Sensor Letters, 2005, vol. 3, no. 2, pp. 174-178.

Mapping a Sensor Interface and a Reconfigurable Communication System to an FPGA Core

Volodymyr Kindratenko[1] and David Pointer
National Center for Supercomputing Applications
University of Illinois at Urbana-Champaign
405 N. Mathews Ave.
Urbana, IL 61801, USA
Phone: (217) 265-0209
Fax: (217) 244-2909
Email: kindr@ncsa.uiuc.edu

Received: April 1, 2005
Accepted: April 11, 2005

ABSTRACT

In this short communication, we present a novel approach for integrating a standard sensor interface and a reconfigurable wireless data communication channel on a single Field Programmable Gate Array (FPGA) chip. In a traditional wireless sensor device, sensor and communication electronics are built as separate components and their integration consists of designing a board to accommodate multiple integrated circuits (ICs). In contrast, the FPGA core allows us to put a standard I²C sensor interface, embedded Digital Signal Processing (DSP) algorithms for Software-Defined Radio (SDR), and a microprocessor control system on a single-chip reconfigurable device. In our experimental FPGA-based sensor platform, the I²C interface is used both to receive data from the temperature sensor and to control the analog Radio Frequency (RF) transceiver board. Amplitude Shift Keying (ASK) modulation and demodulation are implemented using DSP techniques within an SDR framework, and the microprocessor system is used to read sensor data and control various components. The prototype design was deployed on a Nallatech development board containing a Xilinx Virtex 2V3000 FPGA chip and was successfully tested with Dallas Semiconductor DS1721 digital temperature sensor. One instance of the prototype transmitted the temperature red by the sensor while second instance acted upon the received over the air temperature readings by turning LED on or off.

Keywords: Electronic interfaces and data processing; Sensor signal processing electronics; Sensor platform; Field Programmable Gate Array (FPGA); Software-defined radio (SDR).

[1] To whom correspondence should be addressed (kindr@ncsa.uiuc.edu)

TEXT

In a wireless sensor network, sensor systems require some type of wireless data communication fabric. While many examples of wireless sensor devices exist, such as the Berkeley Motes[1,2], these devices have a fixed radio subsystem and a fixed sensor interface that cannot be altered after the system is realized in hardware. We envision a programmable Software Defined Radio[3] (SDR) coupled with a standard sensor data bus and a microprocessor system – all implemented on a single Field Programmable Gate Array[4] (FPGA) core – which yields a flexible wireless sensor platform. For such a device, the target application space drives the sensor interface and communication channel characteristics, not the other way around. In this letter, we present a prototype of such a system, the Extensible Sensor Platform (ESP). The concept of FPGA-based ESP was first introduced in our conference paper[5] that focuses primarily on the SDR implementation on FPGA. In this letter, we report on the complete system and provide results of field tests using temperature sensor. We first describe the hardware components used to develop and test ESP prototype. Next, we describe in details the FPGA implementation followed by experimental results. Finally, we discuss lessons learned and future work directions.

ESP hardware prototype consists of a Xilinx Virtex 2V3000 FPGA, analog and digital domain converters, analog radio frequency (RF) front-end, and an I^2C sensor and control bus. We chose the Nallatech XtremeDSP Development Kit[6] to host the ESP design. The kit consists of two printed circuit boards, i) the BenONE, the host board that provides I/O expansion, power, programmable clock generation, USB interface for FPGA programming file download, and other support hardware, and ii) the BenADDA, the core of the ESP implementation containing two 65 Mega-Sample-Per-Second (MSPS) ADC devices, two 160 MSPS DAC devices, a Xilinx Virtex 2V3000 user FPGA and a Xilinx Virtex 2V80 ADC/DAC clock generation FPGA (Figure 1).

The RF front-end hardware design (Figure 2) is based on the Maxim Integrated Products MAX2460 900 MHz image-reject transceiver IC[7]. This RF front-end was originally designed for use by the National Center for Advanced Secure System Research (NCASSR) Software-Defined Radio Project[8]. The MAX2460 utilizes conventional analog mixing to downconvert a 4 MHz-wide block of 900 MHz spectrum to an intermediate frequency (IF) block centered at 10.7 MHz. For transmission, a 10.7 MHz IF input is upconverted to 900 MHz. The tuning range covers the 902-928 MHz U.S.-unlicensed Industrial-Scientific-Medical (ISM) band. Control of the MAX2460 is provided by I^2C devices on a shared bus.

