THE EXTENSIBLE SENSOR PLATFORM

David Pointer; Volodymyr Kindratenko; Paul Zawada; Meenal Pant
(National Center for Supercomputing Applications, University of Illinois at Urbana-Champaign, Champaign, IL; email: pointer@ncsa.uiuc.edu)

ABSTRACT

The Extensible Sensor Platform (ESP) combines a sensor bus with a Software Defined Radio (SDR). The first field programmable gate array (FPGA) implementation of the ESP supports the Phillips Semiconductors I^2C sensor bus. The SDR provides the system with a programmable, flexible means of communicating sensor data over wireless data network available in a given application space. A wide band antenna and pluggable radio frequency (RF) front-end analog hardware with a common digitized intermediate frequency (IF) allows system deployment in a wide range of RF spectrum. The main advantage of the ESP is that it gives researchers the ability to programmatically adapt their wireless sensor data communication network fabric to different portals available within a given application space: a cellular tower, packet radio, 802.11, Bluetooth, etc. Additionally, an FPGA implementation allows researchers to deploy an ESP with multiple data channels and separate control and data channels. This paper describes the ESP architecture as well as the first FPGA based implementation of the ESP.

1. INTRODUCTION

The Extensible Sensor Platform (ESP) project merges sensor technology and Software Defined Radio (SDR) technology. This platform may be applied to a wide range of applications and environments in which the use of a conventional sensor system with a fixed radio (such as Berkeley mote [1] [2] derivations) is limited. Additionally, the ESP is dedicated to supporting standard sensor interfaces, such as the Phillips Semiconductors I^2C bus [3] and the newly emerging IEEE 1451.4 smart sensor interface [4] [5].

Environment monitoring, be it animal habitat monitoring, weather observation, or enemy territory surveillance, are examples of where the ESP has great potential. At the current state of the art, each environment monitoring system is designed for a specific, fixed environment application in mind. When additional sensory or communication capabilities are required, the entire system is re-engineered to include the needed sensor, conditioning electronics, or the radio-related hardware. The ESP, on the other hand, requires no hardware changes, other than attaching the actual sensor device via the standard sensor interface. Instead of hardware changes, the ESP requires loading new software to support the new sensor or to change the characteristics of the SDR for a new data communication channel. Once deployed, an ESP-based environment monitoring system can be upgraded over the radio waves to include a new communication protocol or a new data encryption algorithm. It can automatically discover available nearby communication infrastructure, such as a cellular telephone tower or other ESP devices in the neighborhood. In the latter example, the ESP will have the ability to form a self-organizing ad-hoc network and pass the sensor data on to an ESP in the network that does have access to existing network communication infrastructure.

The SDR portion of the ESP is what gives the platform the ability to mesh with any available wireless data communication fabric. It allows the platform to utilize existing data communication infrastructure in a given application space to transport sensor data back to the data collection servers.

A standard sensor bus is what gives researchers the capability of changing or adding sensors without redesigning hardware. This is useful in application spaces in which the user requires an iterative approach to determining the correct mix and types of sensors for a given environment.

This paper will describe the architecture of the ESP as well as the first FPGA implementation of a simple sensor data transmitter and receiver. This first implementation serves as a proof of concept prototype - a step towards a truly flexible, heterogeneous, multi-modal wireless sensor platform.

2. ARCHITECTURE

Any sensor network may be viewed abstractly as a data generation sensor, a network connection to transport that data, and the data collection server endpoint (Figure 1). The

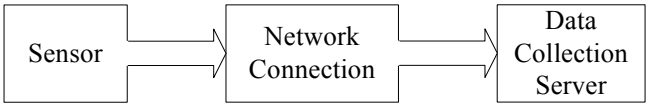

Figure 1 – Abstract Sensor Network

ESP builds upon this abstraction (Figure 2) by providing a standard sensor bus as a common sensor interface, allowing the use of many different types of commercial off the shelf sensors. The ESP network connection to the data collection servers consists of two parts: an existing data network infrastructure and the SDR portion of the ESP. The SDR provides programmable connectivity into the application space's existing data network infrastructure.

