Research on Optimization Real-Time Multi-Process System For Integrated Modular Avionics

Ruina Xu[1, a],Weijie Li[2,b]and Bin Chen[3,c]

[1]Commercial Aircraft Corporation of China, Jinke Road NO.5188, Shanghai, China

[2] Commercial Aircraft Corporation of China, Jinke Road NO.5188, Shanghai, China

[3] Commercial Aircraft Corporation of China, Jinke Road NO.5188, Shanghai, China

[a]xuruina@comac.cc, [b]liweijie@comac.cc, [c]chenbin@comac.cc

Keywords: Integrated Modular Avionics(IMA) system, ARINC653 standard, two-level scheduling, idle time shared partition-level scheduling, multi-task evaluation system.

Abstract. As high real time performance of mission execution is required by avionics system, this paper solves the problem and proposes a modified partition-level scheduling algorithm with shared idle time on the Integrated Module Avionics system based on ARINC653 scheduling scheme. The case system development shows that the algorithm can shorten the average response time of mission and be able to achieve the schedule and control under the requirement of real time with multi-mission input of different states.

Introduction

With the rapid development of computer technology, integrated modular avionics, characterized by modularized architecture and highly interconnection, was presented to adapt to new avionics environment. Meanwhile, several standards have been proposed for IMA,like ARINC 653 ,which is the standard to define a general-purpose APEX（APplication/EXecutive）interface between the Operating system of an avionics computer resource and the application software. this document is intended to define a optimized partition schedule model used for heterogeneous multi-core processors, and to propose a partition scheduling algorithm closest to the optimal solution, according to result from the real-time multiprocessing system probabilistic model. This document is supposed to propose a modified partition-level scheduling algorithm and build a simulation evaluation system for improving resource distribution and partition scheduling.

The Overall Architecture Of Multitask Evaluation System Based On ARINC653

According to ARINC653, the architecture of the ARINC653 IMA multi-task evaluation system is as below:

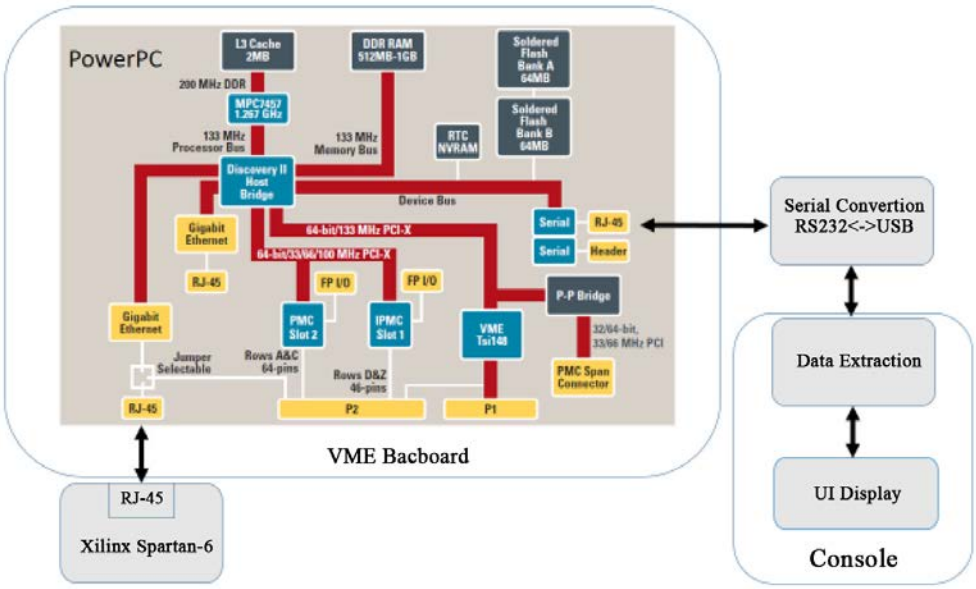

Fig. 1 architecture of IMA multitask evaluation system

The architecture consists of three parts, which are PowerPC main board, hardware accelerators and control station. The PowerPC main control board based on VME runs VxWorks operating system, hardware accelerators applies spartan-6 development board from Xilinx, and the whole system accelerates the partition scheduling algorithm with Ethernet. , control station tracks the output of multi-process scheduling through recording the serial port data provided by PowerPC main board.

The functions residing on the system are partitioned, without affecting one another spatially or temporally, so as to improve system safety and reliability.

