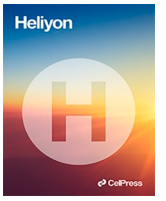
Heliyon 10 (2024) e25987

Contents lists available at ScienceDirect

Heliyon

journal homepage: www.cell.com/heliyon



Research article

FPGA architecture based on OpenCL for studying the acoustic backscattering by an immersed tube

Mhamed Hadji [a,*], Abdelkader Elhanaoui [a,c], Rachid Skouri [b], Said Agounad [c]

[a] REPTI, Faculty of Sciences and Technology, BP 509, Boutalamine, Errachida, Moulay Ismail University of Meknès, Morocco
[b] AMES, High School of Technology, Km 5, Road of Agouray N6, Moulay Ismail University of Meknès, Morocco
[c] LMTI, Faculty of Sciences, Ibn Zohr University Agadir, Morocco

ARTICLE INFO

Keywords:
Acoustic scattering (AS)
Signal processing system
OpenCL programming
Embedded architectures

ABSTRACT

At a time when the semiconductor industry is facing major difficulties in maintaining sluggish growth, new high-level synthesis tools are repositioning FPGAs as a leading technology for hardware-based algorithm acceleration, in the face of CPUs based clusters. As they stand, however, these tools do not guarantee that a software engineer can use these technologies to their full potential without expertise in the underlying hardware. This particularity can be an obstacle to their democratization. When it comes to acoustic scattering (AS) and its various applications, there is a growing need for autonomous and integrated systems that can operate in real time with high accuracy. This is why we propose our methodology for accelerating algorithms on FPGAs. After presenting a high-level architecture model of this target, we detail various possible optimizations in OpenCL, to finally define a relevant exploration strategy for algorithm acceleration on FPGAs. Applied to various case studies, to characterize and identify an immersed metal tube in the frequency range between 0 and 46.8 kHz. We evaluate our methodology according to three main performance criteria: execution time, resource utilization and energy efficiency. The experimental results show that the proposed methodology is efficient and effective. Indeed, the computation times using the DE1 Soc FPGA and a modern CPU are about 3.5s and 74s respectively. In addition, the absolute error did not exceed 10^{-5}.

1. Introduction

Over the past decades, acoustic scattering (AS) has been used in many different fields, such as nondestructive testing, monitoring, detection and characterization of objects and structures in marine and engineering environments [1–16]. Recently, other applications of AS have been developed, such as crack detection and characterization, sound identification, noise measurement, and noise attenuation techniques [17]. Many scientific researchers have performed different studies on AS using advanced signal processing approaches. The Wigner-Ville time-frequency technique and neural networks were used by Dariouchy and al [4] to predict the characteristic cutoff frequencies of an elastic tube. With the help of data-driven techniques such as Artificial Neural Network (ANN), the Adaptive-Network-based Fuzzy Inference System (ANFIS), and the Support Vector Machine (SVM), reduced cutoff frequencies of a considered tubing structure are predicted by Agounad and al [18]. However, the issues of computational complexity, data consumption, and processing time are not often considered [10,11] and Little research work involving the use of FPGAs has been carried

* Corresponding author.
E-mail address: hadjimhamed@gmail.com (M. Hadji).

https://doi.org/10.1016/j.heliyon.2024.e25987
Received 19 May 2023; Received in revised form 2 January 2024; Accepted 6 February 2024
Available online 11 February 2024

out [[7,8]]. This is why our search for new growth drivers to improve architecture performance, and software optimization of programs, has motivated our methodology for developing an FPGA-based embedded system for signal processing to accelerate algorithms on heterogeneous CPU/FPGA architectures.

In order to ensure real-time processing, to overcome this difficulty, one of the aims of our paper was to use the OpenCL language to implement relevant optimization concepts to accelerate the performance of algorithms dedicated to signal processing (AS).

In this sense, the approach adopted is based on low cost systems, and applying series of OpenCL optimizations. Its missions are to propose adequate solutions for the processing of the acoustic signal backscattered by a metallic target immersed in water, and to reduce relatively the computation time necessary for this task which is generally marked by its complexity. The algorithms that we have established for the calculations of an acoustic signal backscattered by a considered metallic tube have been implemented on the one hand with heterogeneous architectures (DE1 FPGA), and on the other hand with a laptop computer equipped with MATLAB software. The form function and the spectrum of resonances, characteristic of the studied tube, were also obtained. Excellent results are expected compared to the Matlab implementation. Lower latency and high connectivity were achieved. The time and power consumption constraints were then largely satisfied.

