International Journal of Smart Home
Vol. 10, No. 4 (2016), pp. 87-94
http://dx.doi.org/10.14257/ijsh.2016.10.4.09

Hardware Implementation of Two-level Scheduling Algorithm in μC/OS-II

Guangwu Zhang[1], Yan Li[1*], Yidong Chen[1], Huaiguo Dong[1], Huanhuan Chi[2], Min Shi[1] and Junfeng Gao[1]

1. Computer Science College, Harbin University of Science and Technology, Harbin, 150080, China;
2. Computing Center of Heilongjiang Nursing College, Harbin, 150036, China
liyan@hrbust.edu.cn

Abstract

Aiming at the problem that μC/OS-II does not support round-robin scheduling of the same priority task, a two-level hybrid task scheduling strategy was proposed. In the first level, by putting the task priority as criterion for task scheduling, a preemptive scheduling of different priority task was implemented. And in the second level, adopting time slice circulars scheduling strategy, round-robin scheduling of same priority task was implemented. The waiting list of tasks was designed by on-chip registers of FPGA and the ready list of tasks was designed by RAM of FPGA, and to implement time slice circulars scheduling, hardware circuit for finding successor of task was designed. The system adopted VHDL, and simulated by the software ISE10.1. The simulation results show that the hardware implementation of the system is well-worked.

Keywords*: Real-time operating system; Time slice circulars scheduling; Hardware task scheduler; Field Programmable Gate Array (FPGA)*

1. Introduction

μC/OS-II is a preemptive [1] real-time kernel of priority-based, it supports a maximum of 64 tasks, 56 tasks of them are available to users. However, the priority of each task must be different in the application [2-4]. Therefore, μC/OS-II does not support the task scheduling with same priority currently. But, some tasks require the task scheduling with same priority in practice [5]. In addition, RTOS kernel is generally the software which added to the application. It increases the cost of ROM and RAM, and the additional burden of the application [6]. For the existing software-based real-time operating system, improvement to the scheduling algorithm can not make it better to improve real-time performance [7-9]. Based on the above problems, this paper modified μC/OS-II kernels firstly, so that it supported preemptive scheduling on priority-based and time slice circulars scheduling with same priority at the same time. Then the hardware was designed for the modified algorithm and the hardware task scheduler was implemented based on the combinational circuits.

2. Improvement of Task Scheduling Algorithm

μC/OS-II kernels adopts the priority to task as the unique identifier [10]. Therefore, to find the ready task with the highest priority is the key of the scheduling algorithm. Ready list is an important data structure of μC/OS-II, which records all the ready tasks in μC/OS-II by the array OSRdyTbl[] and the variable OSRdyGrp. The ready task with the highest priority will be found by searching the ready list.

The basic idea of time slice circulars scheduling algorithms is that all the ready tasks in the system form a queue according to first-come-first-service [11]. The processor is assigned to

ISSN: 1975-4094 IJSH

the first task in the queue for every scheduling, it executes a time slices [12], then the task re-entering the end of the ready queue, waiting for the next scheduling.

The disadvantage of the scheduling management algorithm of μC/OS-II is that the number of task supports is a small number (maximum 64). It doesn't support multiple tasks scheduling with same priority and time slice circulars scheduling algorithm [13]. Aiming at these problems, the algorithm is modified as follows, the method of one task corresponds to one priority was extended to 4 tasks shares the same priority in μC/OS-II. So the number of it supports can up to 256 tasks. The task identification (ID) is the unique identifier instead of priority. The scheduling algorithm is divided into two levels, in the first level, adopting the search algorithm of ready list to find the ready task group which has the highest priority. And in the second level, adopting different scheduling strategies according to the different of the number of ready task which has the highest priority. If there is only one task is ready, the system will output the task ID, and CPU executes this task; if there are two or more ready tasks which have the highest priority, the system will start the clock, and time slice circulars scheduling algorithm will be adopted to schedule ready tasks.

