Collaborative Hardware/Software Partition of Coarse-Grained Reconfigurable System Using Evolutionary Ant Colony Optimization

Dawei Wang
College of Comuter Science
University of Defense Technology
Changsha, P.R.China 410073
Tel : +86-731-4575981
Fax : +86-731-4575981
e-mail : daweiwang@nudt.edu.cn

Sikun Li
College of Comuter Science
University of Defense Technology
Changsha, P.R.China 410073
Tel : +86-731-4575981
Fax : +86-731-4575981
e-mail : lisikun@263.net.cn

Yong Dou
College of Comuter Science
University of Defense Technology
Changsha, P.R.China 410073
Tel : +86-731-4573647
Fax : +86-731-4573647
e-mail : yongdou@163.net

Abstract - The flexibility, performance and cost effectiveness of reconfigurable architectures have lead to its widespread use for embedded applications. Coarse-grained reconfigurable system design is very complex for multi-fields experts to collaborate on application algorithm design, hardware/software co-design and system decision. However, existing reconfigurable system design methods and environments can only support hardware/software co-design, ignoring the collaboration between multi-field experts. This paper presents a collaborative partition approach of coarse-grained reconfigurable system design using evolutionary ant colony optimization. We create a distributed collaborative design environment for system decision engineers, software designers, hardware designers and application algorithm developers. The method not only utilizes the advantages of ant colony optimization for searching global optimal solutions, but also provides a framework for multi-field experts to work collaboratively. Experimental results show that the method improves the quality and speed of hardware/software partition for coarse-grained reconfigurable system design.

I Introduction

Existing EDA (Electric Design Automation) systems support single designer to design with human-computer interaction only. With the emergence and rapid growth of CSCW (Computer Supported Collaborative Work) [1] [2] [3], EDA is not only the tool for functional computing, design decision and performance analysis, but also for multi-field experts to communicate and collaborate with human-human interaction.

Reconfigurable SoC design is very complex for multi-field experts to collaborate on application algorithm design, hardware/software co-design and system decision, while hardware/software partition is the key problem of reconfigurable system design. Since Gupta have advanced hardware/software partition algorithm for automating design space exploration [4], most research efforts use heuristic algorithm for hardware/software partition, such as ant colony optimization proposed by G.Wang et al. [5]. They show that intelligent ants can cooperatively search for global best optimal solutions using both heuristic information and pheromone information. However, these methods are well for hardware/software co-design, while not for distributed collaborative design with multi-field experts.

In this paper, we present a Distributed Collaborative Partition (DisCoPar) approach of coarse-grained reconfigurable system using evolutionary ant colony optimization. Based on the previous work on distributed cooperative design of embedded systems, we developed DisCoPar Environment, in which we build a distributed collaborative framework for multi-field experts [6]. In the framework we use evolutionary ant colony optimization to search for global best optimal solutions. Experimental results show improvement on design efficiency and quality.

II. Related Work

CSCW design is a novel product design method, which supports product designers and related experts at different locations to design the product by use of network and various computer aided tools. During this process, each user is aware of the existence of other users, and interacts with them.

Cutkosky [2] [3] advanced the concept of distributed cooperative design. Since then, researchers apply many techniques to implement distributed cooperative design, such as network and communication, distributed computing, computer supported cooperative design, agent and Web service. Currently, the combination of intelligent agent and Web service techniques has gained better application effect, and it supports distributed cooperative design of complex products efficiently.

For collaborative partition of coarse-grained reconfigurable system, we in general assume that the styles of collaboration between multi-field experts are synchronous and remote. Experts include algorithm and reconfigurable system designer. They work on algorithm design, hardware/software design and system decision making respectively.

III. Problem Formulation for Coarse-Grained Reconfigurable System Partition

A. *Problem Definition*

Usually, we use TG (Task Graph) to describe the behaviors of SoC system, rAG (reconfigurable Architecture Graph) to describe the architectures of reconfigurable SoC system, and the mapping from TG to rAG to describe hardware/software partitioning of reconfigurable system [7].

