SCHEDULING TOOL AND ALGORITHM FOR INTEGRATED MODULAR AVIONICS SYSTEMS

Yann-Hang Lee and Daeyoung Kim, CISE Dept. University of Florida, Gainesville, FL

Mohamed Younis and Jeff Zhou, Honeywell International, Columbia, MD

Abstract

As computer and software technologies have advanced, the approach of Integrated Modular Avionics (IMA) has emerged in the field of avionics systems. The IMA approach can dramatically reduce production and maintenance costs and increase reliability of these safety-critical real time systems. The IMA hardware and foundation software must be able to provide guarantees to the application software so that the real-time constraints of all applications are simultaneously satisfied. In addition, each application must be protected from interference by other applications and the operating system software must itself be protected while physically sharing resources such as processors and communication hardware and busses. In other words, an IMA implementation requires that the concepts of spatial and temporal partitioning are provided and guaranteed.

This paper introduces a scheduling tool and its algorithms that can be used to solve the fundamental temporal partitioning problems together with implementation related practical constraints.

Based on the two-level scheduling hierarchy architecture of ARINC IMA standards, we model an IMA system composed of multiple partition servers and channel servers. A partition server models a protected application that may be composed of multiple concurrent tasks. A channel server provides temporally and spatially protected message transmission among applications. The ultimate objective of the tool is to provide schedules for both tasks and messages that provide for robust temporal partitioning.

1 Introduction

The concept of Integrated Modular Avionics (IMA) is one of the major trends in building current state-of-the-art avionics systems. The advancement of computer and software technologies have made it possible to deploy this new concept in real avionics systems. Traditional avionics architectures have been built with a federated method, which lead to numerous equipment boxes, one for each subsystem. This federated approach is a convenient and stand-alone reliable, but it is very expensive solution. In IMA architecture, all these formerly separated sub-systems can be put into one multiprocessor system while sharing computing resources, power, hardware communication resources, software libraries, tools, displays, and information like data bases.

Since IMA approach allows multiple applications of different critical levels share common computing resources, it is important to keep individual application away from potential interference. Both temporal and spatial partitioning principles are the key ideas to protect integrated applications and system resources. Temporal partitioning guarantees that an application or communication server has temporal exclusive access to its pre-allocated resources. With guaranteed pre-scheduled temporal partitioning, an application can meet their timing requirements. Spatial partitioning guarantees that an application has exclusive control over its own data and state information. With spatial partitioning, an application can be protected from any erroneous behaviors of other applications while sharing same physical resources.

The main concern of the paper is temporal partitioning for which we must schedule shared

0-7803-6395-7/00/$10.00 © 2000 IEEE

resources while guaranteeing timing constraints of all applications. Spatial partitioning can be implemented by relatively easier mechanism such as memory management unit of the processor.

Following ARINC IMA standards [5,6,7], we modeled IMA system such that it is composed of a number of communicative applications on multiprocessors that are connected by fault tolerant time division multiplexing (TDM) bus. In order to schedule applications that share processors and fault tolerant TDM bus, we developed two-level scheduling algorithm [1,2,3]. The approach is based on a fixed-priority and cyclic two-level scheduling algorithm for tasks and messages. In the processor execution model, all tasks are assigned to the specific partition server and allocated feasible fixed priorities within a partition server. A task is preemptible by higher priority tasks in the same partition. Then partitions are scheduled according to a cyclic time-based manner while guaranteeing distance constraint. Thus, the execution time for each partition is pre-allocated and no temporal interference can exist between partitions. In the case of message scheduling, each message is assigned to shared channel server and also preemptible by higher priority messages of the same channel. Channels, then, are allocated with a sequence of slots every communication frame and have enough bandwidth to facilitate message communication.

