# Hardware/Software Partitioning Algorithm Based on Genetic Algorithm

Guoshuai Li
Aeronautics and Astronautics Engineering College, Air Force Engineering University, Xi'an, China
Email. lgsman1@163.com

Jinfu Feng, Junhua Hu, Cong Wang and Duo Qi
Aeronautics and Astronautics Engineering College, Air Force Engineering University, Xi'an, China
Email. wcsfjf@163.com, wcshjh9231@163.com, chenwang6666@126.com, wcsqd5267@163.com

***Abstract*—To solve the hardware/software(HW/SW) partitioning problem on the system that contains only one CPU, a new algorithm based on GA is studied. Firstly, the concept of hardware orientation is put forward, and then used to create the initial colony of GA and in mutation process, which reduces the randomicity of initial colony and the blindness of search. Secondly in the process of GA, crossover and mutation probability become more and more small, this not only ensures a big search space in the early, but also keeps the good solution later. Experimental statistics show that the efficiency of the proposed algorithm outperforms the algorithms in comparison by up to 23% in large-scale problem. What's more, it can obtain better solution. In conclusion, the proposed algorithm has higher efficiency and appears to be a better solution under specific conditions.**

***Index Terms*—hardware/software partitioning, hardware orientation, genetic algorithm**

## I. INTRODUCTION

HW/SW partitioning technology is a crucial step in SOC HW/SW co-design and embedded systems' realization. The function of it is to decide which components of the system should be realized through hardware and which ones through software, and finally provide the best scheme for system while satisfying design constraints. Clearly, this step has dramatic impact on the cost and performance of the whole system.

Most formulations of HW/SW partitioning problem have been proved NP-hard[1], so exact algorithms tend to be quite slow for large input, hence for large partitioning problem, heuristic methods are generally applied to explore the search space in order to get the nearly optimal solutions, although they cannot guarantee the optimum solution, they comprise the majority of the research and some significant research has been done such as Genetic Algorithm(GA)[2][3], Particle Swarm Optimization(PSO)[4][5], Tabu Search(TS)[6][7], Ant Algorithm(AA)[8][9], Simulated Annealing(SA)[10] as well as some improved schemes.

GA is one of the most widely used random search techniques. It aims to obtain near-optimal partitioning by imitating the process of biological evolution. This technique has been used for HW/SW partitioning in many studies. Paper[11] expounded basic application of GA in HW/SW partitioning where some implementation methods such as coding, fitness function, selection of individuals, crossover, mutation and convergence rule were discussed in detail. Paper[12] partitioned the system into hardware and software components using GA where an enhanced resource constrained scheduling algorithm was used to determine system performance, as a result, execution time and power consumption were reduced greatly. Paper[13] proposed an Advanced Non-Dominated Sorting Genetic Algorithm (ANSGA), by introducing a removing method for building non-dominated sets (NDS) and an elitism preserving strategy for generating NDS and new sets, it not only reduced computational burden but also obtained global convergence. Paper[14] used the idea of two levels of implementation methodology, the first level provided initial solution for GA which run in the second level, results indicated that this method can improve throughput and efficiency with only a small amount of increased design space. In addition, [15][16][17] improved GA algorithm in different aspect and obtained better solution too.

To improve partitioning quality and algorithm efficiency, we propose a new partitioning algorithm based on GA. In this paper, the concept of hardware orientation is put forward and used in the process of producing initial colony and mutating, as a result, it not only avoided the blindness of creating initial colony through random method but also controlled the direction of mutation. Furthermore, we design an adaptive method for crossover/mutation probability. Experimental results demonstrate superiority of proposed approach over existing algorithms in comparison in terms of efficiency and solution quality.

This paper is organized as follows. In section 2 we introduce HW/SW partitioning problem, as well as provide the objective function in this paper. In section 3, we propose the concept of hardware orientation and its calculation. In section 4, we describe the detail of proposed hybrid algorithm based on GA. In section 5, we show experimental results and analysis, and then compare proposed algorithm based on GA with existing algorithms. Finally, section 6 draws conclusions of our work and makes prediction for future work.