We chose the Phillips Semiconductors I^2C standard[9] for interfacing with sensors and actuators. 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[10], and is well-supported by many sensor manufacturers. Another factor that led us to choose the I^2C bus over other sensor interfaces is that the RF front-end tuning and gain control devices 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.

The actual ESP FPGA implementation (Figure 3) contains the digital Amplitude Shift Keying[11] (ASK) transmitter and receiver, built from high-level DSP block libraries provided by Xilinx, and a microprocessor subsystem, including an I^2C master controller, built from Xilinx soft core

Intellectual Property (IP) designs. In short, everything in the system that is digital fits into the FPGA. The analog and digital domain converters, the analog RF front-end, and the actual sensors are all external to the FPGA.

We use Simulink from Mathworks[12], Xilinx System Generator and Xilinx Platform Studio[13] tools to create the ESP FPGA design. The VHDL[14] files automatically generated from the Simulink Xilinx block model by System Generator for DSP serve as input files to the Xilinx Integrated Synthesis Environment (ISE) tool. The ISE tool is central to the FPGA design tool flow. It takes input from System Generator for DSP, manually created VHDL source code and the Xilinx Platform Studio tool and generates a single programming bit file for the target FPGA hardware. The Nallatech FUSE software is used to download the FPGA programming bit file into the XtremeDSP Development Kit hardware.

As it was said earlier, everything in the system that is digital fits into the FPGA: a microprocessor subsystem, the I^2C master controller, ASK transmitter and receiver. The FPGA hardware is programmed to execute all these tasks as they would be implemented by separate hardware components. The microprocessor subsystem (Figure 4) controls the ASK SDR transmitter, the I^2C controller, and decodes the bit stream provided by the SDR receiver. All of the blocks in the microprocessor subsystem are Xilinx IP soft core designs. Two timers are used in our design: one is used to provide a sensor sampling timer, the second timer provides a bit timing reference for ASK data. The I^2C controller is attached to external I^2C sensors as well as the control devices for the RF front-end. The microprocessor subsystem and the SDR receiver and transmitter are all clocked at 25 MHz. The 25 MHz system clock is also routed through the 2V3000 FPGA to the 2V80 FPGA on the BenADDA. The 2V80 FPGA is designed to generate the differential clocks that the DAC and ADC require.

Amplitude Shift Keying digital data transmission[11], 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[15] as shown in Figure 5. In order for an ASK receiver to synchronize with an ASK transmitter, a synchronization bit pattern ("sync") is defined and sent out by the transmitter immediately before the encoded data bits. In our design, the sync pattern shown in Figure 5 is used.

The Simulink model of the ASK Data SDR Transmitter consists of only a single device from the System Generator for DSP block set: a Direct Digital Synthesizer (DDS). When enabled, the DDS generates a 10.7 MHz digital sinusoid that is input to the BenADDA DAC. The DDS enable line is controlled directly by the Microblaze microprocessor.

Figure 6 shows a block diagram of the ASK Data SDR Receiver. The digitized 10.7 MHz IF is downconverted to baseband and the sample rate is reduced by a factor of 50. The 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 $y_1(t) = \int_0^T s_1{}^2(t)dt = \frac{A^2T}{2}$ and $y_0(t) = 0$ where A is the amplitude, and ω_c is the carrier frequency in radians[11]. 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 input to the microprocessor subsystem.

The first prototype of an ESP system can read and transmit sensor data to a receiver, which serves as a proof of concept for our FPGA-based sensor interface and reconfigurable communication system. The complete design occupies less than 40% of the Virtex 2V3000 FPGA's resources and could support a 63 MHz clock, which meets the timing requirements of the actual 25 MHz system clock. Below we present measurement results to validate our design.