3. Hardware Implementation of Scheduling Algorithm

3.1. Implementation of Task Control Block

Each task has a task control block (TCB) after being created in μC/OS-II. TCB is used to manage the task [14], and it is a data structure. When the CPU usage right of task is deprived, TCB is used to save the task state. The task scheduling kernels implements TCB queue by using on-chip register of FPGA. TCB structure is designed as follows. Each priority corresponds to a set of registers for storing the information of 4 tasks with same priority. PrioStat is the priority state, and it is ready when one of 4 tasks is ready at least. OSTCBCount is the number of ready tasks with same priority. OSTCBId is task identifier, OSTCBStat is task state, OSTCBDly is task waiting latency, OSTCBStkBottom is task stack bottom pointer and OSTCBEventPtr is task event control block pointer. The parameter list is shown as follows:

```
type tcb is record
PrioStat          : std_logic;
OSTCBCount      : std_logic_vector( 2 downto 0);
OSTCBId1          : std_logic_vector( n downto 0);
OSTCBStat1       : std_logic;
OSTCBDly1       : std_logic_vector( n downto 0);
OSTCBStkBottom1  : std_logic_vector( n downto 0);
OSTCBEventPtr 1   : std_logic_vector(n downto 0);
……
OSTCBId4         : std_logic_vector( n downto 0);
OSTCBStat4       : std_logic;
OSTCBDly4        : std_logic_vector( n downto 0);
OSTCBStkBottom1     : std_logic_vector( n downto 0);
OSTCBEventPtr4    : std_logic_vector(n downto 0);
end record;
```

The data structure cuts part parameters of original TCB and keeps the meaning of part parameters of original TCB, see the specific reference [15].

3.2. Implementation of Search Algorithm of Ready List

Ready list enquiry circuit is shown in Figure 1. It is the key of the first level algorithm, the ready task with the highest priority will be found by ready list. Priority decision tables OSUnMapTbl[] uses on-chip RAM of FPGA to store. OSRdyTbl[] and OSRdyGrp, two

variables in the ready list, are implemented by using on-chip register of FPGA, which are used to store ready priority instead of ready task(a ready priority may correspond to a plurality of ready tasks). 64 priority status, PrioStat0, PrioStat1, ... PrioStat63, as the trigger signal of ready list enquiry circuit.

It requires two clock cycles to find the highest ready priority. In the first clock cycle, signal Y is effectively, which means to find the high 3 bit of the highest ready priority, the data of OSRdyGrp input address register Addr. The system reads data from this memory unit and output to the register prio_y. And in the second clock cycle, signal X is effectively, which means to find the low 3 bit of the highest ready priority, prio_y is the selection signal. The system selects the group of the highest ready priority and runs this group data as the address of RAM. Then the data are read from this memory unit and outputted to the low 3 bit of priority register prio_x. The output of ready list is the highest priority of ready task, it is the key parameter of the second level scheduling algorithm.

