Bulletin of Electrical Engineering and Informatics
Vol. 8, No. 2, June 2019, pp. 414~421
ISSN: 2302-9285, DOI: 10.11591/eei.v8i2.1395

Task mapping and routing optimization for hard real-time Networks-on-Chip

M. Norazizi Sham Mohd Sayuti[1], Farida Hazwani Mohd Ridzuan[2], Zul Hilmi Abdullah[3]
[1]Faculty of Engineering and Built Environment, Universiti Sains Islam Malaysia (USIM), Malaysia
[2]Faculty of Science and Technology, Universiti Sains Islam Malaysia (USIM), Malaysia
[3]Faculty of Information Technology and Science, INTI International University, Malaysia

Article Info

Article history:

Received Dec 4, 2018
Revised Jan 31, 2019
Accepted Feb 22, 2019

Keywords:

Design space exploration
Network routing
Networks-on-Chip
Real-time systems
Schedulability
Task mapping

ABSTRACT

Interference from high priority tasks and messages in a hard real-time Networks-on-Chip (NoC) create computation and communication delays. As the delays increase in number, maintaining the system's schedulability become difficult. In order to overcome the problem, one way is to reduce interference in the NoC by changing task mapping and network routing. Some population-based heuristics evaluate the worst-case response times of tasks and messages based on the schedulability analysis, but requires a significant amount of optimization time to complete due to the complexity of the evaluation function. In this paper, we propose an optimization technique that explore both parameters simultaneously with the aim to meet the schedulability of the system, hence reducing the optimization time. One of the advantages from our approach is the unrepeated call to the evaluation function, which is unaddressed in the heuristics that configure design parameters in stages. The results show that a schedulable configuration can be found from the large design space.

Corresponding Author:

M. Norazizi Sham Mohd Sayuti,
Faculty of Engineering and Built Environment (FKAB),
Universiti Sains Islam Malaysia,
Bandar Baru Nilai, 71800 Nilai, Negeri Sembilan, Malaysia.
Email: azizi@usim.edu.my

1. INTRODUCTION

A scheduling policy such as the rate monotonic priority pre-emptive scheduling [1] arranges the execution of tasks on processing cores and messages on network links based on priority levels. The pre-emption mechanism of the policy, however, cause delays to the low priority tasks and messages. In a worst-case scenario, their response times are further exacerbated from network congestion to the extent the system experiences deadline misses. Figure 1(a) shows an example of a task mapping and network routing on a NoC platform. In this platform, message $\mathbf{F_2}$ shares links with message $\mathbf{F_1}$ and message $\mathbf{F_5}$. Depending on the priority levels assigned to the messages (in this example from high to low order: $\mathbf{F_2}>\mathbf{F_1}>\mathbf{F_5}$), the scheduling policy pre-empts the low priority messages ($\mathbf{F_1}$ and $\mathbf{F_5}$) to allow the high priority message ($\mathbf{F_2}$) accessing the links. If a link used by the high priority message is blocked, due to a congestion at a location somewhere in the network, both low priority messages' response times are further delayed, potentially exceeding their deadlines.

Since the pre-emption occurs because of the existence of contention between messages, one way to avoid it is by changing the task mapping and network routing as shown in Figure 1(b). With the new task mapping, processing cores are allocated for tasks in a way that the assignment of network links able to divert contention among messages. However, configuring both task mapping and network routing creates a large multi-dimensional design space. Without a proper searching technique, a good configuration consisting a schedulable task mapping and network routing is difficult to find. The exhaustive search is not a preferred approach, since it is a time-consuming process. Nevertheless, as the number of tasks and messages, and the size of NoC increases so does the time complexity of the algorithm.

