Chapter 2
Genetic Algorithms on Network-on-Chip[⋆]

Abstract. The aim of the work described in this chapter is to investigate migration strategies for the execution of parallel genetic algorithms in a Multi-Processor 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 analyze the speedup and efficiency each one can provide, considering the communication costs they require.

2.1 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 System-on-Chip (SoC), in order to run multimedia, Internet and wireless communication applications [9]. 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 [2]. In this case, the communication is best implemented using an intrachip network, implemented by a Network-on-Chip (NoC) [6] [5] [1] 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 chapter, we present a parallel genetic algorithm that

[⋆] This chapter was developed in collaboration with Rubem Euzébio Ferreira.

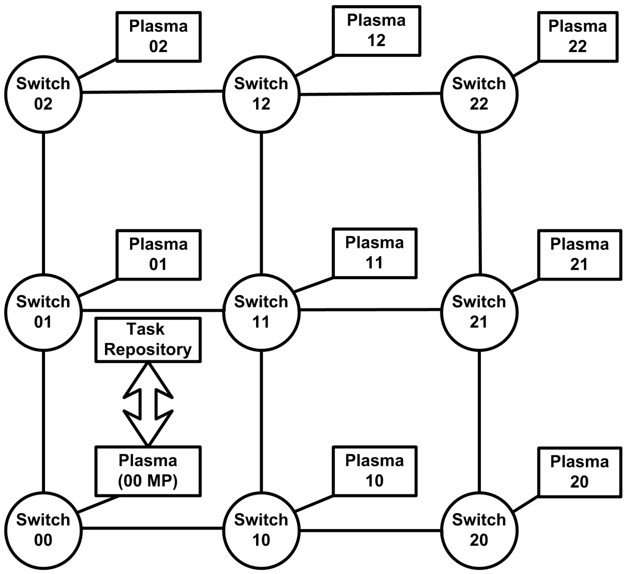


Fig. 2.1 HMPS architecture, with 9 RISC Plasma processors connected to a 3×3 mesh network

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 parallel genetic algorithm, used in this chapter, 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.

2.2 Multi-processor System-on-Chip Platform

Figure 2.1 shows the Multi-Processor System-on-Chip (MPSoC), called Hermes Multiprocessor System (HMPS) [3]. 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 [7]. 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 interconnection network is based on HERMES [4], 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 [10], a 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 the microkernel and pages 1 to 3 the tasks. Each task can hold 256 Kbytes (0x40000).

The HMPS communication primitives, *WritePipe*() and *ReadPipe*(), essentially 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 transferring the object code from the network interfaces to the local memory.

2.3 Parallel Genetic Algorithm

The Parallel Genetic Algorithm (PGA) is based on the island model, in which serial isolated subpopulations evolve in parallel and each one is controlled by a single processor. This processor periodically sends its best individuals to neighboring subpopulations and receives their best individuals. These individuals are used to substitute the local worst ones. It is obvious that the GA time processing increases with population size. Therefore, small subpopulations tend to converge quickly when isolated.

The PGA is executed by the HMPS platform. 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 2.1. Premature convergence occurs less in a multi-population 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.

Algorithm 2.1. PGA

```
Initialize the evolutionary parameters
$t \leftarrow 0$
Initialize a random population $p(t)$
Evaluate $p(t)$ in order to find th best solution
while ($t <$ NumGenerations) do
    $t \leftarrow t+1$
    Select $p(t)$ from $p(t-1)$
    Crossover
    Mutation
    Evaluate $p(t)$ in order to find th best solution
    if ($t$ mod MigrationRate = 0) then
        Migrate local $best[p(t)]$ to the next processor
        Receive remote $best[p(t)]$ from the previous processor
        Replace $worst[p(t)]$ by $best[p(t)]$
    end if
end while
```

The PGA requires the definition of some parameters: number of processors, 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.