A Dallas Semiconductor DS1721[16] digital temperature sensor generates sensor data for testing. Every 2 seconds, the most significant 8 bits of temperature data are read by the transmitting ESP's microprocessor, encoded and transmitted. Reading the sensor data from the I^2C bus takes approximately 400 µs. Transmitting the synchronization bit pattern and 8 bits of sensor data takes approximately 23 ms. The receiving ESP's microprocessor is constantly scanning for the rising edge of a sensor data message's first synchronization bit. Once that is detected, the message is read in approximately 23 ms, the synchronization pattern is checked, the temperature data decoded, and an LED is turned on or off based on a preset temperature threshold.

The ASK data SDR transmitter waveform is of little interest as it is simply a 10.7 MHz sinusoid. However, tracing the waveforms through the ASK data SDR receiver (Figure 6), starting at the ADC output, allows us to verify and validate the ASK receiver design. Figure 7a shows the reception of the first 5 bits of the synchronization bit pattern (10110) starting near 1.1 ms. At this resolution, the digitized 10.7 MHz IF appears as solid vertical bars in the figure. After passing through the frequency downconverter, the digitized 10.7 MHz IF waveform is mixed down to baseband (50 KHz in this design). Since the sampling rate of the mixer is the same as the ADC (25 MHz), the baseband signal appears in Figure 7b as a 50 KHz sinusoid superimposed on the higher frequency mixer products. The decimation stage has the effect of reducing the sampling rate from 25 MHz to 500 KHz, which filters out most of the higher frequency mixer products (Figure 7c). The decimation stage also reduces the required clock rate of down-stream parts of the receiver. This has the effect of reducing the number of required taps for a defined filter response, relaxing logic timing requirements, and reducing FPGA power consumption. The receiver's envelope detector simply multiplies its input by -1 if the input is less than 0, or multiplies its input by 1 of the input is greater than or equal to 0. This has the effect of giving the input an offset of 1 and converting the input to positive integers. The output from this operation may be seen in Figure 7d. The output of the receiver's envelope detector still contains some significant frequency components above baseband which are further filtered out

by the order 50 equiripple FIR lowpass filter. Figure 7e shows the resulting filtered output. The filter output is squared up and restricted to magnitudes between 0 and 1 by a simple threshold detector in the bit decision block. The output of this detector becomes the binary bit input to the microprocessor (Figure 7f). Thus, at the end of the ASK receiver sensor data bits are recovered and passed to the microprocessor for interpretation.

So far, we addressed the issue of integrating sensing, processing and communication into a reconfigurable FPGA core to serve as a software-reconfigurable wireless sensor platform. Although the first prototype functioned as we expected, much needs to be done to convert it into a true wireless sensor platform. Perhaps the most critical issue that we even have not begun to address is power consumption. FPGA's high degree of flexibility comes at a high power consumption cost. High-performance ADC and DAC used in our design are very significant power consumers as well. It is very likely however that the FPGA-based wireless sensor platform will not allow us to reach power consumption levels present in ASIC-based designs, such as the Berkeley Motes. For example, our FPGA uses 733 mW of power, the ADC 1300 mW, and the DAC 250 mW. This yields 2283 mW of power consumption without including the analog radio components or sensors. By contrast, one version of the Berkeley Motes, the Mica, uses 100 mW of power in active mode[22], including the processor, sensors, and radio.

Much needs to be done before the developed 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 that also takes in to account power management and power consumption constraints.

Current design allows to send sensor data and to act on the received data (e.g., turn on an LED), but such a simplistic functionality is not sufficient. More general microprocessor functionality is needed to enable a more general ESP functionality, which is closely related with the need to make the ESP easier to program and use. As part of this effort, we plan to implement various common DSP and control functional blocks in FPGA that may be parameterized and connected together via high level commands. A set of interpreted commands will allow a programmer to select radio and command functional blocks (objects) already present in an FPGA. The GNU Software Radio project[17] provides one example how such a configuration protocol can be implemented. An alternative approach may be to implement a subset of the Software Control Architecture[18] (SCA) developed by the U.S. Department of Defence Joint Tactical Radio System (JTRS) project[19]. In principle, any medium access control (MAC) protocol can be programmed as part of FPGA's functionality, but we have not investigated this issue in the framework of our existing ESP design.

The analog RF front-end hardware used in the ESP prototype 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 produce a 10.7 MHz IF.

True plug-and-play sensor flexibility cannot be easily achieved with I^2C interface. We plan to investigate the proposed IEEE 1451.4 standard[20,21] for use in the ESP as an additional sensor interface.

