Trade-offs in the Design of Mixed Hardware-Software Systems - a perspective from industry

Kees A. Vissers
Philips Research
Prof. Holstlaan 4
5656 AA Eindhoven, Netherlands
vissers@natlab.research.philips.com

1. Introduction

Many systems in the field of consumer electronics devices and computers consist of a hardware platform and of software running on that platform. In the design of these systems many trade-offs have to be made. In the design of the hardware platform trade-offs have to be made between programmable components and dedicated components. The programming of the hardware platform also contains many trade-offs. Here a "software architecture" needs to be developed that spans several layers, using well defined interfaces, e.g. Application Programmers Interfaces (APIs). The software contains often device drivers, an operating system, and end-user applications. In embedded systems the end-user can often not program the system directly, e.g. one cannot program the look and feel or contents of your on-screen display of your TV. In practical situations system design is based on many constraints, and seldom starts from scratch. The hardware interface to the system can be given, the models of processors that can be used can be limited, and software interfaces can be required. The trade-offs are in the hardware platform design and in the software design.

2. Application Domains

It is important to make a distinction between several application domains. The outcome of the trade-offs in system design in one application domain are not readily applicable in other application fields. Complete systems contain several application domains and are heterogeneous in nature. The methods and tools for the several domains can be significantly different, but it is required to combine these to solve the heterogeneous system design problem.

In the domain of control applications it is important to handle asynchronous events correctly and sufficiently fast. Programming is often done using an event driven model of computation. The number of events in typical applications lies in the range of several thousands. A modern microcontroller can handle this, where the software design can often be fully dedicated and is synthesized for the application at hand.

In the domain of signal processing it is important to process static or dynamic streams of data. Programming is often done using a dataflow model of computation. In the field of audio processing the data rates are in the tenths of samples per second. In the field of video processing the rates are in the millions of samples per second, and the field of telecommunications spans the full range of audio rates for modems to video rates and higher for ATM and satellite communications. For audio platforms digital signal processors and dedicated solutions using high-level synthesis are current practice. For video processing the rates and corresponding number of operations are very demanding, therefore a mixture of dedicated hardwired synthesized solutions and programmable domain specific solutions (e.g. media processors) are used. A highly underestimated part of system design is the memory, both in size and bandwidth. This is one of the critical components in modern high speed real-time embedded systems, and often a significant part of the total bill of material.

3. Flexibility

What needs to be flexible in the system is a difficult design choice. The flexibility is extremely hard to quantify. The fact that some part of the system is programmable is often essential to the total system. For many problems more than one solution exists with different degrees of programmability at different costs. A requirement like "low power" can have an enormous impact on these trade-offs. These trade-offs are illustrated for some industrial designs for both the hardware architecture and the software architecture.