ARINC653 Scheduling Scheme

The functional partition is a core concept of ARINC653, which consists of one or more processes, sharing the resource of processors. ARINC653 use the two-level scheduling, which are partition-level scheduling and process-level scheduling. Partition-level scheduling means processor allocates time slice to each partition. Process-level means that processes within a partition get time slice.

Partition-level Scheduling. The main characteristics of partition-level scheduling model are:

A. The scheduling unit is a partition.
B. Partitions have no priority.
C. The scheduling algorithm is predetermined, repetitive with a fixed periodicity, and is configurable by the system integrator only. At least one partition window is allocated to each partition during each cycle, the minimum system master time frame is shown as below.

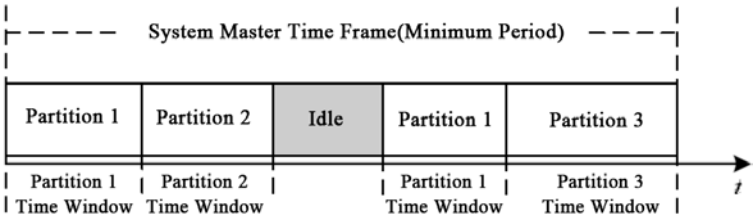

Fig. 2 the minimum system master time frame

Process-level Scheduling. The main characteristics of process-level scheduling model are:

A. A process is a programming unit contained within a partition, the partition level O/S is responsible for managing the individual processes within the partition.
B. each process has a priority
C. the scheduling algorithm is priority preemptive

ARINC653 Scheduling Model

ARINC653 Scheduling Model.ARINC653 utilizes the two-level scheduling, which is partition-level scheduling and the process-level scheduling. We used model as below.

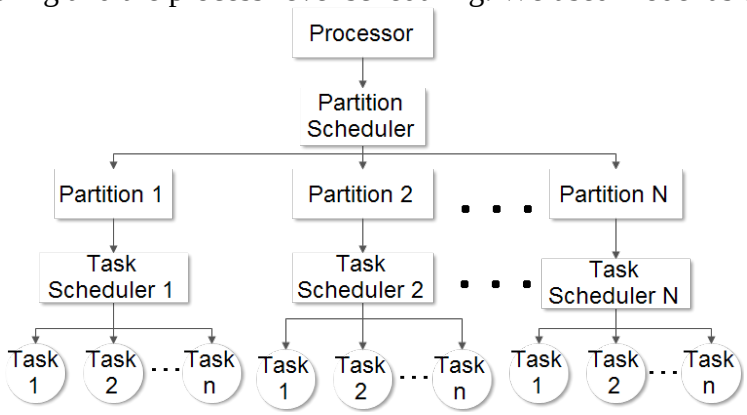

Fig. 3 ARINC653scheduling model

In IMA multitask evaluation system, a process within the partition is described as below:

Assume the system has a process set $T = \{t_1, t_2, \ldots t_n\}$, each process is defined with several parameters, $t_i = (C_i, T_i, D_i, O_i, E_i, W_i, L_i, P)$, where:

C_i: the process execution time.

T_i: the period of the process.

D_i: the deadline of process.

O_i:the time point of first arrival.

E_i: the execution clock of process, and only when the process is in running state, the clock starts ticking.

W_i: the response clock of process Ti, which starts ticking when process arrives, and stops ticking when execution is over.

L_i: current state of processt_i. Each process has four scheduling states, which are suspending, ready, running and error.

P: the partition containing t_i. A partition is defined with several parameters (P_t, P_c, Offset). P_t is the activation period, $P_c$is the amount of processor time given to the partition, Offset is the offset between the first activation time and processor clock.

Process $t_i$state transitions are shown in the following diagram. To simplify the diagram, the process with smaller identify number has higher priority.

