Strategies for Hardware Reduction on the Design of Portable Ultrasound Imaging Systems

D. Romero-Laorden, J. Villazón-Terrazas,
O. Martínez-Graullera and A. Ibáñez

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/55910

1. Introduction

In the last decade, ultrasonic imaging systems have been an essential tool for diagnosis in medical and industrial applications, especially in the Non Destructive Testing area (NDT). Conventional ultrasonic imaging devices produce high quality images with good resolution and contrast. However, these machines are usually associated to a high cost in hardware resources, as well as in the time required for the data acquisition and processing stages. This fact hinders the development of good quality, compact and low-power systems that can operate in a wide range of real-time applications.

In this sense, the Synthetic Aperture techniques (SAFT) have demonstrated to be an effective method to achieve these goals, minimizing the size of the systems and accelerating the image acquisition processes. Consequently, both power consumption and overall cost of the systems can be reduced making possible their miniaturization and portability. Conventional SAFT techniques are based on the sequential activation in emission and reception of every transducer element. Once all acoustic signals have been stored in memory, a beamforming process is applied in a post-processing stage in order to focus the image dynamically in emission and reception, obtaining the maximum quality at each image pixel. Despite of this, conventional SAFT techniques present some inconveniences which are summarized in the following points:

1. **Artifacts**. Conventional SAFT techniques produce grating lobes in the images due to the acquisition processes.
2. **Low contrast**. As a consequence of firing only one element at time the received signals have low signal-to-noise ratio, which results in low contrast images that are not feasible for regular imaging visualization (e.g. echography imaging needs very good images in order to reduce the fails in the diagnostic).

INTECH
open science | open minds

3. **Medium penetration**. And for the same reason, the penetration deep of ultrasound in the region of interest is smaller than the achieved using conventional imaging techniques (e.g. needed by cardiac imaging or industrial inspections).

In order to reduce some of these drawbacks, more sophisticated SAFT techniques have been proposed. Total Focusing Method (TFM) [1] is one of them, where each array element is sequentially used as a single emitter and all array elements are used as receivers. Thus, it is possible to obtain a set of $N \times N$ signals (Full Matrix Array capture, FMA) that is used to form the image. According to the description of professors Drinkwater and Wilcox [1–3], its name refers to the possibility of implementing dynamic focusing in emission and reception, which enables to obtain images perfectly focused at all points in the region of interest. However, the complexity of the acquisition process and the computational requirements of the beamforming make this method not appropriate for real-time purposes [1]. Other solutions that use an emission and reception sub-aperture have been also proposed [4–6], although they maintain a certain degree of hardware complexity (focussing is needed in emission and reception) and also require intensive computational capabilities to produce a real-time ultrasonic image.

To overcome the last inconveniences we propose a SAFT methodology based on a new paradigm, known as coarray [5, 6], which allows to use only one element in emission and a limited number of parallel channels in reception at each time. With the proposed solution, a strategy for a hardware reduction in ultrasonic imaging systems is possible, and it involves the following aspects:

- Optimization of the acquisition strategies to achieve the completeness of the coarray with a minimum number of hardware elements. In this sense, our objective is to establish a trade-off between the number of electronic channels, image quality and acquisition velocity [6].
- The use of pulse compression techniques to overcome the reduced capability of penetration when emission is limited to one element [5].
- The development of GPGPU[1] parallel beamforming techniques to achieve real time imaging [7].

This chapter is divided into two main sections. The first one is dedicated to analyse the use of the coarray paradigm as a tool for the design of ultrasonic imaging systems and to present several minimum redundancy coarray techniques. Moreover, Golay codes are presented and their integration within the presented SAFT methods is described. The second section presents the general ultrasonic imaging system's overview, its architecture and the parallel beamforming as a solution for ultrafast beamforming. Finally, we expose our conclusions and future research developments.

---

[1] General-purpose computing on Graphics Processing Units is the utilization of a graphics processing unit (GPU), which typically handles computation only for computer graphics, to perform computation in applications traditionally handled by the central processing unit (CPU). http://gpgpu.org

2. Coarray: New paradigm for the design of imaging systems

This section is focused on the development of ultrasonic imaging systems based on the pulse/echo aperture model which is known as coarray. In order to clarify this point, we are going to briefly review this mathematical concept and its principal implications.

The coarray is a mathematical tool that is often used by several authors as a way to quickly study the radiation properties of an imaging system [5, 6, 8, 9]. This concept is frequently referred to as *effective aperture* in ultrasound literature, and it basically is the virtual aperture which produces in one way the same beam pattern as the real aperture working in emission and reception as Figure 12 suggests.

Suppose a linear array with N elements. In far-field and assuming very narrow band signals, the radiation pattern could be written as:

$$f(u) = \sum_{n=0}^{N-1} a_n e^{jkx_n u} = \sum_{n=0}^{N-1} a_n e^{jkndu} = \sum_{n=0}^{N-1} a_n (e^{jkdu})^n \tag{1}$$

where a_n are the complex weights of the transducers and $u = sin(\theta)$ being θ the angle measured from the perpendicular to the array. Substituting e^{jkdu} by the complex variable z, the radiation pattern can be expressed as a polynomial, which corresponds with the Z-Transform of the sequence a_n. Thus, considering a pulse-echo system, the complex radiation pattern will be the product of two polynomials with degree $N-1$:

$$f_{total}(z) = Z\{c_n\} = \sum_{n=0}^{2N-2} c_n z^n = \sum_{n=0}^{N-1} a_n z^n \; . \; \sum_{n=0}^{N-1} b_n z^n \tag{2}$$

where a_n and b_n are the gains applied to the transducers in emission and reception, and c_n is the coarray ($Z\{c_n\}$ represents the Z-Transform of the sequence c_n). Returning to the unit circle ($|z| = 1$, $z = e^{jkdu}$) and considering equation 1 then the radiation pattern of the system in continuous wave is directly the DFT of the coarray [10].

In synthetic aperture systems, each scanned image is obtained after several firing sequences of the elements. According to this, the coarray can then be expressed as a sum of several sub-coarrays. Each of these sub-coarrays will be obtained as the convolution of two sub-apertures that represent the weights of the active elements used to emit and receive the signals each time.

Figure 1 illustrates the coarray generated by TFM method, which has been applied in ultrasound area since the late 60's and early 70's [11, 12]. As we briefly introduced in Section 1, it consists on the sequential emission with each one of the array elements in turn, and the reception in each shot with the full transducer aperture. As we can see, its coarray is fully populated what ensures a grating-lobe free radiation pattern.

The image quality achieved when TFM is employed is the highest possible, but it has, as its counterpart, the huge volume of data which is necessary to acquire. Thus, it requires more storage resources and processing capability than other techniques, which makes difficult its

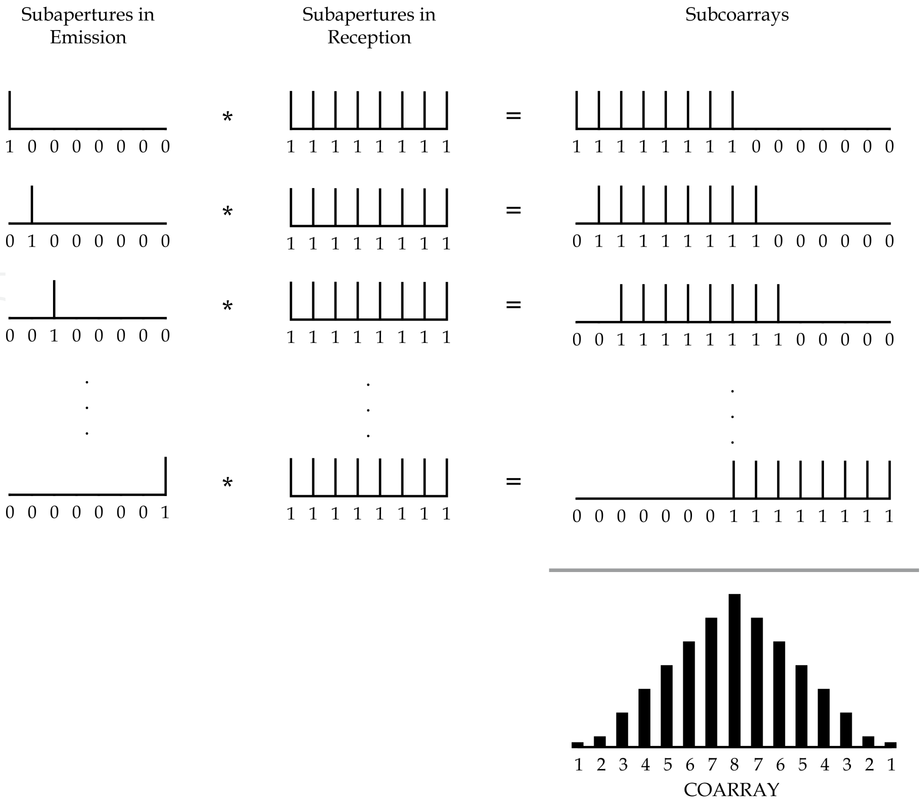

Figure 1. Firing sequences of the elements in TFM, and its corresponding generated coarray

practical implementation with todays' technology. To illustrate this, consider the following example: a 15 cm depth image, 40 MHz sampling rate, 64 channels, 1500 m/s medium velocity and 2 bytes per sample. Each firing generates approximately 1 MB of pulse-echo data, what supposes 64 MB of data to generate a single image frame when TFM is applied. For a frame rate of 20 images per second, it would be necessary to acquire and process 1.2 Gbytes of data per second.

The bandwidth of most I/O standards available today put in evidence that any of current data protocols can not deal with TFM requirements. Supposing a good efficiency and use of the resources (around 80)%, USB 2.0 port (released in April 2000) would be able to transfer less than one image per second (48 MB/s). A similar situation occurs if USB 3.0 (released in November 2008) is employed, being the maximum transmission speed up to 480 MB/s allowing to transfer around 7 images per second, even far respect to the maximum number of images which could be theoretically achieved. Finally, the most recent standard released in February 2011, known as Thunderbolt port and developed by Intel [13], combines PCI Express and DisplayPort into a new serial data interface that can be carried over longer and less costly cables. Thunderbolt has twice the transfer speed of USB 3.0 over copper wire (960 MB/s) giving us transferences of 14 images per second.

