A Simulation Framework to Analyze the Scheduling of AVR tasks with respect to Engine Performance

Paolo Pazzaglia, Alessandro Biondi, Marco Di Natale and Giorgio Buttazzo
Scuola Superiore Sant'Anna, Pisa, Italy
Email: {name.surname}@sssup.it

***Abstract*—We present a simulation framework, based on Simulink and an extension of the T-Res scheduling simulator tool to help provide a better characterization of the very popular problem of scheduling and analysis of Adaptive Variable Rate Tasks (AVR) in engine control. The purpose of the tool is to go beyond the simplistic model that assumes hard deadlines for all tasks and to study the impact of scheduling decisions with respect to the functional implementations of the control algorithms and the true performance of the engine.**

I. Introduction

The study of the schedulability conditions for engine control tasks (or adaptive variable rate - AVR) is gaining popularity in the real-time research community because of the novel nature of the problem and the special activation conditions that apply to some of the system tasks. These tasks are not periodic or sporadic, but are activated by the rotation of the engine crankshaft (a parameter of the physical controlled system). In addition, to compensate for the increased CPU load at high rotation speeds (and more frequent activation times), the code implementation of these tasks is defined in such a way that at given speed boundaries, the implementation is simplified and the execution time is reduced. A typical engine control application consists of time-driven periodic tasks with fixed periods, typically between a few milliseconds and 100 ms (see [1], page 152), and angular tasks triggered at specific crankshaft angles. The activation rate of such angular tasks hence varies with the engine speed (variable-rate tasks). For example, for engines where the speed varies from 500 to 6500 revolutions per minute (RPM), the interarrival times of the angular tasks range from about 10 to 120 ms (assuming a single activation per cycle).

With respect to the set of activation instants, the dependency from a physical phenomenon characterizes this problem as truly belonging to the class of problems in cyber-physical systems (CPS). However, in many papers the dependency of the timing and scheduling problem from the physics of the controlled system is restricted to the set of activation events and every other concern is hidden under the typical assumption of hard deadlines.

In reality, this problem (as many others) is representative of a class of control systems in which deadlines can be missed without catastrophic consequences, and the problem should actually be defined as a design optimization, where the objective is to select the controls implementations and the scheduling policy in such a way that a set of engine performance functions are optimized (including power, emissions, noise, pollution). These performance functions depend in complex ways from timing parameters, such as jitter and latency. Informally, the objective of the scheduler is not to miss too many deadlines or produce actuation signals that are too much delayed.

Formally, the problem is quite complex and extremely unlikely to be solved in a simple, closed analytical form or even with a general procedure for expressing the dependency of th performance from scheduling. This is the reason for the investigation of alternative approaches that are based on the simulation of the three system components in a joint environment:

- A model of the engine and the combustion process in it (the physical system or plant)
- A model of the engine controls
- A model of the task configuration and the scheduling

II. Our Simulation Framework for the Analysis of the Performance impact of Scheduling

Our cosimulation framework follows the principles of CPS system analysis. It is based on the popular Simulink toolset and leverages the T-Res cosimulation environment for the simulation of the task scheduling [2].

For the development of the engine model we leveraged information from several sources, including engine models for the steady state and event-based models as described in [1] and other empirical models found online.

The engine controls are currently extremely simple and only contain a simple analytical formula that computes the angle of injection and the injection time that is defined by a calibration table.

Finally, the T-Res simulation framework described in [2] is used for modeling the scheduling delays.

III. Extending T-Res for modeling AVR tasks

T-Res consists of a set of custom Simulink blocks representing tasks and kernels and allows to interface the Simulink simulation engine, acting as master, with a scheduling simulator in a co-simulation environment (see Figure 1). The scheduling simulator (we use RTSim [3], but the backend simulation engine can be changed) computes the scheduling delays and latches the outputs of the corresponding tasks until their simulated completion time. This allows to simulate delays in the production of output values and the corresponding impact on the control function.