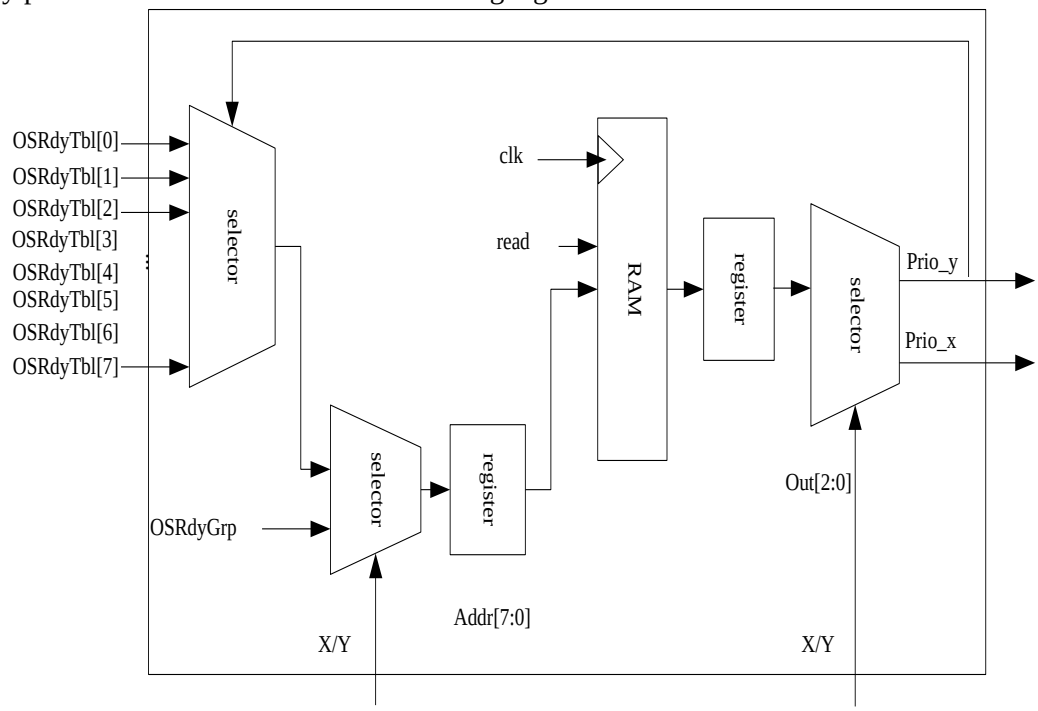

Figure 1. Ready List Enquiry Circuits Diagram

3.3. Implementation of Task Scheduling Algorithm with Same Priority

According to the output of the first level scheduling algorithms, the second level scheduling algorithms is started. If only one task is ready with the highest priority, the system will find the task ID and output parameters which the task saves to CPU register and CPU executes this task. If there are two or more ready tasks which have the highest priority, time slice circulars scheduling algorithms will be adopted to schedule ready tasks.

Time slice circulars scheduling is implemented by combination enquiry circuits in this paper. Hardware implementation of time slice circulars scheduling circuits is shown in Figure 2.

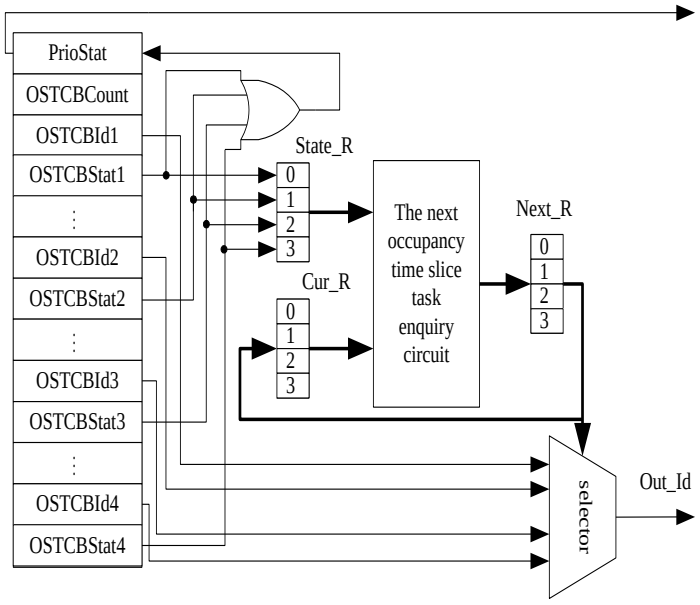

Figure 2. Hardware Implementation Circuit Diagrams of Time Slice Circulars Task Scheduling

State_R is a register with 4bit. If State_R=0110, which means the task 2 and task 3 are ready. Cur_R is the task number register which occupying time slice currently. Cur_R has only one bit is 1. If Cur_R=0010, which means time slice is occupied by the task 3. Next_R is the task that it will occupy time slice next time. If Next_R=0100, which means the task 2 will occupy time slice next time. 4 task status, OSTCBStat1, OSTCBStat2, OSTCBStat3, OSTCBStat4, will sent to priority status register PrioStat after or operation, which will be used by the first level scheduling. Cur_R is the selection signal and the task ID which is occupying time slice will be outputted.