Therefore, it is clear that a reduction of data volume is desirable. In this sense, applying the coarray concept permits us to propose system designs that use less channels simultaneously working in emission and reception, but maintaining the same level of image quality. The key point for this is to use the coarray to search for solutions of minimum redundancy. This approach in conjunction with parallel computing techniques will offer an increment of acquisition velocity maintaining the highest quality and producing high frame rates with low power consumption. This topic will be the main focus of next two sections.

2.1. Minimum redundancy coarray solutions

Coarray analysis identifies which emitter-receiver combination completes each of its elements. In the TFM method seen before, we find that some of the elements are formed by a single signal (in concrete boundary elements) while the others increase progressively until reaching coarray centre with a value of N elements (Figure 1). Thus, we can consider as a minimum redundancy coarray that in which each element is composed of only one signal. Therefore, using the minimum possible number of signals the aperture's diffraction properties can be improved by manipulating the gain of the elements. With this goal in mind, it is possible to establish several strategies which maintain a balance between the number of parallel channels and the number of shots during acquisition processes.

2.1.1. 2R-SAFT acquisition strategy

2R-SAFT technique [14] has some particular advantages that make it very useful for ultrasonic imaging systems. 2R-SAFT uses only one element to transmit and two elements to receive. As it is shown in Figure 2, all elements are consecutively activated as single emitters, without the use of any beamformer in emission. At each shot, two consecutive channels are used as receivers requiring to store two signals per emission.

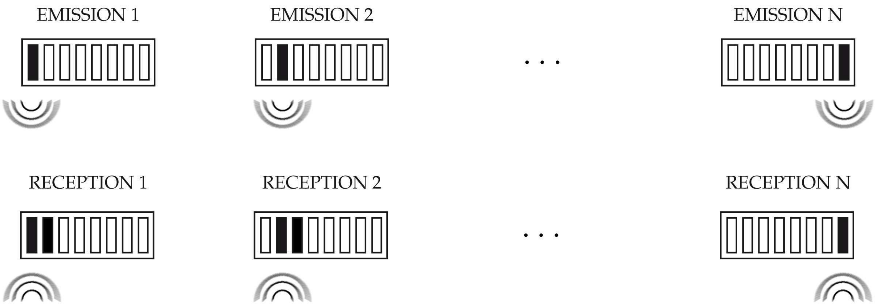

Figure 2. Firing sequences of the elements in 2R-SAFT

Thus, when the i^{th} element is used to emit a waveform, i and $i+1$ elements are used for receiving signals. For the last element of the array, only one signal is recorded. By employing an emitter in each shot all the received signals are completely uncorrelated, containing only information of a single transmitter-receiver pair.

Figure 3 shows the coarray generated when 2R-SAFT is employed. As we can observe, the coarray is fully populated ensuring the suppression of grating lobes in the radiation pattern which produces good quality images [14, 15].

2.1.2. Accelerated-SAFT acquisition strategy

Here we present a minimum-redundancy technique we have denominated Accelerated-SAFT or, in its short form, kA-SAFT. The k subscript refers to the acceleration factor carried out during the acquisition stage which can go from 2x to Nx depending on the number of

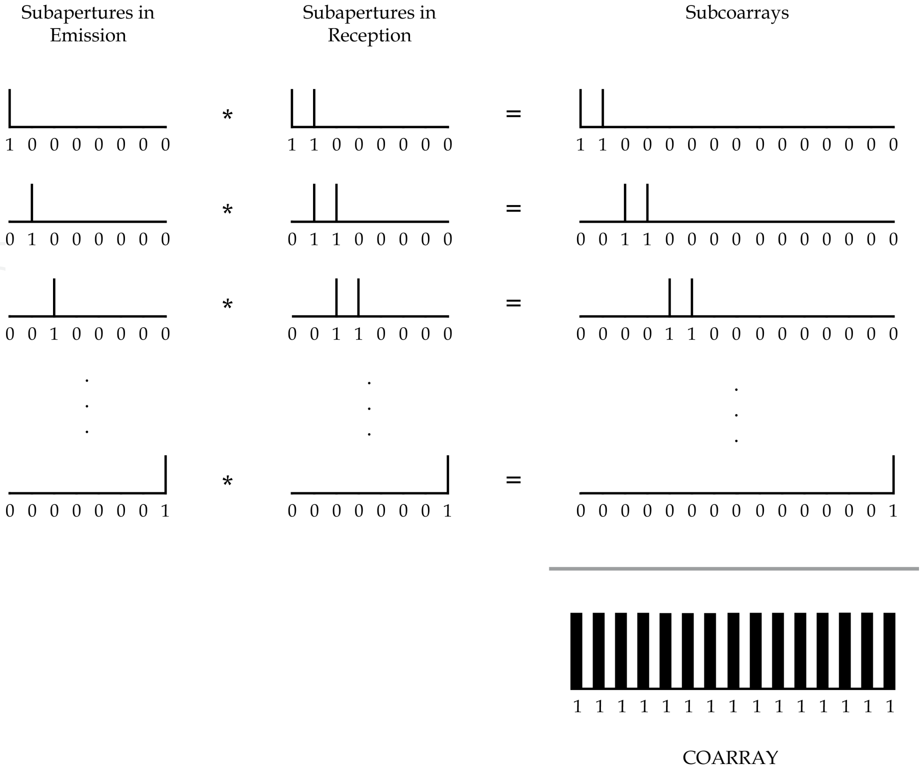

Figure 3. Coarray sequences for 2R-SAFT

channels used for the reception. This strategy increases a little bit the cost involved in the acquisition system respect to 2R-SAFT, but at the same time, reduces the number of shots by k times.

The kA-SAFT uses n_A consecutive elements to receive and a single element to emit which is centred in the active subaperture. As shown in Figure 4, the elements on emission are sequentially activated with a shift of $\frac{n_A}{2}$ elements. At each shot n_A consecutive channels are used as receivers, needing to store n_A signals per emission except for the first and the last array elements where half of the signals is acquired.

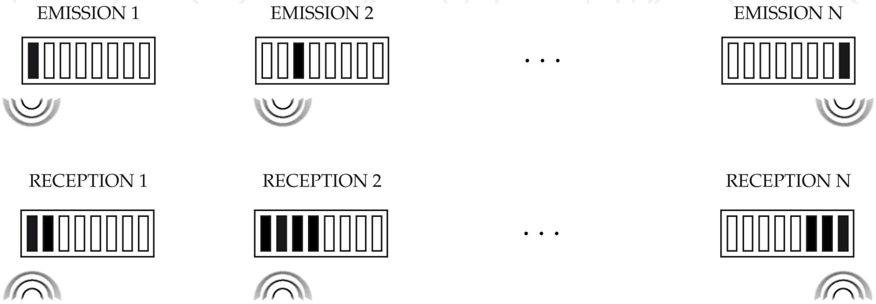

Figure 4. Firing sequences of the elements in kA-SAFT being k = 2x and $n_A = 4$

In this sense, when the i^{th} element is used to emit the elements that are going to use as receivers are given by:

$$Elements_{rx} = \left\{ i - \frac{n_A}{2} + j \right\} \qquad 0 \leq j \leq n_A \tag{3}$$

Figure 5 shows the coarray generated when kA-SAFT is employed for the case of $n_A = 4$. As we can observe, the coarray is identical to that obtained with 2R-SAFT (Figure 3) preserving all its advantages but multiplying by 4 the frame rate in acquisition.

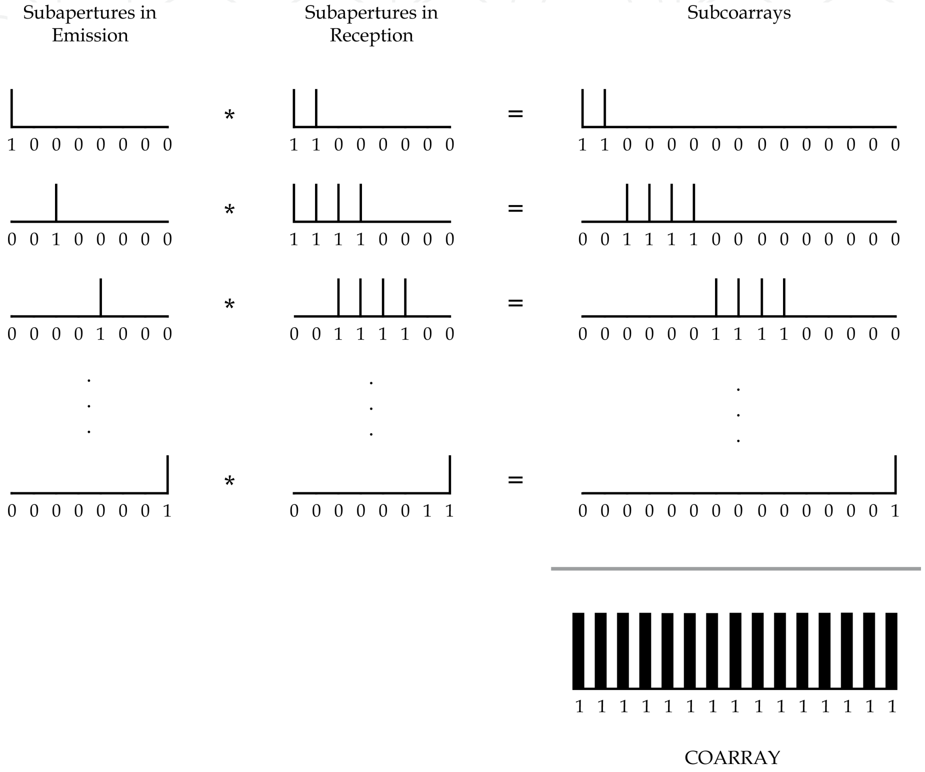

Figure 5. Coarray sequences for kA-SAFT being k = 2x and $n_A = 4$

2.1.3. Experimental results

We present some experimental results that have been done on a tissue phantom (Model 040GSE - CIRS Inc.) with 0.5dB/cm attenuation, where several cysts and wires of 0.1mm diameter are located at different depths (Figure 6). We have used a 2.6MHz phased array transducer with $N = 64$ elements, 0.28mm of pitch (Vermon Inc.) for the measurements. We will use the Total Focusing Method as a reference model to examine the cysts and wires in the tissue covering an area starting from 25mm to 80mm depth, and we will compare it to 2R-SAFT and kA-SAFT techniques. All images have been obtained by applying the DAS algorithm. TFM uses the complete set of signals N^2 = 4096 while 2R-SAFT and kA-SAFT images have been calculated using $2N - 1$ = 127 signals.