Definition 1 (TG) A task graph TG = (T, E, C, P) consists of a set of nodes T, a set of directed edges $E \subseteq (T \times T)$, configuration of nodes C, and a set of node ports P.

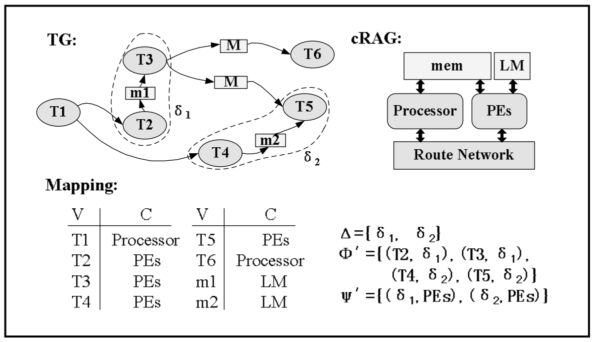
The nodes T present the tasks of SoC system, and the edges E present control/data flow of tasks' communication.

Definition 2 (rAG) A reconfigurable architecture graph rAG = (C^P, C^{PE}, C^M, R^N) consists of microprocessor computation resources C^P, reconfigurable computation resources C^{PE}, memories C^M, and route networks R^N.

Definition 3 (k way partition) Given a set of modules $M = \{m_1, m_2 \dots m_n\}$, a k way partition problem is to find a set of clusters P = $\{p_1, p_2, \dots, p_k\}$, which meets:

$$\begin{cases} p_i \subseteq M, & 1 \le i \le k \\ \bigcup_{i=1}^{k} p_i = M & \\ p_i \bigcap p_j = \Phi, & 1 \le i, j \le k, i \ne j \end{cases}$$

In the problem of reconfigurable SoC hardware/software partitioning, M presents TG, and P presents rAG. Generally, the performance objects of hardware/software partitioning are various due to the complexity of application field. In this paper, we assume the performance object is to minimize the weighted cost sum of running time and area of all the tasks. A typical partitioning for coarse-grained reconfigurable SoC is shown in figure 1.



V	C	V	C
T1	Processor	T5	PEs
T2	PEs	T6	Processor
T3	PEs	m1	LM
T4	PEs	m2	LM

$\Delta = \{\delta_1, \delta_2\}$
$\Phi' = \{(T2, \delta_1), (T3, \delta_1), (T4, \delta_2), (T5, \delta_2)\}$
$\psi' = \{(\delta_1, PEs), (\delta_2, PEs)\}$

Fig. 1. Problem formulation for reconfigurable SoC partition.

B. *Target Architecture for Coarse-Grained Reconfigurable System Partition*

We deal with reconfigurable SoC hardware/software partition by two main steps: mapping tasks to microprocessor or reconfigurable array, and partial partitioning for the tasks mapping on reconfigurable arrays. To meet the needs of various applications, many reconfigurable architectures have been proposed, which have different details of design implementation, such as the topology of reconfigurable array, communication protocol, and the strategy of reconfiguration. To facilitate the research of reconfigurable partition problem, we abstract reconfigurable architecture and take it as the target architecture for partitioning. We design a RAAM (Reconfigurable Architecture Abstract Model) for the hardware/software partitioning of coarse-grained reconfigurable Systems [8]. RAAM has many configurable parameters, so it supports most kinds of reconfigurable architectures.

Figure 2(a) shows the task graph that describes the function and behavior of reconfigurable systems. There are three types of task nodes: computing task, storage segment and communication task. The task nodes are connected by dependencies or channels. Figure 2(b) shows the reconfigurable architecture that consists of microprocessor and reconfigurable array. The microprocessor uses 32-bit RISC instruction set and some enhanced instructions for special propose processing. Reconfigurable array consists of PEs (Process Elements), route network, LM (Local Memory), configurable logic and control logic, etc. PEs array based reconfigurable architectures are popular because of their advantage of high performance, low power and good flexibility for embedded system applications. Multi-field experts use distributed collaborative environment to configure RAAM and map task graph to reconfigurable architecture.