Considering real systems, above fundamental scheduling algorithm is not sufficient to solve a variety of practical constraints. So we have extract l six additional practical constraints from real hardware platform which is also in the development stage. To solve these practical constraints, we have developed corresponding scheduling algorithm modules that can be dynamically plug into basic two-level scheduling algorithm. These six constraints are replicated partition scheduling, incremental changing, time tick based scheduling, fixed bus major frame size scheduling, fixed message size scheduling, and fault tolerant cyclic allocation for clock synchronization.

We have developed a scheduling tool on Windows NT platform with user-friendly graphical interface. The tool provides fundamental two-level scheduling algorithm and six additional algorithms for solving practical constraints.

In the following, we discuss general IMA system architecture and modeling efforts in section 2 and introduce a scheduling tool in section 3. We describe basic two-level scheduling algorithm in section 4. We briefly discuss scheduling algorithms for six additional constraints in section 5.

2 IMA System Architecture &Model

ARINC is a standardization body in the area of avionics systems. They publish a series of recommendations for IMA such as ARINC 651, ARINC 653, and ARINC 659, etc. Therefore, it is reasonable to follow their standards to build scheduling model of the IMA system. According to the standards, the IMA system is composed of several computer cabinets that are interconnected by ARINC 629 global data bus. Inside the cabinet, they use ARINC 659, fault tolerant time division multiplexing bus, to interconnect hardware modules such as processing, I/O, and memory modules. In this paper, we only concentrate on intra-cabinet scheduling. We depict the model of IMA intra-cabinet architecture in Figure 1.

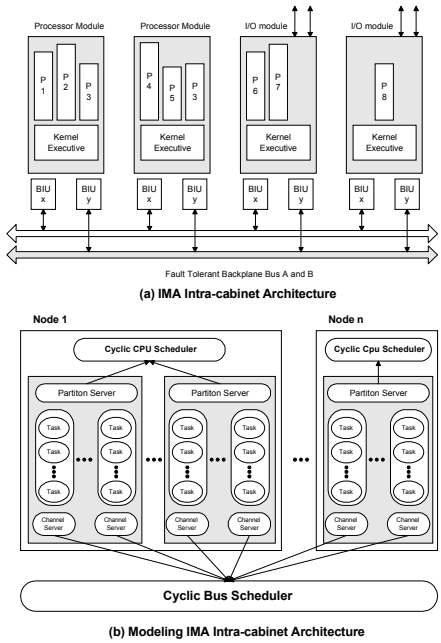

Figure 1. IMA intra-cabinet architecture model

Figure 1-(a) shows the IMA intra-cabinet architecture. Based on Figure 1-(a), we have built IMA scheduling model as shown in Figure 1-(b). In the heart of each processor and I/O module, there is a kernel executive that is responsible for scheduling multiple partitions, denoted by P_i, while guaranteeing temporal and spatial partitioning. A

partition is an independent application that is composed of multiple concurrent tasks. Usually in each partition, there is a partition executive which schedules its own tasks according to their scheduling policy, fixed priority based preemptive scheduling. The bus uses table driven proportional access (TDPA) protocol to ensure temporal and spatial partitioning concepts. It implies that every bus interface unit (BIU) has a transmit/receive table with which the unit can recognize the exact timing of transmission/reception operation and the size of messages.

We show two examples of IMA system. Boeing 777 has equipped with Honeywell's AIMS(Airplane Information Management System) under the flight deck. An AIMS consists two integrated avionics cabinets, one is primary and the other is secondary for fault tolerance. In each cabinet, there are four processing modules and four I/O modules interconnected by TDM SAFEbus (commercial version of ARINC659 bus). They integrate primary airplane functions such as flight management, displays, navigation, central maintenance, airplane condition monitoring, flight deck communication, thrust management, engine data interface, digital flight data processing, and data conversion gateway into only one cabinet. Supported by strong partitioning concepts, critical applications such as thrust management can not be interfered by non-critical central maintenance application. Another example is Honeywell's Integrated Hazard Avoidance System (IHAS) which will integrate their Enhanced Ground Proximity Warning System (EGPWS), Traffic Alert and Collision Avoidance System (TCAS) and Weather Radar System in a single cabinet.