In Figure 7, images for all strategies are presented. It is easily observed how Figures 7(a,b,d,e,g) are very similar in terms of quality. Consequently, the strategy to be chosen relies fundamentally on the hardware requisites.

Nevertheless, at a depth greater than 60 mm none reaches the same contrast level as TFM (Figure 7(h)), highlighting the limited signal to noise ratio suffered by all minimum

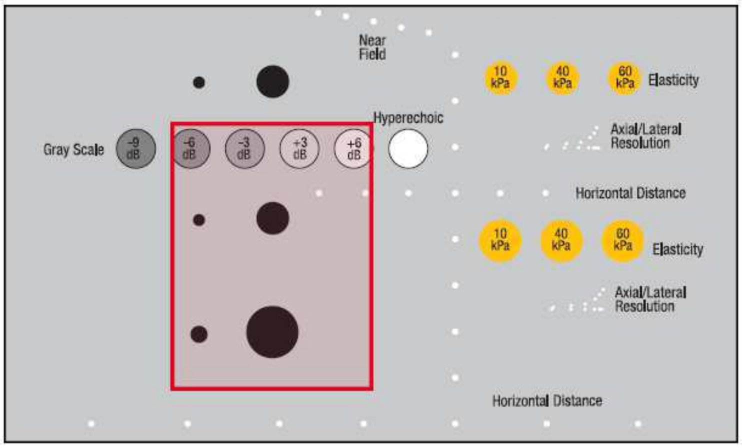

Figure 6. Region of interest analysed from tissue phantom model 040GSE by CIRS Inc.

redundancy techniques. In table 1, a comparison between the number of channels in emission and reception, number of firings, acquisition frame rates and memory buffers needed is performed for the different strategies presented. As we can see, TFM is the technique which more storage as well as more hardware channels needs. By contrast, minimum redundancy techniques requisites are more affordable and suitable for applications where size matters.

Strategy	Channels (tx,rx)	Firings	Framerate	Buffer
2R-SAFT	(1,2)	$N_{firings} = N$	$f_{frame} = \frac{f_{prf}}{N}$	$2N - 1 \times L$
2xA-SAFT ($n_A = 4$)	(1,4)	$N_{firings} = \frac{N}{2}$	$f_{frame} = 2\frac{f_{prf}}{N}$	$2N - 1 \times L$
4xA-SAFT ($n_A = 8$)	(1,8)	$N_{firings} = \frac{N}{4}$	$f_{frame} = 4\frac{f_{prf}}{N}$	
8xA-SAFT ($n_A = 16$)	(1,16)	$N_{firings} = \frac{N}{8}$	$f_{frame} = 8\frac{f_{prf}}{N}$	
16xA-SAFT ($n_A = 32$)	(1,32)	$N_{firings} = \frac{N}{16}$	$f_{frame} = 16\frac{f_{prf}}{N}$	
TFM	(1,N)	$N_{firings} = N$	$f_{frame} = \frac{f_{prf}}{N}$	$N^2 \times L$

Table 1. Comparison of the several acquisition strategies presented

2.2. Golay Codes

As we have seen, synthetic aperture images have low contrast due to the poor signal to noise ratio (SNR). Along this section, we will study how the use of pulse coding based on Golay codes [16, 17] can help to improve the dynamic range and SNR, in order to achieve an image quality comparable to that of Total Focusing Method.

2.2.1. Golay encoding for ultrasonic excitation

Golay complementary pairs have been widely used for transducer excitation because the sum of its auto-correlation function has a main peak and zero side-lobes [16]. A complementary pair is composed of two binary sequences, $A[n] = [a_0, a_1, \ldots, a_{N-1}]$ and $B[n] = [b_0, b_1, \ldots, b_{N-1}]$, of the same length N such that $a_i, b_i \in \{-1, +1\}$.

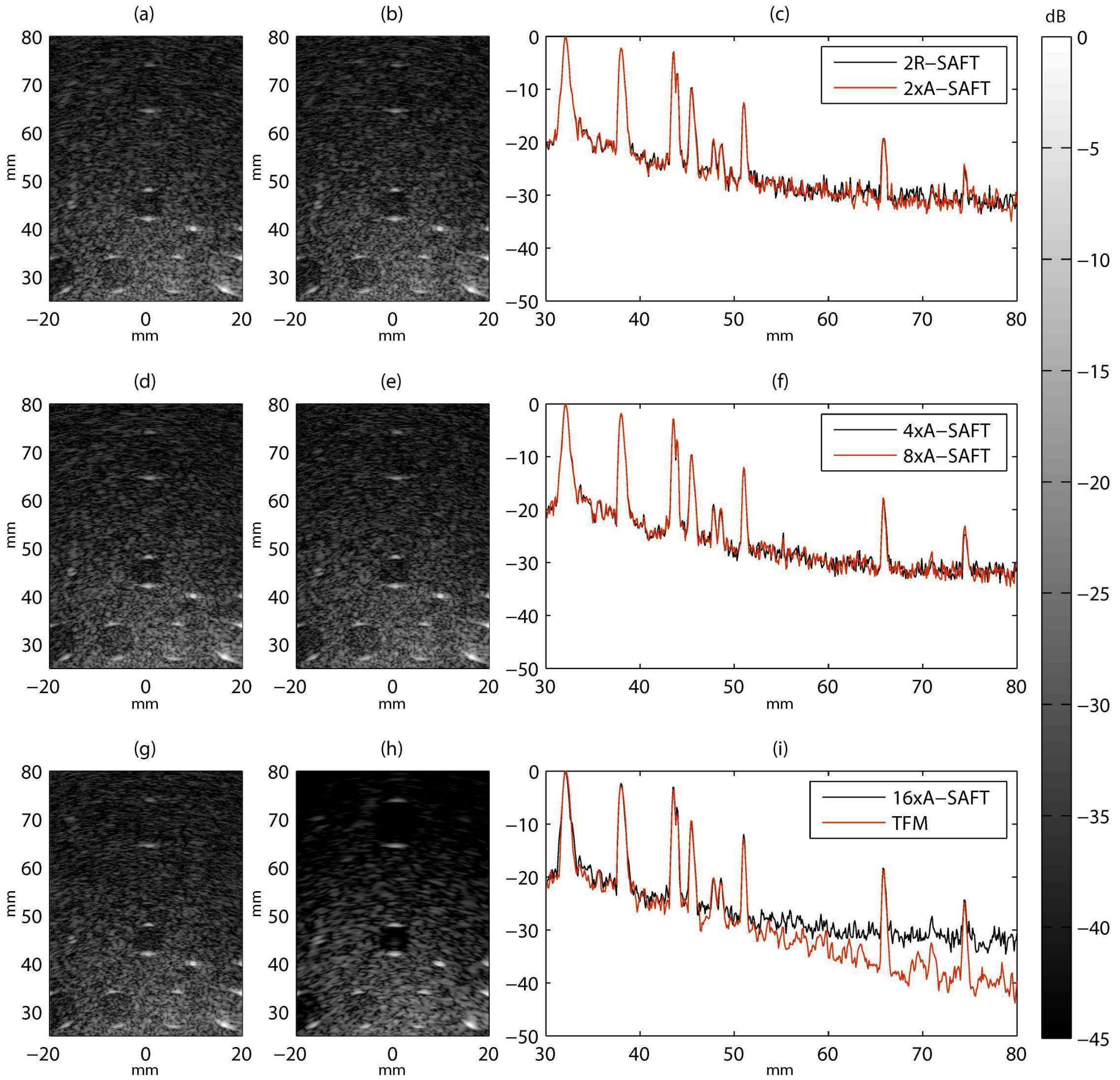

Figure 7. Experimental images from tissue phantom. (a) 2R-SAFT, (b) 2xA-SAFT, (c) Lateral profiles comparison between 2R-SAFT and 2xA-SAFT, (d) 4xA-SAFT, (e) 8xA-SAFT, (f) Lateral profiles comparison between 4xA-SAFT and 8xA-SAFT, (g) 16xA-SAFT, (h) TFM, (i) Lateral profiles comparison between 16xA-SAFT and TFM

The auto-correlation functions of $A[n]$ and $B[n]$ have side lobes with equal magnitude but opposite sign. The sum of these independent auto-correlation functions provides an ideal delta function according to:

$$C_A[n] + C_B[n] = \begin{cases} 0, \, n = 0 \\ 2N, \, otherwise \end{cases} \tag{4}$$

where $C_A[n]$ and $C_B[n]$ are the auto-correlation functions of $A[n]$ and $B[n]$, respectively, for any integer n satisfying the equation 4. The construction of Golay code pairs is done recursively with the *"negate and concatenate"* method, a technique used by Golay [16] to create longer pairs from shorter hand-constructed given pairs. Specifically, if $A[n]$ and $B[n]$ are the N-digit binary representations of a complementary pair of codes, then a new pair of complementary codes $A'[n]$ and $B'[n]$ of length $2N$ can be formed by concatenating $B[n]$ to $A[n]$ and concatenating $\sim B[n]$ to $A[n]$ where $\sim B[n]$ is the complement of $B[n]$. Thus, $A'[n] = A[n] \mid B[n]$, and $B'[n] = A[n] \mid\sim B[n]$.