Most of our difficulties, while developing the prototype, were rooted in the development tool environments. The tool vendors seem to imply that little or no hardware or FPGA design experience is necessary; that there is no need to be concerned about the details of FPGA implementation. This was mostly true as long as a design did not stray outside the bounds of a single tool set. If separate tool sets were used to implement a design, the complexity of combining the sub-designs is difficult to manage and actually does require delving into the FPGA implementation details. Another source of difficulty is that FPGA design has a different conceptual modality than software design. An FPGA is not like a computer with seemingly limitless resources – the FPGA target has a very constrained and limited set of specific resources. A developer with a pure software background needs to understand and adopt the limitations peculiar to FPGA design.

ACKNOWLEDGMENT

This work was performed at the National Center for Advanced Secure System Research funded by the Office of Naval Research (ONR) grant N00014-3-1-0765. The authors would like to thank Paul Zawada for his 900 MHz transceiver design.

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. J. Reed, *Software Radio*, Prentice Hall PTR, Upper Saddle River, NJ (2002).
4. W. Wolf, *FPGA-Based System Design*, Prentice Hall PTR, Upper Saddle River, NJ (2004).
5. D. Pointer, V. Kindratenko, P. Zawada, and M. Pant, The extensible sensor platform, Proceedings of Software Defined Radio Technical Conference, Phoenix, AZ (2004), Vol. A, pp. 201-205.
6. Nallatech XtremeDSP Development Kit, http://www.nallatech.com/.
7. MAX2420/MAX2421/MAX2422/MAX2460/MAX2463 900 MHz Image-reject Transceivers, Maxim Integrated Products, Sunnyvale, CA (2003).
8. A. Betts, M. Hall, V. Kindratenko, M. Pant, D. Pointer, V. Welch, and P. Zawada, The GNU Software Radio Transceiver Platform, Proceedings of Software Defined Radio Technical Conference, Phoenix, AZ (2004), Vol. C, pp. 41-46.
9. Phillips Semiconductors, The I^2C Bus Specification, Version 2.1 (2000).
10. Philips Semiconductors, Bi-directional level shifter for I^2C bus and other systems, application note AN98055 (1997).
11. D. Smith, *Digital Transmission Systems*, 2nd ed., Van Nostrand Reinhold, New York (1993).
12. *Simulink® Model-Based and System-Based Design: Using Simulink Version 5*, The MathWorks, Inc., Natick, MA (2003).
13. Xilinx Platform Studio Tool Suite, http://www.xilinx.com/.
14. *IEEE Standard VHDL Language Reference Manual*, IEEE Standard 1076-1993.
15. HT600/680/6207 3^{18} Series of Encoders Data Sheet, Revision 1.10, Holtek Semiconductor, Inc., Hsinchu, Taiwan (2003).

16. DS1721 2-Wire Digital Thermometer and Thermostat Data Sheet, Dallas Semiconductor, Inc., Sunnyvale, CA (2001).
17. GNU Software Radio Project: http://www.gnu.org/software/gnuradio/.
18. Software Communications Architecture Specification, Technical report, U.S. Army (1998).
19. Joint Tactical Radio System: http://jtrs.army.mil/.
20. 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.
21. A Smart Transducer Interface for Sensors and Actuators – Mixed-mode Communication Protocols and Transducer Electronic Data Sheet (TEDS) Formats, Draft IEEE Standard P1451.4.
22. M. Horton, D. Culler, K. PIster, J. Hill, R. Szewczyk, and A. Woo, MICA, The Commercialization of Microsensor Motes, Sensors, April 2002, Vol. 19, No. 4, pp 40-48.

FIGURE LEGENDS

Figure 1. Nallatech XtremeDSP development kit architecture.
Figure 2. RF front-end diagram.
Figure 3. ESP System design.
Figure 4. Microprocessor subsystem design.
Figure 5. Bit encoding (top) and bit synchronization pattern (bottom).
Figure 6. ASK data receiver design.
Figure 7. **a**) receiver ADC output, **b**) receiver downconverter output, **c**) receiver decimation output, **d**) receiver envelope detector output, **e**) receiver lowpass filter output, and **f**) bit decision output.