The logic of the next task that occupies time slices enquiry circuits are showed as follows:

$State\_R_i$, Cur_R_i , $i = 0,1,2,3$ are the input variables of circuits, and respectively connect corresponded to the bit of register State_R and Cur_R. $Next\_R_i$, $i = 0,1,2,3$ are the output variables of circuits, and respectively connect corresponded to the bit of register Next_R. M_i, N_i, Z_i, B_i and C_i ($i = 0,1,\cdots 7$)are the intermediate variables.

In order to implement the round-robin search of task, the input variables $State\_R_i$, $i = 0,1,2,3$ are substituted twice, the method of substitution is shown in formula(1).

$$\begin{cases} M_i = State\_R_i & i = 0,1,2,3 \\ M_i = State\_R_{i-4} & i = 4,5,6,7 \end{cases} \quad (1)$$

$Cur\_R_i, i = 0,1,2,3$, the input variables, are the task sequence number which are occupying time slice. It is extended to 8 bits and shown in formula (2).

$$\begin{cases} N_i = Cur\_R_i & i = 0,1,2,3 \\ N_i = 0 & i = 4,5,6,7 \end{cases} \quad (2)$$

The mask circuit is configured by using the task sequence number $N_i, i = 0,1,\cdots 7$ which are extended and occupying time slice. It is shown in formula (3).

$$\begin{cases} Z_i = N_i & i = 0 \\ Z_i = Z_{i-1} + N_i & i = 1,2\cdots 7 \end{cases} \tag{3}$$

The variables M_i that represent task ready state, the variables N_i that represent task sequences number which occupy time slice and the variables Z_i that represent masks are used to configured circuits, so that all the subsequent ready tasks which are occupying time slice currently will be found. The method is shown in formula (4).

$$B_i = (Z_i \oplus N_i)M_i, i = 0,1,\cdots 7 \tag{4}$$

To find the first ready task from all the subsequent ready tasks. C_i are the variables of task sequences number. The method is shown in formula (5).

$$\begin{aligned} C_0 &= B_0 \\ C_1 &= \overline{B_0}B_1 \\ C_2 &= \overline{B_0}\,\overline{B_1}B_2 \\ C_3 &= \overline{B_0}\,\overline{B_1}\,\overline{B_2}B_3 \\ C_4 &= \overline{B_0}\,\overline{B_1}\,\overline{B_2}\,\overline{B_3}B_4 \\ C_5 &= \overline{B_0}\,\overline{B_1}\,\overline{B_2}\,\overline{B_3}\,\overline{B_4}B_5 \\ C_6 &= \overline{B_0}\,\overline{B_1}\,\overline{B_2}\,\overline{B_3}\,\overline{B_4}\,\overline{B_5}B_6 \\ C_7 &= \overline{B_0}\,\overline{B_1}\,\overline{B_2}\,\overline{B_3}\,\overline{B_4}\,\overline{B_5}\,\overline{B_6}B_7 \end{aligned} \tag{5}$$

8 bits sequence number variables C_i are cut to 4 bits sequence number variables and outputted. The method is shown in formula (6).

$$Next\_R_i = C_i \oplus C_{i+4} \quad i = 0,1,2,3 \tag{6}$$

For example, if State_R=0011, Cur_R=0001, then Next_R=0010, which is the next task that time slices scheduling algorithms will be executed.

3.4. Hardware Overall Implementation of Task Scheduling

Task management of μC/OS-II mainly includes create tasks, delete tasks, suspend tasks, resume tasks, query tasks, task scheduling and so on. The task status will be changed when create tasks, delete tasks, suspend tasks and resume tasks, and the change will trigger task scheduling. Hardware overall implementation of task scheduler circuits is shown in Figure 3.

