NOC-based Embedded Systems Co-synthesis – Developmental Genetic Programming Approach

Dariusz Dorota

Cracow University of Technology; Krakow, Poland

Summary
In the presented paper presents a new method to generate the architecture of embedded systems based on the developmental genetic programming. Previously used methods, based on developmental genetic programming in their chromosomes represent solutions. In the presented approach, chromosomes are used to construct a system. Evolution in the depicted case does not concern but the co-synthesis process. A proposal of represent the process in the form of trees forming so-called. Genotype. Despite the fact that the architecture of Network-on-Chip (NoC) is an interesting alternative to a bus architecture based on multi-processors systems, it requires a lot of work that ensures the optimization of communication. This paper proposes an effective approach to generate dedicated NoC topology solving communication problems. Network NoC is generated taking into account the energy consumption and resource issues. Ultimately generated is minimal, dedicated NoC topology. The proposed solution is assumed to be a simple router design and the minimum number of lines.

Key words:
Network on Chip, NoC-based embedded system co-synthesis, Developmental Genetic Programming.

1. Introduction

Currently designed embedded systems are characterized by increasing complexity, which is a result of integration of many different functions in a single system [1]. Another aspect of the development of embedded systems is ever increasing hardware requirements related to cost, power consumption, short time of design and speed. To meet these requirements it is necessary to use computer-based tool supporting the design of specialized embedded systems architectures. Should like to concentrate on the development of effective methods to optimize the design of embedded systems, taking into account the above described parameters. The term synthesis system, refers to the process of computer architecture design. If that process surrender embedded systems, combining both hardware components and software, it assumes the name co-synthesis hardware-software. Some papers of co-synthesis [2]. adopt the simplified architecture of the system, which consists of one general purpose processor and one processor in the form of dedicated ASIC or FPGA, for such cases the problem boils down to division of functions between hardware and software. Generally not accepted such limitations, so the co-synthesis results is to obtain the mapping specification systems architecture consisting of different types of processors, specialized modules, etc., the system having a distributed and heterogeneous nature. Co-synthesis process is the automatic generation of embedded systems architectures based on the specification given in the form of current processes. The aim of co-synthesis is to optimize the properties of the system, such as cost, execution time and power consumption. Most modern methods of co-synthesis assumes a distributed architecture consisting of computational elements PE (Processing Element) which are hardware components HC (Hardware Core) or software-PP (Programmable Processor). Co-synthesis process consists the following tasks:

- Allocation of CPUs and communication channels
- Assignment of tasks to resources
- Scheduling and transmission

The allocation is a step in which the system architecture is defined. To make this possible, it is necessary to define the available library modules and their parameters (the speed of execution tasks, cost, etc.). The assignment of tasks to processors is a generalized problem of the division of tasks between hardware and software. This step is being closely related to the allocation is typically performed in parallel with it. And scheduling is to determine the order of execution of each task. Both scheduling, as well as the assignment of tasks to resources is NP-complete problems. Therefore, efficient heuristics should be applied to find the best target architecture for a given system. Moreover, in case of NoC systems, one of the most serious research problem is application to NoC mapping with the respect to performance and power consumption constraints and communication contention control [4], [15], [16].

One of the most important problems and design is a process for mapping application in NoC network, which takes into account the limitation of the operation speed, power consumption and solving communication conflict. Due to the specialized nature of embedded systems, in most cases, they are characterized by a predictable model of consumption and transmission [5]. Such application can be represented by a graph of tasks (TG) and used in the target NoC architecture. Use the graph of tasks can by optimized by applying exists algorithms such as allowing for finding shortest path and then use appropriate ways of

Manuscript received November 5, 2013
Manuscript revised November 20, 2013

mapping in the NoC network. Mapping in NoC network is performed using common heuristic algorithms [6,7] and genetic algorithms [8,9]. Design algorithm have low computational complexity, but also tend to pick up in local minima optimized parameters. The quality of the results can be improved by adding a relapse in the design of solution [10], but this creates a risk of entanglement in the algorithm, thereby increasing the computational complexity. The ability of escaping from local minima has genetic algorithms [11]. Genetic methods are also commonly used to optimize the selected stages of the synthesis system.

This paper presents new method hardware-software co-synthesis based on developmental genetic programming [12,13]. The existing genetic methods of co-synthesis define the evolution of the refining architecture, where chromosomes define solution. An exception is the work[14], which is a process of evolution of co-synthesis. However, in this work the authors assumed that each node of the tree genotype will implement the task corresponding to given node in the graph of tasks, which is not entirely consistent with the principles of genetic programming, in which evolution should proceed without restriction. The approach proposed in this article allows to obtain a method for constructing the target system. Presented at work algorithm is subjected co-synthesis process to evolution, dividing the tasks of the graph on different types of genes. In next section, you will learn the problem co-synthesis of embedded systems.

2. The System on Chip co-synthesis problem

The behavior of the designed embedded system can be described by a task graph G ={V,E}, which is directed acyclic graph. Each node represents a task, and the edge describes the relationship between tasks v_i and v_j. An edge $e_{i,j} \in E$ describes a dependency between tasks v_i and v_j. Each edge is annotated with a number $d_{i,j}$ describing the amount of data that have to be transferred between the two connected tasks. A sample task graph is presented on Fig.1.

