Work-in-Progress: Development of Margin-shared System-level Logical Execution Time Simulator to Support Scheduling Design of Automotive ECUs

Masashi Mizoguchi
Research and Development Group
Hitachi, Ltd.
Hitachi, Japan
masashi.mizoguchi.re@hitachi.com

Yuma Kato
Research and Development Group
Hitachi, Ltd.
Hitachi, Japan
yuma.kato.vd@hitachi.com

Kentaro Yoshimura
Research and Development Group
Hitachi, Ltd.
Hitachi, Japan
kentaro.yoshimura.jr@hitachi.com

Takaaki Nokaido
Mobility Solution Business Unit
Hitachi Astemo, Ltd.
Hitachinaka, Japan
takaaki.nokaido.yh@hitachiastemo.com

Yasuhiro Ikeda
Mobility Solution Business Unit
Hitachi Astemo, Ltd.
Hitachinaka, Japan
yasuhiro.ikeda.to@hitachiastemo.com

Hideyuki Sakamoto
Mobility Solution Business Unit
Hitachi Astemo, Ltd.
Hitachinaka, Japan
hideyuki.sakamoto.br@hitachiastemo.com

Recently, more and more vehicles are shifted to be software-defined, i.e., any functions are expandable by continuously updating software on Electronic Control Units (ECUs) during the entire vehicle life cycle. In addition, some ECUs are composed of multiple Systems on Chips (SoCs) for high computation performance. These trends make software development more challenging than ever before. Then, system-level logical execution time (SL-LET) scheduling has attracted attention as a solution to it. SL-LET fixes the start and end timing of each task and that of each communication between tasks. This allows data flow and end-to-end latency to be deterministic, facilitating timing design and verification.

We are conducting research to support timing design based on SL-LET for ECUs consisting of multiple SoCs. As a first step, we have developed a task scheduling simulator. For each Central Processing Unit (CPU) on each SoC in the ECU, the simulator calculates the time when a task starts or ends. The characteristic of our method is that we extend [1] to handle margin sharing among SoCs. If the worst-case execution time of each task is used to fix the timing, the end-to-end latency may become too long to meet requirements. This can be overcome by margin sharing. Figure 1 shows the summary of the margin shared between `SoC1` and `SoC2`. Let the length of the margin shared by `Task1` and `Task2` be $m \in \mathbb{R}_{>0}$. The margin is limited to the execution period $p \in \mathbb{R}_{>0}$ of inter-SoC communication denoted by `com`. Assume that the processing time of `Task1` exceeds the interval by $e_1 \in \mathbb{R}_{\geq 0}$. If $e_1 \in [0, p)$ holds, the delay of the timing of sending data is p, not e_1. Then, the start timing of `Task2` is shifted by p and the remaining margin is $m - p$. If the exceed of processing time of `Task2` (denoted by $e_2 \in \mathbb{R}_{\geq 0}$) is less than $m - p$, it is assured that other tasks such as `Task3` are not affected. Figure 2 shows a part of the output from the simulator with $p = 10$[ms]. The start of subsequent `Task2` was successfully shifted from 650[ms] to 660[ms] when the preceding `Task1` did not finish by its deadline 630[ms].

Using the task scheduling simulator, we will continue research to support timing design. We plan to estimate the end-to-end latency possibly spanning multiple SoCs and the bus load for inter-SoC communication in the ECU by considering the relationship of the input and output data of each tasks.

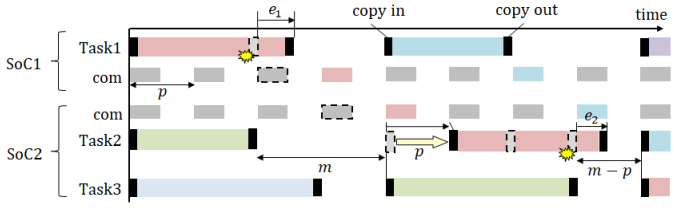

Fig. 1. Summary of the margin shared among `SoC1` and `SoC2`.

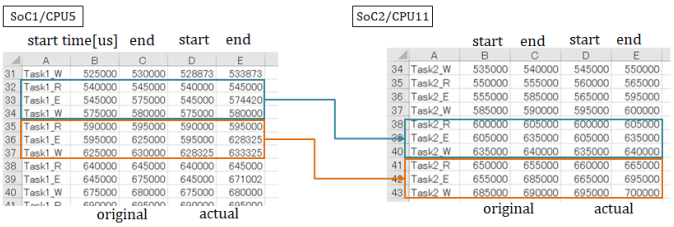

Fig. 2. Simulator output showing the margin sharing among SoCs.

References

[1] A. Biondi and M. Di Natale, "Achieving predictable multicore execution of automotive applications using the let paradigm," in *2018 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 2018, pp. 240–250.

2771-571X/24/$31.00 ©2024 IEEE
DOI 10.1109/EMSOFT60242.2024.00007