FIGURES

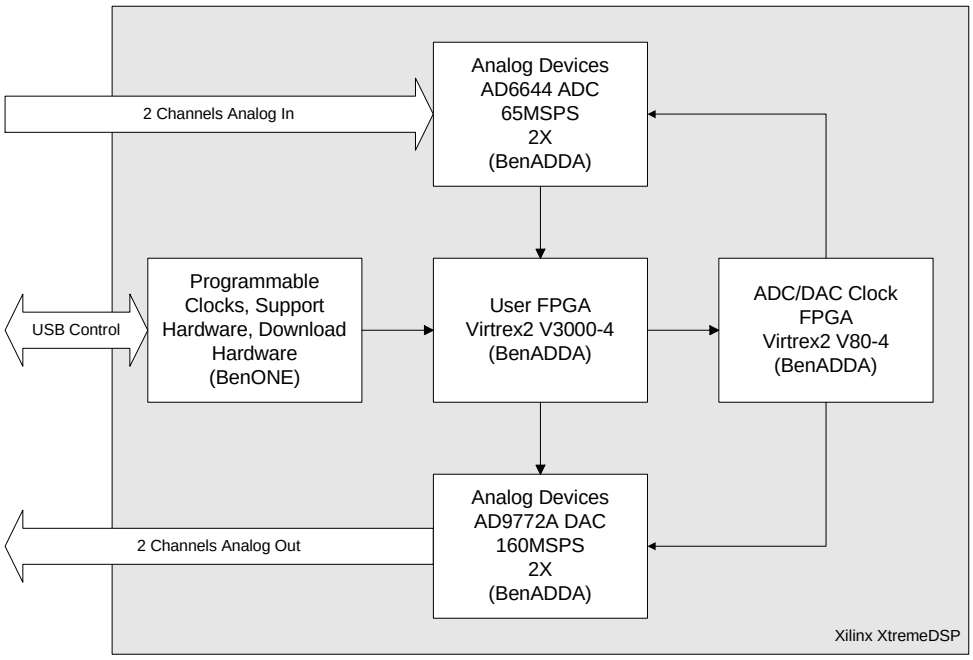

Figure 1. Title: *Mapping a Sensor Interface and a Reconfigurable Communication System to an FPGA Core*

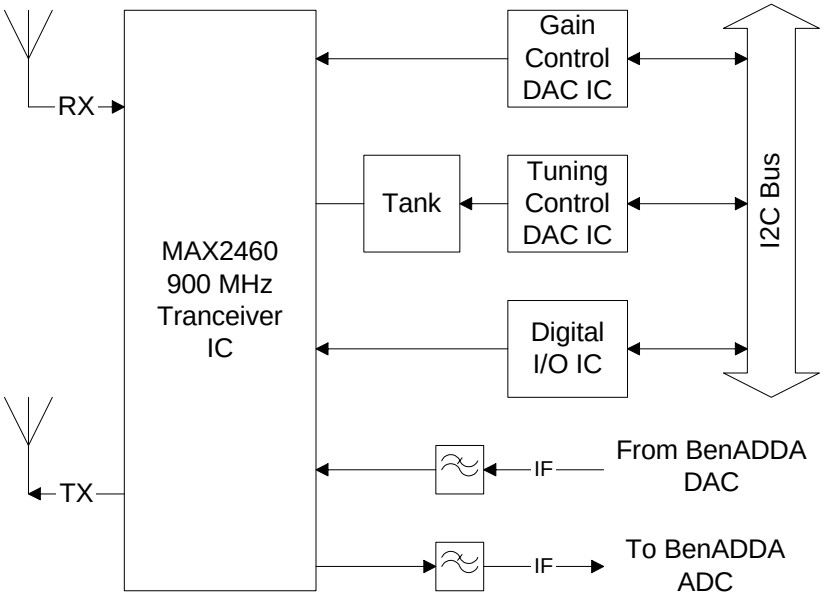

Figure 2. Title: *Mapping a Sensor Interface and a Reconfigurable Communication System to an FPGA Core*