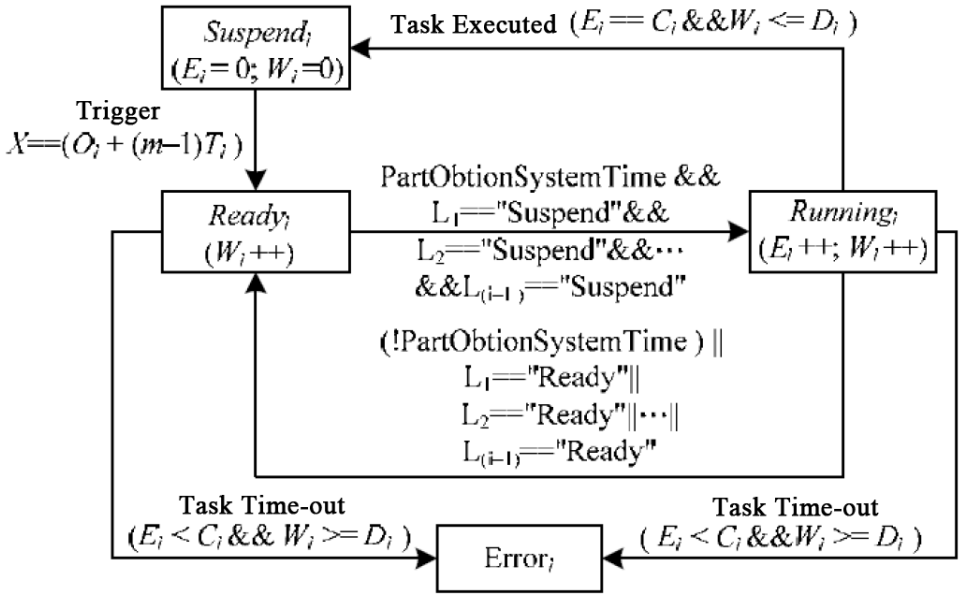

Fig 4.Process state transitions diagram

A. Suspend→Ready

When the process is started, the process state transits into ready.

B. Ready→Running

When the partition containing ti gets the PartObtainSystemTime, and all the other processs ,within the partition, with higher priority are suspended, the process gets resource and is selected for execution.

C. Running→Ready

When the partition does not get the PartObtainSystemTime, or other processs within the partition, with higher priority preempt, the process releases resource and process states transit to ready.

D. Running→Suspending

When the ti finishes execution before deadline , process state transits from running to suspending.

E. Running→Error/Ready→Error

when ti does not finish execution before deadline, the process cannot be scheduled ,neither can the process set.

Schedulability Analysis Algorithm. This algorithmused in IMA multitask evaluation system, can judge wether a process set can be scheduled.The algorithm is described as follows:

Input: *ProcessList[n]*defines the process set within a partition, ti is a process, n is the number of processes.

Output:*isScheuler(true/false)*is the determination result of each process set schedulability,e.g.True or false.Process*WT[n]* is the list of worst-case response time for all processes.

The decision algoritnm follws steps as below:

Step 1: set up all the parameters of each process, including execution time ,response time, processor time, process state. And calculate the simulation time window mulpart for each process set.

Step 2: descending Sort the processes within list by priority, *sort(Processlist[n]).*

Step 3:determin the schedulability when the processor time is *X*.

A. Dtermine the curren process ti schedulability:

1) Update the status L_i of the current task ti, the execution clock Ei and respomce clock W_i

i. When L_i==“Suspend“&&X==Oi+(m-1)Ti., the process arrives, responce clock starts ticking, (L_i=“Ready“, W_i++).

ii. When L_i==“Ready“, the partition containing ti gets *PartObtainSystemTime*, and all the other tasks with higher priority haven’t arrived ($\forall j = 1 \dots i-1, L_j == "Suspend"$), $t_i$starts running, and responce clock starts ticking($L_i = \text{Running}, E_i++,\ W_i++$).

iii. When$L_i == "Running"$, any process with higher priority than t_i ($\exists j = 1 \dots i-1, L_j == "Ready"$), or the partition containing t_i does not get the *PartObtainSystemTime*($!PartObtainSystemTime$), state of t_i transits to“Ready“($L_i = \text{Ready}", W_i++$).

iv. When$L_i == "Error"$,the process can not be scheduled, the process set can not be scheduled($\text{isScheduler} = \text{false}$), move to step 5.

2) If $t_i$competes execution($E_i == C_i \&\& W_i \le D_i$), the clock is cleared, calculte responce time, add the maximum value into list TaskWT[i].

Step 4: the process set is scheduled by dynamic priority based scheduling policy, resort process set *sort(Tasklist[n])*; $X++$, If $X > mulpart$, then skip to step 5, otherwise continue to step 3.

Step 5: determination process ends, if $\text{isScheduler} == \text{true}$, the *isScheduler* and *TaskWT[n]*; otherwise the process set cannot be scheduled.

End Algorithm:

The ARINC653 Process-level scheduling Core Decision Algorithm supports strategies containing static and dynamic priority based preemptive scheduling strategies.

