído para o protocolo em questão.

A população inicial é obtida por uma função que gera números reais aleatoriamente, utilizando um banco de sementes e o relógio do computador. Os valores numéricos correspondentes a cada indivíduo são transferidos pelo AG ao modelo de parâmetros do Tangram-II. Desta forma é possível executar um dos métodos numéricos de solução da cadeia de Markov existentes no Tangram-II através de uma rotina específica, que é chamada pelo AG, e determinar para cada indivíduo da população as probabilidades em estado estacionário da cadeia de Markov, obtendo-se as respectivas medidas de interesse para cada um destes indivíduos. A aptidão bruta é determinada através do valor da função objetivo, dado pela soma de erros entre os valores das medidas obtidas e os valores desejados, conforme a equação 1. Ao final do processo de otimização, o melhor indivíduo é usado para definir a partição HW/SW, a qual é avaliada pela ferramenta de síntese de HW.

4.5.1. Custo computacional do AG

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.

5. Resultados

Nesta seção são apresentados os resultados obtidos no projeto do protocolo ARM utilizando a metodologia apresentada. São realizadas a análise de desempenho, escolha da partição HW/SW e avaliação final do desempenho da partição escolhida.

5.1. Avaliação do desempenho

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.

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:

$$\pi_i * \lambda_3 = \pi_j * K * \lambda_4 \tag{3}$$

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 3, 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:

$$t_{proc} = \pi_i/(\pi_j * K * \lambda_4) \tag{4}$$

Segundo Towsley [22], 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, conseqüentemente, da parcela de tarefas do protocolo selecionadas para implementação em HW à medida que se aumenta o número de receptores, até o limite em que todo o protocolo tem que ser implementado em HW, para uma dada tecnologia. Cada tipo de tecnologia de HW utilizada fornece um limite superior para o desempenho do protocolo. A equação 4 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 fornecidos pela 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 [1] 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.

5.2. Escolha da Partição

Devido à complexidade do protocolo, utilizou-se 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 1.

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

Utilizando as taxas da tabela 1 e as probabilidades calculadas pelo Tangram-II, pode-se aplicar o critério de partição da seção 4.4. Os resultados encontrados podem ser vistos na tabela 2.

Evento	**Produto (Vazão)**
$\lambda_1 * \pi_1$	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 2: 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 HW, enquanto as demais transições podem ser implementadas em SW. Pode-se perceber, analisando os resultados, que as transições a serem implementadas em HW são aquelas que tornam mais rápido o processamento de um pacote no roteador ativo.

5.3. Avaliação da Partição

A síntese realizada utiliza a biblioteca padrão ES2 com tecnologia 0.7μm. A tabela 3 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 HW. Dessa forma, todas as transições indicadas para implementação em HW 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 pelo 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 3: Medidas de área e atraso.

5.4. Análise do Desempenho da Partição Escolhida

A figura 3 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.

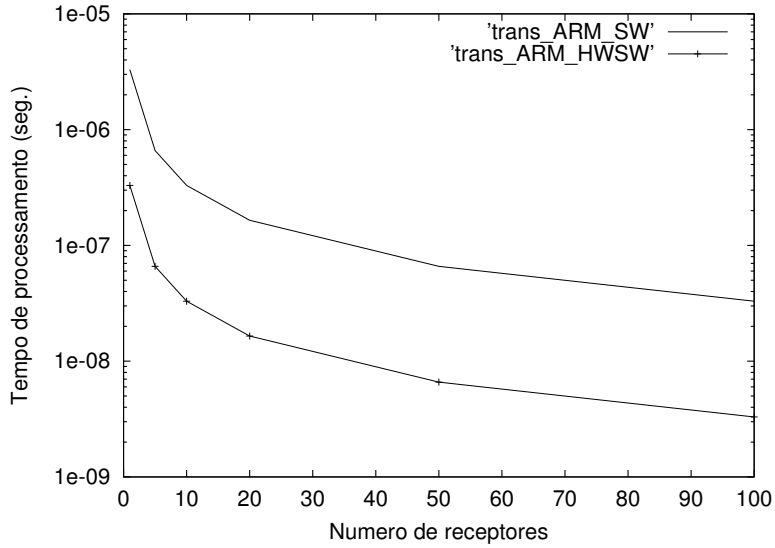

Figura 3: Implementação em SW versus implementação com a partição HW/SW.