In our task model, each task is composed of two consecutive jobs, processor execution and message transmission. To schedule tasks concurrently, including both processor execution and message transmission, we decompose the system into two scheduling entities, partition and channel. A partition represents an application composed of its interacting tasks. A channel then serves the set of messages originated from a partition. Therefore, in terms of processor scheduling, there are two levels of servers, a cyclic processor scheduler and several priority driven partition server in each processor and I/O module. The cyclic processor scheduler corresponds to kernel executive and priority driven partition server resides in partition executive of each partition. In the same way, there are two levels of servers, cyclic bus scheduler and priority driven channel server in viewpoint of bus scheduling. Within a partition, a set of channel servers serve communication requests from tasks that share a common channel server. There is only one cyclic bus server in a cabinet. Multiple channel servers can exist in each processor and I/O module.

The two-level scheduling is accomplished by first finding feasible assignment of fixed-priorities to all tasks and messages within its own partition server and channel server respectively and concurrently searching for the optimal number of channel servers together with a feasible assignment of messages to channel servers. Second, it finds feasible capacities for the partition and channel servers and computes a cyclic schedule while guaranteeing distance constraints.

Each task has a period (T_k), worst-case execution time (C_k), deadline (D_k), and message transmission size (M_k) as input parameters. It is also required to find optimal split of D_k into message deadline (MD_k) and computation deadline (CD_k) while strong partitioning scheduling. We show the abstract task model in Figure 2. The decomposition of deadline influences capacity requirements for tasks and messages. We can observe a tradeoff that a loose message deadline implies a less amount of bandwidth, but the task has to meet a much tighter computation deadline.

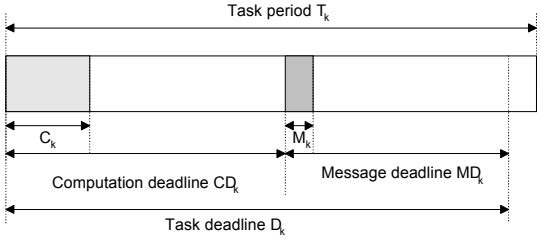

Figure 2. Abstract Task Model

3 Scheduling Tool Suite

We have developed GUI-based integrated strong partitioning scheduling tool which implements all previously described scheduling algorithms for IMA system on Windows NT platform. Figure 3. shows the role of the tool in developing IMA system. For guaranteeing timing requirements of all integrated applications in the