2.3.1 Topology Strategies

In this work, we investigate two topology strategies to migrate individuals from one subpopulation to another: ring and neighborhood. In the ring topology, the best individuals from one subpopulation can only migrate to an adjacent one. As seen in Figure 2.2, 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. In Algorithm 2.2, migration is implemented using this kind of strategy. In the neighborhood topology, the best individuals from one subpopulation can migrate to a left and to a right neighbor, as seen in Figure 2.3. For this kind of strategy, migration is implemented as in Algorithm 2.3.

Choosing the right time of migration and which individuals should migrate are two critical decisions. 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. Since there are no fixed rules that would give good results, intuition is still strongly recommended to fix the migration rate [11].

Sending an individual from one subpopulation to another increases the fitness of the destination subpopulation and maintains the population diversity of the other subpopulation. As in the sequential GA, issues of selection pressure and diversity

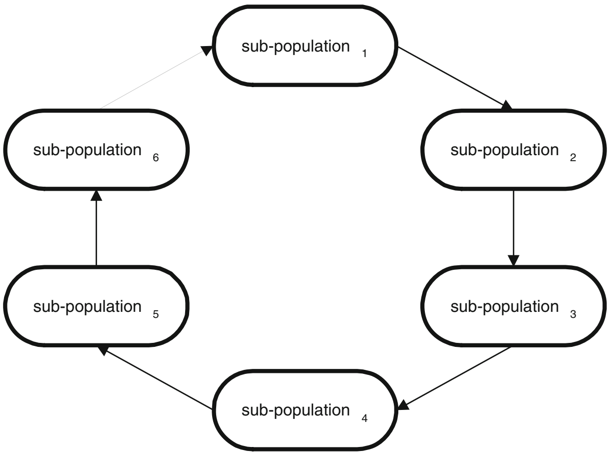


Fig. 2.2 Ring migration topology

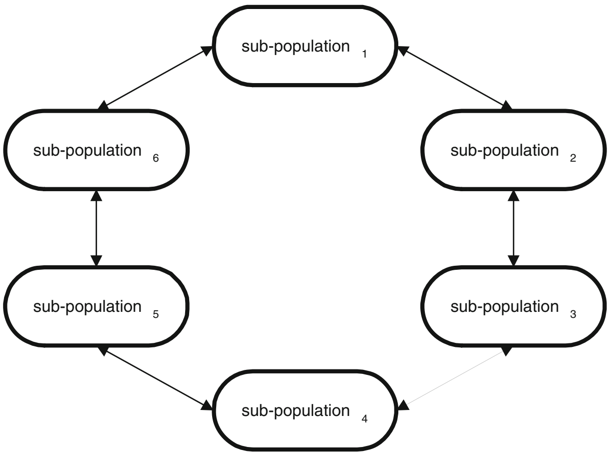


Fig. 2.3 Neighborhood migration topology

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.

Algorithm 2.2. Migration function for the ring communication

```
$local := getprocessid()$;
**if** $local = 0$ **then**
    $next := 1$; $previous :=$ number of tasks $-1$;
**end if**
**if** $local > 0$ e $local <$ number of tasks $-1$ **then**
    $next := local + 1$; $previous := local - 1$;
**end if**
**if** $local =$ number of tasks $-1$ **then**
    $next := 0$; $previous := local - 1$;
**end if**
**Send** the best individuals to the task, whose identifier is *next*;
**Receive** the best individuals from the task, whose identifier is *previous*.
```

Algorithm 2.3. Migration function for the neighborhood communication

```
$local := getprocessid()$;
**if** $local = 0$ **then**
    $next := 1$; $previous :=$ number of tasks $-1$;
**end if**
**if** $local > 0$ e $local <$ number of tasks $-1$ **then**
    $next := local + 1$; $previous := local - 1$;
**end if**
**if** $local =$ number of tasks $-1$ **then**
    $next := 0$; $previous := local - 1$;
**end if**
**Send** the best individuals to the task, whose identifier is *previous*;
**Send** the best individuals to the task, whose identifier is *next*;
**Receive** the best individuals from the task, whose identifier is *previous*;
**Receive** the best individuals from the task, whose identifier is *next*.
```

2.4 Simulation Results

Three non-linear functions were used by the PGA for optimization. The definition and main characteristics of these functions are listed below

