A Novel Approach to Hardware/Software Partitioning for Reconfigurable Embedded Systems

Linhai Cui
School of Software, Harbin University of Science and Technology, Harbin, China
Email:cuilinhai@hrbust.edu.cn

***Abstract*—Hardware/software partition is a crucial point in the design of a reconfigurable embedded system. Reconfigurable computing is a promising approach to overcome the traditional trade-off between flexibility and performance in the design of computer architectures which adapt their hardware to each application to achieve a high performance of dedicated hardware. In this paper, some hardware and software partitioning algorithms were analyzed and summarized first, then a innovative algorithm for task partition and scheduling is proposed based on new features of reconfigurable hardware such as dynamic reconfiguration and the delay of reconfiguration. In the proposed algorithm, a large-scale application is decomposed into multiple sub-tasks of suitable granularity and each sub-task has constraint relationship with each other. And a directed acyclic graph (DAG) which presents the relationship between tasks was drawn according to the execution order of tasks. Then the specific application presented in the DAG is mapped to the hardware and software platform by a strategy called GATS which combine the Genetic Algorithm and the Tabu Search algorithm together. The shortest time of assignment and task execution order can be found by the priority-based scheduling method. The experimental results show that the method is of high performance and can effectively mapping the application task to the reconfigurable system.**

***Index Terms*—Reconfigurable embedded system, Task scheduling, Hardware/software partitioning, Genetic algorithm; Tabu search algorithm**

I. INTRODUCTION

Most modern electronic systems are composed of both hardware and software. Embedded system is some combination of computer hardware and software to perform a particular function. It can be found in many applications such as automobiles, telecommunication systems, intelligent home devices, medical equipments, and in systems for the military.

Comparing to the hardware parts, the software parts are much easier and faster to develop and modify. Thus, software is less expensive in terms of the development cost and time. Hardware, however, provides better performance. For this reason, an embedded system designer's goal is to minimize the weighted sum of the software delay, Hardware area, and power consumption.

There are two basic implementations of embedded system. One is hardware and the other is software. The hardware method achieves system functionality through the design of dedicated hardware logic circuits while the software method is based on microprocessor software to complete the system functions through the design of the program.

The main task of hardware/software partition is to assign the system functions to the target structure on the software and hardware domain under the condition of meeting the design constraints, and its essence is a kind of combination optimization problem. It includes the following three aspects: first, processing unit allocation, i.e. to determine the type and number of required software and hardware processing unit; second, task assignment, i.e. to assign tasks and communication to the target structure in the processing unit and communication resources to execute and to meet the performance and cost constraints; third, task scheduling, i.e. to determine the order of execution and the start time of the assigned task and communication in each processing unit to meet the dependency relationship of control and data between the system tasks. The solution space is a huge multi-dimensional non-contiguous space, so it is difficult to solve if taking the solution quality and solution time into account. Therefore, only the execution time, cost, power and other major overhead are considered when studying the hardware and software partition to reduce the difficulty of solving the whole problem by simplifying the model of the target structure.

Most embedded systems use CPU (Central Processing Unit) + ASIC (Application Specific Integrated Circuit) structure. For the systems which only have a hardware processing unit (ASIC) and a software processing unit (CPU), it is relatively simple to partition the system, and it is called binary partitioning. For the systems which have multi-processing units, the hardware processing units and software processing units may not be the same, and it may be more complicated to partition the system, such problems are called multi-way partitioning.

In addition to the CPU + ASIC structure, FPGA (Field Programmable Gate Array)-based reconfigurable hardware system has been developed. There are two kinds of reconfiguration concern to the time when

doi:10.4304/jcp.7.10.2518-2525

reconfiguration take place. One is static reconfiguration and the other is dynamic reconfiguration. More and more studies have been focused on dynamic reconfigurations. And a growing number of embedded systems employ dynamic reconfigurable architecture. In order to use the dynamic reconfiguration efficiently, one needs a support of operating systems to manage both software and hardware. Therefore, the structure of the traditional CPU + ASIC method is no longer suitable to apply to software and hardware reconfigurable systems.

Reconfigurable hardware components such as FPGAs are used more and more in embedded systems, since such components offer a sufficient capacity for a complete SoC (System on a Chip) or even NoC (Network on a Chip). The advantage of a reconfigurable hardware platform over pure software reconfiguration is that it can provide a system implementation flexibility to be adaptable to new functional requirements while meeting constraints for critical system parameters such as data throughput rates and latencies.