T-Res provides a custom block for representing the kernel and its scheduler. The block is configured with the selection of the scheduling policy and the behavior in case of deadline (period) overrun. The kernel block provides a set of activation signals as output. These activation signals go to instances of the the second type of custom blocks, representing tasks. Each task receives an activation signal from the kernel (indicating when the task begins or resumes execution), and is characterized by an execution time estimate (a configuration parameter), and a signal going back to the kenel and providing the amount of time that is still required by the task at each point in time. The task block produces as output a set of activation and latch signals for all the functional subsystems that are executed by the task.

With respect to the activation, sporadic tasks are characterized by an activation event going as input to the kernel block, or a periodic activation specification, provided as a configuration parameter to the kernel (for details, refer to [2]). The execution time description is provided to the kernel for each task using a simple language.

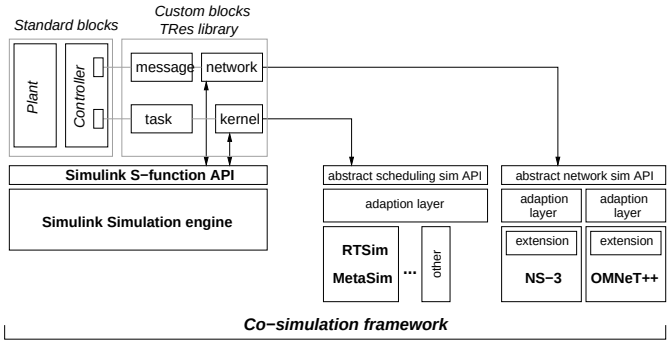

Figure 1. The TRes cosimulation architecture.

For the purpose of this project we entended the task model block and the timing information associated with it to allow for the modeling of the AVR behavior, as shown in Figure 2. The task block in T-Res includes a signal for the explicit activation in case of event-triggered tasks. This signal is used to define the activation of the task in correspondence to given angular positions of the engine crankshaft. In addition, the block has been extended to include another imput that refers to a *mode* index. This input can be used for multiple purposes and defines a different execution time behavior for a finite and enumerated set of conditions.

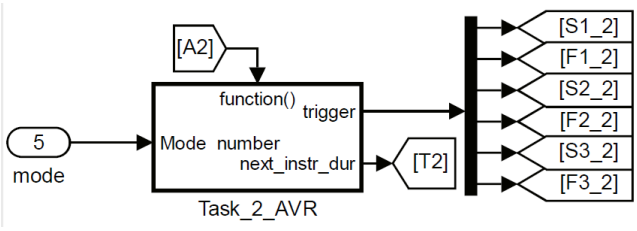

Figure 2. A custom block for modeling an AVR task.

In the case of AVR tasks, the mode index is provided from a simple block that looks at the engine rotation speeds and, based on the speed range, defines the execution time that the task requires.

The task will have different execution times for different speed modes according to a specification of execution times as a function of the mode (speed) index provided as a workspace variable.

IV. Simulink Models of the Engine and the Control Tasks

Figure 3 shows the model of the engine and the control functionality in Simulink. The blocks in the upper part of the figure represent the engine subsystems that are currently considered and includes the turbocharger, the compressor manifold, the intercooler, the intake and exhaust manifolds and the model of the engine cylinders. The subsystem on the bottom part of the figure wraps our model of the engine controller, with its outputs: the injection angle and duration and the VGT.

Figure 4 shows the subsystems realizing the controller functions and the task model of the controller. The model consists of a kernel (top left side), and four tasks on the bottom left side. One of the four tasks is an AVR, two are periodic and one represents background computations. The chains of subsystems on the right side represent the control functions implemented by the tasks. The second from the top contains the six subsystems that are executed by the AVR task (matching the six output signals from the AVR task block).

