Demo Abstract: MSPsim – an Extensible Simulator for MSP430-equipped Sensor Boards

Joakim Eriksson, Adam Dunkels, Niclas Finne, Fredrik Österlind, Thiemo Voigt, Nicolas Tsiftes
Swedish Institute of Computer Science
{joakime,adam,nfi,fros,thiemo,nvt}@sics.se

***Abstract*— Software development for wireless sensor networks is a challenging and time consuming task. The resource limited hardware with limited I/O and debugging abilities combined with the often cumbersome hardware debugging tools makes debugging on the target hardware difficult. We present MSPsim, an extensible sensor board platform and MSP430 instruction level simulator. MSPsim is intended to be used for reducing development and debugging time by allowing low-level and fine grained instrumentation of various aspects of software execution. The use of a simulator also enables development and testing without access to the target hardware.**

I. Introduction

Due to the distributed nature of sensor networks and resource-constraints of sensor nodes, code development for wireless sensor network is a challenging and time consuming task. Furthermore, the application development and debugging tools are still cumbersome.

One of the most commonly used methods for debugging sensor nodes is using on-chip emulation via JTAG that makes it possible to single-step and debug a running application on the target hardware. This is useful for understanding execution patterns, stack usage, etc, but less useful for debugging communication, sensor drivers and other timing sensitive parts of the application.

For the development of wireless sensor network applications, system simulators exist that simplify the development of algorithms and enable researcher to study the algorithms' behaviour and interaction in a controlled environment [1].

Cross-level simulation enables simultaneous simulation at different levels of the sensor network and hence supports simultaneous low-level debugging and application development [2]. For cross-level simulation of our MSP430-based sensor node platforms we required an extensible instruction level simulation. Towards, this end, we designed and implement MSPsim. As Avrora [3] MSPsim is a sensor network simulator simulating nodes at the instruction-level, but for the MSP430. Unlike ATEMU that emulates the operations of individual nodes and simulates communication between them [4], MSPsim is designed for instruction-level simulation and for integration with COOJA's cross-level simulation environment.

The contribution of this paper is MSPsim, an extensible instruction level simulator for the MSP430 microcontroller that can be used as a component in a larger sensor network simulation system supporting cross-level simulation [2]. For this reason MSPsim is designed to run multiple instances of the simulator in a single process unlike other MSP430 simulators such as the GDB MSP430 simulator [5]. MSPsim also contains a sensor board simulator that simulates hardware peripherals such as sensors, communication ports, LEDs, and sound devices such as a beeper. The design of MSPsim, together with its implementation in Java, makes it easy to adapt the simulator to new sensor boards.

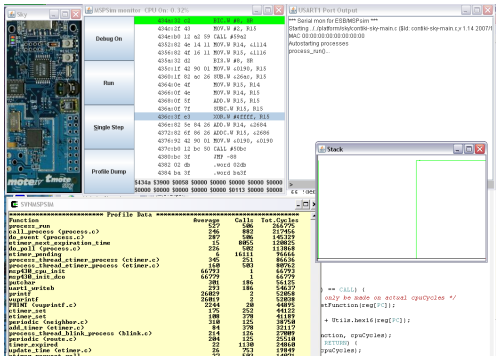

Fig. 1. MSPsim simulating Contiki's Blinker application on a Sky mote.

II. The MSPsim Simulator

The MSPsim is a Java-based instruction level simulator for the MSP430 microcontroller that simulates unmodified target platform firmware. MSPsim is an instruction-level simulator which made it easy to achieve accurate timing simulation. Further, MSPsim can load and run unmodified target platform firmware files in IHEX and ELF format. The simulator is easily extensible with peripheral devices making it possible to simulate various types of MSP430 based sensor nodes. It is also easy to add instrumentation for monitoring the execution of the application.

In addition to simulate the MSP430 and sensor board hardware, MSPsim can show a graphical representation of the sensor board in an on-screen window. LEDs on the sensor board are displayed using the correct colors. Figure 1 shows the graphical output from MSPsim simulating a Sky mote.

The graphical output and input (buttons) of the sensor board hardware combined with UART/Serial output allows a system designer to visually verify that an application is executing correctly by inspection of the LEDs and output over the serial interfaces.

MSPsim have built-in support for setting break-points, read/write monitoring and C-level profiling.

A. Sensor Board Simulation

At SICS we are working with the ESB [6] and the Telos Sky [7] platforms, which both use the MSP430 microcontroller. Therefore, one of the design objectives of the MSPsim simulator is to simplify the adaptation to different types of sensor node platforms. To add support for a new sensor node platform only implementations of peripherals such as sensors, actuators such as beepers or LEDs, and radio and communication peripherals are needed. The implementation of those peripherals are typically relatively easy to make as many of them do not need to conform to strict timing requirements. Figure 2 shows the complete MSPsim simulation system with an MSP430 microcontroller and connected peripherals.