## II. Problem Description

HW/SW partitioning is one of the most crucial steps in the design of embedded systems that typically consist of hardware and software components. Before partitioning, it's necessary to identify the construction of implementation platform. This paper discusses platform with single CPU, that is to say, the system consist of one CPU and FPGA or other reconfigurable logic module. The assignment of HW/SW partitioning is to map tasks to either CPU or hardware components while satisfying design constraints.

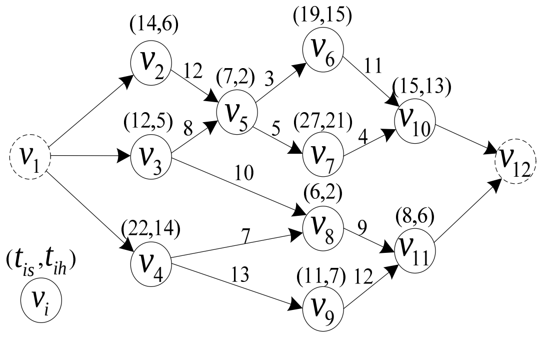

Figure1. The DAG model of system.

We now formalize the problem as follows. Tasks of system which will be partitioned are given in the form of DAG(directed acyclic graph) just as is shown in Fig 1, $G=(V,E)$ [18]. $V=\{v_1,v_2,\cdots,v_n\}$ denotes the set of nodes, while $v_i$ denotes the $i$ th task node, $E$ indicates the set of edges, $c_{ij}$ denotes communication cost between $v_i$ and $v_j$ , $t_{is}$ and $t_{ih}$ are execution time of task through software and hardware respectively, $s_{is}$ and $s_{ih}$ denote area cost of $v_i$ through software and hardware, $S_s$ , $S_h$ and $C$ represent software area constraint, hardware area constraint and communication constraint respectively, $T_{cost}$ is execution time which is defined to be the sum of processing time of all tasks. Assume that communication cost between two adjacent nodes that are all carried out through hardware or software can be overlooked, thus the objective function (1) can be formulated as the following minimization problem using the method in paper[18].

$n$ denotes the number of nodes, $x_i$ denotes how $v_i$ is realized where $x_i=1(x_i=0)$ means $v_i$ is realized through hardware(software), $s_{is}$ and $s_{ih}$ are area cost of $v_i$ realized through software and hardware respectively, the sum of all the $s_{is}$ should be less than $S_s$ , and so is $s_{ih}$ to $S_h$ .

$$\begin{cases} \min\left(T_{cost}=\sum_1^n\left(t_{ih}x_i+t_{is}\left(1-x_i\right)\right)\right) \\ s.t.\quad \sum_1^n\left(s_{is}\left(1-x_i\right)\right)\le S_s \\ \quad\sum_1^n\left(s_{ih}x_i\right)\le S_h \\ \quad\sum_1^n\left(\sum_{k\in V_s}c_{ik}x_i+\sum_{k\in V_h}c_{ik}\left(1-x_i\right)\right)\le C \end{cases} \tag{1}$$

## III. Calculation of Hardware Orientation

In this paper, hardware orientation is defined as superiority of implementation of one task through hardware over software, it is characterized by three metrics which are Area, Time and Communication.

### A. Area-Hardware Orientation

$$S_{iorien}=\begin{cases} A\left(1-B\right)+B & , \quad S_{cons}<S_{sum} \\ 1 & , \quad S_{cons}\ge S_{sum} \end{cases} \tag{2}$$

Because the requirement of typesetting, in (2) we use $A$ to replace $\left(S_{\max}-S_i\right)/\left(S_{\max}-S_{\min}\right)$ , $B$ to replace $S_{cons}/S_{sum}$ . $S_i$ denotes additional area of $s_{ih}$ than $s_{is}$, while $S_{\max}$ and $S_{\min}$ indicate respectively the maximum and minimum value of $S_i$, $S_{sum}$ is total area cost when the whole of the nodes are realized through hardware. In addition, $S_{cons}$ denotes area constraint.