When doing partition, we need not only to assign the system tasks to the software or hardware domain, but also need to divide the tasks which is possibly assigned on the reconfigurable devices into different segments which is not overlapping in time. Measures should be also taken to reduce the delay caused by the reconfiguration when deigning a reconfigurable embedded system.

In this paper, we proposed a portioning algorithm called GATS which employs the advantages of traditional algorithms such as Genetic and Tabu, and a task schedule approach by using DAG.

II. Related Work

With the development of integrated circuit technology, embedded system is moving towards small size, mobile, portable, light in weight, low power consumption, more complex and so on. Traditional design approach has become a bottleneck restricting the development of embedded systems. Software design and hardware design are required to be integrated closely and coordinated with each other. This leads to the development of a new design theory - hardware and software co-design.

A. *Evolution of Hardware/software Partitioning*

The studies of hardware/software co-design began in the early 1990s, the idea of hardware/software co-design is formally proposed in the first International Workshop on Hardware/Software Codesign (CODES), held in 1992. Then many famous universities set up research group on embedded systems engaged in software and hardware co-design theory and research. Some EDA venders have also introduced some tools supporting hardware and software co-design.

SOS system [1], developed by Prakash and Parker from the University of Southern California, is the first hardware and software co-design systems in the world. The system can schedule tasks on multiple processors, but it was slow, not suitable for large-scale systems.

The COSYMA (Co-synthesis for Embedded Architecture) [2] system, developed by the German Technical University of Branunshweig, is mainly restricted to a single processor and a single ASIC system. Its partition method is mainly for software to optimize the calculation through co-processors. The main drawback of COSYMA is that the processor and the coprocessor can not work concurrently [3].

The Corsair system [4], developed by Frank Slomka, is an embedded system design environment suitable for multi-processor and multi ASIC structure. The system generates the system model by using tabu search algorithm. But it uses static method to assess the system performance when the system model is generated, so it can not evaluate complex systems.

In 1997, Eles proposed to achieve the hardware and software partition by using simulated annealing and tabu search algorithm. He described a model called condition task graph using list scheduling algorithm to realize the structure for each processing unit to form the scheduling table and as a basis for selection of software and hardware. Experiments result shows that the tabu search algorithm is more suitable for hardware and software partition compared to simulated annealing, [5].

In 1999, Henkel introduced the IP-based low-power embedded systems hardware and software partitioning algorithm. The idea is to reduce the system's idle time to reduce the power consumption [6]. In 2002, Theerayod Wiangtong and others compared and analyzed the three heuristic algorithms of hardware and software co-design and finded that tabu search algorithm is proved to be superior in hardware and software partition to the genetic algorithm and simulated annealing algorithm [7]. In 2006, Michalis D Galanis proposed a hybrid reconfigurable system [8].

B. *Research on Task Scheduling*

Scheduling problem is a kind of combinational optimization problem and is applied in many computer and communications fields. It has a close relationship with the algorithm design, complexity theory.

The research related to task scheduling was first proposed by Liu and Layland [9] which ignores some implementation details. It is the basis of many real-time task models, and extends up to the processor environment with task scheduling and feasibility analysis in algorithm design.

Reference [10] proposed a grouping appropriate algorithm, which is a non-dynamic scheduling algorithm for periodic tasks. Due to the use of the grouping strategy and appropriate scheduling policy, the utilization of platform resource and processor is increasing in recent years.

Hsu Heng Ruey from China Taiwan National University researched real-time periodic task scheduling problem by using dynamic voltage scheduling technique for a given energy constraints [11]. In recent 20 years, the research on the parallel task scheduling on multiprocessor represented in directed acyclic graph DAG (Directed Acyclic Graph) has been developed rapidly. DAG-based task scheduling is to map the distribution of tasks to processors and coordinate the

implementation. Under the condition of meeting constraints, the overall execution time, power consumption, area, and other indicators of the task are best. It is NP-complete problem.

Becchi and Crowley thought that task management is the key to raising computing performance of a multi-processor platform, and they developed a run-time monitoring program to capture the dynamic behavior of the process, allowing the process to migrate between multiple processors. Experiments show that to use dynamic process allocation method on a heterogeneous multi-processor platform can significantly improve overall system performance [12]. In recent years, some researchers began to use some new methods to solve the multiprocessor scheduling problems such as genetic algorithm for multi-processor task scheduling [13].