The rest of the article is organized as follows: section 2 summarizes the mathematical equations for acoustic scattering. Section 3 describes the design of the FPGA-based architecture and shows the detail of the different blocks, the methodology for using FPGA as coprocessor in section 4, the results are provided and discussed in section 5 and Conclusion is presented in section 6.

2. Brief recall of acoustic scattering from a tube

An air-filled metallic tube immersed in water, is considered. The scattering of an ultrasound plane wave from the target, may be studied based on the solution of the wave equation, and the boundary conditions [1]. Fig. 1 shows the polar coordinate (r, θ) orientation and the direction of the incident wave which characterizes by the incident wave numberk_1 ($k_1 = \omega/c_W$); ω is the angular frequency and c_W is the acoustic velocity in the external fluid (water).

The scattering pressure in far field is given by the summation of the normal modes. This summation takes into account different contributions of the incident wave, the reflected echo, and the circumferential waves. The general form of the scattered pressure field at normal incidence is formulated as [4]:

$$P_{scatt}(r,\theta) = P_0 \sum_n \left[R_n(\omega)\varepsilon_n \, H_n^{(1)}\,(k_1 r)\right].cos(n\theta) \tag{1}$$

where ε_n is the Neumann factor (if $n = 0$, $\varepsilon_n = 1$ else $\varepsilon_n = 0$), and P_0 is the amplitude of the plane incident wave. The scattered coefficients $R_n(\omega)$ are computed from the boundary conditions at interfaces r = a, and r = b, $R_n(\omega)$ is a coefficient that characterizes the AS in surrounding fluid medium. The function $H_n^{(1)}(k_1 r)$ is the Hankel function of the first kind [1,9]. The module of Hankel function in a far-field can be expressed as:

$$\left| H_n^{(1)}\,(k_1 r\,)\right| \approx (2\,/\,\pi\, k_1\, r)^{1/2}\, e^{\left(k_1 a - \frac{n\pi}{2} - \frac{\pi}{4}\right)} \tag{2}$$

The module of the backscattered pressure in a far-field, denoted by F∞, is then obtained from equ2 and equ.3 by the following equation:

$$F_\infty(\nu\,) \approx (4\,/\pi k_1\, a)^{1/2}.\left|\sum_n\,(-1)^n\, R_n\, \varepsilon_n \cos(n\theta)\right| \tag{3}$$

3. Proposed computing environment

3.1. Field-Programmable Gate Arrays

The FPGAs (Field-Programmable Gate Arrays) are parallel electronic circuits that allow us to develop applications that are

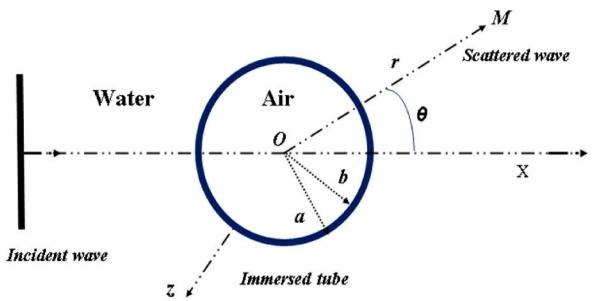


Fig. 1. Acoustic scattering by an immersed tube.

increasingly powerful in terms of execution speed and greedy in terms of hardware resources. In our work, we implement on FPGAs, the program that calculate backscattering spectrum response from cylindrical shell and some signal processing algorithms that allows us to perform temporal and spectral analyses. We study the feasibility of processing in a minimum of time, thus lifting the complexity of calculations, due to the presence of Bessel functions in the resolution of the ultrasonic acoustic diffusion problem. Fig. 2 shows the Intel-DE1 Altera's board.

FPGAs are in an intermediate position between ASICs and CPUs, given their ability to reconfigure their architecture according to the application and their good energy efficiency. In the last decade, multiple efforts have been made to reduce the energy consumption of large computing systems and FPGAs are consolidating as aviable alternative to achieve this goal. That is why several companies and organizations have incorporated this kind of hardware devices to their systems, like Microsoft [10], Baidu [11], CERN [12] or Amazon [13].

The purpose of the configurability of FPGA is to combine various data monitoring and analysis processes. The host part is used in this work to interface the initial data. VHDL and Verilog are two languages of Hardware Description Language (HDL), which is used to create FPGA-based architecture.