- Function $f_1(x)$ is defined in (2.1). This function plots into the curve depicted in Fig. 2.4. It presents 14 local maximum e one global maximum in the interval [-1, 2], with an approximate global maximum of 2.83917, at $x = 1.84705$.

$$\max_{x} f_1(x) = sen(10\pi x) + 1 \tag{2.1}$$

- Function $f_2(x,y)$ is defined in (2.2). This function plots into the curve depicted in Fig. 2.5. It has many local minimum and one global minimum in the interval $-3 \leq x \leq 3$ and $-3 \leq y \leq 3$, and an approximate global minimum of -12.92393, at $x = 2,36470$ and $y = 2.48235$.

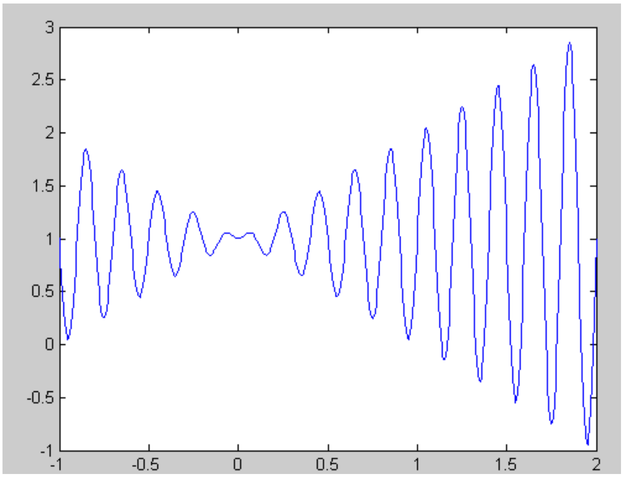


Fig. 2.4 The graphical representation of $f_1(x)$

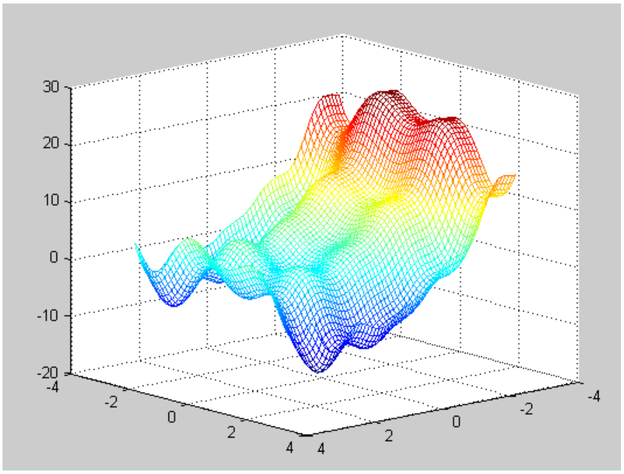


Fig. 2.5 The graphical representation of $f_2(x,y)$

$$\min_{x,y} f_2(x,y) = cos(4x) + 3sen(2y) + (y-2)^2 - (y+1) \qquad (2.2)$$

- Function $f_3(x,y)$ is defined in (2.3). This function plots into the curve depicted in Fig. 2.6. It has 2 local maximum and one global minimum in the interval $-3 \leq x \leq 3$ and $-3 \leq y \leq 3$, and an approximate global maximum of $8,11152$, at $x = 0,01176$ and $y = 1,58823$.