The introduction of the new method of calculation improves the solution accuracy. But the efficiency of the algorithm need to be further improved. For the task scheduling on CPU + FPGA structure, more research focused on how to place the hardware tasks on the FPGA dynamically. However, less corresponding research are made in task allocation, task migration and other issues of mixed task scheduling [14].

III. Reconfigurable System Besed on FPGA

With the emergence of programmable device, especially field-programmable gate array (FPGA), the reconfigurable technology is developed rapidly in embedded applications. The development of reconfigurable technology makes the traditional boundaries over hardware and software blurred.

The so-called reconfigurable means that in an information processing system under control of software, if the system can be reformed into a different information processing systems to adapt to different application requirements by using reusable resources, the information processing system is called reconfigurable [15].

By using reconfigurable technology, the system can be realized in software and hardware in the case only a little more resources are needed. On one hand, the calculation task can be accomplished by building a dedicated hardware circuit on FPGA, similar to the ASIC. On the other hand, different tasks can be optimized by building different circuit on FPGA.

Reconfigurable Systems based on Large-scale programmable device, FPGA, perform reconfigurable of circuitry at runtime by using the features of can be repeated programming and configuration of FPGA. It can dynamically change the circuit structure while a real-time electronic systems work. Its essence is to achieve the time-sharing reuse of all or part of the internal FPGA logic resources. It can make the logic circuits which are discrete in time works in order on the same FPGA.

A. *Dynamic Reconfigurable Technology*

For dynamic reconfigurable, a special study group RAW introduced the following description in 2005 [16]: The characteristics of dynamic reconfigurable are that the hardware architecture or devices can quickly change (while the system running) its functions and connections.

As shown in Figure 1, the three key issues for the research on dynamic reconfigurable system are the hardware platform, the mapping from specific application to the hardware platform, and the controls needed during the running of the system.

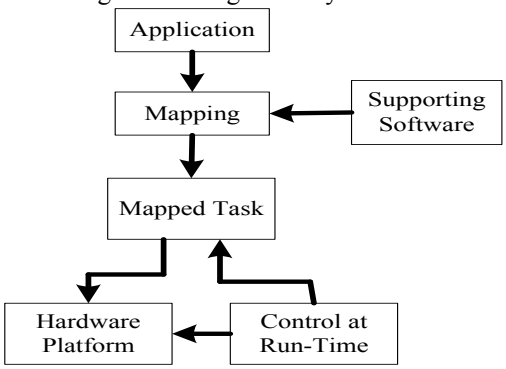

Figure1. Research contained in dynamic reconfiguration system

According to the reconfiguration ways of the reconfigurable logic, the hardware supporting dynamic reconfiguration can be divided into context configure devices and part reconfigurable configure device [17]. Typically, the variable and time-consuming parts of the system are implemented in hardware and the complex controls and data structures are implemented in software [18].

B. *Unified Management of Hardware and Software*

The emphasis of unified management of hardware and software of dynamically reconfigurable system is focused on the management of the hardware tasks. The hardware modules are converted into hardware tasks, and to be managed under the operating system, and then the unified management by the extended operating system is implemented.

In order to convert hardware modules into the hardware tasks, certain constraints need to be imposed on the tasks first to enable it to response to the basic communication and control primitives in the operating system making the user call the hardware tasks as normal software tasks.

After the completion of the hardware tasks building, the expanded operating system will be able to manage them. The next step is how to manage that task, that is when the hardware can be downloaded into a piece of programmable logic resources.

In short, the difference between hardware tasks and software tasks need to be fully taken into accounts when they are managed under the operating system. Some measures must be taken to reduce the preparation time such as by pre-configured configuration, scheduling, etc.

IV. Partition Strategy

Hardware/software co-design is to give an algorithm which can automatically search for the best compromise point between the hardware and software under certain

constraints and to produce the actual system architecture.

A. *Dynamically Reconfigurable System Modeling*

There are significant differences in performance and cost between software and hardware implementation. Hardware and software partition is one of the key issues in co-design. Its goal is to maximize resource utilization, and to minimize application execution time under the constraints to meet the time and shared resources conflict conditions constraints.

