A Genetic Method for Hardware-Software Par-Synthesis

Mieczysław Drabowski

Department of Computer Engineering, Cracow University of Technology; Krakow, 31-155 Poland

Summary
The paper presents a coherent approach to solving the problems of computer system synthesis based on genetic methods assisted with simulated annealing strategy. We describe algorithm realizations aimed to optimize resource selection and task scheduling, as well as the adaptation of those algorithms for coherent co-synthesis realization. This is new approach, which we propose call a par-synthesis. We then present selected analytical experiments proving the correctness of the coherent synthesis concept and indicate its practical motivations.

Key words:
Task Scheduling, Resource Selection, Task and Resource Allocation, Co-synthesis, Par-synthesis

1. Introduction

The computer systems synthesis process consists in concurrent definition of both hardware and software components of the designed system. Rapid prototyping of such systems requires assisting the synthesis, which may be achieved through its automation based on implementing synthesis algorithms. When realizing the synthesis, it is essential to define the tasks (representing the requirements and constraints of system design), which are to be completed by the designed computer system, as well as the libraries (pool) of available resources which may realize all system tasks. In the synthesis process, the best resources according to given criteria ought to be chosen. In most cases, they are cost and power consumption, as well as the speed of completing all tasks. Moreover, it is necessary to optimize scheduling and allocating tasks to resources.
So far, resources selection, task scheduling and task and resources allocation in synthesis systems have been realized independently [1, 2, 5]. Optimizing resource selection and task scheduling were also realized irrespectively of each other. In this paper a new process of the so-called par-synthesis is presented, in which resources selection, task scheduling, allocation of tasks and resources, as well as system optimization are realized coherently. Once the tasks have been scheduled and allocated, the resources are reselected, changing the ones chosen previously, in search of a satisfactory solution meeting all the established criteria. The schematic diagram of a par-synthesis of multiprocessor computer systems is on Fig. 1. Due to the fact that synthesis problems and their optimizations are NP-complete [6], we suggest meta-heuristic approaches – genetic in this paper – with Boltzmann tournament selection strategy [7].

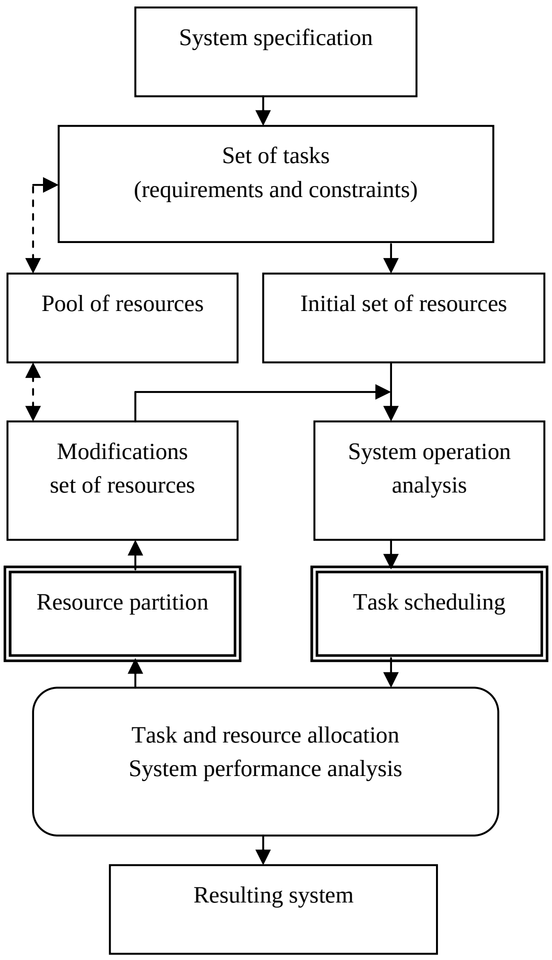

Fig.1. Par-synthesis of multiprocessor computer system

2. Evolutionary Approach

In order to eliminate solution convergence [7] in genetic algorithms, we use data structures which ensure locality

Manuscript received November 9, 2006.
Manuscript revised November 17, 2006.

