A Partial Reconfiguration based Microphone Array Network Emulator

Bruno da Silva*, Federico Dominguez§, An Braeken* and Abdellah Touhafi*
*Dept. of Industrial Sciences (INDI), Vrije Universiteit Brussel (VUB), Brussels, Belgium
§Escuela Superior Politecnica del Litoral (ESPOL), Guayaquil, Ecuador

Abstract—**Nowadays, microphone arrays are used in many applications for sound-source localization or acoustic enhancement. The current Micro-Electro-Mechanical Systems (MEMS) technology allows the development of networks of microphone arrays at a relatively low cost. Unfortunately, the evaluation of these networks requires controlled acoustic environments, such as anechoic chambers, to avoid possible distortions and acoustic artifacts. In this paper, we present a partial reconfigurable FPGA platform to emulate a network of microphone arrays. Our platform provides a controlled simulated acoustic environment, able to evaluate the impact of different network configurations such as the number of microphones per array, the network's topology or the used detection method. Data fusion techniques, combining the data collected by each node, are used in this platform. In addition, our platform is also capable to converge to the ideal network with regards to power consumption, while still maintaining the desired level of sound-source localization accuracy. A graphical user interface provides a friendly control of the network and the parameters under test during the execution of the partial reconfiguration operations. Several experiments are presented to demonstrate some of the capabilities of our platform.**

I. Introduction

Wireless sensor networks (WSN) composed of microphone arrays are becoming popular thanks to a relatively low cost of Micro-Electro-Mechanical Systems (MEMS) sensors. However, validation and verification of these networks are time consuming procedures. Furthermore, before the deployment of a WSN composed of microphone arrays, the network must be tested in controlled environments such as anechoic chambers to avoid undesired reflections, possible distortions or acoustic artifacts. Simulators are normally used first to test and estimate the capabilities of a network. However, simulation processes are computationally intensive tasks which usually require hours or days to complete. We present an FPGA-based microphone array network emulator to accelerate the simulation of such type of networks. Our proposed network emulator exploits the inherent parallelism that microphone arrays present by mimicking the node's response, combining the response of the network's nodes and providing an estimation of the network's response under a certain acoustic scenario. Therefore, instead of a pure software-based network emulator like the one presented in [1], the proposed network emulator uses an FPGA to accelerate the node's computation by implementing exactly the same HDL code [2], [3] that is going to be deployed in the nodes of a real network. Other authors such as in [4] or in [5] have already proposed the use of an FPGA to accelerate WSN emulators and wide-band wireless channel emulators respectively. Both works are complementary to ours as we focus on a detailed emulation of a network node while simplifying the wireless communication aspect.

The presented network emulator exploits the bandwidth presented on the PCI-express (PCIe) interface to not only communicate host and FPGA but also to partially reconfigure the FPGA-based network emulator to adapt the network topology and the node's configuration when needed. Partial reconfiguration (PR) through PCIe has been already used in [6] and [7], however our network emulator uses the Xilinx's technology [8] in a more complex application. As far as we are aware, the presented network emulator is one of the first applications using the recently introduced Xilinx MCAP [9] to partially reconfigure the FPGA through PCIe.

The main contributions of this work can be summarized as follows:

- An architecture of an FPGA-based network emulator.
- The use of partial reconfiguration through PCIe to reconfigure the network's characteristics.
- The comparison of different sound-source detection methods from a network perspective.

This paper is organized as follows. The description of the nodes and the detection methods are proposed in II. In Section III, the description of the network emulator architecture, the data fusion technique and the partial reconfiguration is done. In Section IV, the proposed network emulator is used to evaluate a few representative network configurations. Finally, our conclusions are presented in Section V.

II. Node Description

The nodes of the network are designed for far-field and non-diffuse sound fields like the one presented in [1]. Their architecture is an improved version of the one presented in [2], [3] and is composed of four stages: a Sensing stage, a Filter stage, a Beamforming stage and a Detection stage.

Sensing Stage: The audio data is acquired by a microphone array composed of four concentric subarrays of 4, 8, 16 and 24 digital ADMP521 MEMS microphones mounted on a 20-cm circular printed board. Each subarray is dynamically activated or deactivated in order to facilitate the capture of spatial acoustic information using a beamforming technique. This array modularity increases the efficiency of the sensing stage because the node can adapt its computational resources in accordance to the surrounding acoustic field.

Authorized licensed use limited to: ESCUELA POLITECNICA DEL LITORAL (ESPOL). Downloaded on January 12,2023 at 16:38:01 UTC from IEEE Xplore. Restrictions apply.