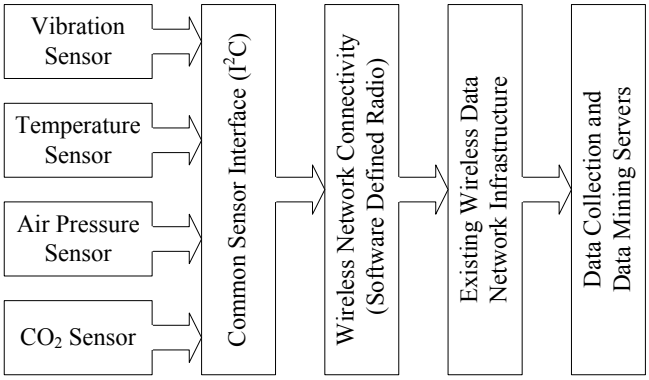

Figure 2 – ESP Sensor Network

The block diagram level view of the ESP architecture is detailed in Figure 3 showing the analog RF front-end hardware, the data converters, the sensor bus, and the FPGA based SDR and control sections of the ESP.

We chose the Phillips Semiconductors I^2C [3] standard

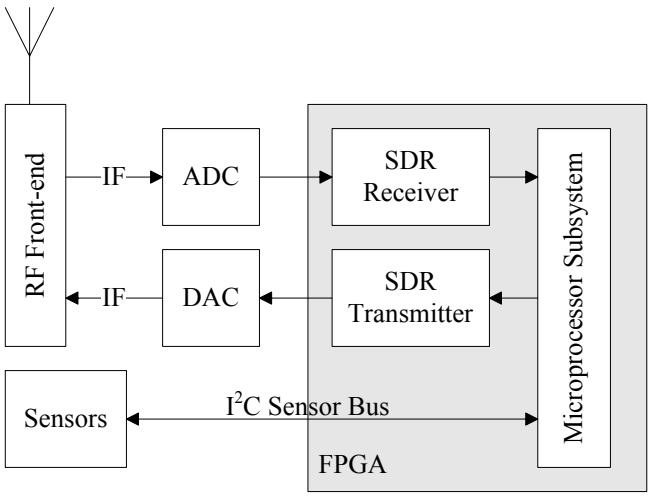

Figure 3 – ESP Block Diagram

for interfacing with sensors. I^2C is a ubiquitous, simple two-wire (clock and data) bi-directional bus, works well with hybrid 5 volt and 3.3 volt systems [6], and is well-supported by many different sensor manufacturers. Another factor that led us to choose the I^2C bus over other sensor interfaces is that the radio front-end tuning and gain control used in our design support an I^2C interface. We chose to operate the I^2C bus at the slower 100 KHz rate and with the older 7 bit addressing since these parameters are supported by more sensors than the new 400 KHz rate and 10 bit addressing.

Ideally, the SDR portion of the ESP would directly digitize some segment of the radio frequency (RF) spectrum at the antenna. This ideal system cannot be easily implemented since modern analog to digital converters (ADC) and digital to analog converters (DAC) are fast enough to directly convert only a very small portion of the RF spectrum [7]. Instead, an analog conversion stage (RF Front-end) is inserted between the antenna and the data converters. This conversion stage uses analog multiplication [8] to mix the RF frequency down to a lower intermediate frequency (IF) or to up convert the IF to the higher RF carrier frequency [7] [9] [10].

The analog RF front-end hardware is limited to specific regions of the RF spectrum by design. This inflexibility is addressed in the ESP by allowing any analog RF front-end hardware to work with the ESP as long as it can function with a 10.7 MHz IF.

ESP system control is provided by a microprocessor subsystem in the FPGA. A minimal microprocessor subsystem for the ESP includes a microprocessor, program storage memory, random access memory, timers tied to the system clock, bit-wise binary input-output for control, and an I^2C master bus controller.

3. IMPLEMENTATION

The goal of the first ESP implementation was to prototype two 900 MHz ESP systems, one a transmitter with a temperature sensor, and the other with a receiver and LED. The LED would be turned on or off based on the temperature data from the transmitter and a set threshold value.

The prototype design target was the Nallatech BenONE and BenADDA boards. The BenADDA provides two ADCs, two DACs, and a Xilinx Virtex 2V3000-4 FPGA. The BenONE provides motherboard support for the BenADDA, including clocks, power, and a USB download interface.

We used the Matlab/Simulink software tools for high-level digital signal processing (DSP) design for the SDR part of the ESP. The Xilinx System Generator for DSP tool converted the Matlab/Simulink design into Xilinx design specific files. The Xilinx XPS tool generated Xilinx design

specific files from our soft core microprocessor system design. We also wrote VHDL code for certain aspects of the design. The Xilinx ISE Foundation tool set combined rendered all of the Xilinx design specific files and VHDL into a single bit file used to program the FPGA.

The 900 MHz RF front-end transceiver was provided by the National Center for Advanced Secure System Research (NCASSR) SDR project [15]. Tuning and automatic gain control (AGC) is controlled by the ESP's microprocessor systems over the I^2C bus shared with the sensor.