The Cyclone V that contains DE1 board is a single system-on-a-chip (SoC) with two different components: a hard system processor (HPS) and an FPGA. First portion encloses a microprocessor unit (MPU) subsystem with one or two ARM Cortex-A9 MP Core processors, flash memory controllers, SDRAM L3 Interconnect, on-chip memories, support peripherals, interface peripherals, debugging capabilities, and phase-locked loops are all found in the HPS (PLL). Asymmetric and symmetric multiprocessing (AMP and SMP) are supported by the dual-processor HPS.

The second contains FPGA structure, control block (CB), phase-locked loops (PLL), and, depending on the device model, high-speed serial interface (HSSI) transceivers, PCI Express (PCIe) hard controllers, and hard memory controllers are all found in the FPGA component. The device's HPS and FPGA components differ significantly. The HPS can be booted from an external memory card. However the FPGA must be configured using the HPS or an external device like the Quartus Prime programmer. Each of the device's HPS and FPGA components has its own pins. Pins between the HPS and the FPGA are not freely shared. An FPGA setup image via the HPS or any other external source supported by the device configures the FPGA I/O pins.

3.2. Heterogeneous system

The high demand for computer performance in science and engineering has led to the usage of heterogeneous computing, with FPGAs, GPUs and other accelerators acting as co-processors for arithmetic-intensive parallel applications. The tendency to massively parallel designs and heterogeneous computing has prompted a critical necessity for software development infrastructure in the form of parallel programming languages and standards libraries supporting heterogeneous computing on soc card. Many existing scientific and technical applications have been adapted to efficiently utilize multi-core processors and massively parallel GPUs, and FPGAs. Generally for the heterogeneous architecture, we find CPU-GPU, CPU-FPGA, CPU-DS, etc. In this work, we will talk about only the CPU-FPGAs that consist of hard processor system (HPS) portion and an FPGA evolved from hardware accelerators to very powerful System-on-Chip platforms, mainly thanks to the availability of increasingly powerful embedded processors. The efficient integration of embedded processors with the FPGA fabric with fast exchange of information between both sides, lead to the performance improvement of these architectures [15–19].

3.3. Opencl programming

OpenCL (Open Computing Language) is an open free standard programming in parallel generally across microprocessors, GPUs, FPGA and other, providing software designer's portable and practical access to the performance of these systems treatment platforms.

It supports a wide range of different applications, from embedded and consumer software to high performance computing (HPC)

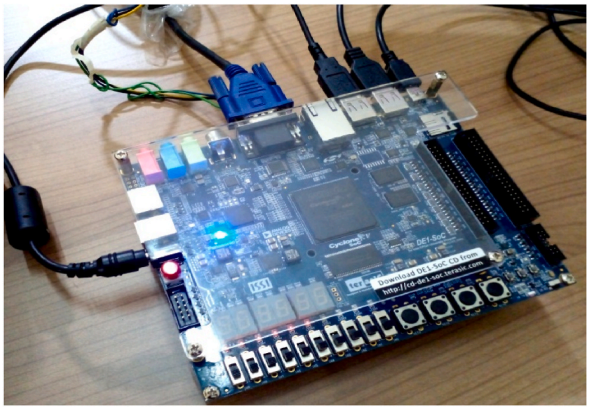


Fig. 2. DE1 FPGA board.

that enables high-speed data processing and complex computations, technical solutions using a low-level, high-performance, portable abstraction.

OpenCL is particularly suited to take on a more prominent role in new interactive interface applications that merge general purpose parallel computing algorithms with graphics rendering pipelines. While OpenMP (Open Multi-Processing) is an API (Application Programming Interface) for parallel programming on shared memory systems, OpenMP is a scalable and portable programming language [20]. It allows developers to quickly create parallel applications with fine granularity while remaining close to sequential code. For many reasons explained and justified in the following sections, the OpenCL heterogeneous programming language was considered in this study. The implementation of digital systems allowing the simultaneous processing of data is a matter of parallelism. Its objective is to execute as many processing operations as possible in the shortest possible time. The competition between the exchange of instructions and data is the basis of the idea of parallelism. Hereafter are some techniques that focus on instruction and data flows:

- SIMD: Single Instruction Multiple DATA programming model in which a kernel is run on numerous processing elements at the same time, each with its own processing elements, data, and a shared program counter. Program counter shares all processing elements following a set of completely identical instructions.
- SPMD: Single program, multiple data. A programming approach in which a kernel is executed on many processing elements at the same time, each with its own data and program counter. Therefore, even though all the computing resources are running the same kernel, they retain their separate instruction counters. Because of branching in a kernel, the instruction counter and program counter can be very different in each kernel.
- TPPM: Task-parallel programming model. A programming model in which computations are described as multiple concurrent tasks, each task consisting of a kernel running in a single workgroup of size one.