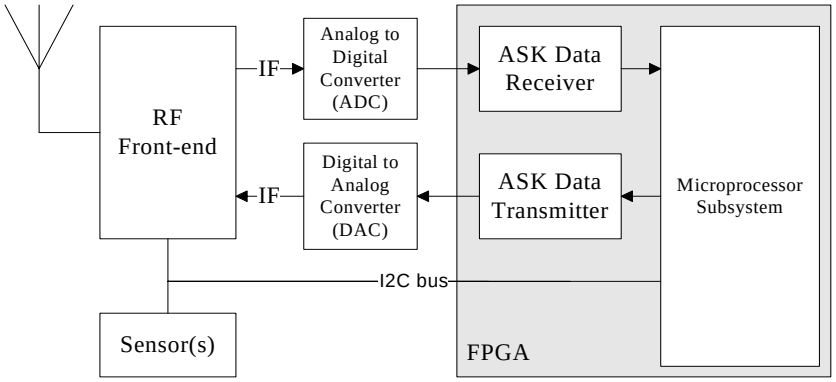

Figure 3. Title: *Mapping a Sensor Interface and a Reconfigurable Communication System to an FPGA Core*

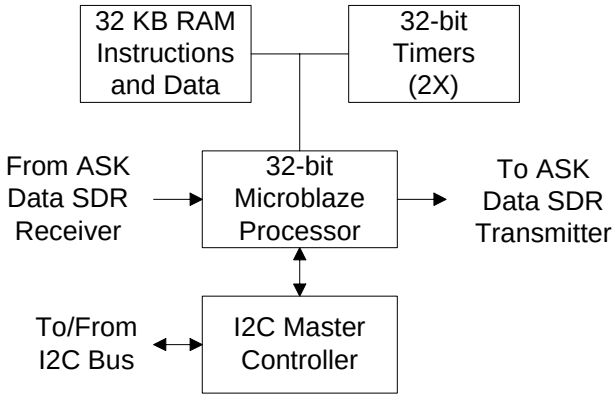

Figure 4. Title: *Mapping a Sensor Interface and a Reconfigurable Communication System to an FPGA Core*

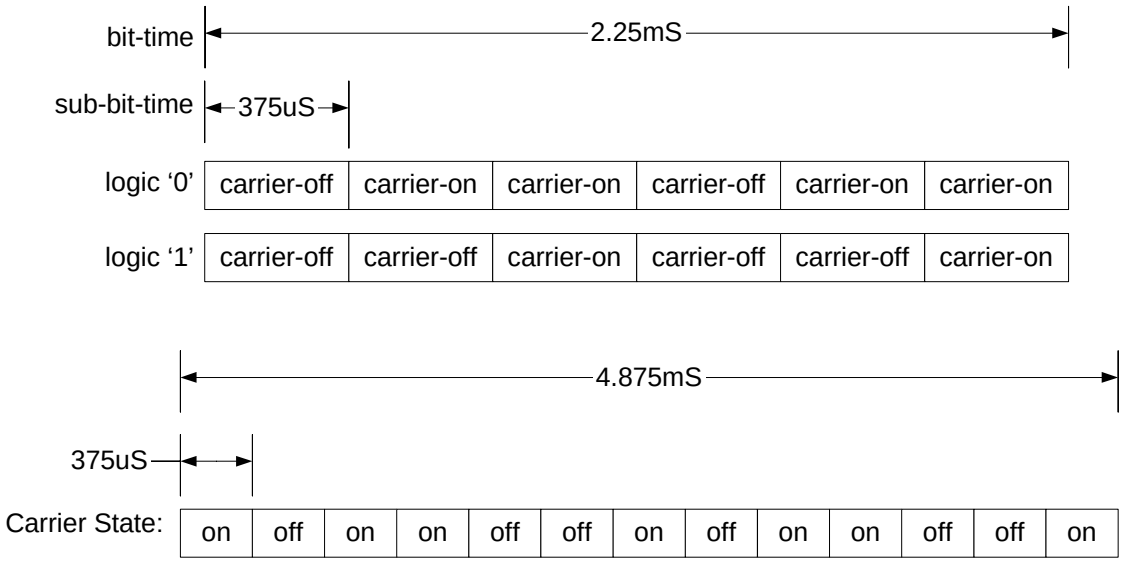

Figure 5. Title: *Mapping a Sensor Interface and a Reconfigurable Communication System to an FPGA Core*

