Asociación Argentina de Mecánica Computacional

Mecánica Computacional Vol XXIX, págs. 9243-9251 (artículo completo)
Eduardo Dvorkin, Marcela Goldschmit, Mario Storti (Eds.)
Buenos Aires, Argentina, 15-18 Noviembre 2010

INVESTIGATING MIGRATION STRATEGIES ON A NETWORK-ON-CHIP BASED PARALLEL GENETIC ALGORITHM

Rubem Euzébio Ferreira[a], Luiza de Macedo Mourelle[b] and Nadia Nedjah[c]

[a]*Center of Informatics, Universidade do Estado do Rio de Janeiro gfrubem@yahoo.com*

[b]*Departamento de Engenharia de Sistemas e Computação, Faculdade de Engenharia, Universidade do Estado do Rio de Janeiro, ldmm@eng.uerj.br, http://www.eng.uerj.br~ldmm*

[c]*Departamento de Engenharia Eletrônica e Telecomunicações, Faculdade de Engenharia, Universidade do Estado do Rio de Janeiro, nadia@eng.uerj.br, http://www.eng.uerj.br~ldmm*

Keywords: Genetic algorithm, network-on-chip, migration.

Abstract. The aim of this paper is to investigate migration strategies for the execution of parallel genetic algorithms in a Multiprocessor System on Chip (MPSoC). Some multimedia and Internet applications for wireless communications are using genetic algorithms and can benefit of the advantages provided by parallel processing on MPSoCs. In order to run such algorithms, we use a Network-on-Chip platform, which provides the interconnection network required for the communication between processors. Two migration strategies are employed, in order to analyse the speedup and efficiency each one can provide, considering the communication costs they require.

1 INTRODUCTION

The increasing demand of electronic systems, that require more and more processing power, low energy consumption, reduced area and low cost, has lead to the development of more complex embedded systems, also known as SoC (System on Chip), in order to run multimedia, Internet and wireless communication applications Ruiz and Antonio (2003). These systems can be built of several independent subsystems, that work in parallel and interchange data. When these systems have more than one processor, they are called Multi-Processor System on Chip (MPSoC). Currently, several products, such as cell phones, portable computers, digital televisions and video games, are built using embedded systems. While in embedded systems the communication between Intellectual Property (IP) blocks is basically done through a shared bus, in multiprocessor embedded systems this kind of interconnection compromises the expected performance Mello (2003). In this case, the communication is best implemented using an intrachip network, implemented by a Network on Chip (NoC) Benini and Micheli (2002) Jantsch et al. (2004) Ivanov and Micheli (2005) platform.

Some multimedia and Internet applications for wireless communications are using genetic algorithms and can benefit from the advantages provided by parallel processing on MPSoCs. In this paper, we present a parallel genetic algorithm that runs on Hermes Multi-Processor System (HMPS) architecture and discuss the impact of migration strategies on performance. In Section 2, we describe the HMPS architecture. The island model parallel genetic algorithm, used in this paper, is presented in Section 3 and some simulation results are introduced in Section 4. Finally, we draw some conclusions and future work in Section 5.

2 THE NETWORK-ON-CHIP PLATFORM

Figure 1 shows the NoC platform, called Hermes Multiprocessor System (HMPS) Woszezenki (2007). MPSoC architectures may be represented as a set of processing nodes that communicate via a communication network. Switches compose the network and RISC processors the processing nodes (Plasma). Information exchanged between resources are transfered as messages, which can be split into smaller parts called packages Terry Tao Ye and Micheli (2003). The switch allows for retransmission of messages from one module to another and decides which path these messages should take. Each switch has a set of bidirectional ports for the interconnection with a resource and the neighboring switches. As the total number of tasks composing the target application may exceed the MPSoC memory resources, one processor is dedicated to the management of the system resources (MP - Manager Processor). The MP has access to the task repository, from where tasks are allocated to some processors of the system.