preservation of features occurring in chromosomes and represented by a value vector. Locality is interpreted as the inverse of the distance between vectors in an n-dimension hypersphere [1, 2]. Then, crossing and mutation operators are data exchange operations not between one-dimensional vectors but between fragments of hyperspheres. Thanks to such an approach, small changes in a chromosome correspond to small changes in the solution defined by the chromosome. The presented solution features two hyperspheres:

i. Task hypersphere – two-dimensional, representing the task graph structure. Each of the vertexes is defined by two coordinates: an indicator obtained through topological sorting (the tasks are "closest" if one of them is a direct successor of the other), and an indicator calculated from the BFS [5] algorithm parallel tasks are equally distant from the beginning of the graph).
ii. Resource hypersphere – three-dimensional, representing the dependencies of resource features. Each of the resources may be defined by the following coordinates – cost, speed and power consumption.

The solutions sharing the same allocations form the so-called clusters [1]. The introduction of solution clusters separates solutions with different allocations from one another. Such solutions evolve separately, which protects the crossing operation from generating defective solutions. There are no situations in which a task is being allocated to a non-allocated resource. Solution clusters define the structures of the system under construction (in the form of resources for task allocation). Solutions are the mapping of tasks allocated to resources and task scheduling. During evolution, two types of genetic operations (crossing and mutation) take place on two different levels (clusters and solutions).

A population is created whose parameters are: the number of clusters, the number of solutions in the clusters, the task graph and resource library. For the synthesis purposes, the following criteria and values are defined: optimization criterion and algorithm iteration annealing criterion if solution improvement has not taken place, maximum number of generations of evolving solutions within clusters, as well as the limitations – number of resource, their overall cost, total time fort the realization of all tasks, power consumption of the designed system and, optionally, the size of the list of the best and non-dominated individuals.

.

3. Resource selection

The input data for resource selection are the task graph, the library of available resources and the optimization criteria, and its goal is to divide tasks into the software and the hardware part and to select resources for the realization of all tasks consistent with the established optimization criteria. The diagram of the algorithm of resource selection is showed on Fig. 2.

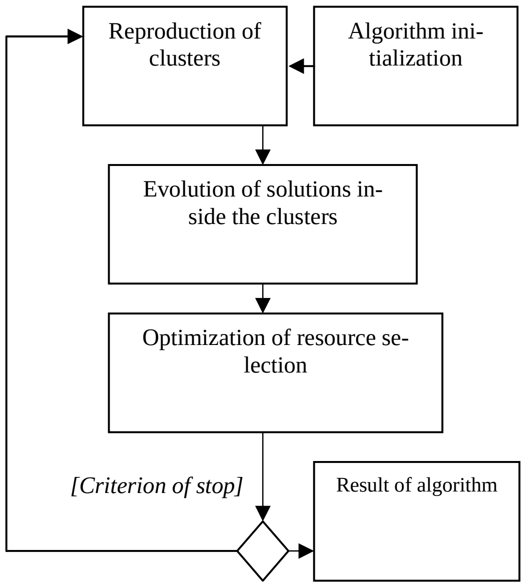

Fig. 2. Algorithm of resource selection

3.1. Algorithm initialization

Initialization of the resource selection algorithm aims at defining preliminary system structure based on the library of available resources matching the required functionality. The global algorithm "temperature" is initialized at this stage as well.

Hypersphere definitions

- filling multi-dimensional vectors with data defining a given object (resources, tasks)
- calculating the diameters of the hyperspheres, i.e. the distance between the two most remote points and determining the hypersphere center on the basis of the extreme coordinates

Population initialization

Clusters and solutions are initialized randomly:

- for every task, a resource capable of completing the task is selected,
- if the resource is allocated, the algorithm proceeds to the next task,
- a resource capable of completing the task is selected and they are allocated.

Initializing the allocation of tasks to resources

- a vector of resources for allocation is taken for each task,
- resource type and number are randomly assigned to the tasks,
- task scheduling by the ASAP [5] algorithm is initialized - eliminates the violations of sequence limitations.

