Smart Structures and Systems, Vol. 3, No. 1 (2007) 69-88
DOI: http://dx.doi.org/10.12989/sss.2007.3.1.069

Development of a smart wireless sensing unit using off-the-shelf FPGA hardware and programming products

Chetan Kapoor†

School of Electrical & Computer Engineering, University of Oklahoma Norman, OK 73019-1024, USA

Troy L. Graves-Abe†

Department of Electrical Engineering, Princeton University Princeton, NJ 08544, USA

Jin-Song Pei‡

School of Civil Engineering & Environmental Science, University of Oklahoma Norman, OK 73019-1024, USA

(Received December 19, 2005, Accepted June 20, 2006)

Abstract. In this study, Field-Programmable Gate Arrays (FPGAs) are investigated as a practical solution to the challenge of designing an optimal platform for implementing algorithms in a wireless sensing unit for structural health monitoring. Inherent advantages, such as tremendous processing power, coupled with reconfigurable and flexible architecture render FPGAs a prime candidate for the processing core in an optimal wireless sensor unit, especially when handling Digital Signal Processing (DSP) and system identification algorithms. This paper presents an effort to create a proof-of-concept unit, wherein an off-the-shelf FPGA development board, available at a price comparable to a microprocessor development board, was adopted. Data processing functions, including windowing, Fast Fourier Transform (FFT), and peak detection, were implemented in the FPGA using a Matlab Simulink-based high-level abstraction tool rather than hardware descriptive language. Simulations and laboratory tests were carried out to validate the design.

Keywords: smart wireless sensing; FPGA.

1. Introduction

1.1. Motivation

This study was motivated by the need to seek computational and power efficient means of embedding algorithms into wireless sensors. The advantages of using wireless sensor networks for structural health monitoring are well known and the advantages of processing data onboard, and then transmitting the processed and/or identified results (rather than raw data) have been well recognized (Liu and Tomizuka

†Graduate Student
‡Assistant Professor, Corresponding Author, E-mail: jspei@ou.edu

Table 1 General guideline for three options to implement onboard intelligence: ASICs, FPGAs, and microprocessors

Characteristic	ASIC	FPGA	Microprocessor
Implementation of data Processing functions	Application specific hardware	General purpose hardware	General purpose software
Computational efficiency	Very high	High	Low
Cost	Very high	Moderate	Low
Power consumption	Lowest	Moderate	Low

2003, Spencer 2003, Smyth and Betti 2004). As structural engineers working on smart sensing technology for smart structures, it is important to note that microprocessors are not the only means to bring onboard intelligence to sensors in order to make them smart. Indeed, the computational and power efficiencies of microprocessors are not always the ultimate that one can expect. Although their ubiquity and fast-improving performance could lead one to neglect this important fact, it would be imprudent to overlook other available options. For example, Texas Instruments has found that the speed of one its digital signal processors was increased by a factor of 18 by sending certain functions to custom-designed logic circuits (Tredennick and Shimamoto 2003). Comparable increases in power efficiency, which in many cases is more important than computational efficiency, can also be realized. It is proposed herein to introduce Field-Programmable Gate Arrays (FPGAs) as a complement and/or an alternative to microprocessors in smart sensing technology. The goal is to achieve greatly enhanced computational efficiency, increased/ manageable power efficiency, superior flexibility of functionality and generally optimized performance of embedded systems.

For embedding intelligence, a hardware designer typically has three options. One could design a digital system using the traditional microprocessor (a Digital Signal Processor (DSP) or Reduced Instruction Set Computer (RISC) can be considered as subsets), develop an Application Specific Integrated Circuit (ASIC), or use an FPGA (Table 1). ASICs are restricted to applications which require mass production due to their high initial setup costs (e.g. the cost of a single mask, implemented with a 0.09 μm technology is about $1,000,000 (Wolf 2004)). Besides phenomenal development costs, a relatively lengthy time-to-market can also limit the the usage of ASICs. On the other hand, FPGAs, which can be roughly characterized as Programmable ASICs, offer substantial performance gains over the contemporary microprocessors with very low development costs and faster time-to-market.

1.2. FPGA architecture -an intrinsic advantage

A Field-Programmable Gate Array (FPGA) is an integrated circuit consisting of an array of programmable logic cells. The basic architecture of an FPGA (better known as its fabric) consists of a few fundamental elements, such as Combinational Logic Blocks (CLB), interconnects and Input/Output Blocks (IOB) as shown in Fig. 1 (Roth 1998, Maxeld 2004, Yalamanchili 2001). In a Xilinx Virtex-II FPGA, for example, a Virtex-II XC2V40 (Xil 2004d), there are 64 such CLBs organized in a eight row × eight column fashion. Each CLB consists of four slices, while each individual slice internally consists of various components such as Look-Up-Tables (LUT), multiplexers, carry logic, AND-gates, and sequential elements (flip-flops) (Xil 2004b,c). Beside slices, local routing is used to provide feedback between slices in the same CLB, and also allows routing to neighboring CLBs. For global routing, a Switch Matrix is used.

Compared to microprocessors, in an ideal design, FPGAs would have dedicated hardware resources allocated for every task. This basic design architecture is the main factor for their high computational

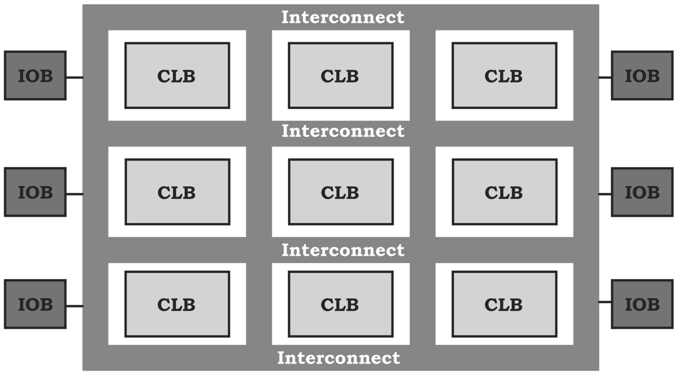

Fig. 1 A basic FPGA fabric following Wolf (2004) where CLB and IOB stand for Combinational Logic Blocks and Input/Output Blocks, respectively

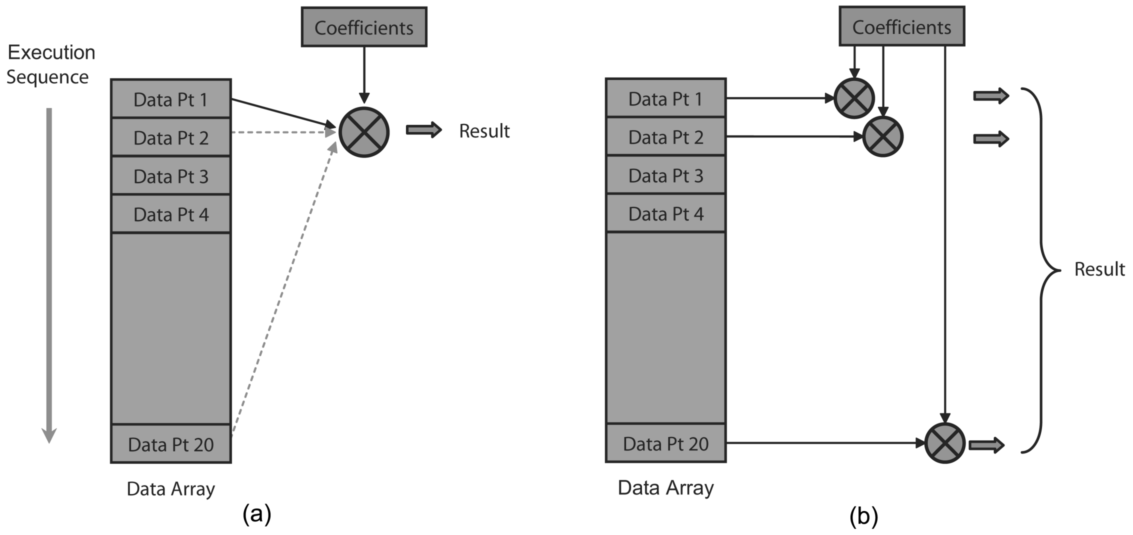

Fig. 2 An example to illustrate the difference between (a) a software execution model, and (b) a hardware execution model. Note that in (a), the execution follows a chronological sequence. In (b), however, the execution follows a concurrent sequence. In this simple example, if the microprocessor (having the software execution model) and FPGA (having the hardware execution model) required similar time for a multiple operation, the FPGA would be 20 times faster with a higher throughput

capabilities. This type of design model is termed a *hardware execution model* (Dehon and Wawrzynek 1997). In microprocessors, general purpose hardware is used to accomplish different tasks. This type of design model is termed a *software execution model* (Dehon and Wawrzynek 1997). The key difference is illustrated with the example shown in Fig. 2, where 20 data points are multiplied by 20 corresponding coefficients in a dot product fashion for two vectors. The computational efficiency of the hardware execution mode using an FPGA is improved substantially over the software execution mode using a microprocessor, if all the required multipliers are available (i.e. using fully combinational logic).