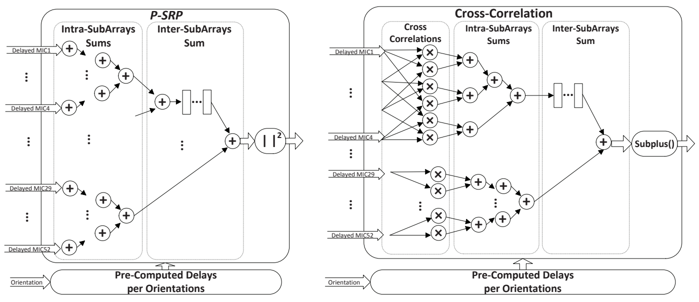


Fig. 1. *Detection methods considered for the Detection stage when using the two innermost sub-arrays.*

Filter Stage: The MEMS microphones are clocked at 2 MHz to oversample the audio signal to generate an output multiplexed pulse density modulation (PDM) signal. Each microphone signal has one cascade of filters to retrieve the audio signal. The first filter is a 16th decimator low-pass 4th order Cascaded Integrated-Comb (CIC) decimator filter, which only involve additions and subtractions to reduce the resource consumption. The CIC filter is followed by a running average block to reduce the microphone DC offset and to increase the dynamic range of the filtered signal. The last component of the filter stage is a 15th order serial low-pass FIR filter with a cut-off frequency of 12 kHz and a decimation factor of 4. The serial design of the FIR filter drastically reduces the resource consumption, but limits the maximum order of the filter, which must be lower than the decimation factor of the CIC filter.

Beamforming Stage: The filter stage is followed by the beamforming stage, which amplifies the sound coming from a pre-determined direction while suppressing the sound coming from other directions. Beamforming uses a spatial filtering technique to estimate the angle of arrival of sound signals (see [1], [2] or [3] for more details).

Detection Stage: The detection stage generates a *polar steering response map*, whose lobes are used for the localization of sound sources. Figure 1 details the implementation of the two different methods under evaluation: *Polar Steered Response Power* (*P-SRP*) and Cross Correlation (CC). On one hand, the *P-SRP* method [1] uses the added beamformed values to calculate the output power of the signal per orientation θ in the time domain ($P(\theta, t)$). On the other hand, a CC method based on the cross-correlated pairs of microphones offers a higher accuracy, specially for frequencies higher than 8 kHz, but demands a high number of multiplications because all possible pairs of microphones need to be correlated. The total number of multiplications (M_{am}) drastically increases when increasing the number of active microphones. For instance, $M_{am} = 6$ when only the inner subarray, composed of 4 microphones, is active, but drastically increases from 66 to 1326 when activating the first two inner subarrays (12 microphones) or all subarrays (52 microphones) respectively. As a result, CC is only considered when one or both of the inner subarrays are active. Independently of the detection method, the architecture needs 155.86 ms to generate a *polar steering response map* when using all the subarrays.

III. Network Emulator Description

The main purpose of the network emulator is to mimic the functionality of a network composed of different microphone array nodes and to evaluate the network's response for certain acoustic scenarios. This network increases the accuracy of the sound-source location by combining the response of each node. This information is used as an early estimation about how the network would react in real-world scenarios and allows a fast design space exploration in order to target priorities like overall power consumption or the accuracy of the sound-source localization. As a result, our network emulator is flexible enough to support multiple network topologies, different sound-source detection methods or a variable number of nodes and sound sources.

Figure 2 depicts the main components of the network emulator, which are distributed between the host and the FPGA.

Host: The host contains the sound source generator, the data fusion of the polar maps and the evaluation of the data fusion. A graphical user interface (GUI) abstracts the user from these computations and from the host-FPGA communication and the partial reconfiguration. The GUI consists of a front-end generated in Matlab that communicates with the FPGA back-end. The front-end is capable of simulating a sound field with multiple sound sources and nodes. Each sound source can have different frequency bands and each node can have different array configurations and calculation methods. Multiple sound sources are converted to PDM format in order to be compatible with the expected input data format of the nodes. The front-end uses data fusion to locate sound sources. Data fusion techniques combine the information gathered by different sensors measuring the same process to enhance the understanding of that process. In the context of this article, data fusion is performed by aggregating and combining the acoustic directivity information, represented as a *polar steering response map*, gathered by each node to produce a probability map of the location of the observed sound sources in a two-dimensional field. This technique is originally presented in [1] and has been used to validate the capacity of their microphone array design to locate sound sources (Figure 3). Based on the data fusion results, the front-end calculates three error parameters: the localization error (in meters), the number of undetected sound sources, and the number of phantom sound sources.