### B. Time-Hardware Orientation

$$T_{iorien}=\begin{cases} \dfrac{\left(t_{is}-t_{ih}\right)/t_{is}-T_{1\min}}{T_{1\max}-T_{1\min}} & , \quad t_{is}>t_{ih} \\ 1-\dfrac{\left(t_{ih}-t_{is}\right)/t_{ih}-T_{2\min}}{T_{2\max}-T_{2\min}} & , \quad t_{is}\le t_{ih} \end{cases} \tag{3}$$

Here $t_{is}$ and $t_{ih}$ denote time cost of $v_i$ realized through software and hardware respectively, while $\left(t_{is}-t_{ih}\right)/t_{is}$ and $\left(t_{ih}-t_{is}\right)/t_{ih}$ are performance ratio, $T_{1\max}$ , $T_{1\min}$ , $T_{2\max}$ and $T_{2\min}$ are the maximum and minimum value of performance ratio.

### C. Communication-Hardware Orientation

$$C_{isrorien}=\sum_j\left[\left(1-D\right)\times t_{sr}+D\times t_{hr}\right]c_{ij} \tag{4}$$

$$C_{isworien}=\sum_j\left[\left(1-D\right)\times t_{sw}+D\times t_{hw}\right]c_{ij} \tag{5}$$

$$C_{ihrorien} = \sum_j \left[ (1-D) \times t_{hr} + D \times t_{sr} \right] c_{ij} \quad (6)$$

$$C_{ihworien} = \sum_j \left[ (1-D) \times t_{hw} + D \times t_{sw} \right] c_{ij} \quad (7)$$

$$C_{iorien} = \frac{C_{isrorien} + C_{isworien}}{C_{ihrorien} + C_{ihworien}} \quad (8)$$

$t_{sr}$, $t_{sw}$, $t_{hr}$ and $t_{hw}$ are reading and writing delay of hardware and software, $D=S_{iorien} \times T_{iorien}$.

Consequently, the synthesized factor of hardware orientation can be described as following.

$$Z_{iorien} = \frac{\alpha S_{iorien} + \beta T_{iorien} + \eta C_{iorien}}{S_{iorien} + T_{iorien} + C_{iorien}} \quad (9)$$

$$\alpha + \beta + \eta = 1,\ \alpha > 0,\ \beta > 0,\ \eta > 0.$$

## IV. Algorithm

In recent years, research based on GA for solving multi-objective optimization problem makes good progress[19][20][21]. Here we mainly consider the following aspects, creation of initial colony, selection of individuals, crossover and mutation.

(1) Coding. In a number of related articles, there are two familiar coding method for GA, the binary and the decimal, by contrast, the binary-biased genetic algorithms have higher searching efficiency, less time- consumption for convergence, wider selecting domain of crossover and mutation probability and stronger robustness of optimized value than decimal-biased genetic algorithms[22]. What's more, considering that the state of the node here includes hardware realization and software realization, we choose binary as coding mechanism. $X = (x_1, \ x_2, \ \cdots, \ x_n)$ denotes a partitioning plan, $1 \le i \le n$.

(2) We make reciprocal of objective function $F(X) = 1/T_{cost}$ as fitness function in this paper.

3. Creation of initial colony. In this paper, we create initial colony on the base of hardware orientation, the bigger the hardware orientation is, the higher the probability that the node is initialized to hardware realization will be, and vice versa. Concretely, we first generate a random number $r_i \in (0,1)$, if $r_i < Z_{iorien}$, $v_i$ will be initialized to hardware realization, otherwise to software realization. In addition, Hamming distance $H(X^i, X^j) = \sum_{k=1}^n \left| X_k^i - X_k^j \right|$ is adopted for the difference among individuals and $H(X^i, X^j) > 4$, $X^i$ and $X^j$ denote partitioning plans which are called individuals here. Repeat the operation above until we have $N_X$ individuals.