1.3. "FPGA vs microprocessors"-applications in wireless sensing

The technology associated with FPGAs is rapidly growing. Having been around for more than two decades, FPGAs have recently evolved from limited traditional roles (e.g. glue logic) into a new alternative for offloading computationally intensive digital signal processing and beyond. Though relatively new, many application topics using FPGAs for implementing algorithms have emerged in literature (ACM 2002, 2003, 2004). They include embedding Finite-Impulse-Response (FIR) filters, Fast Fourier Transform (FFT), and multilayer feedforward neural networks (Kung, *et al.* 2002, Ohtani, *et al.* 2002). FPGAs are considered for smart sensing applications in structural health monitoring based on the following reasons:

1.3.1. Computational efficiency

On a typical chip with dimensions of 1.3 in. by 1.3 in. (for the Xilinx Virtex-II Pro-XC2VPX20 FPGA in FF896 package (Xil 2004c)), an FPGA spatially composes primitive operations rather than temporally composing them as in a traditional microprocessor (Tredennick and Shimamoto 2003, Verkest 2003, Xil 2004d). This fundamental difference can be viewed as parallel computing architecture in FPGAs versus serial computing in microprocessors. This parallelism is the primary reason for FPGAs computational efficiency since it maximizes data throughput at the level of machine computation.

1.3.2. Power efficiency

For sensor networks including the application in structural health monitoring, energy is generally consumed for data transmission, signal processing and hardware operation (Mahgoub and Ilyas 2006). Loosely speaking, the enhanced computational efficiency of FPGA's can directly result in increased power efficiency. In practice, it can be difficult to fully realize these gains as any logic blocks that are not used by a designer will nevertheless contribute to power use. Thorough power efficiency analysis (Shang, *et al.* 2002, Choi, *et al.* 2003) has indicated that the interconnects and the IOB (see Fig. 1) and clocking are amongst the primary power dissipation sources. There are new power saving technologies currently being developed that involve modifying the FPGA's architecture in an effort to address the speed-power dilemma (Tuan, *et al.* 2006).

1.3.3. Reconfigurability

In an FPGA, the major functional blocks traditionally found on microprocessors (e.g. processing cores, memory blocks, hard-wired functional units) are interconnected in a reconfigurable manner, allowing them to be organized and deployed according to the needs of an application. A major advantage that an FPGA possesses over other Programmable Logic Devices (PLDs) (Roth 1998) is its ability to dynamically change its configuration and thus perform different tasks without the need to be reprogrammed. This capability can be exploited in embedding multiple situation-specific configurations and hence creating a very potent and flexible system.

1.3.4. Cooperation with microprocessors

Reconfigurable hardware like an FPGA by itself might not provide the best solution for all situations. However, FPGAs containing both an FPGA architecture and an embedded microprocessor are available to allow developers to readily utilize both, to facilitate various application needs for optimized efficiency, flexibility and performance, as was selected in this study (Xil 2004c). The microprocessor available on the selected FPGA development board was not used in this particular proof-of-concept demonstration, however it can be utilized for an improved functionality in a future study.

In short, the high processing power and customizability of FPGAs make them very suitable for enabling the current trend towards task-focused sensors in civil engineering applications (Liu and Tomizuka 2003). FPGAs will greatly enhance onboard data processing (i.e. DSP) and data interpretation (i.e. system identification) capabilities. For example, Los Alamos National Laboratory has developed a system called “HERT” for structural health monitoring that uses FPGAs for onboard data processing (Farrar 2004). It is also reported that NASA Goddard Space Flight Center is developing an FPGA version of their Hilbert-Huang Transform (HHT) (NASA 2004).

1.4. Objectives

The goal of this study is to create a proof-of-concept FPGA based wireless sensing unit. The objectives are listed as follows:

1. To replace a microprocessor with an FPGA in a measurement and instrument environment for wireless-sensing based structural health monitoring as shown in Fig. 3. This requires multiple interfacing efforts.
2. To adopt off-the-shelf products for the FPGA and other hardware components as often as possible. This simplies design as was explored in microprocessor-based wireless sensing in the structural health monitoring community (e.g. Lynch, *et al*. 2004).
3. To program an FPGA to perform onboard data processing to make the unit “smart”. In particular, examine an off-the-shelf high-level abstraction programming tool, Matlab Simulink-based *System Generator* from Xilinx (Xil 2004f), as opposed to using only VHDL-based programming. More abstract programming tools provide civil engineers with more convenient access to FPGA-based DSP design.
4. To adopt a low-cost, i.e., non-high-end FPGA product. The consideration of cost is always critical in large-scale civil engineering applications. This design constraint imposed many challenges to this study, which in turns led to an optimal design.

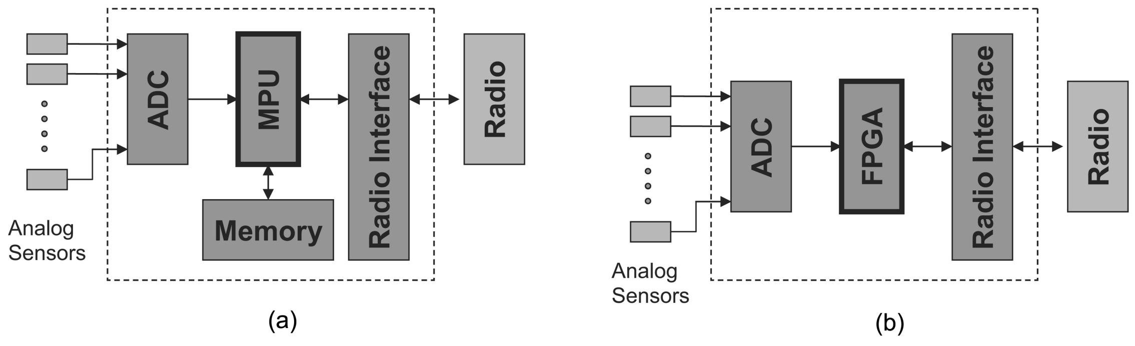

Fig. 3 Comparison between (a) a microprocessor-based, and (b) an FPGA-based wireless sensing unit. Note that modern FPGAs can have sufficient built-in memory to store sensed data and processed results. ADC stands for analog-to-digital converter while MPU, microprocessor processing unit

2. Proposed work

2.1. Project overall

Fig. 4 illustrates the scope of both the hardware and software design in this proof-of-concept study. A Single-Degree-of-Freedom (SDOF) building model on a shaking table was excited by a swept sine signal, while analog accelerometers were used at the client end to collect two channels (Channel 0 and Channel 1) of time histories as shown in Fig. 4(a1). The time series were collected using Analog-to-Digital Converters (ADCs) and processed in an FPGA rather than a microprocessor. After applying a **channel subtraction algorithm** to obtain the relative acceleration of the floor mass, the resonance frequency of this relative acceleration was derived inside the FPGA by applying a **windowing algorithm**, **Fast Fourier Transform (FFT) algorithm** and **peak detection algorithm** consecutively. This processed frequency in its equivalent integer format (i.e. a frequency register as will be presented in Section 4.4) was then sent to the server end (see Fig. 4(a3)) through wireless data transmission and recorded using a Graphical User Interface (GUI). This entire FPGA-based smart sensing unit shown in Fig. 4(a2) was built on an off-the-shelf FPGA development board employing a Xilinx Virtex-II Pro chip and other components. Tools providing a high-level abstraction for programming FPGAs, such as Matlab Simulink-based Xilinx *System Generator* (Xil 2004f), were used to embed the DSP algorithms.

