AC 2007-864: SMART SENSOR INSTRUMENTATION DEVELOPMENT EXAMPLE INCLUDING THE NEW PARADIGM OF AN FPGA BASED SYSTEM

Jonathan Hill, University of Hartford

Dr. Jonathan Hill is an assistant professor in the College of Engineering, Technology, and Architecture (CETA) at the University of Hartford, Connecticut (USA). Ph.D. and M.S. from Worcester Polytechnic Institute (WPI) and BS from Northeastern University. Previously an applications engineer with the Networks and Communications division of Digital Corporation. His interests involve embedded microprocessor based systems.

Devdas Shetty, University of Hartford

Dr. Devdas Shetty is the founding chair holder of the Vernon D. Roosa Endowed Professorship at the University of Hartford, Connecticut (USA), where he is also Dean of Research, which is a University-wide function. In addition, he is the Director of the highly reputed Engineering Applications Center (EAC) at the University of Hartford. His areas of expertise are Mechatronics, Product Design, Manufacturing and Automation.

Jun Kondo, University of Hartford

Mr. Jun Kondo is a research engineer in the Engineering Applications Center (EAC) of the College of Engineering, Technology, and Architecture (CETA) at the University of Hartford, Connecticut (USA) M.B.A., M.E., B.S. from the University of Hartford and M.A., B.A. from Western Illinois University. He is specialized in Data Acquisition using National Instruments PXI and cRIO systems.

© American Society for Engineering Education, 2007

SMART SENSOR INSTRUMENTATION DEVELOPMENT EXAMPLE INCLUDING THE NEW PARADIGM OF AN FPGA BASED SYSTEM

Abstract

This paper showcases two complementary approaches for the design and implementation of smart sensor systems. They are (a) Hardware-in-the loop approach (b) Using a single field programmable gate array (FPGA) to construct an entire intelligent instrumentation system. The first example presents a mechatronic approach, which is a blend of mechanical, electrical and software elements along with smart sensors. The major elements of the hardware in the loop approach are (1) Use of a language-neutral approach to code development, created using visual object oriented simulation. (2) Design of the smart sensor that composes of sensors, DSP unit, data acquisition and PC. (3) Use of system dynamics and computer simulation in the system development. These basic principles have helped to develop the hardware-in-the-loop simulation concept and at the same time, assist in rapid prototyping.

For the second approach the paper showcases a new computer engineering paradigm, use of a single field programmable gate array (FPGA) to construct an entire mechatronic intelligent instrumentation system, which is uniquely tailored to the application. Apart from the FPGA itself, this research is based on classical computer engineering principles, where a processor uses a bus to access peripherals. The elements of the FPGA based system approach are: (1) Software tools such as the Xilinx embedded developers kit (EDK) are used to implement an entire microprocessor system using FPGA and an analog adapter board. (2) The operating system is selected and required libraries and device driver software is written. (3) The application software is written and tested. While such a processor system is described using a hardware description language such as VHDL, the resulting description is not considered software but rather is used to configure the hardware. Once configured, writing the libraries, drivers, and application involves conventional software development tools.

In contrasting the approaches, the mechatronic approach is more abstract, simulation based, and eventually produces a PC based system. The second approach involves details of an FPGA based microprocessor system. The measurement and smart sensing example chosen here is a new non-contact, in-process surface roughness inspection probe that is portable, low cost and accurate. Rather than using a contact method such as a contact stylus that could scratch the sample surface, this new probe was developed by the author's and it uses a precision laser and measures the surface scatter to evaluate roughness. The paper highlights the results of sensor application and provides a comparative evaluation of two instruments. This new paradigm also provides opportunities for research in combining the two approaches.

Introduction

This paper showcases two complementary approaches to the design of smart sensor systems. Hardware-in-the-loop simulation is a cost-effective mechatronic approach to perform system tests in a virtual environment. While conducting system tests, the mathematical models are used in the virtual environment along with the components undergoing tests in the closed loop system. As such, rapid prototyping and hardware-in-the-loop simulation are an integral part of today's product development process. The use of a single field programmable gate

array (FPGA) to construct an entire mechatronic intelligent instrumentation system is a new computer engineering paradigm. The key to designing with a FPGA is tailoring a system to the application. Apart from the FPGA itself, the approach is based on classical computer engineering concepts. Underlying both approaches, the principle objective of this paper is to present the design and integration of smart sensors into a typical real time control system.