One of the major drawbacks of Golay codes is that two shots are needed for each emitting element in order to complete both A and B codes respectively. In our work, Golay codes of length equal to 8 bits have been used, being $A[8] = [+1+1+1+1+1-1-1+1]$ and $B[8] = [+1-1+1-1+1+1-1-1]$, producing a gain of 24dB according to equation 4.

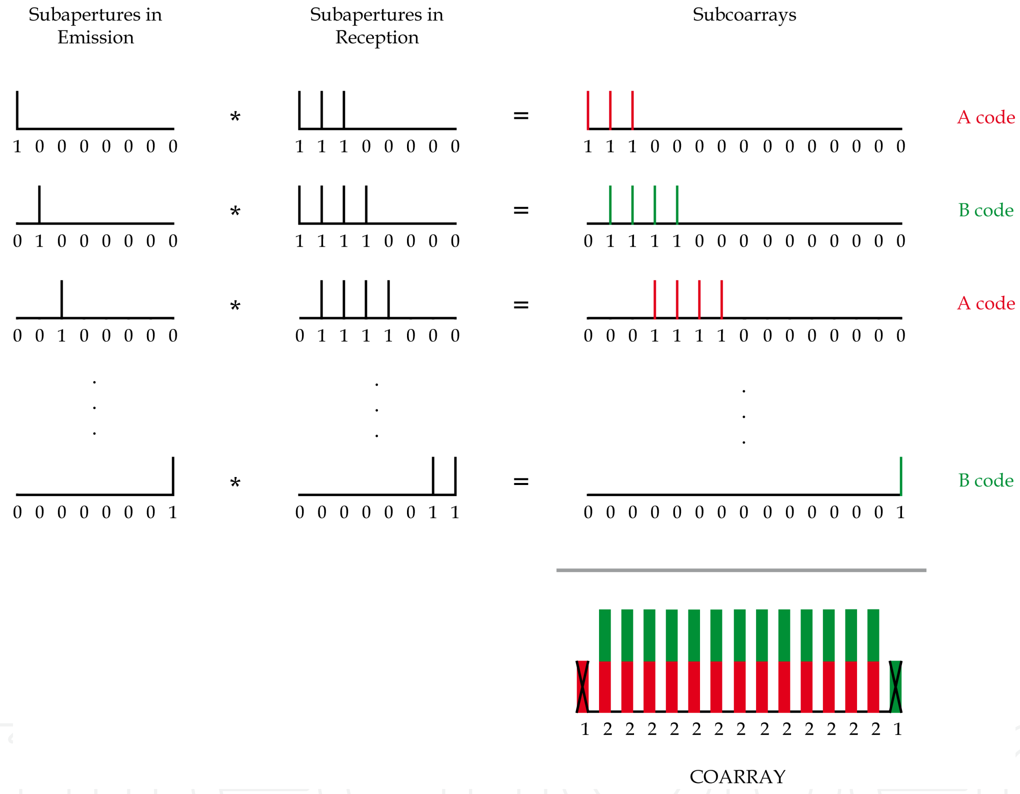

Figure 8. Golay encoding integration example

2.2.2. Coarray for Golay encoding

Golay codes, described previously, and minimum redundancy techniques can be combined. In order to illustrate how this can be done, Figure 8 shows an example using a 4R-SAFT (four receivers) [15] plus Golay codes. Here, two signals per coarray element are acquired and because Golay encoding needs to fire twice, A or B codes are alternated between shots.

This process is mathematically identical for the formerly presented strategies 2R-SAFT and kA-SAFT but with some particularities:

1. The number of channels in reception (sensors) must double the original number, in order to have two signals per coarray element for A and B codes.
2. The amount of acquired data signals also doubles the original, because of the first point.
3. The original firing rate is preserved, which means achieving identical performance at the expense of doubling the hardware involved in the reception process.

2.2.3. Experimental results

With the use of Golay codes to image the same area than in previous results, the panorama has changed. As before, TFM image has been composed from the complete set of signals N^2 = 4096, but now 2R-SAFT and kA-SAFT have been calculated using $4N - 2 = 254$ signals. From Figure 7 in section 2.1.3, where the corresponding images with no encoding were analysed, it can be seen how the reduction in the number of signals employed produces a loss of dynamic range respect to TFM method. Thus, with the use of Golay codes in Figure 9 we can observe how the contrast and level of detection have substantially increased. Now, both 2R-SAFT and kA-SAFT techniques distinguish the complete set of defects. Thus, in relation to TFM the number of signals is drastically reduced from N^2 to $4N - 2$, accelerating acquisition and processing velocities and the system's frame rate.

3. Ultrasonic imaging system

3.1. General system's overview

As we have said, our goal is centred in the design of ultrasonic imaging systems based on solutions which require fewer resources and storage capacity than conventional systems. Thus, in Figure 10 is schematically represented our vision of the system, which is composed by three parts:

1. **The array or probe**. It is usually composed by 64, 96, 128 or even more transducers depending on the type of application.
2. **Acquisition subsystem**. The hardware subsystem used for transducer excitation and data acquisition (represented by the box in the center). Nowadays, several electronic manufacturers have in their catalogues electronic boards and systems, which are small and can be easily used for our purposes. For example, National Instruments has 32-channel digitizer module capable of sampling on all channels at 50 MS/s with 12-bit resolution. The module is optimized for ultrasound applications [18]. Additionally, both multiplexer and bipolar programmable pulser are required. Specific architectures depending on the type of acquisition strategy will be studied in the next section.
3. **Image generation subsystem**. It is the software system which can take place in any computational device (PC, laptop, ...) shown on the right side of Figure 10. These processes include the digital signal pre-processing of the received signals and filtering; beamforming of the image, delaying and adding signals according to emission and reception lenses, post-processing the image and its representation to properly show data on the screen. To achieve these tasks, the use of GPU's great power for parallel computing will allow us to quickly and efficiently accelerate the algorithms.

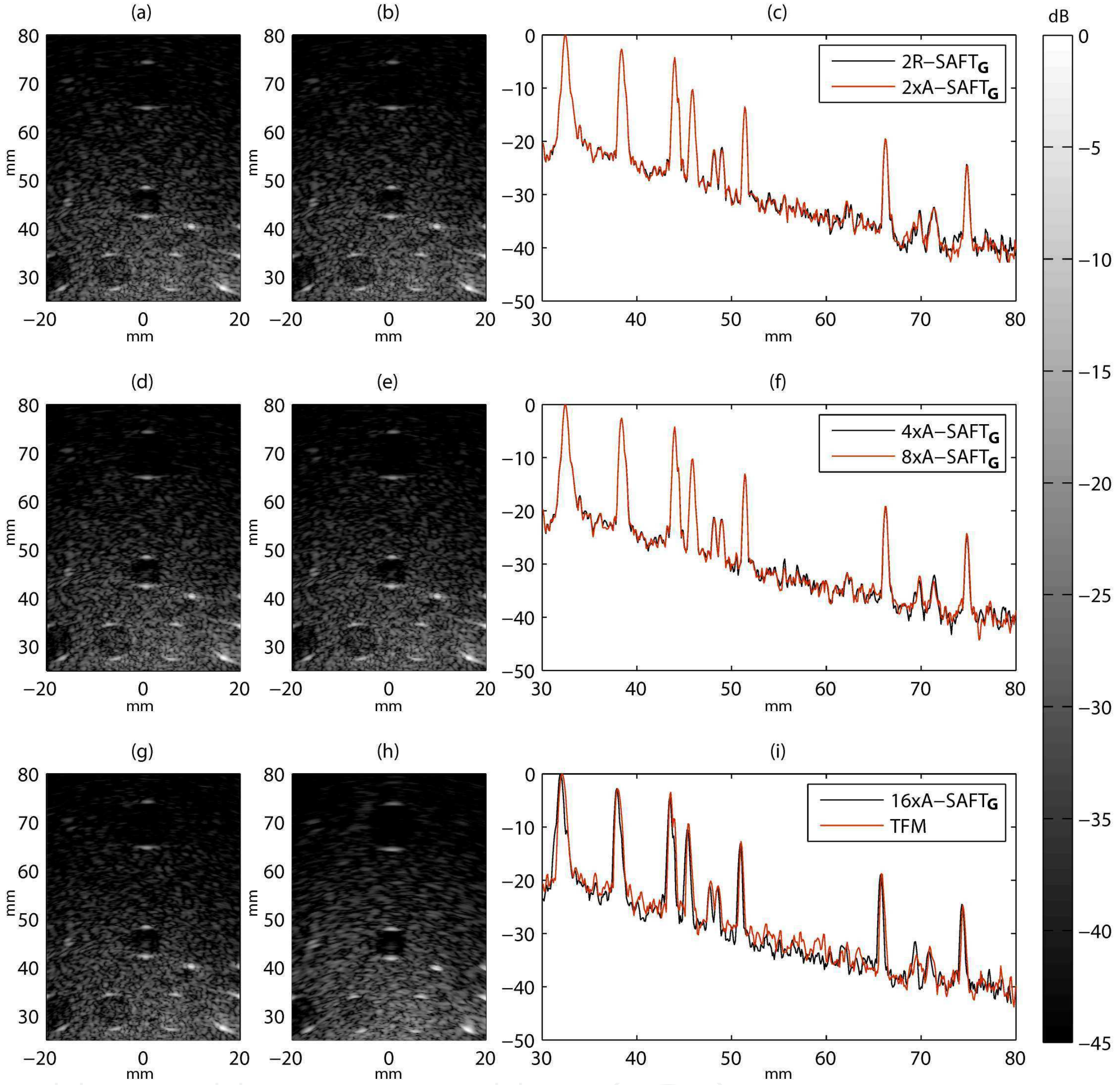

Figure 9. Experimental images from tissue phantom. (a) 2R-SAFT + Golay, (b) 2xA-SAFT + Golay , (c) Lateral profiles comparison between 2R-SAFT + Golay and 2xA-SAFT + Golay, (d) 4xA-SAFT + Golay, (e) 8xA-SAFT + Golay, (f) Lateral profiles comparison between 4xA-SAFT and 8xA-SAFT + Golay , (g) 16xA-SAFT + Golay, (h) TFM , (i) Lateral profiles comparison between 16xA-SAFT and TFM

3.2. Acquisition subsystem

In this section, two acquisition architectures are exposed. On one hand, a minimal system for 2R-SAFT strategy which allows a low-cost and small imaging system and, in the other hand, the architecture which implements 8xA-SAFT plus Golay encoding strategy and uses more hardware but yields better quality images. Which strategy to use depends on the concrete application. Any of these configurations can be carried out using boards systems available in the market.