The NoC communication network is based on HERMES Moraes et al. (2004), a parameterizable infrastructure that implements wormhole packet switching with a 2D mesh topology. The HERMES switch employs input buffers, centralized control logic, an internal crossbar and five bi-directional ports. The local port establishes the communication between the switch and its local IP core. The other ports of the switch are connected to neighboring switches. A centralized round-robin arbitration grants access to incoming packets and a deterministic XY routing algorithm is used to select the output port. The processor is based on the PLASMA processor Rhoads (2006), a compact RISC microprocessor. It has a compact instruction set comparable to a MIPS-1, 3 pipeline stages, no cache, no Memory Management Unit (MMU) and no memory protection support, in order to keep it as small as possible. A dedicated Direct Memory Access (DMA) unit is also used for speeding up task mapping, but not for data communications. The processor local memory (1024 Kbytes) is divided into four independent pages. Page 0 receives

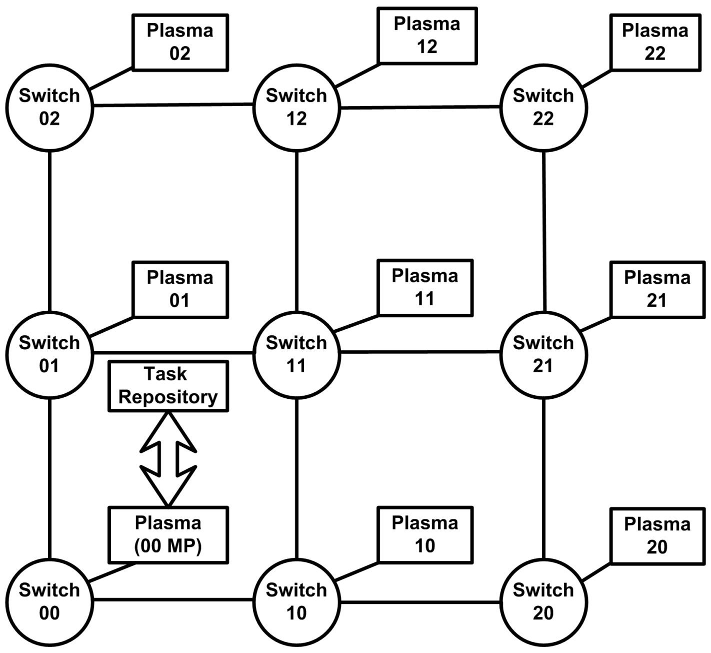


Figure 1: HMPS architecture, with 9 RISC Plasma processors connected to a 3×3 mesh network

the microkernel and pages 1 to 3 the tasks, where each task can hold 256 Kbytes (0x40000).

The HMPS communication primitives, *WritePipe*() and *ReadPipe*(), essentialy abstract communications, so that tasks can communicate with each other without knowing their position on the system, either on the same processor or a remote one. When HMPS starts, only the microkernel is loaded into the local memory. All tasks are stored in the task repository. The manager processor is responsible for reading the object codes from the task repository and transmit them to the other processors. The DMA module is responsible for transfering the object code from the network interface to the local memory.

3 ISLAND MODEL PARALLEL GENETIC ALGORITHM

In the island model, serial isolated subpopulations evolve in parallel, where each one is controlled by a single processor and periodically sends its best individuals to neighboring subpopulations, receiving from them their best individuals as result. These individuals are used to substitute the local worst ones. It is obvious that the GA execution time increases with population size. Therefore, small subpopulations tend to converge quickly when isolated.

3.1 The Algorithm

The Parallel Genetic Algorithm (PGA) is implemented using the HMPS communication primitives. Each processor corresponds to an island and its initial subpopulation is randomly generated, evolving independently from the other subpopulations, until the migration operator is activated, as described in Algorithm 1. Premature convergence occurs less in a multipopulation GA and can be ignored, when other islands produce better results. Each island can use a different set of GA operators, i.e. crossover and mutation rates, which causes different convergence. Migration of the chromosomes among the islands prevents mono-race populations, which converge prematurely. Periodic migration, which occurs after some generations, prevents a common convergence among the islands.

When the MPSoC is initialized, the microkenel of the MP obtains the slave processors addresses, the maximum number of processors used by the platform, the maximum number of tasks to be executed by each processor and the maximum number of used by the platform. Then, the microkernerl of the MP makes the tasks allocation of the PGA among the slave processors. Each slave processor executes one instance of the PGA.