Mechatronic Design Process

A typical mechatronic design process[1] is shown in Figure 1. This 11 step design process has three main phases: Modelling / Simulation, Prototyping and Development. Starting with steps 5 through 9, software tools are available to aid the designer in creating and debugging the mathematical system models. Some tools that are particularly useful allow the designer to represent the system by creating a system block diagram from simple building blocks such as integrators, gain stages, summing junctions, and non-linear switches. These graphical simulation tools run on generic platforms such as desktop PC compatibles running Windows® operating systems and workstations running UNIX. Some examples of these tools are LabView, LabTech, Lab Windows, Simulink/Matlab, Matrix-x, ACSL, SimPack, Hypersignal, and VisSim™. With any of these tools, the designer can create a plant model, and then validate it against real-world measurements (step 5). Once the plant model has been validated, the designer can then design the control system and optimize it until the correct response is achieved (steps 6 & 7). In some cases, completely accurate plant models cannot be made and certain assumptions must be made about that plant model that cannot be validated. In these cases, it is advantageous to be able to test the control system within the plant environment (step 8). This is sometimes referred to as hardware-in-the-loop simulation since some of the actual hardware (mechanical and electrical parts) is used in the system control loop (acting as the plant that is to be controlled). This hardware-in-the-loop simulation testing provides the designer reassurance that any assumptions made on the plant model were correct. If any assumptions were incorrect, the designer has the opportunity to optimize the design (step 9) before committing to the real target hardware platform.

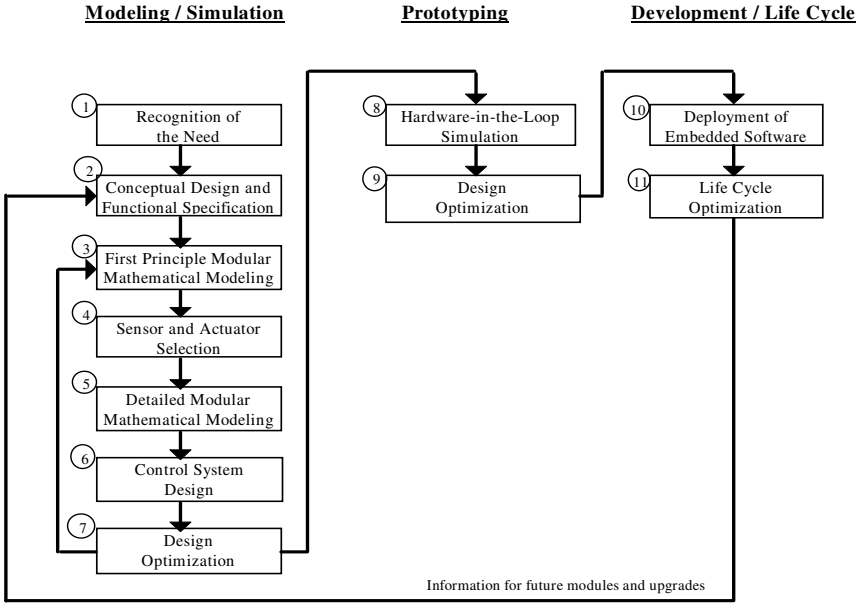

Figure 1: Mechatronic Design Process

Language Neutral Approach

In mechanical engineering, the focus has typically been on machine improvement through mechanical design. The majority of design solutions are deployed in the mechanical design itself and do not involve cross-discipline (software and electronic) technologies. One of the major challenges of any mechatronics sequence is the process for software design, implementation, and testing. Typically the focus is on the embedded software programming and embedded hardware aspects, which include language, computer architecture, and development tools. However, focusing on visual language-neutral programming applications, such as Simulink, Labview, VisSim, Hypersignal and others, will generate code which does not require the time needed to gain an intimate knowledge of a specific language and its development environment.

Smart Sensor Development Using Hardware-In-The-Loop Simulation

Smart sensors are devices that are optimally designed into a system to measure specific physical phenomenon that are normally difficult to measure. Such sensor systems are inherently dependent on using microprocessors. So-called hardware-in-the-loop approaches account for the physical plant and sensors together in a system. There are two main methods currently used to accomplish hardware-in-the-loop simulation testing[1, 2]. (a) PC based "Hardware-In-the-Loop Simulation" as in Figure 2. (b) Embedded DSP based "Hardware-In-the-Loop Simulation" as in Figure 3.