We opted for SIMD to increase the data processing efficiency of an OpenCL kernel by executing multiple work items in a SIMD (single instruction multiple data) manner without manually vectorizing our kernel code.

4. Methodology for using FPGA as coprocessor

4.1. Modeling an FPGA

The ability to be reprogrammed is one of the main advantages of an FPGA, However, this feature makes it challenging to create an accurate performance statistic.

Based on the architecture's intrinsic properties, then we will describe the modeling process of an FPGA with the goal to better comprehend the performance of FPGA as a coprocessor, and so describe the novel setting that we have mentioned to enhance its performance.

To model FPGA operation in the co-processing dynamics enabled by the OpenCL standard, we'll use Fig. 3, which outlines the steps involved in running an algorithm on this type of architecture.

In this model, the host platform is a CPU, and our complementary platform is an FPGA.

In a typical configuration, the data is on the host, or at least shared with it. At some point, a function needs to be executed on the FPGA, at which point the following tasks take place:

Step 1. Copy input data from the host to the FPGA's global memory.

Step 2. Some data may then be cached.

Step 3. The computational pipeline then performs the requested processing within the elementary pipeline.

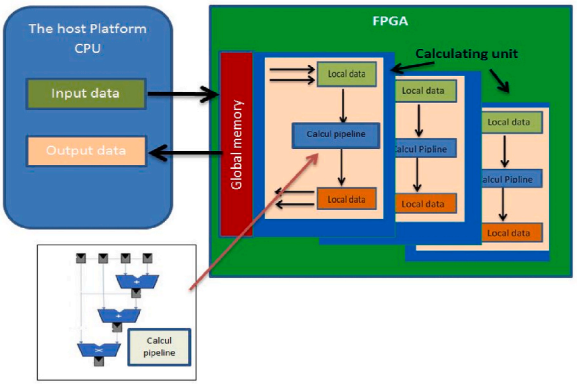


Fig. 3. Architecture model.

Step 4. The processing loop ends with the reverse copy of output data from local memory to FPGA global memory.

Step 5. Copy back to host memory To optimize the implementation of a program on a CPU-FPGA coprocessing platform.

4.2. OpenCL optimizations

This section is based in part on the various concepts and optimizations presented by and integrated into their tools.

In this section, we present a manual for optimizing FPGAs with OpenCL. We have integrated the advanced optimization concepts we have been able to define, as well as the optimizations proposed by manufacturers that we found most interesting, and for which we have defined the conditions under which their use is or is not advantageous.

we analyzed and optimized a number of points, such as:

- data copying between host and FPGA.
- memory storage strategy (types, banks, sizing and partitioning, etc.)
- data caching within the FPGA factory and management of access policy.
- replication of Elementary Pipelines, notions of pipeline, loop unwinding and vectorization.
- computational pipeline efficiency.

As Fig. 3 has been simplified so as not to detract from its readability, there are other mechanisms, such as communication between functions implemented on FPGAs, or the management of different types of global memory.

This section therefore focuses on the simplified modeling of an algorithm implemented on FPGA with OpenCL tools, and recalls certain notions commonly used in the field, to which we have added our own custom metrics that allow us to quickly characterize the relevance of optimizations according to the particularities of the application and platform studied, in order to measure the efficiency of an implementation.

We begin by looking at the computational pipeline, and then at data caching in the elementary pipeline. This is followed by a discussion of possible replication of the elementary pipeline, then the use of large memory structures on FPGAs (global and constant memory), and finally the CPU ↔ FPGA copy.

4.3. Ongoing methodology

Our methodology is based on a multi-criteria optimization framework. In fact, we place ourselves in the situation where we have constraints on raw performance (execution time) as well as on the resources consumed. Depending on the specifications of a given application, it is then possible to define a domain of acceptable solutions, and the aim of our methodology is to quantify the various optimization levers which, depending on the constraints, enable us to get as close as possible to this domain.

The first criterion is therefore raw performance, while the second may be more or less important depending on the application. In order to remain consistent with the model, the notion of resources used corresponds to the aggregation, in the worst-case scenario, of the various elementary blocks accessible to the FPGA user.

