International Journal of Multimedia and Ubiquitous Engineering
Vol.10, No.4 (2015), pp.251-258
http://dx.doi.org/10.14257/ijmue.2015.10.4.24

Study on a New Heterogeneous Multi-core Hardware/Software Partitioning Method

Lanying Li and Longjuan Chen

The College of Computer Science and Technology, Harbin University of Science and Technology, Harbin 150080,China
lulu08521@sina.com

Abstract

As to the high request of low power consumption for hardware/software partitioning algorithm in multi-core processor structure, based on reducing the system power consumption, the relevant formal model was defined, and the problem was decomposed into two periods: the task division of heterogeneous multi-core platform and low power scheduling. The quantum genetic algorithm was proposed in the paper by taking the advantage of quantum computing combined with the traditional genetic algorithm, and the scheduling method based on critical tasks was used to solve scheduling problems of task after division. Experimental results show that the quantum genetic algorithm increases the diversity of population in the process of task partitioning and the task scheduling algorithm based on the key tasks determines the optimal execution order. The whole algorithm achieves the design goals of reducing the power consumption of the system and lessening the time complexity.

Keywords: *hardware/software partition; heterogeneous multi-core processor; low power consumption; quantum genetic algorithm; critical tasks.*

1. Introduction

In the last few years, with the rapid development of technology and the urgent request of low power consumption in embedded system, low-power system design under heterogeneous multi-core platform has received extensive attention in academia and industry [1]. In the early period, hardware/software partitioning was a two-way partitioning, and the target structure only contained a single processor and a FPGA. Reasonable distribution of the system functions in multiple processor and FPGA is called multiple division [2]. Task scheduling of heterogeneous multi-core processors can be divided into static task scheduling and dynamic task scheduling according to whether it can be dynamically adjusted in the process of scheduling [3]. Unique architecture of heterogeneous multi-core processor makes traditional scheduling algorithms in single-core processor platform no longer suitable for it. For low power consumption of system structure in embedded heterogeneous multi-core processor, the current research direction mainly focus on decomposing the problem into task partitioning and low power scheduling in two stages. The algorithm of reducing power consumption is adopted at each stage, and finally optimal algorithm of two stages will be integrated in order to achieve the system goal of maximum reduction of the power consumption [5]. For heterogeneous multi-core platform task partitioning problem, researchers put forward many solutions, which commonly used ones are dynamic programming method[6]and Lippmann processes [7],geometric method[8],combination method [9], multi-layer classification method [10] and the heuristic algorithm and so on. Among them, the heuristic algorithm is the most widely used. How to coordinate with the heuristic algorithm and scheduling algorithm to reduce the power consumption of the system as a whole becomes the focus of current research. To solve above problems, this paper

ISSN: 1975-0080 IJMUE

presents a quantum genetic algorithm to solve the problem of low power hardware/software partitioning and low power scheduling algorithm based on critical tasks.

2. Hardware/software Partitioning Model

2.1. Target System Structure Model

The essence of hardware/software partitioning is a kind of distribution, at the same time is also a kind of optimization [11]. References [12-13] have proved that under the environment of heterogeneous multi-core processor, task partitioning problem is N-P problem. Reference [14] has proved that no algorithm can adapt to all of the constraints and situations. System model is defined as follows:

Definition 1: Heterogeneous multi-core embedded system structure of target is represented by a directed acyclic graph: *Ga(PE,CL)*. *PE* stands for the collection of processing unit nodes, $PE = \{PE_0, PE_1, PE_2, ..., PE_n\}$, PE_j stands for the j th processing unit, $CL = \{CL_0, CL_1, CL_2, ..., CL_n\}$ on behalf of the communication link set. At least there is one directly connected communication path between any two processing units. The system energy consumption is represented by formula 1:

$$E_p = \sum_{S_m \in PE_S} \sum_{V_i \in V_{S_m}} e_i^{S_m} + \sum_{h_n \in PE_h} \sum_{V_i \in V_{h_n}} e_i^{h_n} + \sum_{C_i \in CL} \sum_{eg_{ij} \in EG_{cl}} e_{ij}^{c_l} \quad (1)$$