A typical dynamic reconfiguration system is shown in Figure 2. It consists of microprocessor, configuration controller, reconfigurable hardware (FPGA), memory and configuration file memory. To rescue the microprocessor from the task of configure FPGA, and to make them perform parallel computing, an additional configuration controller is added to configure the FPGA.

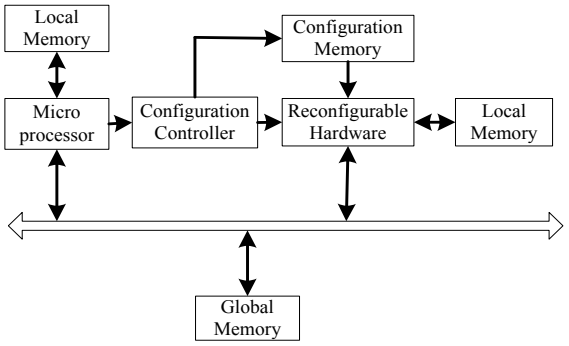

Figure2. Dynamic reconfiguration system

When the microprocessor executes a hardware configuration instruction, data is passed to the configuration controller, and the latter retrieve the corresponding configuration data from the configuration memory, and download it to the FPGA to complete the configuration. Microprocessor and the reconfigurable hardware communicate by sharing memory, so the two tasks located on the microprocessor and reconfigurable hardware not only need time for data transfer and communication, but also need data reading and writing time.

The calculation model must be considered as an important element in the hardware and software partition. Different levels of abstraction of the system forms the different calculation model, and the partition can be performed in a different granularity. These models have different features and application areas, and they can be divided into the following categories: finite state machine model, data flow diagrams, Petri nets, data / control flow graph, task flow diagram.

Task flow diagram, also called a directed acyclic graph (DAG), is the behavior level description of the system. It's purpose is to describe the controls between the tasks in the reconfigurable system, data relationship and the cost information for each task. It has nothing to do with the system architecture. It can be expressed as a triple G = (V, R, E), as shown in Figure 3, the V on behalf of the task node set, R is the edge connecting node, the node E on behalf of its right to inter-communication.

Each node in the diagram represents a system task (or a function module), including its software, hardware, cost information, the edge represents the data flow or control relationship between tasks, and its weight is on behalf of the communication overhead between two tasks.

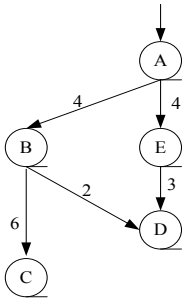

Figure3. Task flow diagram

B. *Hardware and Software Partition*

Functional partition between hardware and software and its implementation vary with the applications. The solution is to find the time / space mapping from the application described with the task flow diagram to the reconfigurable system consist of microprocessor and reconfigurable hardware.

There are three main methods to solve the hardware and software partition problem. The first one is planning method which uses planning to partition the problem and to get the optimal solution. Its drawback is the high computational complexity and large memory overhead. The second method is construction approach. The construct method compares the pros and cons of the solution layer by layer first, and then the optimal solution will be drawn from each comparison group to obtain the optimal solution, such as poly-clustering technology. The advantage is that you can find the optimal solution or near optimal solution more efficiently. The third one is search methods. It includes local search method and oriented random search method. By searching its neighborhood and replacing the current solution to achieve the optimization.

Oriented random search methods include tabu search, simulated annealing, genetic algorithms, and etc. The main idea is to use some guidance rules to guide the search within entire solution space for good solutions. Randomly oriented search method does not rely on objective information, and can be used to solve complex issue. At present, most researchers have adopted oriented random search method.

C. *Traditional Partitioning Algorithm*

Genetic Algorithm (GA) is an algorithm based on Darwin's biological evolution theory. It simulates the biological mechanisms of natural selection and genetic search heuristic. Each element in the solution space is encoded and then divided optimal solution space into groups by iteration to find the optimal solution. Crossover and mutation operator are the two most important components of GA hardware and software partition which are repeatedly applied to the solution of the problem encoding and form chromosomes.

As a global optimization search algorithm, genetic

algorithm is simple, universal, robust, and has wide applications. It can meet the requirements of multi-partition, and it has become the key technology to deal with traditional search methods to solve difficult, complex and nonlinear problems. But it also has many shortcomings, such as the genetic algorithms search the solution space in parallel way. This makes it have strong global search ability, but the local optimization ability is poor. The remature convergence phenomena may occur.

