A Hardware-Software Operating System for Heterogeneous Designs*

José Manuel Moya, Francisco Moya, Juan Carlos López
University of Castilla–La Mancha
Escuela Superior de Informática
Paseo de la Universidad, 4. 13071 Ciudad Real, Spain
{jmmoya,fmoya,jclopez}@inf-cr.uclm.es

Abstract

Current embedded systems are made of multiple heterogeneous devices interconnected. These devices present a great variation of functionality, performance, and interfaces. Therefore, it is difficult to build applications for these platforms.

In this paper we present some techniques to introduce component-based methodologies into hardware-software codesign. We make special emphasis on the use of simple, homogeneous interfaces to hide the inherent complexity of current designs. A key contribution is the definition of a HW-SW Operating System that makes system resources available to application developers in a clean, homogeneous way. This greatly simplifies the task of designing complex heterogeneous embedded systems.

As complexity increases, designing embedded systems is becoming too hard. A typical embedded system consist of an heterogeneous network with many different devices. These devices usually have very different interfaces (analog subsystems, microprocessors, field programmable devices, ASICs, etc.). With modern IP-based design strategies, the integration of different IP blocks is becoming the key problem. This confirms that heterogeneity is bad and should be eliminated.

Thus, we are looking for a way to design really complex, distributed, HW-SW embedded systems, hiding the multiple interfaces of the different HW and SW resources through simple and homogeneous interfaces. To the best of our knowledge, no design system meets these requirements. Therefore, we are working on FLECOS[1] to build such a system.

Virtual processors We describe an embedded system as a set of *virtual processors*. A virtual processor is an imaginary microprocessor containing not only the internal functional units of the real microprocessor, but also other external hardware resources. The *virtual instruction set* includes also the operations that these resources implement. And we use a GCC-based HW-SW compiler to map the behavior specification, written in a high-level programming language, into the available resources, using the specified virtual processor as the target architecture.

Adding new hardware resources into the compiler does not change the system interfaces from the user's point of view. However, when the new resource is too complex, this approach does not work, because the compiler is not able to find very complex behavior patterns in the specification.

HW system calls In software systems, the operating system is the "software that securely abstracts and multiplexes physical resources". This definition does not imply that these abstractions should be implemented in software. Thus, we have extended this idea also for hardware-software systems. The advantage is that system designers see these new resources as simple *system calls*.

The implementation details of these system calls are contained in the low-level synthesis tools for the corresponding subsystem.

HW-SW Operating Systems Certainly, we can use the method described above to provide easy-to-use interfaces for all the possible resources we can add to our system, but *many* easy-to-use interfaces are not so easy to use. We have to provide uniform interfaces for all the available resources to really simplify the design process.

Thus, we have created a HW-SW operating system following the simple interface of the *Off++ microkernel*[2]. There is only one abstraction for all the resources: the Box. Everything is represented as a box: a display, an A/D converter, a TCP/IP stack. A box is typed, but can be implicitly converted to other types in a similar spirit to objects in most object-oriented programming languages.

It contains only three methods: `copy` copies data from a source to a target location, `share` allows to share resources, and `select` provides a way to access to the low-level details of a particular resource.

Not to impose any extra overhead that might limit the effectiveness of this approach, the operating system does not specify the exact semantics of these operations. The details (links, protocols, timing, etc.) are only defined for concrete box types. Copy and share operations can only be used between compatible boxes, but it is always possible to define *hardware type converters*.

*This work has been supported by Spanish CICYT (grant TIC2000-0583-C02-01).

[1] See `http://arco.inf-cr.uclm.es/flecos.html`

[2] See Off++ web site: `http://gsyc.escet.urjc.es/off`