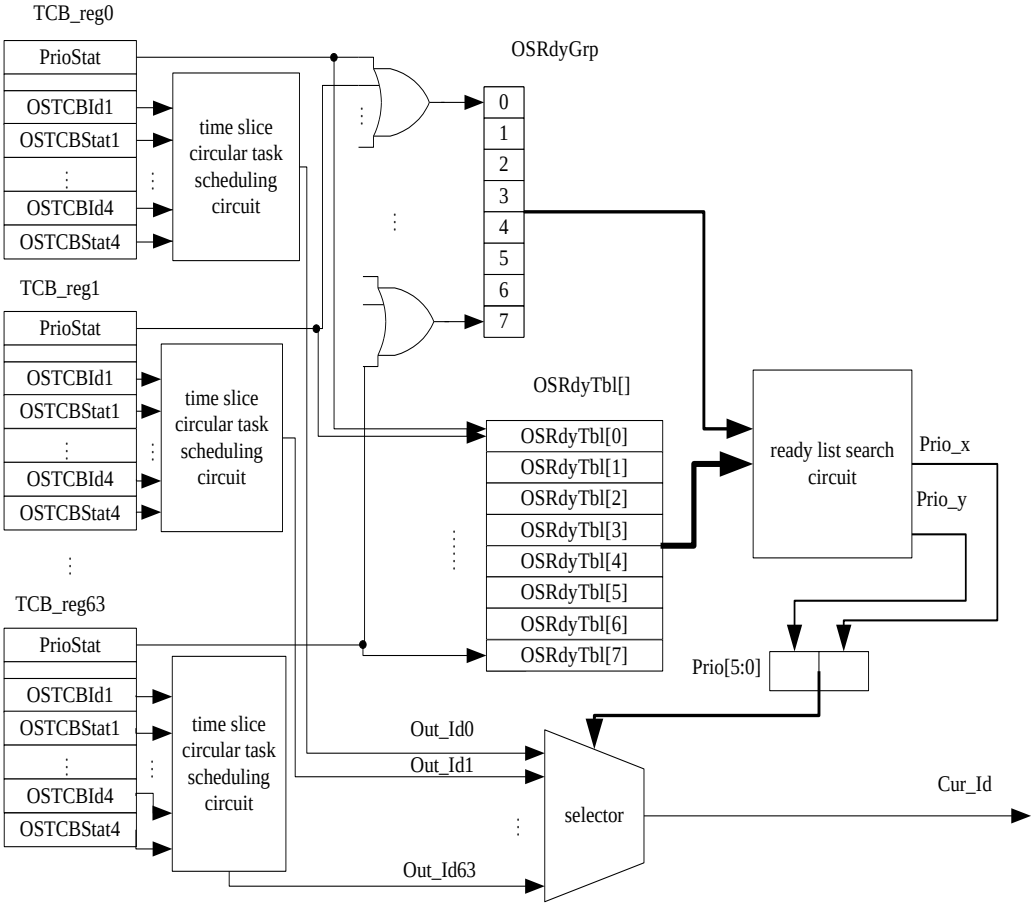

Figure 3. Hardware Overall Implementation of Task Scheduler Circuit Diagrams

In the improved real-time operating system, the task is divided into 64 levels according to the priority. Each priority can create 4 tasks at most, and 4 task states determine the ready state in this priority. The priority is ready when one of 4 tasks is ready at least. The change of 64 priority status triggers the priority ready list, the highest priority ready task Prio will be found by searching ready list, then the second level scheduling will work. The number of ready tasks will be got by searching the state of 4 tasks with this priority OSTCBStat, which will be stored in register OSTCBCount. If OSTCBCount=1, ready task will be searched, the parameters will be sent to CPU register and CPU executes this task. If OSTCBCount>1, which means more than one task with the highest priority is ready, the ready task with same priority enquiry circuits is started and these tasks are executed in the way of time slice circulars.

4. Experimental Results and Analysis

In order to verify the correctness and effectiveness of task scheduling algorithms in this paper, the design adopted VHDL, and synthesized and simulated by ISE 10.1 which is the development software of Xilinx Company. Function simulation of hardware tasks management scheduler is shown in Figure 4.