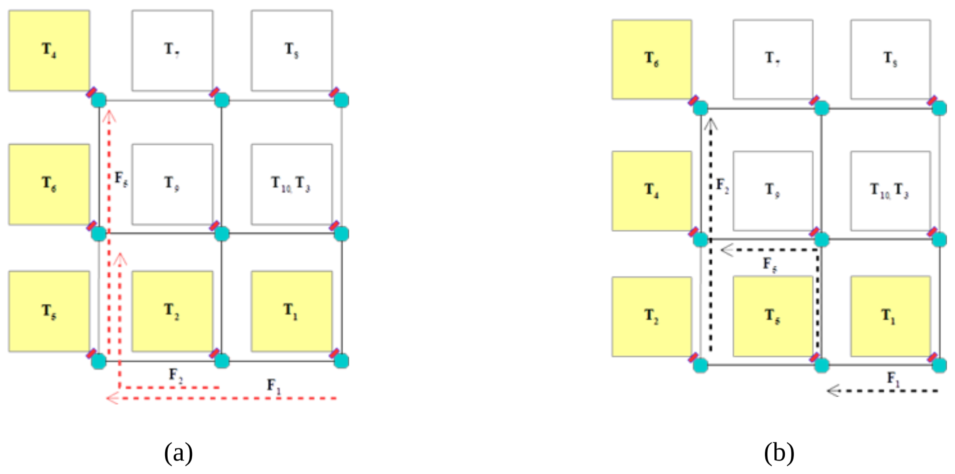


Figure 1. Contention of tasks and messages before and after remapping and rerouting, (a) Before, (b) After

Every point in the design space is a potential configuration. The schedulability of configuration is determined by the algorithm’s evaluation function. The search algorithm calls the function during optimization process until a breaking point is reached. For the search to be time-efficient, the function calls must be minimized, otherwise exploring as many as configurations in the design space within the stipulated design time is difficult to achieve. In other word, to implement this mechanism, the algorithm must have the capability to navigate efficiently in the design space.

2. RELATED WORK

Genetic Algorithm (GA) based task mapping optimization approaches [2, 3] with schedulability analysis as the fitness function proved to be efficient in addressing task mapping problems for NoC based hard real-time embedded systems. The results show that a schedulable system is achievable by merely change the task mapping. In some cases where network links are congested by many messages, without alternative routings to reduce congestion the approaches may render the system unschedulable.

According to the taxonomy presented in [4], routing algorithms can be categorized based on their characteristics. The two main categories are deterministic routing and adaptive routing. In a network with deterministic routing, a packet is routed according to a pre-determined routing path between the sender and receiver. One of the examples is XY routing. The NoCs that implement XY routing algorithm include Dally [5], QNoC [6] and Hermes [7]. Based on the XY routing, a task mapping algorithm [8] was proposed to minimize energy and meet the bandwidth constraints. The algorithm was further extended [9] to optimize routing as well.

Configuring more than one parameters such as mapping and routing is preferred because the impact from their settings affects each other. Otherwise optimization will occur only to one parameter but not the others. An attempt to configure more than one parameters was conducted by [10] for multimedia application. The configuration is done in a way that the task mapping is configured first then followed by network routing. Although it finds the feasible configuration that meets the requirement of the application, the time taken by the stages approach may increase due to the repetitive evaluation during search. Approaches proposed by [11, 12] requires repeating the routing and evaluation for pair-wise swaps of nodes in the topology. Nevertheless, all the approaches [8-12] do not support the hard real-time requirements.

Designing hard real-time embedded systems often requires analysis of the end-to-end response time in the worst-case scenario. For hard real-time systems, the end-to-end response time, which includes the execution time of task and the latency of message, must not exceed the timing constraints of the systems. Indeed, different task mappings produce varying performances and changing the task mapping must ensure this requirement is met. State-of-the-art task mapping techniques with deterministic XY routing for the NoC-based hard real-time embedded systems exists [13, 14], however, their proposed techniques assume that the NoC routing protocol is deterministic. This may hinder alternative designs to be found from the design space, since there is a probability that an alternative configuration may exist with both parameters optimized.

To the best of our knowledge, simultaneous exploration of task mapping and the message routing for NoC-based hard real-time embedded systems are still open issues to be addressed. Although, efficient routing schemes are necessary to avoid congestion or blocking of messages on the routing paths, it affects the hard real-time performance of the systems. If the performance is not met, then mapping or routing needs to be performed again. This is a time-consuming process. In this case, simultaneous exploration of both parameters is needed to make the optimization process efficient, at the same time ensures the overall optimization time minimum.