The transmitter prototype reads a temperature sensor and transmits the sensor data via amplitude shift keying (ASK) modulation. ASK digital data transmission [16], in its simplest form, uses a transmitter carrier-on condition of duration t to represent a digital logic '1' and a transmitter carrier-off condition of duration t to represent a digital logic '0'. The double sideband (DSB) form of ASK is represented by

$$s(t) = \frac{A}{2}[1 + m(t)]\cos \omega_c t$$

where $m(t)$ is the modulating signal (-1 or 1), A is the amplitude, and ω_c is the carrier frequency in radians.

In order to improve the ASK receiver's ability to discriminate between carrier-off representing a logic '0' and carrier-off representing a transmitter turned off, the digital bits are encoded. In this design, a "sub-bit-time" is 375 micro-seconds. Six sub-bit-times make one bit-time. Given these timings, the digital data bits are encoded via carrier-on and carrier-off states as shown in Figure 4.

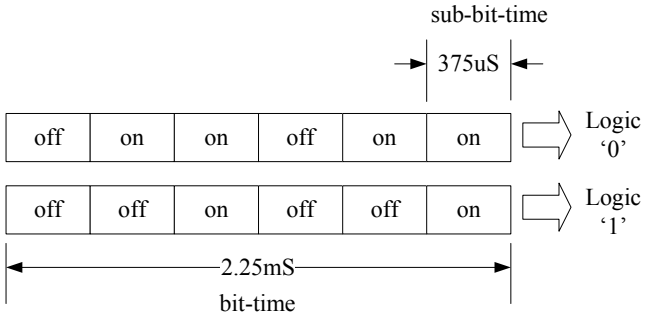

Figure 4 – Digital Bit Encoding

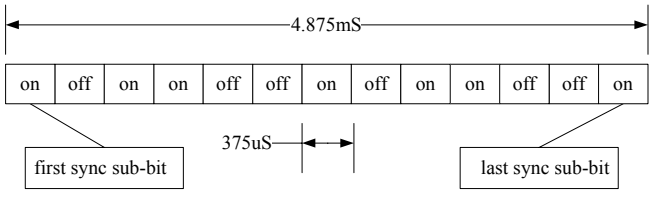

Figure 5 – Carrier On / Carrier Off Sync Pattern

In order to allow an ASK receiver to synchronize with an ASK transmitter, a defined synchronization bit pattern ("sync") is sent out by the transmitter immediately before the encoded data bits. In this design, the carrier-on and carrier-off sync pattern is shown in Figure 5.

Note in Figure 5 there are 13 sub-bit-times of the carrier-on state or the carrier-off state in this pattern. Note also that the first sub-bit of the pattern is a carrier-on, which notifies an ASK receiver to start keeping track of what it is receiving. The actual digital data message payload starts immediately after the last sync sub-bit. This bit encoding and synchronization pattern is derived wholly from Holtek Semiconductor's HT-680 Encoder [17].

Figure 6 shows a block diagram of the prototype's data transmitter. The two timers in the design are used for time references: one is used as an interval timer for checking the temperature sensor, and the other is used as a 375 micro-second sub-bit timing reference when the design is