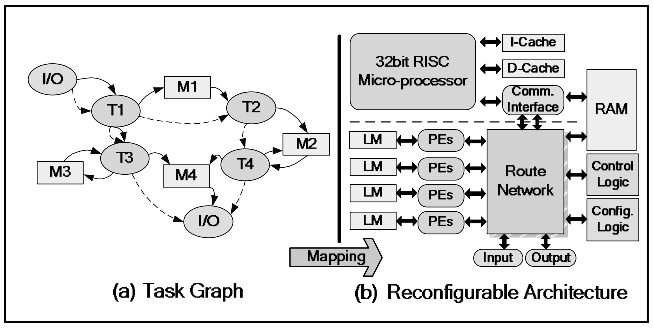


Fig. 2. Reconfigurable architecture abstract model for hardware/software partitioning, (a) task graph describes the function and behavior of reconfigurable system; (b) reconfigurable architecture consists mainly of microprocessor and coarse-grained reconfigurable PEs array.

IV. Distributed Collaborative Partition

A. *Collaborative Partition Framework*

We use a Client/Server structure for distributed collaborative partition, called DisCoParFrame. It supports data share, communication among group and the control of parallel operation. The client consists of some semi-intelligent agents for hardware/software designer, application algorithm designer to collaboratively partition, as shown in Figure 3. It supports state watching, action reacting and RAAM configuring. The server supports system decision maker to control the course of partition. It sees after global information storage, communication transfer, harmony and decision.

In DisCoParFrame, system decision maker on server informs the experts on clients to configure the parameters of RAAM. Then every expert submits the configuration results of RAAM to server by co-access agent. After receiving all the configuration results, the server starts ant colony optimization algorithm and simulation to do partition. The experts adjust the configuration according to the feedback of runtime simulation information from server until the objective is met. The experts can stop simulation if they find some constraints are not met in the process of simulation.

In CSCW system, experts want to know about not only the results of collaborative operation, but also the whole course of collaborative operation. They should know about the activity of expert, the change of state and the log of operation. Experts can adjust their behavior according to current actions and states of collaborative environment. So it is convenient for experts to collaborate with each other.

In DisCoParFrame, we define the communication protocol of multi-field experts, which includes:

- Task protocols, such as query of current task, requisition for operation of task, user register, user login and logout.
- Data transfer protocol, including transfer mechanism of the configuration of RAAM parameters, the operations of experts and the state of simulation.
- Parallel and cooperation control protocol, which maintain the consistency of global share data.
- Notification protocol, including aware mechanism of the state of tasks and the chatting among experts.

Conflicts are inevitable for DisCoParFrame supports multi-experts access global share data. To avoid the conflicts, we designed a logic clock based concentrative parallel control method. The basic idea is that: according to the global logic clock in server and the global exclusive integer allocated to each client, we define the order of operation events that send to the server. So it can ensure the consistency of the order of operation events running in client.

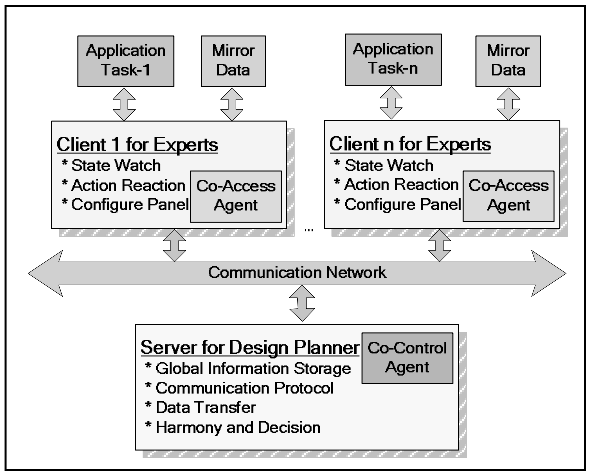


Fig. 3. Distributed collaborative partition framework for coarse-grained reconfigurable system design.