3. PROPOSED OPTIMIZATION METHOD

3.1. System model

An application model describes the design-time characteristics of computation and communication load imposed by an application. Tasks and messages of the application are released either periodically or sporadically. We assume that a task can only transmit a message after its execution finishes and the overhead of the NoC Network Interface is negligible. Packetisation of a message before its transmission produces one or several packets. A traffic flow represents a series of packets sent over the NoC from a source task to a destination task. Therefore, the end-to-end response time upper bound of a task that communicates with another task at a different core includes the amount of time since its release until the last packet that it sends arrives at the receiving task in a worst-case scenario.

Every task is described as a 4-tuple Task={c,t,p,d} where c is the worst-case execution time of task, t is the minimum inter-arrival interval of task, p is the priority level of task and d is the end-to-end deadline of task. Each message is described as a 5-tuple Msg={So, De, P, L, C}, where So is the source task, De is the receiving task, P is the priority level of message, L is the length of message and C is the no-load latency of message. The worst-case execution time (WCET, denoted by variable ci) is the maximum time that a task can take to finish execution when running on its own over a processing core. Likewise, given a specific path over the NoC and a maximum packet size, the no-load latency of a traffic flow (Ci) is its latency when no traffic-flow contention exists. For most NoCs, the no-load latency can be deterministically found and it is a function of the number of hops, the number of flits, the time needed for a flit to traverse a link and the time needed for a router to route and arbitrate packet headers.

The platform is modelled to represent the architecture of an on-chip multicore system interconnected with NoC. We implemented the platform as a 2D Mesh with wormhole switching. The arbitration unit supports flit-level pre-emption to control the access of packets to shared links in accordance with their priority levels. A processing core can run more than one task and a link can be shared by multiple traffic flows.

3.2. Optimization objective

The proposed optimization technique relies on the end-to-end schedulability analysis [14] to determine the schedulability of all tasks and flows in the system. To reduce the complexity of the analysis, we assume the deadline of a task is equivalent to its period (i.e. d=t) and a traffic flow shares the same deadline as its sending task. If for each task i of a task set, r_i ≤ d_i, then the task set is deemed schedulable on the platform, where ri is the worst-case response time and di is the deadline of task i. If task i is not schedulable, the schedulability indicator of task i (Uti) has value one.

$$Task_i : if\ r_i > d_i \rightarrow Ut_i = 1$$

The release of a message as a traffic flow only occurs after the sending task finishes its execution. It is assumed that the overhead to packetize its payload is negligible. Therefore, each flow's maximum release jitter can be deduced as the worst-case response time (r_i) of the task that releases it. With these assumptions, the end-to-end response time of a task is defined as the time taken from its release until the last packet it sends arrives at the receiving task. The traffic flow may not be schedulable if it fails to meet the deadline of their sending task (e.g. due to the interference from other packets sharing the same path or to the delay in execution of the task itself). A traffic flow is deemed schedulable only if $r_i + R_i \leq d_i$, where R_i is the worst-case latency of flow i. Otherwise, the schedulability indicator of message i (Uf_i) is one.

$$Flow_i: if\ r_i + R_i > d_i \rightarrow Uf_i = 1$$

The metric representing the schedulability of the system is the total number of unschedulable tasks and flows, and (1) is the function that calculates this metric. If S is zero, the system is deemed as fully schedulable.

$$S = \sum_{i=1}^{k} Ut_i + \sum_{i=1}^{l} Uf_i \qquad (1)$$

In order to guide the optimization algorithm the objective function is minimization of S as derived in (2), where S is given by (1).

$$Obj_1 = min(S) \qquad (2)$$

3.3. Design space

We chose GA as the optimization algorithm in our approach, as it is widely known for solving combinatorial optimization problems. The algorithm is based on evolutionary principles; it produces a group of individuals in a population and evolves them to create better individuals over a number of evolution cycles, or generations. In the concept of GA, individuals are evolved through several processes involving manipulation of chromosomes. These processes are selection, crossover and mutation as shown in Figure 2. The selection process is responsible for finding the best parents for the crossover process based on a binary tournament. The parents' chromosomes are then shifted by the crossover process to produce more than one new offspring in the population. Some of the parents are transferred to the population without being mated and this probability is determined by the crossover rate. A diversity is introduced into the population by mutating some of the genes in the individuals. The probability of a gene to be selected for mutation are determined by the mutation rate.

