Towards a High-Performance Platform for Sonic Interaction Interfaces

Stefano Fasciani
Faculty of Engineering and Information Sciences,
University of Wollongong in Dubai
Department of Musicology,
University of Oslo
stefano@fasciani.xyz

Manohar Vohra
Faculty of Engineering and Information Sciences,
University of Wollongong in Dubai
mv800@uowmail.edu.au

ABSTRACT

In this paper we introduce a hardware platform to prototype interfaces of demanding sonic interactive systems. We target applications featuring a large array of analog sensors requiring data acquisition and transmission to computers at fast rates, with low latency, and high bandwidth. This work is part of an ongoing project which aims to provide designers with a cost effective and accessible platform for fast prototyping of complex interfaces for sonic interactive systems or musical instruments. The high performances are guaranteed by a SoC FPGA. The functionality of the platform can be customized without requiring significant technical expertise. In this paper, we discuss the principles, the current design, and the preliminary evaluation against common microcontroller-based platforms. The proposed platform can sample up to 96 analog channels at rates up to 24 kHz and stream the data via UDP to computers with a sub millisecond latency.

1. INTRODUCTION

The user interface is an essential component in sonic interactive systems. The manipulation of such an interface determines aspects of the sound generation process that in turn affects the user's manipulation [1]. The closed-loop architecture requires tight coupling, or low latency, between action and auditory feedback. This requirement holds across a large spectrum of applications, including interactive sonic installations, electronic musical instruments, Virtual Reality (VR) and videogames.

Independent of the input modality, these interfaces integrate sensors to transduce the user's gesture into electric signals. Features extracted from these signals control parameters of the sound synthesis by explicit or generative mapping techniques [2]. However, before extracting features, the analog signals must be sampled, digitized, and then transferred to a computational system. The platform we introduce in this paper is designed to carry out these tasks in context to requiring large number of channels, a high sampling rate, low latency and a large bandwidth towards the sound synthesis module.

When prototyping sonic interactive systems, the hardware for acquisition of analog signals is often taken off-the-shelf due to a lack of time or expertise. Moreover, choices are often oriented towards platforms that are easy to customize and integrate, even if it is not specifically designed for this application domain. This allows focusing on gesture-capturing, mapping, sound synthesis and creative applications, which are core components of sonic interaction studies. However, these platforms deliver poor sampling rates and latency with a large number of sensors, not to mention the incapability of simultaneous sampling of signals. The detrimental consequences can significantly deteriorate the resulting sonic feedback and the overall user experience.

High-end data acquisition systems that can provide sufficient performances are available on the market, but these are costly and cannot be customized. Moreover, their latency is often undocumented or excessive as they are not designed for sonic interactive applications. The development of such data acquisition system requires a significant engineering effort which is not suitable in resource-constrained projects, or when rapid prototyping is needed. The platform we describe here enables fast-prototyping of interfaces for demanding sonic interaction systems. We selected a cost-effective, yet powerful platform based on a System on Chip (SoC) Field Programmable Gate Array (FPGA) which can be customized by designers without modifying the internal architecture.

2. EXISTING PLATFORMS

In the last decade we assisted to the proliferation of microcontroller-based boards and single-board computers that made prototyping of interactive systems accessible to everyone. The cost of these boards is significantly lower than those produced by silicon manufacturers. In addition, integrated development tools, rich sets of libraries, online user communities, and the open-source nature have drastically simplified the development workflow.

2.1 Microcontroller-based Platforms

The concept of accessible development boards was pioneered by Arduino. Introduced in 2005, this platform aimed at reducing cost of student's projects at the Interaction Design Institute Ivrea, and providing an easy-to-use Integrated Development Environment (IDE) [3]. The popularity and pedagogical value of Arduino was also determined by the simplified approach to introduce complex hardware and software concepts in their tutorial and example projects [4]. This enabled novices to implement embedded systems for physical computing in a relatively short amount of time. Most Arduino boards are based on 8-bit AVR microcontrollers. The popular Arduino Uno, released in 2010, is equipped with an ATmega328P running at 16 MHz. Thereafter, a variety of compatible expansion boards (or shields), and clone or compatible

ISBN 978-84-09-08518-7

boards have been commercialized. These support the same Application Programming Interface (API), Hardware Abstraction Library (HAL), libraries and IDE of the original Arduino boards, making programs easier to port across platforms. Interaction designers can choose among a wide spectrum of easy-to-use boards with different size, computational power, interfacing capabilities, and price. An increasing number of Arduino compatible boards feature an ARM Cortex-M 32-bit microcontroller. These run at higher clock rates than the AVR, provide native 32-bit computation, a nested vector interrupt controller and a richer set of I/O.

Despite the relatively high-level API, programming in Arduino IDE is generally bare-metal. Libraries supporting threads and simple real-time Operating Systems (OS) that do not require Memory Management Units (MMU), such as FreeRTOS [5], are available for both AVR and ARM Cortex-M architectures. Easy-to-use libraries for sound synthesis, such as Mozzi [6] are available to designers. Firmata [7] further minimizes the programming burden for designers using Arduino compatible boards only to capture sensors data and transfer it to a computer. Firmata bundles a microcontroller program, a communication protocol, and clients for programming environments including a Graphical User Interface (GUI) to configure the physical I/O.