PE_S and PE_h stand for software processing unit set and hardware processing unit set respectively. CL stands for communications link set. EG_{cl} stands for the path set from task i to tasks j. S_m, h_n stand for software processing unit and hardware processing unit respectively. C_i stands for the i th communication link, V_i stands for the i th task allocation in a processing unit, eg_{ij} stands for the path from the task i to the task j, $e_i^{S_m}$ stands for the power consumption value in execution of the i th task allocated on S_m, $e_i^{h_n}$ stands for the power consumption value in execution of the i th task allocated on h_n, e_{ij}^{Ci} stands for the consumed power value when selecting the C_i path from all paths from task i to task j. All cost is defined as formula 2:

$$\begin{cases} T_p = \sum_{C_i \in CL} \sum_{eg_{ij} \in EG_{ci}} t_{ij}^{Ci} + \sum_{S_m \in PE_S} \sum_{V_i \in V_{s_m}} t_i^{S_m} \\ + \sum_{h_n \in PE_h} \sum_{V_i \in V_{h_n}} t_i^{h_n} \\ M_P^{S_m} = \sum_{V_i \in V_{S_m}} m_i^{Sm}, m \in [1, \mathrm{a}] \\ A_P^{h_n} = \sum_{V_i \in V_{h_n}} a_i^{h_n}, n \in [1, \mathrm{b}] \end{cases} \quad (2)$$

Heterogeneous multi-core hardware/software partitioning based on quantum genetic algorithm is described below:

Minimize $$E_P \quad (3)$$

Subject to $$\begin{cases} T_P \le T_0 \\ M_P^{S_m} \le M_0^{S_m}, m \in [1,a] \\ A_P^{h_n} \le A_0^{h_n}, n \in [1,b] \end{cases} \quad (4)$$

T_0, $M_0^{S_m}$, $A_0^{h_n}$ stand for processor, memory and the FPGA time limit respectively. Fitness function is defined as formula 5:

$$minf = E_P + \sum_{S_m \in PE_S} P_S max(0, M_P^{S_m} - M_0^{S_m}) + \sum_{h_n \in PE_h} P_h max(0, A_P^{h_n} - A_0^{h_n}) + P_t max(0, T_p - T_0) \quad (5)$$

P_S, P_h and P_t, as larger integers, are used to eliminate over memory limit, the FPGA area limit and execution time limit of the whole system hardware/software partitioning scheme.

2.2. Task Model

The task design of the embedded system is expressed by a DGA [15] (Directed Acyclic Graph).

Definition 2: Attribute of each node in the model is defined as a six-meshes as follows:

$$V_i = < v_i^{es}, v_i^{ts}, v_i^{ms}, v_i^{eh}, v_i^{th}, v_i^{ah} > \quad (6)$$

i stands for task node number, v_i^{es}, v_i^{ts}, v_i^{ms} stand for power consumption of task i in the software processor, time needed for execution and memory respectively. v_i^{eh}, v_i^{th}, v_i^{ah} stand for power consumption of node i in FPGA, time needed for execution and area respectively.

Definition 3: Attribute of each edge in the model is defined as a binary set as follows:

$$eg_{ij} = < eg_{ij}^{tij}, eg_{ij}^{eij} > \quad (7)$$

eg_{ij}^{tij} stands for the time consumption in transmission from task i to task j, eg_{ij}^{eij} stands for the power consumption in the transmission from task i to task j.

3. Quantum Genetic Algorithm

Quantum computing is different from traditional computing in that its natural characteristics of parallelism are able to handle the huge amounts of information quickly, and greatly reduce the processing time in scale complex problems [16]. Combining this advantages of quantum computation with the genetic algorithm produces quantum genetic algorithm QGA (Quantum Genetic Algorithm). Using qubit coding to express chromosome and getting the next generation of group by the quantum gate rotating could effectively avoid the lack of species diversity in the traditional genetic algorithm and avoid the selection pressure successfully.

3.1. Qubits Encoded:

If a system has n quantum bits, then the system will contain 2^n linear superposition of quantum ground state, then its general condition is expressed as:

$$|\theta> = \sum_{i=1}^{s} Ci\,|i> \quad (8)$$

$|i>$ is one of the 2^n ground states, C_i stands for the superposition coefficient of the ground state.

Each individual quantum could be coded by the following quantum coding form:

$$q_j^t = \begin{bmatrix} a_{j11}^t & a_{j12}^t & \dots & a_{j1k}^t & a_{j21}^t & a_{j22}^t & \dots & a_{j2k}^t & \dots & a_{jp1}^t & a_{jp2}^t & \dots & a_{jpk}^t \\ b_{j11}^t & b_{j12}^t & \dots & b_{j1k}^t & b_{j21}^t & b_{j22}^t & \dots & b_{j2k}^t & \dots & b_{jp1}^t & b_{jp2}^t & \dots & b_{jpk}^t \end{bmatrix} \quad (9)$$

q_j^t stands for the t generation of the j individuals, a_{jab}^t , b_{jab}^t stand for b th probability amplitude of a genes in the chromosome. When k is the independent variables of each component, used in quantum bit number, p is the chromosome number of genes.