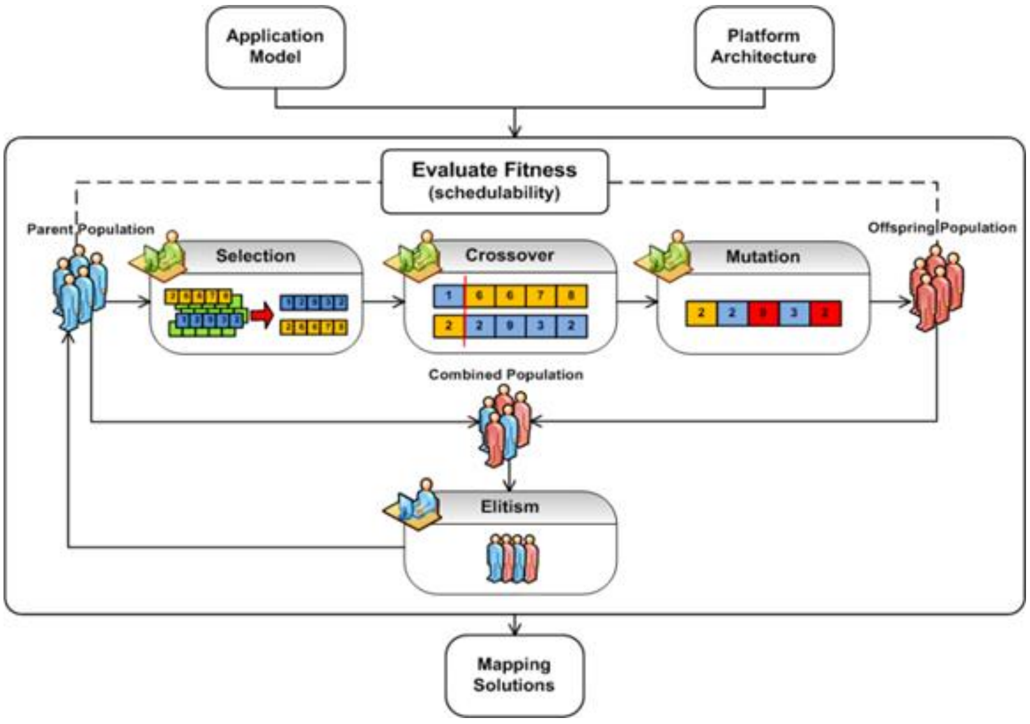


Figure 2. Optimization pipeline

The thread-like structure known as chromosome, which forms every individuals in the population of the proposed GA-based algorithm, is encoded with information on the mapping and routing. As shown in Figure 3, the chromosome structure contains small units called genes and these units are divided into two halves. The first half of the chromosome (green) represents the task mapping encoding. Each core on the NoC is numbered in an ascending order from *0* to *n-1*, where *n* is the total number of cores on the NoC. Each gene in the first half represents a task and the algorithm assigns a processing core's number to each gene. As the result, a core may execute one or more tasks. For example, genes representing task τ_1 (1st gene) and task τ_3 (3rd gene) are assigned with number five (the index of 6th core on the NoC), that is both tasks are run by the same processing core.

Network routing information is encoded in the second half of the chromosome. Genes marked by the orange color in the chromosome are configured to represent a flow. A message flow, as previously described, is defined as a traverse of packets between a source and a destination. The message flow is routed through several nodes on the NoC by the network routing protocol. The message can be diverted from its original course by changing a node on its current path to avoid interference from other messages. This produces an alternative routing path in addition to the one provided by the network routing protocol of the NoC.

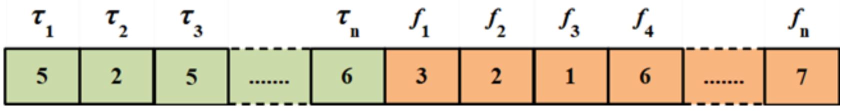


Figure 3. GA chromosome encoding of task mapping and routing

Based on this notion, the proposed algorithm is configured to assign each gene on the second half (orange colors) with a processing core's number. For example, the gene representing a message flow f_1 contains a number 3 (4th core on the NoC) indicating that it must travel through the router connected to that core first before heading to the destination. The role of the network routing protocol is to select the path from the source to the selected node, and then from the selected node to the destination. This way the network routing protocol provides the paths for the message flows based on the nodes determined by the proposed algorithm. For example in Figure 3 changes to the routing path of f_1 is performed in this manner; first a new routing path is assigned from the source node to the node 3 (4th core on the NoC) based on the node number in f_1 gene and then the path continues from the node 3 to the destination node.