A more recent platform that is gaining popularity among interaction designers is ARM Mbed, a collaborative project managed by ARM Holdings. ARM Mbed boards are cost effective and produced by several manufacturers, including major semiconductor companies. These boards are based on the ARM Cortex-M architecture (a few boards mount a Cortex-A), sharing the same API, HAL and IDE. Compared to the Arduino, ARM Mbed provides a more sophisticated OS and software libraries, along with powerful microcontrollers (clock rates between 30 and 200 MHz) and a richer set of peripherals. In terms of programming complexity, ARM Mbed is as accessible and well supported as Arduino. C libraries for sound synthesis can be easily ported to Mbed platforms, such as OOPS [8].

Although Arduino and ARM Mbed are relatively easy-to-use platforms, but the development of a complete sonic interactive system on these platforms is challenging due to a lack of high-level programming languages and OS. Indeed, these are generally used to capture sensor data and transfer it to computers, where feature extraction, mapping and synthesis can be easily implemented using high-level programming environments. Critical information such as maximum sampling rate on multiple analog inputs, synchronicity, bandwidth, and latency of communication channels are not available nor can be estimated from the documentation. Hence it is difficult to select a platform matching the design requirements.

2.2 Single-board Computers

To date, a large variety of single-board computers are available in the market [9]. The majority are equipped with a single or multi-core 32-bit ARM Cortex-A with MMU and clock rates typically within 1 to 2 GHz. Recent boards are switching to 64-bit ARM Cortex-A architectures. Most boards support at least one Linux distribution, whereas Android, Windows CE and BSD are other common OSs. The most popular are the Raspberry Pi and the Beagleboard. Both are affordable, well supported and relatively simple to use. Both support Debian OS and provide an HDMI video output, hence the development workflow on these platforms is much alike to general-purpose computers. For instance, it is possible to use high-level audio programming environment such as Pure Data, FAUST and Csound.

The computational power of these single-board computers is suitable to implement simple end-to-end sonic interactive system, from gestural signal acquisition to sound synthesis, although only a minority of these provides on-board Analog to Digital Converter (ADC). Also, response latency and jitter of these platforms do not meet the minimum requirement of most sonic interactive systems. These issues were addressed by Satellite CCRMA [10], and particularly by Bela [11]. Bela integrates an expansion board with multiple analog I/O, a customized real-time operating system, and an audio driver delivering a sub-millisecond gesture-to-sound latency. It provides a stereo audio input and output with a sampling rate of 44.1 kHz and 16-bit resolution, 8 analog inputs and 8 analog outputs for sensors and actuators with a sampling rate of 22.05 kHz and 16-bit. Using a dedicated multiplexer board, the number of analog inputs can be expanded to 64 with 2.75 kHz sampling rate.

Single-board computers provide only a fraction of the computational power and memory available on general-purpose computers, hence the implementation of demanding sonic interactive applications may not be possible. Using these boards only to capture sensor data and transmit it to a computer is wasteful in term of computational resources, while latency and jitter are higher than microcontroller-based boards due to the OS.

2.3 FPGA-based Platforms

FPGAs can meet demanding requirements with respect to throughput and latency. Recent technological advancements have enabled the manufacturing of powerful but relatively low-cost and low-power FPGA chips [12]. The Xilinx 7th Series FPGAs are featured on Digilent boards and are available for approximately 100 USD, including Spartan, Artix and Zynq FPGAs [13], [14]. Tools and languages for FPGA development have significantly improved, but the workflow to design and implement an FPGA-based system still requires significant engineering expertise. Platforms such as Mojo, Papilo (both featuring an old Spartan 6), and Alchitry provided developers with a low-cost, small-size, and developer-friendly boards, including an Arduino-compatible connector. However, tools are still far from being accessible when compared to platforms discussed in the previous sections. Hardware Description Language (HDL) still represents as an entry barrier for most designers. Arduino has recently launched the MKR Vidor 4000, a board featuring a 32-bit ARM Cortex-M and a small Intel (formerly Altera) Cyclone 10 FPGA. The company has announced, but not yet disclosed, a promising and easy-to-use online visual programming tool to program the chip.

ISBN 978-84-09-08518-7

To date, only a handful of sonic interactive systems or New Interfaces for Musical Expression (NIME) feature an FPGA-based platform, such as Overholt's Matrix [15], the continuous keyboard by Freed and Avizienis, the SLABS by Wessel, Freed and Avizienis [16], a physical modeling drum controller by Chuchaz, O'Modhrain and Woods [17], a controller for physical modeling synthesis by Pfeifle and Bader [18], and the Arcontinuo by Cadiz and Sylleros [19]. Freed, Avizienis, Wessel and Wright underlined necessity for high-performance interfaces (in terms of bandwidth and latency) between an array of sensors and computers almost two decades ago [20], [21]. A few FPGA-based platforms to capture gestural data from sensors have been proposed, such as SensorLab from Steim, the Connectivity Processor from CNMAT [22], and the Gluion [23]. Bandwidth and latency benchmark for the SensorLab and the Gluion are not available, but analyzing their technical features is evident how these cannot match the performances of the Connectivity Processor, that features 10 channels of 24-bit audio sampled at 48 kHz, 64 channels of sample-synchronous control-rate gesture data sampled at 6 kHz, and overall latency of 7 milliseconds with Max/MSP. None of these systems is on the market nor details for their fabrication are available.