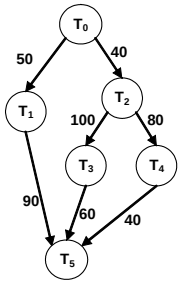

Figure 1. Sample task graph.

One of the objectives of the existing database of computing elements, containing the task execution times(t), surface module (s) for implementation System on Chip. There are two basic types of PE: purpose programmable processor (PP) and specialized hardware modules (HC). Each PP_j can perform all the tasks that are compatible with them. By PP_i task size is determined by the memory footprint required by the task, and SPP_i determines the area of PP_i. HC hardware module performs only one task v_i, but there may by multiple hardware implementations of the same task. All communication channels have the same bandwidth defined by b_{NOC}.

Table 1 shows an example of the resource base for the system described by the task graph in figure 1.

	PP₁ S=300		**PP₂** S=400		**HC₁**		**HC₂**	
	t	**s**	**t**	**s**	**t**	**s**	**t**	**s**
T₀	200	6	150	6	60	180	40	250
T₁	50	4	40	3	24	90	15	150
T₂	250	16	190	18	170	200	150	500
T₃	220	13	140	14	110	140	100	300
T₄	150	12	120	15	70	20	30	50
T₅	60	5	55	5	35	110	-	-

Task T_2 is not compatible with component PP_1, and T_5 have only one hardware implementation. All other tasks are four alternative implementations. Each task v_i can have a specific requirement and time c_i, indicated the moment of time in which the task must by performed. Let ts_i will be a start time of task v_i. Created system is the correct solution if and only if the following conditions are met.

$$\forall e_{i,j}: ts_j \geq ts_i + t_{i,r_i} + tc_{i,j} \quad (1)$$

$$\forall i: ts_i + t_{i,r_i} \leq c_i \quad (2)$$

If the task v_i are assigned to the same pe_i, the $tc_{i,j} = 0$. Condition (1) implies the correct ordering of tasks, while (2) ensures that all timing requirements. If we assume that the target architecture for the system described task graph containing n process, consists of m programmable processors, p communication bus, the total area occupied by the SOC system is given by:

$$tc_{i,j} = \left\lceil \frac{a_{i,j}}{b_{NOC}} \right\rceil \quad (3)$$

Where s_k is the surface k-th communication channel. Thus, if A is the cost system, the purpose of the optimization is to find architecture witch smallest A under the conditions

1 and 2. If tIn the proposed methodology, the minimization of two components contained in the first of (4) is performed during co-synthesis, while the other parameters are optimized when the NoC topology is generated.

$$A = \sum S_{PE_i} + \sum s_j + \sum s_k \quad (4)$$

Result of the co-synthesis is specified using the Attributed Task Graph (ATG), which is an input model for the second step performing topology generation. ATG is a directed acyclic graph, where vertices T_i are tasks and edges $M_{i,j}$ are messages sent between them (i,j are sender – receiver task indices). Every graph node T_i has information about assigned processor PE ($P(T_i)$), execution time $|T_i|$ (measured in clock cycles), and pre-scheduled start time $t_{START}(T_i)$. Similarly, edges are marked with transmission volume $|M_{i,j}|$ (measured in flits) and with pre-scheduled start time $t_{START}(M_{i,j})$. In our model, volume of the message equals its transmission time through the network, regardless of the hop count. It is because many current NoC routers offer latency of 1-2 cycles, so for wormhole switching and large messages latency caused by routers is omitted.

3. NoC system architecture with router.

Depending on the input model applications and target NoC topology, synthesis methods can be divided into the following groups: Synthesis of ACG in architecture regular/irregular, TG synthesis in regular/irregular architecture. Selection of regular NoC topology simplifies the design process, because the design decision then narrows the search for subset of solutions. Previously, the algorithms take as input the task graph. This approach allows for the benefits, but also some limitations. Task Graph (TG) is used as a representation of the application process, used with good results in [3-18].

Most of the existing algorithms of co-synthesis is a refinement [4,5]. In these methods, starting from sub-optimal solutions, during an operation algorithmic is performed improvement of the system by local modifications of architecture. Typically, in such cases as the initial solution is proposed architecture ensures the execution of all tasks in the shortest possible time, where each process is handled by another PE. Aiming to obtain the best system are transferred process to other PE, removes and adds PE, etc.

Network on chip is constructed from multiple point-to-point links interconnected by routers. In this paper assume a router capable of establishing many simultaneous full-duplex connections [17], with wormhole switching and source routing. The router has one Local Port (LP, connected to one NI - Network Interface of PE) and up to 4 inter-router, full-duplex ports – like in many popular mesh approaches. One port can use two unidirectional links: input and output. Only one-flit input buffer is necessary. One router supports one PE module, one PE is attached to one router, and there is no routers without PE attached to it. The concept of the router is presented in Figure 2.

The restriction to impose heuristic algorithms have been level, after propose to use in the synthesis of NoC network probabilistic algorithm [13]. Particularly good results were obtained after application of the genetic algorithms in [11]. In many works are presented approach to the multi-mapping space, based on mesch structure of the NoC architecture, using evolutionary computation techniques. Usually in this work are used heuristic mechanisms based on multitasking genetic algorithm to explore the space and finding the Pareto mappings the optimize performance and power consumption. NoC design allows for hardware virtualization, which can map one or more logical tasks on a single PE. This allows the preparation of PE for the calculation for one or more logical calculations.