The selected linear single-degree-of-freedom (SDOF) system subjected to a base excitation can be considered as a model of a single-story building under an earthquake excitation. Application of Newton's second law results in the following equation of motion for the system:

$$m\ddot{x} + c\dot{x} + kx = -m\ddot{x}_g \tag{1}$$

where m, c and k are the mass, damping and stiffness of the SDOF system respectively, and $\ddot{x}_g$ is the

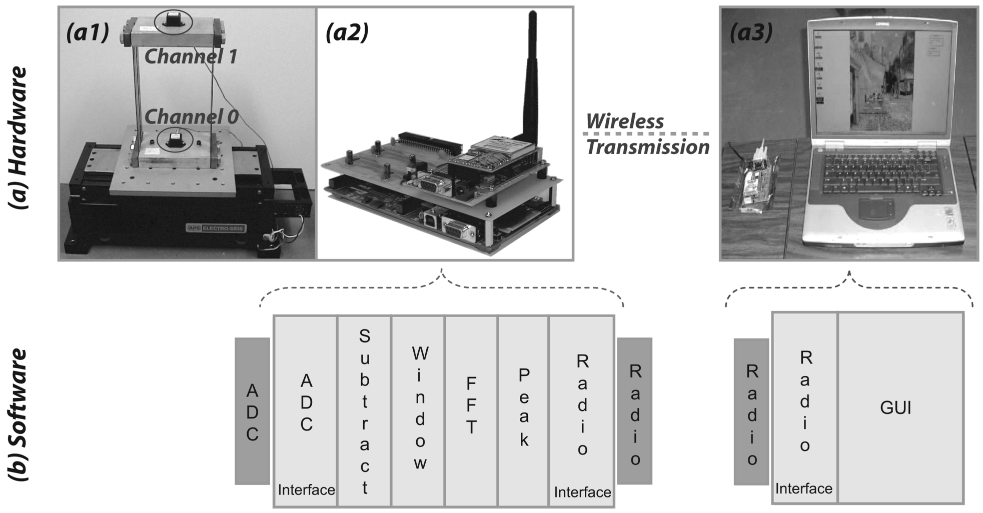

Fig. 4 Overview of the project -an application of an FPGA in a smart sensing environment: (a) hardware, and (b) software development scope. In detail, (a1) shows a shaking table, building model and two channels of analog accelerometers, (a2) the proposed FGPA-based smart wireless sensing unit at the client end, and (a3) the server end

input to the system, i.e., the ground acceleration collected at Channel 0. $\ddot{x}$ denotes the relative acceleration of the roof with respect to the ground, i.e., the reading of Channel 1 minus that of Channel 0. Processing the frequency information of this relative acceleration is the foremost step in deriving the frequency response function and transmissibility of such a system (Chopra 2000, Inman 1994). Therefore, it was decided to showcase the application of FPGAs in a real-world measurement environment for structural health monitoring by extracting the resonant frequency of this relative acceleration and then sending it through the selected OEM radio. This entails the key technical focus of this exploratory research. The proposed work consists of two FPGA operation modes:

2.1.1. "Pass through" operating mode

In the "pass through" mode of operation, the FPGA simply collects acceleration time histories from the two channels of the analog sensors, transmits them using the selected radio and displays the time histories using a GUI at the server end. This mode can be considered an intermediate step to achieve the "smart" mode and is mainly to verify the proper functioning of the designed hardware unit. Even this mode alone can also demonstrate the appropriateness of using FPGAs in such a measurement and instrumentation application.

2.1.2. "Smart" operating mode

Once the "pass through" works, the proposed DSP algorithms from **channel subtraction algorithm** to **peak detection algorithm** are to be implemented into the entire design to make the FPGA-based unit operate in a "smart" mode, so that an identified resonant frequency or its equivalent quantity rather than raw time histories will be received at the server end.

2.2. Choice of FPGA development board and its features

The foremost decision in this study was the appropriate selection of the FPGA device. There are many vendors in the market, providing designers with several options when considering a specific application. Furthermore, there are companies that offer specific Intellectual Property (IP) cores which could be utilized to shorten the overall development and testing cycle. Some well-known vendors for FPGA devices include Actel Corp, Altera Corp, Atmel Corp, Lattice Semiconductors Corp, Leopard Logic Inc, QuickLogic Corp and Xilinx Inc.

The selection of the FPGA development board was based on several considerations. The key factor was the FPGA core. As outlined in Section 1.4, the defined tasks for the selected FPGA include acquiring data from the ADCs, performing a series of FFT-oriented data processing, and transmitting the result through a wireless link to a base station. The desired low-cost and the availability in a development board in a proper size narrowed the choice to either a Virtex-II Pro or Spartan-3 FPGA based system board. Other features, such as a moderate level of board complexity and future extendability using an embedded microprocessor made the low-level Virtex-II Pro based-system board the final choice. Its price was comparable to that of an off-the-shelf microprocessor development board that has been successfully used for structural health monitoring in Lynch, *et al.* (2004).

Fig. 5(a) illustrates the selected FPGA development board containing a Xilinx XC2VP4-5FG456C FPGA (Mem 2004) as the system core. A photograph of the entire FPGA-based smart wireless sensing unit is shown in Fig. 5(b) in addition to the one in Fig. 4(a2), while the details will be presented later in Section 3. The FPGA product name denotes a Virtex-II Pro FPGA (XC2VP4), capable of operating at -5 speed grade (speed grade ranges from -5 to -7, with -7 being the best), housed in a Fine-Pitch Ball

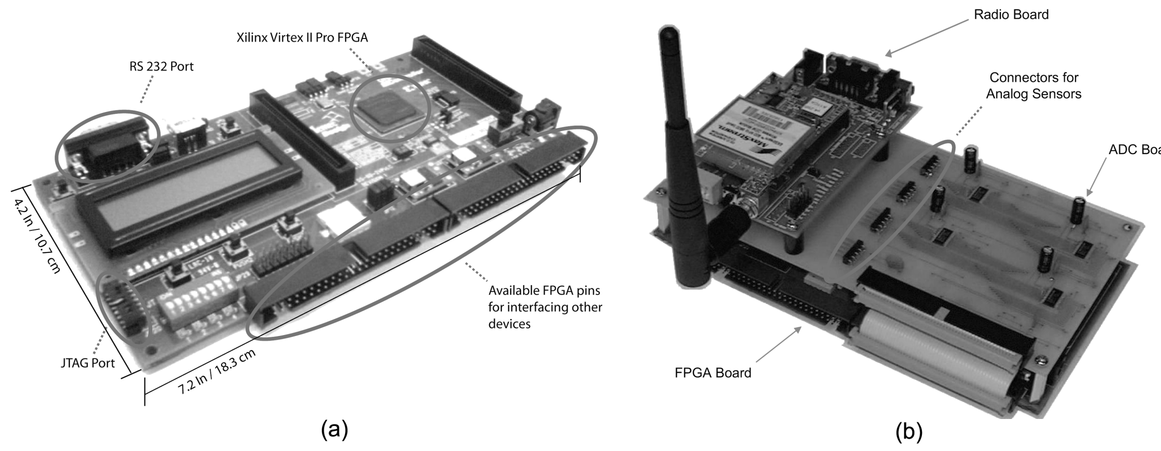
Xilinx Virtex II Pro FPGA
RS 232 Port
4.2 In / 10.7 cm
Available FPGA pins for interfacing other devices
JTAG Port
7.2 In / 18.3 cm
(a)
Radio Board
Connectors for Analog Sensors
ADC Board
FPGA Board
(b)

Fig. 5 (a) Prototyping FPGA board based on Xilinx Virtex II Pro FPGA, and (b) overall hardware design

Grid Array (FPBGA) package with 456 pins. The FPGA itself consists of an embedded Harvard Architecture based PowerPC processor block (Xil 2004c) capable at operating up to 400 MHz. The system board has a 100 MHz low-voltage transistor-transistor logic (LVTTL) oscillator clock that drives the FPGA. Other important features of the board that were utilized for this study are an RS232 port, a 50-pin user I/O connector providing 64 user-accessible I/O lines, a PC4 JTAG programming/ configuration port, and user LEDs (see Kapoor, *et al.* 2005 for details).