3.2. Update of Quantum Genetic Operations

The quantum genetic algorithm adjusts the rotation angle of quantum gate, and multiplies the initial qubits probability to complete updating and then to produce new individual qubits. Formula is shown as follows:

$$\begin{bmatrix} a_i^{'} \\ b_i^{'} \end{bmatrix} = \begin{bmatrix} cos(q_i^{'}) & -sin(q_i^{'}) \\ sin(q_i^{'}) & cos(q_i^{'}) \end{bmatrix} * \begin{bmatrix} a_i \\ b_i \end{bmatrix} \quad (10)$$

$\theta_i^{'}$ stands for the variation of θ , a_i and b_i stand for probability amplitude at some point.

3.3. Quantum Genetic Algorithm

Supposed there are p tasks assigned to k processing unit, the quantum genetic algorithm in the chromosomes of each individual consists of p genes, each gene includes m quantum bits, so each gene will be able to stand for a qubit string, which meet the conditions. The specific process is as follows:

(1) Initialize the current iteration number t , produce the population size of p , $Q(t)=\{q_1^t,q_2^t,q_3^t...q_n^t\}$, q_j^t stands for the j individuals of the t generation, as shown in formula 9 chromosome expression form.

(2) Do a measurement for each individual in the population, randomly generated a number of [0, 1], If it is greater than $|a_1|^2$, measurement value is 1, otherwise the value is 0, corresponding to get state: $S(t)=\{x_1^t,x_2^t,x_3^t...x_m^t\}$, x_i^t stands for the binary string which length is m .

(3) According to each x_i^t in $S(t)$ actual distribution of tasks to processing unit, calculate the fitness function value of each scheme.

(4) According to the fitness function value remove t highest in fitness of the individual, assignment plan and record the corresponding fitness function value.

(5) According to quantum gate rotation for the next generation, repeat (2), (3), (4) step operation, to every individual's fitness value and the current optimal scheme of the fitness value is used in the comparison, according to the comparison results to decide whether to update scheme.

(6) If the number of iterations out of the scheduled times, then the end, otherwise repeat step (5) operation.

4. Scheduling Algorithm based on Key Tasks

Task division and task scheduling play important roles in reducing the power consumption of the system to the whole problem. This paper combines with DVS (Dynamic Voltage Scaling) technology to design a low-power scheduling algorithm which is based on the analysis of the key tasks called CT_LPS (Critical Task Analysis Low Power Scheduling Algorithm).

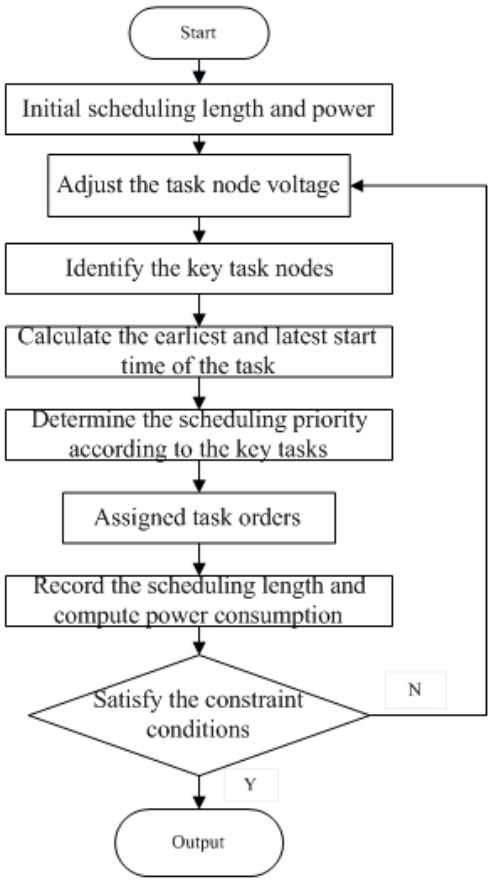

Figure 1. Flow Chart of Scheduling Algorithm based on the Task Scheduling

Definition 4: In figure DAG, the path from the initial node to the end node weights of maximum is called critical path, all the task nodes on the critical path are called critical tasks.

Definition 5: The time period between the initial node to the end node is called scheduling length. The scheduling length of DAG figure depends on the key task.

Definition 6: The relaxation time of task refers to the difference between the deadline of the task and the completion time export nodes.