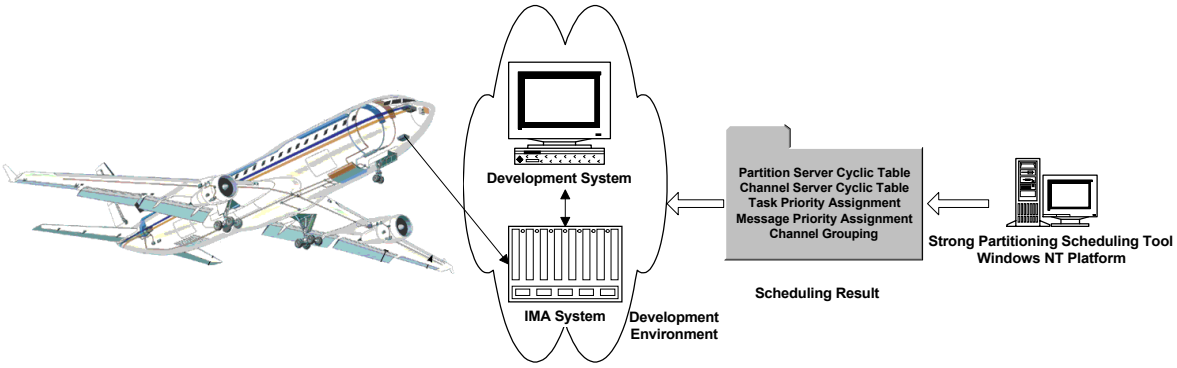

Figure 3. Usage of scheduling tool in avionics system design

IMA system, system integrators need information such as partition server cyclic scheduling table in each processor, task priority assignment in each partition server, message to channel server assignment, channel server cyclic scheduling table, and message priority assignment in each channel server. The strong partitioning scheduling tool provides these feasible schedules in system integration stage. Practically, these scheduled results are put into such as schedule table of kernel executive, and transmission/reception table of bus interface unit in Figure 1.

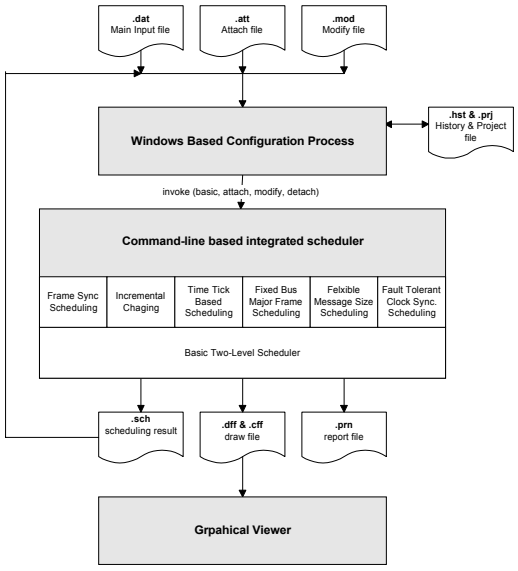

Figure 4. The structure of scheduling tool

The tool provides convenient user interface functions such as managing project, editing files, performing scheduling algorithms, and browsing scheduling results. Figure 4. shows the simplified structure of the tool which composed of three main components, windows based configuration process, command-line based integrated scheduler, and graphical viewer. The windows based configuration-process-module is a top level user interaction environment. It provides graphical user interface with which user can configure the target system, input application information and scheduling options. In order to track the evolution of the system, it also provides project and history management. Figure 5. shows the snapshot of the system configuration dialog box with top-level screen as a background. The command-line based integrated scheduler is responsible to execute our scheduling algorithms with given configuration and inputs either in command line or in integrated environment. A detailed result of scheduling is stored in *.prn file. Additionally, user can view the scheduled timing diagram of processor and bus allocation with graphical viewer.

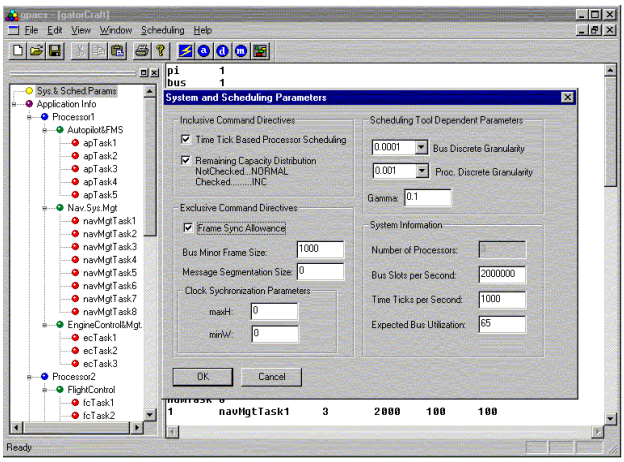

Figure 5. System and scheduling parameters

The scheduling algorithms for solving six additional practical constraints are implemented as plug-in modules of the basic two-level scheduling algorithm. So any combination of six constraints can be easily applied to the application. The basic two-level scheduling algorithm and six additional algorithms will be explained in next sections. Also more detailed information can be found in [2].