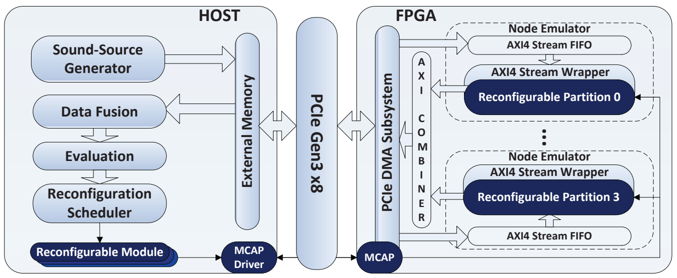
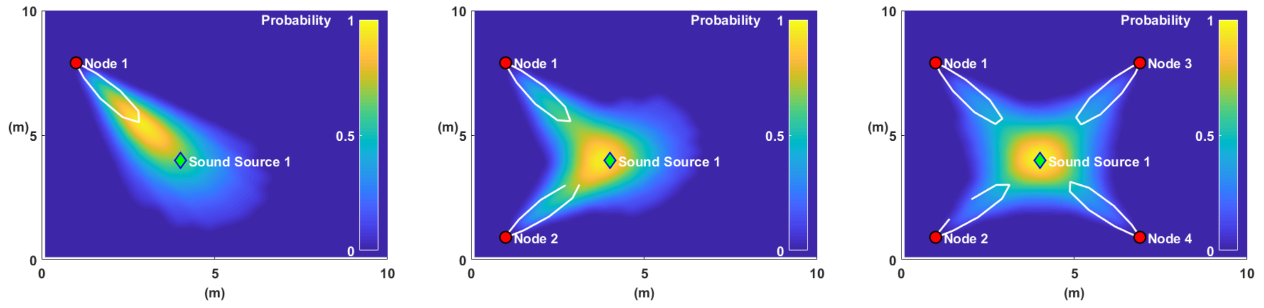


Fig. 2. *Distribution of the network emulator's components. The partial reconfiguration of the nodes and the data communication between the host and the FPGA are done via PCIe. The dark blue boxes represent the components involved in the PR.*

Authorized licensed use limited to: ESCUELA POLITECNICA DEL LITORAL (ESPOL). Downloaded on January 12,2023 at 16:38:01 UTC from IEEE Xplore. Restrictions apply.



(a) *One node* (b) *Two nodes* (c) *Four nodes*

Fig. 3. *Our data fusion technique combines the polar steering response map produced by each node to generate a probability map that estimates the location of the observed sound sources. As more nodes are used, the localization accuracy is improved. This technique has been adapted from [1].*

These values are used to evaluate the network configuration and to reconfigure the FPGA back-end if needed. The front-end allows us to create a network of nodes and to validate our architecture with a permutation of different scenarios: array architecture, detection method, sound spectrum, and sound-source positions.

PCIe Communication: The communication between the host and the FPGA uses the Xilinx PCIe DMA driver available in [11]. This driver allows the software running on the host to interact with the DMA endpoint IP via PCIe.

FPGA: On the FPGA side, the network emulator uses the IP core DMA subsystem for PCIe [12] with PR support. The DMA capability of this core is configured to act like an AXI4 bridge, operating at 125 MHz and with an AXI4-Stream (AXI4-S) interface of 256 bitwidth. Each reconfigurable partition (RP) is encapsulated in an AXI4-S wrapper that interfaces an input AXI4-S FIFO. Both blocks are integrated as a Node Emulator entity (Figure 2). The network emulator is composed of 4 Node Emulators operating at 62.5 MHz and with a 64-bits AXI4-S interface each. Finally, the output AXI4-S of the Node Emulators are combined in a 256-bits AXI4-S to interface the PCIe DMA Subsystem IP core.

Partial Reconfiguration over PCIe: Our PR uses the Media Configuration Access Port (MCAP) [8], which is a new configuration interface available for UltraScale devices that provides a dedicated connection to the ICAP from one specific PCIe block per device. This interface is integrated into the PCIe hard block and provides access to the FPGA configuration logic through the PCIe hard block when enabled. The bitstream loading across the PCIe to configure the RPs of the network emulator is detailed in [9]. The detailed process is only applicable for UltraScale architectures since these architectures need clearing bitstreams in order to prepare the RP for the new configuration. Consequently, each new RP reconfiguration requires a preceding clearing operation, otherwise the subsequent reconfigurable module (RM) cannot be initialized [8]. It demands a knowledge of what RM is configured at each RP. This task is done at the host side by the reconfigurable scheduler, which monitors the status of the nodes and applies the mandatory clearing reconfiguration before each PR of a node.