2.3. FPGA programming environment

Programming an FPGA basically encompasses configuring the available FPGA hardware to accomplish a specific task. The FPGA's programming directly implements the desired logic functions and the required interconnections. VHDL and Verilog are the two hardware descriptive languages that are available to the designers for programming FPGAs. They offer the least amount of programming abstraction, therefore they can be considered equivalent to assembly language that is used for programming microprocessors.

To reduce the development time associated with implementing FPGA-based system designs, manufacturers provide high-level programming abstraction. *Core Generator* (Xil 2004e) and *System Generator* (Xil 2004f) are examples of such options. In this study, *Core Generator* was used for the ADC interfacing, while *System Generator* was utilized to implemented **all** the proposed DSP algorithms shown in Fig. 4(b) (Kapoor 2005). Noteworthy is that *System Generator* is built upon a MATLAB Simulink (Mat 2005) design environment, where a designer can completely create a FPGA-based DSP design in Simulink by using Xilinx provided block-sets and then convert the design into a suitable format for downloading it into the FPGA. By using *System Generator*, hardware description languages like VHDL or Verilog were not used to develop the proposed DSP algorithms, rather they were used for a limited scope in this study (see Section 4) following Objective #3 of this study stated in Section 1.4.

3. Hardware implementation

As shown in both Figs. 3(b) and 4(a), the hardware for the proposed wireless sensing unit can be functionally categorized into analog sensors, an FPGA-based processing unit and a radio transceiver. The FPGA-based wireless unit was designed to accommodate any analog sensor with an output voltage ranging from 0 to 4 V. In this project, it was interfaced to two uniaxial ± 5 g accelerometers from Analog Devices for the purpose of sensing accelerations at the base (i.e. Channel 0) and floor (i.e. Channel 1) levels of the model structure. The FPGA-based processing unit consists of multiple Analog-to-Digital Converters (one 16-bit ADC per channel), an FPGA and an RS-232 converter, as shown in Fig. 3(b). Overall, the analog signals from the sensors were digitized by the ADCs and relayed to the FPGA. Data was then processed by the FPGA and the result was passed onto the RS-232 converter, which forwarded it to the radio transceiver for transmission to a base station. Challenges and solutions in selecting the off-the-shelf components and interfacing them together to perform the defined tasks are elaborated in the rest of this section.

3.1. Selection of ADCs: CMOS vs TTL

Two main factors were involved in the selection of an Analog-to-Digital Converter (ADC) in this study: the choice of either a serial or parallel interface for communication between the ADC and the FPGA, and the operating voltage. As detailed in Kapoor (2005), the large number of interface lines available through the 50-pin I/O connector enabled the adoption of a parallel ADC interface to avoid the more complicated design requirements of a serial interface. The second factor was caused by a unique interfacing concern in this study: modern FPGAs use low-voltage-logic Input/Output standards such as Low-Voltage Transistor-Transistor Logic (LVTTL) or Low-Voltage Complementary Metal-Oxide Semiconductor (LVCMOS). The operating range for these standards is 0 to 3.3 V. However, the output from the selected analog sensors ranges from 0 to 5 V.

An ADC with independent reference voltage (V_{ref}) and supply voltage (V_{ss}) was desired so that V_{ref} could be set close to the maximum voltage for the analog sensing range while still retaining V_{ss}= 3.3 V (for communication with the FPGA). It was found that the MAX1165 (Max 2004a) by Maxim Dallas fitted these requirement perfectly, with an independent V_{ref} and V_{ss}. Furthermore, it had a 16-bit parallel bus interface, and its maximum sampling rate of 165 KSPS was found to be adequate for this measurement application. The only drawback of using this ADC was that it lacked an internal multiplexer for the input analog channel, thus limiting the number of analog sensing channels to one. This deficiency was overcome by using multiple ADCs corresponding to the number of channels required. A custom printed circuit board (PCB) was developed for this sensor unit to house the external ADCs as shown in both Figs. 4(a2) and 5(b). Four separate ADCs were employed to digitize the analog signals from the accelerometers (the two will be used for future expansion). These four ADC shared a common 16-bit wide databus, but had individual sets of control lines. The digital and analog voltages of 3.3 and 5.0 V respectively, were supplied by the FPGA board. All these signals and voltages were connected to the FPGA board using a 50-pin connector on the ADC board through a 50-wire ribbon cable.

3.2. Off-the-shelf radio

A MaxStream OEM radio board with a carrier frequency of 900 MHz (Max 2004b) was adopted in this study. This selected radio uses an RS-232 interface for communication and data transfer from the host controller (i.e. the FPGA at the client side and the computer at the server side). Frequency hopping spread spectrum (FHSS) modulation technique is employed by the radio, which provides better immunity from radio interference as compared to other available radios (or radio modules) operating on single frequencies. Furthermore, the radio is able to form a transparent serial link over the air creating a virtual wired connection and thus hiding the complexity usually involved in creating and managing a wireless data transfer link. This conceals the issues of handshaking, error detection and correction and frequency management. Besides using a radio board which incorporates the RS232 protocol, separate radio modules can be directly interfaced to the FPGA, similar to the case of a microprocessor-based design (Lynch, *et al.* 2004).

3.3. Hardware interfacing

Details on interfacing the selected ADC to the FPGA are documented in Kapoor (2005). Overall, the manufacturer recommended details were followed except making 3.3 V as the input digital voltage since the selected FPGA follows LVCMOS standards. The ADC internal referencing has been set to 4.096 V by the manufacturer, therefore the analog sensing voltage range is 0 to 4.096 V. Also, precise timing and waveform of the the ADC Chip-Select ($\overline{CS}$) and Read/Convert ($R/\overline{C}$) signals, are critical to the proper operation of the ADC according to the timing diagram of the selected ADC (Max 2004a). As detailed in Kapoor (2005), two shift registers were employed to generate these control signals. The contents of these registers were sequentially shifted out (starting from the least signicant bit) at rate of 1.6 KHz to generate the required signals.

In terms of the radio interface, two available pins on the RS-232 port on the selected FPGA development board, namely Transmit Data (TD) and Receive Data (RD), were utilized in the hardware design, while an MAX3221 on the board was used as the RS-232 driver (Mem 2004). This is required because the FPGA's input/output data lines operate on a LVCMOS voltage level (i.e. 0 to 3.3 V), whereas the RS-232 protocol requires the voltage level of ± 12 V for data communication (Kapoor 2005).

4. Software implementation

To meet the dual Objectives #3 and #4 specified previously in Section 1.4, namely, using an off-the-shelf high-level abstraction tool and working with a low-cost, non-high-end FPGA, the capacity of the selected low-cost FPGA and the demand from various algorithms and execution models (i.e. parallel vs sequential) is examined in Section 4.1. Based on this analysis, an overall design guideline was obtained to increase the overall throughput of the design (i.e. the computational efficiency) while balancing the available resources including slices, multipliers and block RAMs to implement the proposed scope of software development outlined in Section 2.1 and illustrated in Fig. 4(b).