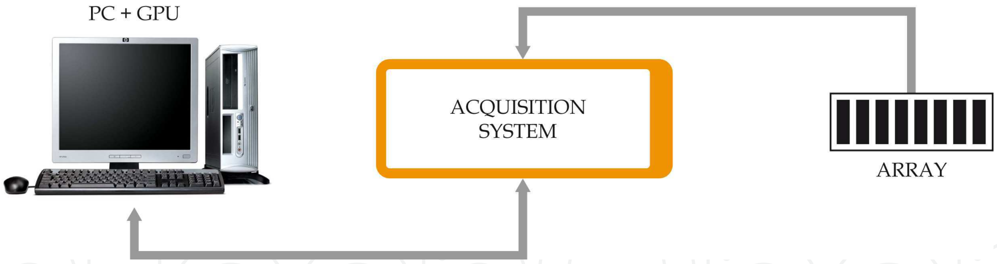

Figure 10. Hardware/Software system proposed

3.2.1. 2R-SAFT architecture

As we study in section 2.1.1, it is basically composed of one channel in emission and two channels in reception. Figure 11 shows the complete architecture for 2R-SAFT implementation. As we can see, a multiplexer is connected to the transmission channel for sequentially activate each element as an emitter, and a second multiplexer will be on charge of connecting the selected elements to both reception channels.

All the acquisition process is managed by a hardware control system which is located in a field-programmable gate array (FPGA). In addition, a local memory is also used to store every received signal. Finally, the signals are transferred to the imaging system using any communication interface (USB, Ethernet, PCI Express). In the imaging system, raw data is stored in a RAM memory of $2N-1$ signals of capacity to be used for compose and beamform the ultrasonic images using a GPU.

3.2.2. 8xA-SAFT with Golay encoding architecture

As we see in sections 2.1.2 and 2.2, and in order to combine 8xA-SAFT with Golay codes, we will double the number of channels in reception to maintain the number of original firings. Thus, in this case the system is composed of one channel in emission and 32 channels in reception as Figure 12 suggests. A multiplexer connects the transmission channel to elements for sequentially activate one of them, in steps of 8 elements, to transmit an A or B code for odd or even shots respectively. A second multiplexer will be on charge of connecting the 32 reception channels to the receiving aperture ensuring that every coded signal is stored in a local memory. Therefore, two signals per coarray element are overlapped, each one belonging to an A or B code respectively. Additionally, an offset is added to the coarray structure in order to centre its elements, and the boundary coarray elements are removed from it as we illustrated in section 2.2.2.

Now the software imaging system requires a bigger memory and an additional decoding stage, where the complete set of signals is deconvolved, generating a $2N-8$ data set. Later on, as usual, the data will be beamformed using the graphics processing unit.

3.3. Image generation subsystem: Parallel beamforming

In recent years, computing industry has been opened a way to parallel computing. Nowadays, all consumer computers ship with multi-core processors. Dual-core processors (CPUs) were introduced in personal systems at the beginning of 2006, and it is currently common to find them in laptops as well as 8 and 16-core workstation computers, which

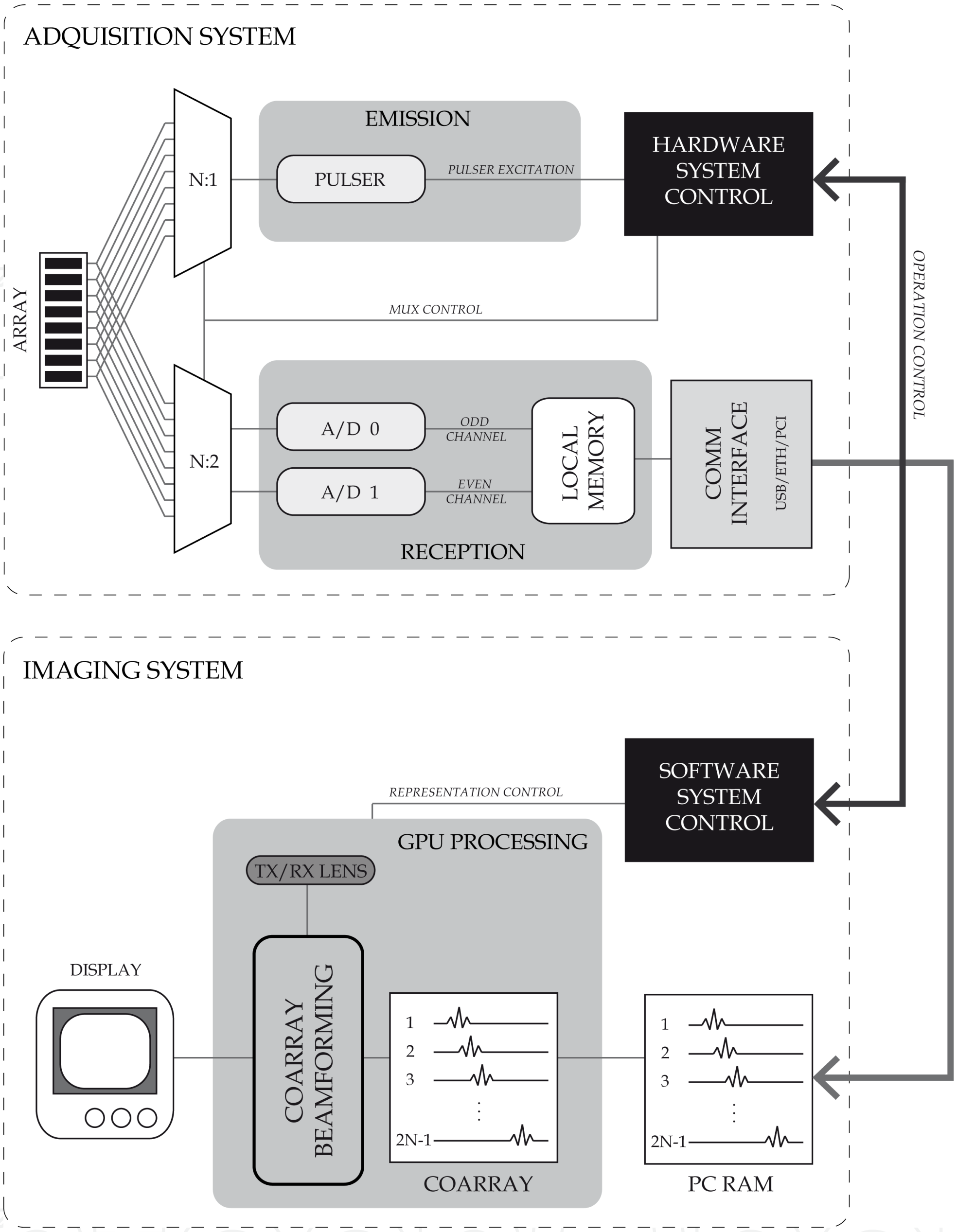

Figure 11. 2R-SAFT Minimal Architecture

means that parallel computing is not relegated to big supercomputers or mainframes computers. However, Graphics Processor Units (GPUs), as their name suggests, came about as accelerators for graphics applications, predominantly those using the OpenGL and DirectX programming interfaces. Although originally they were pure fixed-function devices, the demand for real time and 3D graphics made them evolve into increasingly flexible highly parallel, multithreaded processors with extremely high computational power and very high memory bandwidth converting them into massively parallel machines.

Unlike earlier GPU generations, where computing resources were partitioned into vertex and pixel shaders, nowadays they can be programmed directly in C using CUDA or OpenCL [19], APIs which include a unified shader pipeline, allowing each and every arithmetic logic