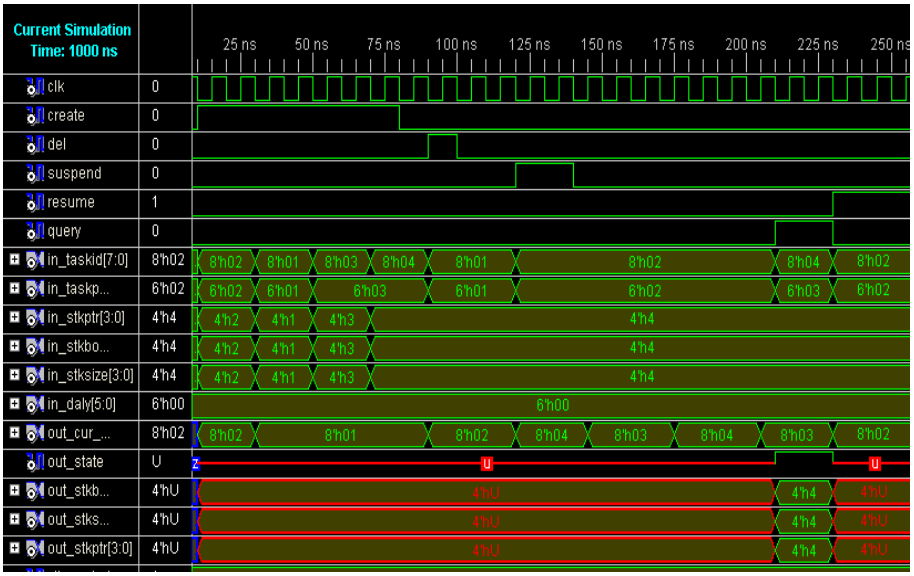

Figure 4. Function Simulation Diagrams of Hardware Tasks Management Scheduler

The simulation results are analyzed as follows:

1) Create the task whose ID is 2 and priority is 2 at 10ns. At this point there is only one task, task 2, so out_cur_taskid is task 2 currently.
2) Create the task whose ID is 1 and priority is 1 at 25ns. The priority to task 1 is higher than the priority to task 2, CPU usage right of task 2 is deprived and task 1 is executed at this time. So out_cur_taskid is task 1 currently.
3) Create the tasks, ID is 3 and priority is 3 at 50ns and ID is 4 and priority is 4 at 70ns. The priority of task 1 is the highest at this time, so out_cur_taskid is also task 1 currently.
4) Delete task 1 at 90ns. The priority of task 2 is the highest at this time, so task 2 is executed.
5) Suspend task 2 at 120ns. Task 2 and task 3 have the same priority 3. Task 3 and task 4 adopt time slice circulars scheduling, which is shown in out_cur_taskid.
6) Request signals are effectively, the input ID is 4, and the output port will output the parameters of task 4 at 210ns. Resume task 2 at 230ns, the priority of task 2 is 2, which is the highest, so out_cur_taskid is task 2 currently.

Suppose there are 6 tasks running in the operating system, the usage of hardware resources in entire system is shown in Table 1. According to Table 1, resources of FPGA and performance meet the need of real-time operating system.

The analysis of the results verifies the correctness and high efficiency of using hardware logic to implement operating system scheduler, which meets the requirement of real-time operating system.

Table 1. The Practical Usage of Hardware in xc4vlx15-12sf363 Platform

Logic Utilization	Used	Available	Utilization
Number of Slices Latches	465	12288	3%
Number of Slices Flip Flops	712	27392	2%
Number of 4 input LUTs	946	12288	7%
Number of bonded IOBs	58	240	24%
Number of BUFGCTRLs	6	32	18%

5. Conclusion

After researching task management and scheduling algorithms of μC/OS-II carefully, the scheduling algorithm was improved, the real-time operating system was changed to the system with task priority scheduling and time slice circulars scheduling by adding time slice circulars scheduling algorithms. Hardware task scheduler of μC/OS-II was implemented by FPGA and TCB was implemented by on-chip register. Hardware implemented of task management kept the correctness of system calls, gave full play to multitasking potential parallelism, reduced processors overhead and the execution time of the system calls. So, hardware scheduler has its value of research and use.