2.3.1 The SLABS

The SLABS is among the most complex and expressive musical interfaces ever built. The elegant and effective design is the result of almost a decade of development by Avizienis, Freed and Wessel at CNMAT. It features a matrix of touch-sensitive pads that can be mapped to computer software in a variety of ways. The SLABS "was designed to engage the body, to be both musically expressive and inspiring, to be easy to play at the entry level, and to be accepting of a lifelong development of virtuosity" [16]. Each pressure-sensitive touchpad produces three analog signals related to the touch coordinates and pressure. These are sampled at a high rate and sent to the computer via Ethernet as audio streams, enabling a tight coupling between gesture and data. Gestural data can be used directly as a signal within the synthesis algorithm or processed for the fast detection gestures. Wessel developed various Max/MSP patches interacting with the SLABS, including demos with simple oscillators, granular synthesis, and control of percussive loops.

Two versions of the SLABS have been fabricated, featuring respectively 24 and 32 VersaPad pressure-sensitive touchpads by Interlinks. The VersaPad signal acquisition hardware was modified to enable simultaneous sampling at high rate [24], [25]. In particular, the 96 analog signals (three from each touchpad) are first sampled at 6 kHz, then up-sampled to match the audio rate of 48 kHz and sent to the computer running Max/MSP as multichannel UDP audio stream. The SLABS can also transmit sensor data via Open Sound Control but at a much lower rate.

The core of the SLABS is the Xilinx Virtex-4 FX12 Evaluation Kit manufactured by Avnet. The module features an SXC4VFX12-FF668 SoC FPGA with an embedded PowerPC core running at 300 MHz, 64 MB of DDR SRAM, 8 MB of flash memory, and a Gigabit Ethernet interface. The module is installed on a custom motherboard, as visible at the top of Figure 1, that includes other I/O such as ADAT and MIDI, and the ADC chips sampling the signals from the touchpads. Samples are converted to 32-bit floating point before UDP transmission, therefore the bandwidth of the payload between the SLABS and the computer (excluding UDP framing) is approximately 147.5 Mbit/s. The cost of the FPGA module is approximately 450 USD. Details on the motherboard are not available, such as the model and resolution of the ADC chips, however, the layout of the board reveals 8 ADC chips sampling 12 signals each. We estimate the cost of the ADC chips to be about 150 USD.

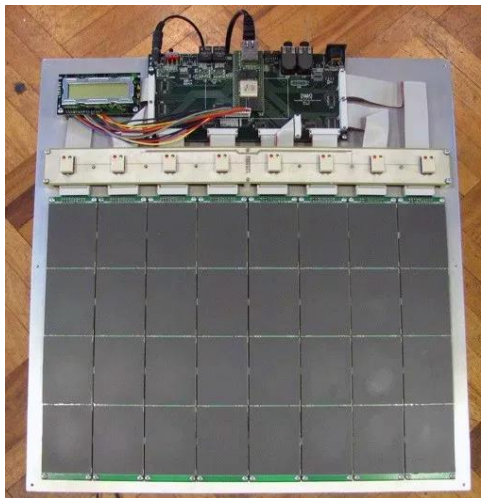

Figure 1. The SLABS32, with motherboard and FPGA module visible on the top of the matrix of touchpads.

3. BENCHMARK OF MICROCONTROLLER-BASED PLATFORMS

As discussed in Section 2, microcontroller-based platforms are suitable and easy-to-use to acquire and transfer a small number of analog signals to a computer. From their documentation, it is not possible to determine the resulting latency and maximum sampling rate when acquiring a given number of analog signals. This is due to the complex interdependency between the software and hardware architectures. McPherson, Jack and Moro proposed a setup to measure latency and jitter on these platforms [26]. Their latency measurements include computer-based sound generation, showing how popular boards such as Arduino Uno and Teensy 2.0 struggle to achieve end-to-end latency below 10ms. These measurements were based on the acquisition of a single channel. The latency can increase with a larger number of inputs.

The benchmarks we present here focus on bandwidth and the measurement setup does not include a computer-based sound generation unit. In particular, we investigate the maximum rate at which multiple signals can be acquired and transferred to computers. We selected three boards: the Arduino Uno, the Teensy 3.6 by PJRC, and NUCLEO-F746ZG by ST Microelectronics. The Teensy 3.6 and the NUCLEO-F746ZG are respectively top-of-

ISBN 978-84-09-08518-7

the-line among Arduino and Cortex-M ARM Mbed boards. According to the specifications, the on-chip USB of the Teensy 3.6 microcontroller outperforms the one in the NUCLEO-F746ZG, while the on-chip Ethernet of the NUCLEO-F746ZG microcontroller outperform any Ethernet expansion module for the Teensy 3.6. For this reason, we did not carry out Ethernet-based benchmarks on the Teensy 3.6 and USB-based benchmarks on the NUCLEO-F746ZG. Links based on IEEE 802.11 wireless LAN standards were not considered due to their excessive latency. The benchmark programs use only standard libraries and API for Arduino and Mbed IDE available to average developers. Where possible, the programs configure the link and ADC to run at the maximum speed using simple instructions only. We did not use Direct Memory Access (DMA) to improve performances as this option is not available through standard API, but it requires microcontroller-specific expertise.