4. RESEARCH METHOD

The proposed optimization algorithm (GA-MR) simultaneously configures mapping and routing of the hard real-time application on the NoC platform. While the current techniques [13-16] able to produce schedulable mappings with a fixed routing (XY routing), no support is provided for alternative routings and no guarantees that the mapping is still schedulable if other routing mechanisms are applied. Thus, the aims of the study are twofold; to find out whether GA-MR could find a schedulable configuration from the design space and to find out if the mapping that it finds is schedulable with alternative routings. If the latter is supported, it implies not only the GA-MR's ability to produce a schedulable configuration but also its mapping's feasibility with alternative routings.

The existing GA-based NoC mapping algorithm (GA-XY) utilizing fixed XY routing scheme, a Nearest Neighbor (NN) algorithm and random (RAN) mapping were used as benchmarks. As GA-MR, the benchmarks were executed under the same platform: 2D NoC with Mesh topology such as 4x4, 5x5 and 6x6. This is to gain insight on how GA-MR scales with the size of the platform. Two types of hard real-time test benches was applied in this study. The first contains a group of tasks and messages based on the real autonomous vehicle application (AVA) and the other application was modeled as a synthetic test bench (SA) with increased number of tasks and messages, for the purpose of simulating higher utilization of cores and links than AVA. Priority levels are assigned to tasks of both application based on the rate monotonic scheduling policy [1] and every message inherits the same priority as its originating task.

Since GA-MR and GA-XY are based on the same optimization flow, the same settings as [13] as shown in Table 1 were used by both algorithms. GA-XY only configures task mapping, hence its chromosome and encoding is similar with the first part of the GA-MR's chromosome (highlighted with green as in the Figure 3).

Table 1. GA settings

Group Population	Crossover Rate	Mutation Rate	Generation
100	0.5	0.01	500

5. RESULTS AND ANALYSIS

In terms of schedulability, GA-MR's mapping is fully schedulable compared to NN and random mapping for AVA mapped on the 4x4 platform. This is depicted in Figure 4 whereby the level of schedulability of GA-MR is 100%, NN is 31.9% and random mapping is 80.56% respectively. Similar performance produced by GA-MR is shown in Figure 5. For SA mapped on the 5x5 platform, GA-MR achieved 100% schedulability, NN 44.0% and random mapping 74.0%. These results imply that GA-MR able to find a schedulable mapping from the design space, better than NN and random search.

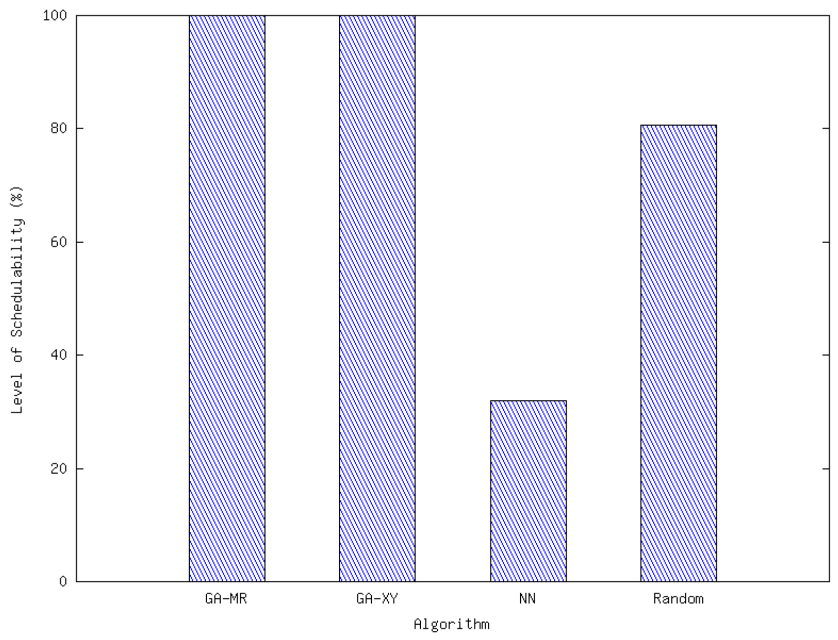


Figure 4. Schedulability level comparison between GA-MR and benchmarks for mapping autonomous vehicle test bench on the 4x4 platform