(4) Selection of individuals. In order to prevent the precociousness phenomenon from happening, the proposed algorithm selects individuals adaptively according to the change of fitness, consequently, selection probability of $x_i$ can be defined using the method mentioned in paper[23].

$$p_i = f'(x_i) \Big/ \sum_{i=1}^n f'(x_i) \quad (10)$$

$$f'(x_i) = af(x_i) + \frac{(f_{\max} - f_{\min})(e - e^{g/g_{\max}})}{e + e^{g/g_{\max}}} \quad (11)$$

$f(x_i)$ expresses fitness value of $x_i$, while $f_{\max}$ denotes the maximal value of $f(x_i)$ in current colony and $f_{\min}$ indicates the minimal one, $g$ indicates the number of iterations while $g_{\max}$ indicates its maximal value, $a$ denotes the weight of $f(x_i)$ in $f'(x_i)$, it is a constant that is greater than zero and $a = 0.75$ in this paper.

According to this selection strategy, selection probability of individuals with big fitness value can reduce greatly at the beginning of algorithm, this is beneficial for global searching, as the processing of algorithm, selection probability of individuals with big fitness value gradually grow bigger, this is beneficial for the convergence of algorithm.

(5) Crossover and mutation. In this process, some individuals( $N_X/2$ ) are selected for crossover using two-point crossover method[18]. The selection of crossover probability $P_c$ and mutation probability $P_m$ will influence the whole process of genetic algorithm. In other words, the bigger the difference of colony and fitness of individual are, smaller $P_c$ and $P_m$ can help to protect individuals that have bigger fitness, meanwhile, the convergence speed can be improved too. The smaller the difference of colony and fitness of individual are, bigger $P_c$ and $P_m$ can not only help to produce excellent individuals but also prevent algorithm from entering local optimum. Experiments have shown that adaptive change of $P_c$ and $P_m$ can improve algorithm performance than fixed value[24]. Thus we put forward an adaptive method for crossover and mutation probability.

$$P_c = \begin{cases} p_{c1} \dfrac{f_{\max} - \max(f_i, f_j)}{f_{\max} - f_{avg}}, & \min(f_i, f_j) > f_{avg} \\ p_{c2}, & \min(f_i, f_j) \le f_{avg} \end{cases} \quad (12)$$

$$P_m = \begin{cases} p_{m1}\dfrac{f_{\max} - f}{f_{\max} - f_{avg}}, & f > f_{avg} \\ p_{m2}, & f \le f_{avg} \end{cases} \quad (13)$$

$f_{avg}$ denotes average fitness of all the current individuals, while $f$ denotes fitness of individual waiting for mutation, $f_i$ and $f_j$ are fitness of individuals that are waiting for cross, $p_{c1}$, $p_{c2}$, $p_{m1}$ and $p_{m2}$ are constants. If the individual fitness is smaller than $f_{avg}$, bigger $P_c$ and $P_m$ should be selected for promoting melioration of its fitness, whereas, smaller $P_c$ and $P_m$ should be selected for preserving the individual. Considering that individuals with bigger fitness should have smaller crossover probability, thus $P_c$ uses $\min(f_i, f_j)$ as boundary.

(6) Termination criterion. For the sake of ending GA at appropriate time, the proposed algorithm uses dynamic termination criterion. Concretely, we define the largest number of iterations $Gene_{\max}$ and the minimum evolution rate $GeneImproRat_{\min} = 4\%$, if the evolution rate in three successive colonies is not larger than $GeneImproRat_{\min}$ or , GA will terminate.

The pseudo code of GA is as shown in Table1.