V. Objective and Status

A detailed modeling of the control function is necessary to better understand the impact of deadline misses or long latencies. Depending on the implementation of the control function, a deadline miss may result in a late actuation, or a missed actuation or even an actuation with old data. In our controls implementation, the AVR task computes the phase and duration of the injection and passes them to the task that simulates the injection actuators. Hence, a missed deadline results in actuating the injectors with the values computed in the previous cycle with a likely error in phase and duration with respect to the ideal values.

The objective of our framework is multifold:

- To understand the effect of the scheduling on the engine performance and to use the environment for analyzing the impact of scheduling policies and parameters, such as evaluating fixed priority vs EDF or different possible priority assignments and task configurations.
- To analyze the timing parameters that truly of interest for evaluating the performance of the engine and possibly attept a characterization that isolates the attributes of interest. This includes, among others, the evaluation of schemes like m-k deadline misses, or overload management (maximum lateness).

- To better characterize the design problem consisting in the optimal selection of the transition speeds for AVR tasks.

Currently, within the assumptions of our model, the simulation is able to show how the scheduling delays result in errors in the angle/duration of the injection actuation. Figure 5 shows preliminary results. In the figure graph, the vertical axis shows the phase error in the actuation of the injection for a sample manoeuvre consisting of a sudden acceleration and a corresponding increase in the engine rotation speed from low to high values. Two graphs are plotted in the figure. The graph in red (lighter) color shows the angle error when the execution time of the AVR task is kept constant, regardless of the engine speed. At high rotations, the task misses deadlines and the injection angle error grows to almost 50 degrees. When the execution time of the AVR task is reduced at high rates, the scheduling delays are much lower and, correspondingly, the angle error of the injection is much lower, as shown by the blue line in the graph. The angular error in the injection is related to a variation (loss) in the power performance of the engine.

Our objective is to relate the errors in phase and duration of the injection to a possible loss of power, providing ways to analyze the impact of scheduling with respect to the first performance function of interest. However, even within the limited scope of power performance analysis, the evaluation of the scheduling impact (and the AVR characteristics of tasks), requires that the model includes multiple representations of the control functionality, one for each possible execution mode of the AVR tasks. When these are available, the model will provide an early capability of expressing the performance impact of control implementations at different levels of complexity (for variable execution times or WCETs). Clearly, this is only the initial objective, given that a realistic model should also include the characterization of pollution, noise and efficiency.

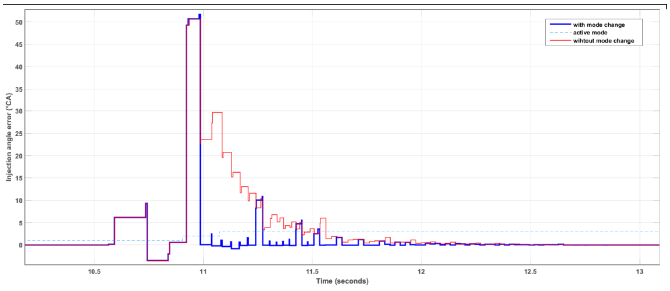

Figure 5. Angular error in the injection caused by scheduling delays of the AVR task: error with fixed execution times (red) and with adaptive excution (in blue).

VI. Related Work

The presentation of the task model in which engine control tasks are implemented with a variable computational requirements for increasing speeds is in [4],

These tasks are also referred to adaptive variable-rate (AVR). Analyzing the schedulability of tasks sets consisting of both periodic and AVR tasks is a difficult problem that has been addressed by several authors under various simplifying assumptions, under both fixed priority scheduling [5]–[7] and Earliest Deadline First (EDF) [8]–[10]. Other authors proposed methods for computing the exact interference [11] and the exact response time [7] of AVR tasks under fixed priority scheduling. It has been shown [10] that, given the large range of possibile interarrival times of an AVR task, fixed priority scheduling is not the best choice for engine control systems since, while EDF exhibits a nearly optimal scheduling performance. Based on this fact, Apuzzo et al. [12] provided an operating system support for AVR tasks under the Erika Enterprise kernel [13].

