Design of a Reliable Multicast Protocol Using HW/SW Codesign Based on Performance Optimization with Genetic Algorithms

Marcio N. Miranda, Ricardo N. B. Lima, Aloysio C. P. Pedroza, Antônio C. M. Filho
Universidade Federal do Rio de Janeiro, Rio de Janeiro RJ, Brasil

Abstract— **The design of a reliable multicast protocol using a methodology based on performance analysis techniques and genetic algorithms is presented. The main objective of this work is to find a faster implementation with lower costs from the HW/SW Codesign technique. The process requires a choice between the tasks to be implemented in hardware and those to be implemented in software, i.e., the selection of the best HW/SW partition. This methodology uses a genetic algorithm to optimize an error function defined by the required performance and the implementation cost of the protocol. 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.**

I. Introduction

RELIABLE multicast over the Internet is a difficult problem to deal with due to the limited capacity of both sender and the network for responding to reports of data losses. Simultaneous retransmission requests from large numbers of receivers can lead to sender and network overload, causing the NACK implosion problem. Additionally, receivers in a multicast group may experience widely different packet loss rates depending on their locations in the multicast tree. Having the sender retransmit to the entire group when only a small subset of the receivers experience losses wastes network bandwith and degrades overall performance. In order to scale loss recovery to large multicast groups, software optimization of efficient mechanisms are needed to control NACK implosions, to distribute the load for retransmissions, and to limit the delivery scope of retransmitted packets [1].

Another important factor to improve performance of high computational costs protocols, such as reliable multicast protocols, is the integration between software (SW) and hardware (HW) during their implementation. In general, no timing constraints tasks can be implemented in SW while those that require higher performance must be implemented in HW. Due to the increasing demand for high throughput protocols, solutions using HW are being more frequently investigated. For example, full ASIC-based forwarding engines deliver more than 10-times higher throughput than routers that rely entirely on microprocessors [2].

The basic principle of *Codesign* is the development of a cooperative design based on two specific project environments, HW and SW, where verification, test and simulation can be performed at any design level. During the specification refinement it is useful and necessary to apply techniques that aid the designer to decide between the tasks to be implemented in SW and those to be implemented in HW, i. e., the selection of the best HW/SW partition. One should note that because of the higher HW implementation costs, upon deciding for a given partition, a compromise between the desired performance and the cost of the implementation must be taken into account. This *partitioning process* has a direct impact on the performance of the communication system where the protocol is inserted.

Generally, protocol designers are not concerned with performance issues *during* the design process and relies on their own experience and informal techniques to define a HW/SW partition, which considerably limits the extent of the solutions space search. In this case it is useful to provide the designer with performance evaluation tools, in order to derive quantitative measures of the effectiveness of a given partition. In spite of the existing efforts to automate this process, yet there are few resources to aid the specification refinement, development and partitioning of a protocol [3][4].

Several projects currently in progress are trying to integrate both HW and SW into the same design process: COSMOS [5], SpecSyn [6], Ptolemy [7], LYCOS [8], Chinook [9] e PISH [10]. Fischer [4] proposes an integrated HW/SW design to meet high performance in distributed systems, such as multimedia systems. It presentes an environment for development and support of the design and implementation stages. Hidalgo [11] proposes a methodology for HW/SW partitioning based on genetic algorithms (GAs) [12]. The choice between HW and SW functional blocks are based on the value of an objective function using tabulated performance values for HW and SW. The performance parameters are not calculated during the search of the best partition, unlike it is done in this work.

This work implements the Active Reliable Multicast (ARM) protocol [1] using HW/SW Codesign methodology based on performance requirements. The design relies on the use of Tangram-II [13][14], a performance analysis tool, on the Synopsys [15] synthesis tool and on a genetic algorithm (GA) as optimization method. The protocol behavior is represented by a state diagram, where each state transition is a set of clauses and expressions to be evaluated, in order to select the best HW/SW partition in terms of a performance/cost compromise.

Section II briefly describes the ARM protocol. Section III presents a detailed description of the main design techniques used in the methodology which is described in section IV. In section V the results obtained are presented. Final remarks are drawn on Section VI.

Grupo de Teleinformática e Automação, Universidade Federal do Rio de Janeiro, COPPE/EE - Programa de Engenharia Elétrica, Caixa Postal 68504 - CEP 21945-970 - Rio de Janeiro - RJ, Brasil, Phone: +55 21 2260 5010, Fax: +55 21 2290 6626. E-mails: {miranda,belem,aloysio}@gta.ufrj.br.