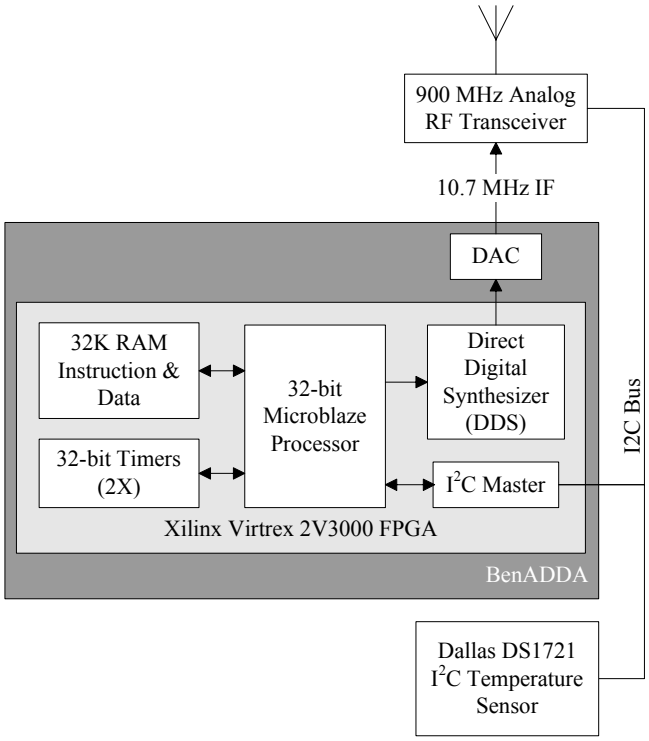

Figure 6 – ESP Prototype Sensor Data Transmitter

transmitting data. The processor's program controls the carrier-on and carrier-off conditions via a single bit enable control line to the Direct Digital Synthesizer (DDS). This DDS is provided by Xilinx as a single black box functional unit for the Matlab/Simulink tools. When enabled by the microprocessor, the DDS generates a digital data stream of 14 bit fixed point number representing a 10.7 MHz sinusoid. The output of the DDS is sampled at 25 MHz by the BenADDA's DAC and passed through a passive analog

regeneration filter (not shown) to the 900 MHz transceiver board and out the antenna. The I²C bus is under the control of the design's microprocessor and is used to tune the transmitter frequency, set the transmitter AGC, and read the temperature sensor.

Figure 7 shows a block diagram of the prototype's data receiver. The timer in the design is used as a sample time reference on the input data stream from the ASK data receiver. The analog 900 MHz transceiver passes the 10.7 MHz IF through a passive analog input filter (not shown). This analog IF is digitized by the BenADDA's ADC using a sample rate of 25 MHz. The digitized IF is passed to the ASK data receiver, described below. The I²C bus is under the control of the design's microprocessor and is used to tune the receiver frequency and set the receiver's AGC.

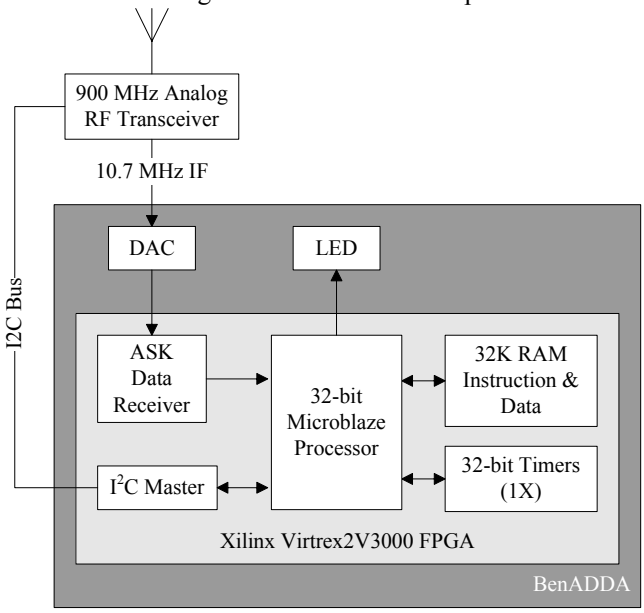

Figure 7 – ESP Prototype Sensor Data Receiver

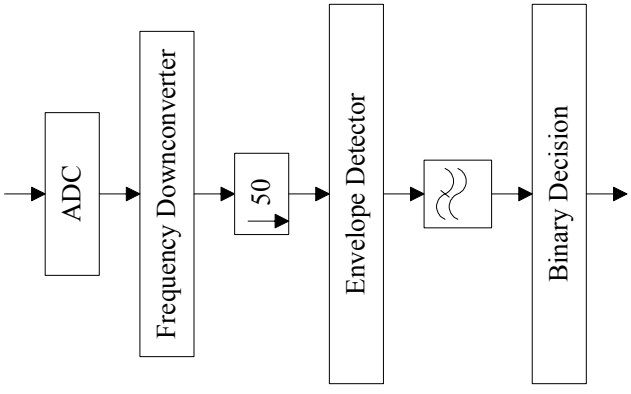

Figure 8 – ASK Data Receiver Block Diagram

Figure 8 shows a block diagram of the ASK data receiver section of the prototype sensor data receiver. The filtered 10.7 MHz IF from the analog 900 MHz transceiver board is digitized at a 25 MHz sample rate. The IF is down-converted and the sample rate is reduced by a factor of 50. This ASK receiver uses non-coherent detection by applying an absolute value type of envelope detector to the fixed point data stream. The output of the detector is presented to a low pass filter, which detects the presence or absence of signal energy. With $s_0 = 0$ and $s_1 = A\cos\omega_c t$, the output of the filter at the sampling time T is

$$y_1(t) = \int_0^T s_1^2(t)dt = \frac{A^2 T}{2} \quad \text{and} \quad y_0(t) = 0,$$

where A is the amplitude, and ω_c is the carrier frequency in radians [16]. The output of the low pass filter is sampled, and a detected energy level below a set threshold is a logic '0' and a detect energy level below a set threshold is a logic '1'. The recovered data bit stream is fed to the microprocessor in the receiver design.

