# Daedalus$^{RT}$: The System-Level Design Flow for Hard-Real-Time Embedded MPSoCs Platforms

Mohamed A. Bamakhrama, Jiali Teddy Zhai, Hristo Nikolov, and Todor Stefanov
Leiden Institute of Advanced Computer Science
Leiden University, Leiden, The Netherlands
Email: {mohamed, tzhai, nikolov, stefanov}@liacs.nl

***Abstract*—Daedalus$^{RT}$ is a system-level design flow for automated design of real-time embedded multiprocessor system-on-chip (MPSoC) platforms. It extends the existing Daedalus design flow with capabilities to support multiple applications and provide real-time guarantees. Daedalus$^{RT}$ design flow consists of a set of tools that provide automatic parallelization of sequential programs, real-time schedulability analysis, and system-level synthesis.**

## I. Introduction

The increasing complexity of embedded streaming applications (e.g., multimedia, imaging, and signal processing) imposes new challenges on embedded system designers [1]. Addressing the new challenges increases significantly the complexity of system design. However, the design time must remain acceptable, which requires novel systematic, and moreover, automated design methodologies. In the following we describe briefly such a methodology realized as the Daedalus$^{RT}$ design flow.

## II. Daedalus$^{RT}$ Design Flow

The Daedalus$^{RT}$ framework considers only dataflow dominated applications (e.g., multimedia, imaging, and signal processing), in which streams of data are processed by different tasks. It is based on the Daedalus design flow [2] targeted towards system-level design of MPSoCs. The key differences between Daedalus$^{RT}$ and Daedalus are the following: 1) support for multiple applications. The initial Daedalus flow supports only a single application, whereas Daedalus$^{RT}$ flow supports multiple applications, 2) replacing the complex design space exploration (DSE) with very fast yet accurate schedulability analysis to determine the minimum number of processors needed to schedule the applications, and 3) using hard-real-time multiprocessor scheduling algorithms that provide temporal isolation to schedule the applications.

Daedalus$^{RT}$ consists of three phases: parallelization, analysis, and system synthesis. The first phase, i.e., parallelization, accepts as an input a set of independent streaming applications specified as Static Affine Nested Loop Programs (SANLP) [3] without cyclic dependencies. Each application is then parallelized using the `pn` compiler [3] to produce two models: an *analysis model* which is based on the Cyclo-Static Dataflow (CSDF) model [4], and an *implementation model* which is based on the Polyhedral Process Networks (PPN) model [5]. During the parallelization phase, each application is analyzed/profiled to determine the worst-case execution time (WCET) of each task in the application. The output of the parallelization phase is a set of CSDF graphs and a set of PPN graphs, where each input application has a corresponding CSDF and PPN graphs.

The second phase, i.e., analysis, accepts as an input the set of CSDF graphs produced by the parallelization phase. Afterwards, it applies the real-time schedulability analysis proposed in [6] on these graphs to determine the minimum number of processors needed to schedule the applications. The output of this phase is platform and mapping specifications. The platform specifications describe the hardware architecture which is tiled homogeneous MPSoC with distributed memory. The mapping specification describes the mapping of tasks to processors.

The third phase, i.e., system synthesis, accepted as an input the set of PPN graphs produced by the parallelization phase together with the platform and mapping specifications produced by the analysis phase. The system synthesis phase is realized using the ESPAM tool [7]. Xilinx Platform Studio (XPS) backed of ESPAM is used to generate an implementation that can be synthesized on FPGA.

## III. Conclusion

Daedalus$^{RT}$ is a very fast automated design flow which enables rapid prototyping of hard-real-time MPSoC systems running a set of embedded streaming applications.

## References

[1] L. Karam *et al.*, "Trends in multicore DSP platforms," *IEEE Signal Processing Magazine*, vol. 26, no. 6, pp. 38–49, November 2009.
[2] H. Nikolov *et al.*, "Daedalus: toward composable multimedia MP-SoC design," in *Proceedings of the 45th DAC*, 2008, pp. 574–579.
[3] S. Verdoolaege, H. Nikolov, and T. Stefanov, "`pn`: A Tool for Improved Derivation of Process Networks," *EURASIP Journal on Embedded Systems*, vol. 2007, no. 1, 2007.
[4] G. Bilsen *et al.*, "Cycle-static dataflow," *IEEE Transactions on Signal Processing*, vol. 44, no. 2, pp. 397–408, February 1996.
[5] S. Verdoolaege, "Polyhedral Process Networks," in *Handbook of Signal Processing Systems*, S. Bhattacharyya *et al.*, Eds. Springer US, 2010, pp. 931–965.
[6] M. Bamakhrama and T. Stefanov, "Hard-real-time scheduling of data-dependent tasks in embedded streaming applications," in *Proceedings of the 9th EMSOFT*, October 2011, pp. 195–204.
[7] H. Nikolov, T. Stefanov, and E. Deprettere, "Systematic and Automated Multiprocessor System Design, Programming, and Implementation," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, no. 3, pp. 542–555, March 2008.