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 bem melhor.

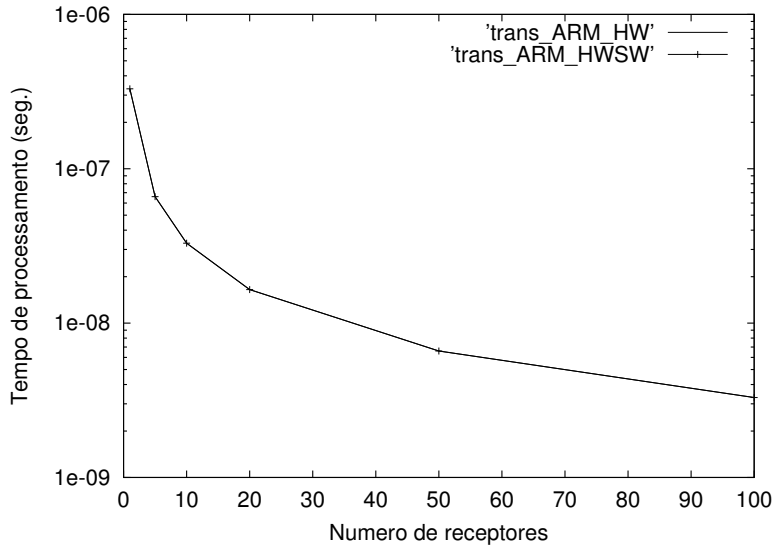

Figura 4: Implementação em HW versus implementação com a partição HW/SW.

A figura 4 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 e, desta forma, as curvas ficam superpostas. 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 2). No entanto, a implementação que utiliza a partição HW/SW permite uma economia de aproximadamente 31% na área de HW em relação a uma implementação toda em HW, conforme pode ser comprovado somando-se as áreas referentes à cada transição da tabela 3 para ambos os casos.

Foi analisado também um outro caso: a transição t_3 implementada em SW e as transições 1 e 4 implementadas em HW. Verificou-se [19] 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. Um processo subseqüente 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 3, procedendo-se da mesma maneira mencionada no parágrafo anterior.

6. Comentários Finais

A transmissão *multicast* confiável na internet é um problema de difícil tratamento devido à capacidade limitada tanto do emissor quanto da rede de responder a sinais de perda de pacotes. Com o objetivo de tornar escalável a recuperação de perdas para grandes grupos *multicast*, são necessários mecanismos para controlar a implosão de NACKs, dis-

tribuir a carga de retransmissões e limitar o escopo dos pacotes retransmitidos. O protocolo ARM utiliza roteadores intermediários para reduzir tanto o número de NACKs quanto o tráfego de pacotes de reparo. Esses roteadores também ajudam a distribuir a carga de retransmissão atrvés do armazenamento local dos pacotes.

Outro aspecto importante para melhorar o desempenho de protocolos de alta velocidade é a integração entre o SW e o HW durante a implementação. Devido à crescente demanda por protocolos com altas vazões, soluções utilizando HW têm sido cada vez mais investigadas. Protocolos que possuem um alto custo computacional e cujo desempenho está diretamente relacionado à velocidade de processamento nos nós da rede são aqueles que aproveitam melhor a metodologia descrita nesse trabalho. Esse é o caso do protocolo ARM, como foi visto.

Foi apresentado o projeto do protocolo ARM utilizando a metodologia descrita com o objetivo de encontrar uma implementação mais rápida e com um baixo custo. A integração da ferramenta Tangram-II com os AGs foi fundamental para simplificar a formulação do problema. Embora os AGs tenham um alto custo computacional, eles são de implementação simples e evitam a utilização de funções de erro complexas como, por exemplo, a função exponencial utilizada no método de otimização *Simulated Annealing* [23].

São fornecidas ao projetista medidas e critérios objetivos que o auxiliam a tomar decisões no processo de partição HW/SW. Não existe na literatura um ambiente de HW/SW Codesign que utilize análise de desempenho e algoritmos genéticos **durante** o ciclo de projeto de protocolos de comunicação.

O processo de partição tem um impacto direto no desempenho do sistema de comunicação onde o protocolo está inserido. Os resultados obtidos mostram que a partição HW/SW escolhida melhora em 10 vezes o desempenho do protocolo em relação à implementação em SW e o custo desta implementação é menor do que uma implementação em HW. Benefícios significativos podem ser obtidos se for realizado um refinamento da partição escolhida, reduzindo ainda mais os custos.