Algorithm 1 PGA

```
1: Define and initialize the evolutionary parameters
2: $t \leftarrow 0$
3: Initialize a random population $p(t)$
4: Evaluate $p(t)$ in order to find the best solution
5: while ($t < NumGenerations$) do
6:    $t \leftarrow t + 1$
7:    Select $p(t)$ from $p(t-1)$
8:    Crossover
9:    Mutation
10:   Evaluate $p(t)$ in order to find the best solution
11:   if ($t$ mod $MigrationRate = 0$) then
12:      Migrate local $best[p(t)]$ to the next processor and receive remote $best[p(t)]$ from the
         previous processor
13:      Replace $worst[p(t)]$ by $best[p(t)]$
14:   end if
15: end while
```

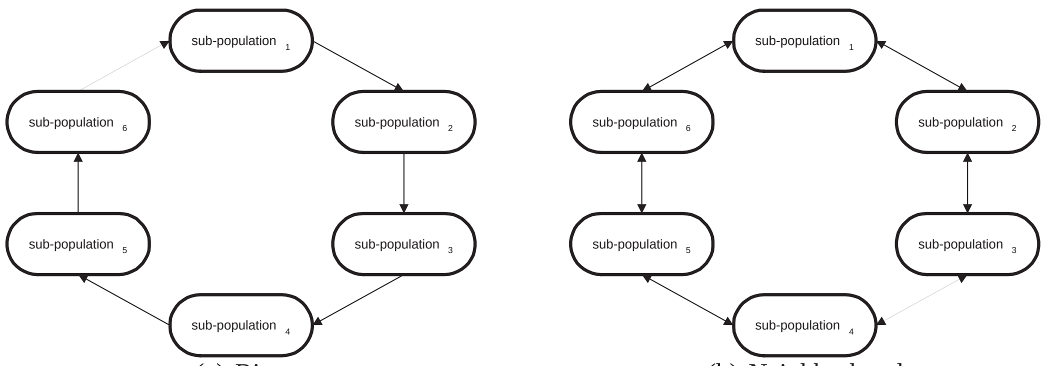
An PGA requires the definition of some parameters: number of processor, how often the migration will take place, which individuals will migrate and which individuals will be replaced due to migration. The island model introduces a migration operator in order to migrate the best individuals from one subpopulation to another.

3.2 Migration Strategies

There are several strategies used to migrate individuals from one subpopulation to another. Among them we can mention the ring topology and the neighborhood topology. In the ring topology, the best individuals from one subpopulation can only migrate to an adjacent one. As seen in Figure 2, for example, the best individuals from subpopulation 6 can only migrate to subpopulation 1 and the best individuals from subpopulation 1 can only migrate to subpopulation 2. For this kind of strategy, migration is implemented as in Algorithm 2. In the neighborhood topology, the best individuals from one subpopulation can migrate to a left and to a right neighbor, as seen in Figure 2. For this kind of strategy, migration is implemented as in Algorithm 3.

Choosing the right time of migration and which individuals should migrate are two critical decisions. Species may evolve quickly in small populations. However, migrations should occur after a time long enough for allowing the development of good characteristics in each subpopulation. Migration is a trigger for evolutionary changes and should occur after a fixed number of generations in each subpopulation. The migrant individuals are usually selected from the best individuals in the origin subpopulation and they replace the worst ones in the destination subpopulation. Intuition is still strongly recommended to fix the migration rate and there are no fixed rules that may give good results Hue (1997).

There are two reasons to send an individual from one subpopulation to another. One is to