All the papers considered above, however, focused on analyzing the schedulability of task sets consisting of periodic and AVR tasks, without any concern on engine performance. A performance-driven design approach has been addressed in [14] for finding the transition speeds that trigger the mode changes of an AVR task.

A very large number of projects target the evaluation of scheduling policies and the analysis of task implementations. A necessarily incomplete list includes Yartiss [15], ARTISST [16], Cheddar [17], and Stress [18].

Finally, TrueTime [19] is a *freeware*[1] Matlab/Simulink-based simulation tool that has been developed at Lund University since 1999. It provides models of multi-tasking real-time kernels and networks that can be used in simulation models for networked embedded control systems. TrueTime is used by many research groups worldwide to study the (simulated) impact of lateness and deadline misses on controls. In TrueTime, the model of task code is represented by *code functions* that are written in either Matlab or C++ code. Several research works investigate the consequences of computation (scheduling) and communication delays on controls. An overview on the subject can be found in [20].

References

[1] L. Guzzella and C. H. Onder, *Introduction to Modeling and Control of Internal Combustion Engine Systems*. Springer-Verlag, 2010.

[2] F. Cremona, M. Morelli, and M. D. Natale, "Tres: A modular representation of schedulers, tasks, and messages to control simulations in simulink," in *Proc. of the 31st ACM Symposium on Applied Computing (SAC 2016)*, Pisa, Italy, April 4-8, 2016.

[3] L. Palopoli, G. Lipari, L. Abeni, M. D. Natale, P. Ancilotti, and F. Conticelli, "A tool for simulation and fast prototyping of embedded control systems," in *LCTES/OM*, S. Hong and S. Pande, Eds. ACM, 2001, pp. 73–81.

[4] D. Buttle, "Real-time in the prime-time," in *Keynote speech at the 24th Euromicro Conference on Real-Time Systems*, Pisa, Italy, July 12, 2012.

[5] J. Kim, K. Lakshmanan, and R. Rajkumar, "Rhythmic tasks: A new task model with continually varying periods for cyber-physical systems," in *Proc. of the Third IEEE/ACM Int. Conference on Cyber-Physical Systems (ICCPS 2012)*, Beijing, China, April 2012, pp. 28–38.

[6] R. I. Davis, T. Feld, V. Pollex, and F. Slomka, "Schedulability tests for tasks with variable rate-dependent behaviour under fixed priority scheduling," in *Proc. 20th IEEE Real-Time and Embedded Technology and Applications Symposium*, Berlin, Germany, April 2014.

[1] http://www3.control.lth.se/truetime/LICENSE.txt

[7] A. Biondi, M. D. Natale, and G. Buttazzo, "Response-time analysis for real-time tasks in engine control applications," in *Proceedings of the 6th International Conference on Cyber-Physical Systems (ICCPS 2015)*, Seattle, Washington, USA, April 14-16, 2015.

[8] G. Buttazzo, E. Bini, and D. Buttle, "Rate-adaptive tasks: Model, analysis, and design issues," in *Proc. of the Int. Conference on Design, Automation and Test in Europe (DATE 2014)*, Dresden, Germany, March 24-28, 2014.

[9] A. Biondi and G. Buttazzo, "Engine control: Task modeling and analysis," in *Proc. of the International Conference on Design, Automation and Test in Europe (DATE 2015)*, Grenoble, France, March 9-13, 2015, pp. 525–530.

[10] A. Biondi, G. Buttazzo, and S. Simoncelli, "Feasibility analysis of engine control tasks under EDF scheduling," in *Proc. of the 27th Euromicro Conference on Real-Time Systems (ECRTS 2015)*, Lund, Sweden, July 8-10, 2015.