Solution evaluation

- the following are calculated: resource cost, task completion time and power consumption; the cost is the sum of allocated resources' costs, the time of completed tasks is the time of completing the tasks on all allocated resources, power consumption is the sum of power inputs taken by selected resources. Additionally, the idle power consumption (for an inactive resource) is allowed for,
- if for an individual representing a solution any of the optimized criteria exceeds the maximum value acceptable, the individual is punished and the survival chances of a punished individual diminish considerably,
- as the result of the above operations, we obtain a vector containing the values of optimized criteria (time, cost, power consumption),
- a solution ranking is determined (the rating of a given solution is the number of solutions in a population which do not dominate the solution),
- a solution is dominated if each of its costs is lesser from or equal to the costs of the dominant solution (for optimization in the Pareto sense).

Cluster evaluation

A solution cluster ranking is created. The rating of a cluster is the sum of the ratings of all solutions within the cluster.

3.2. Algorithm of resource selection

Cluster reproduction

Clusters are reproduced with the use of genetic operators: crossing and mutation. At the reproduction stage, the cluster population is doubled and its initial size is restored at the elimination stage. This method was introduced arbitrarily and ensures that within a population some new individuals appear and fight for survival with their parents. The mutation operator creates one and the crossing operator two new clusters. The likelihood of using either of the genetic operators is defined by the algorithm parameters.

Genetic operators

The cluster mutation operator consists in mutating allocation vectors in the following way: a cluster with identical likelihood is picked at random and copied. The number of the resource which will be mutated in a new cluster is picked randomly. Then, a number in the 0-1 range is picked - if the number is smaller than the global temperature, the resource is added, otherwise it is subtracted. Adding resources is limited by the maximum resource number parameter. At the beginning of the algorithm operation, resources will be added to the structure. As the algorithm approaches the end of the run defined by the cooling process, resources will be subtracted. This is aimed at creating a cost-effective structure.
The cluster crossing operator consists in randomly picking two clusters and copying them. Crossing is achieved through cutting the resource hypersphere with a hyperplane. The information contained on "one side" of the hyperplane is exchanged between clusters.

The algorithm for cutting the hypersphere with a hyperplane

- determining the cutting hyperplane by picking n points inside an n-dimensional hypersphere,
- creating a random permutation, e.g. for n = 3, the permutation can be (2, 1, 3),
- constructing the point displacement vector in respect to the hypersphere center; square coordinates are picked consistent with dimension permutations, e.g. for three dimensions with the permutation (2, 1, 3): y2 = rand() % r2, x2 = rand() % (r2 – y2), z2 = rand() % (r2 – (y2 + x2)), where: r – hypersphere radius, and (x, y, z) are the coordinates of the constructed point in a three-dimensional space,
- the roots of square coordinates are calculated,
- a coordinate radical sign is picked,
- the hypersphere center coordinates are added to the new point resulting in obtaining a new point inside the n-dimensional hypersphere,
- the equation of the hyperplane cutting the hypersphere is calculated and the obtained system of equations is solved.

Saving the best solutions

After solution reproduction, a new procedure is called to save the globally non-dominated solutions generated during evolution.
This procedure executes:

- searches for non-dominated solutions in the present generation,
- creates the ranking of the best solutions saved so far and in the present generation,
- saves the non-dominated solutions from both the "old" and the "new" solutions,

- deletes the solutions saved in the past if they were dominated by new solutions; if there are more than one solution whose all optimized criteria values are identical, only one of those solutions is saved (the "newest" one),
- if the new solutions dominated none of the ones saved in the past, the population was not improved,
- the number of non-dominated solutions that the algorithm can save is defined by an algorithm parameter.

Cluster evaluation

At this stage of the algorithm, half the individuals are removed from the population. The initial number of individuals is restored. The elimination of individuals is carried out using Boltzmann tournament selection strategy.

4. Task Scheduling

Task scheduling is aimed at minimizing the schedule length (the total tasks completion time). The diagram of the algorithm of task scheduling is showed on Fig. 3.