PC based Hardware-In-The-Loop Simulation

This method utilizes the desktop/workstation Graphical User Interface (GUI) coupled with standard Data Acquisition and Control (DAC) interface card. The interface card is inserted in one of the PC expansion slots of the chassis. The actual plant environment is used in place of the plant simulation model, and actual sensors and actuators are connected between the plant and DAC interface circuit. The control simulation model then is modified by deleting the plant model, and adding input and output blocks to the interface card. The model is "executed" through the desktop/workstation operating system. The operating system allows general calls to be made to the hardware for video display, keyboard input, and interface card. Figure 2 shows a typical configuration for this type of hardware-in-the-loop system.

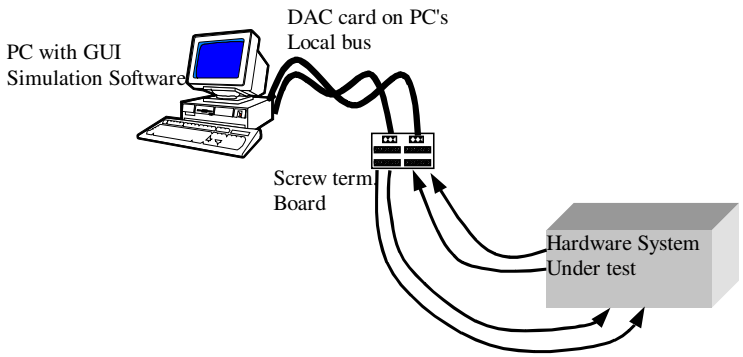

Figure 2: Typical PC based "Hardware-In-The-Loop" System[3]

One major drawback that the PC based simulation systems suffer from is the inability to work in systems where loop responses need to be fast (less than 100ms). This shortcoming is the result of non-optimized software code running through an interpreter which then interfaces to a Windows® operating system that is not designed for "real-time" processing. This creates "real-time" simulation results that vary depending on what other programs are running on the PC platform such as network connections, printing utilities, virus scanning software, disk accesses, etc. Although hardware pricing for PCs continue to decline, a PC system has many features that are not needed for typical control systems. These added cost features include large fixed disk drives, high-speed / high-pixel count graphics capabilities, serial and parallel communication channels, keyboard and mouse interfaces. Once the control system is designed and debugged, these features are not needed. As a result, most real-time control systems are not implemented on a PC platform. In addition, since PCs do require these added features, they are not as portable and as small as an embedded processor platform. Therefore once the control algorithms are designed and debugged, the algorithms must then be re-implemented, re-tested and debugged on an embedded platform.

Embedded DSP based Hardware-In-the-Loop Simulation

The second method for accomplishing hardware-in-the-loop testing involves cross-compiling the control algorithm to target an embedded real-time processor platform. The embedded processor platform often is a Digital Signal Processor (DSP), with I/O that is tailored for embedded system products. This I/O may be in the form of analog inputs and outputs as well as digital inputs and outputs. The cross-compiled code is then downloaded to the embedded processor, sensors are connected to the inputs of the embedded processor board, and actuators are connected to the outputs of the embedded processor board. Since DSPs are designed for signal processing and embedded controls, already proven software routines exist for each DSP platform. These software routines include optimized code for fixed and adaptive filters, audio signal compression and decompression, modulation and demodulation schemes for data communication, and motion control routines such as reference frame conversions (Clarke & Park Transforms) and pulse width modulation (PWM) schemes.