Depending on the purpose of an algorithm, the notion of "good" optimization may vary. Some applications have a strong resource constraint, as part of the FPGA must, for example, be able to be used for other processing, whereas in other use cases, execution time takes priority, and there is no particular resource limitation. Fig. 4 shows the domain of acceptable solutions as a function of two arbitrary resource and time constraints, and the marks within this zone represent the set of acceptable solutions.

Algorithm optimization methodology is therefore highly dependent on the existence of these constraints. Indeed, if there is only a

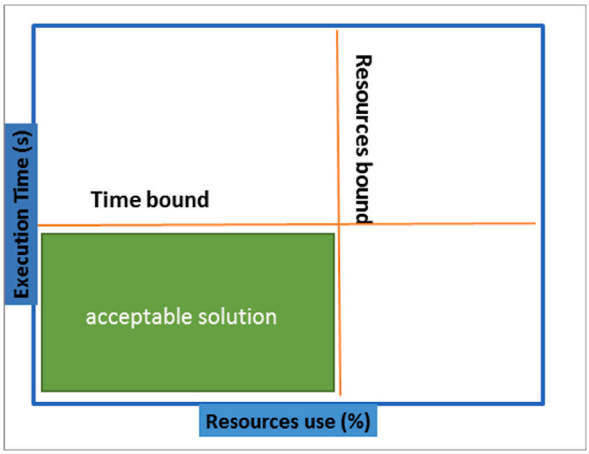


Fig. 4. Acceptable solution.

resource constraint, then the solutions to be favored are those close to the corresponding boundary, but on the left-hand side of the graph, whereas if there is a time constraint, the lowest part, i.e. on the right-hand side of our Fig. 4, will be favored.

In the case of two constraints, the set of acceptable solutions is restricted to a rectangle, which is also found in the lower left-hand part of the graph.

It is therefore important to identify which constraints have priority before embarking on an optimization approach, as the notion of "good" optimization depends on them.

Step 1. Architecture modeling.

Step 2. Identification of functionalities to be adapted.

Step 3. Implementation of a first functional version.

Step 4. Rapid modeling using HLS metrics and tools.

Step 5. Selection and implementation of relevant optimizations.

Step 6. Performance-based looping.

Step 7. Hardware testing and performance measurement.

5. Results and discussion

5.1. Acoustic backscatter signatures

Let's take a look at three types of single-layer thinner cylindrical metallic tubes: the first have a polymer inner lining, while the fourth, taken as a reference, has no inner lining. The characteristics of the shells studied are specified below. Its outer and inner radius are respectively $a = 1\ m$ and $b = 0,\ 97\ m$. Physical parameters of the chosen material are:

- Stainless steel: the density $\rho = 7900$ kg m^{-3}, the longitudinal wave velocity $c_L = 5790$ m s^{-1} and the transverse wave velocity $c_T = 3100$ m s^{-1}.
- Polymer: the density ρ = 1200 kg.m-3, the longitudinal wave velocity cL = 2000 m.s-1 and the transverse wave velocity cT = 1000 m.s-1.
- Alimunium: the density ρ = 1500 kg.m-3, the longitudinal wave velocity cL = 2500 m.s-1 and the transverse wave velocity cT = 1200 m.s-1.
- The external fluid (water) is characterized by its density $\rho_1 = 1000$ kg m^{-3} and speed of sound $c_1 = 1470$ m s^{-1}. The cavity of the tube is failed with air.

After studying the three cases and obtaining the same results, we will confine ourselves to illustrate the first one. Fig. 5(a and b) shows the acoustic backscattering response of the first studied structure, using DE1 FPGA board, and a processor architecture.

The backscatter spectrum is obtained (Fig. 5(a and b)), in which transitions appear, linked to resonances obtained from circumferential waves when an integer number of wavelengths is established in the tube circumference. This spectrum is not very easy to exploit in practice, so it is possible to obtain another spectrum called the resonance spectrum by removing the specular echo from the time signal (Fig. 6(a and b)).

The time signal obtained after diffusion by the tube is shown in Fig. 6(a and b). On the left of this Fig. 6(a and b), the high-amplitude specular echo is linked to the reflection on the tube, while the other echoes are linked to the progressive re-emission of circumferential waves in the shell as they propagate. Between each echo, the wave has circled the circumference of the tube. This various amplitudes in Fig. 5(a and b) is linked to the resonances of circumferential waves. The processing by the inverse Fourier transform (IFFT) provides the total time signal. A series of echoes thus appears in Fig. 6(a and b).

