



## What came first, the software or the chip?

When it comes to developing that next generation low-cost electronic product, companies face an important chipset choice at the very early stages of conception – one that has major consequences on the success of the product. One option is to introduce an application-specific integrated circuit (ASIC), which enables companies to reduce product cost in medium to high volume applications and meet size and power requirements. In lower volume projects, i.e. below hundreds of thousands, an FPGA (field-programmable gate array) solution may be more appropriate, with its alternative position of increased unit cost but lower development cost. Either way, before the correct solution is chosen, the system architecture requirements must be defined and this almost invariably leads to software architecture.

To generalise, most electronic products process information acquired from sensors or communication channels and deliver an output or control a system. In between, a processor core and/or a digital signal processor has to decode and process data using software for task scheduling, input/output interfaces, interrupts, protocol stacks and application programs. Defining this software architecture, its algorithms, its security or encryption techniques, code size,

critical response times and programming environment is every bit as important as defining the hardware architecture; in fact it often dictates how hardware should be designed. However, during the product development process it is amazing how many times software is at first overlooked.

We find that before we can specify hardware such as an ASIC we have to completely understand its software, what information any sensors or wireless systems will provide, what processing power is required and what has to be displayed or controlled. We often gain this understanding by performing an architecture study, doing some experimental modelling in MatLab and by exploring the design space of digital signal processing techniques.

A further set of software requirements often arises from analysing power management, as the ASIC may require a low-power mode for when the product is quiescent and then it must turn on quickly to process information on demand. Other considerations are reliability, self test and fault tolerance, as well as applicable software standards for communications and interoperability. All of this defines the division of processing tasks across the computational elements in a system,

leading to an architecture definition. We believe that FPGA or ASIC development cannot begin until both system and software architectures are fully understood.

This appears consistent with trends in the semiconductor industry, where we observe long-established companies going 'fab-less' or 'fab-light' and reducing their investment in manufacturing. Instead they are focusing on market leadership, which comes when increasingly complex ASICs, or application-specific semiconductor products (ASSPs), fulfil the complete application requirement with a complete System on Chip (SoC). Many such companies have become more market-focused and more active in defining ASIC software architecture to meet the needs of demanding and novel applications. Also, the applications themselves are increasingly defined by software-driven standards for communications protocols, data processing or user interface.

Implementation of an ASIC into silicon is one thing, but it is developing a comprehensive understanding of complete system architecture and the software that has to run on it, that can be the critical success factor, or the X factor, in a modern product design.

**Chris.Turner@CambridgeConsultants.com**