

Schedule verification and optimization for partitioned operating systems

Dr. Kai Richter, Dr. Marek Jersak

Symtavision GmbH, Frankfurter Straße 3b, D-38122 Braunschweig, Germany
jersak@symtavision.com



1. INTRODUCTION

With the trend to higher function integration in the avionics domain, multiple applications need to run in parallel on one LRU (line replaceable unit). To fulfill the safety requirements of avionics, these applications must essentially be protected against each other, such that a failure in one application does under no circumstances lead to a failure in another application due to memory corruption, locked resources, or CPU stealing. In avionics, this is done by partitioned operating systems according to the Arinc 653 standard in which each partition runs independently from all other partitions, and mutual interference is prohibited through appropriate mechanisms. This provides a virtualization framework to integrate several applications on one LRU without compromising real-time correctness and functional safety. However, the optimal configuration of an Arinc 653 system is not addressed by existing methods. In this paper, we show how scheduling analysis can be used to verify and optimize an Arinc 653 system.

According to the Arinc 653 standard, partitioned scheduling relies primarily on the time-division multiple access (TDMA) scheduling scheme. Each partition has one (or more) predefined time-window(s) for execution, and the windows are executed in a fixed order. This ensures a deterministic schedule. However, the performance of each of the applications (each in its own partition) depends on the set-up of the so called major frame (MAF) and on the internal performance and timing requirements of each application, including deadlines. So, optimizing the MAF layout is important for both safety and cost.

In this paper, we present examples to demonstrate the impact of MAF layouts on the overall performance and on the real-time behavior of several applications in a multi-partition environment. We show that knowing information about the timing of processes within an application can substantially guide us towards optimized MAF layouts. Finally, we also propose a tool flow in collaboration with SYSGO and their PikeOS ARINC 653 operating system.

2. FIXED PARTITIONED SCHEDULING

According to the Arinc 653 standard, the top-level scheduling is a major frame (MAF) that defines a series of time windows. The time windows are specified by their **start time** and their **window duration**. Figure 1 illustrates a default MAF layout for four partitions: all windows have the same time duration and there is a one-to-one correspondence between a partition and a time window in the MAF.

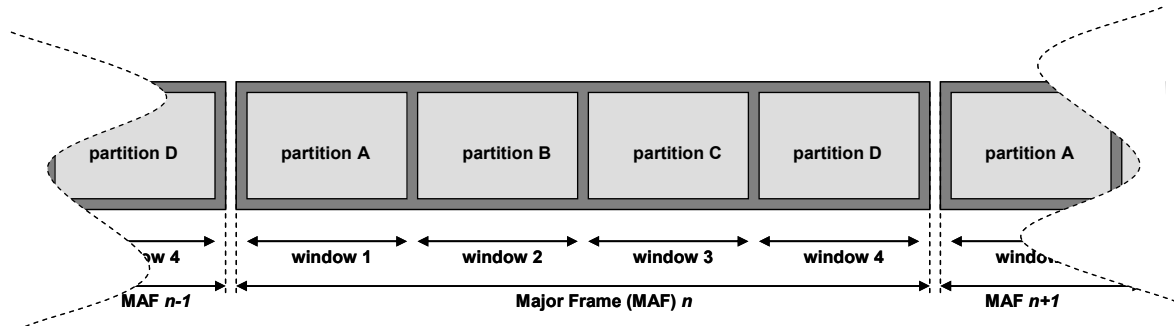


Figure 1: Simple Major Frame Layout

Clearly, this simple MAF layout works well if all partitions have the same (or very similar) performance and timing requirements. When periods and durations differ among the partitions, it is more complex to define an efficient MAF layout. The task is to select **window starts** and **window durations** according to the **partition periods** and **partition durations**.

When the periods differ, there are two basic choices for the overall MAF cycle time: the greatest common divisor (GCD) or the least common multiple (LCM) of all partition periods. Nevertheless, any such MAF layout would require that either partitions must be segmented (in case of the smaller GCD) or occur multiple times within one MAF (in case of LCM) or a mixture of both. Figure 2 shows an example, in which partition A occurs twice within one MAF.

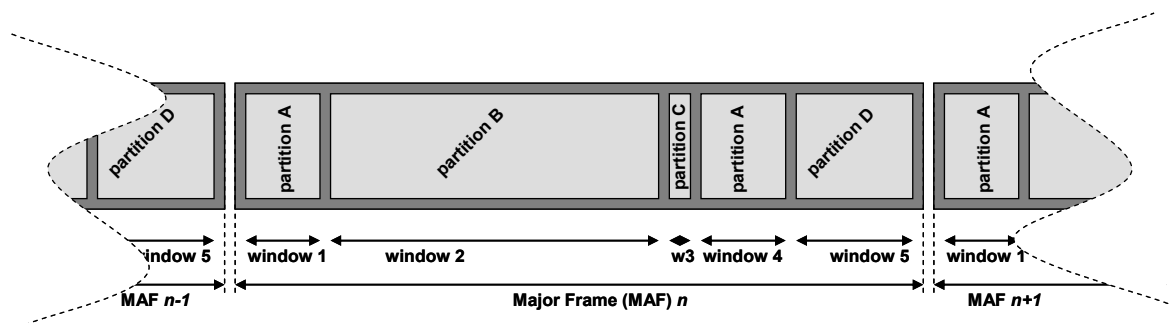


Figure 2: More Complex Major Frame Layout

This little example already shows that there are many options to design a major frame (MAF). Furthermore, the quality of a MAF layout depends on more than the number of partitions.

