Chapter 4
Designing Resource-Constrained Embedded Heterogeneous Systems to Cope with Variability

Ian Gray
University of York, UK

Andrea Acquaviva
Politecnico di Torino, Italy

Neil Audsley
University of York, UK

ABSTRACT

As modern embedded systems become increasingly complex, they also become susceptible to manufacturing variability. Variability causes otherwise identical hardware elements to exhibit large differences in dynamic and static power usage, maximum clock frequency, thermal resilience, and lifespan. There are currently no standard ways of handling this variability from the software developer's point of view, forcing the hardware vendor to discard devices that fall below a certain threshold. This chapter first presents a review of existing state-of-the-art techniques for mitigating the effects of variability. It then presents the toolflow developed as part of the ToucHMore project, which aims to build variability-awareness into the entire design process. In this approach, the platform is modelled in SysML, along with the expected variability and the monitoring and mitigation capabilities that the hardware presents. This information is used to automatically generate a customised variability-aware runtime, which is used by the programmer to perform operations such as offloading computation to another processing element, parallelising operations, and altering the energy use of operations (using voltage scaling, power gating, etc.). The variability-aware runtime affects its behaviour according to modelled static manufacturing variability and measured dynamic variability (such as battery power, temperature, and hardware degradation). This is done by moving computation to different parts of the system, spreading computation load more efficiency, and by making use of the modelled capabilities of the system.

DOI: 10.4018/978-1-4666-6194-3.ch004

INTRODUCTION

It is becoming increasingly difficult to efficiently exploit complex Multiprocessor Systems-on-Chip (MPSoC) architectures using existing programming languages and approaches. This is due to two main issues:

1. Modern MPSoC platforms have a very complex programming model, but the languages commonly used to develop software for them (C, C++ etc.) present a very simple view of hardware. This is the "programming model gap."
2. Hardware variability causes systems that were designed as regular architectures to become irregular once they are manufactured, and to change over time.

Commonly used languages such as C, Java and C++ all assume a homogeneous implementation architecture with a uniform, shared memory space. This is incompatible with the application-specific, heterogeneous architectures of MPSoCs – specifically parallelism, non-uniform memory architectues (NUMA) and non-standard communications (i.e. on-chip networks). This problem is compounded when variability is considered.

Variability is the observation that as the manufacture of integrated circuits moves to lower and lower process nodes, the transistors become increasingly variable. This gate-level variation leads to large differences in the performance of the final design. Therefore, multiple copies of the same design may exhibit considerable differences in static and dynamic power consumption, lifespan and clock frequency. A system designed as a homogenous MPSoC will be heterogeneous after manufacturing variability is considered. This is a major challenge for the development of both the hardware and software of future embedded systems.

This chapter begins by describing in detail the kinds of variability that exist in modern embedded systems. The chapter then discusses existing approaches that attempt to mitigate the effects of such variability. The next sections detail the approach taken in the ToucHMore project, an EU FP7 research project which is focussed on the development of variability-aware systems. Finally, the chapter summarises potential areas for future work in this area and concludes.

BACKGROUND: VARIABILITY IN MULTICORE SYSTEMS

Due to the increasing demands placed on modern embedded devices, multicore devices are now commonplace. They are deployed to address the high performance and energy efficiency requirements imposed by audio, video, mobile telephony, and gaming applications. Moreover, multicore systems are becoming widespread in the automotive infotainment and power-train domains; especially in the context of hybrid and electric vehicles where energy efficiency is critical.

Technology scaling has traditionally offered advantages to embedded systems in terms of reduced energy consumption and increased performance without requiring significant additional design effort. Developers could expect performance improvements "for free". However, scaling to and past the 22 nm and 14 nm technology nodes brings a number of problems. Random intra-die process variability, reliability degradation mechanisms, and their combined impact on system-level quality metrics (i.e. power consumption or maximum clock speed) are prominent issues that will need to be tackled in the next few years. In particular, due to aggressive technology scaling, sub-65 nm CMOS technology nodes are increasingly affected by variation phenomena, and multicore architectures are impacted in many ways by the variability of the underlying silicon fabrics (Flamand, 2009) (Tiwari & Torrellas, 2008).

Variability causes significant perturbations to the performance and power consumption of