The Arduino Uno features an 8-bit AVR ATmega328P running at 16 MHz. We measured data transmission to the computer via USB through the onboard ATmega8U2 acting as a USB-to-Serial with a baud rate of 2 Mb/s, and via the 10/100 Ethernet through the W5500 Ethernet controller on the Ethernet Shield 2. We increased ADC conversion clock from the default 125 kHz to 8 MHz. The Uno can acquire up to 6 analog inputs multiplexed to a single bit 10-bit ADC, and up to 11 digital inputs (serial transceiver and LED lines are excluded).

The Teensy 3.6 features a 32-bit Freescale ARM Cortex-M4 processor with the Floating-Point Unit (FPU) running at 180 MHz. We measured data transmission to the computer through the on-chip USB peripheral configured as a USB-to-Serial with a baud rate of 4.608 Mb/s. We overclocked the Cortex-M4 to 240 MHz, and compiled the code using the Fastest optimization option. The Teensy 3.6 can acquire up to 24 analog inputs multiplexed to two 12-bit ADCs, and up to 55 digital inputs. Simultaneous sampling of two channels at a time is possible, but the available libraries provided worst performances than sequential sampling used in the benchmark.

The NUCLEO-F746ZG features a 32-bit ST Microelectronics ARM Cortex-M7 with FPU running at 216 MHz. We measured data transmission to a computer via User Datagram Protocol (UDP) through the on-chip 10/100 Ethernet Media Access Controller (MAC) with dedicated DMA. The NUCLEO-F746ZG can acquire up to 24 analog inputs multiplexed to three 12-bit ADCs, and up to 90 digital inputs. Simultaneous sampling of three channels at a time is possible only by complex low-level register programming and DMA. The simple Mbed API for sequential sampling was used within the benchmarks.

The benchmark programs repeatedly acquire a set of analog and/or digital inputs and sent a packet of data to the computer via USB or Ethernet, without any synchronization mechanism. The computer measures the average inter-arrival time between packets, reflecting the highest rate at which the board can sample and transmit the analog signals. The result of the ADC conversion (10 or 12-bits) is stored in a 16-bit integer, requiring the transmission of 2 bytes. The inter arrival time of packets was measured in Pure Data for USB link (Virtual COM Port), and in Wireshark for the Ethernet link. Details of the benchmarks and results are summarized in Table 1, which shows the maximum rates when acquiring different set of analog and/or digital inputs. We evaluated the performances ranging from a single input to as many inputs as supported by the specific platform. Certain set were selected to facilitate comparison across platforms.

Platform	Link	Digital Inputs	Analog Inputs	Bytes Payload	Bytes Sent	Max Rate kHz
Arduino Uno	USB	1	0	1	1	53.7
		11	0	2	2	15.1
		11	0	11	11	7.1
		0	1	2	2	35
		0	6	12	12	6.2
		11	6	14	14	5.5
	Ethernet	1	0	1	60	2.3
		11	0	2	60	2
		11	0	11	60	2
		0	1	2	60	1.9
		0	6	12	60	0.9
		11	6	14	60	0.8
Teensy 3.6	USB	1	0	1	1	740
		11	0	2	2	378
		11	0	11	11	96.8
		36	0	5	5	222
		55	0	7	7	154
		0	1	2	2	127
		0	6	12	12	22.9
		0	24	48	48	5.8
		11	6	14	14	22
		36	24	53	53	5.6
		55	24	55	55	5.4
NUCLEO-F746ZG	Ethernet	1	0	1	60	17.5
		11	0	2	60	17.5
		11	0	11	60	17.5
		36	0	5	60	16.3
		55	0	7	60	15.8
		90	0	12	60	14.9
		0	1	2	60	15
		0	6	12	60	9.2
		0	24	48	90	3.8
		11	6	14	60	9.1
		36	24	53	95	3.8
		55	24	55	97	3.7
		90	24	59	101	3.6

Table 1. Benchmark details and results on maximum data acquisition rate for microcontroller-based platforms.

As expected, the maximum rates drop with the increase of the number of inputs. CPUs perform instructions sequentially, and the number of operations to perform increases linearly with the number of inputs. Further analysis demonstrated that the data transfer represents the bottleneck of these system, which keeps the CPU busy for a dominant fraction of the execution time. Getting the data from the ADC requires only a small fraction of the execution time. The aggregate sampling rates are significantly below the nominal maximum sampling rate of the ADCs of the platforms. Packing the data, especially for the digital inputs, present an overall advantage, as demonstrated in the tests with 11 digital inputs. When sending data via

ISBN 978-84-09-08518-7

Ethernet, we obtain poor performances with a small number of inputs due to the minimum payload of 18 data bytes in UDP packets, which requires additional 42 framing bytes. Buffering more samples per channel provide slightly higher rates but increase the latency.