The results of this approach is the optimal way to map and to reduce the communication and the highest temperature in comparison with a purely random mapping of the network. In some cases, the proposed hierarchical genetic algorithm reduces the energy demand in NoC. According to the present state of our knowledge this the first work presents the use of developmental genetic programming in the synthesis of embedded systems.

Network on chip is constructed from multiple point-to-point links interconnected by routers. We assume a router capable of establishing many simultaneous full-duplex connections [17], with wormhole switching and source routing. The router has one Local Port (LP, connected to one NI - Network Interface of PE) and up to 4 inter-router, full-duplex ports – like in many popular mesh approaches. One port can use two unidirectional links: input and output. Only one-flit input buffer is necessary. One router supports one PE module, one PE is attached to one router, and there is no routers without PE attached to it. The concept of the router is presented in figure 2. Co-synthesis algorithm is designed to perform allocation and mapping stages, also aims to scheduling of tasks and messages.

During this step every port-based contention is resolved, but potential conflicts between messages transmitted by different PEs are omitted. Has been adopted that must be met four principles:

- PE executes only one task at the same time without pre-emption
- NI (NI - Network Interface of PE) can send only one message at the same time without interleaving the messages

- tasks can start when have received all messages, and message can start when its task (producer) is finished

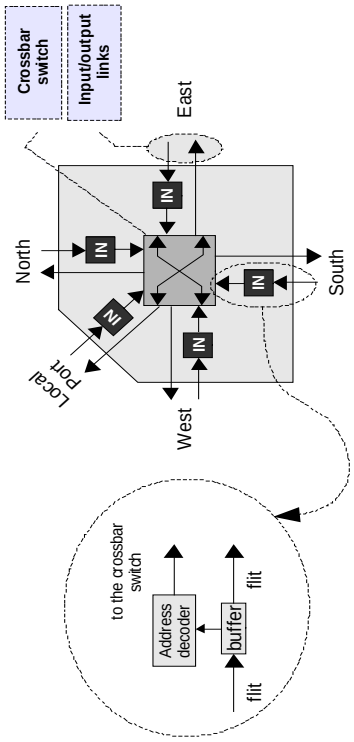

Figure 2. Router architecture

- Every PE module has its own local memory and the programming model is message-passing. If it will be not possible to find the contention-free topology then rescheduling of transmissions and/or tasks is performed.

To estimate the quality of the generated NoC topologies were proposed three measures of assessment:

- execution time of the whole system - the longer time of execution system, makes worse the performance, which also affects the static energy consumption of the system,
- number of links used to build topology - From the point of view of energy and resources in the system is as small as possible all the desired quantity links. [18]. By minimizing the number of connections can facilitate the synthesis of systems.
- average hop count - impacts the both of these factors: performance and power of NoC. The longer route means more latency during transmissions; besides it Affect dynamic energy consumption [19].

We defined weighted average (5), where n is the total number of messages, $|M_{i,j}|$ is duration of the $M_{i,j}$ message and $hop(M_{i,j})$ is the number of hops for $M_{i,j}$. Thus longer transmissions sent through more routers impact more on the average hop than the shorter ones.

$$hop_{AVG} = \frac{\sum_1^n |M_{i,j}| \cdot hop(M_{i,j})}{\sum_1^n |M_{i,j}|} \quad (5)$$

4. Developmental Genetic Progrmming

Genetic Programming (GP) [13] is a method for optimizing involving the generation of a genetic algorithm. Optimization in genetic programming is evolution of the tree structure of the program, in order to get the best program, which, for sets of input data generates the expected results. Genetic Programming has been successful to generate efficient programs that perform mathematical calculations robot control algorithms, text recognition, data analysis, and many other fields. There is 36 known problems for which using the method of genetic programming solution is found comparable or better than found manually.

The need for optimization problems showed strongly reduced, that genetic programming is not sufficient to meet the requirements of a functional or timing. Been proposed an extended version of PG, that is the Developmental Genetic Programming (DGP) [14]. In this method, instead of programs (complete solutions) are generated algorithms for constructing a solution to the problem.

These algorithms are represented by a tree, where the nodes correspond to the decisions of the design, and the edges describe the order. DGP developed an effective method for solving many problems in the field of linear design of electronic circuits (filters, amplifiers) logic synthesis, and others. But so far not tested the applicability of the method DGP in co-synthesis of embedded systems.

Development of genetic programming method is especially effective for the optimization problem strongly limited (for example, the co-synthesis problem). In such cases, the solutions provided by traditional genetic approach or deterministic does not meet the requirements. Traditional genetic approach imposes constraints control the optimization process in a manner considered to be only correct solution. However, these restrictions also narrow the search space, while making it impossible to find the best solutions if they are the result of the evolution of refining or incorrect solutions. These problems do not occur in the DGP, where evolution takes place without any restrictions, while the correctness of this solution is provided by the mapping genotype to phenotype. This feature allows the DGP method in many cases get a lot better results than previously used methods.