4.1. Design constraints and guidelines

The available resources in the selected low-cost FPGA are summarized in Tables 2; note that the FPGA on the selected low-cost development board is next to the smallest FPGA in the V2P family. Various device options are summarized as follows:

4.1.1. FFT

Since the core of the proposed DSP is FFT, the resources required for performing FFT using *System Generator* under three different options are listed in Table 3. Here the terms of "Radix-2" and "Radix-

Table 2 Virtex II Pro (abbreviated as V2P in the table) device parameters (Xil 2004c). See Fig. 1 and Section 1.2 for some of the key terms, refer to Xil (2004d) for the rest of the terms

Device	XC2VP2	XC2VP4	XC2VP100
Rocket I\O transceivers block	4	4	20
Number of PowerPC processor block	0	1	2
Number of logic cells	3,168	6,768	99,216
Number of slices	1,408	*3,008*	44,096
Max distributed RAM (Kb)	44	94	1378
18×18 multiplier blocks	12	*28*	444
18 Kb block RAM	12	*28*	444
Max block RAM (Kb)	216	504	7992
DCM	4	4	12
Max user I\O pads	204	348	1164
Notes	the smallest FPGA in V2P family	adopted in this study	one of the largest FPGA in V2P family

Table 3 Resource requirements for implementing 1024-point complex FFT using one Xilinx *System Generator* FFT block (Xil 2004a) and available resources from the selected FPGA

Required number for specified implementation	FFT Radix-4 streaming mode	FFT Radix-4 block mode	FFT Radix-2	Available number from the selected FPGA
Number of slices	3,840	2,849	*1,262*	*3,008*
18×18 multiplier blocks	18	18	*6*	*28*
18 Kb blocks RAM	27	11	*5*	*28*

4" are two of the variations of the FFT. In short, Radix-2 refers to the commonly seen implementation of the DFT of data points of a power of 2 as originally introduced in the Cooley-Tukey algorithm as appeared in textbooks (such as Oppenheim, *et al.* 1999). Radix-4 has an improved computational efficiency with the DFT of data points of a power of 4. The number of slices required for a 1024-point FFT Radix-4 (streaming mode) exceeds the selected FPGA capacity. Memory capacity is also exhausted as 27 out of 28 Block RAMs are consumed, thus there are not sufficient slices and/or memory available for implementing other proposed functions such as windowing, peak detection as well as ADC/radio interfaces. For the case of a 1024-point FFT Radix-4 (block mode), the resources are just enough for implementing the FFT, however there might be insufficient slices/memory for other proposed functions. An FFT Radix-2 implementation thus seems the most suitable option for the selected FPGA with sufficient margins.

4.1.2. Windowing function

To implement a 1024-point windowing function using fully combinational logic as in Fig. 2(b), 1024 multipliers would be needed to fulfill a concurrent operation in one clock cycle. According to Table 2, even the largest Virtex II Pro FPGA would not be sufficient to meet such a design requirement. Therefore, a designer needs to allocate a specific number of multipliers and then pipeline the data accordingly. Three design flows were considered for implementing the windowing function as detailed in Kapoor (2005), and a hybrid combination of serial and parallel arrangement of data flow was obtained for 1024-point windows with 50% overlapping as will be further presented in Section 4.2.

4.1.3. Data length

Kapoor (2005) provides an example of the calculation of block RAM requirements for this software design assuming a data length of 4096 points per analog channel with eight windows of 1024-point each and a 50% overlapping between windows. It was found that this data length would use up all the available block RAMs thus leaving no margins for other software implementations such as the ADC/radio interfacing. Therefore, the design was finally implemented with the capability to analyze 2048 collected data points per channel, consuming 23 out of 28 block RAMs. The windowing procedure uses four windows of 1024-point length with 50% overlapping.

4.1.4. Division or averaging

For the ease of implementation using an FPGA, it is desirable to handle divisions/averaging by a number that is a power of 2 (e.g. 2, 4, 8, 16 etc). For these numbers, the process of division in an FPGA is the most efficient by simply right-shifting the data bits (Roth 1998) rather than applying additional algorithms (e.g. Paschalakis and Lee 2003).

4.2. Channel subtraction and windowing function

Following the scope illustrated in Fig. 4(b), the software design flow began with the acquisition of two channels of data from the analog sensors using the **ADC interfacing algorithm**. This logic, developed using VHDL, is limited to the collection of a total of 2048 points per channel as discussed above. The collected data with 16-bit resolution was directly passed from the ADCs into the **channel subtraction algorithm**, which subtracted Channel 0 from Channel 1 data with zero latency. The result was then stored in a 2048 point 16-bit block RAM. After channel subtraction, the data was passed onto a **windowing algorithm**. This design of these two last algorithms was carried out using a Matlab Simulink-based *System Generator.*

To perform FFT on a non-periodic data set of finite length, it is imperative to first apply a windowing function on the incoming data from the ADC to avoid the Gibb's phenomenon (Hamming 1989, Oppenheim, *et al.* 1999). Many windowing functions are available, here a Chebyshev windowing function was selected due to its high relative side lobe attenuation and good main lobe width. Other windowing functions can be conveniently programmed into this software design under the Matlab Simulink-based *System Generator.*

Applying a windowing function to the subtracted data is the first situation in this study, where an FPGA manifests its superiority. In stead of using a serial execution as in microprocessor-based wireless sensing unit (see Fig. 3(a)), implementing such a function in an FPGA would only require the multiplication of two block RAMs with one containing the data, and the other, the windowing coefficients (Hun 2003) as illustrated in Fig. 3(b). As presented in Section 4.1, an execution involving only partial level of parallelism was adopted in this study as a result of the limited resources of the selected low-cost FPGA development board. Even then, the computational efficiency was improved many times as compared to a sequentially implemented option. In detail, four block RAMs were created from the parent block RAM after the channel subtraction. Each of these 1024-point 16-bit block RAMs were filled with data from the parent RAM in a sequential manner to form four windows of 1024-point length with 50% overlapping, where the last segment was zero padded to make up the size of 1024-points. Four multipliers were then applied in a parallel fashion to these four data block RAMs. This implementation consumed only 1024 clock cycles to complete the windowing, instead of 4096 cycles required by a sequentially implemented option.

4.3. Fast Fourier Transform (FFT) and calculating magnitude

The windowed data was then forwarded to an **FFT algorithm**, which performed FFT and subsequent squaring and addition of real and imaginary components on the four discrete windowed blocks using *System Generator*. This is the second situation in this study, where the option of FPGA manifests its superiority. One Xilinx *System Generator* block named "FFTx" was adopted in this design, which has been configured to use minimum FPGA resources by operating a 1024-point FFT Radix-2 at a block mode.

From Table 3, it can be seen that a single instantiation of this FFT block would consume about 40% of the available slices in the selected Xilinx XC2VP4 FPGA, therefore a parallel implementation of this block couldn't be carried out in this study, rather this block had to be utilized sequentially on the four windowed block RAMs. The output of each FFTx block contained the real and imaginary parts, and they were stored in separate block RAMs. For the purpose of calculating the magnitude of these FFT data, these real and imaginary data were then squared (by multiplying with themselves) and then added with each other and averaged. The result was again stored in another block RAM for further processing.

4.4. Peak detection algorithm

As shown in Fig. 4(a1), a Single-Degree-of-Freedom (SDOF) building system was used for validating the developed FPGA-based smart sensing unit. A single strong peak in the frequency data of the relative acceleration was thus expected in this proof-of-concept study. To implement the **peak detection algorithm**, a simple loop was implemented using a comparator. The loop cycled through the data block RAM and compared the current data value with a stored value. The stored value was updated to attain the maximum

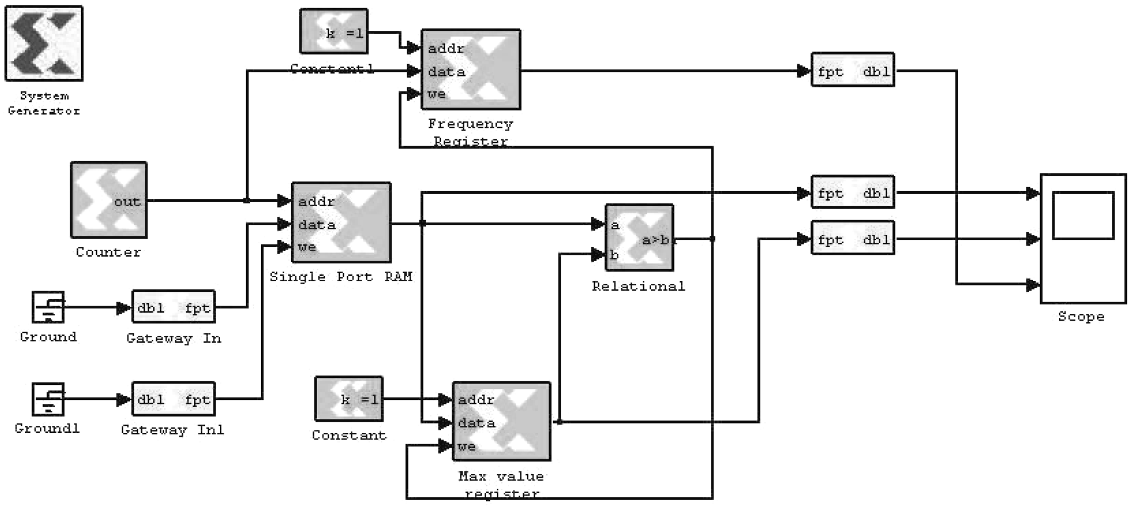

Fig. 6 A sample Matlab Simulink-based *System Generator* model used to implement the proposed peak detection algorithm. This Simulink block diagram was entirely constructed using Xilinx *System Generator* block sets. The data was feed through "Gateway In", and a scope was used to take the outputs. The key component, the comparator, was placed right on the center line of the diagram to identify the maximum frequency magnitude