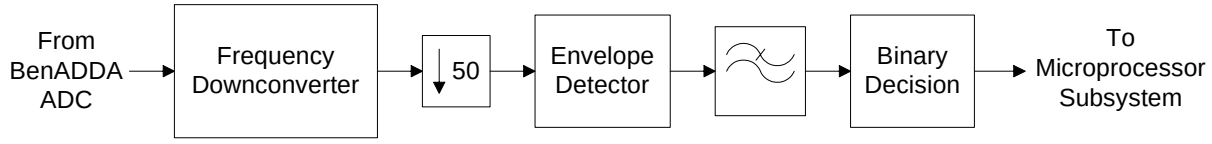

Figure 6. Title: *Mapping a Sensor Interface and a Reconfigurable Communication System to an FPGA Core*

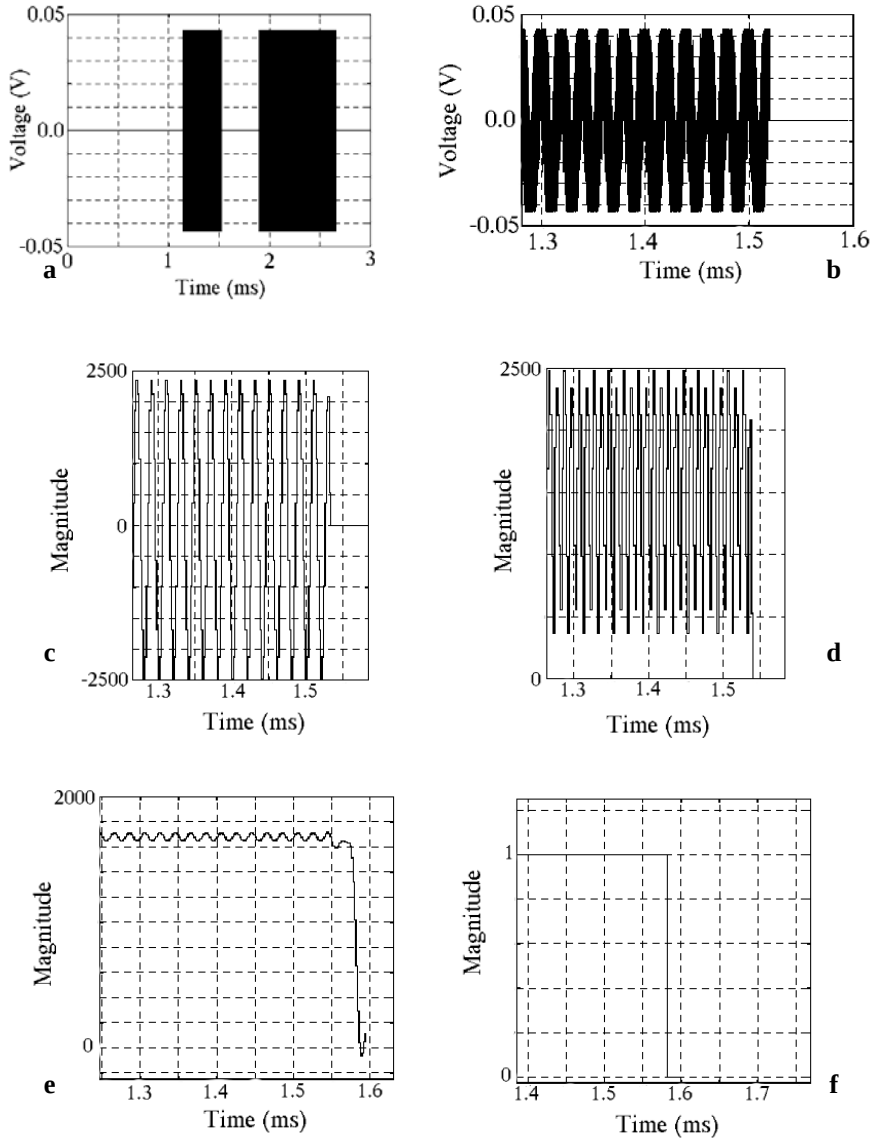

Figure 7. Title: *Mapping a Sensor Interface and a Reconfigurable Communication System to an FPGA Core*