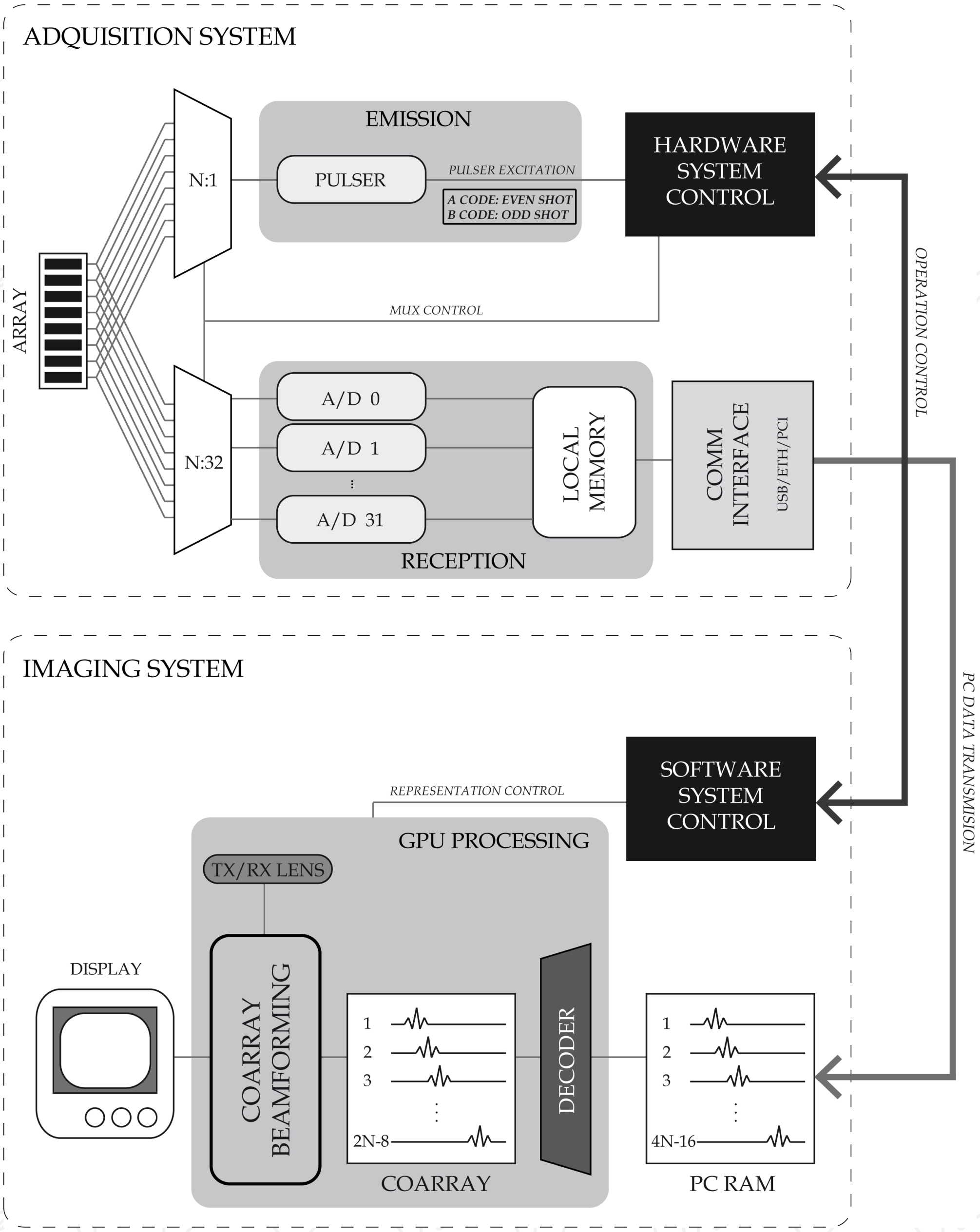

Figure 12. 8xA-SAFT Architecture with 32 channels in reception needed for Golay encoding

unit on the chip to be used by a program intending to perform general-purpose computations (GPGPU). Furthermore, the execution units on the GPU allow arbitrary read and write access to memory as well as access to a software-managed cache known as shared memory. A CUDA program consists of one or more phases that are executed on either the host (CPU) or a device such as a GPU. The phases that exhibit little or no data parallelism are implemented in CPU code. The phases that exhibit rich amount of data parallelism are implemented in the GPU code. The parallel functions (called kernels) typically generate a large number of threads to exploit data parallelism. It is worth noting that CUDA threads are of much lighter weight than the CPU threads. CUDA programmers can assume that these threads take very few cycles to generate and schedule due to efficient hardware support. This differs from

CPU threads which typically require thousands of clock cycles for their generation and their scheduling.

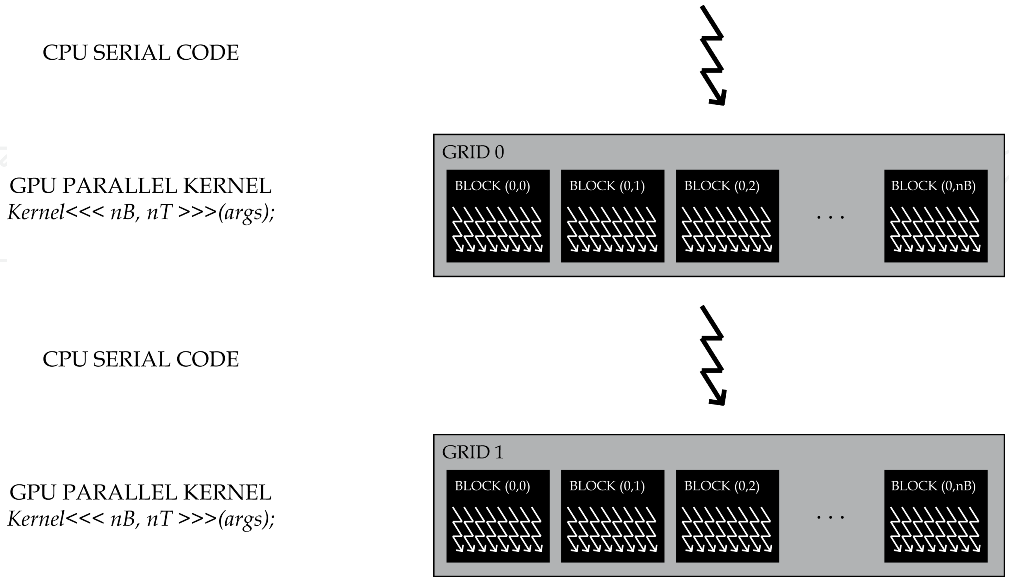

Figure 13. CUDA program execution diagram

The execution of a typical CUDA program is illustrated in Figure 13 where it is observed that the execution starts with host (CPU) execution. When a kernel function is invoked (or launched), the execution is moved to a device (GPU), where a large number of threads are generated to take advantage of huge data parallelism. All the threads generated by a kernel during an invocation are collectively called a grid. Figure 13 shows the execution of two grids of threads. A grid is a 1D, 2D or 3D structure of blocks, and a block is a 1D, 2D or 3D structure of threads. Thus, the program code is composed by classical functions, which run on CPU using only one thread of execution; and kernels, which run on GPU using multiple parallel threads. When all threads of a kernel complete their execution, the corresponding grid terminates, and the execution continues on the host until another kernel is invoked. It is not our purpose to fully cover all the aspects involved in CUDA Architecture. Thus, an extended discussion about the CUDA hardware and programming model is available in multiple sources in the literature [19–21].

Therefore, in this section we will examine different ways to implement the beamforming process on the GPU using the CUDA programming model. From the model, it is extracted that functions which are executed many times independently over different data are the ideal candidates for this kind of computing. In this sense, several algorithms have been implemented to cover the fundamental parts of a conventional Delay-and-Sum Beamformer (DAS) and they have been also evaluated for their performance. This analysis helps to give a better understanding of the GPU architecture and how to write applications for it.

Schematically, Figure 14 show the main stages of a general beamformer. As we can appreciate there are three main operations to be done: pre-processing of signals, beamforming and post-processing. In the software system we propose (Figure 10) all beamforming procedures take place in the GPU.

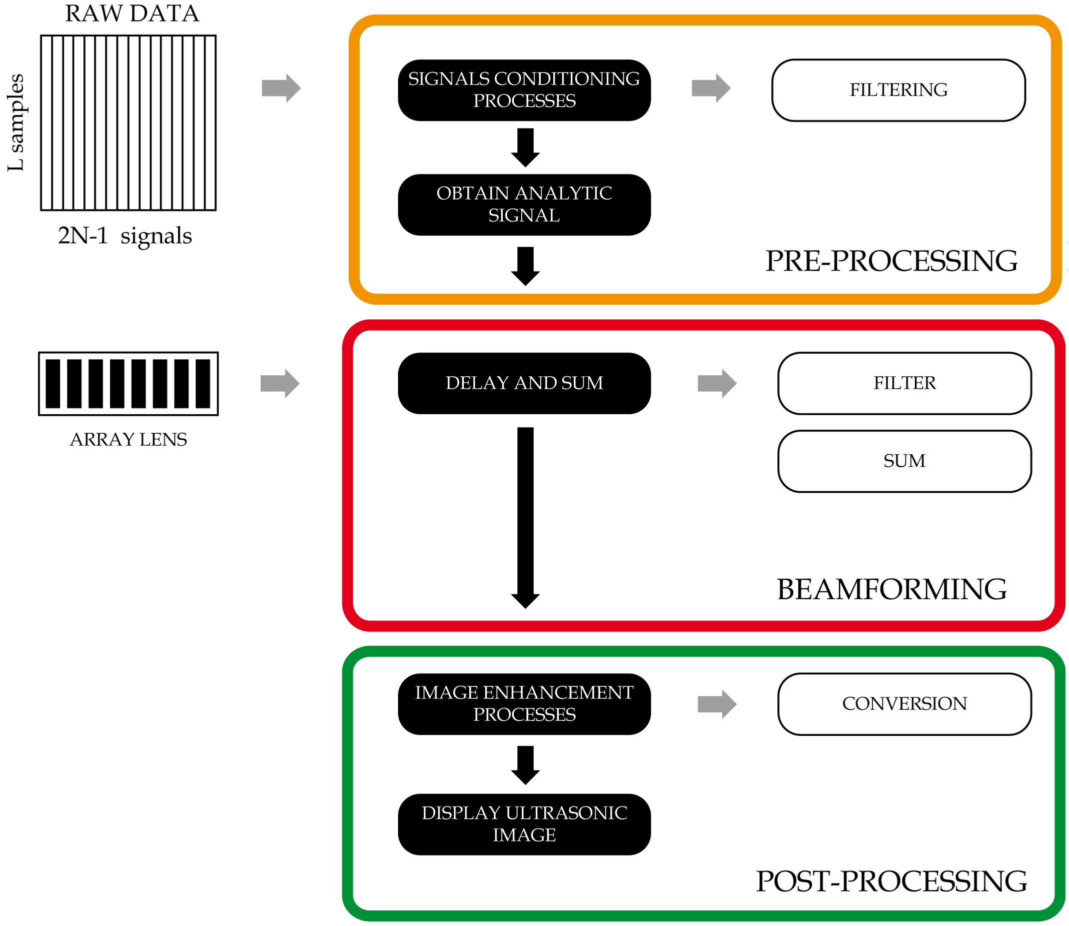

Figure 14. Schematic diagram main parts of a general SAFT beamformer

Implementing the imaging algorithm on GPU systems primarily involves the parallelization of the core algorithm into small independent threads which can be executed by the GPU in runtime. Thus, the imaging process occurs in multiple stages, which follows closely to that has been detailed in Figure 14. Thus, in order to maximize GPUs efficiency and reduce image generation time as much as possible, a specific solution for every different task have been designed. Figure 15 shows how these tasks have been parallelized on the GPU.