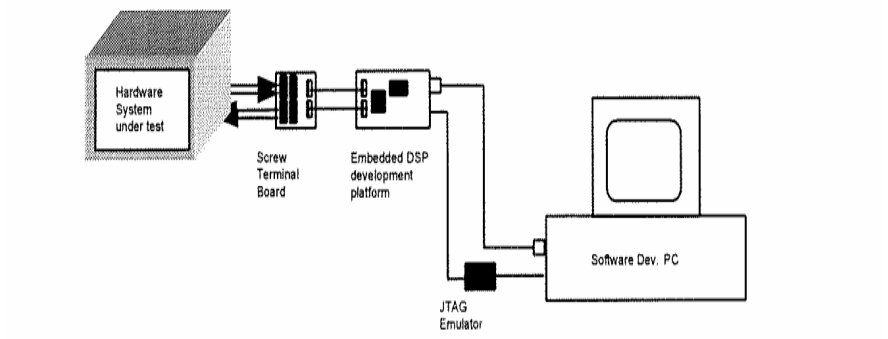

Figure 3: Embedded DSP based "Hardware-In-The-Loop" Simulation Setup

Embedded processor platforms are designed for reduced cost and increased speed, and as such they generally do not have video displays nor standard desktop inputs such as full function keyboards and mouse interfaces. However, DSPs are now becoming available that have integrated peripheral functions that are useful for signal acquisition and control. These peripheral functions include multi-channel A/D conversion, quadrature signal decoders for position and velocity feedback, hardware PWM generation, and serial communication channels. Until very recently, most embedded processors utilized custom real-time operating

systems, or single threaded straight code with no operating system. Figure 3 shows a setup for a DSP based hardware-in-the-loop testing.

One of the drawbacks until recently for an embedded DSP based hardware-in-the-loop simulation was the lack of available tools for development, test, and debugging an embedded DSP. Typically, the programmer must be very familiar with the details of the DSP control registers, must program in C or DSP assembly language, and must rely external devices such as logic analyzers in order to debug the code. However, many PC based block-diagram simulators now do provide a code generation function that allows the designer to create C code that can be ported to the target system of choice. Some of the block diagram simulation software that support embedded DSP platforms are: LabView, Simulink/Matlab, DSpace, Matrix-x, and Hypersignal. These simulation tools allow one to create block diagrams that can be simulated and tested on the PC. Then with relative ease, the blocks can be compiled and executed on the DSP platform, with the PC acting as the user I/O. Therefore if it is possible to simulate the plant, the programmer can debug the control algorithm while simulating the plant, then port the control algorithm to the DSP hardware, and perform the hardware-in-the-loop simulation.

Smart Sensor Development Using an FPGA Based System

We next outline a new design approach made possible by the field programmable gate array (FPGA). Such a system is used in manner like Figure 3, but the DSP board is replaced with an FPGA system board. The FPGA takes a prominent role in the creation of this smart sensor. Code written in a language such as VHDL is compiled, producing an image file or bit file, used to configure the FPGA. Despite being produced by a high level language, the image file is not an executable program but rather is a microprocessor system that can execute machine code. An FPGA is initially an array of uncommitted resources, and quite literally is configured to become the system you describe. Three companies supporting this approach include Actel[4], Altera[5], and Xilinx[6].

The PC based and DSP based design approaches each focus on the application itself. In contrast, the key to designing with an FPGA is tailoring a system to the application. With an FPGA there is an enormous flexibility that allows the system to be as simple or as sophisticated as is required. This observation suggests that design can start with a preliminary PC based or off-the-shelf DSP based system. Once the characteristics of the application are well understood, design of an FPGA system can follow. The surface roughness probe we consider follows this approach.

In general terms, designers can use pre-written peripherals or write their own. A range of pre-written and third-party peripheral descriptions exist, ranging from simple parallel input-output devices that provide a modest user interface, to local area network hardware and video devices that provide a PC like experience. Likewise, the options for system software range from having no operating system, to a real-time operating system, to a modern operating system such as Linux.

Producing a system involves selecting a processor, collecting the descriptions of needed peripherals, writing driver software, and then writing the application software, itself. We have two options for the microprocessor. Some FPGA chips have one or more actual microprocessor cores fabricated along with the FPGA fabric. Such a microprocessor is referred to as a hard-core. In such a device, the FPGA fabric is used to implement required

peripherals. Xilinx Virtex-4 series[7] FPGAs have the option for one or two Power-PC 405 processor cores to be fabricated in the FPGA chip.

Conversely, if the image file also includes the description of a microprocessor core in the FPGA fabric, then once configured the FPGA will execute machine code in the same manner as any microprocessor. Such a configured microprocessor is said to be a soft-core. The Xilinx Microblaze soft core processor is a fairly generic 32-bit pipelined RISC style architecture that can be configured into any Xilinx FPGA[6] currently being manufactured. As an alternative to designing a PC board, many options exist for off-the-shelf development boards and embedded system boards[8].