4 Basic Two-Level Scheduling Algorithm

The objective of the basic two-level scheduling algorithm is to find feasible cyclic schedules for partition and channel servers. With proper capacity allocation and frequent invocation at each server, the combined delays of task execution and message transmission are bounded by the task deadlines. In Figure 6, we show the overall algorithm which first applies a heuristic deadline decomposition to divide the problem into two parts: partition scheduling and channel scheduling. If either one cannot be done successfully, the approach iterates with a modified deadline decomposition.

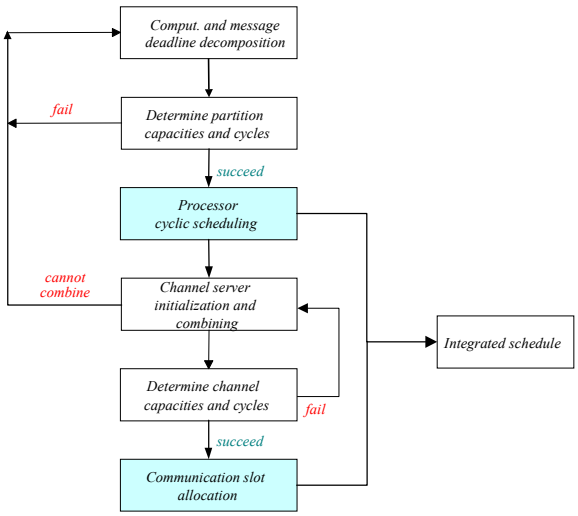

Figure 6. Basic Two-level Scheduling Algorithm

First we consider scheduling periodic partition server, S_k, which schedules the tasks of partition P_k according to a fixed priority preemptive scheduling algorithm. At the system level, the partition server S_k is cyclically scheduled with a fixed partition cycle, η_k. For every partition cycle, the server can execute the task in partition P_k during an interval of $\alpha_k\eta_k$ where $\alpha_k \le 1$. For the remaining interval of $(1-\alpha_k)\eta_k$, the server is blocked. It is our objective to find a pair of parameters α_k and η_k such that the tasks of P_k meet their deadlines when P_k is running by the server S_k.

Suppose that there are n tasks in partition server S_k listed in priority order such that $\tau_1 < \tau_2 < \tau_3 < \ldots < \tau_n$ where τ_1 has the highest priority and τ_n the lowest. The highest priority is given to the task with shortest computation deadline. In order to evaluate the schedulability of the partition server, S_k, we may consider that S_k is executed at a dedicated processor of capacity α_k. Based on the necessary and sufficient condition of schedulability in deadline based rate monotonic analysis, task τ_i is schedulable if there exists a $t \in H_i = \{CD_i \cup lT_j \mid j=1,2,\ldots,i-1;\ l=1,2,\ldots,\lfloor CD_i/T_j \rfloor\}$ such that:

$$W_i(\alpha_k, t) = \sum_{j=1}^{i} \frac{C_j}{\alpha_k} \left\lceil \frac{t}{T_j} \right\rceil \le t$$

The expression $W_i(\alpha_k, t)$ indicates the worst cumulative execution time demand on the processor made by the tasks with a priority higher than or equal to τ_i during the interval $[0,t]$. We now define $B_i(\alpha_k) = \max_{t \in Hi} \{t - W_i(\alpha_k, t)\}$ and $B_0(\alpha_k) = \min_{i=1,2,..n} B_i(\alpha_k)$. Note that, when τ_i is schedulable, $B_i(\alpha_k)$ represent the total period in the interval $[0, t]$ that the processor is not running any tasks with a priority higher than or equal to that of τ_i in the partition server. It is equivalent to the level-i inactivity period in the interval $[0, t]$.

By comparing the task executions at server S_k and at a dedicated processor of capacity α_k, we can obtain the following theorem.

Theorem 1. *The partition* P_k *is schedulable at server* S_k *if* P_k *is schedulable at a dedicated processor of capacity* α_k, *and* $\eta_k \le B_0(\alpha_k)/(1-\alpha_k)$

(Proof) reference [1].