data value as the loop progressed, meanwhile a register was used to store an integer (the location according to the maximum data value), which can be eventually translated into the frequency using the Nyquist frequency. For a 1024-point FFT and 100 Hz sampling rate adopted in this study, for example, a register of 512 is related to the Nyquist frequency of 50 Hz. The content of this frequency register was passed onto **radio interface algorithm**, and then transmitted to the base station and saved under the GUI. The initial value of the frequency register could be initialized as zero, however it was initialized as 19, corresponding to a frequency of $\frac{50}{512}\times 19 = 1.86\text{Hz}$ to filter out the low-frequency noise for the specific building model used in this study.

To showcase the convenience of using *System Generator* to create the DSP design flow specified in Fig. 4(b), Fig. 6 presents a Matlab simulink environment rather than a raw VHDL code to implement the proposed peak detection algorithm.

5. Validation

5.1. Simulation

Throughout the software implementation, simulations were performed to validate the proper functioning of each design algorithm (see Fig. 4(b)). In particular, Matlab Simulink-based Xilinx *System Generator* was used throughout to validate each proposed DSP algorithms (Kapoor 2005). A typical simulation performed to validate the implemented **channel subtraction algorithm** and **windowing algorithm** is presented in Fig. 7. This figure was originally generated by *System Generator* and then segmented into three stages column-wise, numbered and commented for legibility.

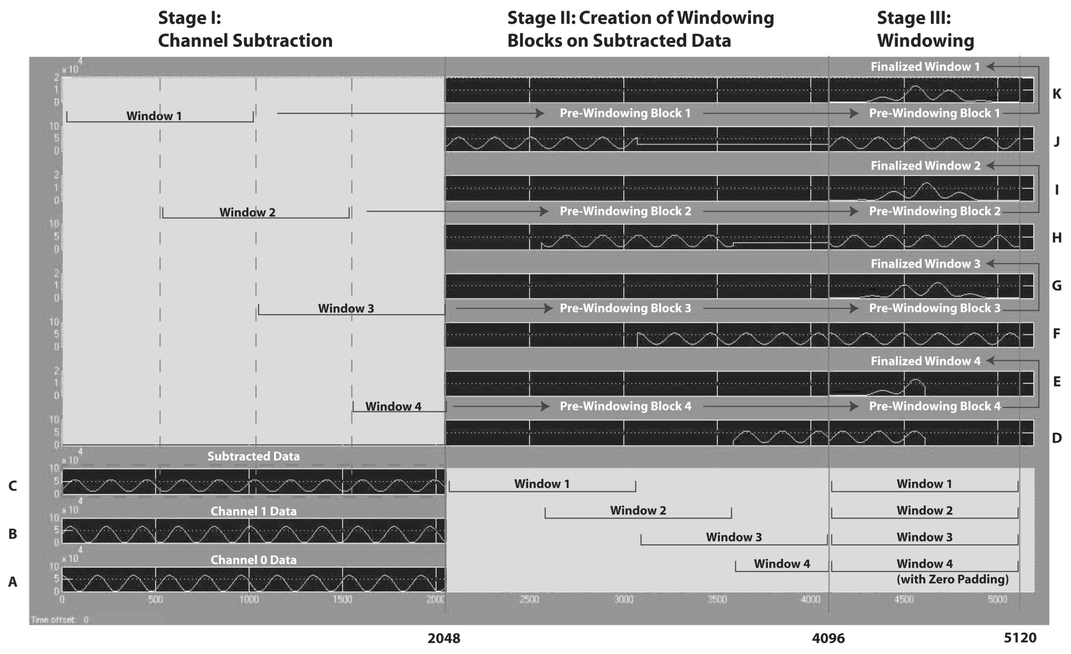

Fig. 7 Simulation for design related to channel subtraction, creation of windowing blocks and windowing. This plot was originally generated by *System Generator*, segmented and commented for legibility

The simulation began by executing Stage I, **channel subtraction**. Data retrieved from the ADCs are referred to as Channels 0 and 1. Two sine waves at Row A and B with a phase difference of $\pi/2$ were used as test signals. The two channels of data were subtracted on-the-fly (i.e. at real-time) and were stored in a 2048-point block RAM as shown in Row C.

As previously discussed in Section 4.1, four 1024-point windows with 50% overlap were adopted in this design in a sequential manner. Stage II of this simulation shows an intermediate step in creating these four data blocks. Rows J, H, F, and D represent the contents of these four data blocks following the time sequence. Although this process was sequential, the routing of data from the parent block was implemented in a parallel fashion so as to consume less clock cycles. As a result, only 2048 clock cycles were consumed versus a minimum of $1024 \times 3 + 512 = 3584$ clock cycles in a fully sequential design.

At Stage III shown in Fig. 7, Rows J, H, F, and D show the four 1024-point data blocks ready for windowing, while Rows K, I, G, and E show the corresponding four 1024-point windowed data blocks. The latter data blocks were then be stored for the subsequent **FFT algorithm**. As mentioned in Section 4.2, four multipliers were used in the **windowing algorithm** to consume only 1024 clock cycles versus 4096 clock cycles in a fully sequential design.

5.2. Laboratory testing

Before laboratory testing, the “pass through” mode defined in Section 2.1 was conducted to validate the proper functioning of the hardware interfacing of the external ADCs to the FPGA and the radio to the FPGA (Kapoor 2005). After a completed design was veried by all the simulations, it was converted into an NGC-Black Box, which was later incorporated into the VHDL wrapper code. This wrapper code accomplished the task of communicating with the ADCs and the radio. The entire design was then converted into a bit stream, a proper format for direct programming the FPGA. According to the two targeted modes of operations, there were two sets of bit streams obtained in this study, one was for the “pass through” and the other, the “smart” mode. Laboratory testing was performed on these two modes respectively.

The test setup is shown previously in Fig. 4(a1), where the height of the first floor mass can be adjusted through a simple clamping mechanism to conveniently vary the resonant frequency of the relative motion. Throughout this study, a periodic swept sine signal was used to drive the shaking table with a frequency varying from 100 mHz to 20 Hz linearly within 15 seconds; only the steady state signals were collected and the sampling rate was fixed as 100 Hz in the testing. A wall socket was used throughout this study to power the FPGA-based wireless sensing unit in this indoor testing validation for this proof-of-concept study.

Two testing configurations involving different heights of the floor mass were used to validate the developed FPGA-based smart wireless sensing unit. The floor mass was first fixed at the full height of the columns as shown in Fig. 4(a1). Under the “pass through” mode, the collected time histories are presented in Fig. 8(a). The power spectral density of the relative acceleration was plotted under Matlab using Welch’s method with four windows and 50% overlapping. This resonant frequency (i.e., the peak) was found to be 9.76 Hz as shown in Fig. 8(b). Immediately after the “pass through” mode, the FPGA was re-programmed to operate at the “smart” mode. Three consecutive readings of the frequency register were identifically 100, which indicated a resonant frequency of $\frac{50}{512}\times 100 = 9.76$ Hz.