Application to Non Contact Surface Roughness Sensing

Surface roughness has been conventionally measured using contact type instruments that traverse along the lay of the surface of measurement. Contact with the surface is made by a stylus that records the undulations of the traverse. These undulations are recorded as the surface roughness value. However, the contact method is time consuming. Light scattering has become a practical tool for measuring surface finish. It is a quick method that is sensitive, area sampling, and is an inherently absolute method. The diffraction pattern from the surface consists of identifiable diffraction spots of maximum and minimum intensity of light as a function of angle. The determination of the frequency of diffraction spots leads to the information of the surface. The necessary link between scattering and surface topography can be made using either empirical correlation or an appropriate scattering theory. For implementation of the integrated intelligent system, this can be accomplished by feeding the appropriate input/output to the relationship. An object and its far-field diffraction pattern have a Fourier Transformation relation with each other. If the object distribution is represented by $f(x, y)$, its Fourier transformation F(x,y) is given by,

$$F(x, y) = f(x,y)e^{-2p(xu+uv)}\,dydx$$

where x and y are spatial coordinates; u and v are spatial frequency variables. The basic light scattering principle is shown in Figure 4. For example, an electromagnetic wave of known wavelength is incident upon the rough surface at an angle θ_1. The scattering surface where the light is projected may have either one or two-dimensional roughness. Machined surfaces tend to exhibit a grating structure on account of tool marks made during the machining process. In the case of periodic roughness on a machined surface, the scattering is made up of a specular component, at an angle predicted by ray tracing optics, and discrete components at angles predicted by the grating equation as shown in (1).

$$\theta_{2m} = \mathrm{Sin}^{-1}(\mathrm{Sin}\,\theta_1 + m\,\lambda/T) \qquad (1)$$

where: $m = 0, \pm1, \pm2,$

T = Surface period

The angle of diffuse scatter, θ_{2m}, is related to the period of the roughness.

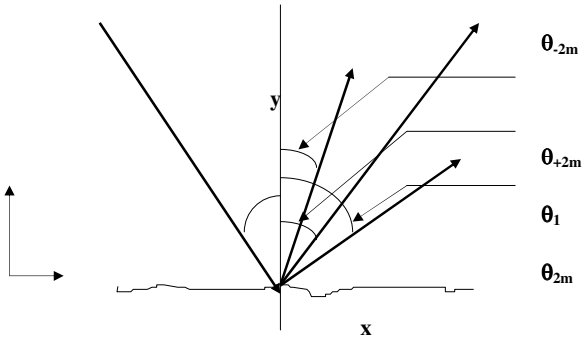

Figure 4: Basic Scattering of Diffracted Light

It could be concluded that the diffraction patterns for machined surfaces, resulting from laser light at different incidences to the surface, are characteristic of that surface. Therefore, from this data, a determination of the type of machining performed on the surface can be made. The amount of dispersion in a diffraction pattern is directly related to the surface roughness of the material. This is true in the case of the non-machined as well as the machined surfaces.

The method developed by Shetty and Neault[14] takes advantage of the fact that a light source reflected off the surface of a work piece provides a signature pattern based on the roughness of the surface. Since surfaces produced by various processes exhibit distinct differences in texture, the specimens of the machined surfaces can easily be identified by looking at the diffraction pattern - such as ground, shaped, milled or turned work pieces. Also, combined with a simple computer algorithm, the engineering surfaces in question can be classified by means of the measurement of the scattering intensities of the diffracted image. The basic principle of the technique, therefore, involves using the diffracted light parameter off an engineering surface plane, digitizing it and further comparing it to a calibration curve. It offers a simple, reliable, and robust approach of evaluating the surface finish of engineering surfaces regardless of the work piece orientation. Figure 5 shows a miniaturized surface roughness probe developed as a part of this research project.