Calculated time pulse response contains two parts. First is the non-resonant part which represents the geometrical reflection. This part is localized at the beginning of the pulse response. The second is the resonant part which is associated to the resonance of

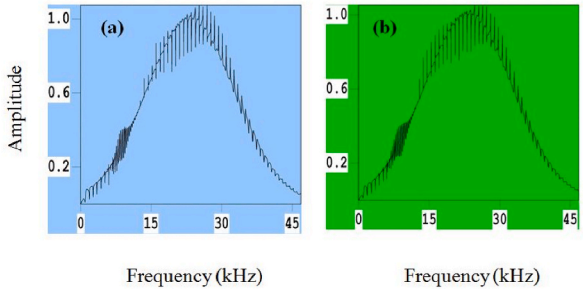


Fig. 5. Acoustic backscattering spectrum response of Stainless steel tube: (a) Using DE1 FPGA board; (b) Using a processor architecture.

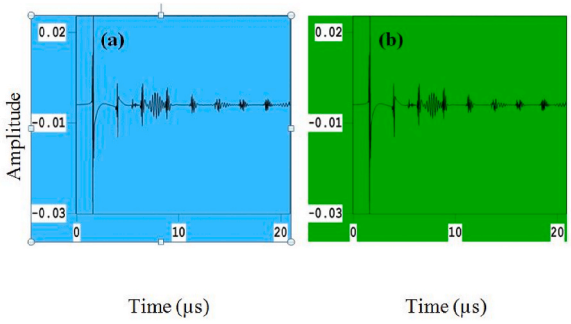


Fig. 6. Temporal signal of Stainless steel tube: (a) Using DE1 FPGA board; (b) Using a processor architecture.

circumferential waves.

When the non-resonant part is replaced by the zeros and FFT is applied to the filtered signal, the resonance spectrum is provided and showed in Fig. 7(a and b).

In this Fig. 7(a and b) this time, resonances appear as well-isolated peaks that can be identified both theoretically and experimentally.

This Fig. 7(a and b) shows resonances of circumferential waves which are manifested in the form of amplitude peaks.

Fig. 8(a and b) presents the comparison between the filtered time signals obtained using the DE1 FPGA board and Matlab simulation with a processor architecture.

An excellent accord has been noted. Indeed, Fig. 8 (a) shows that the results of the calculations obtained by using both approaches are well correlated. In addition, Fig. 8 (b) reflects an absolute error that remained below 10^{-5}.

5.2. Hardware resources

Being marked by a certain algorithmic complexity, such an acoustic scattering problem requires important hardware resources and many logical elements. However, this high hardware demand has been reduced by about 80% by our approach. Table 1 shows the Summary of the percentage of material resources implemented in the DE1 FPGA board.

5.3. Speed of execution

As far as execution time is concerned, our system presents a faster and more efficient solution to potentially complex acoustic signature calculation tasks. Indeed, Fig. 9(a and b) shows that with dedicated hardware to the DE1 board, an execution time limited to only 3.5 s is obtained against 75 s during processor operation for a Matlab code implementation.

5.4. Discussion

All these acoustic backscatter signatures by a thin stainless steel tube immersed in water show a perfect agreement between DE1 card and Matlab simulation tool. The De1-soc gives excellent results in terms of processing time and is a good solution for the study acoustic signals due to its size and portability. Using OpenCL, the program was optimized to reduce the processing time of the acoustic signal backscattered by a thin stainless steel tube to about 3,5s.

5.5. Limits

The model's basic purpose is to provide a rapid, synthetic representation of the performance limits of an application on a platform.

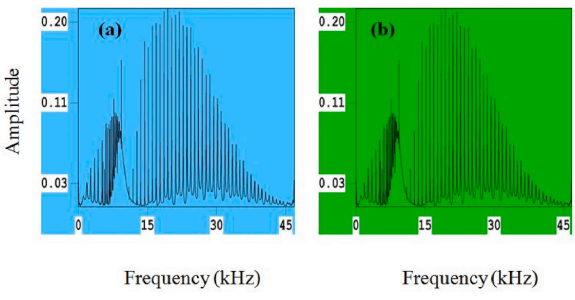


Fig. 7. Resonance spectra of Stainless steel tube: (a) Using DE1 FPGA board; (b) Using a processor architecture.