B. eACOGA Algorithm

ACO was proposed firstly by M.Dorigo in 1991. Then he expounded the basic principle and maths models of ACO in 1996 [14]. He made some simulation experiments to compare ACO with other algorithms, such as genetic algorithm, tabu search and simulated annealing. Because established sound basis for follow-up research of ACO, M.Dorigo gained the prize of Madame Curie Outstanding Achievements.

Researches show that the ants with poor vision could find the shortest route from ant colony to food by some volatility secretions namely pheromone. The subsequent ants select this route according to the strength of pheromone. When more ants pass through the route, much stronger the pheromone of the route becomes. Thus more ants are attracted to this route, forming a kind of positive feedback. Existing researches on ACO show well effects on many optimization problems, such as traveling salesman problem, network route, and job shop scheduling.ACO algorithm can find better solutions of partitioning more effectively [9]. But the strategy of random selection in constructing solutions leads to slow convergence speed. Furthermore, the principle of positive feedback can not only strengthen the solutions with better performance, but also bring on the stagnancy of search. The causation is that the main configurable parameters of the algorithm, such as α, β, ρ, Q, are set to fixed value when initializing, and it has no adaptability to various applications.

We present an eACOGA approach of hardware/software partition for reconfigurable SoC. GA can evolve the configuration parameters of ACO algorithm by cross operation and variation operation. So eACOGA can evolve and optimize itself to search global optimal solutions.

We define the rules of eACOGA as follows:

Objective Function and Fitness Function: We define objective function as S_{best} = arg min C_p, fitness function as Fitness (p) = $1/C_p$. Where, C_p figures the cost function of partition p.

Configure RAAM: According to the need of specific applications, design experts configure the task graph and reconfigurable architecture. In this step, the parameters of the tasks and architectures should be decided.

Strategy of Render to DAG: For any nodes except t_n, ants try to render the color of t_j, the subsequence of t_i. Ants achieve the work according to the global heuristic information ($\tau_{ij}(k)$) of edge e_{ij} and the local heuristic information ($\eta_j(k)$) of node t_j. The ants on node t_i will render the color of node t_j as c_k at the probability of:

$$p_{ij}(k) = \frac{\tau_{ij}(k)^{\alpha}\eta_j(k)^{\beta}}{\sum_{l=1,2}\tau_{ij}(l)^{\alpha}\eta_j(l)^{\beta}}$$

Where, $\tau_{ij}(k)$ is the pheromone on edge e_{ij}, α and β is the factor of them and $\eta_j(k)$ is defined as follows:

$$\eta_j(k) = 1/((w_t * time_t(k)) + (w_a * area_j(k)))$$

Use Genetic Algorithm to Evolve Parameters: We use genetic algorithm to evolve the parameters of ant colony optimization, such as α, β, ρ, Q. First, Configure the probability factor of cross and variation operation according to the size of population and the generation of evolution. The evolution rules of GA use proportional selection, single cross and even variation. Then by taking α, β, ρ, Q as the variable of fitness function, the best optimal partition cost as fitness function and the course of ACO as the individual, we optimize α, β, ρ, Q repeatedly until finding the best optimal solutions.

Pheromone Setting and Refreshing: We adopt MMAS (Max-Min Ant System) introduced by Thomas Stuzle ([10]), the refreshing equation of pheromone is:

$$\tau_{ij}(k) = \begin{cases} (1-\rho)*\tau_{ij}(k) + \Delta\tau_{ij}(k)_{best} \\ \tau_{ij}(k)_{\max}, \text{if } \tau_{ij}(k) > \tau_{ij}(k)_{\max} \\ \tau_{ij}(k)_{\min}, \text{if } \tau_{ij}(k) < \tau_{ij}(k)_{\min} \end{cases}$$