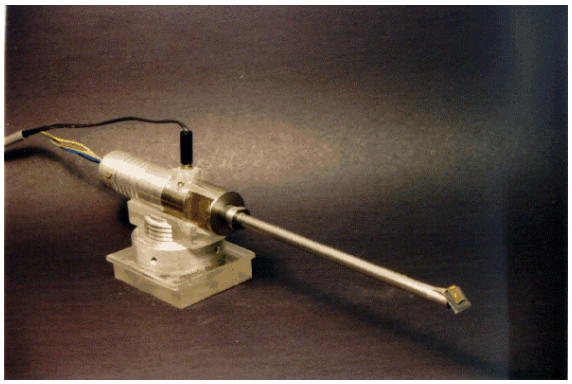

Figure 5: Miniaturized Surface Roughness Probe

Procedures of Measurement – Hardware in the Loop Method

The operation of the instrument is simplified by means of a microcomputer-based procedure which provides the operator interaction in the form of a menu-driven graphical interface. This

helps to guide the operator through the requirements for each phase of the process. There are two phases during the operation, i.e. calibration and measurement. The calibration phase is performed using two standards of high and low roughness values between which the roughness value of the measured sample is expected. This procedure establishes the calibration curve, which is to be followed any time any changes are made. These changes could be related to the machining process or the reorganization of the machine set-up.

In the measurement phase, the data acquired in the measurement procedure is compared to the data acquired in the calibration procedure. The objective of classifying surface roughness of a machined work piece is attained in a measurement method utilizing laser and a microcomputer-based vision system. The intensity of the collimated monochromatic light source diffracted in the spectral direction is captured by a video system that provides an analog signal to a digitizing system for conversion to digital information which is subsequently modified to display the surface roughness value. The intensity of the diffracted light is measured as a function of the gray levels of the image. It is then processed by the digitizing circuit and compared against the previously defined calibration standard (Figure 6). Results of the surface roughness experiments are calculated using computer algorithm. The surface roughness probe is mounted on a Computer Numerically Controlled Machining Center and programmed to take measurement periodically during machining operation.

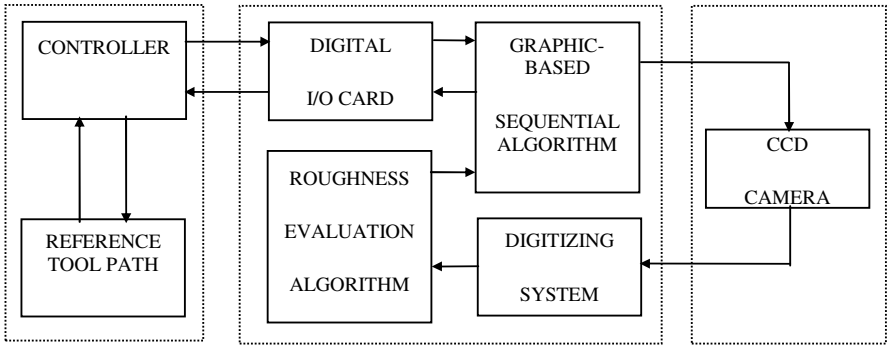

Figure 6: Surface Roughness Sensing System on a CNC Machining Center

In this experiment, three ground samples of 16 μin, 32 μin, and 63 μin were used. Once the sample was set up inside the machining center, the surface analyzer is then used to inspect surface roughness at several different sections of the workpiece. The experimental values were then averaged. The procedures were repeated for the two other samples. The values obtained from the on-line inspection technique were compared with the roughness measurements from the same samples taken using the standard contact-type profilometer. Figure 7 shows that good experimental results of the measured average roughness value, Ra, were obtained with the non-contact optical method as compared to the contact type method.

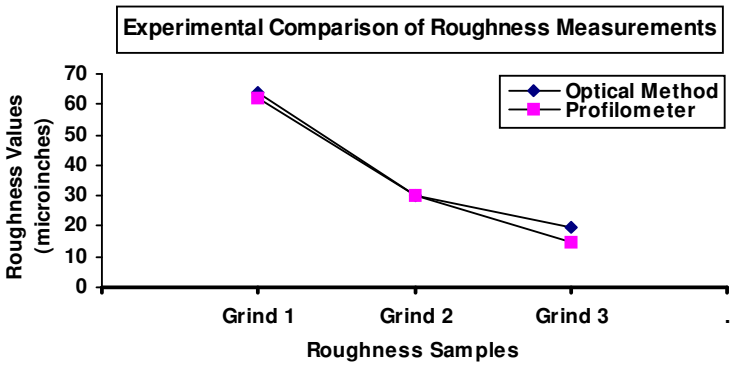

Figure 7: Comparison of Surface Roughness Measurements (Hardware-in the loop method)