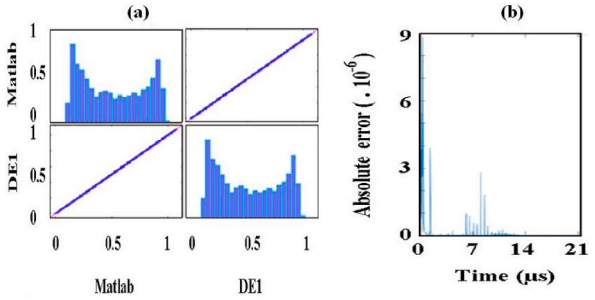


Fig. 8. Comparison between the filtered time signals obtained
using the DE1 FPGA board and Matlab simulation with processor architecture: (a) Correlation matrix; (b) Absolute error.

Table 1
Summary of the percentage of material resources implemented in DE1 Board.

Resource	Usage
Logic utilization	83%
ALUTs	51%
Dedicated logic registers	37%
Memory blocks	69%
DSP blocks	51%

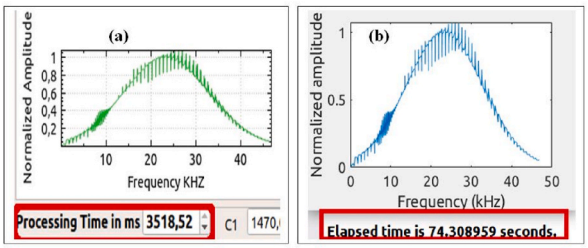


Fig. 9. Elapsed time for computing acoustic signal scattered from Stainless steel tube: (a) Using DE1 FPGA board; (b) Using a processor architecture.

The first step is to identify the elements that can be used to find the upper limits of performance.

Reprogrammable elements are mainly dedicated to calculations, except in specific cases. Thus, it is possible to precisely calculate the bandwidth between the host and the FPGA, or between the computing cores and the global memory of a given FPGA.

On the other hand:

- The main limitation: for blocks dedicated to implementing computations (LUTs and DSPs) on a given chip, the challenge is to be able to characterize their theoretical maximum performance.
- A second important limitation is that, on the one hand, the efficient optimization of an algorithm on an FPGA requires the use of the local and private memory available on the chip itself, and that, on the other hand, it is impossible to measure the maximum bandwidth of this type of memory, as it depends on the implementation given for each algorithm.

6. Conclusion

In this paper, OpenCL-based DE1 FPGA architecture for computing in the frequency range between 0 and 46.8 kHz, of the far-field backscattered pressure modulus, the time-domain acoustic signal, and the resonance spectrum of an immersed stainless steel tube in water has been presented. OpenCL programming is used in the implementation to provide parallel programming on a multiprocessor system with shared memory and an FPGA. Parallelism is achieved by creating a set of computational units. These units independently and simultaneously execute designated tasks. By simulating the analytical method using a laptop computer with Matlab software, the agreement and efficiency of the proposed embedded system has been demonstrated. The speed of our approach compared to Matlab simulation using processor architecture, with only the involvement of 80% of logic gates has been noted. Indeed, the acoustic signal processing time using the DE1 architecture is estimated at 5% of the average time needed for Matlab implementation. Moreover, an absolute error of about 10^{-5} is obtained.

In the end, an efficient implementation using the DE1 FPGA board has been achieved. An interesting step to realize a real time

acoustic signal processing has been taken due to the accuracy of the obtained results.