Acknowledgements

This study was supported by the Technology Innovation Talent Research Foundation of Harbin (No. 2013RFXXJ034), the National Natural Science Foundation of China(No.61103149), the Education Department Foundation of Heilongjiang Province(No.12521100) and the Natural Science Foundation of Heilongjiang Province(No. F2015038).

References

[1] W. Zhongkai and Z. Lei, "Research on task scheduling in μC/OS-II", J. Journal of Shandong University of Technology(Natural Science Edition), vol. 23, no. 2, **(2009)**, pp. 30-35
[2] L. Dapeng, "Research on task scheduling algorithm of same priority in μC/OS-II", J. Control Engineering of China, vol. 19, **(2012)**, pp. 218-221.
[3] S. Moon, J. Rexford and K. Shin, "Scalable hardware priority queue architectures for high-speed packet switch", J. IEEE Transactions on Computers, vol. 49, no. 11, **(2000)**, pp. 1215-1227,
[4] P. Lei, H. Zhongdong and M. Hua, "Solving priority inversion and deadlock problems in μC/OS-II", J. Computer Applications and Software, vol. 8, **(2011)**, pp. 200-202.
[5] Z. Hang and L. Xiaowen, "Improvement on task scheduling algorithm of μC/OS-II kernel", J. Journal of Chongqing University of Posts and Telecommunications, vol. 2, no. 3, **(2010)**, pp. 360-364.
[6] L. Kessal, N. Abel and S. M. Karabernoi, "Reconfigurable computing design methodology and hardware tasks scheduling for real-time image processing", J. Real-time Image Processing, no. 3, **(2008)**, pp. 131-147.
[7] H. Shuhui and C. Jian, "Analysis and comparison of several embedded real-time operating system", J. Microcontroller & Embedded Systems, no. 5, **(2007)**, pp. 5-8.
[8] M. VETRONILLE, L. OST, MARCON and C. A. M. RTOS, "Scheduler implementation in hardware and software for real time application", C. Proceedings of the Seventeenth IEEE International Workshop on Rapid System Prototyping. Washington, DC: IEEE Press, **(2006)**, pp. 163-168.
[9] C.Jiahua, S. Hongsheng and W. Baojin, "The design and realization of hardware RTOS", J. Application of Electronic Technique, vol. 34, no. 5, **(2008)**, pp. 34-37.
[10] S. Beibei, "Implementation of μC/OS-II task scheduling algorithm with hardware instruction", J. Microcontroller & Embedded Systems, no. 009, **(2010)**, pp. 5-7.
[11] H. Sai, Z. Haibi and X. Huijun, "A fair dynamic quantum algorithm", J. Journal of Natural Science of Hunan Normal University, vol. 35, no. 5, **(2006)**, pp. 30-36.
[12] G. Fuqiang, Q. Changshuo and Y. Jiyuan, "Extension of time slice circulars scheduling in μC/OS-II kernel", J. Journal Computer Applications, no. 4, **(2009)**, pp.1128-1142.
[13] Z. Bo, W. Shiji and Q. Weidong, "SHUM-UCOS: A real-time operating system for reconfigurable systems using uniform multi-task model", J. Chinese Journal of Computer, vol. 29, no. 2, **(2006)**, pp. 208-218.
[14] W. Nana and G. Bing, "Improvement of Priority Scheduling Algorithm Based on μC/OS-II", J. Computer Technology and Development, vol. 11, **(2011)**, pp. 11-14
[15] J. J. Labrosse, "Embedded real-time operating system UC/OS-II", Shao Beibei translation. (2nd Edition), BeiJing: Beihang University Press, vol. 126, **(2003).**
[16] L. Yan, L. Jie, Y. Xiaohua and L.Jingsong, "The Implementation of Advanced DES Encryption Algorithm Based on FPGA", J. Journal of Harbin University of Science and Technology, no. 5, **(2012),** pp. 17.