Resources	**RP Node 0**	**RP Node 1**	**RP Node 2**	**RP Node 3**
Slice Registers	117120 (40.93%)	117120 (40.93%)	120960 (39.64%)	129600 (36.99%)
Slice LUTs	58560 (62.56%)	58560 (62.57%)	60480 (60.57%)	64800 (56.53%)
LUT-FF Pairs	58560 (41.63%)	58560 (41.74%)	60480 (40.35%)	64800 (37.58%)
BRAM	180 (28.89%)	144 (36.11%)	180 (28.89%)	216 (24.07%)
DSPs	360 (30.00%)	576 (18.75%)	576 (18.75%)	648 (16.67%)
Bitmap Size	4,789 MB	4,925 MB	4,888 MB	6,934 MB
Clearing Time	0.134 s	0.104 s	0.089 s	0.119 s
Reconfig. Time	1.169 s	1.22 s	1.164 s	1.684 s

TABLE I

Dimensions and highest percentage of occupancy of the four reconfigurable modules.

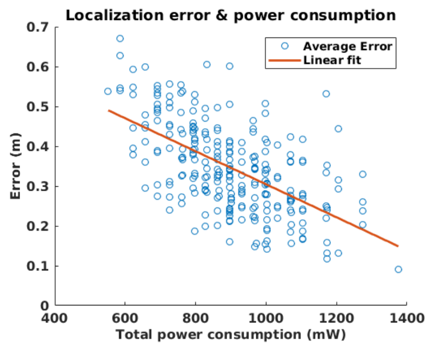


Fig. 4. *The average localization error is shown here for 4 nodes and 4 different architectures in each node, totaling 256 measurements. As the network uses more microphones and therefore consumes more power, the localization error improves. Here the correlation coefficient between error and power consumption is -0.58.*

Reconfigurable Partitions of the Network Emulator: Four RPs are available on the network emulator, each supporting a variable number of RMs. For instance, there are 2 different RMs based on the detection method. However, the RPs must be designed to support the node configuration with all subarrays active. The available resources per RP are shown in Table I. A slight variation occurs because not all RPs can contain the same number and type of slices. The reconfiguration time per RP have been experimentally obtained at the MCAP driver side. This timing ranges from 1.3 s to 1.8 s, including the cleaning operation and the partial reconfiguration.

IV. Results

A couple of experiments are detailed in this section to demonstrate some of the capabilities of the network emulator. The sound field simulation used in the front end has been optimized for a 10 m by 10 m two dimensional open field where sound attenuation, caused by propagation, has been

Authorized licensed use limited to: ESCUELA POLITECNICA DEL LITORAL (ESPOL). Downloaded on January 12,2023 at 16:38:01 UTC from IEEE Xplore. Restrictions apply.

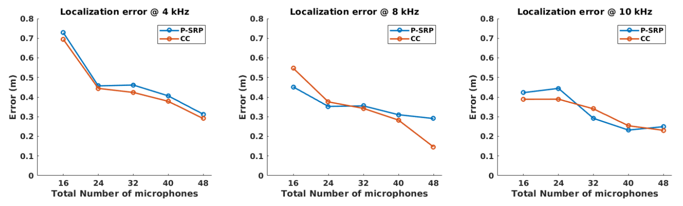


Fig. 5. *The localization error, using one sound source and four nodes at three different frequencies, improved as the total number of microphones in the network increased. Both methods, P-SRP and CC, performed equivalently.*

assumed to be negligible. The performance of several node's configurations and different detection methods are explored. For instance, we relate the localization performance of a network of four nodes to the overall power consumption (Figure 4). Most of the experiments demand a PR of one or more RMs. The presented results are averaged values obtained after a minimum of 15 random positions of a sound source for a certain network topology.

The system has been implemented in an Ubuntu 14.04.1 machine using Vivado 2016.4 and Matlab 2016b combined with C/C++ code and bash scripts. The FPGA card used for the implementation of the network emulator is a Xilinx Kintex Ultrascale from Alpha Data (ADM-PCIE-KU3) with a Gen3 PCIe connection.

A. Analysis of a Variable Number of Active Subarrays

The total power consumption of the network depends on the number of active microphones per node [3], which can be modified in runtime without PR by activating or deactivating subarrays. The capacity of the FPGA to modify the total number of active microphones per node allows a network of nodes to converge to an overall minimum power consumption for a given localization accuracy. To explore the relationship between power consumption and localization accuracy, we performed an experiment with a network of four nodes and one sound source of 4 kHz. The experiment varied the number of active subarrays in each node, corresponding with a total 256 different network configurations. The results, shown in Figure 4, demonstrate the expected negative correlation between power consumption and localization error.