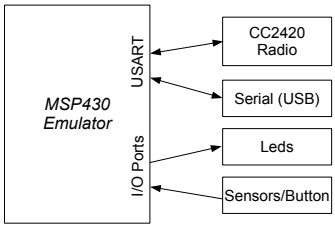

Fig. 2. An MSPsim simulation with an MSP430 microcontroller and connected peripherals.

To illustrate how a peripheral is implemented in MSPsim, Figure 3 contains a complete MSPsim serial peripheral class. The class constructor attaches itself as a listener to the USART object. When the firmware running on the simulated MSP430 writes data to the USART, the dataReceived method of the listener is invoked. In this example, the dataReceived method simply prints out the produced character on screen.

```
public class SerialMonitor {
  public SerialMon(USART usart) {
     usart.setUSARTListener(this);
  }

  public void dataReceived(USART source, int data) {
     System.out.print((char)data);
  }
}
```

Fig. 3. Implementation of a serial output device class attached to a USART

```
USART usart = (USART) cpu.getIOUnit("USART 1");
serial = new SerialMon(usart);
```

Fig. 4. Creating a serial data monitor and attaching it to serial port 1 (USART 1) in MSPsim

Figure 4 shows how a sensor board simulation platform connects the MSP430 USART 1 serial port with a the serial monitor from Figure 3.

III. Evaluation

To evaluate the extensibility of MSPsim we measure the number of interfaces that must be implemented when adding support for a new sensor board in MSPsim. The measurements in the Table I show the amount of interfaces that the ESB sensor board platform implements. Certain peripherals that are present on the ESB are not yet included in the simulator: EEPROM, real-time clock, and active IR. The table shows that the amount of interfaces that need to be implemented for a sensor board is small; only one or two methods need to be implemented for each peripheral. Most peripherals only need a single method for either reading or writing to the peripheral. The interface for the radio is slightly more complex as it requires two write interfaces, one for configuration and one for the data to be transmitted.

TABLE I
NUMBER OF INTERFACES IMPLEMENTED BY THE ESB SIMULATOR.

Peripheral	Read value	Write value
LED	0	1
Beeper	0	1
Digital sensor (PIR, Vibration)	1	0
Analog sensor (Mic, RSSI)	1	0
Radio	1	2
Serial (RS232)	0	1

IV. Conclusions

In this paper we have presented MSPsim, a simulator for MSP430 based sensor nodes. MSPsim is extensible in that adapting the simulator to new sensor boards requires not more than the implementation of a few Java classes. If the sensors and other chips on the new board are already implemented even less work is involved. The source code of MSPsim is available from sourceforge at:

http://sourceforge.net/projects/mspsim/

Acknowledgments

This work was partly financed by VINNOVA, the Swedish Agency for Innovation Systems.

References

[1] P. Levis, N. Lee, M. Welsh, and D. Culler, "Tossim: accurate and scalable simulation of entire tinyos applications," in *Proceedings of the first international conference on Embedded networked sensor systems*, 2003, pp. 126–137.

[2] F. Österlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt, "Cross-level sensor network simulation with cooja," in *Proceedings of the First IEEE International Workshop on Practical Issues in Building Sensor Network Applications (SenseApp 2006)*, Tampa, Florida, USA, Nov. 2006.

[3] B. Titzer, D. Lee, and J. Palsberg, "Avrora: scalable sensor network simulation with precise timing," in *IPSN '05: Proceedings of the 4th international symposium on Information processing in sensor networks*, 2005.

[4] J. Polley, D. Blazakis, J. Mcgee, D. Rusk, and J. S. Baras, "Atemu: a fine-grained sensor network simulator," 2004, pp. 145–152.

[5] D. Diky and C. Liechti, "The GCC toolchain for the Texas Instruments MSP430 MCUs," http://mspgcc.sourceforge.net/ Visited 2006-11-11.

[6] J. Schiller, H. Ritter, A. Liers, and T. Voigt, "Scatterweb - low power nodes and energy aware routing," in *Proceedings of Hawaii International Conference on System Sciences*, Hawaii, USA, 2005.

[7] J. Polastre, R. Szewczyk, and D. Culler, "Telos: Enabling ultra-low power wireless research," in *Proc. IPSN/SPOTS'05*, Los Angeles, CA, USA, Apr. 2005.