Channel server, G_k, serves its allocated messages according to a fixed priority preemptive scheduling methods. At the system level, the channel server G_k will be allocated certain amounts of slots in a fixed slot period. We denote this period as the distance, μ_k. For every distance, the channel

server can provide bandwidth to the messages in channel server G_k by total number of slots of $\beta_k\mu_k$ where $\beta_k \le 1$. For the remaining slots of $(1-\beta_k)\mu_k$, the server is blocked. With similar approach to processor scheduling, we can find a pair of parameters β_k and μ_k such that the messages meet their deadlines.

The heuristic deadline decomposition algorithm tries to decompose task deadline into computation and message deadline by searching sub-optimal value f for all tasks in following equation.

$$\text{Message Deadline, } MD_k = (D_k \frac{F_k}{C_k + F_k})f$$

$$\text{Computation Deadline, } CD_k = D_k - MD_k$$

The channel combining algorithm tries to assign and combine messages for channel servers to reduce bandwidth requirement.

As a final step, the cyclic scheduling algorithm allocates processor and bus capacities to all partition and channel servers. Since processor and bus cyclic scheduling are similar, we only show the algorithm for bus cyclic scheduling. Let feasible bus bandwidth capacity allocation set for all channel servers be $(\beta_1, \mu_1), (\beta_2, \mu_2), \ldots, (\beta_n,\mu_n)$. It is already sorted by non-decreasing order of μ_k. If μ_i is equal to μ_j where $i < j$, then β_i is less than or equal to β_j. Using distance constraint scheduler, the set will be transformed to harmonic set in which m_i divides m_j where $i < j$. Given a base partition cycle μ, find a period m_i for each μ_i that satisfies

$$m_i = \mu * 2^j \le \mu_i < \mu * 2^{j+1} = 2*m_i,$$

for some integer $j \ge 0$, i.e. $m_i = \mu * 2^{\lfloor \log(\mu i/\mu) \rfloor}$.

To find optimal base μ in the sense of utilization, it tries all candidates of μ, $\mu_1/2 < \mu \le \mu_1$, and computes total bandwidth requirement $\psi(\mu)$. To obtain $\psi(\mu)$, it first transforms set of μ_i to set of m_i based on corresponding μ and then find least capacity requirement, β_k^h, for channel cycle m_i from channel cycle vs. bus capacity graph. Then we can compute actual bandwidth requirement $\psi(\mu)$ by summing all β_k^h. We choose μ which has minimum $\psi(\mu)$. Finally with optimal μ, we can obtain harmonic sets of (β_k^h, m_k).

Figure 7. shows the fixed cyclic bus slot allocation example guaranteeing distance constraint for final harmonic set, (β_k^h, m_k), of *A(0.1,10), B(0.2,10), C(0.1,20), D(0.2,20), E(0.1,40),* and *F(0.3,40)*. We use maximum channel cycle 40 as a major frame size. This bus slot allocator is also easily implemented by priority based scheduler either on-line or off-line.

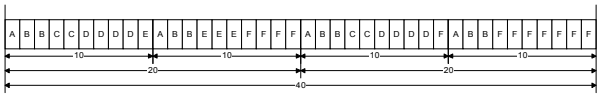

Figure 7. Example of cyclic scheduling

5 Solving Practical Constraints

To achieve reliability and ease of maintenance or due to the limit of the hardware and software, we face practical constraints while scheduling the integrated real time system. In this section, we describe these constraints and show how to solve these problems. Due to the space limits, we briefly discuss the topics. More detailed information can be found in [2].

Replicated Partition Scheduling - To achieve fault tolerance, safety-critical real time systems usually adopt redundancy mechanism by replicating application on multiple processors. It is required that the replicated partition instances in the same group must be guaranteed to meet the same real time constraints while executing on different processors. In the processor execution model, each replicated partition instance may be scheduled differently because they can be run in different node configuration. However, in the case of channel scheduling, all instances must be scheduled same because they have to share one global channel server.