5. The Methodology

During the first step the co-synthesis of the system specified by task graph is performed. In accordance with the principle of developmental genetic programming the

developed algorithm evolution of the subject tree (genotype) describing the construction of the designed system. Root of tree sets embryonic system design, nodes correspond to those building the system. To ensure the implementation of all tasks, the structure of the genotype is a tree mapper graph tasks. Each node of the tree genotype implements the corresponding part, which had previously separated. The structure of the genetic operations. Co-synthesis allocates PEs, assigns tasks to the allocated PEs and performs initial scheduling of all tasks and messages. Communication links are not considered in this step, it is assumed that all messages may be sent as soon as possible i.e. without collisions (the goal of the second step is to satisfy this assumption). Co-synthesis optimizes the total system cost (taking into consideration only the cost of PEs and routers) while preserving all time constraints. Optimization is performed using the developmental genetic programming.

Results of the co-synthesis method is presented as attributed task graph, and then use to generate a dedicate NoC topology using the proposed approach. If it is impossible to create a desired topology according to the proposed methodology is performed rescheduling tasks. If rescheduling is not possible, then proposed methodology make modifications of violates time requirements and then re-synthesis is performed for tighter time constraint, but this case is very sporadic. During this step the NoC is optimized using deterministic methods.

5. 1 The embryonic

Embryo implements the first of the generated (a division algorithms graph) part of the graphs. The number of embryos is equal to use the number of split graph algorithms. For each embryo solution is randomly generated. The root of this tree specifies the construction of an embryonic system, while all other nodes correspond to functions progressively building the system.

5.2 Genotype and initial population

Genotypes are structured in a tree. Each node corresponds to function implementing some tasks. To the root node has been assigned the whole graph. Function assigned to a node select a specified number of tasks and then implement them. Other and unallocated tasks are partitioned into groups and assigned to parent nodes. The functions associated with the nodes are responsible for the criteria that determine the allocation, task selection and task partitioning. The initial population for each solution are generated randomly. N determines size of initial population, where N is number of tasks. Size of the initial population is defined by the parameter α, such that: population_size=α*N*r (N is the number of tasks and r is the number of PEs in the resource database).

5.3 Algorithm of creating system

Mapping genotype to phenotype should always generate valid system. Creating system should meet the all the constraints. System is constructed by executing functions corresponding to the following nodes of the genotype browsed in the breath-first order. Extreme system is creating according to the algorithm presented on Fig 3. As extreme system in this paper is considered the system with largest sum of stack times for all deadlines. In this paper slack time mean difference between the deadline and the finish time of constrained task. First, each task is allocated to the PE which executes this task in shortest time and the system is scheduled using ASAP (As Soon As Possible) method. Next, if it is possible to increase the system speed by reassigning any pair of communicating tasks to the same PP (in this case transmission is omitted) then such reassignment is registered. Each task may by reclassification more than once, thus all multiple reassignments should be resolved by choosing the best one. All possible conflicts are presented on Fig. 4, in all cases the reassignment giving largest increase of the slack time is chosen.

```
1: for every T_i in TG do
2:      assign_fastest_PE(T_i);
3: ASAPScheduling();
3: for every adjacent_pair(T_i,T_j) do
4:     for every PP_k in resource_database
do
5:          if ( t_k(T_i)+t_k(T_j) < t_f(T_i)+
t(M_i,j)+t_f(T_j)) then
6:            save_reassignment(T_i,T_j,PP_k)
7: ResolveAllReassignments();
```

Figure 3. Algorithm for construction of extreme system

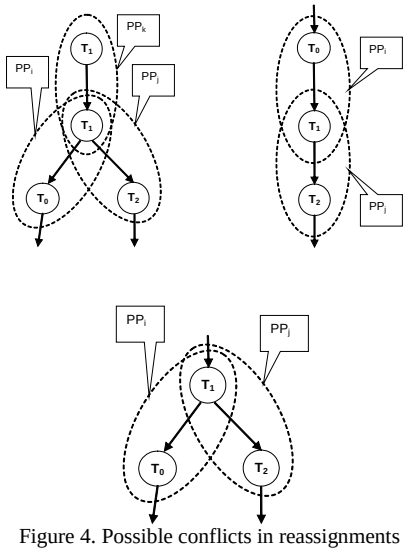

Figure 4. Possible conflicts in reassignments

All genes possess sets out the rules to changing the implementation and through the use of specific genes is constructed system. There is possibility to applied only design decisions that do not violate time constraints. In this way each genotype is always mapped onto valid phenotype.

5.4 System-construction functions

Genotype which has tree structure represents in every node a function implementing some tasks from the task graph. First it allocates one PE. If PE is a hardware core then only one task is chosen (first task from the group) and assigned to this core, otherwise a group of tasks is implemented. In the proposed system, we can distinguish two types of genes: genes and gene mapping division. In each population genotype is randomly generated. On fig5 is showed simple random choosing genotype and on fig 6 is shown graph shared by genotype from fig5. For the selected tracks are laid down time limits that the performance of the functions of the system are examined using the algorithm Dynamic Time Limit (DTL). Figure 7 shows an alignment tasks in the processors in the chosen region NoC network. The last step is to create a complete network of NoC architecture, combining all its regions and, if necessary, appropriate rescheduling task, as shown in Figures 8a and 8b. Then, using the corresponding genes are created NoC network regions, which are mapped to a particular task. Each node in the tree genotype represents a function to generate the target a network of NoC or region of NoC. Constructing the system is in accordance with the following steps:

1. Random selection genotype. With the available resources genes sharing and mapping is drawn genotype.
2. Selection of the gene responsible for the division of the graph, the set of genes sharing. This step is always carried out, as the input data is assumed the graph or a portion thereof.
3. Performing split graph / part of the graph according to the rules defined in the selected gene division. After selecting the appropriate gene in step 1 is carried out the division of input data to the next layer genotype (graph / part of the graph) into smaller parts.
4. Selection of the gene responsible for the implementation of the graph / graph fragment to the region of the NoC network
5. Mapping graph / graph fragment to NoC network. After selecting the appropriate gene in step 3 is performed mapping input data (graph / part of the graph) in the region NoC network.
6. The combination of all regions NoC networks grouped into one network. After mapping all the tasks in the respective regions NoC network is made one single network NoC.

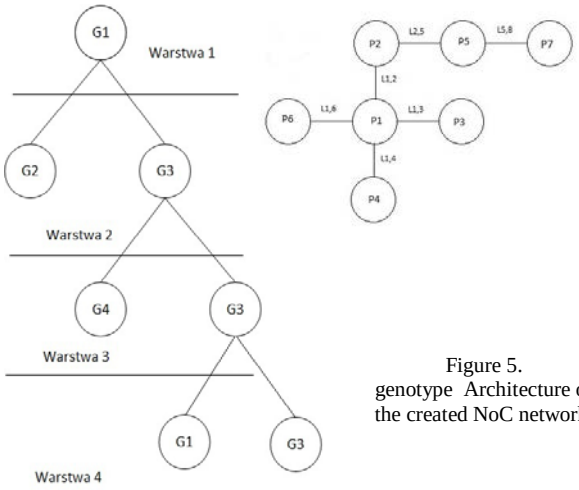

Figure 5. genotype Architecture of the created NoC network.

Figure 7.Simple genotype

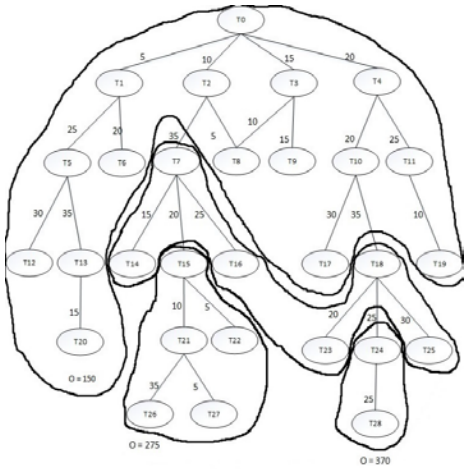

Figure 6. Task Graph after sharing by genotype from Figure 5

In Table 2 are presented preferences for each system-construction function. At the beginning of the initial generation is generated composed of randomly generated composed of randomly generated genotypes. The size of initial generation defines the parameter α and is: $\pi = \alpha * n * e$ (n - is the number of tasks, and e is the number of possible embryos). In the initial population, preferences for all steps are selected randomly, with the probability given in the last column. During mutation to modify the gene is used the same probability.

Preferences for allocation of a new PE are the following:

(a) Hardware component – type of HC is chosen according to preferences for step 2;
(b) PP occupying the smallest area;
(c) PP executing in shortest time all tasks assigned to gene;

(d) PP minimizing the critical path (path with shortest slack time), i.e. if after executing steps 2 and 3, the critical path is the shortest one;
(e) PP that executes the first task from the group in the shortest time;

Table 2. System-construction preferences used for genotype to phenotype mapping

Step	Preference	P
1	a. HC	0.3
	b. PP - smallest area	0.2
	c. PP - fastest	0.2
	d. PP – fastest critical path	0.1
	e. PP – fastest first task	0.2
2	a. Smallest area	0.1
	b. Fastest	0.1
	c. Critical path	0.1
	d. Minimizing transmission	0.2
	e. Next	0.2
	f. minimal	0.3
3	LLF scheduling	1
4	a. Min-cut – comm. size	0.3
	b. Min-cut – transmissions	0.3
	c. Min-cut – crit. path	0.2
	d. no-cut	0.2

Column the probability represents the probability of choosing a particular gene. The choice of gene sharing and gene mapping decide: optimization of speed, optimization of NoC traffic or network energy optimization NoC. The highest probability of being selected is the gene responsible for the optimization of NoC traffic. Because the transmission of the longest execution time can cause collisions. In proposing the division of the graph by cutting at the site of the longest transmission, which disables the selected transmission in the region considered NoC network. The inclusion of the longest transmission network in a region NoC offers greater possibilities of other transmission scheduling.

The system can distinguish two types of genes: Genes division graph and mapping genes. The first type of genes responsible for the division of the graph, or its part into smaller fragments. The second type of gene is responsible for the mapping portion of the graph (which is the result of the division by the gene / genes share) in the region NoC network. The system is constructed by the performance at the level of the node in the tree. The tree is the genotype of specific genes at different levels, depending on the location of the node in the tree algorithm is performed assigned to the corresponding gene. If the node is a leaf genotype is performed gene mapping that performs the mapping in question the graph in the network NoC. Otherwise, if the node is not a leaf is made of the graph division / section of the graph according to the rules of a particular gene. Each gene sharing and gene mapping ensures that the time constraints imposed on individual tracks will be met. Compliance with time limits and, as proposed here, the algorithm Dynamic Time Restriction.