Where, ρ is the evaporation ratio of pheromone, $\tau_{ij}(k)_{max}$ ($\tau_{ij}(k)_{min}$) is maximum (minimum) strength of c_k pheromone on edge e_{ij}, and $\Delta\tau_{ij}(k)_{best}$ is increment of c_k pheromone on edge e_{ij} done by the "best ant" in current ant system algorithm iteration. According to Ant-Cycle Model, $\Delta\tau_{ij}(k)_{best}$ is defined as:

$$\Delta\tau_{ij}(k)_{best} = \begin{cases} Q/C_{p_{best}}, \text{in } p_{best}\ e_{ij} \textit{ is rendered by } c_k \\ 0, \quad \text{otherwise} \end{cases}$$

The pseudo-code for eACOGA algorithm is shown in Algorithm 1. First, the configurations of RAAM and DAGs are input, and after the execution of eACOGA algorithm the best optimal solutions for partition are output. In eACOGA, ACO is built as a class, which has two main functions: GetAnt() and StartSearch(), as shown in Line 23-25. GA randomly encodes the variables of α, β, ρ, Q, as shown in Line 18-22. GA evolves the variables continuously by the operation of select, crossover and mutate, as shown in Line 6-14. In ACO, we put the ants randomly into DAGs and begin the search for best optimal solutions, as shown in the function on Line 28-36 and Line 37-47. After evaluating the fitness function values we output the best optimal solutions, as shown on Line 12 and Line 15.

Algorithm 1. Pseudo-code for eACOGA algorithm

```
//Input: the configuration of RAAM and DAGs
//Outputs: the optimal solutions for partition
//Q, α, β, ρ is the main parameters of ACO
1    Main(){
2        Generation = 0;
3        Initialize();
4        Evaluate();
5        Keep_the_Best();
6        foreach generation
7        {
8            select();
9            crossover();
10          mutate();
11          report();
12          Evaluate();
13          elitist();
14       }
15       Output the best fitness values;
16   }

17   Evaluate(){
18   For (mem = 0; mem < POPSIZE; mem++)
19   {
20      For (i = 0; i < NVARS; i++)
21          x[i+1] = population[mem].gene[i];
22      Q = x[1], α = x[2], β = x[4], ρ = x[4];
23      ACO partition = new ACO;
24      partition.GetAnt();
25      partition.StartSearch();
26      population[mem].fit = CostFunction_to_Fit;
27   }

28   ACO::GetAnt(){
30      Randomly put ant into DAGs;
31      for (i = 0; i < nAntCount; i++)
32      {
33         task = rnd( nTaskCount);
34         ants[i].AddTaskIntoTabu(task);
35      }
36   }

37   ACO::StartSearch(){
38      Foreach ant
39      {
40         Select_NextNode_Accordto_heuristic_Inf();
41         Moveto_NextNode();
42         Update_Tabu_Table();
43         find out the best solution of the step and put
it into temp;
44      }
45      Update_Trail();
46      Find_theBest_Solutions_Of_Partition();
47   }
```

C. *Automatic Partitioning Flow*

We have designed an automatic partitioning flow for mapping applications on reconfigurable SoC, as shown in Figure 4.