The overall scheduling algorithm takes two steps, replicated partition scheduling and global scheduling. Since replicated partitions are shared by multiple nodes, they have to be scheduled before non-replicated partition scheduling.

Incremental Changing Scheduling - Since the expected life-time of aircraft is several tens of years, it is highly expected to attach, detach and modify the application components of an existent system in the lifetime of the system. In the current practice and FAA requirement, validation and certification processes for the whole system must be

done after each change. Incremental changing algorithms are to seek a minimal modification of the previous schedule when a subsystem is changed. According to the implemented algorithms, we just need to re-allocate the resources for either modified partitions or new partitions only. We still can re-use the original feasible schedule for unaffected partition and channel servers such that they can have unchanged processor capacity, channel allocation, channel bandwidth, major and minor periods. Thus, we can avoid significant system re-design efforts and reduce the costs for re-certification.

Time Tick Based Scheduling – The basic two-level scheduling algorithm assumes high-precision real clock driven method for processor capacity allocation. It means that kernel executive should have capability to cyclic-schedule partitions with period and execution time of real number. For example, the basic two-level scheduling algorithm is allowed to allocate execution time 2.357 (ms) in the period 10 (ms) for a partition. But, most practical systems like Honeywell's GPACS - our target system, uses time tick method with fixed resolution to allocate processor time to each partition. Similar to message scheduling algorithm that uses fixed slot as a scheduling unit, processor scheduling algorithm must be modified to allocate integral number of time ticks to each partition. The disadvantage of time tick based scheduling is that processor utilization is poorer than high-precision real clock driven method because it has to allocate fixed size time ticks, a result of ceiling real number.

Fixed Bus Major Frame Size Scheduling - In original scheduling algorithm, it tries to find optimal major frame size for TDM bus with given channel bandwidth requirement. But, in real world, it is also needed to schedule channels with fixed major frame size. This is because major frame size may be tightly coupled to bus hardware design in some situation. In fixed bus major frame size scheduling, it is limited to choose the base minor frame size, denoted as μ in previous section, in cyclic scheduling. The algorithm requires that the fixed major frame size must be multiples of the base minor frame size.

Fixed Message Size Scheduling - In original message scheduling algorithm, we assume that the unit of TDM time slot (32bit data in GPACS) is a preemption unit for message scheduling. It means that a higher priority message does not wait longer than one time slot unit. In general, the smaller preemption unit, the higher schedulability we can achieve. But, in real situation, it is demanded that the size of preemption unit used by message scheduler should be bigger enough to compensate the slower speed of the scheduler than that of TDM bus. It is also improper to permit unlimited size of preemption unit because it can cause higher priority message should wait the completion of lower priority message too long.

Fixed message size scheduling algorithm will be given parameter MSIZE, fixed message size, additionally. The difference from original schedulability condition is caused from that the higher priority message should wait for MSIZE time at maximum to allow former dispatched lower priority message be transmitted.

Fault Tolerant Cyclic Allocation For Clock Synchronization – ARINC 659 fault tolerant TDM bus is designed for safety critical real time systems that require very high reliability. So it is equipped with four redundant data serial buses and four clock lines for fault tolerance. And clock synchronization is achieved by bit- and frame-level synchronization messages in a distributed way. In the tool, we implemented fault tolerant cyclic allocation algorithm that can be used in simpler distributed clock synchronization environment.

We assume that in our target distributed clock synchronization environment, when one of bus controllers gets bus mastership and transmits either real or idle data, it also distributes reference clock to all other bus controllers on the same bus simultaneously. Then other bus controllers try to synchronize their own clock to current reference clock. This approach might cause global system clock failure if current bus master controller goes to fail for longer than certain threshold time. Our new algorithm ensures that bus master can not be allocated longer than this threshold. Also the failed node is prevented from re-gaining bus mastership in another pre-determined threshold time.