Tabu search (TS) is a algorithm simulating human intelligence process. It is the expansion of the local field search, and is a global step by step optimization algorithm. A flexible memory structure and the corresponding Tabu search criteria was introduced to avoid circuitous search.

Fields, tabu table, tabu length, candidate solutions and amnesty criteria are the key factors in tabu search algorithm design. There are two tables in the tabu search algorithm which impact the performance of search algorithm: the length of taboo objects. Compared with traditional optimization algorithms, tabu search algorithm has flexible memory and amnesty guidelines, and in the search process, it can accept inferior solutions, with a strong climbing ability, it can jump out of local search optimal solution, turn to the other regions of solution space, leading to better probability of a global optimal solution. It is a strong local search global iterative optimization algorithm.

However, the tabu search algorithms have significant deficiencies, such as: strong dependence on the initial solution, a good initial solution can make the tabu search in the solution space to search for good solutions, and poor initial solution will reduce convergence speed of tabu search. It is not efficient to search the solution space in individual, serial way. Its global search capability is not strong.

D. *Proposed Partitioning Algorithm*

After analyzing the genetic (GA) and Tabu Search (TS) algorithm, a hybrid approach based on GA and TS strategy, called GATS was proposed. It can map the specific application to a software and hardware platforms under the reconfigurable system resource constraints and other conditions.

The main idea of GATS is to make the TS as a mutation operator of GA, TSM. It employ the strong memory function and the climbing ability of TS to overcome the weaknesses of poor climbing ability of GA. And it remains the advantage of multi starting point maintaining in the GA.

Genetic Algorithm can not be directly used in the solution of the problem space. So what you need to do is to encode the solution by using chromosome. Encoding is the most important thing required to apply genetic algorithms to solve the problem, and it also is a critical step in designing genetic algorithm. The encoding method will directly affect the solution quality, restrict the choice of genetic manipulation. For a single microprocessor and single reconfigurable hardware architecture, the binary encoding makes the encoding and decoding simple, and also makes the crossover operation and mutation operation easy to implement.

Each task is represented as a binary genes, gene is mapped to the microprocessor if its value is 0, and it is mapped to the reconfigurable hardware if it is1. The chromosome is a vector whose length is N standing for the number of the tasks. The coding chart of tasks is shown in Figure 4. This encoding method makes the crossover and mutation operation easy to use and does not produce invalid chromosomes.

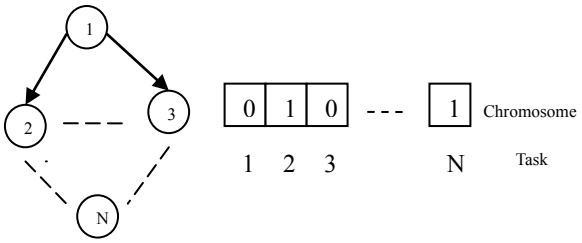

Figure4. Coding Chart of Tasks

For reconfigurable hardware, the above logic resources are part dynamically reconfigurable at runtime. So that for any a different chromosome, the number of different modules in reconfigurable hardware, the process of reconfiguration, the time of reconfiguration, and the calculation overlap will change according to the assigned tasks on reconfigurable hardware that is genetically different.

The choice of fitness function can be the first target that is based on algorithm to optimize the overall system running time T and the cost to construct a generalized objective function C, then by scaling the objective function to be generalized. Running time is converted to the general penalty term in the objective function. Note that the choice of penalty term should ensure that the results meet the constraint requirements, and partition results must also ensure the full utilization.

In addition, costs and the values of run-time may vary on the scope and magnitude. Normalization is needed. Equation (3-1) and (3-2) were used to introduce the normalization factor σ_c and σ_t.

$$\sigma_C = CostHw - CostSw \tag{3-1}$$

$$\sigma_t = \max(TimeSw - Time\,\mathrm{Req}, Time\,\mathrm{Req} - TimeHw) \tag{3-2}$$

Where *CostHW* stands for the cost when all the nodes are implemented in hardware, and *costSW* stands for the cost when all the nodes are implemented in software. *TimeHW* and *TimeSW* stand for hardware execution time and software execution time respectively. *Time*Req is the constraint time provided by the designer, and its value is between *TimeHW* and *TimeSW*. So the generalized objective function can be defined as:

$$COBJ = \alpha \exp\left(\frac{T - Time\,\mathrm{Req}}{M_i \sigma_t}\right)\left|\frac{T - Time\,\mathrm{Req}}{\sigma_t}\right| + (1-\alpha)\frac{C}{\sigma_c} \tag{3-3}$$

And the fitness function is defined as:

$$Fitness = \frac{1}{1 + COBJ} \tag{3-4}$$

The C and T in formula (3-3) stand for the partition cost and execution time respectively. And α here is used to adjust the relative weight of C and T. Since

system performance determines whether the design is available, so α equals 0.6.