For Arduino-compatible platforms, we also measured the maximum reading rates of Firmata [7] using the same baud rates and clock settings described above. Compared to previous benchmarks, Firmata presents a larger overhead when transmitting data over serial, since data is transmitted with an additional integer channel identifier, and samples from analog inputs are sent as 32-bit float numbers. By default, inputs are sampled approximately every 20 ms (50 Hz sampling rate), and this interval can be changed within the source code. Analog and digital data is transmitted to the computer only if the values are different from previously sent data. Therefore, the duration of one iteration of the loop can vary significantly, and signal sampling at regular interval cannot be guaranteed. On the Arduino Uno, Firmata supports up to 11 digital inputs and 6 analog inputs. Whereas on the Teensy 3.6, we have up to 36 digital inputs and 16 analog inputs. To benchmark the maximum rate of change that Firmata can handle on digital inputs, we drove these pins with a square wave and disabled analog input data acquisition. We measured the respective rate of change in the Firmata client for Pure Data. We increased the frequency of the square wave until measurement mismatching was above 1%. We repeated this measurement driving respectively 1, 11, and one 36 digital inputs (Teensy 3.6 only). In a separate benchmark we measured the maximum rate at which Firmata can acquire analog inputs, which were connected to a white noise source, while digital input acquisition was disabled. The inter-arrival time of packet of samples was still measured in the Firmata client for Pure Data. In both benchmarks, the sampling interval was set to 0 ms, forcing the data acquisition loop to run as fast as possible. Results are summarized in Table 2, where rates are rounded down to the nearest integer. For the digital inputs, the maximum rate is twice as the highest frequency of the square wave that the platform can acquire, taking one reading on both low and high levels. For the analog inputs, we can sample signals with spectral components up to half of maximum rates.

Platform	Inputs	Max Rate
Arduino Uno	1 digital	24 kHz
	11 digital	14 kHz
	6 analog	4 kHz
Teensy 3.6	1 digital	108 kHz
	11 digital	78 kHz
	36 digital	28 kHz
	6 analog	18 kHz
	16 analog	7 kHz

Table 2. Benchmarks on maximum data acquisition rate for analog and digital inputs using Firmata.

All considered platforms feature Successive Approximation Register (SAR) ADCs, which contributes to minimize the latency from acquiring the electrical signal to completing the associated data transmission. The latency from signal acquisition to data transmission was not explicitly measured. However, since the CPU iterates on a branch-free sequential loop that includes acquisition, data packing and transmission, the latency is at most one sampling period. This is significantly less than 1 ms and can be further reduced by interleaving data acquisition and data transfer when acquiring multiple channels. However, this can slightly worsen the maximum acquisition rate.

4. HIGH-PERFORMANCE PLATFORM PRINCIPLES AND DESIGN

The systems we reviewed in Section 2 can sample multiple gestural signals with high temporal resolution. Gestural samples are locked to the audio sampling clock. This synchronous approach provides jitter free encoding of the gestures resulting in more control intimacy. Indeed, Wessel argues that “the high-bandwidth approach is the future when it comes to ultra-expressive electronic instruments because it allows so much performance data to be captured”. McPherson and Zappi state that using high sampling rates enables “capturing subtle details like audio-rate vibrations or detailed temporal profiles within sensor signals” [27]. The development of customized data acquisition systems with such characteristics is time consuming and requires significant expertise. Hardware cost, especially with flagship FPGAs from the Virtex family, are likely to exceed 1000 USD. Maintaining and preserving these complex artifacts is also a challenge, due to the rapid obsolescence of hardware and software [28].

The aim of this work is to provide designers with a ready-to-use platform, which is cost effective and simple to customize to meet specific application requirements. The platform supports a common design pattern where the hardware gesture controller is attached by a high-speed link to a computer running a media programming language such as Max/MSP or Pure Data. The principles we followed in our design include:

- FPGA-based platform with core functionalities implemented in hardware minimizing latency and jitter.
- Number of analog inputs and sampling rate at least matching the data acquisition system of the SLABS.
- Performances independent of the number of inputs.
- Set of acquisition expansion board supporting simultaneous or sequential sampling of the analog inputs.
- Filter bank for fast signal processing on board.
- Flexible and easy-to-configure number of inputs, sampling rate, buffer size, and filter coefficients.
- Cost in the range of 150 to 250 USD.

The simultaneous sampling of a large group of analog signals is costly because it requires one ADC per input (or at least one sample and hold circuit per input). Simultaneous sampling is necessary when phase information exists between different signals. Otherwise, it’s possible to use a fast ADC and analog multiplexers to sample sequentially all inputs but losing the phase information between signals. As platform designers, we cannot predict if preserving the phase relationship between signals would be required in the final application. However, this has a significant impact on the acquisition hardware and cost. Therefore, as detailed in Section 4.2, we propose three

ISBN 978-84-09-08518-7

acquisition subsystems: fully simultaneous, fully sequential, and a tradeoff between the two. For instance, in presence of multiple pressure-sensitive touchpads, sampling simultaneously three signals at a time is a tradeoff between cost and performance.

4.1 System Architecture

The platform we selected to implement the system is the Digilent Cora Z7-07S [29], which features an Zynq-7000 SoC including an FPGA and an ARM Cortex-A9 processor running at 667 MHz, on board 1 Gbps Ethernet PHY and USB-UART bridge, priced at 99 USD. The proposed system architecture is illustrated in Figure 2.