3. ARINC PROCESSES WITHIN A PARTITION

According to the Arinc 653 standard, a partition is comprised of processes that (within the corresponding partition) have a priority, a period, a capacity (deadline), and possibly a start delay. Figure 3 shows an example of six processes. Start times, execution, termination, and the deadline (capacity) are drawn. We see that several processes can be active at the same time. In this case, the scheduling is determined by the priorities of the processes, where higher-priority processes preempt lower-priority ones.

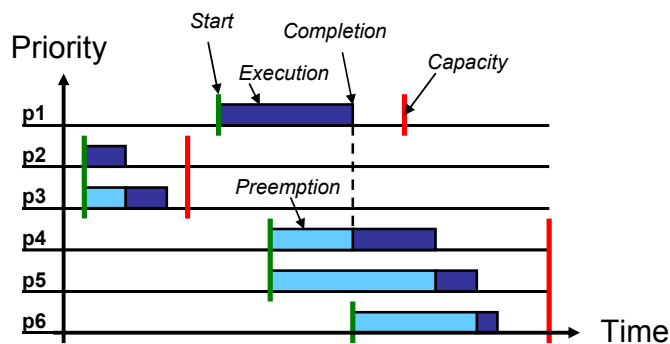


Figure 3: A process Schedule within a Partition

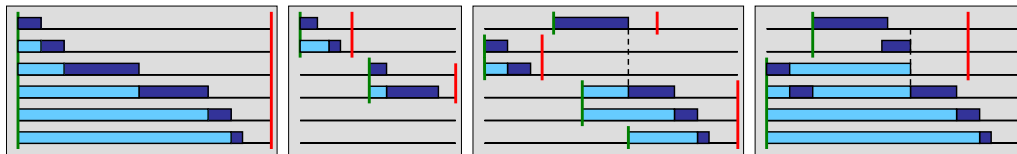


Figure 4: Four Different Partition Schedules

This means, there is a hierarchical combination of TDMA scheduling at the MAF level with SPP (static-priority preemptive) scheduling at the partition level. Of course, the partition schedules (SPP on processes) can significantly differ among different partitions, examples are shown in Figure 4. There can be more or less processes that are more or less CPU hungry.

4. SYSTEM TIMING AND SCHEDULING ANALYSIS

It is clear that performance requirements of each partition must match the available performance given by the MAF layout. The question is: How can the different roles – the MAF layout developer (or system integrator) and the partition developer (or application designer)– collaboratively come to the most efficient solution? Spending too much window duration clearly increases idle times and wastes resources. Spending too little increases the risk of capacity overruns that must be avoided.

This can be achieved by systematically exploring the different options that both roles have. At the level of abstraction sketched out here already, scheduling analysis is a suitable tool. Scheduling analysis calculates a-priori the timing behavior of a system. Examples are a) the worst-case response times for the different processes, which can then be verified against deadlines (capacities), and b) the performance of the overall MAF cycle, which can be checked against the requirements of the individual partitions.

The SymTA/S scheduling analysis tool suite requires relatively few key parameters, mostly those ones mentioned thus far: period and duration of windows and partitions as well as process periods, priorities and execution times. The later can be determined by worst-case execution time analysis tools such as aiT from AbsInt, and the others can be imported from the appropriate configuration files, or they can be explored and configured as part of scheduling analysis and optimization.

SymTA/S is well established in the automotive domain already. It is used for ECU (electronic control unit, comparable to LRU) design and bus dimensioning. Applications are: verification of deadlines, securing the system availability, optimizations for cost and/or future extensibility, and more. A key property of scheduling analysis is that it systematically covers all timing corner cases and is therefore well suited for the verification of real-time properties of avionics safety critical systems.

5. DESIGN FLOW

The integration of SymTA/S scheduling analysis with existing design tools shall be validated in collaboration with SYSGO for their PikeOS ARINC 653 operating system in the context of the European-funded INTERESTED project.