V. TASK SCHEDULING ALGORITHM

One of the key technologies of traditional operating system is task scheduling. In dynamically reconfigurable systems, especially in large-scale applications, the hardware task can not be configured to the reconfigurable device at one-time. Then scheduling becomes more important, and the scheduling algorithm have a direct impact on the system performance.

Task scheduling has two main purposes. One is to fully use the resources on the device and the second is to optimize the device configuration sequence. Scheduling should be done to have the tasks whose execution order are more close or simultaneously at one-time schedule to the device. To optimize the configuration sequence can reduce the configuration time overhead caused by the configuration process, and it can reduce the configuration time is too long to the system bottleneck effect.

Scheduling on reconfigurable hardware is also a constrained layout problem. What you need to do is not only to find out the schedule start time, but also to identify the layout position of tasks in reconfigurable hardware under certain conditions and resource constraints.

In summary, the purpose of this section is to find the shortest time of assignment in entire task flow diagram and the task execution order based on the partition result by using genetic and tabu search algorithm.

A. *Scheduling Algorithms based on DAG*

In practice, for any DAG graph in which the weight values of the nodes and edges can be any value, heuristics method is the first choice for solving those DAG scheduling problems. To sum up, the current DAG scheduling algorithm can be divided into four categories: List Scheduling algorithm, Clustering scheduling algorithm, Scheduling algorithm based on Task Duplication, and random search technology.

The basic idea of list scheduling algorithm is to sort the priority of node, to construct a scheduling list, so that all the ready tasks are in the schedule list, and then the highest priority task is selected from a list and put it into idle computing resources began to run.

The basic idea of Clustering scheduling algorithm is that if there are infinite numbers of processors, the DAG task graph nodes are taken as a cluster when starting scheduled, and merger all of these clusters without increasing the overall task completion time until there is no cluster can be merged finally.

The basic idea of the scheduling algorithm based on task duplication is to duplicate the precursor mission when the processor is idle to avoid some of data transmission from the precursor task, thus to decrease the gap of waits time for the processor.

Random search technique is mainly driven by a random choice to search for the problem solution. The searching results are better than other algorithms, but its scheduling length is longer, so it is not employed so much.

B. *Configuration Prefetch Scheduling Algorithms*

Dynamically reconfigurable hardware architecture or device is characterized by changing its functionality and connectivity rapidly. The computing time of a system employ dynamic reconfiguration can be divided into work time and preparation time. The real effective operation time is the time required for the module and the communication time between modules. Preparation time is the delay caused by the configuration between the function switching.

Now the bottleneck of a dynamically reconfigurable system is that the preparation time is too long. The preparation time can be shortened by hiding the critical software configuration time in addition to improving the speed of reconfigurable devices.

The basic idea is that to configure the successor node in advance when node in the DAG to be scheduled for execution. And save the configuration node that need to be configured but because of the FPGA configuration port is occupied and can not start immediately by using a configuration wait queue.

VI. EXPERIENMENTAL RESULTS

In order to verify the hardware/software partitioning algorithm presented here and the effectiveness of the configuration scheduling strategy, we perform the following software simulation.

First, we use the TGFF tool (Windows Version) to randomly generate the task flow graph in which 30, 40, 50, 60, 70, 80 nodes are included. Each node contains hardware and software implementation costs, hardware resources area, and other information, reconfiguration time and the area occupied by the specific tasks related to the number of resources.

For the same computing tasks, a reconfigurable hardware implementation is usually 10 times faster than using a microprocessor[23], it is assumed that for each task on the processor, the average execution time of reconfigurable hardware in the average execution time of 10 times. Simulation environment for hardware and software test are Inter 1GHz processor, 512MB RAM, Linux operating system, GNU compiler. And assuming that the target system consists of a single processor and Virtex Ⅱ series xc2v1000 FPGA which has 1280 CLBs. Table1 shows the comparisons between the three algorithms used to get the fitness value.