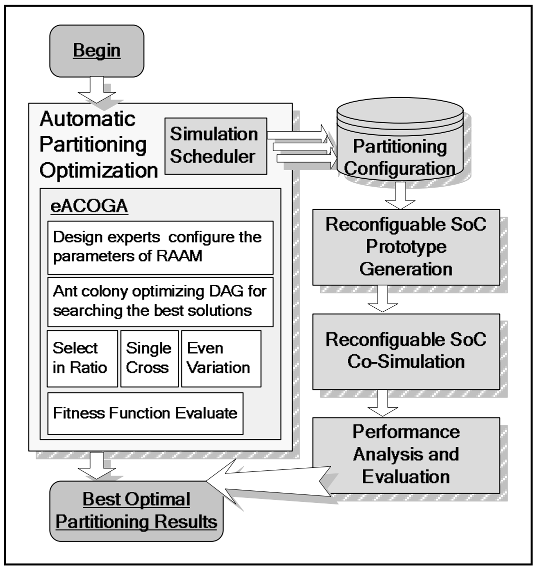

Begin
Automatic Partitioning Optimization
Simulation Scheduler
Partitioning Configuration
eACOGA
Design experts configure the parameters of RAAM
Ant colony optimizing DAG for searching the best solutions
Select in Ratio
Single Cross
Even Variation
Fitness Function Evaluate
Reconfiguable SoC Prototype Generation
Reconfiguable SoC Co-Simulation
Performance Analysis and Evaluation
Best Optimal Partitioning Results

Fig. 4. Automatic partitioning flow of reconfigurable SoC. It integrates eACOGA for partitioning configuration.

First, design experts configure the tasks and reconfigurable architecture of RAAM. Then, application specific reconfigurable SoC prototype is generated according to existing reconfigurable architecture templates [11]. Finally, we run reconfigurable SoC transaction level co-simulation to adjust and output the best optimal partitioning solutions.

The automatic partitioning flow has two main advantages:

(1).For each individual of genetic population in eACOGA, the flow of partition and reconfigurable SoC co-simulation can run automatically. When some constraints cannot be met, experts can request to stop the simulation.

(2).Transaction level simulation in SystemC can describe various behaviors of reconfigurable SoC with faster speed and nicer accuracy. Architecture template enhances reuse of existing SoC design and achieves exploration speedup well.

IV. Experimental Results

A. *Target Architecture and Benchmarks*

We use eACOGA algorithm for the partitioning of a coarse-grained reconfigurable SoC system, which consists mainly of 32-bit RISC microprocessor called Estar and reconfigurable arrays called LEAP [12]. Both Estar and LEAP are developed by our research group. We use Estar for common computing. It has 8KB instruction cache and 8KB data cache, 266M Hz and 220mW of CPU core. Besides, we use LEAP for reconfigurable computing. It can accelerate applications through loop self-pipelining technique. LEAP steps loop iteration automatically and it has the ability to exploit parallelism at loop level, instruction level, and task level.

The SoC system integrates some typical algorithms in the field of Software Defined Radio, Synthetic Aperture Radar imaging and high precision digital image encode/decode. We have designed some typical algorithms running on reconfigurable systems, such as FFT (Fast Fourier Transformation), Sobel Edge Detection, Median Filter, Matrix Multiply, FDCT (Forward Discrete Cosine Transform), IDCT (Inverse Discrete Cosine Transform), etc. We have tested the performance of these typical algorithms on Estar and LEAP and translate execution time into time cost and resources used into area cost.

B. Result Analysis

We generate some test DAGs based on SUIF [13], the nodes of which consist of typical algorithms and basic blocks. We set the parameters region of α, β, ρ, Q respectively as [5, 1], [5, 1], [0.8, 0.2], [100, 40]. In eACOGA algorithm, we set $w_t = 1$, $w_a = 10$, ACO iteration counts = 100, GA population size = 5, GA max generation = 50, GA cross probability factor = 0.8, and GA variation probability factor = 0.15.

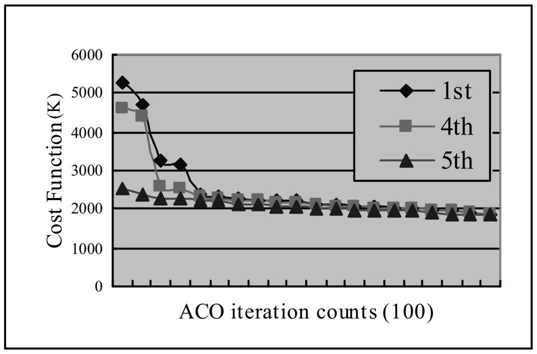


Fig. 5. The evolution curve of eACOGA for partition. The convergence speed of 5th generation ACO is much faster than that of 1st and 4th one.