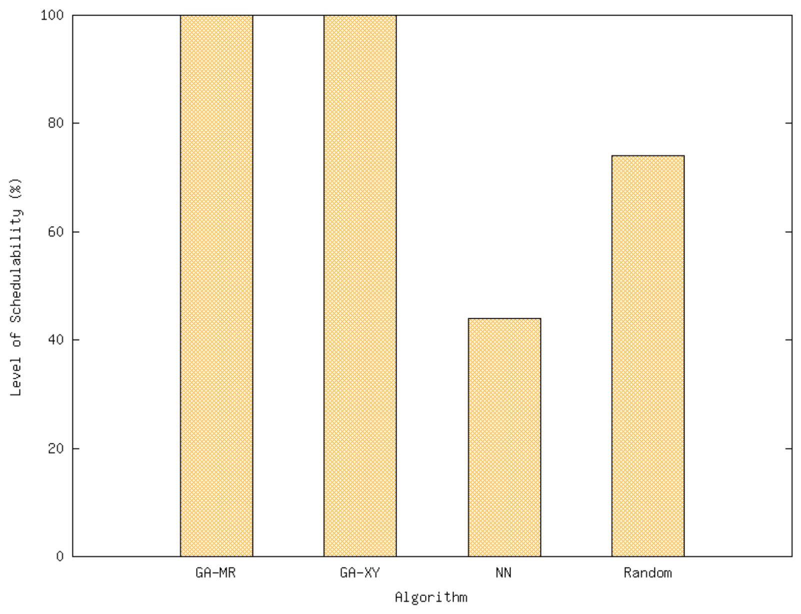
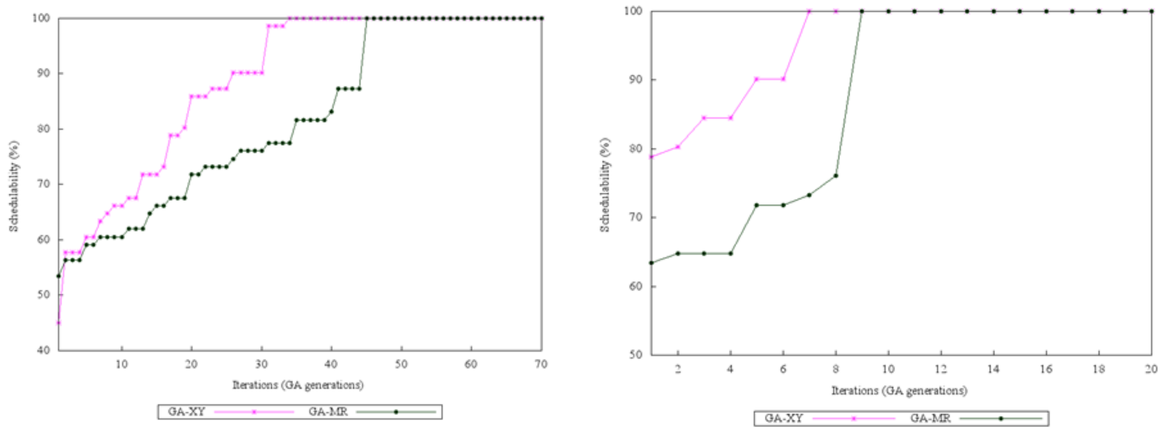