Procedures of Measurement - FPGA Based System

For this research, we are currently using a Xilinx Spartan-3 series FPGA configured with the Microblaze processor[9] on a Spartan-3 starter board, from Digilent Inc[10]. The FPGA system itself is designed using classic microprocessor design principles. Figure 8 outlines the surface roughness measurement system. On the FPGA the processor (PROC) connects to peripheral devices using the on-chip peripheral bus (OPB). Each such device is referred to as an IP core. In this example, the probe acquisition circuit has a digital component (ACQD) in the FPGA, as well as an external analog component (ACQA). Input-output logic (I/O) provides a user interface (U.INT). The external memory controller (EMC) provides access to additional memory (MEM). In general terms, apart from the use of an FPGA, this is truly an embedded microprocessor system.

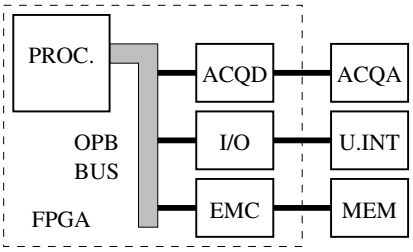

Figure 8: FPGA based processor system

For the surface roughness measurement system, a custom interface adapter card is designed. The acquisition peripheral comprises the actual probe, the interface card (ACQA) containing analog electronic components and an analog to digital converter (ADC), as well as digital logic (ACQD) on the FPGA. In selecting an ADC for this application, the following design goals are selected:

- Operation with 3.3 volt power
- The sensitivity must be suitable to measure input changes in the order of millivolts to a bias voltage of approximately half that of the power supply
- A sampling rate of approximately ten Hertz

- The time for the ADC to settle, allowing a first measurement must be less than two seconds

In this application, changes to the input that are of the order of 1 mV, or 64dB smaller than the bias voltage are typical. An ADC with a 3.3 Volt reference producing 14 bit values has a discretization size of 0.2 mV, which is reasonable for such measurements. Several options exist. To make more use of the FPGA resources, we use the circuit in Figure 9, which is adequate for low-bandwidth applications. Xilinx application notes[11] describe a similar circuit. This ADC directly tracks the input using an up-down counter (Up/Dn) that provides an estimate of the resultant code, which acts as a threshold for the pulse-width modulator (PWM). The PWM and local low-pass filter act together as a digital to analog converter, producing the local comparison voltage. The comparator is the only active analog component in this circuit.

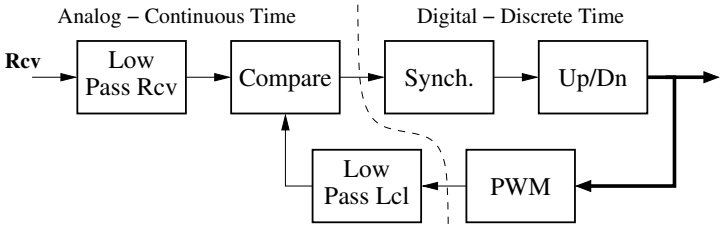

Figure 9: ADC Block Diagram

In hindsight however, given the state of the art in monolithic ADCs, companies such as Analog Devices[12], Maxim[13] and others provide many options in terms of devices that are small, efficient, and cost effective. While 14 bits may be adequate, with such parts, producing a 16 bit value as with the PC design approach may be more standard.

Figure 10 is the prototype FPGA based surface roughness sensing unit consisting of surface probe, FPGA board, and a display module with MicroBlaze 32-bit RISC soft-processor architecture. The probe is attached to the vertical stand. Below the probe is a plate with calibration surface samples. The FPGA board is inside the black box to the right, providing RS232 serial output as well as LED visual output. The main program is written in C, and as outlined earlier, the peripherals are described with VHDL.

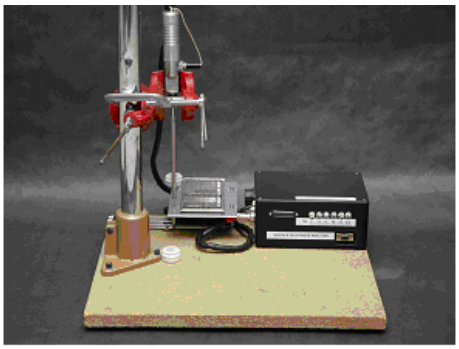

Figure 10: FPGA Based Surface Roughness Sensing Prototype

Figure 11 summarizes measurements taken with the FPGA based surface roughness sensing prototype probe. The results are in agreement with the typical reading obtained by contact type surface roughness measurement units such as Talisurf.