(a) Ring (b) Neighborhood

Figure 2: Migration topologies

Algorithm 2 Migration function for the ring communication

```
1:  $local$ := $getprocessid()$;
2:  **if** $local$ = 0 **then**
3:      *next* := 1; *previous* := number of tasks $-1$;
4:  **end if**
5:  **if** $local > 0$ e $local <$ number of tasks $-1$ **then**
6:      *next* := $local + 1$; *previous* := $local - 1$;
7:  **end if**
8:  **if** $local$ = number of tasks $-1$ **then**
9:      *next* := 0; *previous* := $local - 1$;
10: **end if**
11: **Send** the best individuals to the task, whose identifier is *next*;
12: **Receive** the best individuals from the task, whose identifier is *previous*.
```

increase the fitness of the destination subpopulation. The other reason is to help maintaining the population diversity of the other subpopulation. As in the sequential GA, issues of selection pressure and diversity arise. If a subpopulation receives frequently and consistently highly fit individuals, these become predominant in the subpopulation and the GA will focus its search on them at the expense of diversity loose. On the other hand, if random individuals are received, the diversity may be maintained, but the fitness of the subpopulation may not be improved as desired. As migration policy, the best individual is chosen as the migrant, replacing the worst one in the receiving subpopulations. For the migration frequency, an empirical value was adopted based on the number of generations.

4 SIMULATION RESULTS

The two non-linear functions defined by Equation 1 were used by the PGA for optimization. Function $f_1(x)$ has 14 local maximum e one global maximum in the interval [-1, 2], with an approximate global maximum of $2,83917$, at $x = 1,84705$. Function $f_2(x, y)$ has various local minimum and one global minimum in the interval $-3 \leq x \leq 3$ and $-3 \leq y \leq 3$, and an

Algorithm 3 Migration function for the neighborhood communication

```
 1: $local := getprocessid()$;
 2: if $local$ = 0 then
 3:     next := 1; previous := number of tasks $-1$;
 4: end if
 5: if $local > 0$ e $local <$ number of tasks $-1$ then
 6:     next := $local + 1$; previous := $local - 1$;
 7: end if
 8: if $local$ = number of tasks $-1$ then
 9:     next := 0; previous := $local - 1$;
10: end if
11: Send the best individuals to the task, whose identifier is previous;
12: Send the best individuals to the task, whose identifier is next;
13: Receive the best individuals from the task, whose identifier is previous;
14: Receive the best individuals from the task, whose identifier is next.
```

approximate global minimum of -12.92393, at $x = 2,36470$ and $y = 2,48235$.

$$\begin{aligned} \max_x f_1(x) = &\quad sen(10\pi x) + 1 \\ \min_{x,y} f_2(x,y) = &\quad cos(4x) + 3sen(2y) + (y-2)^2 - (y+1) \end{aligned} \tag{1}$$

The performance of the PGA can be evaluated based on its speedup and efficiency. Speedup S_p Chiwiacowsky et al. (2003) is defined according to Equation 2, where T_1 is the execution time of the sequential version of the genetic algorithm and T_p is the execution time of its parallel version.

$$S_p = \frac{T_1}{T_p} \tag{2}$$

Efficiency E_p Chiwiacowsky et al. (2003) is defined according to Equation 3, where $\frac{1}{p} < E_p \leq 1$ and p is the number of processors employed.

$$E_p = \frac{S_p}{p} \tag{3}$$

Based on simulation results for the optimization of $f_1(x)$ and $f_2(x,y)$ using the ring and neighborhood topologies, we obtained the graphics for speedup and efficiency shown in Figure 3 and Figure 4 respectively. The data are presented as triples consisting of the number of slave processors used (*NumProc*), the migration rate (*MigRate*) and the migration interval (*MigInt*).

5 CONCLUSIONS

For the ring topology, the behavior of the two functions shows that, keeping the migration interval constant and varying the migration rate, if the increase in the migration rate resulted in an increase in speedup and efficiency, the fitness of the individuals, received by one or more populations during the migration phase, accelerated the evolutionary process, decreasing the convergence time. On the other hand, if the increase in the migration rate resulted in the decrease of speedup and efficiency, then we can say that the fitness of these individuals did not influence enough the evolutionary process of the populations that received them. In this case, the convergence time increases.

REFERENCES

Benini L. and Micheli G.D. Networks on chips: a new soc paradigm. *IEEE Computer*, 1(1):70–78, 2002.

Chiwiacowsky L.D., Velho H.F.d.C., Preto A.J., and Stephany S. Identifying initial conduction in heat conduction transfer by a genetic algorithm: a parallel aproach. In *Proceedings of XXIV Iberian Latin-American Congress on Computational Methods in Engineering*, volume 28, pages 180–195. 2003.