B. Impact of the Detection Methods

The following experiment intends to demonstrate how PR allows to compare two different detection methods from the network point of view. On one hand, the *P-SRP* is the original detection method introduced and analysed in [1], [2] and [3]. On the other hand, the proposed CC is a different detection method which promises better accuracy when using a lower number of microphones. The theoretical implementation of CC when using only two inner subarrays needs 66 multiplications in order to reach all possible combinations of the 12 active microphones. However, our proposed implementation only considers the combinations between the microphones of a subarray and not the combinations between the microphones of different subarrays. Therefore, the number of multiplications is reduced to 32, with 6 and 28 multiplications for the innermost subarray and the second inner subarray respectively. Figure 5 shows the average error in the estimation of the sound source when applying data fusion of 4 nodes using the two inner subarrays. The values have been obtained for a random position of a standalone sound-source at 3 different frequencies (4, 8 and 10 kHz). The RMs are partially reconfigured to switch between both methods. The evaluation also considers the permutations of all possible combinations of the two inner subarrays. The results show that the CC method does not offer a significant improvement compared to the *P-SRP* method. Despite offering a lower estimation error, its implementation in a distributed network of microphone arrays is not completely justified if considering the additional resource consumption due to the required multiplications.

V. Conclusion

The presented network emulator has shown the capacity to evaluate different WSN configurations thanks to its ability to mimic the node's response to several sound sources. Although the partial reconfiguration through PCIe was not as fast as expected, this technique allows the network emulator to quickly explore multiple acoustic scenarios. A future automation of the network reconfiguration promises to determine, in real-time, the best strategies to obtain the lowest power consumption with the lowest estimation error.

References

[1] Tiete, J., et al., *SoundCompass: a distributed MEMS microphone array-based sensor for sound source localization*, Sensors 14.2 (2014): 1918-1949.

[2] da Silva, B., et al., *Runtime reconfigurable beamforming architecture for real-time sound-source localization.* 26th International Conference on Field Programmable Logic and Applications (FPL), EPFL, 2016.

[3] da Silva, B., et al. *Design Considerations when Accelerating an FPGA-Based Digital Microphone Array for Sound-Source Localization.* Journal of Sensors (2017): 2.

[4] Nasreddine, N., et al. *Wireless sensors networks emulator implemented on a FPGA*. International Conference on Field-Programmable Technology (FPT), (pp. 279-282). IEEE, 2010.

[5] Val, I., et al., *FPGA-based wideband channel emulator for evaluation of Wireless Sensor Networks in industrial environments*, In Emerging Technology and Factory Automation (ETFA), (pp. 1-7). IEEE, 2014.

[6] Vu, D. V.,et al., *Enabling partial reconfiguration for coprocessors in mixed criticality multicore systems using PCI Express Single-Root I/O Virtualization*, In ReConFigurable Computing and FPGAs (ReConFig), (pp. 1-6) . IEEE, 2014.

[7] Kizheppatt, V.et al., *DyRACT: A partial reconfiguration enabled accelerator and test platform*, 24th International Conference on Field Programmable Logic and Applications (FPL), IEEE, 2014.

[8] Xilinx, *Vivado Design Suite User Guide - Partial Reconfiguration*; Xilinx User Guide 909 (v2016.4), https://www.xilinx.com/support/documentation/sw_manuals/xilinx2016_4/ug909-vivado-partial-reconfiguration.pdf.

[9] Xilinx, *Bitstream Loading across the PCI Express Link in UltraScale Devices for Tandem PCIe and Partial Reconfiguration*; Xilinx Answer 64761, https://www.xilinx.com/support/answers/64761.html.

[10] AnalogDevices. *ADMP521 datasheet Ultralow Noise Microphone with Bottom Port and PDM Digital Output*, Technical Report, Analog Devices: Norwood, MA, USA, 2012.

[11] Xilinx, *Xilinx PCI Express DMA Drivers and Software Guide*; Xilinx Answer 65444, https://www.xilinx.com/support/answers/65444.html.

[12] Xilinx, *DMA Subsystem for PCI Express v2.0*; Xilinx Product Guide 195, https://www.xilinx.com/support/documentation/ip_documentation/xdma/v2_0/pg195-pcie-dma.pdf.

Authorized licensed use limited to: ESCUELA POLITECNICA DEL LITORAL (ESPOL). Downloaded on January 12,2023 at 16:38:01 UTC from IEEE Xplore. Restrictions apply.