The first step consists on copying the complete set of acquired signals from CPU memory to GPU memory. We already know that this transaction is slow, and therefore it is recommended to copy all signals at the same time rather than doing it signal by signal.

3.4. Pre-processing

The pre-processing of the complete set of signals is a fundamental part of the image generation process. Supposing $X_{tx,rx}(t)$ the received signal from any emitter tx and receiver rx pair, a function $H(t)$ is applied to every signal as the following expression suggests:

$$Y_{tx,rx}(t) = X_{tx,rx}(t) \cdot H(t) \tag{5}$$

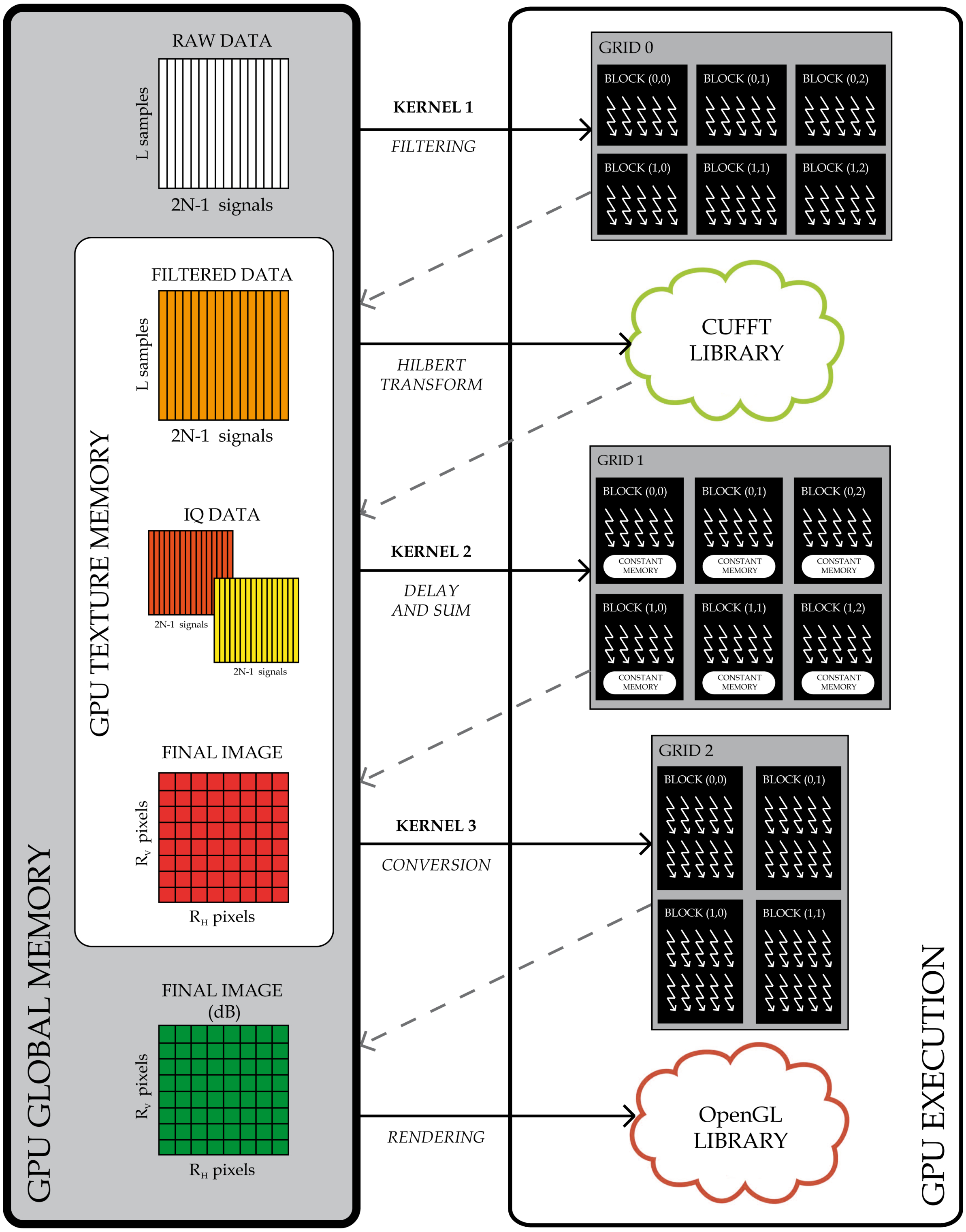

Figure 15. System beamforming loop parallelized on GPU for SAFT implementation

and

$$H(t) = H_F(t) \cdot H_{IQ}(t) \tag{6}$$

where $H_F(t)$ is a signal conditioning process where a filter is applied in order to remove the offset level introduced during the acquisition system and to reduce the noise.

Additionally and for convenience, the acquired signals can be decomposed into their analytic signals form [22] (in-phase I and quadrature components Q) . Thereby, the second function $H_{IQ}(t)$ is the Hilbert Transform in order to reduce errors and artefacts which appear at the envelope detection stage. Then, the signals $X_{tx,rx}(t)$ can now be expressed as:

$$X_{tx,rx}(t) = I_{tx,rx}(t) + jQ_{tx,rx}(t) = \mathbf{X}_{tx,rx}(t)e^{j\phi_{tx,rx}(t)} \tag{7}$$

where $\mathbf{X}_{tx,rx}(t)$ is the modulus and $\phi_{tx,rx}(t)$ its corresponding signal phase.

3.4.1. Parallel implementation

In order to carry out a parallel implementation of these operations, the proposed parallelism strategy lies in a signal-oriented parallelization. This means that a GPU computational thread will be associated to each stored signal sample. Thus, considering signals with L samples, the computational grid of the kernel will be formed as shown in Figure 15 being the number of blocks in x-dimension $BX = \lceil \frac{L}{T_{BX}} \rceil$ and the number of blocks in y-dimension $BY = 2N - 1$. As we know, the number of threads per block T_{BX} is an empirical value and the designer should evaluate what is the best according to the GPU resources. Typical values are 32, 64 or 128 threads per block, generally any power of two, and attending to our tests we have chosen 256 as the optimal value.

There is no limitation on filter length because its coefficients are stored in texture memory, which resides in the device memory and is cached in texture cache to optimize read accesses. Thus, each thread reads from memory the filter coefficients and L samples of a signal, convolving them to obtain a filtered sample.

Later on, the Hilbert Transform is applied to every filtered signal so we can obtain their analytic signals. In this case, FFT algorithms provided by CUDA (CUFFT libraries [20]) are used to compute the IFFT of the product of the corresponding signal and the Hilbert Transformer FFTs, as it is defined in [23]. With these libraries, there is no need to define a new kernel nor specify grid and block dimensions, since they are responsible for properly parallelizing and splitting the algorithm, computing the FFT of the data set directly on the GPU. In our particular case, a total of $2N - 1$ FFTs of L points are calculated in parallel. The whole resultant I/Q (in-phase/quadrature-phase) signals pairs (Figure 15) are stored in texture memory, and they are passed to the next stage via global device memory.

3.4.2. Optional stage: Decoding

When Golay encoding is used during acquisition, it is necessary to first merge and deconvolve the $4N - 16$ received signals, where 50% of signals belong to A and B codes respectively. This can be done very fast making a parallel implementation where the parallelism strategy is also signal-oriented. In this sense, the previous kernel can be modified in order to include the sum of both signals in parallel before the application of the filters coefficients to finally obtain $2N - 8$ signals.

3.5. Beamforming: Delay and Sum

All time-domain imaging algorithms are based on the principle of delay-and-sum beamforming. Typically, these algorithms emulate an acoustic lens by applying appropriate time delays to the array elements in order to focus or steer the beam as desired. SAFT beamformers focus the beam at every point in the image, giving better defect detectability as we mentioned [2, 4, 5, 7]. DAS beamforming is not difficult to implement and permits the use of arbitrary array geometries what makes suitable for a wide range of applications.

According to the Hilbert transformation of the first step, two processing streams have been created where two parallel images will be calculated following these equations:

$$A_I(x,z) = \sum_{i=1}^{N} \sum_{k=1}^{N} I_{tx_i,rx_k}(D(x,z)) \tag{8}$$

$$A_Q(x,z) = \sum_{i=1}^{N} \sum_{k=1}^{N} Q_{tx_i,rx_k}(D(x,z)) \tag{9}$$

where $A_I(x,z)$ and $A_Q(x,z)$ are the in-phase and quadrature images respectively, and $D(x,z)$ is the focussing delay for the spatial point (x_p, z_p) in the grid which is calculated as follows:

$$D(x,z) = \frac{\sqrt{(x_p - x_{tx})^2 + z_p^2} + \sqrt{(x_p - x_{rx})^2 + z_p^2}}{c} \tag{10}$$

being x_{tx} and x_{rx} the coordinates of the transducer elements tx and rx, respectively.

Henceforth, we will focus on the all the operations involved in Delay-and-Sum algorithm, studying the diverse alternatives and their parallel implementation as well as the best way of their optimization.

- **Lens calculation**. A fundamental part of beamforming is calculating the differences in wave arrival time between array elements. Therefore, each signal sample has to be properly delayed according to the distance from the spatial point to the emitter or receiver array elements. The calculation of delays is achieved using equation 10. Although in a conceptual form is a delay, what is actually done is a mapping to the memory buffer (at the sampling frequency) where the corresponding sample value of the signal is retrieved. Therefore, the number of delays to be calculated is usually large and it is given by:

$$Memory|_{lens} = R_H \times R_V \times 2N - 1 \tag{11}$$

where $R_H \times R_V$ are the dimensions of the desired ultrasonic image. Thus, the lens calculation can be afforded using two different approaches:

- *Load pre-calculated delays*. The delays are pre-calculated before beamforming and they are recovered from a look-up table inside the image generation process. The necessary memory to store all the delays is not a significant problem, but the main drawback is the requirement of high bandwidth to make the process faster as well as the fact of updating the table each time. Thereby, this would be a good solution for no in-vivo inspections, where the scenario is known and the delays are calculated only once for the complete acquisition.
- *Calculate delays on-the-fly*. The delays are dynamically calculated inside the beamforming process. This task, which can be at first computationally more expensive than the first alternative, is however not a heavy computational problem because of the great power of actual systems. In this regard, dynamic calculation of the lenses inside the threads will simplify other operations on images, such as scrolling and zooming.