Table1. The best fitness value comparison of GA, TS and GATS

	30	40	50	60	70	80
TS	0.853600	0.852139	0.853721	0.860228	0.863175	0.864006
GA	0.848012	0.848369	0.842773	0.846343	0.847218	0.845983
GATS	0.922487	0.924224	0.920106	0.914286	0.928246	0.949677

The relationship between the fitness values is shown in Figure5. It is clear that the values get from GATS are greater than that of the genetic algorithm GA and tabu

search algorithm TS. It is shows that the GATS algorithm has the advantage of multi-start point and a strong hill-climbing ability. Although the running time of GATS algorithm is longer than the GA, TS algorithm, the accuracy is high. So the GATS algorithm can be applied in the applications demanding high accuracy.

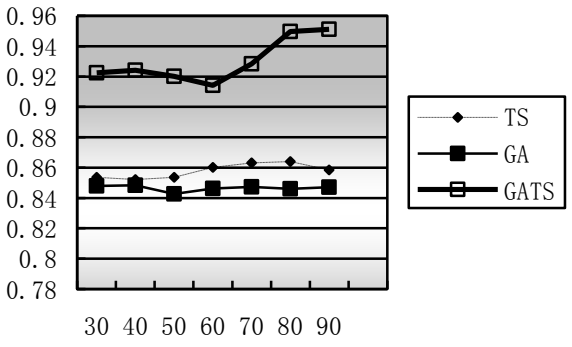

Figure5. Fitness Value/Nodes Curve

Table2 shows the results using these three algorithms in different scale applications, in which S and NS stand for the scheduling strategies with and without configuration prefetch respectively.

Table2 Data comparison of GA,TS and GATS

Task	GA		TS		GATS	
Node	NS	S	NS	S	NS	S
30	2071	1644	2058	1950	2056	1538
40	2845	2241	2835	2743	2809	2076
50	3680	3087	3573	3402	3362	2458
60	4316	3085	4311	4001	4310	2770
70	4944	3324	4806	4522	4776	3014

VII. CONCLUSIONS

This paper focuses on the hardware/software partition technologies in reconfigurable embedded systems. The characteristics of reconfigurable systems and the key issues involved in dynamic reconfigurable technology are analyzed in detail. A reconfigurable hardware architecture model consists of microprocessor, configuration controller, reconfigurable hardware (FPGA), memory, and configuration file memory is proposed. And it gives a directed acyclic graph DAG for reconfigurable embedded systems modeling. After comparing the advantages and disadvantages of GA and TS algorithms, a mixture of GA and TS strategy, called GATS algorithm is proposed. The GATS approach drawing on the strengths of genetic algorithms and tabu search algorithm respectively, and achieved good results.

Task scheduling algorithms based on DAG models such as configuration prefetch, priority-based scheduling algorithm, especially for the CPU + FPGA structure, are proposed. The scheduling algorithms are used to evaluate the system partitioned in GATS algorithm. The results show that it effectively reduces the reconfiguration time and the overall application execution time.

REFERENCES

[1] Parkash S, Parker A C. SOS: Synthesis of Application-Specific Heterogeneous Multiprocessor Systems. Journal of Parallel and Distributed Computing, Vol.16, 1992, pp: 338-351.

[2] J.Henkel and R.Ernst. High-Level Estimation Techniques for Usage in Hardware/Software Co-Design. In IEEE/ACM Proc of Asia and South Pacific Design Automation Conference, Yokohama, Japan, February 1998: 353-360.

[3]ERNST, R, J HENKEL, and T. BENNER. Hardware-Software Co-Synthesis for Micro-Controllers[J], IEEE Design & Test of Computer, 1993, 10(4): 64-75.

[4] Frank Slomka, Matthias Dorfel Ralf Munzenberger, Richard Hofmann. Hardware/Software Codesign and Rapid Prototyping of Embedded Systems. IEEE Design & Test of Computers, 2000, 17(2): 28-38.

[5] P ELES, Z PENG, K KUCHCINSKI, et al. System Level Hardware/Software Partitioning Based on Simulated Annealing and Tabu Search [J]. Design Automation for Embedded Systems, 1997, 2(5): 5-32.