TABLE I.
GA PROCESS

```
Input:
    Task graph $G$ and constraints $S_s$, $S_h$, $C$.
Output:
    The HW/SW partitioning result $X = (x_1, x_2, \cdots, x_n)$ and runtime.

1. begin
2. Calculate the comprehensive factor of hardware orientation $Z_{iorien}$ and set termination criterion.
3. Create initial colony on basis of $Z_{iorien}$, make $g = 0$, $N_{GeneImproRat_{\min}} = 0$. // $N_{GeneImproRat_{\min}}$ denotes the successive
   generations that evolution rate is smaller than $GeneImproRat_{\min}$.
4. Calculate fitness of individuals in $P(0)$ and average fitness.
5. Perform the operation between 6 and 21 again and again before meeting termination criterion.
6. Calculate selection probability $p_i$ for every individual in $P(g)$.
7.  for($k = 0; k < N_X; k = k + 2$)
8.  {
9.      Select two individuals on basis of $p_i$.
10.     Create a random number $0 < u < 1$.
11.     if($u < P_m$)
12.         Perform mutation operation for selected individuals, if $Z_{iorien} > 0.9$, no matter what state of the node is, the state is set to 1,
        otherwise, perform the routine mutation operation, the result is put into next colony.
13.     elseif($u < P_m + P_c$)
14.         Perform crossover operation and put the result into next colony.
15.     else
16.         put the individuals into next colony without change.
17.  }//end for
18. $g = g + 1$, calculate the fitness of individuals in $P(g)$ and the average fitness.
19. if($GeneImproRat_{cur} \le GeneImproRat_{\min}$)
20.     $N_{GeneImproRat_{\min}} = N_{GeneImproRat_{\min}} + 1$.
21. end if
22. Output $X_{GA}$ which has the biggest fitness value in $P(g)$ and runtime.
23.end
```

## V. Experiment and Analysis

Creation of test set. For testing, we firstly create randomly several DAGs that have specified number of nodes and average branches, then let every node associated with one function whose cost(hardware area, software area, communication cost and runtime etal) is used to simulate task cost. Eventually, we get 6 DAGs with 30, 60, 90, 120, 200, 400 nodes respectively.
Experimental environment: (1) Pentium(R) Dual-Core 2.5GHz CPU, 2G internal storage. (2) Windows XP operating system. 3. Programming environment is Matlab R2007a.

To verify the effectiveness of proposed algorithm, we choose GA[12]and ANSGA[13] as comparison. Also, in order to make a fair comparison, all related parameters in our experiment are set on the same benchmark, initial crossover probability is set to 0.8 while initial mutation probability to 0.13.

Table3 shows partitioning results of three algorithms, it can be observed that, (1) Proposed algorithm has higher convergence speed, because hardware orientation can reduce randomicity of initial colony and affect search direction, to sum up, these two aspects reduce the number of iterations. What's more, selection strategy is also helpful to improve search speed. (2) On small-scale problem when the number of nodes is less than 60, proposed algorithm has lower efficiency than ANSGA, because the calculation of hardware orientation costs a long time. However, with the addition of scale, when the number of nodes is more than 90, the proposed algorithm can not only obtain better solution but also improve operating efficiency of nearly 23%. What's more, the larger the scale is, the bigger the improvement will be. The reason is that ANSGA must array the colony, construct non-dominated set and new groups in every iteration and this will cost much time. (3) Compared with the other two algorithms, GA has the shortest runtime, whereas it only obtains solution with lowest quality, because basic GA has great dependence on initial solution, a good initial solution could result in a good final solution, while a bad one will affect the quality of final solution.

TABLE II.
EXPERIMENTAL RESULTS

| Number of nodes | $S_h$ | $S_s$ | $C$ | Algorithm | $T_{\cos t}$ | Cost of time |
|---|---|---|---|---|---|---|
| 30 | 2850 | 1409 | 591 | GA | 5897 | 81 |
| | | | | ANSGA | 5692 | 847 |
| | | | | Our algorithm | 5685 | 896 |
| 60 | 5909 | 2943 | 1356 | GA | 11904 | 240 |
| | | | | ANSGA | 10859 | 8263 |
| | | | | Our algorithm | 10813 | 8379 |
| 90 | 8684 | 4308 | 2014 | GA | 17730 | 1158 |
| | | | | ANSGA | 16728 | 57573 |
| | | | | Our algorithm | 16347 | 49413 |
| 120 | 11611 | 5770 | 2637 | GA | 22493 | 1372 |
| | | | | ANSGA | 21022 | 114437 |
| | | | | Our algorithm | 19844 | 95108 |
| 200 | 17192 | 9416 | 3783 | GA | 37441 | 1836 |
| | | | | ANSGA | 34739 | 280347 |
| | | | | Our algorithm | 33053 | 224318 |
| 400 | 28154 | 15681 | 6539 | GA | 75620 | 2925 |
| | | | | ANSGA | 70209 | 674301 |
| | | | | Our algorithm | 66681 | 510302 |

To intuitively show experimental results, we let the three algorithms run 30 times respectively for the 6 DAGs, then calculate average value of the results for each DAG, finally we obtain the comparison between proposed algorithm and the other two algorithms on partitioning results and runtime, as is shown in Fig 3 and Fig 4.