During mapping genotype to phenotype each functions take into consideration constraints. If the selected allocation/assignment causes constraint violation then the next matching design decision will be chosen. To avoid infinite growth of genotypes during evolution, unused genes are removed from genotypes. Fig. 9a presents a sample genotype for the task graph from Fig.1. Assume that the deadline for task T_5 is defined as c_5=620, then the system is constructed as follows. Final system is given on Fig. 5b.

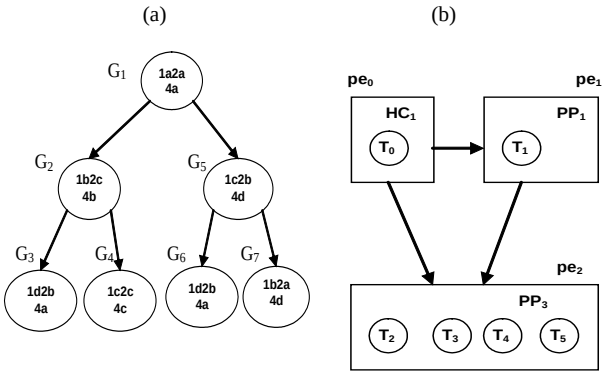

Figure 8. Sample genotype (a) and the corresponding phenotype (b)

5.5 Evolution

On the evolution of the use of genetic operators: crossover, reproduction and mutation generates a new generation of solutions. The number of solutions in each population is always equal to *population_size*. After the genetic operations are performed on the current population, the new population replaces the current one. The evolution is controlled by parameters reproduction_rate, crossover_rate and mutation_rate, as follows:

- *reproduction_rate*population_size* is the number of solutions created using reproduction,
- *crossover_rate*population_size* is the number of solutions created using crossover,
- *mutation_rate *population_size* is the number of solutions created using mutation,
- *reproduction_rate+crossover_rate+mutation_rate*=1, this condition ensures that each population will have the same number of individuals.

All solutions are ranked by the cost defined in previous chapters, which defines the fitness function. Solutions are selected randomly, but with different probability, which is defined according to the position in the ranking *r* as:

$$P = \frac{population\_size - r}{population\_size}$$

Thus the best solutions are preferred for reproduction, but worse solutions are not necessarily passed over.

Crossover randomly selects two solutions, then a single point of intersection is selected at random for both parents. All data beyond this point are swapped between parents and the resulting solutions are children added to the new population. Figure 9 shows an example of crossing two solutions. The dotted line is a cutting point. Mutation randomly selects one solution, then randomly selects a node and replaces it by generating new options. The algorithm stops if the best found solution does not improve in the last ε steps. The best solution obtained during the run is the result of the algorithm.

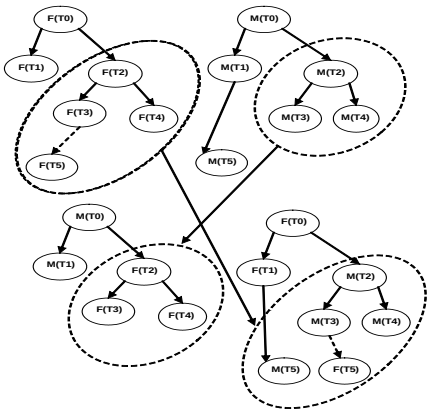

Figure 9 Crossover of genotypes

6. Experimental results and discussion

The effectiveness of the proposed method was tested after a certain amount of testing. First, systems represented with random generated task graphs were synthesized using DGP, Yen-Wolf [2] and EWA [3] methods. The results of conducted experiments are given in Table 3. The following columns contain: number of tasks, constraint for the whole graph, and synthesis results. The target system architecture cost and for each method the time required for execution of all tasks are given. In all cases four types of PE were available in the resource library. The Developmental Genetic Programming method found comparable or better solution for all systems. We assumed the following parameters: α=10, reproduction_rate=0.2, crossover_rate=0.7, mutation_rate=0.1 and ε=5. In the worst case (system with 90 tasks), the algorithm stopped after 150 seconds, and 53 generations were evaluated. Figure 10 presents the comparison of the optimization flow for all methods mentioned above. Graph with 50 tasks was synthesized with constraint Tmax=2200. The Y (vertical) axis represents the cost of a solution, found in the following optimizations steps (generations). For DGP approach the X (horizontal) axis represents the population number, while for EWA and Yen-Wolf it represents the following refinement steps. The Developmental Genetic Programming method was run 3 times, with populations equal to 10000, 20000 and 40000 individuals (α*50*4). In this methods solutions with costs: 4042, 4024, 3919 were found, while using iterative improvement methods the best solution found had a cost equal to 4151.