II. The ARM Protocol

ARM is a loss recovery scheme for large-scale reliable multicast. Intermediate routers are used to protect the sender and network from unnecessary feedback and repair traffic. This leads to a significant drop in bandwidth consumption. Active routers employ three error recovery strategies: duplicate NACKs suppression, router-based local recovery scheme and partial multicasting. Supressing duplicate NACKs reduces the number of NACKs travelling back to the source and the traffic crossing the bottleneck links. Local recovery reduces the end-to-end wide-area latency and distributes the load of retransmissions. Active routers placed at strategic locations perform "best-effort" caching of data for possible future retransmission. Usually, active routers are placed just before lossy links. This scheme allows receiving end-hosts to quickly recover from packet losses. Partial multicasting reduces the network bandwidth consumption limiting the scope of retransmitted data [1].

It is assumed that the network provides IP-multicast style multicast routing, in which a tree rooted at the sender is formed to deliver multicast packets. ARM is receiver-reliable, i.e., the receivers are responsible for detecting and requesting lost packets which are numbered. Receivers detect losses by sequence gaps in the data packets. One should consider a scenario where there is one sender and multiple receivers in the multicast group.

In ARM, a receiver sends a NACK towards the sender as soon as it detects a loss. Multiple NACKs from different receivers are cached and "fused" at active routers along the multicast tree. The sender responds to the first NACK by multicasting a repair to the group. Subsequent NACKs are ignored for this packet for a fixed amount of time. When a router receives a NACK, indicating that a receiver has detected a packet loss, it retransmits the requested packet if that packet is in its cache; otherwise, it considers forwarding the NACK towards the sender. Active routers control implosion by dropping duplicate NACKs and forwarding only one NACK upstream towards the source per multicast subtree. Moreover, active routers perform partial multicasting of retransmitted packets so that they are delivered only to receivers that previously requested them.

This paper is interested in investigating the behavior of the active router processing time as a function of the number of receivers per group for each partition HW/SW determinated by the genetic algorithm. Figure 1 presents the specification of the ARM protocol given by a Petri net.

III. Design Techniques

The implemented methodology involves the use of many techniques in a unique process. The process begins with the adoption of a cooperative design, HW/SW Codesign, based on two specific environments. The modelling and performance analysis of the protocol are done using the Tangram-II tool, which is a general purpose modelling tool to specify and analyse the computer and communication systems performance. The main features of this tool are: graphical interface based on the object oriented paradigm and a set of solution methods to obtain the measures related to the model.

The partitioning determines which functions will be implemented in HW and which ones will be implemented in SW. Clustering, iterative-improvement and GAs are some examples of algorithms used in the partitioning process.

GAs are used as optimization tool to minimize an objective function of the performance parameters. GAs have been widely used due to the simplification that it introduces in the formulation and solution of optimization problems. This feature is particularly useful in complex optimization problems, involving a large number of variables and, consequently, solution spaces of high dimensions. Moreover, in many cases where other strategies fail in finding a solution, genetic algorithms converge.

The process of solution adopted in the GAs consists in generating, through specific rules, a large number of individuals, called *population*, in order to cover the solution space as much as possible. After the initialization, each iteration of the GA corresponds to the application of a set of four basic operations: fitness calculation, selection, cross-over and mutation. At the end of these operations, a new population is generated, which is expected to represent a better approximation to the solution of the optimization problem.

The HW synthesis is performed using the VHDL language [16]. The behavioral description of the protocol in VHDL is used as input to the Synopsys tool, that give as output a logic circuit from which the delay and area measurements can be obtained.

IV. HW/SW Partitioning of the ARM Protocol

Figure 2 illustrates the design methodology. Initially, the protocol is described by a state machine, where each state transition represents a task or a set of tasks to be performed. In this phase, the state diagram is analyzed in order to verify any live-locks or dead-locks. At this stage, the design process follows two parallel branches. In the first one, the specification is detailed and the model is built using the Tangram-II environment. The parameters of the model which meet the performance requirements are found by the use of Tangram-II tool and a genetic algorithm (GA). GA is used as an optimization tool to minimize an objective function of the performance parameters. In the second branch the delay and area measures are obtained by Synopsys tool. Such tool gives, as output, a logic circuit from which the delay and area measurements can be obtained.

A. Performance Optimization

The protocol specified by a state machine is used to build up the model in the syntax of the Tangram-II. This is a parametric model, where the parameters are the exponential rates of events that describe the behavior of the protocol. The problem is to find the state transition rates which meet the performance requirements. These rates help the designer to establish a partition criterion.

