

# How to Choose Semiconductor IP: Embedded Software

Grant Martin

Cadence Design Systems,

2001 Addison Street, Third Floor, Berkeley CA 94704, U.S.A.

gmartin@cadence.com

## Abstract

*Embedded software Intellectual Property (IP) is becoming vital for today's complex System-on-Chips. We first define the notion of Hardware-dependent Software, and then review the multidimensional criteria for choosing ESW IP, including retargetability and portability, flexibility, optimisation, validation and certification.*

## 1. Embedded Software IP

Embedded software intellectual property (ESW IP) is vital to the creation of complex System-on-Chip devices (SoCs), and the products that use these devices. This type of software can be divided into several categories [1]:

- Hardware-dependent SW (HdS)
- Middleware
- Applications SW

In many ways, middleware and applications SW on embedded SoCs do not differ from this kind of software on less integrated systems and thus do not pose the challenges represented by Hardware-dependent SW. HdS represents the basic API into the underlying hardware functionality of an SOC – i.e. the hardware *platform* [2], and includes [1]:

- The hardware-dependent layers of the realtime operating system (RTOS)
- Access to peripheral device drivers
- System debug and maintenance functions
- Chip level configuration software including all device and system registers
- Often, DSP algorithms which are implemented in HW-dependent C code or assembly

The key issue with HdS is its inherent lack of reusability and portability, due to deep hardware dependencies built into it. Strong motivations for reuse, rapid and low risk design time-to-market, have led to one solution to HdS creation and reuse: platform-based design [2], in which pre-assembled, pre-validated, possibly configurable HW-SW architectures and libraries of virtual SW and HW components are created for specific product application domains.

## 2. Criteria for Choice

By its nature, HdS is not inherently portable, but much of it may be retargetable. Some effort has gone into creating retargetable intermediate functional libraries for DSP software, for example – the VSIPL library is a good example [3]. Higher levels of middleware and application software may be available pre-ported to a variety of processor and RTOS platforms, or may be configured with APIs that allow easy retargeting. Another approach to retargeting is automatic code generation from models of embedded software functions captured using UML or more formal mathematical notations [2]. Configurability, giving flexibility, is especially important for RTOS's and networking stacks. The inherent contradiction between portability and optimisation is especially telling for embedded SW and has led in the past to use of optimised assembly code for critical realtime DSP functions, for example. More capable DSPs and use of key hardware accelerators will lessen the need for as much hardware dependent code in the future; automated code generation with target-dependent optimisation will also help. Validation of ESW is in a primitive state, especially in comparison with HW validation although there may be some hope with formal methods combined with automated ESW flows. Finally, certification is completely lacking in this field, although advances in standards for the exchange of ESW IP for SoC would help lay the basis for this.

## 3. References

1. Jack Shandle and Grant Martin, "Embedded Software for SoCs: A New Challenge and a Tested Solution", Unpublished White Paper, Virtual Socket Interface Alliance, December, 2001.
2. Alberto Sangiovanni-Vincentelli and Grant Martin, "Platform-Based Design and Software Design Methodology for Embedded Systems", *IEEE Design and Test of Computers*, Volume 18, Number 6, November-December 2001, pp. 23-33.
3. Randy Janka, *Specification and Design Methodology for Real-Time Embedded Systems*, Kluwer Academic Publishers, November, 2001.