[11] A. Biondi, A. Melani, M. Marinoni, M. D. Natale, and G. Buttazzo, "Exact interference of adaptive variable-rate tasks under fixed-priority scheduling," in *Proceedings of the 26th Euromicro Conference on Real-Time Systems (ECRTS 2014)*, Madrid, Spain, July 8-11, 2014.

[12] V. A. A. Biondi and G. Buttazzo, "OSEK-like kernel support for engine control applications under EDF scheduling," in *Proceedings of the 22nd IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS 2016)*, Vienna, Austria, April 11-14, 2016.

[13] "Erika enterprise: an OSEK compliant real-time kernel." [Online]. Available: http://erika.tuxfamily.org/drupal/

[14] A. Biondi, M. D. Natale, and G. Buttazzo, "Performance-driven design of engine control tasks," in *Proceedings of the 7th International Conference on Cyber-Physical Systems (ICCPS 2016)*, Vienna, Austria, April 11-14, 2016.

[15] Y. Chandarli, F. Fauberteau, D. Masson, S. Midonnet, M. Qamhieh *et al.*, "Yartiss: A tool to visualize, test, compare and evaluate real-time scheduling algorithms," in *Proceedings of the 3rd International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems*, 2012, pp. 21–26.

[16] D. Decotigny and I. Puaut, "Artisst: an extensible and modular simulation tool for real-time systems," in *Object-Oriented Real-Time Distributed Computing, 2002.(ISORC 2002). Proceedings. Fifth IEEE International Symposium on*. IEEE, 2002, pp. 365–372.

[17] F. Singhoff, J. Legrand, L. Nana, and L. Marcé, "Cheddar: a flexible real time scheduling framework," in *ACM SIGAda Ada Letters*, vol. 24, no. 4. ACM, 2004, pp. 1–8.

[18] N. C. Audsley, A. Burns, M. F. Richardson, and A. J. Wellings, "Stress: A simulator for hard real-time systems," *Software: Practice and Experience*, vol. 24, no. 6, pp. 543–564, 1994.

[19] A. Cervin, D. Henriksson, B. Lincoln, J. Eker, and K.-E. Årzén, "How does control timing affect performance?" *IEEE control systems magazine*, vol. 23, no. 3, pp. 16–30, 2003.

[20] K. J. Astrom and B. Wittenmark, "Adaptive control," in *Prentice Hall*, 2016.