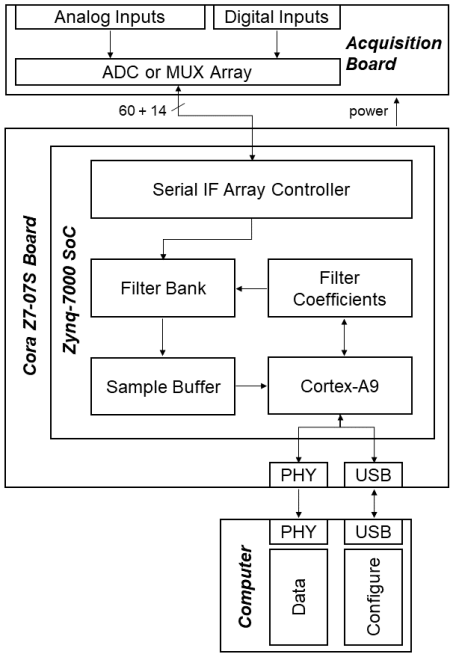

Figure 2. Architecture of the proposed platform, with FPGA-based board and dedicated data acquisition board.

The platform communicates with one or more computers using Ethernet and USB (Virtual COM Port). These interfaces are handled by a bare metal program running the Cortex-A9, providing a predictable timed execution. The Ethernet only transmits packets of acquired data to a client. Control and configuration messages are exchanged with the computer asynchronously via USB. Configuration is allowed only when the data acquisition is not running. This approach maximizes the packet rate and minimize the jitter. Control and configuration messages include enabling/disabling data acquisition and transmission, destination IP address and port, selection of acquired inputs to be transmitted, sampling rate, buffer size and filter coefficients. The ADC and/or multiplexer chips on the acquisition board are handled by an array of dedicated serial interfaces. The data pass through a bank of biquad filters, and finally gets packed into buffers of samples. An optional stage can convert the sampled 16-bit integer to 32-bit floating point within the unitary range, aligned with representation of most computer-based applications. The conversion to float will double the required Ethernet bandwidth, but it reduces the workload on the computer side. Serial interfaces, filters, and buffering are implemented in the FPGA fabric. On the Cora Z7-07S board, there are up to 74 FPGA pins exposed through stackable connectors, and 14 of these can be internally routed to the on-chip ADC. These are used in different configurations to interface in parallel the chips on the acquisition boards. This aspect is the bottleneck when using microcontrollers, as the number of serial interfaces cannot be expanded, and their control is usually sequential. Instead, the FPGA fabric allows implementing a large number of serial interfaces running in parallel.

4.2 Acquisition Boards

When simultaneous sampling of a large set of analog signals is not necessary, we can use the XADC of the Zynq-7000 FPGA, which is a dual channel 1 Mega Sample Per Second (MSPS) SAR ADC with 12-bit resolution. Generally, this is accurate enough to sample non-audio analog signals from sensors. The Cora z7-07S connectors allow connecting 6 single ended and 4 differential inputs to the on-chip multiplexer of the XADC. These 14 lines are connected to the acquisition board and configured to 10 single ended analog inputs with reference voltage of 3.3 V. Each single ended input is connected to a high-speed analog multiplexer such as the MAX4617 from Maxim Integrated, obtaining a total of 80 analog inputs combining internal and external multiplexers. The 8 external multiplexers are controlled by 4 lines from the FPGA. The XADC and the multiplexer are controlled in hardware. When a single channel of the XADC is used, we achieved almost 200 kHz sampling rate on 10 inputs. The performance scales down almost linearly when using the external multiplexers, obtaining up to 24 kHz on 80 analog inputs. If using both converted of the XADC, the resulting rate does not improve significantly, but pairs of signals can be acquired in parallel. The remaining 56 FPGA pins can be used as digital inputs. The total cost of the components for this board is approximately 20 USD.

The second acquisition board we propose is based on the AD7991 from Analog Device, which is a non-simultaneous 4-channel single-ended 12-bit SAR ADC with an I2C interface. The AD7991 is also featured on the Digilent Pmod AD2, that we used for our measurements. When using the maximum serial clock of 3.4 MHz, it is possible to acquire one sample per channel at a rate slightly above 24 kHz. With 24 AD7991 we can obtain a total of 96 analog input, where groups of 24 inputs are sampled simultaneously. This requires 24 I2C controllers implemented on the FPGA fabric and a total of 48 FPGA pins. This configuration allows using the remaining 26 FPGA pins as digital inputs. The total cost of the components for this board is approximately 150 USD. Alternatively, we can use 12 AD7699 sampling simultaneously only 12 inputs but increasing the resolution to 16-bit. The number of used FPGA pins used is unchanged while the cost of the components increases to 180 USD.

Finally, to sample all 96 analog inputs simultaneously, the acquisition board can be based on the LTC2320-16

ISBN 978-84-09-08518-7

from Linear Technology, which is a 1.5 MSPS single-ended 8 channels simultaneous sampling SAR ADC with 16-bit resolution. To acquire 96 analog inputs, we require 12 LTC2320-16 interfaced using 32 FPGA pins, and 8 fast SPI interfaces implemented in the FPGA fabric. These chips enable sampling all channels simultaneously at 24 kHz, but also at the audio rate of 48 kHz. This configuration allows using the remaining 42 FPGA pins as digital inputs. The total cost of the components to implement this high-end board is approximately 240 USD.