The increase of relaxation time can increase the potential of reducing power consumption, while the relaxation time is determined by the length of the scheduling, the scheduling length depends on the key tasks, and therefore, the treatment of key tasks will affect the consumption of the system. The DVS technology shows that the properties of the task will be changed under different voltage levels, and the earliest and latest start time of the task needs to be recalculated, producing the possible new task execution order. Based on the facts above, the scheduling algorithm based on key task flow chart shows in figure 1.

5. Experimental Results

To verify the hardware/software partitioning algorithm and the effectiveness of the scheduling strategy based on critical tasks, we adopt TGFF toolkit to generate different nodes of a directed acyclic graph, whose nodes are between 10-70, matching processing units between 3 and 6.

In order to evaluate performance and efficiency of the algorithm, we conduct the contrast test in this paper. In view of the problem about task partitioning, we realize the traditional genetic algorithm-GA and quantum genetic algorithm proposed in this paper. These two algorithms are both the related to the theory of genetic algorithm, so they can show more advantages of QGA. In view of the problem about task scheduling, we realize the scheduling algorithm of CTLPS implemented in this paper and order task analysis of low power scheduling algorithm –OTLPS. We compare QGA_CT_LPS with GA-OT_LPS and find the maximum iteration is 200 and the optimal solution for 10 generations has no obvious improvement will terminate the operation.

This paper mainly evaluates the performance from the two aspects: the effect of energy saving and the time complexity of algorithm. By the formula 11, we can know that the larger the η, the higher consumption rate, and the greater energy saving effect of the algorithm.

$$\eta = (E_0 - E) / E_0 \tag{11}$$

E_0 stands for the energy dissipations of the system at the highest performance runtime, E stands for the energy dissipations of the system after using the algorithm of low power.

We set the starting point of algorithm is run at T_S, end to T_E, and the algorithm operation time can be defined as follows:

$$T_{exe} = T_E - T_S \tag{12}$$

Statistics in Table 1 show the experimental results of GA-OT_LPS and 200 generations evolve QGA-CT_LPS. We can find that QGA_CT_LPS algorithm is superior to GA-OT_LPS in reducing the power consumption and operation time.

Table 1. Performance Comparison between GA-OT_LPS and QGA-CT_LPS

Task Graph	Node/Processing Unit	GA-OT_LPS		QGA-CT_LPS	
		power consumption reduce rate（%）	Execution time（ms）	power consumption reduce rate（%）	Execution time（ms）
TF0	10/3	35.584	680.496	40.296	656.120
TF1	30/5	18.578	1938.104	22.274	1670.269
TF2	50/4	27.614	3383.939	32.099	2754.267
TF3	70/6	41.716	4779.716	51.147	3855.005

From Table 1, we can conclude that GA-OT_LPS can reduce more power consumption than GA - OT LPS under the same system structure because QGA can provides more

solution space for the division of scheme, and the energy saving effect of CT_LPS is better than OT_LPS.

Figure 2 represents the runtime efficiency of the algorithm and we can conclude that QGA-CT_LPS is lower than GA-OT_LPS in the respect of time complexity, and with the increase of nodes numbers, the advantages of QGA-CT_LPS are becoming more obvious because QGA can quickly deal with huge amounts of data in a short time with the characteristics of quantum computing.