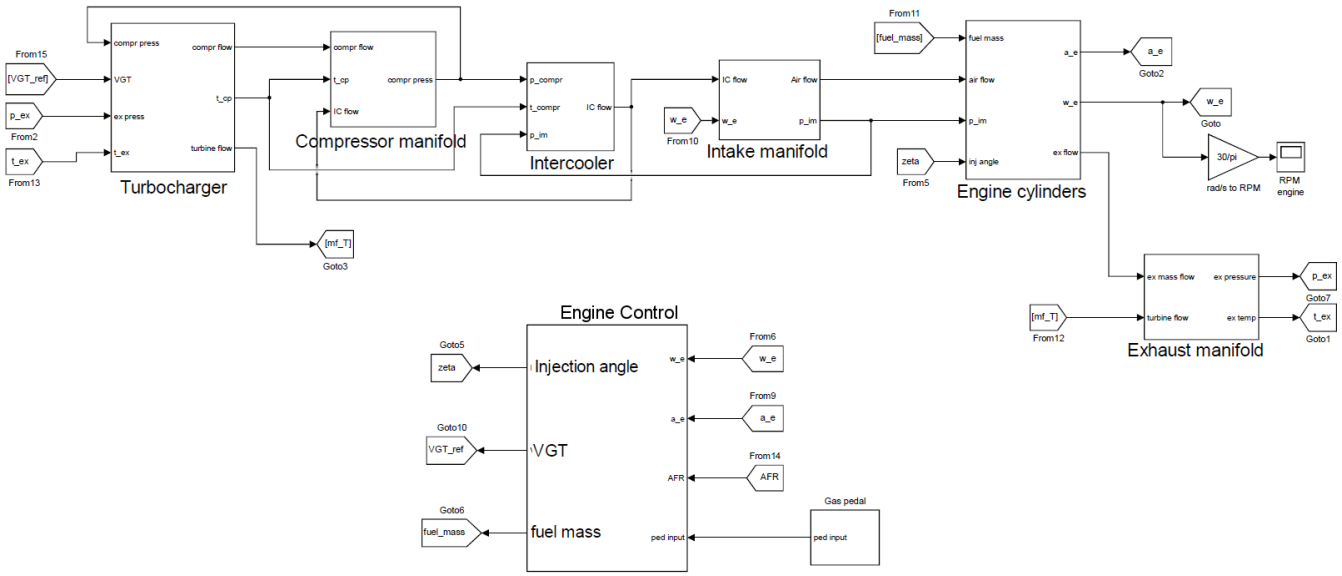

Figure 3. Engine control model in Simulink.

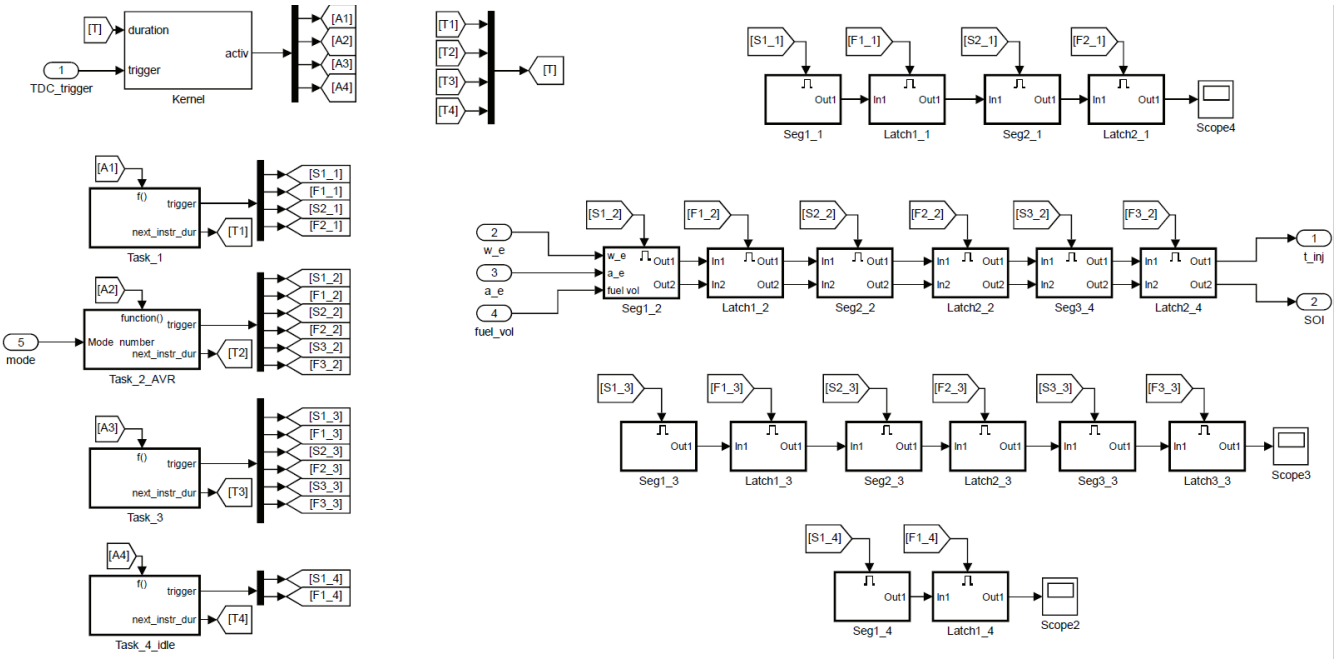

Figure 4. Task model in TRES.