4.3 Current Prototype and Performances

With multiple serial interfaces communicating with the ADC chips, the bottleneck of the architecture in Figure 2 is the Ethernet link communicating with the computer. In preliminary tests, we measured the bandwidth of the Ethernet link driven by a bare metal program based on the Xilinx porting of the LightWeight IP library and running on the Cortex-A9. The benchmarks are representative of different scenarios varying the number of analog inputs and the size of the sample buffer. Settings and results are summarized in Table 3.

Analog Inputs	Buffer Size	Bytes Payload	Packet Rate kHz	Sampling rate kHz
18	1	36	84.5	84.5
	2	72	81.5	163.1
	4	144	72.4	289.8
	8	288	70.3	562.4
	16	576	61.1	977.5
	32	1152	49.4	1582.6
	64	2304	31.5	2017.5
34	1	68	81.1	81.1
	2	136	85.1	170.1
	4	272	63.6	254.4
	8	544	72.9	583.3
	16	1088	50.2	803.9
	32	2176	35.5	1138.1
	64	4352	15.3	985.5
96	1	192	68.7	68.7
	2	384	74.1	148.3
	4	768	60.1	240.5
	8	1536	34.6	277.1
	16	3072	27.1	434.9
	32	6144	12.9	413.0
	64	12288	6.4	410.4

Table 3. Benchmarks on Ethernet link bandwidth.

In the benchmarks of Table 3, independent of the ADC resolution, each sample is transmitted using 2 bytes. The payload of UDP packets does not include 4 bytes used to communicate with the client on the computer, and up to 5 additional bytes to transmit the data from digital inputs. When the payload is above 512 bytes, the data is split over multiple UDP packets. The results in Table 3 shows that the Ethernet PHY of Cora Z7-07S driven by the Cortex-A9 can easily accommodate the target sampling rates of 24 kHz or 48 kHz. Moreover, it is also possible to accommodate double payloads we obtain when performing the integer to float conversion in the FPGA, transmitting 4 bytes per sample. With no buffering, we observed that the inter arrival packet time of UDP packets on the computer was affected by a jitter of up to 4 sampling periods due to the computer OS controlling the Ethernet.

All components in the architecture of Figure 2 have been individually tested and were verified that they meet the required performances. Moreover, we estimate that the FPGA on the Cora Z7-07S board has sufficient resources to implement the filter bank and array of serial controllers. Currently, we developed two complete prototypes of the system. The first is representative of the acquisition board with multiplexers. The board has been fabricated but most functionalities are still implemented on the Cortex-A9, which can deliver a maximum sampling rate of 16 kHz on 80 analog inputs. In the second prototype, we have two AD7991 controlled via two I2C with a serial clock of only 1 MHz. Additionally, we also take the samples from the XADC for a total of 18 analog inputs. The prototype uses four Pmod AD2 modules, as visible in Figure 3. Most functionalities are integrated in the Cortex-A9 and we obtained a maximum sampling rate of approximately 4 kHz. Settings such as the buffer size and enabling/disabling the data acquisition are available through on-board buttons or the serial monitor. In both prototypes, the filter bank has not yet been integrated. On the computer side, we used Wireshark to monitor timeliness and correctness of the received UDP packets. The worst-case latency from the acquisition of the analog signal to the transmission of the UDP packet is equal to the sampling period multiplied by the buffer size.

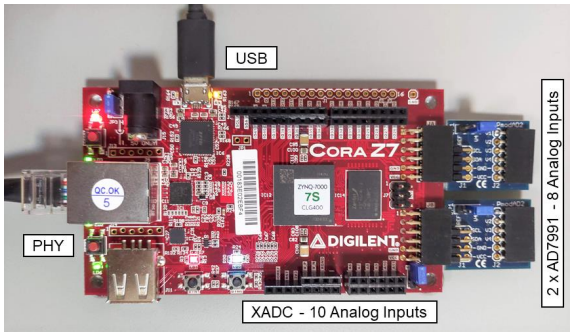

Figure 3. Platform prototype based on Digilent Cora Z7-07S and two Pmod AD2 modules.

5. FUTURE WORK

In this paper, we presented a platform to implement a complex interface for sonic interactive systems or musical instruments. The platform is capable of acquiring and transmitting a large number of analog signals from sensors to a computer, at rates significantly higher than microcontroller-based platforms. We designed the platform keeping in mind prototyping time and cost. Users can customize the functionality through simple commands without the need of developing any hardware or software components. The preliminary result of this ongoing pro-

ISBN 978-84-09-08518-7

ject shows that the it is possible to achieve the target performance with the proposed architecture and acquisition boards, outperforming all microcontroller-based boards. In the immediate future, we will complete the schematics and the fabrication of the second and third acquisition board. Thereafter we will integrate the filter bank and complete the software for the computer, such as clients to receive the data and configure the platform will be developed as well. Sending a subset of the acquired analog signals to separate IP addresses via UDP is a possibility we will further explore. Moreover, we will consider exploiting unused FPGA resources to drive a DAC or generate Pulse Width Modulated (PWM) signals to drive actuators, hence enabling duplex communication between the platform and the computer.