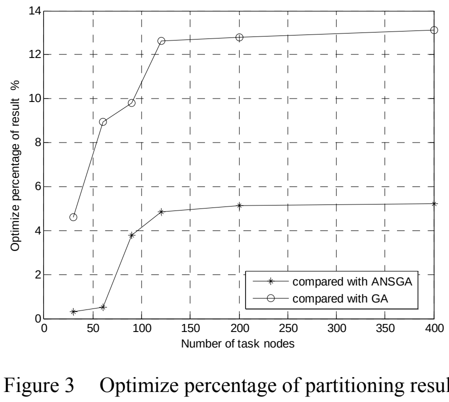

Figure 3 Optimize percentage of partitioning result

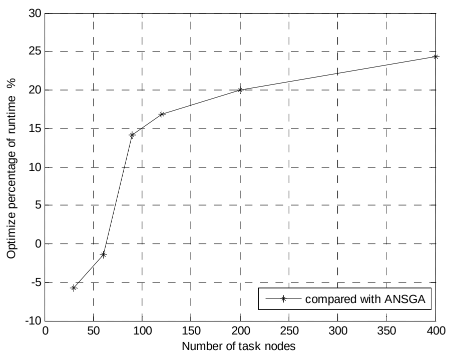

Figure4. Optimize percentage of runtime

Assume that there are 400 task nodes. We obtain initial solution using random method and hardware orientation respectively, the simulation results of ANSGA are shown in Fig 5, as can be seen, initial solution coming from hardware orientation has faster convergence rate and bring on better solution. Fig 6 shows simulation results of ANSGA and our algorithm, it intuitively illustrates the advantage of proposed algorithm on large-scale problem.

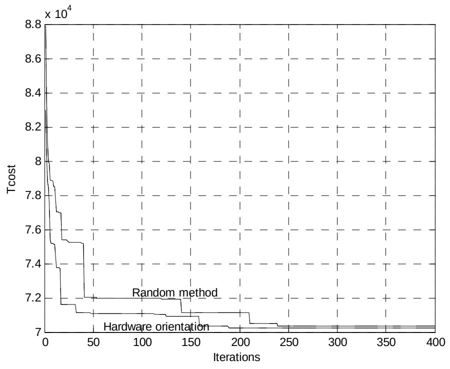

Figure 5 Simulation results of ANSGA with diffident initial solution

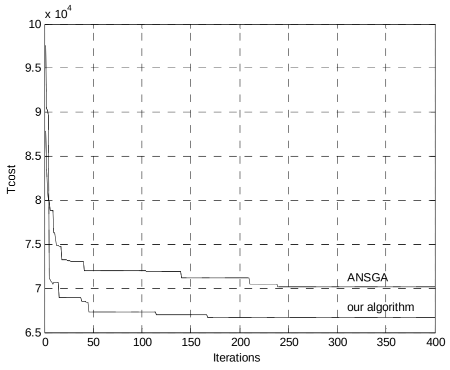

Figure6. Simulation results of proposed algorithm and ANSGA

## VI. Conclusions

Based on GA, we present a simple but very efficient algorithm for solving HW/SW partitioning problem in this article. Compared with the other two algorithms, time complexity of proposed algorithm includes additional time for calculating hardware orientation except for calculating genetic operation and fitness. However, the application of hardware orientation reduces the number of iterations and it only need calculated once, as a result, the proposed algorithm can obtain higher efficiency especially on large-scale problem.

The value of crossover and mutation probability is larger than classical GA in this paper, this may induce blindness of search, but the use of hardware orientation and adaptive method prevent its occurrence. Furthermore, they increase the probability of introducing new chromosome which can increase GA's ability of local search to some extent.

In order to simplify the problem, the proposed objective function doesn't take into account power cost which may impact partitioning accuracy, our on-going work will improve the objective function.