CRediT authorship contribution statement

Mhamed Hadji: Conceptualization, Methodology, Software, Writing – original draft. **Abdelkader Elhanaoui:** Data curation, Investigation, Project administration. **Rachid Skouri:** Supervision, Validation. **Said Agounad:** Visualization, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] S. Agounad, E.H. Aassif, Y. Khandouch, G. Maze, D. Décultot, Investigation into the bistatic evolution of the AS from a cylindrical shell using time-frequency analysis, J. Sound Vib. 412 (2018) 148–165.
[2] S. Agounad, E.H. Aassif, Y. Khandouch, D. Décultot, G. Maze, A. Elhanaoui, Acoustic scattering from immersed composite cylindrical shells: existence of zero group velocity circumferential waves, Compos. Struct. 182 (2017) 12–24.
[3] J. Li, H. Hua, Transient vibrations of laminated composite cylindrical shells exposed to underwater shock waves, Eng. Struct. 31 (2009) 738–748.
[4] A. Dariouchy, E.H. Aassif, D. Decultot, G. Maze, Acoustic characterization and prediction of the cut-off dimensionless frequency of an elastic tube by neural networks, IEEE Trans. Ultrason. Ferroelectrics Freq. Control 54 (2007) 1055–1064.
[5] L. Haumesser, A. Baillard, D. Décultot, G. Maze, Behavior of first guided wave on finite cylindrical shells of various lengths: experimental investigation, J. Acoust. Soc. Am. 109 (2001) 583–590.
[6] A. Elhanaoui, E. Aassif, G. Maze, D. Décultot, Acoustic scattering by a two-layer cylindrical tube immersed in a fluid medium: existence of a pseudo wave, Ultrasonics 65 (2016) 131–136.
[7] R. Nane, V.M. Sima, C. Pilato, J. Choi, B. Koen, A survey and evaluation of FPGA high-level synthesis tools, IEEE Trans. Comput. Aided Des. Integrated Circ. Syst. 35 (2016) 1591–1604.
[8] M. Turqueti, J. Saniie, E. Oruklu, MEMS acoustic array embedded in an FPGA based data acquisition and signal processing system, in: Proceedings of the 2010 53rd IEEE International Midwest Symposium on IEEE Circuits and Systems (MWSCAS), Seattle, WA, USA, 1–4 August 2010, pp. 1161–1164.
[9] P. Hernandez, Microsoft uses Intel FPGAs for smarter bing searches [Online]. Available: https://www.eweek.com/cloud/microsoft-uses-intel-fpgas-for-smarter-bing-searches, 2018.
[10] X. Inc., Baidu Deploys Xilinx FPGAs in New Public Cloud Acceleration Services, SAN JOSE, Calif., July 5, 2017. PRNewswire/ – **Xilinx, Inc. (XLNX)**, https://www.prnewswire.com/news/xilinx%2C-inc./.
[11] L. Barney, CERN openlab explores new CPU/FPGA processing solutions [Online]. Available: https://www.hpcwire.com/2017/04/14/xeon-fpga-processor-tested-at-cern/, Apr. 2017.
[12] K. Freund, Amazon and xilinx deliver new FPGA solutions [Online].Available: https://bit.ly/3ofmBsI, 2017.
[13] L. Cohen, Time-frequency distribution— a review, Proc. IEEE 77 (7) (1989) 941–981.
[14] F. Hlawatsch, G. Boudreaux-Bartels, Linear and quadratic time-frequency signal representations, IEEE Signal Process. Mag. 9 (2) (1992) 21–67.
[15] R.F. Molanes, F. Salgado, J. Fariña, J.J. Rodríguez-Andina, Characterization of FPGA-master ARM communication delays in Cyclone V devices, IECON 2015 - 41st Annual Conference of the IEEE Industrial Electronics Society (2015) 4229–4234, https://doi.org/10.1109/IECON.2015.7392759.
[16] S. Agounad, E.H. Aassif, Y. Khandouch, A. Elhanaoui, Signal processing techniques of circumferential waves for characterization of bilaminated cylindrical shells, J. Nondestr. Eval. 39 (1) (2020) 18.
[17] A.C. Naef, S.E.J. Knobel, N. Ruettgers, M.-M. Jeitziner, Mg Holtforth, B. Zante, J.C. Schefold, T. Nef, S.M. Gerber, Methods for measuring and identifying sounds in the intensive care unit, Front. Med. 9 (2022) 836203, https://doi.org/10.3389/fmed.2022.836203.
[18] Said Agounad, El Houcein Aassif, Younes Khandouch, Abdelkader Elhanaoui, Analysis of the prediction of a bilayered cylindrical shell's reduced cutoff frequency with data-driven approaches : https://doi.org/10.1016/j.ymssp.2019.03.028Get.
[19] Book Embedded Systems Using Quartus and Build Root for Building Embedded Linux Systems (De1-SOC) V1.9 Mariano Ruiz Antonio Carpeño.
[20] M. Hadji, Abdelkader Elhanaoui, R. Skouri, S. Agounad, Embedded system of signal processing on FPGA: implementation OpenMP architecture, in: S. Motahhir, B. Bossoufi (Eds.), Digital Technologies and Applications. ICDTA 2023, Lecture Notes in Networks and Systems, vol. 668, Springer, Cham, 2023, https://doi.org/10.1007/978-3-031-29857-8_78.