The numerical values needed to solve the Tangram-II model are given by the GA through a random number generator. Tangram-II uses these numerical values to calculate the steady state probabilities of the Markov chain associated to the model. These probabilities are used to estimate the protocol performance through the value of an objective function evaluated for each set of parameters of the model, i.e., each individual generated by the GA. The value of the objective function being the *fitness* of an individual.

At this stage, GA begins the optimization process, generating a new *population* to be evaluated by Tangram-II until a solu-

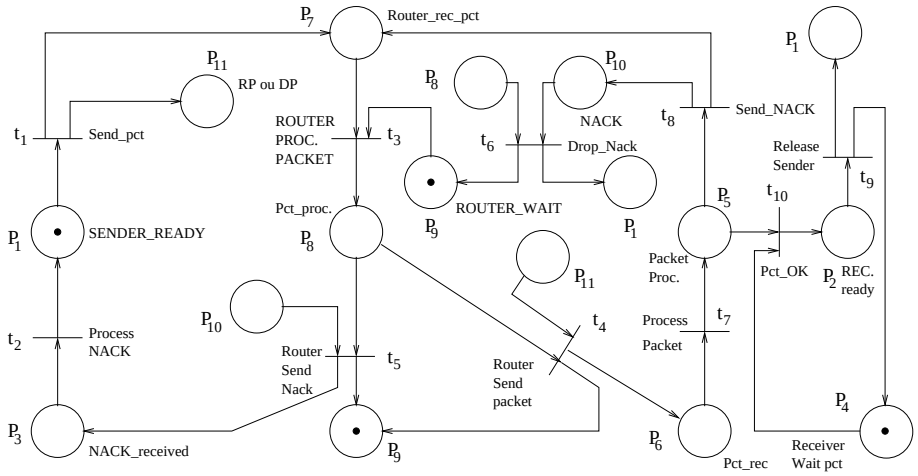

Fig. 1. The ARM protocol specification given by a Petri Net.

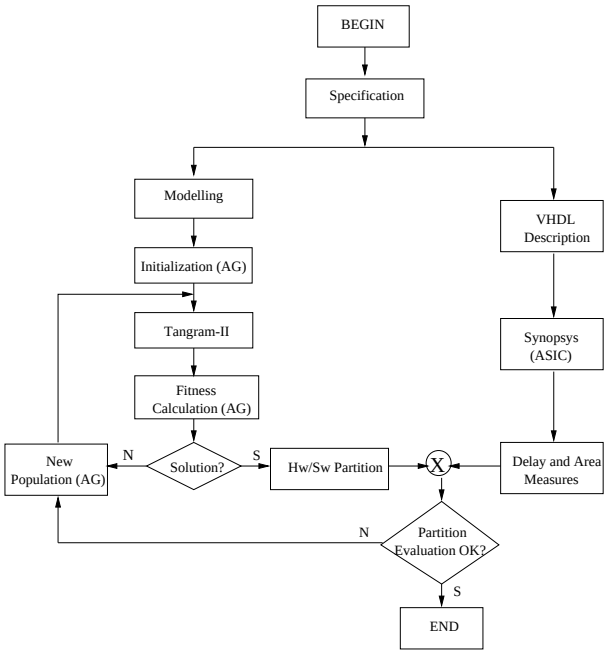

Fig. 2. The design methodology.

tion is found. The rates found at a solution are used to obtain a HW/SW partition that will be evaluated by the delay and area measures supplied by Synopsys tool. If the delay and area of the HW partitions meet the design specification, the solution is found, otherwise another population is generated.

B. HW Implementation

The VHDL behavioral descriptions specified from the protocol state machine are inputs to Synopsys tool. Each state transition that performs a set of operations is associated to a VHDL *entity*. For each transition a logic circuit is synthesized by the Synopsys tool. This implementation is obtained using a standard cell library for a integration specific technology. The cells allocation and routing process and the optimization step are made automatically.

At the end of synthesis process, delay and area measures for each transition are obtained. These measures are used to evaluate if the area and delay ofthe chosen partition are within the bounds calculated by the GA.

