

Handling Dynamism in Embedded System Design by Application Scenarios

Stefan Valentin Gheorghita¹,
Twan Basten, Henk Corporaal

** EE Department, Electronic Systems Group, Eindhoven University of Technology,
PO Box 513, 5600 MB, Eindhoven, The Netherlands
{s.v.gheorghita,a.a.basten,h.corporaal}@tue.nl*

ABSTRACT

In the past years, real-time embedded systems became more and more complex. From the user perspective, these systems have stringent requirements regarding size, performance and power consumption, and due to business competition, their time-to-market is a crucial factor. Therefore, much work has been done in developing design methodologies for embedded systems to cope with these tight requirements. In this work, we present a basic methodology based on the concept of *application scenarios* for handling in real-time embedded systems design the dynamism that appears within the applications. As a case study, we apply this methodology for reducing the energy consumption on a dynamic voltage aware processor.

KEYWORDS: Application Scenarios, Dynamic Voltage Scaling (DVS), Embedded Systems

1 Introduction

Modern multimedia applications usually have real-time constraints and they are implemented using heterogeneous multiprocessor systems-on-chip. Dimensioning a system requires accurate estimations of resources needed by the applications. Overestimation leads to over-dimensioning. If the entire application is analyzed at once, a large over-estimation will result as modern applications have a lot of dynamism, and due to the limited information that may be derived. For a more accurate estimation, all the operation modes in which an application can run must be considered. To avoid an explosion in the number of different modes, those that are similar with respect to required resources should be analyzed together. To do this, we combine them into an *application scenario*.

2 Scenario Based Design

Scenario based design has been used for a long time in different areas, including embedded system design ([Paul05, Doug04]). These scenarios describe, in an early phase of the

¹This work was supported by the Dutch Science Foundation, NWO, project FAME, number 612.064.101.

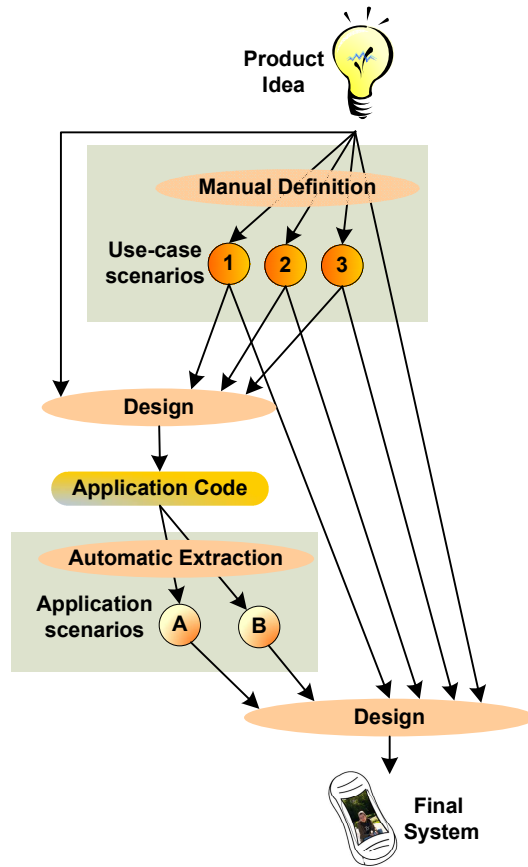


Figure 1: A scenario based design flow for embedded systems.

development process, the use of a future system. They appear like narrative descriptions of envisioned usage episodes, or as unified modeling language (UML) use-case diagrams. Moreover, these scenarios, also called *use-case scenarios*, focus on the functional and timing behaviors of the application and on its interaction with the users and the environment, not on the resources required by an application to meet its constraints.

Usually to design and dimension an embedded system the worst case estimations for the needs in hardware resources are taken into account. However, the large dynamism that appears in the new streaming oriented applications creates a large gap between the worst case and the average case in resource needs. To handle and exploit this gap during the design process, the so-called *application scenarios* were introduced. This different kind of scenarios groups the possible runtime operation modes of an application that have similar resource needs. These scenarios are incorporated in the decision making process during the embedded system design process.

Figure 1 depicts a design trajectory using use-case and application scenarios. It starts from a product idea, for which the stakeholders define the product's functionality as use-case scenarios. These scenarios characterize the system from a *user perspective* and they are used in a user-centric development process to design both the software and hardware components. In order to optimize the design of the system, the extraction and exploitation of application scenarios augments this trajectory (the bottom gray box within the figure). The run-time behavior of the application can be automatically classified from a resource usage perspective into several application scenarios, where the cost within a scenario is always