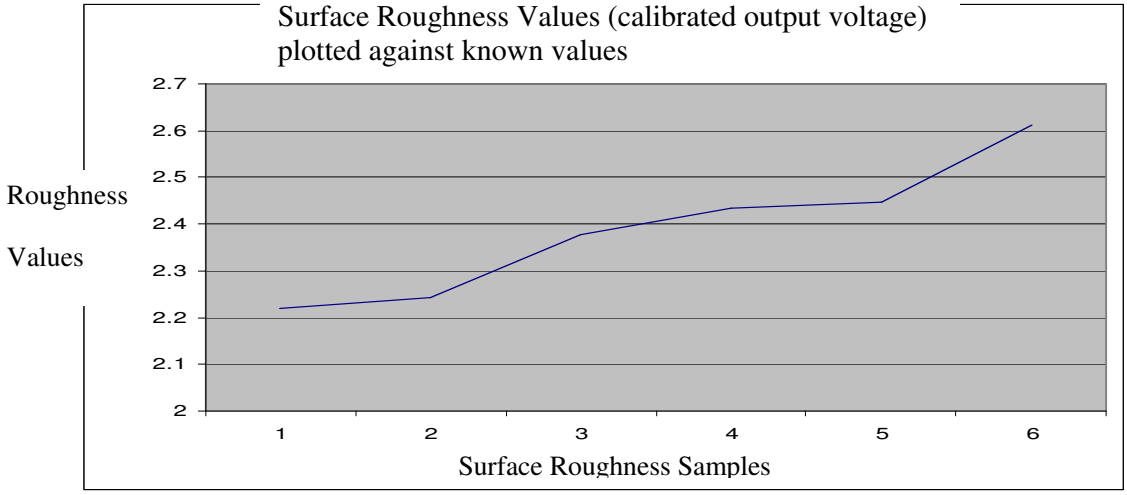

Figure 11: Comparison of Surface Roughness Measurements (FPGA results)

Conclusion

This paper has demonstrated a unique approach for the design and implementation of smart sensors. The research highlights how the mechatronic system design principles can be extended to the domain of embedded process and rapid prototyping. The procedure offers the design engineer, a level of interaction with the modelling of a system before committing to the real target hardware platform. Testing and development times for the sensor prototype are substantially reduced. Microcontrollers embedded in the sensor makes the sensor more cost effective, modular and easy to use in a wide variety. The paper also demonstrated how a sensor instrumentation can be implemented using two different approaches (a) mechatronic hardware-in-the lop (b) FPGA techniques. The test results show high level of agreement

Bibliography

1. Shetty, D. & Kolk, R. (1998), Mechatronics System Design, International Thompson
2. Bhatt, S. (2001), *Design and Development of Smart Sensors,* Master Thesis, University of Hartford
3. Bogli, C. (2000), Study of a New Design Approach to Mechatronics Real-Time Hardware-In-The-Loop Testing, Master Thesis, University of Hartford
4. Actel, http://www.actel.com/products/arm7/
5. Altera, http://www.altera.com/technology/embedded/emb-index.html
6. Xilinx, http://www.xilinx.com/
7. Xilinx, http://www.xilinx.com/products/silicon_solutions/fpgas/virtex/virtex4/capabilities/index.htm
8. Xilinx, http://www.xilinx.com/products/devboards/index.htm
9. Xilinx, http://www.xilinx.com/bvdocs/ipcenter/data_sheet/MB_sell_sheet_s3.pdf
10. Digilent, Inc., http://www.digilentinc.com/
11. John Logue, "Virtex Analog to Digital Converter," Xilinx Application Note XAPP155, Sept. 23, 1999, http://direct.xilinx.com/bvdocs/appnotes/xapp155.pdf
12. Analog Devices http://www.analog.com/
13. Maxim / Dallas Semiconductors http://www.maxim-ic.com/
14. Shetty and Neault (U.S. Patent) Methodology and Techniques for Surface Roughness Evaluation Using Non Contact Techniques, 1993