C. Objective Function

In order to formulate the optimization problem, the multiple parameters must be combined into a single *cost* or *objective function*, whose value defines the quality of a partition. Once one parameter generally competes with another, it is useful to associate a weighting factor to each parameter in the objective function. These weighting factors are used to equalize possible differences between the parameters sensitivities. Thus, the objective function allows to compare two partitions and to select those that satisfy the constraints. The used performance parameters depend on the QoS requirements. The objective function used in this work has the following general structure:

$$F(x_1, ..., x_n) = \sum_{1 \leq i \leq n} w_i * |x_{id} - x_{io}| \qquad (1)$$

where:
x_{id} - desired values for the performance parameters;
x_{io} - calculated values for the performance parameters;
w_i - weights atributed to each parameter.

D. The Partition Criterion

In order to assist the designer in the partitioning process, a criterion must be established to correlate the state transition rates calculated by the GA with a specified HW/SW partition. The value of the rate is proportional to the number of times that a set of protocol tasks is performed. But even if an event has a high rate, the probability of the system being on a state in which this event is enabled may be low.

Regarding the above consideration, a partition criterion has been established as product $\lambda_i * \pi_i$, where:

λ_i - rate of the event i determined by the GA;

π_i - sum of the steady state probabilities for which the event i is enabled.

The presented partition criterion is used as an initial guess of the best HW/SW partition. This partition is analyzed through the delay and area results obtained from Synopsys.

E. The implemented algorithm

The implemented optimization algorithm is a hybrid genetic algorithm since discrete gradient information is used to determine bounds to the values of the parameters along the algorithm iterations. The parameters are initially constrained to vary in small intervals and, after a few steps of the algorithm, the bounds of the parameter values are adjusted regarding the sensitivity of the objetive function to each parameter.

In the GA, the chromosome is coded as a real vector containing the state transition rates of the protocol model to be determined. The dimension of the chromosome vector is a function of the number of transitions. The chromosome is used by TANGRAM-II to calculate the steady state probabilities of the Markov chain assotiated to the protocol model, which enables the calculation of each individual fitness in the GA. The objective function or raw fitness is given by equation 1. At the end of the GA the best individual is used to define a HW/SW partition which is evaluated for area and delay by the hardware synthesis tool.

V. RESULTS

In this section, the obtained results with the design of the ARM protocol using HW/SW Codesign methodology based on performance optimization with GA are presented. The performance evaluation, choice of the partition, partition evaluation processes and a final performance analysis of the partition are discussed.

A. Performance Evaluation

This section analyses the performace of a chosen HW/SW partition that implements the active reliable multicast (ARM) protocol, specified by figure 1. The performance analisys will be focused on the packet processing time at an active router as a function of the number of receivers per multicast group. To find a relation between these two measures, one should consider the number of data and repair packets processed by time unit at an active router plus the NACK packets processed when the router has the packet in cache. In all these cases, the router will send a packet to the receiver. If the sum of the steady state probabilities in which these cases are true (π_i) is multiplied by the rate of the transition t_3 (λ_3), it can be obtained a value called the "throughput" for the transition t_3 for the mentioned cases. This value must be equal to the "throughput" of the transition t_4, i.e.:

$$\pi_i * \lambda_3 = \pi_j * K * \lambda_4 \quad (2)$$

where:

π_i - sum of the steady state probabilities for transition t_3, for the mentioned cases;

π_j - sum of the steady state probabilities in which t_4 is enabled;

K - number of receivers per multicast group.

Given equation 2, we can find a relation between the processing time (T_{proc}) at the router and the number of receivers. The packet processing time at the router, T_{proc}, is equal to $1/\lambda_3$. The used rate by Tangram-II is equal to $K * \lambda_4$, i. e., the increase in receivers number is simulated multiplying λ_4 by the desired receivers number. The same criterion is used in the receivers rates, λ_7, λ_8, λ_9 and λ_{10}. Then, the packet processing time is given by:

$$T_{proc} = \pi_i/(\pi_j * K * \lambda_4) \quad (3)$$

In a multicast protocol, when the number of receivers increases the processing time increases and the throughput decreases [17][18]. Equation 3 shows that to keep a maximum specified processing time constant, the packet processing time per receiver must decrease when the number of receivers increases. This means that the number of tasks selected to be implemented in HW must increases when the number of receivers increases, until the limit in which the hole protocol must be implemented in HW.

Equation 3 will be used by the objective function of the GA in order to fit the specification established by the designer. To begin the optimization process, six values for the desired processing time are calculated. It was considered that a multicast group contains 300 receivers. This number gives a maximum value for the processing time equal to 0.0001 second. Even when the number of receivers increases this value must stay constant.

B. Choice of the Partition

The population size has 1000 individuals and the criterion used to stop the GA was a number of generations equal to 10. The objective function was calculated based on the difference between the obtained and the desired processing time for different receivers number. The results are presented in table I.