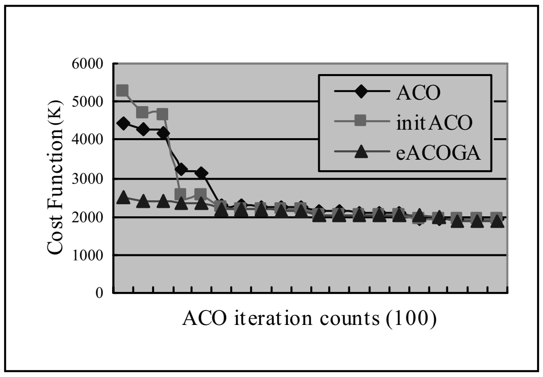


Fig. 6. Comparing eACOGA with ACO and initACO. It shows eACOGA can find optimal solutions with fewer iteration counts.

We achieve the object of hardware/software partitioning using eACOGA, as shown in Figure 5. The total number of tasks is 10, and ant counts = 6. The global best optimal solutions are 1862843, in which Q = 70.540, α = 4.836, β = 3.580, ρ = 0.738. The average value of 1st generation is 1918421, by evolution the average value of 4th generation is 1891026, and that of 5th generation is 1890922.

To compare the quality of eACOGA with that of other researches, we select ACO algorithm in literature [8]. We set ACO parameters as: Q = 1000, α = 1, β = 1 and ρ = 0.8. Figure 6 show that eACOGA has better ability than ACO in searching for the global best optimal solutions. The algorithm of ACO gets into local best optimal solutions (1905396). However, eACOGA can find the global best optimal solutions (1862843) effectively for the advantage of self-adaptive optimization. Besides, another algorithm we have researched, called initACO (ACO with init pheromone), has the performance between them [14].

The possible explanation for the proposed eACOGA approach to outperform the basic ACO method with better optimal solutions is that the main control parameters of ACO affect its performance greatly. The pheromone ($\tau_{ij}(k)$) is the carriers of the past information, while the heuristic function ($\eta_{ij}(k)$) is the carriers of the future information. Many experiments on basic ACO shows that the factor α which controls $\tau_{ij}(k)$ has important impact on ACO performance and the factor β which controls $\eta_{ij}(k)$ has a substantial effect on global convergence. The factor ρ which reflects the change of pheromone affects the ability of global search and the speed of convergence. The factor Q which reflects the amount that the searching ants release, concerns the positive feedback ability of the searching ants and the rapid convergence of ACO.

The basic ACO algorithm initializes these factor parameters with fixed values, as shown in literature [5], which set the factor parameters in all the experiments with the same values: Q=1.000, α = β = 1, ρ = 0.8. While different DAGs needs a different combination of the factor parameters. So the information motivates us to a hybrid approach of ACO and GA together. That is we use GA evolve and select the suitable factor parameters of ACO for different DAGs. Our experiments show that eACOGA has statistically robust in finding close to optimal solutions.

The algorithm of eACOGA has some configurable parameters. We can optimize the performance of eACOGA by configuring the parameters and running simulation. Table I shows that when cross factor = 0.8, eACOGA has better ability for random search and can reduce iteration counts. In the same way we make experiments on three group probability factor of variation (0.15, 0.20 and 0.25). The results show that 0.15 is the best value for our problem.

TABLE I
Average iteration counts with different cross factor

Scale of	Single Cross Factor		
Tasks	0.6	0.7	0.8
8	9	11	8
16	15	11	13
32	28	26	25
64	53	51	46
128	89	83	85

IV. Conclusions

Existing EDA systems support human-computer interaction only, not touch on human-human interaction. This paper proposes a collaborative hardware/software partition approach of coarse-grained reconfigurable system, which supports both human-computer and human-human interaction well. We design distributed collaborative partition framework and integrate automatic collaborative partition flow for hardware/software partition of reconfigurable SoC. By configuring parameters and running simulation we can get the global best optimal partition solutions. It can not only reduce the time of waiting for simulation, but also provide convenient collaborative framework for multi-field experts to work.