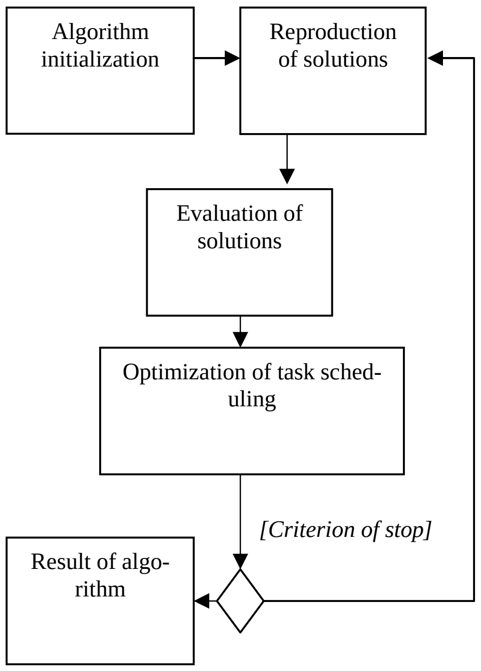

Fig. 3. Algorithm of task scheduling

The scheduling algorithm initialization resembles the initialization of resource selection algorithm. The difference is that there is solely one cluster in which solutions evolve. The cluster allocation remains unchanged during the algorithm's run because all the resources are known for the task scheduling algorithm.

Solution reproduction

Solutions are reproduced using the genetic operators: crossing and mutation. Solutions are reproduced until their number doubles (the number of new solutions has been chosen arbitrarily).
The mutation operator produces one and the crossing operator two new solutions. The likelihood of using either of the genetic operators is defined by the algorithm parameters.

Genetic operators

The mutation operator of task allocation to resources acts in the following manner: a solution is randomly selected and copied. Then, the number of tasks in the system is multiplied by the global temperature. When the global temperature is high, the number of tasks changed in the allocation to resources will be greater than that in later stages of the evolution. Tasks are picked at random and allocated to resources.
The schedule mutation operator acts in the following manner: if due to the mutation operation of task allocation to resources, the resource the task had been running on was changed, then the task is removed from the schedule for the "old" resource and boundaries are set on the new resource schedule between which the task may be allocated. A location within the boundaries is picked and the task is allocated.
The crossing operator of task allocation to resources resembles cluster crossing, however, the task graph hypersphere is used for that purpose. Schedule crossing operator acts in the following way – after the allocations have been crossed, a map is created defining which parent a given feature of an offspring comes from. The offspring stores the allocation vector (obtained after crossing task allocations to resources) and the empty vector of lists with schedules of tasks on available resources.
The algorithm analyzes the tasks by checking their position on the graph. For all tasks in one position, the resources on which the tasks will be performed (defined by the vector of allocation to resources) are put on the list. If in a position there is only one task ran on a given resource, the task is entered into the resource schedule, otherwise the tasks are sorted according to the starting time they had in the parent and are placed in the schedule in ascending order.

Solution evaluation, saving the best solutions and solution elimination

They are the same algorithms which were employed in the resource distribution algorithm. Analogical solutions are eliminated using Boltzmann tournament selection strategy.

Algorithm report

If within the number of generations determined by the annealing criterion a better individual did not appear, the evolution is stopped and the evolution report is created. The result of the algorithm operation is a set of non-dominated individuals (in the scale of the whole calculation process).

5. Coherent Resource Selection and Task Scheduling – Par-Synthesis

The diagram of the algorithm of the coherent resources selection and tasks scheduling according to genetic approach, is showed on Fig. 4.

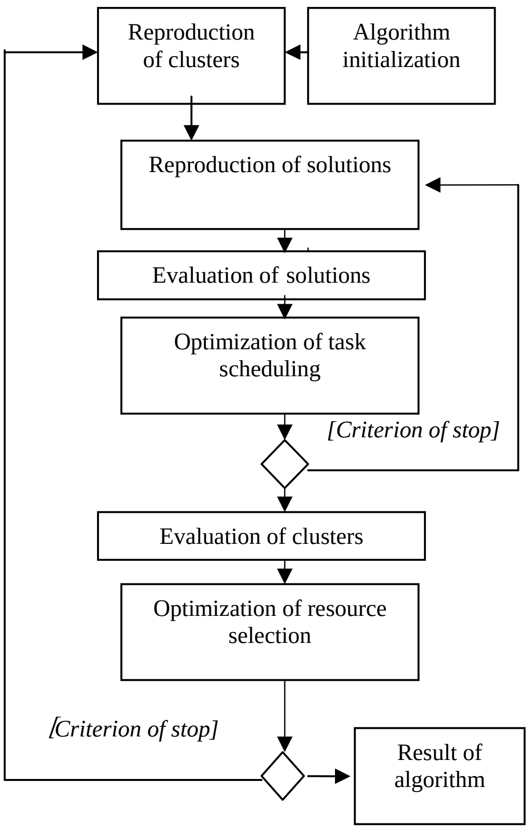

Fig. 4. The coherent co-synthesis (par-synthesis) of computer system – genetic approach