TABLE I
RATES FOR THE PROTOCOL EVENTS FOUND BY GA.

Event	Rate (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

Using the rates of table I and the steady state probabilities calculated by Tangram-II, one can apply the partition criterion of section IV-D and calculate the product $\lambda_i * \pi_i$ for each event of the protocol. The results can be seen in table II.

Tasks of the protocol related to the transitions t_1, t_3 and t_4 must be implemented in HW because these transitions have a higher $\lambda_i * \pi_i$ product, while those related to the other events, with a lower product, must be implemented in SW. This is the first partition to be evaluated by Synopsys. If the HW area meets the specification and the rates found by GA can be implemented by the synthetized circuits, the solution is found. Otherwise, a new population is generated to continue the optimization process. One should note that, according to the results, the transi-

TABLE II
VALUES FOR PARTITION CRITERION ESTABLISHED.

Event	Value
$\lambda_1 * \pi_1$	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

tions to be implemented in HW are the ones which speed up the processing of the packet at the router.

C. Partition Evaluation

The VHDL behavioral descriptions of each state transition of the protocol are the inputs to the Synopsys tool. The synthesis was performed using a standard cell library ES2 with technology 0.7μm. Table III shows the delay and area measures obtained for each protocol transition.

TABLE III
DELAY AND AREA MEASURES.

Transition	SYNOPSYS Area (mm^2)	SYNOPSYS Delay
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

Concerning the results presented in table III, the proposed partition in section V-B was evaluated. One might consider, for example, a specified area cost of 4 mm^2 for HW, it can be noted that all the transitions indicated by the partitioning process may be implemented in HW because the area measures are below 4 mm^2 and the associated delay measures satisfy the rates calculated by GA.

D. Performance Analysis of the Chosen Partition

Once the chosen partition was evaluated, Tangram-II uses as input the calculated measures given by Synopsys in order to obtain the final performance of the protocol implementation. Figure 3 compares the ARM protocol performance entirely implemented in SW with the performance of the protocol implemented by the chosen HW/SW partition. This comparison is done using the processing times at an active router versus the number of receivers, in both cases.

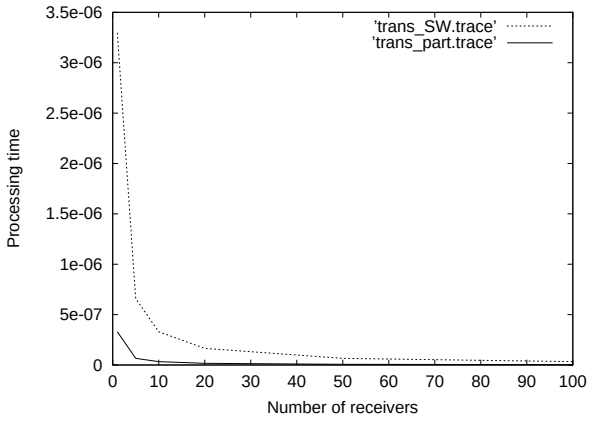

Fig. 3. Software implementation versus chosen partition.

One can note that processing times of the implemented protocol using the chosen partition is almost 10 times lower than processing times of the SW implementation. For example, when the number of receivers is equal to 100 the processing time using the chosen partition is 3.296 ns, while using SW implementation this time is 32.963 ns.

Figure 4 compares the ARM protocol performance entirely implemented in HW with the performance the protocol implemented by the chosen HW/SW partition. One can note that processing times are almost equal. This fact is already expected once the "throughputs" ($\pi_i * \lambda_i$) of the transitions implemented in HW are very higher than the "throughputs" of the other transitions.

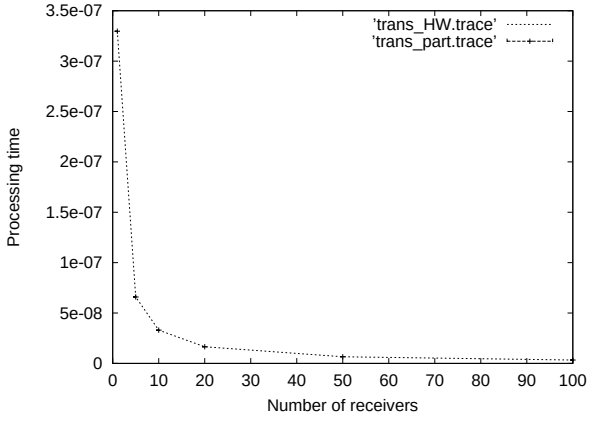

Fig. 4. Hardware implementation versus chosen partition.