ACO has been well proved to be suitable for fine-grained reconfigurable SoC partitioning, but not mentioned for coarse-grained reconfigurable SoC. This paper uses eACOGA for hardware/software partition of coarse-grained reconfigurable system, which achieves better results than ACO. The remarkable advantage of eACOGA is that it combines both global and local heuristics in the search of exploration space. And it can evolve the main control parameters of ACO by GA to tune for different applications requirement.

The algorithm of eACOGA can evolve the main control parameters (α, β, ρ, Q) of ACO, so that it can find global best optimal solutions efficiently and rapidly. The suitable combination of the control parameters can also achieve rapid speed of convergence. It overcomes the disadvantage of ACO that be inclined to get into local best optimized solutions. Furthermore, the method combining ACO and GA that yields even better results than using each of the algorithms individually.

Acknowledgements

This work is sponsored by the National Science Foundation of China under the grant NO.90707003 and NO.60633050.

References

[1] Reinhard W., Schweitzer and Volksen G, “CSCW tools: concepts and architectures”, *IEEE Computer*, Vol.27, pp. 28-36, 1994.

[2] Regli, W., “Internet-enabled Computer-Aided Design”, *IEEE Internet Computing*, pp. 39-51, 1997.

[3] Pahng, F., Sein, N., Wallace, D.R., “Distributed Modeling and Evaluation of Product Design Problems”, *Computer-Aided Design*, pp. 411-423, 1998.

[4] R. Gupta, G. D. Micheli, “System-Level Synthesis Using Re-programmable Components”, In *Proceedings of the European Conference on Design Automation*, pp. 2-7, 1992.

[5] Gang Wang, Wenrui Gong and Ryan Kastner, “System Level Partition for Programmable Platforms Using the Ant Colony Optimization”, *In 13th International Workshop on Logic and Synthesis*, 2004.

[6] Sikun Li, Dawei Wang, Tun Li and Yong Dou, “Distributed Collaborative Partition Method of Reconfigurable SoC Using Ant Colony Optimization”, *In 11th International Conference on CSCW in Design*, Melbourne, pp. 133-138, 2007.

[7] Michael H. Eisenring, “*Communication channel synthesis for heterogeneous embedded systems*”, Ph.D. thesis, SWISS Federal institute of Technology ZURICH, No.14640, 2002.

[8] M. Kaul, V. Srinivasan, et al, “Partitioning and Synthesis for Run-Time Reconfigurable Computers Using the SPARCS System”, *In Proceedings of the 1998 Military and Aerospace Applications of Programmable Devices and Technologies Conference*, NASA Goddard Space Flight Center, 1998.

[9] M. Dorigo, V. Maniezzo, and A. Colorni, “Ant System: Optimization by a Colony of Cooperating Agents”, *IEEE Trans on Systems, Man and Cybernetics*, Part-B, Vol. 26, pp. 29-41, 1996.

[10] T. Stutzle, H. H. Hoos, et al, “MAX-MIN Ant System”, *Future Generation Computer System*, Vol. 16, pp. 889-891, 2000.

[11] Sikun Li, Dawei Wang and Peng Zhao, “An Architecture Template based SoC Transaction Level Modeling and Simulation Method”, *The Seventh International Conference on Computer-aided Industrial Design&Conceptual Design*, Hangzhou, pp. 362-367, 2006.

[12] Yong Dou, Xicheng Lu, “LEAP: A Data Driven Loop Engine on Array Processor”, *The 4th Int'l Conf on Parallel and Distributed Computing, Applications and Technologies*, 2003.

[13] M. D. Smith and G. Holloway, “An Introduction to Machine SUIF and Its Portable Libraries for Analysis and Optimization”, *Division of Engineering and Applied Sciences*, Harvard University, July 2002.

[14] Xiong Zhihui, Li Sikun and Chen Jihua, “Hardware/Software Partitioning Based on Ant Optimization with Initial Pheromone”, *Journal of Computer Research and Development*, Vol. 42, pp. 2176-2183, 2005.