Hue X. Genetic algorithms for optimization – background and applications. 1997.

Ivanov A. and Micheli G.D. The network-on-chip paradigm in practice and research. *IEEE Design and Test of Computers*, 1(1):399–403, 2005.

Jantsch A., Öberg J., and Tenhunen H. Special issue on networks on chip. *Journal of Systems Architecture*, 1(1):61–63, 2004.

Mello A.M. Arquitetura multiprocessada em SoCs: estudo de diferentes topologias de conexão. 2003.

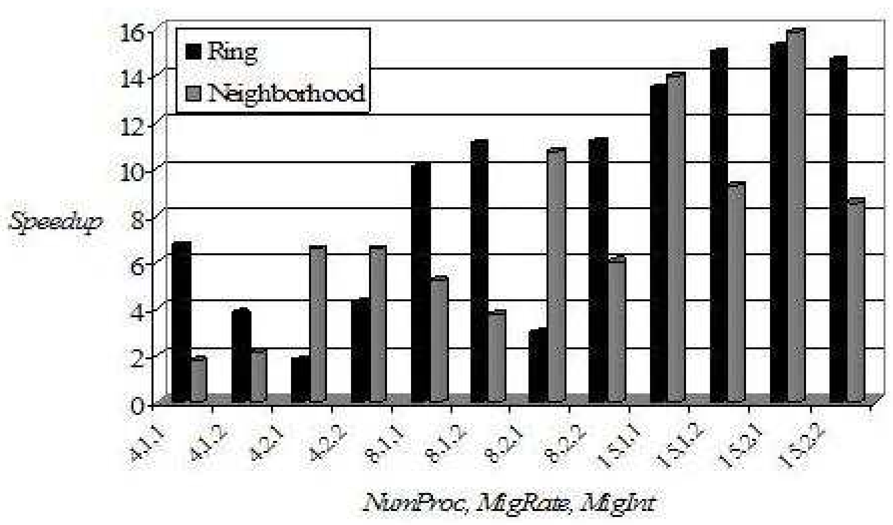
Moraes F., Calazans N., Mello A., Möller L., and Ost L. Hermes: an infrastructure for low area overhead packet-switching networks on chip. *Integration, the VLSI Journal*, 38(1):69–93, 2004.

Rhoads S. Plasma microprocessor. 2006.

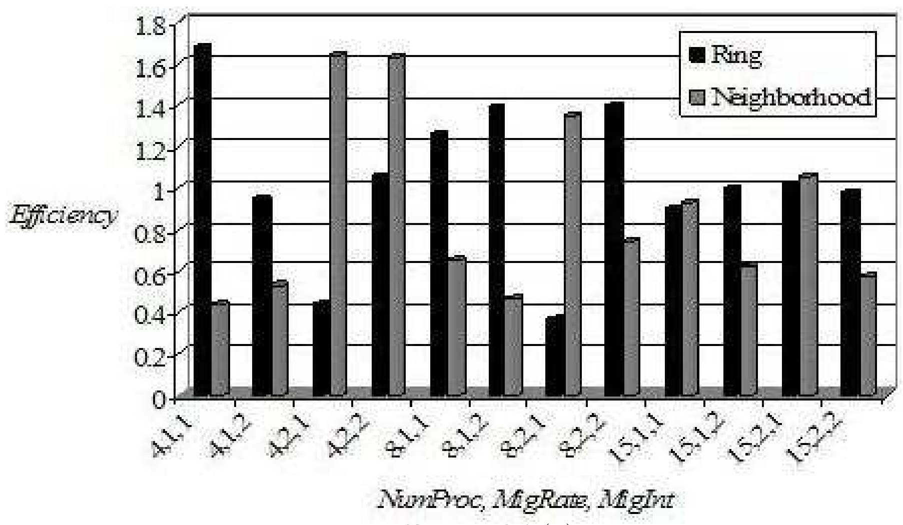
Ruiz P.M. and Antonio. Using genetic algorithms to optimize the behavior of adaptive multimedia applications in wireless and mobile scenarios. In *IEEE Wireless Communications and Networking Conf. (WCNC'2003)*, pages 2064–2068. IEEE Press, 2003.