The initialization of the coherent synthesis algorithm resembles the initialization of resource selection algorithm. The input parameters are the number of clusters in the population and the number of solutions in clusters.
Solution clusters represent the structures sharing the same resource allocation, but with different task allocation to resources and different schedules. The outer loop of the algorithm (realizes resource selection) is ran until the number of generations without population improvement is exceeded. This value is defined by the annealing criterion parameter. There are few outer loops at the beginning of the algorithm operation. Their number grows until it reaches the value of k with the falling of the temperature. Fewer task allocations and scheduling processes are performed at the beginning. When the temperature falls sufficiently low, each inner loop has k iterations. The number of iterations may be regulated with the temperature step parameter. The greater the step, the faster the number of inner iterations reaches the *k* value.

6. The Results of Analytical Experiments

We present the analytical results obtained by testing the presented algorithms. In the tests represented by the flowcharts below, we compared the results from the incoherent and coherent syntheses.
During cost optimization, both algorithms yielded similar cost values for all tested task sets.
However, the coherent algorithm improved time optimization for graphs exceeding 30 tasks. For 50 tasks it achieved a 15% improvement of the task completion time – Chart 1.

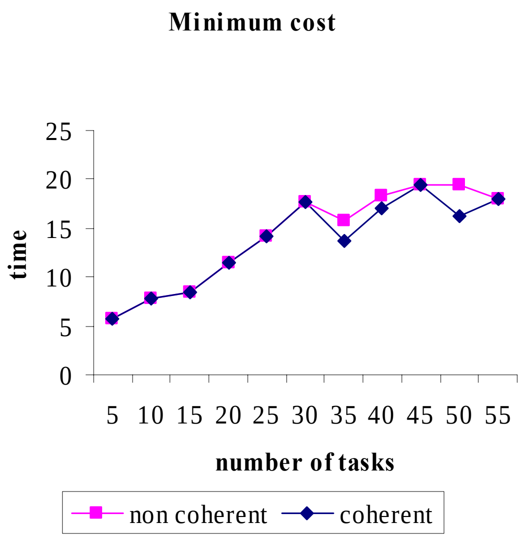

Chart 1. Non coherent and coherent synthesis (par-synthesis) for minimum cost

When the flowchart reflecting the dependence of time from the number of system tasks is considered (Chart 2), time minimization is comparable for both algorithms. Nevertheless, once the flowchart showing the interdependence of cost and the number of tasks is analyzed (Chart 3), it is clear that the solutions yielded by the coherent algorithm are far less expensive than those from the incoherent algorithm. The coherent algorithm achieves similar task completion times in solutions much cheaper from those found by the incoherent algorithm.