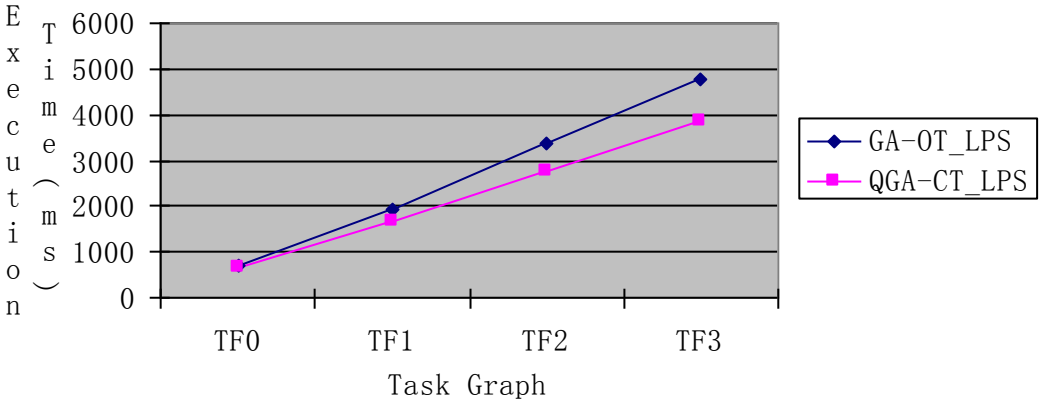

Figure 2. Comparison of Running Time between GA-OT_LPS and QGA-CT_LPS

6. Conclusion

We put forward QGA and the heterogeneous multi-core hardware/software partitioning algorithm for low power consumption based on the critical task scheduling algorithm. First, we take advantage of the parallelism of quantum computing combined with quantum algorithm and traditional genetic algorithm to solve the problem of low power hardware/software partitioning, and realize the evolution update operation by quantum revolving door, reducing the time consumption and ensuring the diversity of the population. Scheduling strategy based on the critical tasks reduces the power consumption of the system significantly. The contrast experiments show that the design of low-power heterogeneous multi-core algorithm effectively reduces the energy consumption of the system and improve the previous problem of high time complexity. However, in the process of theoretical analysis and experimental simulation of the algorithm, we find that the heterogeneous multi-core algorithm proposed in this paper has the potential to improve more. How to determine the optimal solution of distribution and improve the population initialization operation and make the algorithm on the basis of original chromosome converge to the optimal solution more quickly by the hardware/software partitioning algorithm proposed in this paper become important research contents after this article.

Acknowledgements

Supported by Scientific Research Fund of Heilongjiang Provincial Education Department (NO: 2531107).

References

[1] F. Chen, D. Zhang and Z. Wang, “Research of the heterogeneous multi-Core processor architecture design [J]”, Computer Engineering and Science, vol. 33, no. 12, (**2011**), pp. 27-36.
[2] S. Han, “Hardware/Software partitioning based on combination of genetic algorithm and simulated annealing [D]”, Journal of Harbin University of Science and Technology, (**2013**).

[3] O. Beaumont, L. Caterb and J. Ferrante, "Bandwidth-centric allocation of independent tasks on heterogeneous platforms", In International Parallel and Distributed Processing Symposium, (**2002**), pp. 67-72.
[4] J. Li and S. Jin, "Research on static task scheduling strategy based on heterogeneous multi-core processors [J]", Computer Engineering and Design, vol. 34, no. 1, (**2013**), pp. 178-184.
[5] J. Jiang, "Research on embedded software key issues of heterogeneous multi-core processor [D]", Chong Qing University, (**2011**).
[6] Y. Jing, "Different constraints of reconfigurable system hardware/software partitioning algorithm research [D]", Hu Nan University, (**2013**).
[7] A. Pothen, H. Simon and K. Lioum, "Partitioning sparse matrices with eigenvectors of graphs [J]", SIAM Journal of Matrix Ana1ysis and Applications, vol. 11, no. 3, (**1990**), pp. 430—452.
[8] M. Berger and S. Bokhari, "Partitioning strategy for non-uniform problems on Multiprocessors [J]", IEEE Transaction on Computers, vol. C-36, no. 5, (**1987**), pp. 70-580.
[9] A. George and J. Liu, "Computer Solution of Large Sparse Positive Definite Systems [M]", Prentice-Hall Press, (**1981**).
[10] G. Karypis and V. Kumar, "A fast and high quality multilevel seheme for partitioning irregular graphs [J]", SIAM Journal on Scientific Computing, vol. 20, no. 1, (**1998**), pp. 359—392.
[11] X. Zhu and Z. Zhu, "Efficient critical path hardware/software partitioning algorithm of particle swarm[J]", Microelectronics and Computer, vol. 30, no. 4, (**2013**), pp. 160-163,168.
[12] S. Baruah, "Feasibility analysis of preemptive real-time systems upon heterogeneous multiprocessor platforms", In: Proc. of the 25th IEEE Int. Real-Time Systems Symposium, Los Alamitos, CA: IEEE Computer Society Press, (**2004**), pp. 37-46.
[13] S K. Baruah, "Partitioning real-time tasks among heterogeneous multiprocessors", In: Proc. of the 2004 International Conference on Parallel Processing, ICPP, Toronto, Canada, (**2004**), pp. 467-474.
[14] E. Rotem, A. Mendelson and R. Ginosar, "Multiple Clock and Voltage Domains for Chip Multi Processors", In: Proc. of ACM MICRO'09. (**2009**), pp. 459-468.
[15] R. Li, Y. Liu and X. Cheng, "A Survey of task scheduling research progress on multiprocessor", Journal of Computer Research and Development, vol. 45, no. 9, (**2008**), pp. 1620-1629.
[16] Y. Zhang, K. Lu and H. Gao, "Multiprocessor scheduling research progress on a chip system [J]", Chinese Journal of Computers, vol. 36, no. 9, (**2013**), pp.1835-1842.