Acknowledgments

This work is supported by grant from the University of Wollongong in Dubai (No. 2018-007-SFMB). The authors would like to thank Adrian Freed for providing additional information on the SLABS.

6. REFERENCES

[1] K. Franinović and S. Serafin, *Sonic Interaction Design*. MIT Press, 2013.

[2] A. Hunt and M. M. Wanderley, "Mapping performer parameters to synthesis engines," *Organised Sound*, vol. 7, no. 2, pp. 97–108, 2003.

[3] D. Kushner, "The making of arduino," *IEEE Spectr.*, vol. 26, 2011.

[4] M. Banzi, *Getting Started with Arduino*, Ill. Sebastopol, CA: Make Books - Imprint of: O'Reilly Media, 2008.

[5] R. Barry, *FreeRTOS reference manual: API functions and configuration options*. Real Time Engineers Limited, 2009.

[6] T. Barrass, "Mozzi: Interactive Sound Synthesis on the Open Source Arduino Microprocessor," in *Proc. of ICMC 2013*, Perth, Australia, 2013.

[7] H.-C. Steiner, "Firmata: Towards Making Microcontrollers Act Like Extensions of the Computer.," in *Proc. of NIME 2009*, Pittsburgh, US, 2009, pp. 125–130.

[8] M. Mulshine and J. Snyder, "OOPS- An Audio Synthesis Library in C for Embedded (and Other) Applications," in *Proc. of NIME 2017*, Copenhagen, Denmark, 2017.

[9] "Comparison of single-board computers," *Wikipedia*. 27-Jan-2019.

[10] E. Berdahl and W. Ju, "Satellite CCRMA: A musical interaction and sound synthesis platform," in *Proc. of NIME 2011*, Oslo, Norway, 2011.

[11] G. Moro, A. Bin, R. H. Jack, C. Heinrichs, A. P. McPherson, and others, "Making high-performance embedded instruments with Bela and Pure Data," in *Proc. of Int. Conf. of Live Interfaces*, Brighton, UK, 2016.

[12] S. M. Trimberger, "Three Ages of FPGAs: A Retrospective on the First Thirty Years of FPGA Technology," *Proc. IEEE*, vol. 103, no. 3, pp. 318–331, 2015.

[13] Xilinx Inc., "7 Series FPGAs Overview." 2018.

[14] Xilinx Inc., "7 Series Product Selection Guide." 2018.

[15] D. Overholt, "The MATRIX: A Novel Controller for Musical Expression," in *Proc. of NIME 2001*, Singapore, Singapore, 2001, pp. 1–4.

[16] D. Wessel, "SLABS: Arrays of Pressure Sensitive Touch Pads | CNMAT," 2009. .

[17] K. Chuchacz, S. O'Modhrain, and R. Woods, "Physical Models and Musical Controllers: Designing a Novel Electronic Percussion Instrument," in *Proc. of NIME 2007*, New York, US, 2007, pp. 37–40.

[18] F. Pfeifle and R. Bader, "Performance Controller for Physical Modelling FPGA Sound Synthesis of Musical Instruments," in *Sound - Perception - Performance*, Springer, Heidelberg, 2013, pp. 331–350.

[19] R. F. Cádiz and A. Sylleros, "Arcontinuo: the instrument of change.," in *Proc. of NIME 2017*, Copenhagen, Denmark, 2017.

[20] D. Wessel and M. Wright, "Problems and prospects for intimate musical control of computers," in *Proc. of ACM Computer-Human Interaction Workshop on New Interfaces for Musical Expression*, Seattle, USA, 2001.

[21] A. Freed, R. Avizienis, and M. Wright, "Beyond 0-5V: expanding sensor integration architectures," in *Proc. of NIME 2006*, Paris, France, 2006, pp. 97–100.

[22] R. Avizienis, A. Freed, T. Suzuki, and D. Wessel, "Scalable Connectivity Processor for Computer Music Performance Systems.," in *ICMC*, 2000.

[23] S. Kartadinata, "The Gluion Advantages of an FPGA-based Sensor Interface," in *Proc. of the 2006 Conf. on NIME*, Paris, France, 2006, pp. 93–96.

[24] D. Wessel, R. Avizienis, A. Freed, and M. Wright, "A Force Sensitive Multi-Touch Array Supporting Multiple 2-D Musical Control Structures," in *Proc. of NIME 2007*, New York, US, 2007

[25] A. Freed, "Novel and Forgotten Current-steering Techniques for Resistive Multitouch, Duotouch, and Polytouch Position Sensing with Pressure," in *Proc. of NIME 2009*, Pittsburgh, US, 2009.

[26] A. P. McPherson, R. H. Jack, and G. Moro, "Action-Sound Latency: Are Our Tools Fast Enough?," in *Proc. of NIME 2016*, Brisbane Australia, 2016.

[27] A. McPherson and V. Zappi, "An environment for submillisecond-latency audio and sensor processing on BeagleBone Black," in *Audio Engineering Society Convention 138*, Warsav, Poland, 2015.

[28] A. Freed, "David Wessel's Slabs: a case study in Preventative Digital Musical Instrument Conservation," in *Proc. of SMC2016*, Hamburg, Germany, 2016.

[29] Digilent Inc., "Cora 7Z Reference Manual." 2018.