We use the two parameters to specify the reliability of distributed clock synchronization

mechanism and introduce Idle Clock Synchronization Channel Server (ICSCS). Every bus controller in a node has at most one ICSCS. The two parameters are *MaxH* and *MinW* that are decided by the reliability of underlying hardware. The unit of two parameters is TDM time slot. The Maximum Bus Hold Time, *MaxH*, indicates that clock synchronization mechanism can maintain synchronization for a *MaxH* time during the failure of reference clock. So the algorithm should guarantee that every bus controller should not monopolize the bus more than *MaxH* number of consecutive slots. The Minimum Bus Wait Time, *MinW*, means that every bus controller should wait at least *MinW* time to get back the bus mastership after yielding their bus mastership to others. The function of ICSCS is to distribute reference clock to the bus with idle data when there is no scheduled bus activity at that time interval.

6 Conclusion

We have developed scheduling tool for Integrated Modular Avionics Systems that require implementation of strong partitioning concepts. We modeled the system based on integrated avionics standard, ARINC IMA architecture. We have developed the two-level hierarchical scheduling, lower cyclic scheduling satisfying distance constraint to schedule partition and channel servers and higher priority driven scheduling to schedule tasks and messages within each server. The contribution of the algorithm is that it can efficiently distribute processor and bus capacity to all applications properly so that their timing constraints are guaranteed while sharing resources. For the purpose of supporting fault tolerance, we added scheduling algorithms for replicated partitions and fault tolerant cyclic allocation for clock synchronization. Added incremental changing algorithms are useful for efficient lifetime maintenance of the IMA system. The tool also solves the problems due to hardware and software limits using time tick based scheduling, fixed bus major frame size scheduling, and fixed message size scheduling.

ACKNOWLEDGEMENT

The authors wish to thank James McElroy, Principal Engineer at Honeywell Aerospace Electronic Systems, for his participation, project perspective and practical advice during the course of this research.

References

[1] Y.H. Lee, D. Kim, M. Younis, and J. Zhou, "Partition Scheduling in APEX Runtime Environment for Embedded Avionics Software," *Proc. IEEE Real-Time Computing Systems and Applications*, pp. 103-109, Oct. 1998.

[2] Y.H. Lee and D. Kim, "Partition and Message Scheduling in GPACS architecture," *Final Research Report to Allied Signal Inc.,* Mar. 1999.

[3] Y.H. Lee, D. Kim, M. Younis, J. Zhou, and J. McElroy, "Resource Scheduling in Dependable Integrated Modular Avionics," *Proc. IEEE International Conference on Dependable Systems and Networks*, Jun. 2000.

[4] C.L. Liu, J.W. Layland, "Scheduling Algorithms for Multiprogramming in a Hard Real-Time Envrionment," *J. Assoc. Comput. Mach.,* vol. 20, No. 1, pp.46-61, 1973.

[5] *Design Guide for Integrated Modular Avionics*, ARNIC Report 651, Aeronautical Radio Inc., Annapolis, MD, Nov. 1991.

[6] *Avionics Application Software Standard Interface*, ARNIC Report 653, Aeronautical Radio Inc., Annapolis, MD, Jan. 1997.

[7] *Backplane Data Bus*, ARNIC Specification 659, Aeronautical Radio Inc., Annapolis, MD, Dec. 1993.

[8] C.-C. Han, C.-J. Hou and K.-J. Lin, "Distance-Constrained Scheduling and its Applications to Real-time Systems," *IEEE Trans. on Computers*, vol. 45., pp. 814-826, July 1996.

[9] N. Audsley, and A. Wellings, "Analyzing APEX Applications," *Proc. IEEE Real-Time Systems Symposium,* pp. 39-44, Dec. 1996.

[10] T. Carpenter, "Avionics Integration for CNS/ATM," *IEEE Computer*, pp. 124-126, Dec. 1998.