This design easily met the 25 MHz system clock timing requirements using the slowest (-4) speed version of the Xilinx 2V3000. In terms of space utilization, the transmitter and receiver implementations each occupied 16% of the 2V3000 FPGA.

4. FUTURE WORK

Much needs to be done before the working prototype can be converted into an integrated single-board design suitable for mass-production. For a first step in this direction, we currently plan to move the ESP off its current prototype development hardware host onto a tightly integrated, minimal chip count board design.

We also plan to make the ESP easier to program and use. As part of this effort, we plan to implement various common radio functional blocks in an FPGA that may be parameterized and connected together via high level commands. A set of interpreted commands will allow a programmer to select radio functional blocks (objects) already present in an FPGA. The commands will, in effect, allow a programmer to specify a wireless communication system in much the same manner as the GNU Software Radio distribution [11]. The GNU Software Radio project instantiates functional block-objects for a given radio design and link the objects together. An alternative approach may be to implement a subset of the Software Control Architecture (SCA) [12] developed by the U.S. Department of Defence Joint Tactical Radio System (JTRS) project [13].

We are currently investigating the proposed IEEE 1451.4 standard [4] [5] for use in the ESP as an additional sensor interface. The IEEE1451.4 standard will provide a plug-and-play approach to adding sensors to the ESP which

will make the ESP easier to use. We plan to provide both the I^2C and the IEEE-1451.4 interfaces.

Longer term, we are planning to adopt much of the public domain work on mobile ad-hoc sensor networking released by the Naval Research Laboratory's PROTEAN group [14].

5. ACKNOWLEDGEMENT

This work was performed at the National Center for Advanced Secure System Research (NCASSR) and funded by the Office of Naval Research (ONR) grant N00014-3-1-0765.

6. REFERENCES

[1] J. Hill and D. Culler, "A wireless embedded sensor architecture for system-level optimization", Technical report, Computer Science Department, University of California at Berkeley, 2002.

[2] J. Hill, R. Szewczyk, A. Woo, S. Hollar, and D.C.K. Pister. "System architecture directions for networked sensors", Proceedings of ACM SIGMOD, San Diego, CA, June 2000.

[3] Phillips Semiconductors, The I^2C Bus Specification, Version 2.1, January 2000.

[4] Standard for a Smart Transducer Interface for Sensors and Actuators - Transducer to Microprocessor Communication Protocols and Transducer Electronic Data Sheet (TEDS) Formats, IEEE Standard 1451.2-1997.

[5] A Smart Transducer Interface for Sensors and Actuators – Mixed-mode Communication Protocols and Transducer Electronic Data Sheet (TEDS) Formats, Draft IEEE Standard P1451.4.

[6] Philips Semiconductors, "Bi-directional level shifter for I^2C bus and other systems", application note AN98055, 1997.

[7] Jeffrey H. Reed, Software Radio, Prentice Hall PTR: Upper Saddle River, NJ, 2002, ch. 1, pp. 169-171, pp. 377-379.

[8] The ARRL Handbook for Radio Communications, 80th ed., Amateur Radio Relay League, Newington, CT, 2003, pp. 15.1-15.4, pp. 16.37–16.40.

[9] Andrew Bateman and Iain Paterson-Stephens, The DSP Handbook: Algorithms, Applications, and Design Techniques, Pearson Education Limited: Essex, England, 2002, cha. 5, cha. 6, p. 620.

[10] Wes Hayward, Rick Campbell, Bob Larkin, Experimental Methods in RF Design, Amateur Radio Relay League: Newington, CT, 2003, p. 10.7.

[11] GNU Software Radio Project web site, http://www.gnu.org/software/gnuradio/

[12] "Software Communications Architecture Specification", Technical report, U.S. Army, 1998.

[13] Joint Tactical Radio System web site, http://jtrs.army.mil/

[14] U.S. Naval Research Laboratory PROTEAN Group web site, http://protean.itd.nrl.navy.mil/

[15] A. Betts, M. Hall, V. Kindratenko, M. Pant, D. Pointer, V. Welch, P. Zawada, "The GNU Software Radio Transceiver Platform", Proceedings of the 2004 Software Defined Radio Technical Conference, November 2004.

[16] David R. Smith, Digital Transmission Systems, 2nd ed., Van Nostrand Reinhold: New York, 1993, pp. 360-363.

[17] Holtek Semiconductor, Inc., HT600/680/6207 3^{18} Series of Encoders Data Sheet, Revision 1.10, January 24, 2003.