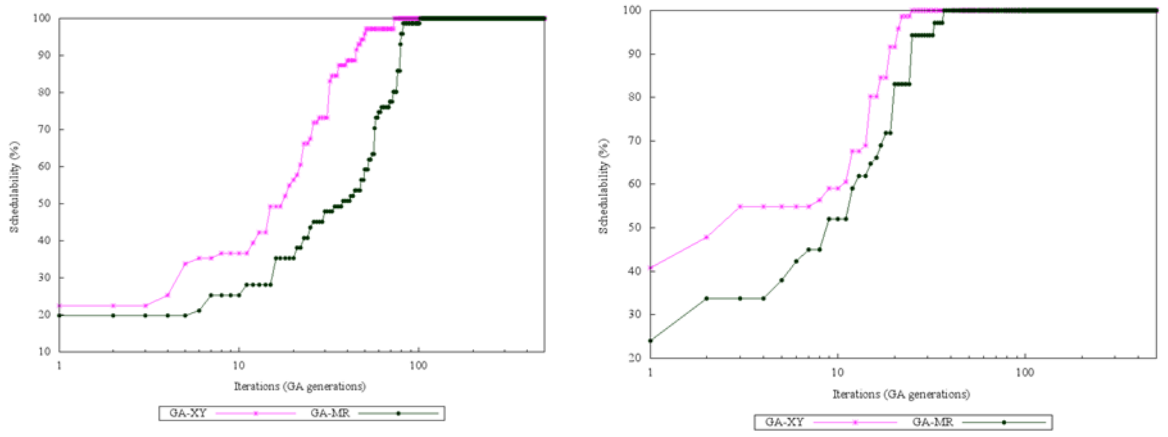
Figure 5. Schedulability level comparison between GA-MR and benchmarks for mapping synthetic test bench on the 5x5 platform

From the results in Figure 4 and Figure 5, it is noted that GA-XY produced similar performance as GA-MR (both achieved 100% schedulability). Since the performance between GA-MR and GA-XY is equal, further studies investigated the convergence rate of both algorithms and the results are shown in Figure 6 and Figure 7. For AVA mapped on the 4x4 platform the results in Figure 6(a) indicates that both algorithms successfully converge to a fully schedulable system in less than 50 iterations. This can be considered fast for both algorithms since convergence is in a short period of time and long before the breaking point (500 generations) was reached. On the larger platform (5x5), whereby both algorithms have more choices of cores to map the tasks and messages, faster convergence in less than 10 iterations was recorded by both algorithms (refer Figure 6(b)). For SA mapped on the 5x5 platform, GA-MR converged to a fully schedulable system at slightly above 100 iterations (as compared to GA-XY), but the results in Figure 7(a) shows a small gap between the convergences of both algorithms. Indeed, both algorithms successfully found a schedulable mapping in less than 100 iterations on the 6x6 platform (refer Figure 7(b)).



(a) (b)

Figure 6. Schedulability convergence of mapping autonomous vehicle test bench onto the, (a) 4x4 platform, (b) 5x5 platform



(a) (b)

Figure 7. Schedulability convergence of mapping synthetic test bench onto the, (a) 5x5 platforms, (b) 6x6 platforms

According to the results (Figure 6 and Figure 7) it is noted that GA-XY achieved early convergence than GA-MR. This is due to the fact that GA-MR explores a wider design space (mapping and routing simultaneously configured), thus to arrive at the schedulable configuration, it may require more iteration. Nevertheless, in spite of the late convergence GA-MR guarantees that the mapping and alternative routing it finds is schedulable for the system, unlike GA-XY (since it only optimize on the mapping).

In spite of the multi-dimension search it can be implied that GA-MR able to find a schedulable configuration from the large design space. This is based on the results shown in Figure 4, Figure 5, Figure 6 and Figure 7. In order to further demonstrate the ability of GA-MR in supporting alternative mappings, we studied the GA-MR's configurations to find out if the mappings it produced is compatible with the static XY routing. This would provide an alternative routing in addition to the one produced through GA-MR search. Table 2 and Table 3 shows the schedulability results of the GA-MR's configurations (3rd column) and the mapping with the substituted XY routing (4th column), occurs at the iteration when it converged to a fully schedulable system (2nd column). As shown in the Table 2, on the 4x4 platform, AVA is 100% schedulable with the XY routing and similar results was yielded for the 5x5 platform as well. For SA, results in Table 3 shows 100% level of schedulability not only with the GA-MR routing but also with the XY routing on the 5x5 and 6x6 platforms.