## References

[1] Arato P, Mann ZA, Orban A. Algorithmic aspects of hardware/software partitioning, ACM Trans Des Autom Electron Syst. 2005. 10(1): pp. 136-156. doi: 10.1145/1044111.1044119.

[2] Glover F, Kelly JP, Laguna M.: Genetic algorithms and tabu search: hybrids for optimization, Computers and Operations Rsearch. 1995. 22(1): pp. 111-134. doi: 10.1016/0305-0548(93)E0023-M.

[3] Jian Pan, Guohong Mao, Jinxiang Dong.: A Web-Based Platform for Intelligent Instrument Design Using Improved Genetic Algorithm, Journal of software, 2012. 7(10): pp. 2333-2340. doi:10.4304/jsw.7.10.2333-2340.

[4] Shaobo Zhong, Zhongshi He.: Application of Particle Swarm Optimization Algorithm based on Classification Strategies to Grid Task Scheduling, Journal of software, 2012. 7(1): pp. 118-124. doi:10.4304/jsw.7.1.118-124.

[5] Shih-An Li, Chen-Chien Hsu, Ching-Chang Wong, Chia-Jun Yu.: Hardware/software co-design for particle swarm optimization algorithm, Information Sciences. 2011. (181): pp. 4582-4596. doi:10.1016/j.ins.2010.07.017.

[6] Jigang Wu, Srikanthan Thambipillai, Ting Lei.: Efficient heuristic algorithms for path-based hardware/software partitioning, Mathematical and Computer Modelling. 2010 (51): pp. 974-984. doi:10.1016/j.mcm.2009.08.029.

[7] Jigang Wu, Pu Wang, Siew-Kei Lam, Thambipillai Srikanthan.: Efficient heuristic and tabu search for hardware/software partitioning, The Journal of Supercomputing. 2013. 66(1): pp. 118-134. doi:10.1007/s11227-013-0888-9.

[8] Yu-don Zhang, Le-nan Wu, Geng Wei, Han-qian Wu, Yong-liang Guo.: Hardware/software partition using adaptive ant colony algorithm, Control and Decision. 2009. 24(9): pp. 1385-1389. doi:10.3321/j.issn:1001-0920.2009.09.021.

[9] T. He, Y. Guo.: Power consumption optimization and delay based on ant colony algorithm in network-on-chip, Engineering Review. 2013. 33(3): pp. 219-225.

[10] Henkel J, Ernst R.: An approach to automated hardware/software partitioning using a flexible granularity that is driven by high-level estimation techniques, IEEE Transactions on VLSI Systems, 2001. 9(2): pp. 273-289. doi: 10.1109/92.924041.

[11] Yun Zheng, Guo-yong Huang.: System Level Software/Hardware Partitioning by Genetic Algorithm, Journal of Computer-Aided Design & Computer Graphics, 2002, 14(8): pp. 731-734.

[12] Madhura Purnaprajna, Marek Reformat, Witold Pedrycz .: Genetic algorithms for hardware-software partitioning and optimal resource allocation, Journal of Systems

Architecture, 2007(53): pp. 339-354. doi: 10.1016/j.sysarc.2006.10.012.

[13] Sheng-qin Luo, Xiao-xiao Ma, Yi Lu .: An Advanced Non-Dominated Sorting Genetic Algorithm Based SOC Hardware/Software Partitioning, ACTA ELECTRONICA SINICA, 2009 (11):pp. 2595-2599. doi: 10.3321/j.issn:0372-2112.2009.11.043.

[14] Nithiyanantham Janakiraman, Palanisamy Nirmal Kumar.: Multi-objective module partitioning design for dynamic and partial reconfigurable system-on-chip using genetic algorithm, Journal of Systems Architecture, 2014, 60(1): pp.119-139. doi: 10.1016/j.sysarc.2013.10.001.

[15] Yan Kang, He Lu, Jing He.: A PSO-based Genetic Algorithm for Scheduling of Tasks in a Heterogeneous Distributed System, Journal of software, 2013, 8(6): pp. 1443-1450. doi:10.4304/jsw.8.6.1443-1450.