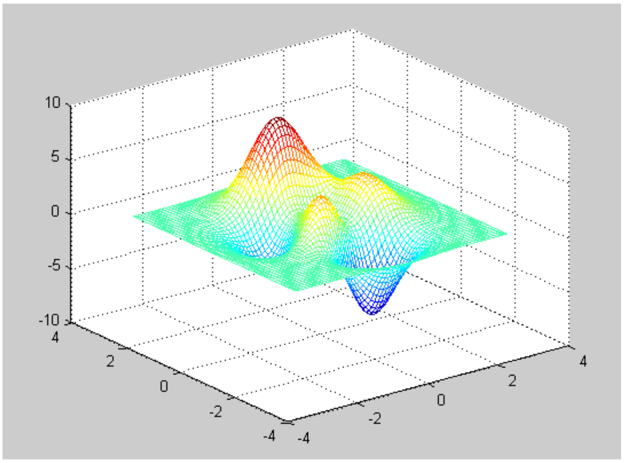


Fig. 2.6 The graphical representation of $f_3(x,5)$

$$\max_{x,y} f_3(x,y) = 3(1-x)^2 e^{(-x^2-(y+1)^2)} - 10\left(\frac{x}{5} - x^3 - y^5\right) e^{(-x^2-y^2)} - \frac{1}{3} e^{-(x+1)^2-y^2} \tag{2.3}$$

The performance of the PGA can be evaluated based on its speedup and efficiency. Speedup S_p [8] is defined according to Equation 2.4, 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.

$$S_p = \frac{T_1}{T_p} \tag{2.4}$$

Efficiency E_p [8] is defined according to Equation 2.5, where $\frac{1}{p} < E_p \leq 1$ and p is the number of processors employed.

$$E_p = \frac{S_p}{p} \tag{2.5}$$

Table 2.1, Table 2.2 and Table 2.3 show the simulation results for the optimization of functions $f_1(x)$, $f_2(x,y)$ e $f_3(x,y)$ respectively using the ring topology for migration of individuals. In those tables, N_p is the number of used processors, M_r is the migration rate, M_i is the migration interval in terms of generation number, S_p is the speedup obtained and E_p is the efficiency yield for each used processor.

Table 2.4, Table 2.5 and Table 2.6 show the simulation results for the optimization of functions $f_1(x)$, $f_2(x,y)$ e $f_3(x,y)$ respectively using the neighborhood topology for migration of individuals. In those tables, N_p is the number of used processors, M_r is the migration rate, M_i is the migration interval in terms of generation number, S_p is the speedup obtained and E_p is the efficiency yield for each used processor.

Based on simulation results for the optimization of $f_1(x)$, $f_2(x,y)$ and $f_3(x,y)$ using the ring and neighborhood topologies, we obtained the graphics for speedup

Table 2.1 Simulation results for the optimization of function $f_1(x)$ for ring migration topology

N_p	M_r	M_i	Time (ms)	S_p	E_p
1	–	–	1127,5724	1	1
6	1	1	168,57284	6,68893	1,67223
		2	298,76094	3,77416	0,94354
	2	1	650,70556	1,73284	0,43321
		2	267,11808	4,22125	1,05531
9	1	1	112,10709	10,05799	1,25724
		2	102,16839	11,03641	1,37955
	2	1	381,15057	2,95834	0,36979
		2	101,16498	11,14587	1,39323
16	1	1	83,86655	13,44484	0,89632
		2	75,29244	14,97590	0,998393
	2	1	73,95938	15,24583	1,01638
		2	77,13687	14,61781	0,97452

Table 2.2 Simulation results for the optimization of function $f_2(x,y)$ for ring migration topology

N_p	M_r	M_i	Time (ms)	S_p	E_p
1	–	–	6024,11201	1	1
6	1	1	2569,06697	2,344863	0,586215
		2	2616,76305	2,302123	0,575530
	2	1	2507,48402	2,402452	0,600613
		2	1448,84485	4,157872	1,039468
9	1	1	1968,24989	3,06064	0,38258
		2	1250,55945	4,81713	0,60214
	2	1	1352,18413	4,45509	0,55688
		2	1112,49588	5,41495	0,67686
16	1	1	718,73197	8,38158	0,55877
		2	797,31202	7,55552	0,50370
	2	1	596,06991	10,10638	0,67375
		2	866,79268	6,94988	0,46332

and efficiency shown in Figure 2.7, Figure 2.8 and Figure 2.9 respectively. The data are presented as triples consisting of the number of slave processors used N_p, the migration rate M_r and the migration interval M_i.