The supported static strategies are:

A. Rate-monotonic Scheduling, RMS.

B. Deadline-monotonic Scheduling, DMS.

C. Fixed-priority Scheduling, FPS.

The supported dynamic strategies are:

A. Earliest Deadline First Scheduling, EDF

ARINC653 Priority Based Scheduling Strategy. The EDF scheduling algorithm is one based on dynamic-priority based preemptive scheme. The process with the earliest deadline is of the highest priority. The priority of processs needs to be adjusted as soon as a new process is ready

The RMS is based on static-priority based preemptive scheme. this algorithm assigns the same static priority to processes with the same period. And processes with shorter period get higher priority, which means processes arrive at higher frequency get higher priority. This algorithm can be used on multi-processors.

The DMS assign resources first to the processes with shorter expected execution time, and allow processes with shorter execution time to preempt.

The LSF assign priority according to the idle time. The idle time is calculated by process deadline minus current time, and time required completing the process. The shorter is the idle time, the higher is the priority of process. The priorities of processes are assigned dynamically according to the calculation point-in-time.

Weakness of Current ARINC653 Partition-level Scheduling.APS algorithm performs partition execution according to a pre-assigned time window rotation, each partition has no priority, the algorithm is simple but inflexible. Each partition can only run in its pre-allotted time window, the idle time window in APS master time frame has not been fully utilized. Because of the time window is fixed , in order to make the process set within one partition be schedulable, the time window assigned to the process set should be greater than or equal to the time required in the worst case. In the result, it is possible to create idle time.

Idle Time Shared Partition-level Scheduling

Idle Time Shared Partition-level Scheduling Algorithm Process-level scheduling is preemptive priority based scheduling, processes with the same priority follow FIFO policy, and we assumed priorities are static. Partition-level scheduling follows principles as follows: partition can occupy more than one time window; partition runs in accordance with pre-assigned time window; idle time window and idle time generated while execution are shared by all the ready partition. The idle time is allocated in accordance with the following principles: idle time is always allocated to the ready partition with the highest priority; as for partition with the same priority, the time is allocated to the partition in ready state with smaller ID number. The priority of partition is only used for allocating idle time.

In practice, considering the time overhead required to switch partition, ifidle time is shorter than time overhead for partition switching, the idle time remain idle. If the idle time is longer than the time overhead, how to find the ready partition with the highest priority and the process within the partition with the highest priority is crucial. We used the priority bitmap algorithm to manage partition and the processes within, in order to improve efficiency.

Schedulability analysis. Consider a single-processor system, the partitions are independent, so as with processes, and ignore the partition switching and process switching overhead. Assume system has two partitions$\prod_1$and$\prod_2$, while$\prod_1 = (\{(0,2),(3,7)\},12)$； $\prod_2 = (\{(2,3),(7,12)\},12)$.

Process T_{11} and T_{12} are processes within partition $\prod_1$, while T_{11} = (c11,p11,d11,X11) , T_{12}=(c12,p12,d12,X12), in which c11=2； p11=12； d11=2； c12=4； p12=12； d12=7; probability distribution of X11, X12 ,T_{11} and T_{12} are generated in the period (0,2);

Process T_{21} and T_{22} are processes within partition $\prod_2$, while T_{21} =(c21,p21,d21,X21) , T_{22}=(c22,p22,d22,X22), in which c21=3； p21=12； d21=9； c22=3； p22=12； d22=12; the probability distribution of X21 , X22 and T_{21}, $T_{22}$are generated in the period (2,3).

The priority number of $\prod_1$ and $\prod_2$ are 8 and 2, the priority number of T_{11}、T_{12}、T_{21} and T_{22} are respectively 2, 8, 2, and 8. The smaller is the priority number, the higher is the priority.

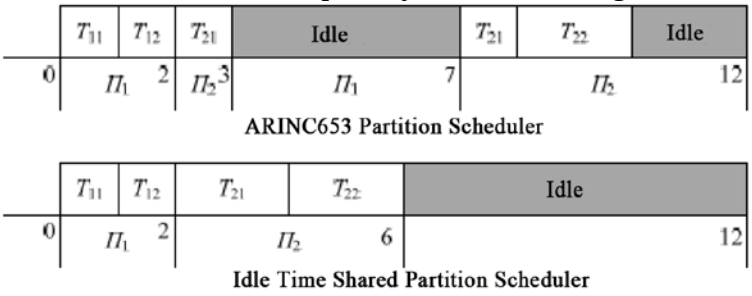

Fig. 5 Idle Time Shared Partition-level Scheduling Algorithm

With the schedulability analysis Algorithm mentioned before, as we can see, the idle time shared partition-level scheduling algorithm can schedule partitions more efficiently. The case system above shows that this algorithm can shorten the average response time of mission and be able to achieve the schedule and control under the requirement of real time with multi-mission input of different states