Table 3. Experimental results obtained using Yen-Wolf, EWA and DGP cosynthesis methods: N-number of tasks in a task graph, T_{max} – deadline, Time - time of execution of all tasks, Cost – area of the target system

N	T_{max}	Yen-Wolf		EWA		**DGP**	
		Time	Cost	Time	Cost	**Time**	**Cost**
10	400	315	1573	287	1517	**395**	**1545**
30	800	773	3441	796	3600	**792**	**3167**
50	1200	1171	6182	1179	5606	**1142**	**5518**
70	1600	1548	6859	1563	6650	**1599**	**5947**
90	2000	1917	13184	1996	7873	**1991**	**7411**

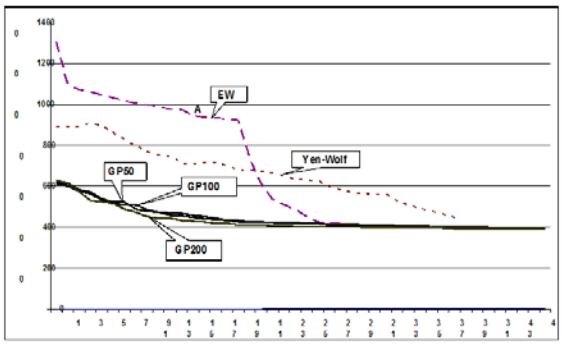

Figure 10. Optimization flows for different co-synthesis methods. GP50, GP100 and GP200 represent the DGP method with α=50, α=100 and α=200, respectively.

For the experimental purposes we applied the co-synthesis algorithm (Deniziak et al. 2008a) to 10 synthetic task graphs (application models) using our tool similar to well-known TGFF (Dick et al. 1998). Every graph has 20-30 tasks, 15-30 messages and 5-9 allocated PEs. For the experimental purposes we applied the co-synthesis algorithm (Deniziak *et al.* 2008a) to 10 synthetic task graphs (application models) using our tool similar to well-known TGFF (Dick *et al.* 1998). Every graph has 20-30 tasks, 15-30 messages and 5-9 allocated PEs. We have also calculated *calculation/message time* and *collision/message time* factors. The first one ranges from 0,36 (communication-oriented system) to 1,83 (calculation-oriented system with less probability of message contention). The second factor we interpret as the probability indicator of the message conflict. It ranges from 0,25 (little probability of contention) to 0,64 (high probability of contention). Next, via mathematical analysis, we compared our custom network topology ("Custom") against randomly generated minimal rectangular mesh ("NxM"), random minimal regular mesh ("NxN"), and against custom topology obtained by applying

methodology similar to bandwidth-based approach (Neeb et al. 2008)(Murali *et al.* 2006) ("Bndwdth"). In particular, we transformed generated ATGs into Core Graphs, next we generated direct links for cores exchanging the largest amounts of data and dedicated links to remove possible contention. Such approach reminds also the one presented in (Leary *et al.* 2009), but is far chip resource conserving – no routers without PE are allowed. It is achieved at the price of possible longer transmission paths. To make comparison with our method fair, we followed the rule "one router - one PE" and no more than 4 bidirectional router ports. Moreover – every link in the topology has the same bandwidth. Meshes use XY routing and are made contention-free by rescheduling. For our topology and bandwidth-based one, performance is the same (*i.e.* execution time of the system), as both solutions are contentionless.

Table 4. Link usage deterioration

App. Name	calc./ msg.	coll./msg.	Custom [%]	NxM mesh [%]	NxN mesh [%]
G01	0,54	0,35	7,08	11,39	11,39
G02	0,69	0,64	10,77	37,25	18,84
G03	0,36	0,48	7,79	15,09	12,25
G04	0,66	0,3	8,66	12,52	19,14
G05	1,32	0,41	9,86	19,8	14,44
G06	1	0,4	5,02	9,55	7,04
G07	0,58	0,5	0	8,34	9,17
G08	1,22	0,4	0	7	7
G09	1,83	0,25	0	0	3,24
G10	1,06	0,52	10,9	14,2	16,18

The information about graphs is showed in Table 4 (columns 2 and 3 contain previously mentioned factors) and performance (i.e system latency) loss compared to system just after co-synthesis (with path-based contention ignored). In every case superiority of topology generated by our methodology over meshes is demonstrated. In four cases our approach gave result equal to the fastest system (performance loss "0" means no additional latency). This means that it is not necessary to prevent network contention because there is no transmission scheduling required. Performance (latency) loss results.

The results for topology generation contains in table 5. In this paper unidirectional links used in every topology are counted. Its a coarse factor of chip resources and energy requirements. Proposed approach if far better than mesh approaches - minimum link saving is over 39%. Also bandwidth-based topology demands more resources than presented in this paper. There is only one case – G03 - where we noticed deterioration of our solution.

Table 5. Performance (latency) loss results

App. Name	*Custom [links]*	*NxM mesh link loss [%]*	*NxN mesh link loss [%]*	*Bndwdth link loss [%]*
G01	9	33,71	60,15	16,17
G02	9	52	61,52	31,17
G03	9	33,17	63,35	-14,5
G04	6	54,12	72	43,45
G05	10	46,5	55,33	27,57
G06	9	33,17	60,35	9
G07	12	37,2	48	0
G08	13	41,38	42,23	26,78
G09	8	39,68	66,67	26,27
G10	13	31	43,33	12,33

7. Conclusion

A novel approach to hardware/software codesign was presented in this paper. The system is generated using developmental genetic programming method during using design procedure. To best knowledge it is the first DGP method dealing with the codesign problem. Preliminary experimental results show that the method is very promising. While there is still a large space for improvements, not examined yet, for all benchmarks we received results comparable or better than using other co-synthesis methods. Future research will concentrate on further improvements of the presented approach. Especially, we will examine alternative implementations of genetic operators, other structures of the system construction tree, other system-construction functions, etc. In future work is planned to achieve better results in a reasonable time for the final DGP co-synthesis method.