Another case was analyzed: transition 3 implemented in SW and transitions 1 e 4 implemented in HW. It was observed that processing times are almost equal to SW implementation processing times. This fact is due to the higher "throughput" of the transition 3 compared with transitions 1 e 4. A refinement process could be performed and a new HW/SW partition generated with only transition 3 implemented in HW, reducing the HW costs.

VI. Final Remarks

Reliable multicast over the Internet is a difficult problem to deal with due to the limited capacity of both sender and the network for responding to reports of data losses. In order to scale loss recovery to large multicast groups, software optimization of efficient mechanisms are needed to control NACK implosions, to distribute the load for retransmissions, and to limit the delivery scope of retransmitted packets. ARM protocol utilizes intermediate routers to reduce both NACKs and repair traffic. Routers also help to distribute the retransmission load by caching multicast data.

Another important issue to improve performance of high speed protocols is the integration between SW and HW during their implementation. Due to the increasing demand for protocols with high throughput, solutions using hardware are being more frequently investigated. Protocols that have high computational costs with their performance directly related to the speed of the processing at the nodes of a network are the best candidates to have the proposed methodology applied. This is the case of the ARM protocol like we have seen.

The design of ARM protocol using a methodology based on performance analysis techniques and genetic algorithms is presented with the objective of finding a faster implementation with a low cost. The described methodology integrates many powerful techniques and simplifies the problem formulation. In this context, the integration of GA with Tangram-II was fundamental. Although the GA has a high computational cost, it is simple to be implemented and avoids complex error functions like, for example, the exponential function used in *Simulated Annealing* [19]. The designer is supplyed with measurements and objective criteria in order to assist him in taking design decisions. There is no other HW/SW Codesign environment in literature that uses performance optimization and genetic algorithms **during** the design cycle of communication protocols.

The partitioning process has a direct impact on the performance of the communication system where the protocol is inserted. The obtained results show that chosen HW/SW partition increases 10 times the protocol performance compared to the SW implementation and the cost of this HW/SW implementation is lower than a HW implementation. Significant benefits can be obtained performing a refinement on the chosen partition, reducing the HW costs.

Acknowledgments

The authors would like to express their gratitude to Professor Edmundo Souza e Silva, coordinator of the LAND laboratory of COPPE/UFRJ and to all his team, for providing the computational resources and technical support related to the Tangram-II tool. We would like to thank the coordinators of LPC and GTA laboratories for providing all the others resources for this work. This work was partially supported by grants from UFRJ, FUJB, CNPq, CAPES, COFECUB, FAPERJ and REENGE.

References

[1] L. Lehman, S. Garland and D. Tennenhouse, "Active reliable multicast", in *Proceedings of IEEE INFOCOM98*, 1998.

[2] Extremenetworks *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 codesign of communication protocols", in *Proceedinds of IEEE 22nd Euromicro Conference*, 1996.

[5] T. B. Ismail, M. Abid and A. Jerraya, "Cosmos: A codesign approach for communication systems", in *3th International Workshop on Hardware/Software Codesign*, pp. 17–24, 1994.

[6] D. Gajski and F. Vahid, "Specification and design of embedded hardware/software systems", in *IEEE Design and Tests of Computer*, pp. 56–67, 1995.

[7] A. Kalavade and E. A. Lee, "Hardware/software codesign using ptolomey - a case study", in *International Workshop on Hardware/Software Codesign*, 1992.

[8] J. Madsen, J. Grode, P. V. Knudsen, M. E. Petersen and A. Haxthausen, "Lycos: The lyngby co-synthesis system", in *Design Automation for Embedded System*, 1997.

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

[10] P. Maciel, E. Barros and 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.

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

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

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

[14] 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.

[15] Synopsys, Inc., *Synopsys Online Documentation, v1999*, 1999.

[16] IEEE Std 1076-1987, *IEEE Standard VHDL Language Reference Manual*, Mar. 1988.

[17] D. Towsley, J. F. Kurose and S. Pingali, "A Comparison of Sender-Initiated and Receiver-Initiated Reliable Multicast Protocols", *IEEE Journal on Selected Areas in Communications*, vol. 15, no. 3, pp. 398–406, 1997.

[18] J. Nonnenmacher, E. Biersack and D. Towsley, "Parity based loss recovery for reliable multicast transmission", in *Proceedings of ACM SIGCOMM'97*, 1997.

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