2.5 Summary

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

Table 2.3 Simulation results for the optimization of function $f_3(x,y)$ for ring migration topology

N_p	M_r	M_i	Time (ms)	S_p	E_p
1	–	–	6209,50022	1	1
6	1	1	2778,47764	2,23485	0,55871
		2	2927,64913	2,12098	0,53024
	2	1	3143,09053	1,97560	0,49390
		2	2925,58322	2,12248	0,53062
9	1	1	1037,66721	5,98409	0,74801
		2	1832,88554	3,38782	0,42347
	2	1	1799,06522	3,45151	0,43143
		2	1433,94829	4,33035	0,54129
16	1	1	873,31097	7,11029	0,47401
		2	723,58761	8,58154	0,57210
	2	1	607,38299	10,22336	0,68155
		2	942,71555	6,58682	0,43912

Table 2.4 Simulation results for the optimization of function $f_1(x)$ for neighborhood migration topology

N_p	M_r	M_i	Time (ms)	S_p	E_p
1	–	–	1127,5724	1	1
6	1	1	645,36593	1,74718	0,43679
		2	535,14461	2,10704	0,52676
	2	1	172,29855	6,54429	1,63607
		2	172,80265	6,52520	1,63130
9	1	1	217,88489	5,17508	0,64688
		2	304,68098	3,70082	0,46260
	2	1	104,90308	10,74870	1,34358
		2	188,31056	5,98783	0,74847
16	1	1	80,62822	13,98483	0,93232
		2	121,45834	9,28361	0,61890
	2	1	71,09218	15,86070	1,05738
		2	131,73707	8,55926	0,57061

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.

In the future, we intend to investigate the impact of other migration strategies on the performance of the parallel Network-on-chip based implementation of genetic algorithms. One of the these topologies is broadcasting, which allows each processor

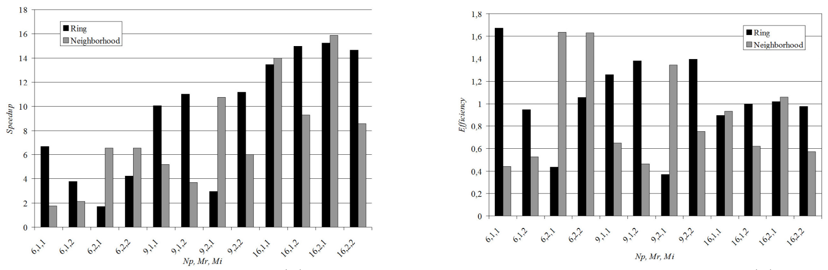
Table 2.5 Simulation results for the optimization of function $f_2(x,y)$ for neighborhood migration topology

N_p	M_r	M_i	Time (ms)	S_p	E_p
1	–	–	6024,11201	1	1
6	1	1	2970,49815	2,02798	0,50699
		2	2241,80203	2,68717	0,67179
	2	1	2977,43556	2,02325	0,50581
		2	2635,64829	2,28562	0,57140
9	1	1	1560,87682	3,85944	0,48243
		2	1370,53135	4,39545	0,54943
	2	1	1772,67139	3,39832	0,42479
		2	1161,60725	5,18601	0,64825
16	1	1	719,73603	8,36989	0,55799
		2	951,55986	6,33077	0,42205
	2	1	574,84260	10,47958	0,69863
		2	700,59551	8,59855	0,57323

Table 2.6 Simulation results for the optimization of function $f_3(x,y)$ for neighborhood migration topology

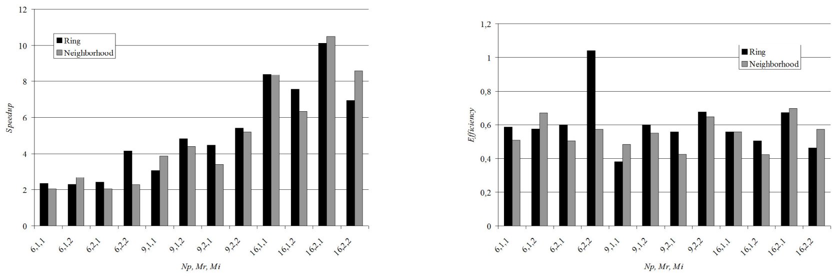
N_p	M_r	M_i	Time (ms)	S_p	E_p
1	–	–	6209,50022	1	1
6	1	1	2534,68066	2,44981	0,61245
		2	2497,41481	2,48637	0,62159
	2	1	3075,95908	2,01872	0,50468
		2	2737,40887	2,26838	0,56709
9	1	1	1698,95341	3,65489	0,45686
		2	1398,89571	4,43885	0,55485
	2	1	830,546335	7,47640	0,93455
		2	1296,38967	4,78984	0,59873
16	1	1	1235,58877	5,02553	0,33503
		2	910,60102	6,81912	0,45460
	2	1	777,34866	7,98805	0,53253
		2	683,45716	9,08542	0,60569