Referências

[1] Lehman, L. H. and Garland, S. J. and Tennenhouse, D. L., "Active Reliable Multicast", in *Proceedings of INFOCOM'98*, Apr. 1998.

[2] Extremenetworks, "Technology - hardware architecture", *Internet Draft*, 2000. `http://www.extremenetworks.com`.

[3] D. Gajski and F. Vahid, "Specification and design of embedded software/hardware systems", *IEEE Design and Tests of Computer*, vol. 12, no. 1, pp. 53–67, Jan. 1995.

[4] S. Fischer, J. Wytrebowicz and S. Budkowski, "Hardware/software co-design of communication protocols", in *Proceedinds of IEEE 22nd Euromicro Conference*, 1996.

[5] T. B. Ismail, M. Abid e A. Jerraya, "COSMOS: A Codesign Approach for Communication Systems", *3th International Workshop on Hardware/Software Codesign*, pp. 17–24, 1994.

[6] D. Gajski, F. Vahid, S. Narayan and J. Gong, “System-Level Exploration with SpecSyn”, *Design Automation Conference*, pp. 812–817, 1998.

[7] A. Kalavade e E. A. Lee, “Hardware/Software Codesign using Ptolemy - A Case Study”, *Proceedings of IEEE International Workshop on Hardware/Software Codesign*, 1992.

[8] A. Kalavade and E. A. Lee, “Hardware/software codesign using ptolemy - a case study”, in *Proceedings of the First International Workshop on Hardware/Software Codesign*, 1992.

[9] C. S. D. U. of California, “Overview of the ptolemy project”, *Internet Draft*, 2001. `http://ptolemy.eecs.berkeley.edu/`.

[10] J. Madsen, J. Grode, P. V. Knudsen, M. E. Petersen and A. Haxthausen, “Lycos: the lyngby co-synthesis system”, *Design Automation for Embedded Systems*, vol. 2, no. 2, pp. 1–43, Feb. 1997.

[11] P. H. Chou, R. B. Ortega and G. Borriello, “The chinook hardware/software co-synthesis system”, in *8th International Symposium on System Synthesis*, 1995.

[12] P. Maciel, E. Barros e W. Rosenstiel, “Estimating Functional Unit Number in the PISH Codesign System by Using Petri Nets”, in *XII Symposium on Integrated Circuits and Systems Design*, pp. 32–35, Sept. 1999.

[13] J. I. Hidalgo and J. Lanchares, “Functional partitioning for hardware/software codesign using genetic algorithm”, in *Proceedings of the 23rd Euromicro Conference*, 1997.

[14] A. P. C. Silva, “Tangram-ii user’s manual”, tech. rep., Universidade Federal do Rio de Janeiro, Oct. 2000. `http://www.land.ufrj.br`.

[15] R. Carmo, L. Carvalho, E. Sousa e Silva, M. Diniz and R. Muntz, “Performance/availability modeling with the Tangram-II modeling environment”, *Performance Evaluation*, vol. 33, no. 1, pp. 45–65, June 1998.

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

[17] Miranda, M. N. and Lima R. N. and Pedroza, A. C. P. and Mesquita Filho, A. C., “Projeto de Protocolos Utilizando HW/SW Codesign Baseado na Otimização de Desempenho por Algoritmos Genéticos”, in *XX Simpósio Brasileiro de Telecomunicações - SBT2003*, Oct. 2003.

[18] Deering, S. E., “Multicast Routing in Internetworks and Extended LANs”, in *Proceedings of ACM SIGCOMM’88*, Aug. 1988.

[19] M. N. Miranda, *Uma Metodologia de HW/SW Codesign de Protocolos de Comunicação Baseada na Otimização de Desempenho por Algoritmos Genéticos*. PhD thesis, Programa de Engenharia Elétrica - COPPE/UFRJ, 2002.

[20] M. Mitchell, *An Introduction to Genetic Algorithms*. MIT Press, 1996.

[21] H. Horner, “A C++ class library for genetic programming: The Vienna University of Economics - genetic programming kernel”, *Internet Draft*, May 1996. `http://www.wu-wien.ac.at/usr/h88/h8850092`.

[22] 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*, vol. 15, no. 3, no. 3, pp. 398–406, 1997.

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