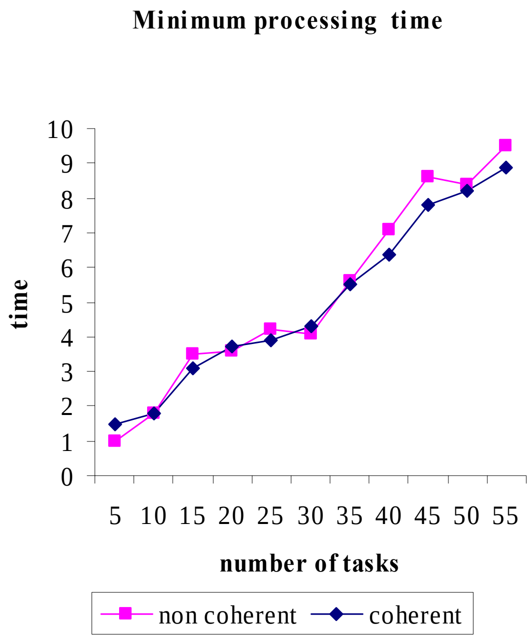

Chart 2. Non coherent and coherent synthesis (par-synthesis) for minimum time

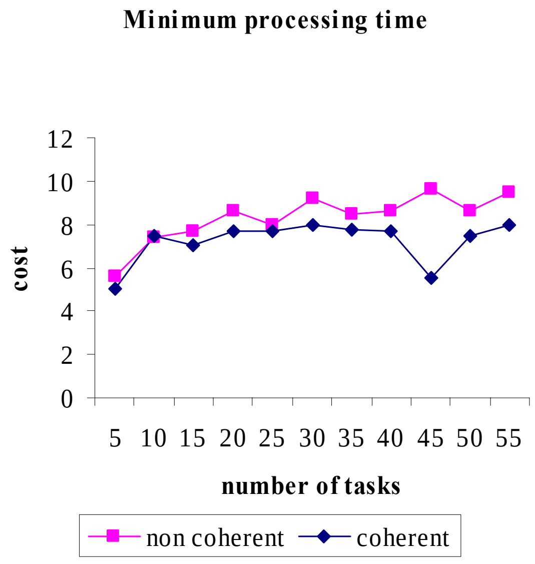

Chart 3. Non coherent and coherent synthesis (par-synthesis) for minimum time

7. Conclusion

The paper describes genetic algorithms and their implementation flowcharts. Moreover, it presents selected results of analytical experiments for resource selection and task scheduling. The paper explores the coherent synthesis algorithm of computer systems, in which resource selection and task scheduling optimization processes are realized concurrently and coherently. The coherent approach in the synthesis generates common and interdependent solutions regarding the system structure (type and configuration of the selected resources), as well as the scheduling of tasks ran on those resources. In the presented approach, the cost of resources (system cost), the time of completing all tasks (system speed) and the power consumption of the system are optimized. The coherent algorithm yields much (up to 40%) better solutions, which is proved by analytical experiments. The solution suggested in the paper may be applied in supporting computer system prototyping, for dependable and fault-tolerant [3, 4] multiprocessors systems, too.

References

[1] Dick R. P., Jha N. K., "CORDS: Hardware-Software Co-Synthesis of Reconfigurable Real-Time Distributed Embedded Systems". In: *Proc. Int. Conf. Computer-Aided Design*, 1998, pp. 62 – 68.
[2] Dick R. P., Jha N. K., "MOGAC: A Multiobjective Genetic Algorithm for Hardware-Software Cosynthesis of Hierarchical Heterogeneous Distributed Embedded Systems". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 17, no. 10, *Oct.* 1998, pp. 920 – 935.
[3] M. Drabowski, K. Czajkowski, "Minimizing Cost and Minimizing Schedule Length in Synthesis of Fault Tolerant Multiprocessors Systems". In: *Lecture Notes in Computer Science, Springer Berlin/Heidelberg*, vol. 3911, 2006, pp. 986 – 993.
[4] Drabowski M, Wantuch E., "Coherent concurrent task scheduling and resource assignment in dependable computer system design". In: *Int. Journal of Reliability, Quality and Safety Engineering*, vol. 13, No. 1, 2006, pp. 15 – 24.
[5] Gajski D. D., *Principles of Digital Design*, Prentice Hall, Upper Saddle River, NJ, 1997.
[6] Garey M. R., Johnson D. S., *Computers and intractability: A guide to the theory of NP-completeness*, San Francisco, Freeman, 1979.
[7] Rothlauf F., "On the Locality of Representations of Integers on the Performance of Selectorecombinative Genetic Algorithms". In: *Genetic and Evolutionary Computation – GECCO, Erick Cantu-Paz et al. (Eds.), Lecture Notes in Computer Science, Springer Berlin/Heidelberg*, vol. 2724, 2003, pp. 1608 – 1619.

Mieczysław Drabowski is the Assistant Professor of Department of Computer Engineering, Faculty of Electrical and Computer Engineering, Cracow University of Technology. He received the M. Sc. degree in automatic control and communication from AGH University of Science and Technology in Krakow, he graduated mathematic from Jagiellonian University in Krakow and he received the Ph. D. degree in computing science from Poznan University of Technology, in 1977, 1979 and 1986, respectively. He has eighteen years of industrial experience in design of computer systems and in software engineering.

His research interest includes scheduling, assignment and allocation for tasks and resources, dependable system modeling, operating system and software engineering. Dr. Drabowski is a member of the council of the Polish Information Processing Society.