to send the best solution found so far to all the other processors in the network. We will assess the impact of heavy message send/receive workload on the overall system performance.



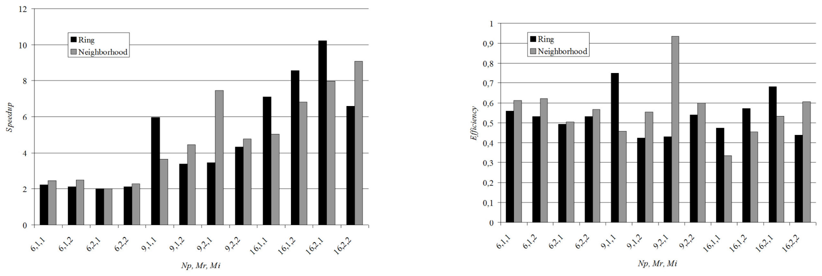
(a) Speedup of $f_1(x)$ (b) Efficiency of $f_1(x)$

Fig. 2.7 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 $f_2(x,y)$

Fig. 2.8 Impact of the migration rate and migration interval on speedup and efficiency for function $f_2(x,y)$, considering the used topology



(a) *Speedup* of $f_3(x,y)$ (b) Efficiency of $f_3(x,y)$

Fig. 2.9 Impact of the migration rate and migration interval on speedup and efficiency for function $f_3(x,y)$, considering the used topology

References

1. Ivanov, A., De Micheli, G.: The network-on-chip paradigm in practice and research. IEEE Design and Test of Computers 1(1), 399–403 (2005)
2. Mello, A.M.: Arquitetura multiprocessada em SoCs: estudo de diferentes topologias de conexão (June 2003) (in Portuguese)
3. Woszezenki, C.: Alocação de tarefas e comunicação entre tarefas em mpsocs. M.Sc., Faculdade de Informática, PUCRS, Porto Alegre, RS, Brazil (June 2007) (in Portuguese)
4. Moraes, F., Calazans, N., Mello, A., Möller, L., Ost, L.: Hermes: an infrastructure for low area overhead packet-switching networks on chip. Integration, The VLSI Journal 38(1), 69–93 (2004)
5. Öberg, J., Jantsch, A., Tenhunen, H.: Special issue on networks on chip. Journal of Systems Architecture 1(1), 61–63 (2004)
6. Beniniand, L., De Micheli, G.: Networks on chips: a new soc paradigm. IEEE Computer 1(1), 70–78 (2002)
7. Benini, L., Ye, T.T., De Micheli, G.: Packetized on-chip interconnect communication analysis for MPSoC. In: Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE 2003), pp. 344–349. IEEE Press (2003)
8. Chiwiacowsky, L.D., de Campos Velho, H.F., Preto, A.J., Stephany, S.: Identifying initial conduction in heat conduction transfer by a genetic algorithm: a parallel approach 28, 180–195 (1980)
9. Ruiz, P.M., Antonio: Using genetic algorithms to optimize the behavior of adaptive multimedia applications in wireless and mobile scenarios. In: IEEE Wireless Communications and Networking Conference (WCNC 2003), pp. 2064–2068. IEEE Press (2003)
10. Rhoads, S.: Plasma microprocessor (2009), `http://www.opencores.org`
11. Hue, X.: Genetic algorithms for optimization – background and applications. Technical report. Edinburgh Parallel Computer Centre, The University of Edinburgh (1997)