Table 2. Schedulability of GA-MR Mapping with XY routing (AVA)

Platform	Iteration	Schedulability (GA-MR)	Schedulability (XY)
4x4	45	100%	100%
5x5	9	100%	100%

Table 3. Schedulability of GA-MR Mapping with XY routing (SA)

Platform	Iteration	Schedulability (GA-MR)	Schedulability (XY)
5x5	102	100%	100%
6x6	37	100%	100%

6. CONCLUSION

Interference suffered by low priority tasks and messages can be reduced by changing the task mapping and routing. Based on this notion, we proposed an optimization technique that configures both task mapping and routing simultaneously using a GA-based algorithm. The algorithm's fitness function evaluates the configurations in terms of tasks and messages schedulability. Experimental results based on the synthetic and realistic test benches shows the configuration found by the proposed algorithm is better than the benchmarks. In future work, we will explore additional design spaces such as network buffer to expand the ability of the proposed algorithm.

ACKNOWLEDGEMENTS

The authors are grateful to the Universiti Sains Islam Malaysia (USIM) for providing financial support for this research work under grant PPP/USG-0116/ISI/FKAB/30/12616.

REFERENCES

[1] J. Lehoczky, L. Sha, and Y. Ding, "The rate monotonic scheduling algorithm: exact characterization and average case behavior," in *Real Time Systems Symposium*, 1989.
[2] P. Mesidis and L. S. Indrusiak, "Genetic mapping of hard real-time applications onto NoC-based MPSoCs–A first approach," in *Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC)*, 2011.
[3] A. Racu and L. S. Indrusiak, "Using genetic algorithms to map hard real-time on NoC-based systems," in *ReCoSoC*, 2012.
[4] J. Duato, S. Yalamanchili, and L. M. Ni, *Interconnection networks: An engineering approach*. Morgan Kaufmann, 2003.
[5] W. J. Dally, "Virtual-channel flow control," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 3, pp. 194–205, 1992.
[6] E. Bolotin, I. Cidon, R. Ginosar, and A. Kolodny, "QNoC: QoS architecture and design process for network on chip," *Journal of Systems Architecture*, vol. 50, pp. 105–128, 2004.
[7] F. Moraes, N. Calazans, A. Mello, L. Möller, and L. Ost, "HERMES: an infrastructure for low area overhead packet-switching networks on chip," *The VLSI Journal Integration*, vol. 38, pp. 69–93, 2004.
[8] J. Hu and R. Marculescu, "Energy-aware mapping for tile-based NoC architectures under performance constraints," in *Design Automation Conference*, 2003.
[9] H. Jingcao and R. Marculescu, "Exploiting the routing flexibility for energy-performance aware mapping of regular NoC architectures," in *Design, Automation and Test in Europe Conference and Exhibition*, 2003.
[10] G. Fen and W. Ning, "Genetic algorithm based mapping and routing approach for network on chip architectures," *Chinese Journal of Electronics*, vol. 19, no. 1, 2010.
[11] S. Murali and G. De Micheli, "Bandwidth-Constrained Mapping of Cores onto NoC Architectures," in *The conference on Design, Automation and Test in Europe*, 2004.
[12] M. Srinivasan and G. De Micheli, "SUNMAP: a tool for automatic topology selection and generation for NoCs," in *Proceedings of the 41st annual Design Automation Conference*, 2004.
[13] M. N. S. M. Sayuti and L. S. Indrusiak, "Real-time low-power task mapping in networks-on-chip," in *VLSI (ISVLSI), 2013 IEEE Computer Society Annual Symposium on*, 2013.
[14] M. N. S. M. Sayuti, L. S. Indrusiak, and A. Garcia-Ortiz, "An Optimisation Algorithm for Minimising Energy Dissipation in NoC-based Hard Real-time Embedded Systems," in *Proceedings of the 21st International Conference on Real-Time Networks and Systems*, 2013.
[15] M. N. S. M. Sayuti and L. S. Indrusiak, "Simultaneous Optimisation of Task Mapping and Priority Assignment for Real-Time Embedded Networks-on-Chip," in *23rd Euromicro Conference on Parallel, Distributed and Network-Based Processing*, 2014.
[16] M. N. S. M. Sayuti and L. Indrusiak, "A Function for Hard Real-Time System Search-Based Task Mapping Optimisation," in *IEEE 18th International Symposium on Real-Time Distributed Computing*, 2015.