Terry Tao Ye L.B. and Micheli G.D. Packetized on-chip interconnect communication analysis for mpsoc. In *Proceedings of the Design,Automation and Test in Europe Conference and Exhibition (DATEŠ03)*, pages 344–349. IEEE Press, 2003.

Woszezenki C. *Alocação de tarefas e comunicação entre tarefas em MPSoCs*. Master's Thesis, Faculdade de Informática, PUCRS, Porto Alegre, RS, Brazil, 2007.

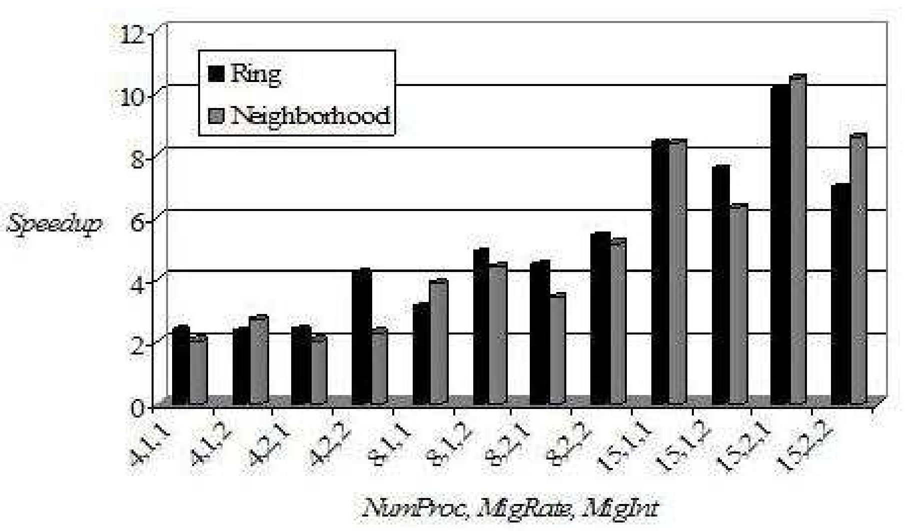


(a) Speedup of $f_1(x)$

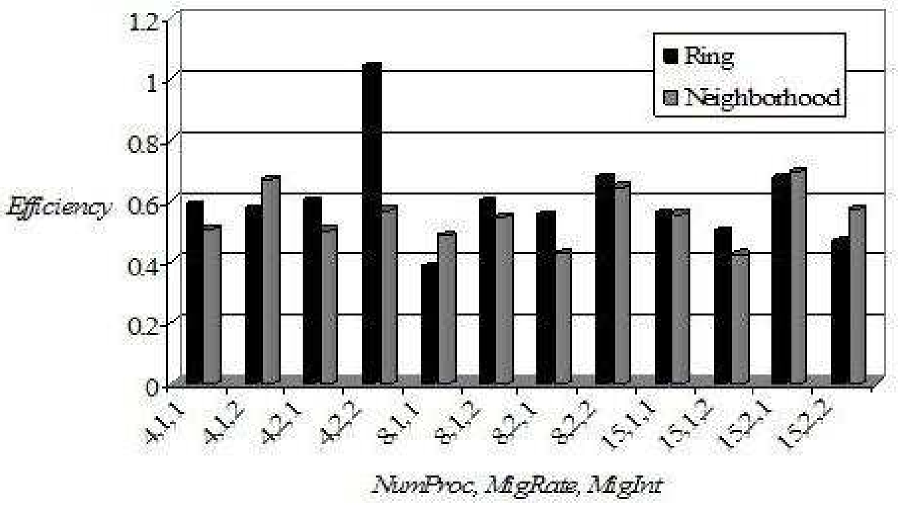


(b) Efficiecy of $f_1(x)$

Figure 3: Impact of the migration rate and migration interval on speedup and efficiency for function $f_1(x)$, considering the used topology



(a) Speedup of $f_2(x, y)$



(b) Efficiency of $f_2(x, y)$

Figure 4: Impact of the migration rate and migration interval on speedup and efficiency for function $f_2(x, y)$, considering the used topology