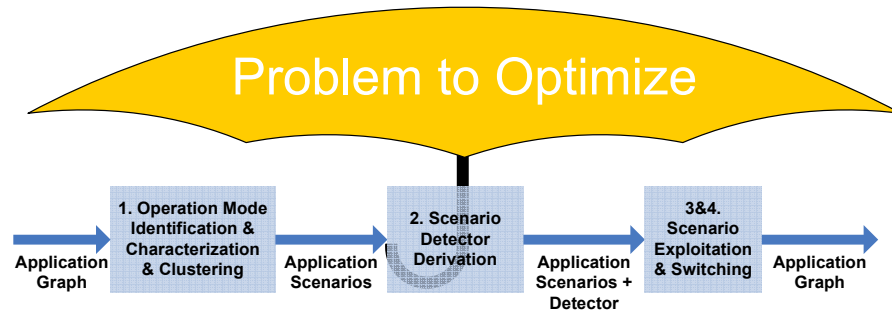


Figure 2: A simple application scenario exploitation methodology.

fairly similar, and between scenarios it is fairly different. For each individual scenario, more specific and aggressive design decisions can be made.

Example: Let us consider that we want to design a small portable MP3 player as a USB stick. At first sight, there are at least two main use-case scenarios: (i) the player is connected to the computer and music files are transferred between them, and (ii) the player is used to play music. These scenarios can be divided in more detailed use-case scenarios, like, for the second one, we may have the song selection, playing, or fast forward scenarios. Let us consider the playing use-case scenario. From the software point of view, this use-case can be split in two different application scenarios: playing songs (i) in mono mode and (ii) in stereo mode. If these scenarios are detected in the application, and the information about them is used during the system design, the system battery lifetime may be increased: in case of playing in mono mode, less computational power is needed, thus a lower supply voltage may be used to meet the timing constraints of the decoding.

3 Application Scenario Based Methodology

Figure 2 shows our methodology based on application scenarios [Gheo06a] that tackles the following four issues:

Identification: given the application, how is it classified into scenarios?

Detection: given a particular run-time situation, to which scenario does it belong?

Exploitation: given a particular scenario, what can be done to optimize the system cost?

Switching: when and how does the application switch from one scenario to another?

All the steps of this methodology are influenced by the design problem that should be optimized. The last two steps, exploitation and switching, depend more strongly on the problem to solve. Moreover, there is earlier work that addresses them. The first two steps, identification and detection could be handled in more general way and these are the ones that we mainly tackle in our work.

Both, identification and detection steps should be applied both at design and run-time. At design time the possible runtime operation modes are identified, characterized and clustered together in scenarios. Also, a way of detecting these scenarios at runtime is derived and introduced in the application. Besides these, a mechanism that counts the operation

modes and the number of deadline misses that appear in the current instance of system should added in the application. Based on the collected knowledge, this mechanism may adapt at runtime both the set of scenarios (e.g. by merging two scenarios) and the scenario detector.

4 Case Study: A Trajectory for Energy Reduction

We developed a semi-automatic trajectory for discovering, predicting and exploiting application scenarios to reduce the energy consumed by a single task streaming-oriented application on a dynamic voltage scaling (DVS) aware processor. The trajectory is adapted for both hard [Gheo05b, Gheo05a] and soft real-time [Gheo06b] constraints. It starts from an application written in C, as C is the most used language to write embedded systems software, and generates the final energy-aware implementation also in C. Under different scheduling constraints, the energy consumed by an MP3 audio decoder and an H.263 video decoder running on a processor similar to an ARM7TDMI may be reduced with 16% - 50% compared to the state-of-the-art methods.

References

- [Doug04] B. DOUGLASS. *Real Time UML: Advances in the UML for Real-Time Systems*. Addison Wesley Publishing Company, 2004.
- [Gheo05a] S. GHEORGHITA, T. BASTEN, AND H. CORPORAAL. Intra-task Scenario-aware Voltage Scheduling. In *Proc. of the International Conference on Compilers, Architecture and Synthesis for Embedded Systems (CASES2005)*, pages 177–184, San Francisco, CA, USA, September 2005. ACM Press, NY, USA.
- [Gheo05b] S. GHEORGHITA, S. STUIJK, T. BASTEN, AND H. CORPORAAL. Automatic Scenario Detection for Improved WCET Estimation. In *Proc. of the 42nd Design Automation Conference DAC*, pages 101–104, Anaheim, CA, USA, June 2005. ACM Press, NY, USA.
- [Gheo06a] S. GHEORGHITA, T. BASTEN, AND H. CORPORAAL. Application Scenarios in Streaming-Oriented Embedded System Design. In *the International Symposium on System-on-Chip (SoC 2006)*, Tampere, Finland, November 2006.
- [Gheo06b] S. GHEORGHITA, T. BASTEN, AND H. CORPORAAL. Profiling Driven Scenario Detection and Prediction for Multimedia Applications. In *Proc. of the International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation (IC-SAMOS)*, pages 63–70, Samos, Greece, July 2006. IEEE Computer Society Press, Los Alamitos, CA, USA.
- [Paul05] J. PAUL. Scenario-Oriented Design for Single Chip Heterogeneous Multiprocessor. In *Proc. of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05) - Workshop 10*, page 227b, 2005.