The floor mass was then lowered to increase the stiffness of the system, which in turns led to a higher resonant frequency. Fig. 9(a) presents the collected time histories under the “pass through”

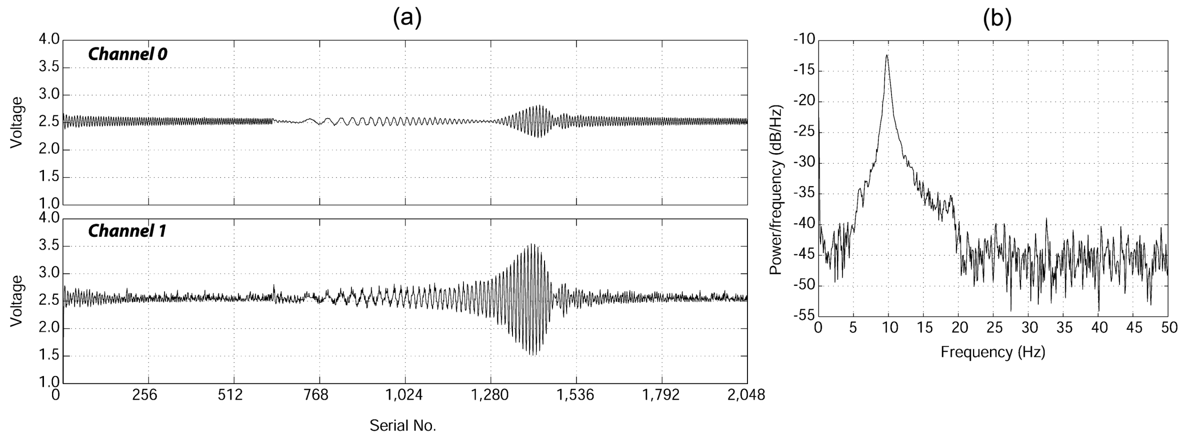

Fig. 8 (a) Collected time histories from Channels 0 and 1 under a "pass through" mode from the SDOF model at the full height of the first floor mass, and (b) power spectral density plot of Part (a) processed with Matlab Welch's method

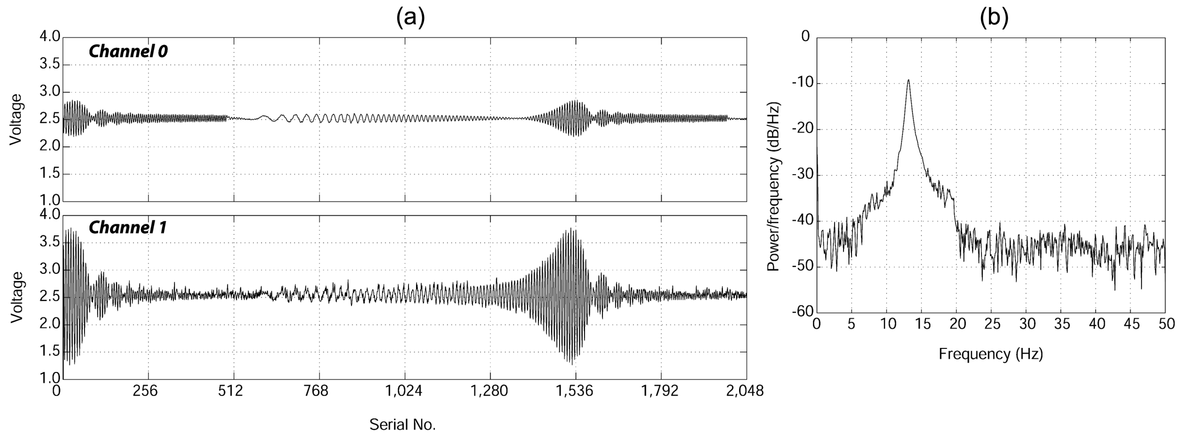

Fig. 9 (a) Collected time histories from Channels 0 and 1 under a "pass through" mode from the SDOF model at a shortened height of the first floor mass, and (b) power spectral density plot of Part (a) processed with Matlab Welch's method.

mode, while Fig. 9(b) shows the resonance frequency of 13.18 Hz calculated with Matlab Welch's method as before. Under the "smart" mode, three consecutively collected readings were 135, 135, and 134, which corresponded to 13.18 Hz, 13.18 Hz and 13.08 Hz, respectively, to make an average value of 13.15 Hz.

The consistency between Matlab and the FPGA processed results validated the proper functioning of the developed FPGA-based smart sensing unit. Also, these frequencies values compare favorably with those obtained from the same building models but using a 16-bit "wired" National Instrument data acquisition card and another 16-bit microprocessor-based wireless sensing unit developed by the authors and their coauthors in another study (Pei, *et al.* 2006b).

6. Discussion

For the FPGA-based smart wireless sensing unit developed in this study, the major limitations were imposed by the limited resources in the selected FPGA to maintain a low cost for a practical application. Longer data length (i.e. longer sampling time), higher sampling rate, and the implementation of frequency response function through a quotient function and even an autocorrelation function for handling real-world data (especially those with noise) are a few among the future extensions of this proof-of-concept design. It is envisioned that higher-end FPGAs and their development boards at a reduced cost will be available following the fast growing FPGA technology, thus these improved functions will be achieved still at a low cost.

Even though a low-cost and non-high-end FPGA off-the-shelf product has been adopted in this study to match the hardware cost of microprocessor-based off-the-shelf products used for structural health monitoring, overall computational efficiency has been greatly improved as compared with its sequentially implemented option in a microprocessor. These improvements when performing 1024-point FFT with 50% overlapping on a data set of 2048 points are mainly from, as presented previously, (1) the faster routing of the data from the ADC to the windowing algorithm (from 3584 to 2048), (2) the reduced number of clock cycles when implementing windowing (from 4096 to 1024), and (3) adopting the Xilinx FPGA-based “FFTx” algorithm, which is expected to be a huge gain (Xil 2004d). Fig. 2 gives a very simplied conceptional comparison of computational efficiency between a microprocessor- and an FPGA-based execution in terms of a fully sequential vs parallel execution mode. Other evidence is also available to show the improvement that commercially available FPGAs can bring in versus the fastest commercially available microprocessors in performing typical DSP algorithms (e.g. Xil 2004d) and certain bit-level communication processing (Wentzlaff and Agarwal 2004). The advantage in computational efficiency demonstrated in this study validates that these trends are true even in a non-high-end FPGA.

As a proof-of-concept study focusing on the hardware and software implementation related to FPGA-based smart wireless sensing for structural health monitoring, this study did not consider a mechanism to identify and/or address possible data loss. Since the reliability issue related to wireless data transmission is critical as studied by the authors and their co-authors (Pei, *et al.* 2006a), future versions of this FPGA-based wireless sensing unit or FPGA-based wireless sensor network should implement packetization protocols to make the data delivery performance robust to better serve the practical needs in structural health monitoring.

Many other topics for future improvement and exploration that can be built on this proof-of-concept study. FPGA related technologies are rapidly growing and power efficiency has been a topic that is drawing considerable attention and research efforts. The state-of-the-art low power FPGA products and power-aware design should also be explored for structural health monitoring.

7. Conclusion

An FPGA-based smart wireless sensing unit with the ability to perform onboard several FFT-related DSP algorithms has been developed at a cost comparable to a microprocessor-based counterpart. Hardware and software implementations related to the unit have been presented in this paper; in particular a Matlab Simulink-based high-abstraction tool for programming FPGAs has been explored and proved to be effective. Simulations and laboratory shaking table tests were carried out to validate the proper functioning of the sensing unit in both “pass-through” and “smart” modes. Even though a low-cost and non-high-end FPGA off-the-shelf product has been adopted in this study, overall computational efficiency has been improved many times in comparison with a sequentially implemented option as in a microprocessor. The proof-of-concept study showcases the authors’ vision that the rapidly growing FPGA technology in off-the-shelf products represents an invaluable opportunity to implement a wide range of system identification and damage detection algorithms for local data interrogation and smart sensing at drastically improved computational efficiency (and flexibility, and possibly, power efficiency).

Acknowledgements

This study is primarily supported by the National Science Foundation (SGER CMS-0332350) under Program Officer Dr. Steven L. McCabe. The generous donation from the university program at Xilinx, Inc. (XUP) is gratefully acknowledged. In particular, several helpful discussions with Mr. Jeff Weintraub at XUP are greatly appreciated. In addition, the assistance during laboratory testing offered by Mr. Michael F. Schmitz from the School of Civil Engineering and Environmental Science at the University of Oklahoma is greatly appreciated.