The integration with SYSGO's PikeOS shall enable systematic timing analysis, budgeting, performance optimization and timing verification throughout an ARINC653-based execution platform design-cycle, providing value to aerospace customers. Main customer benefits foreseen are

- right execution platform dimensioning to avoid late bottlenecks that may require costly re-implementation / re-certification
- reduced amount of testing necessary for execution platform certification
- faster design-time through good integration of synthesis tool (SYSGO) and analysis tool (Symtvision)

This will be achieved through the development of an ARINC 653 scheduling analysis library for Symtvision's tool SymTA/S which will provide the necessary interface to export/import ARINC 653 configuration data to/from SYSGO tools. The traditional design-flow will be augmented at three levels:

- Early: at the concept level, performance prediction and optimization will be supported to enable good architecture decisions and the derivation of detailed timing requirements for later design stages.
- Mid: at the configuration level, all detailed configuration decisions will be pre-verified to make sure that all timing constraints are met under all relevant conditions
- Late: at the integration level, the implemented system will be verified against the earlier models.

A preliminary design flow is shown in Figure 5.

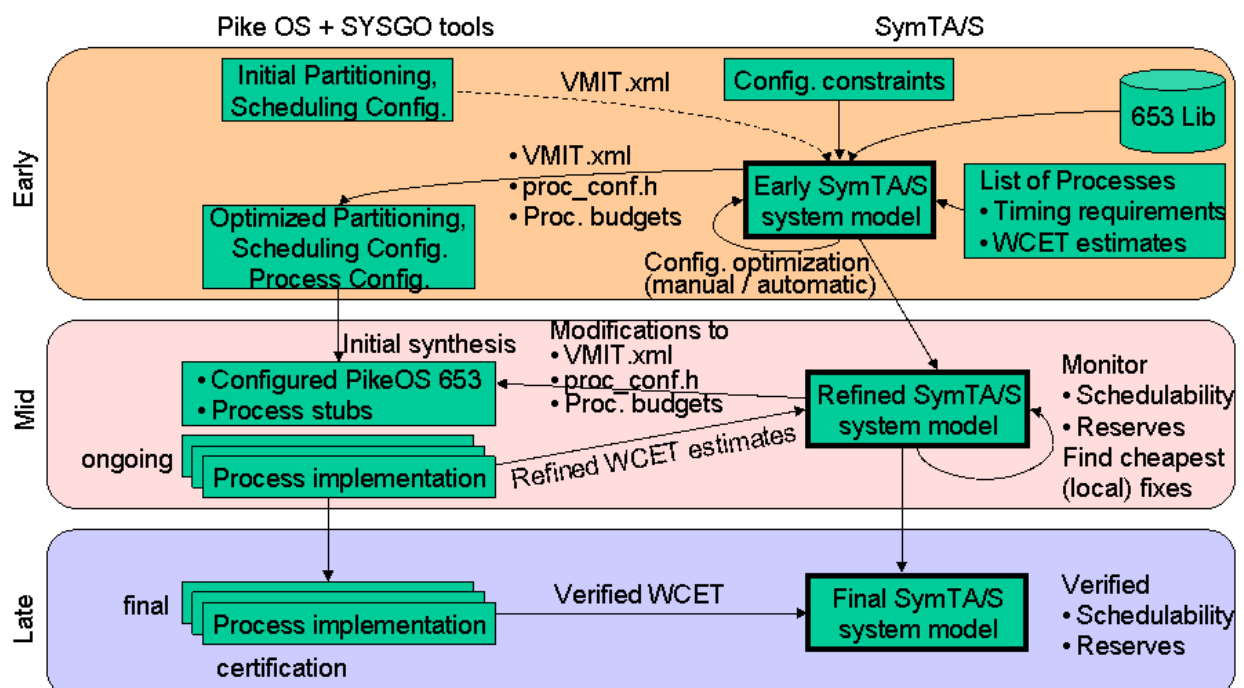


Figure 5: preliminary design flow for scheduling analysis and optimization for an ARINC 653 partitioned OS

6. CONCLUSION & OUTLOOK

In this paper, we have highlighted the most relevant concepts of the hierarchical scheduling of the Arinc 653 standard. Understanding and controlling scheduling is essential to build robust, safety-critical systems that must meet all timing constraints under all relevant conditions.

Scheduling analysis tools like SymTA/S exists that can calculate the timing behavior of MAF layout and partitions a-priory. This enables thorough verification of deadlines, securing the system availability, optimizations for cost and/or future extensibility, and more. In the EU-funded INTERESTED project, Symtavisision and SYSGO will cooperate on establishing an integrated design flow for SYSGO's PikeOS, a certified Arinc653-compliant operating system. The process is based on standardized configuration formats and a refinement of the traditional design process. With such an integrated flow, users of Arinc653 operating systems can access configuration and analysis tools very efficiently, providing maximum productivity in system design.

The next step will then be the consideration of communication within and among different partitions. This will include end-to-end timing of signal that traverse through several buffers, ports, etc. and will require incorporating the Arinc 664 (AFDX, APEX) standards for intra-LRU communication. For inter-LRU communication across networks, AFDX and TTP are possible candidates. Another type of extension targets more complex scheduling hierarchies and the mix of safety-critical and non-safety-critical application with mixed static and best-effort scheduling in one system.