Hardware/Software Partitioning of Operating Systems

Vincent J. Mooney III, School of Electrical and Computer Engineering
Georgia Institute of Technology, Atlanta, GA, USA, {mooney}@ece.gatech.edu

1 Introduction

Traditionally, an Operating System (OS) implements in software basic system functions such as task/process management and I/O. Furthermore, a Real-Time Operating Systems (RTOS) has also been implemented in software to manage tasks in a predictable, real-time manner. However, with System-on-a-Chip (SoC) architectures similar to Figure 1 becoming more and more common, OS and RTOS functionality need not be implemented solely in software. Thus, partitioning the interface between hardware and software for an OS is a new idea that can have a significant impact.

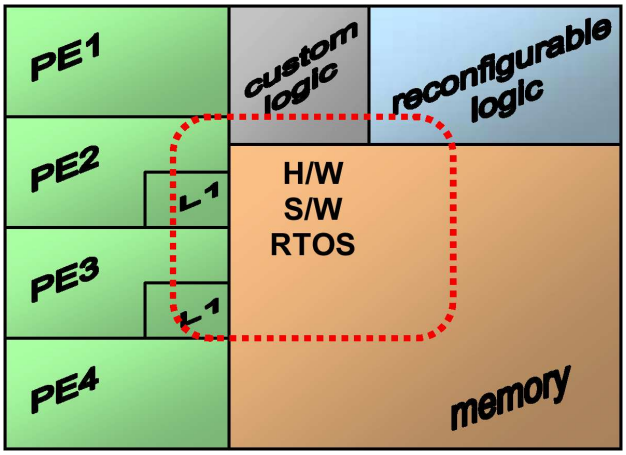

Figure 1: Sample SoC Architecture

We present the δ hardware/software RTOS generation framework for System-on-a-Chip (SoC). We claim that current SoC designs tend to ignore the RTOS until late in the SoC design phase. In contrast, we propose RTOS/SoC codesign where both the multiprocessor SoC architecture and a custom RTOS (with part potentially in hardware) are designed together. Thus, this paper introduces a hardware/software RTOS generation framework for customized design of an RTOS within specific predefined RTOS services and capabilities available in software and/or hardware (depending on the service or capability).

2 Approach

Our framework is designed to provide automatic hardware/software configurability to support user-directed hardware/software partitioning.

A Graphical User Interface (GUI) (see Figure 2) allows the user to select desired RTOS features most suitable for the user's needs [6, 8]. Some RTOS features have both hardware and software versions available.

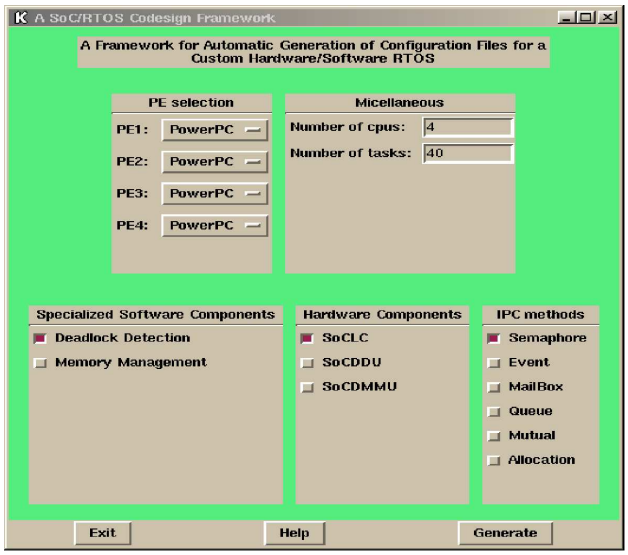

Figure 2: Graphical User Interface for the δ Framework

2.1 Methodology

Figure 3 shows a novel approach to automating the partitioning of a hardware/software RTOS between a few pre-designed partitions. The δ Hardware/Software RTOS generation framework takes as input the following four items:

- Hardware RTOS Library
  This hardware RTOS library currently consists of SoCLC, SoCDDU and SoCDMMU [1, 2, 7, 3, 4, 5].
- Base System Library
  The base system library consists of basic elements such as bus arbiters and memory elements such as various caches (L1, L2, etc.). Furthermore, we also need in the base system library I/O pin descriptions of all processors supported in our system.
- Software RTOS Library
  In our case, the software RTOS library consists of Atalanta [9].

1530-1591/03 $17.00 © 2003 IEEE

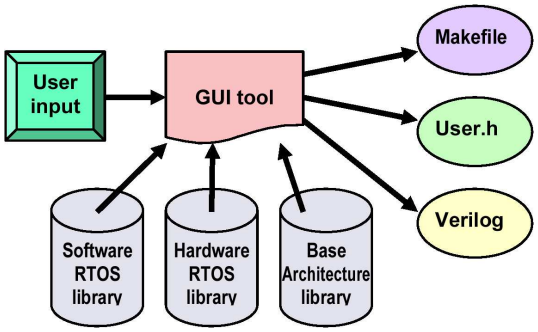

Figure 3: Flow of automatic generation of configuration files

- User Input
  Currently the user can select number of processors, type of each processor (e.g., PowerPC 750 or ARM9TDMI), deadlock detection in hardware (SoCDDU) or software, dynamic memory management in hardware (SoCDMMU) or software, a lock cache in hardware (SoCLC), and different Inter-Procedure Call (IPC) methods. (Note that while all the IPC methods are partly implemented in software, the IPC methods might also depend on hardware support – specifically, if SoCLC is chosen, then lock variables will be memory mapped to the SoCLC.)