[6] HENKEL J. A Low Power Hardware/Software Partitioning Approach for Core-Based Embedded System[C]. In: Proceedings of the 36th ACM/IEEE Conference on Design Automation Conference, 1999, 122-127.

[7] THEERAYOD WIANGTONG, PETER Y.K CHEUNG, WAYNE LUK. Comparing Three Heuristic Search Methods for Functional Partitioning in Hardware-Software Codesign[J]. Design Automation for Embedded Systems, 2002(6): 425-449.

[8] MICHALIS D GALANIS, ATHANASIOS MILIDONIS, GEORGE THEODORIDIS. A Method for Partitioning Applications in Hybrid Reconfigurable Architectures[J]. Des Automation Embedded System, 2006(10): 27-47.

[9] Liu C，Layland J. Scheduling algorithms for multiprogramming in a hard real-time environment [J]. Journal of the ACM,1973,20(1):4-61.

[10] R. Gupta and G. De Micheli. System-level synthesis using re-programmable components. Proceedings of EDAC, IEEE Press, 1992, 2–7.

[11] Institute of Electrical and Electronics Engineers, New York, USA, IEEE Standard VHDL Language Reference Manual (IEEE Std 1076-1987), 1987.

[12] J. Iyoda. ParTS: A Support Tool to Hardware/Software Partitioning. Master thesis, Federal University of Pernambuco, Brazil, May 2000 (in Portuguese).

[13] L. Silva. An Algebraic Approach Hardware/Software Partitioning. PhD. thesis, Federal University of Pernambuco, Recife, Brazil, July 2000.

[14] C.Steiger, H.Walder and M.Platzner. Operating Systems for Reconfigurable Embedded Platforms: Online Scheduling of Real-Time Tasks[J].IEEE Transactions on Computers, 2004, 53(11): 1393-1407

[15] R.B. Hughes and G. Musgrave. The lambda approach to system verification. Hardware/Software Codesign, Kluwer Academic Publisher, 1996, 427–451.

[16] A.Dehon. Comparing computing machines.Configurable Computing: Technology and Applications of Proc.Of SPIE, 1998, 3526:124-133

[17] J. Iyoda, A. Sampaio, and L. Silva. ParTS: A partitioning transformation system. Proceedings of FM99(World Congress on Formal Methods), Vol. 1709, Lecture Notes in Computer Science, 1999, 1400–1419.

[18] A. Kalavade and E. Lee. The extended partitioning problem: Hardware/software mapping, scheduling and implementation-bin selection. Design Automation for Embedded Systems, Vol. 2, No. 2, 1997, 125–163.

[19] J. Madsen, J. Groge, P.V. Knudsen, M.E. Petersen, and A. Haxthausen. Lycos: The lyngby co-synthesis system. Design Automation of Embedded Systems, Vol. 2, No. 2, 1997, 195–235.

[20] R. Niemann and P. Marwedel. An algorithm for hardware/software partitioning using mixed integer linear

Programming. Design Automation of Embedded Systems, Vol. 2, 1997, No. 2, 165–193.
[21] D. Saha, R.S. Mitra, and A. Basu. Hardware/software partitioning using genetic algorithm. Proceedings. of 10th International Conference on VLSI Design, India, 1997, 155–160.
[22] A. Sampaio. An Algebraic Approach to Compiler Design. Vol. 4 of Algebraic Methodology and Software Technology (AMAST) Series in Computing, World Scientific, 1997.
[23] A.Dehon. Comparing computing machines.Configurable Computing: Technology and Applications of Proc.Of SPIE, 1998, 3526:124-133
[24] L. Silva, A. Sampaio, and G. Jones. Serialising parallel processes in a hardware/software partitioning context. Proceedings of Formal Method Europe (FME) 2001, Vol. 2021, Lecture Notes in Computer Science, 2001, 344–363.
[25] DICK R P, RHODES D L, WOLF W. TGFF: Task graphs for free. Proc. Int. Workshop Hardware/Software Co-design [M]. Mar. 1998: 97–101.

Linhai Cui was born in HeiLongJiang, China, in 1961. He received the B.S. degree in computer science from the Xi'an JiaoTong University, China, in 1983. In 2006, he received the M.S. degree in electrical engineering from the Northwestern Polytechnic University, USA. He is an Associate Professor at the Harbin University of Science and Technology, China.