References

ACM-Association for Computing Machinery (2002), *Proceeding of the 2002 ACM/SIGDA 10th international Symposium on Field Programmable Gate Arrays*, Monterey, California, USA, 2002., Sponsored by ACM, Special Interest Group on Design Automation (SIGDA).

ACM-Association for Computing Machinery (2003), *Proceeding of the 2003 ACM/SIGDA 11th international Symposium on Field Programmable Gate Arrays*, FPGA-FFT 20, Monterey, California, USA, Sponsored by ACM, Special Interest Group on Design Automation (SIGDA).

ACM-Association for Computing Machinery (2004), *Proceeding of the 2004 ACM/SIGDA 12th International Symposium on Field Programmable Gate Arrays*, Monterey, California, USA, 2004., Sponsored by ACM, Special Interest Group on Design Automation (SIGDA).

Choi, S., Scrofano, R., Prasanna, V. K. and Jang, J.-W. (2003), “Energy-efficient signal processing using FPGAS”, *Proceeding of the 2003 ACM/SIGDA 11th International Symposium on Field Programmable Gate Arrays*, pages 225-234, Monterey, CA, February 23-25 2003. ACM 1-58113-651-X/03/0002.

Chopra, A. K. (2000), *Dynamics of Structures: Theory and Applications to Earthquake Engineering*, Prentice Hall, 2 edition.

Dehon, A. and Wawrzynek, J. (1997), *The Case for Reconfigurable Processors*, January.

Farrar, C. (2004), “Converting large sensor array data into structural health information”, a speech delivered at a special session on ‘*distributed sensing for structural systems*’, *4th International Workshop on Structural Control (4IWSC)*, Columbia University, New York, NY, USA, 10-11 June.

Hamming, R. W. (1989), *Digitial Filters, 3rd ed.* Prentice Hall, pp. 284.

Implementing an FFT with the HERON-FPGA Family (2003), Hunt Engineering, http://www.hunt-dsp.com.

Inman, D. J. (1994), *Engineering Vibration*, Prentice Hall.

Kapoor, C. (2005), “Reliable and smart wireless sensing units for structural health monitoring”, Master’s thesis, The University of Oklahoma.

Kapoor, C., Graves-Abe, T. L. and Pei, J. S. (2005), “Development of an off-the-shelf field-programmable-gate array based wireless sensing unit for structural health monitoring”, *SPIE International Symposia – Smart Structures* *and Materials/NDE*.

Kung, C.-H., Devaney, M. J. and Jung, C.-M. (2002), "The VLSI implementation of an artificial neural network scheme embedded in an automated inspection quality management system", *IEEE Instrumentation and Measurement Technology Conference*, pages 239-244, Anchorage, AK, USA, 21-23 May.

Liu, S.-C. and Tomizuka, M. (2003), "Strategic research for sensors and smart structures technology", *Structural Health Monitoring and Intelligent Infrastructure*, pages 113-117, Tokyo, Japan, 13-15 November, *Proceedings of the First International Conference on Structural Health Monitoring and Intelligent Infrastructure*, edited by Z.S. Wu and M. Abe, Swets & Zeitlinger B.V., Lisse, The Netherlands.

Lynch, J. P., Law, K. H., Kiremidjian, A. S., Carryer, E., Farrar, C. R., Sohn, H., Allen, D. W., Nadler, B. and Wait, J. R. (2004), "Design and performance validation of a wireless sensing unit for structural monitoring applications", *Struct. Eng. Mech.*, **17**(3-4), 393-408.

Mahgoub, I. and Ilyas, M. (2006), *Smart Dust: Sensor Network Application, Archietcture, and Design*. CRC Taylor& Francis.

MATLAB. The Mathwork Inc. (2005), http://www.mathworks.com/products/matlab/.

Maxfield, C. (2004), *The Design Warrior's Guide to FPGAs: Devices, Tools and Flows*. Newnes.

Maxim MAX1165 Datasheet (2004a), Maxim Integrated Products, Inc, http://www.maxim-ic.com/.

Memec Insight DS-KIT-2VP4LC Product Information (2004), Memec Insight, Inc, http://www.insightelectronics.com.

NASA. Hilbert-huang transform (2004), http://techtransfer.gsfc.nasa.gov/HHT/HHT.htm.

Ohtani, K., Baba, M. and Notohara, D. (2002), "An intelligent position sensor for light spots using an analog scancircuit and FPGA", *IEEE Instrumentation and Measurement Technology Conference*, pages 711-715, Anchorage, AK, USA, 21-23 May.

Oppenheim, A. V., Schafer, R. W. and Buck, J. R. (1999), *Discrete-Time Signal Processing*, Prentice Hall, 2nd edition.

Paschalakis, S. and Lee, P. (2003), "Double precision floating-point arithmetic on FPGAs", *2nd International Conferenceon Filed Programmable Technology (FPT2003)*, pages 352-358, Tokyo, December 15-17.

Pei, J. S., Kapoor, C., Graves-Abe, T. L., Sugeng, Y. P. and Lynch, J. P. (2006a), "An experimental investigation of the data delivery performance of wireless sensing units composed of off-the-shelf components for structural healht monitoring", *Structural Control and Health Monitoring*, 2006a, accepted for publication

Pei, J. S., Kapoor, C., Graves-Abe, T. L., Sugeng, Y. P., Lynch, J. P. and Ferzli, N. A. (2006b), "An experimental investigation of the data quality in a wireless sensing unit as affected by hardware design and operational conditions", *Mechanical Systems and Signal Processing*, in preparation.

Roth, C. H. Jr. (1998), *Digital System Design Using VHDL*, PWS Publishing Company.

Shang, L., Kaviani, A. S. and Bathala, K. (2002), "Dynamic power consumption in VIRTEX-II FPGA family", *Proceeding of the 2002 ACM/SIGDA 10th International Symposium on Field Programmable Gate Arrays*, pages 157-164, Monterey, CA, USA, February 24-26, FPGA-FFT 22.

Smyth, A. W. and Betti, R. (2004). *Proceedings of the Fourth International Workshop on Structural Control*, 2004. International Association on Structural Control (IASC), DesTech Press.

Spencer, B. F. Jr. (2003), "Opportunities and challenges for smart sensing technology", *Structural HealthMonitoring and Intelligent Infrastructure*, pages 65-71, Tokyo, Japan, 13-15 November 2003. *Proceedings of the First International Conference on Structural Health Monitoring and Intelligent Infrastructure,* edited by Z.S. Wu and M. Abe, Swets & Zeitlinger B.V., Lisse, The Netherlands.

Tredennick, N. and Shimamoto, B. (2003), Go reconfigure. *IEEE Spectrum*, pages 36-40.

Tuan, T., Kao, S., Rahman, A., Das, S. and Trimberger, S. (2006), "A 90 nm low-power FPGA for battery-powered applications", *Proceeding of the Internation Symposium on Field Programmable Gate Arrays*, pages 3-11, Monterey, CA, USA, February 22-24.

Verkest, D. (2003), "Machine chameleon, a sneak peek inside the handheld of the future", *IEEE Spectrum*, 41-46.

Wentzlaff, D. and Agarwal, A. (2004), "A quantitative comparison of reconfigurable, tiled, and conventional architectures on bit-level computation", *Proceedings of the 12th Annual IEEE Symposium on Field-Programmable Custom Computing Machines table of contents*, pages 289-290.

Wolf, W. (2004), *FPGA-Based System Design*. Prentice Hall.

Xilinx LogiCORE FFTx Block Datasheet (2004a), Xilinx, Inc, http://www.xilinx.com.
XStream-DEV Wireless Development Kits 900 MHz & 2.4 GHz (2004b), MaxStream, Inc, http://www.maxstream.net.
Virtex II Datasheet (2004b), Xilinx, Inc, http://www.xilinx.com.
Virtex II Pro Datasheet. (2004c), Xilinx, Inc, http://www.xilinx.com.
Presentation Materials at 2004 Digital Signal Processing Design Flow Workshop (2004d), Xilinx Inc.
Xilinx CORE Generator Information (2004e), Xilinx, Inc, http://www.xilinx.com.
Xilinx System Generator Information (2004f), Xilinx, Inc, http://www.xilinx.com.
Yalamanchili, S. (2001), *Introductory VHDL: From Simulation to Synthesis*, Prentice Hall.

CC