The three files on the right-hand-side of Figure 3 (the `Makefile`, `User.h` and `Verilog` files) show the configuration files output to glue together the hardware/software RTOS Intellectual Property (IP) library components chosen in the multiprocessor SoC specified by the user.

2.2 Target SoC

The target system for the δ Framework is an SoC consisting of custom logic, reconfigurable logic and multiple PEs sharing a common memory, as shown in Figure 1. Note that all of the hardware RTOS components (SoCLC, SoCDDU and SoCDMMU) have well-defined interfaces to which any PE – including a hardware PE (i.e., a non Von-Neumann or non-instruction-set processing element) – can connect and thus use the hardware RTOS component's features. In other words, both the custom logic and reconfigurable logic can contain specialized PEs which interface to the hardware RTOS components.

Figure 4 shows the generation of five different hardware/software RTOSes based on five different user specifications (specified using Figure 2).

3 Conclusion

We have briefly discussed some of the issues involved in partitioning an RTOS between hardware and software. Our claim is that SoC architectures can only benefit from early codesign with a software/hardware RTOS to

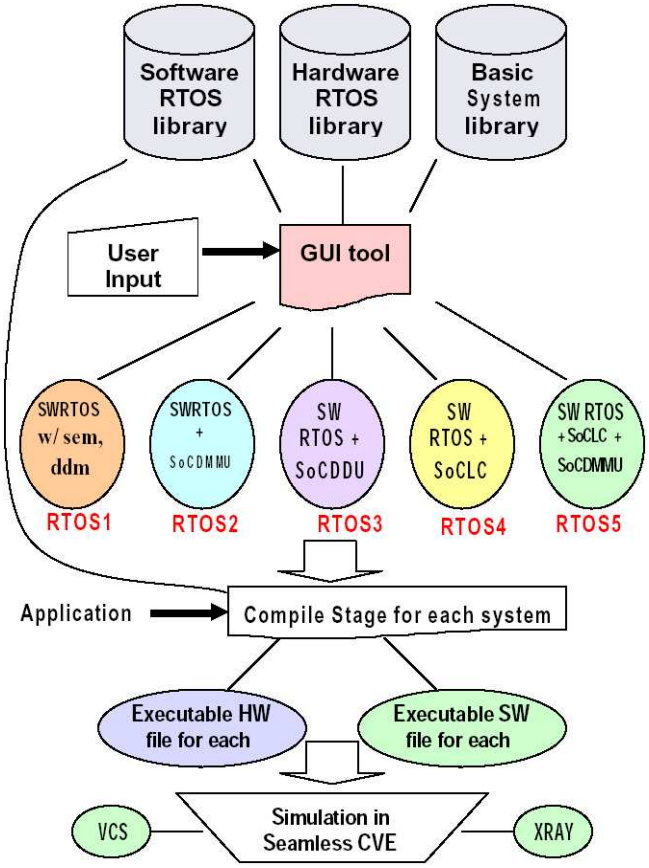

Figure 4: Five Custom Hardware/Software RTOS Examples and Simulation

be run on the SoC.

References

[1] B. Saglam (Akgul) and V. Mooney, "System-on-a-Chip Processor Support in Hardware," *Proceedings of the Design, Automation and Test in Europe Conference (DATE'01)*, pp. 633-639, March 2001.

[2] B. S. Akgul, J. Lee and V. Mooney, "System-on-a-Chip Processor Synchronization Hardware Unit with Task Preemption Support," *International Conference on Compilers, Architecture and Synthesis for Embedded Systems (CASES'01)*, pp. 149-157, November 2001.

[3] P. Shiu, Y. Tan and V. Mooney, "A Novel Parallel Deadlock Detection Algorithm and Architecture," 9^{th} *International Symposium on Hardware/Software Codesign (CODES'01)*, pp. 30-36, April 2001.

[4] M. Shalan and V. Mooney, "A Dynamic Memory Management Unit for Embedded Real-Time System-on-a-Chip," *International Conference on Compilers, Architecture and Synthesis for Embedded Systems (CASES'00)*, pp. 180-186, November 2000.

[5] M. Shalan and V. Mooney, "Hardware Support for Real-Time Embedded Multiprocessor System-on-a-Chip Memory Management," 10^{th} *International Symposium on Hardware/Software Codesign (CODES'02)*, pp. 79-84, May 2002.

[6] V. Mooney and D. Blough, "A Hardware-Software Real-Time Operating System Framework for SOCs," *IEEE Design & Test of Computers*, Volume 19, Issue 6, pp. 44–51, November-December 2002.

[7] B. E. S. Akgul and V. Mooney, "The System-on-a-Chip Lock Cache," *International Journal of Design Automation for Embedded Systems*, Vol. 7, No. 1-2, pp. 139–174, September 2002.

[8] J. Lee, K. Ryu and V. Mooney, "A Framework for Automatic Generation of Configuration Files for a Custom Hardware/Software RTOS," *Proceedings of the International Conference on Engineering of Reconfigurable Systems and Algorithms (ERSA'02)*, pp. 31-37, June 2002.

[9] D. Sun, et. al, *Atalanta: A New Multiprocessor RTOS Kernel for System-on-a-Chip Applications*, Technical Report GIT-CC-02-19, http://www.cc.gatech.edu/pubs.html, Atlanta, GA, 2002.