In this paper is proposed a novel methodology for application-specific and contention-free NoC generation. The goals are achieved through dedicated links generation (topology part) and temporal ordering of the tasks/messages (schedule part). The presented smethod, is suitable for distributed embedded systems, i.e. systems with predictable pattern of communication. As demonstrated through experiments, our approach is far better than typical, random mesh networks.

References

[1] Wolf W. High-Performance Embedded Computing: Architectures, Applications, and Methodologies, Morgan Kaufman, St. Louis, USA 2006.

[2] Gupta R.J., De Micheli G., Hardware-Software Co-synthesis for Digital Systems, IEEE Design & Test, Vol 10, No. 3, 29-41, September 1993.

[3] Deniziak,S.: Cost-efficient synthesis of multiprocessor heterogeneous systems. Control and Cybernetics, Vol.33, 2, 341-355, 2004.

[4] Yen T.-Y., Wolf W.H.: Sensivity-Driven Co-Synthesis of Distributed Embedded Systems. Proc. of the Int. Symposium on System Synthesis, 4-9 , 1995.

[5] Hu J., Marculescu R., Communication and task scheduling of application-specific Networks-on-Chip, IEEE Proceedings of Euromicro Symposium on Digital Techniques, 643-651 , 2005.
[6] Julien Delorme, Dominique Houzet, A complete 4G radiocommunication applicationmapping onto a 2D mesh NoC architecture, INSA/ IETR Laboratory, 20 avenue des Buttes de Coesmes 35043 Rennes Cedex, France, 93-96 ,2006.
[7] Ghosh Pavel and Arunabha Sen, Energy Efficient Application Mapping to NoC Processing Elements Operating at Multiple Voltage Levels, Department of Computer Science and Engineering Arizona State University, Tempe, AZ Alexander Hall Department of EECS UC Berkeley, CA, USA, Networks-on-Chip, 2009. NoCS 2009. 3rd ACM/IEEE International Symposium, 80-85 ,2009.
[8] Lei Tang, Hong Peng, Yang Yanhui, Mapping Concurrent Applications to a NoC, Datang Microelectronics Technology Co., LTD., Beijing, China, Solid-State and Integrated Circuits Technology, 2004. Proceedings. 7th International Conference, 1960-1963 vol.3, 2004.
[9] Deniziak S, Górski A. Co-synthesis of SoC systems using Developmental Genetic Programming, Cracow University of Technology Publishing House, Cracow, 2008 (in polish)
[10] Dve B.P., Lakshinarayana G., Jha N.K., COSYN: Hardware-Software Co-Synthesis of Embedded Systems, Proc. Of the Design Automation Conference, Anaheim, USA, 703-708, 1997.
[11] Dick R. P., Jh N.K., MOGAC: A multiobjective Genetic Algorithm for the Co-Synthesis of Hardware-Software Embedded Systems, IEEE Trans. On CAD, Vol. 17, No. 10, Piscataway, NJ, USA 1998
[12] Koza J. R., Genetic Programing: On the Programming Computers by Means of Natural Selection, MIT Press, Cambridge, MA, USA 1992.
[13] Koza J. R., Genetic Programing: On the Programming Computers by Means of Natural Selection, MIT Press, Cambridge, Sokenou D., Gnerating Test Sequences from UML sequence Diagrams and State Diagrams, MA, USA 1992.
[14] Keller R. E., Banzhaf W., The evolution of genetic code in genetic programming, Proc. of the Genetic and Evolutionary Computation Conference, Orlando, USA, 1077-1082, 1999,
[15] Deniziak, S., Górski, A. (2008a). „Hardware/software co-synthesis of distributed embedded systems using genetic programming", Lecture Notes in Computer Science, Springer, 5216, 83-93, 2008.
[16] Deniziak, S., Tomaszewski, R. (2008b). "Adaptive routing protocols validation in NoC systems via rapid prototyping", Proceedings of the IEEE Human System Interaction, 115-120, 2008.
[17] Sethuraman, B., Bhattacharya, P., Khan, J., Vemuri, R. "LiPaR: A light-weight parallel router for FPGA-based Networks-on-Chip", In 15th Great Lakes Symposium on VLSI, 452–457, 2005.
[18] Wang, D., Matsutani, H., Amano, H., Koibuchi, M., "A link removal methodology for Networks-on-Chip on reconfigurable systems", International Conference on Field Programmable Logic and Application, 269-274, 2008.
[19] Hu, J., Marculescu, R.,"Communication and task scheduling of application-specific Networks-on-Chip", IEEE Proceedings of Computers and Digital Techniques, 643-651, 2005.

Dariusz Dorota. Research and teaching assistant of Laboratory of Computing Science, Faculty of Electrical and Computer Engineering, Cracow University of Technology. He received the M. Sc. degree in Applied Computer Science from AGH University of Science and Technology in Cracow, Poland, in 2010. His research interest includes embedded system co-synthesis, NoC-based system design.