Debugging and verification of IMA Multi-task evaluation system

Architecture of IMA multitask evaluation system includes the PowerPC main control board and VME backplane, hardware acceleration modules and software console. PowerPC main control board based on VME bus runs VxWorks operating system. By using Xilinx's Spartan-6 as a hardware acceleration module, the system accelerate calculation of the algorithm on PowerPC through Ethernet. According to input data of multi-task set provided by some commercial aircraft, the system operated partition-level scheduling and process-level scheduling for flight parameter during flight. Through different scheduling policies, we got flight data scheduling information, and completed the test at the end of the PowerPC. In the following section, performance evaluation indicators of real-time scheduling algorithm are introduced.

PerformanceEvaluation of Real-time Scheduling. A valuable real-time system must have a good time deterministic, high process throughput and good reliability.

Time Deterministic: running result of process is obtained before deadline.

Process throughput: the number of processing tasks within a time unit. To complete tasks as much as possible within a certain time, system maximizes the use of system resource through rational allocation.

Process acceptance rate: the number of processes scheduled by algorithm/total number of processes had by system.

Resource utilization rate: refer to the real-time process worst-case execution time/real-time process active time, which is used to describe process resource occupancy ratio. The sum of all the process resource occupancy ratio is the system resource utilization rate. The higher resource utilization rate is, the more processes can be scheduled per unit time, the greater the process throughput of the system is.

Schedulability: When scheduling algorithm can guarantee a real-time process set can meet their deadlines for all instances, called the process set is schedulable.

Ethernet-based communication platform verification based on LWIP protocol. After the entire hardware and software compilation was complete, the code was debugged by *XMD*, the debugging tool of *EDK*, the compiled code was downloaded to the *Spartan-6* development board. The communication between *PowerPC* and *FPGA* was verified by cyber capture program and hyper terminal.

Software platform attributes configuration uses non-operation system mode, Standalone mode, calls *xilmfs 1.00a* and *1wipl303.00a* library function, setting compiler option to connect *LwIP* library. The application layer software has been written to realize package transmission and reception, setting *Spartan 6* MAC address, gateway, IP address and subnet mask as follows.

/ the mac address of the board/

```
unsigned    char mac_ethernet_address[]={0x00，OxOa，0x35，0x01，0xEB，0x59}；
 IP4_ADDR(&ipaddr，202，120，39，1)；
 IP4_ADDR(&netmask，255，255，255，0)；
 IP4_ADDR(&gw，202，120，39，45)；
#if 1
IP4_ADDR(&ipaddr，202，120，39，102)；
#else
IP4_ADDR(&ipaddr，3，242，18，54)；
#endif
port=5000；
static unsigned rxpeff_server_running = 0；
static unsigned baseaddr=0x86A08000；
```

After we finished the application layer network data packets software , we created a connection script, set the software code, the program start address, and storage memory, compiler generated executable binary file. We combined the binary file and binary file generated by hardware, to complete the verification.

Summary

Based on the study of ARINC653, we proposed the Idle Time Shared Partition-level Scheduling algorithm. The algorithm provided another solution for partition scheduling. And this algorithm was proved to be feasible. Also this algorithm can be used for IMA multi-task evaluation system, to improve reliability and performance.

References

[1] Qiao Naiqiang, Xu Tao, Gu Qingfan: Research and Improvement of ARINC653 Partition Schedule Algorithm, Chinese Journal of Computer Engineering, 2011, 37(20), p. 249-251

[2] Zhang Yongyue, Yun Lijun, Sun Yu: Implementation of Scheduling Analysis Tool for Avionics System Based on Partition, Chinese Journal of Computer Engineering, 2014, 40(4), p.43-47 (Trans Tech Publications, Switzerland 1987).

[3] Deng Zhong, Jane W S L: Scheduling Real-time Applications in an Open Environment, Proceedings of IEEE Real Time Systems Symposium. San Francisco, USA, IEEE Press, (2000), p.308-319.

[4] Wang Guan, Zhang Yu-ping, Gu Qingfan: Research on IMA Task Scheduling Based on Dynamic Resource Allocation , Avionics Technology, 2013, 44(1) , p.48-53.

[5] Li Wei: Research on Construction and Configuration of High-Reliable Embedded Teal-Time Operating Systems. People's Republic of China. Nanjing University of Aeronautics and Astronautics. 2010.