[16] Xiso-dong Guo, Ji-ren Liu, Hui Wen.: A method for hardware/software partitioning using genetic algorithm, Journal of Computer-Aided Design & Computer Graphics, 2001, 13(1): pp. 24-27. doi: 10.3321/j.issn:1003-9775.2001.01.005.

[17] Peng Liu, Ji-gang Wu, Yong-ji Wang.: Hybrid algorithms for hardware/software partitioning and scheduling on reconfigurable devices, Mathematical and Computer Modelling, 2013, (58): pp. 409-420. doi: 10.1016/j.mcm.2012.11.001.

[18] An Liu, Jin-fu Feng, Xiao-long Liang, Xiao-tian Yang.: Algorithm of hardware/Software partitioning based on genetic particle swarm optimization, Journal of Computer-Aided Design & Computer Graphic, 2010. 22(6): pp. 927-933,942.

[19] Yuan-yuan Cui, Xue-hong Qiu, Jian-xian Zhang, Rui Zhou.: SoC hardware/software partitioning algorithm for multi-performance index constraints, JOURNAL OF XIDIAN UNIVERSITY, 2013, 40(5): pp. 92-98. doi: 10.3969/j.issn.1001-2400.2013.05.015.

[20] Pankaj Kumar Natha, Dilip Dattab.: Multi-objective hardware–software partitioning of embedded systems: A case study of JPEG encoder, Applied Soft Computing, 2014 (15): pp. 30-41. doi:org/10.1016/j.asoc.2013.10. 032.

[21] Madhura Purnaprajna, Marek Reformat, Witold Pedrycz.: Hardware implementation of a novel genetic algorithm, Journal of Systems Architecture, 2007 (53): pp. 339-354. doi: 10.1016/j.sysarc.2006. 10.012.

[22] Jin Zhang, Dong-li Li, Ping Li. : Comparative study of genetic algorithms encoding mechanism, Journal of China University of Mining &Technology, 2002. 31(6): pp. 637-640. doi: 10.3321/j.issn:1000-1964.2002.06. 023.

[23] Yong-gang Peng, Xiao-ping Luo, Wei Wei.: New fuzzy adaptive simulated annealing genetic algorithm, Control and Decision, 2009.24(6): pp. 843-848. doi: 10.3321/j,issn:1001-0920.2009.06008.

[24] Yun-xiao Zu, Jie Zhou.: Cognitive radio resource allocation based on combined chaotic genetic algorithm, Acta Phys.Sin, 2011. 60(7): pp. 1-8. doi: 10.7498/aps. 60.079501.

**Guoshuai Li** was born in ShanDong (China) on January 1986. He received the B.E. and M.E. degrees from Aviation University of Air Force, China, in 2008 and 2011 respectively, and he is currently working toward the Ph.D. degree in School of Aeronautics and Astronautics Engineering College, Air Force Engineering University, China. His research interests mainly focus on computer aided design, system analysis and storage management system .

**Jinfu Feng** received the Ph.D. degrees from Nanjing University of Science and Technology, China, in 1996. He is currently a Professor and PhD Supervisor in the School of Aeronautics and Astronautics Engineering College, Air Force Engineering University, China. Her interests are in the areas of storage management system, PIM/PSM technology and computer aided design.

**Junhua Hu** received the Ph.D. degrees from Aeronautics and Astronautics Engineering College, Air Force Engineering University, China. He is currently a teacher in the School of Aeronautics and Astronautics Engineering College, Air Force Engineering University, China. Her interests are in the areas of PIM/PSM technology and computer aided design.

**Cong Wang** received the S.E. and M.E. degree from Armed Police Engineering University, China, in 2008 and 2011 respectively. He is currently working toward the Ph.D. degree in School of Aeronautics and Astronautics Engineering College, Air Force Engineering University, China. His research interests mainly focus on reliability analysis, computer aided design and system design.

**Duo Qi** received the B.E. and M.E. degrees from Aviation University of Air Force, China, in 2010 and 2013 respectively, and he is currently working toward the Ph.D. degree in School of Aeronautics and Astronautics Engineering College, Air Force Engineering University, China. His research interests mainly focus on computer aided design, system analysis.