Which approach to choose relies on the rest of the beamformer implementation. Thus, in order to take full advantage of the GPU it is needed to have a balance between bandwidth use and arithmetic operations. In this regard, it has been proved that it is faster to obtain the values for the lenses inside the kernels instead of having them stored in the device memory. Therefore in our proposal, it makes sense to calculate the delays on-the-fly.

- **Filtering**. In a real implementation, we sample the elements at a rate just above the Nyquist criteria. Although this preserves the frequency content of the signal, this does not give enough steering delay resolution. The solution is to perform a digital interpolation, increasing the steering-delay resolution. In this particular case, linear interpolation and polynomial interpolation can be easily implemented. The results obtained are practically identical, although the cost associated to each solution differs being the polynomial interpolation time the double of linear interpolation. For this reason, we decided to simply interpolate across two consecutive samples. The penultimate operation is the application of a window function which is multiplied with the data from each channel in order to reduce mainly the level of sidelobes.
- **Sum**. The final step in the ultrasonic generation process is to obtain the accumulated sum of all the signals samples which contribute to a given spatial point.

3.5.1. Parallel implementation

The delay-and-sum process is applied to the complex signals obtained in the previous stage. We have identified different strategies to implement the ultrasonic image generation process in a GPU depending on how the algorithm is parallelized with respect to threads and blocks and relative to the use of GPU resources.

As Figure 16 shows, the parallelization is carried out by launching a thread per image pixel. To this end, a computation grid ($GRID_1$) with $BX = \lceil \frac{R_H}{T_{BX}} \rceil$ and $BY = \lceil \frac{R_V}{T_{BY}} \rceil$ blocks of $T_{BX} \times T_{BY}$ threads is defined on the kernel, where R_H and R_V are the desired image resolution in horizontal and vertical directions, respectively. Each thread is responsible then for calculating the coordinates for the spatial point (x_p, z_p) of a specific image pixel and calculating the lens to focus at this point.

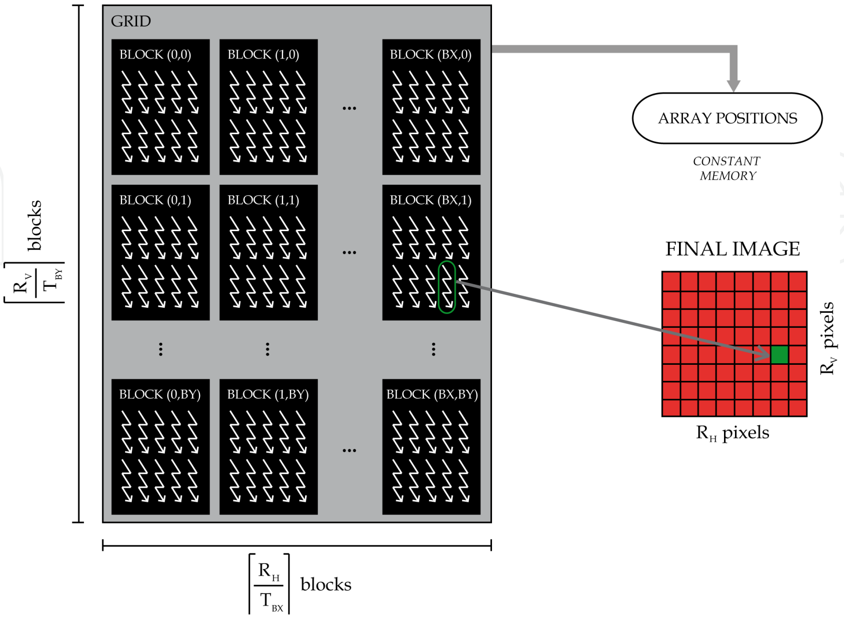

Figure 16. One thread is responsible for a image pixel

In this case, the lens is formed by the $2N-1$ times of flight of each emission-reception pair combination. Thus, in order to accelerate all these calculations, the transducer elements coordinates are stored in constant memory in each GPU multiprocessor. In addition, the computed distances from an array element to an image pixel are reused to save time avoiding duplicate calculations. The lens obtained allow us to index in the complex signals stored in texture memory, and real and imaginary parts are interpolated when needed. To this respect, lineal interpolation was implemented obtaining good performance. Then, the $2N-1$ resultant complex samples are multiplied by the corresponding apodization gains and added together. Finally, the resultant image (final image in Figure 15) is also stored in texture memory, for a quick data access to the post-processing stage.

3.6. Post-processing

The post-processing stage involves firstly calculate the envelope (in essence the modulus) of the beamformed images, according to the following expression:

$$A = \sqrt{{A_I}^2 + {A_Q}^2} \tag{12}$$

where A_I and A_Q are the *In-phase* and *Quadrature* images derived from the beamforming process. This operation prevents the appearance of diverse artefacts associated with the Hilbert Transform.

Likewise, (an optional) stage in the process is in charge of normalizing and converting the image to decibels scale. Although this is not a complex task, it cannot be carried out in the previous stage because we need to know what the maximum value for the image:

$$A|_{decibels} = 20\log_{10}\left(\frac{A}{max(A)}\right) \tag{13}$$

Finally, the generated ultrasonic image (Final image (dB) referenced in Figure 15) is directly displayed on the screen using the OpenGL libraries, which provide specific functionality for graphics representation.

3.6.1. Parallel implementation

The parallel implementation of the envelope calculation is carried out inside the beamforming kernel. This is because at the end of the pixel calculation, we have the final output values for both I and Q components. Thus, we avoid writing twice and we only obtain a single image. For the optional conversion to decibels scale, a new kernel (Kernel 3 in Figure 15) is defined which uses a grid with $\lceil \frac{R_H}{B_x} \rceil$ and $\lceil \frac{R_V}{B_y} \rceil$ blocks of T_{BX} threads having a thread per image pixel as before.

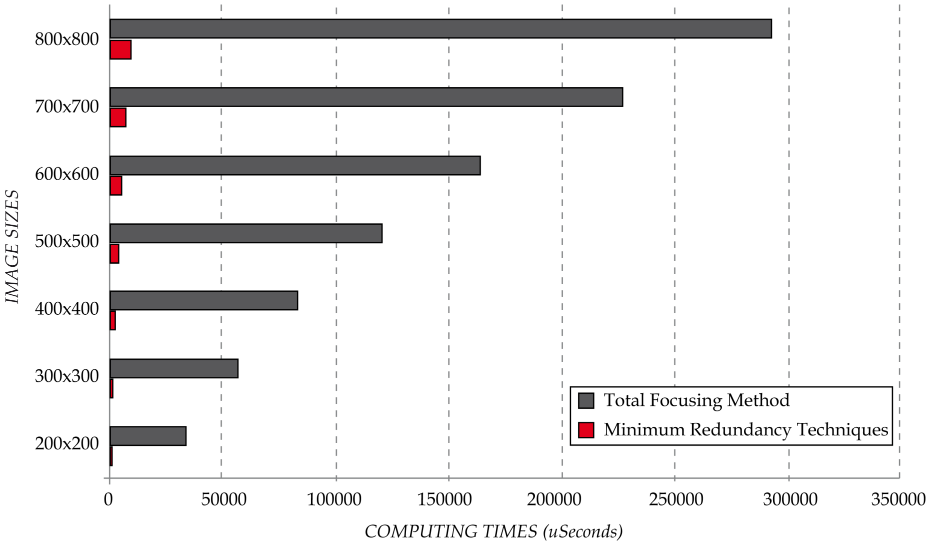

Figure 17. Computing times for TFM and MR solutions using GPU in μseconds

3.7. Performance

A NVIDIA Quadro 4000 graphics card was used to test the beamforming time achieved with the system proposed here. This card has 256 cores and 1GB global memory. It was installed in a computer with a four-core 2.66GHz Intel Q9450 processor and 4GB RAM. GPU-based implementation of the beamformer was done and tested for all acquisition strategies exposed along this chapter. In Figure 17 computing times considering image sizes starting from 200×200 to big size 800×800 for both TFM and minimum redundancy solutions are presented where it is evident than despite using the great power of GPU's the TFM solution is a very intensive procedure.

In Figure 18, the frame rate obtained for different image sizes when 2R-SAFT and kA-SAFT are employed is presented. In particular, attending to the case of an image with 500×500

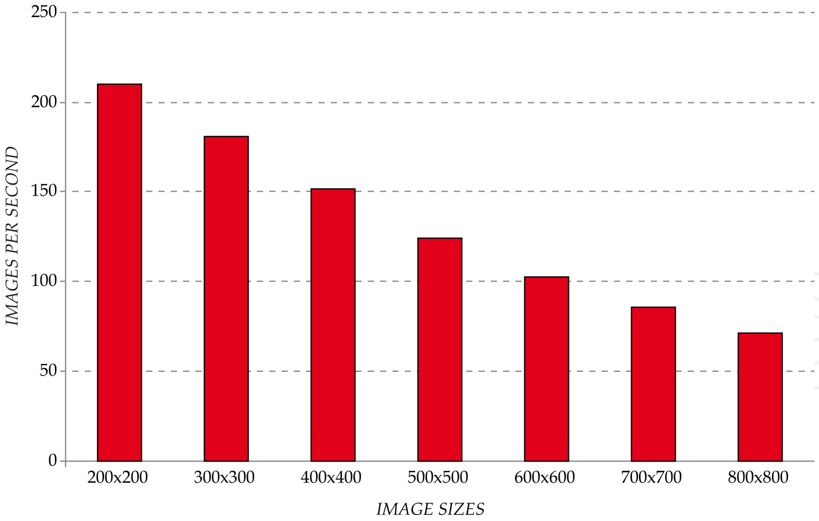

Figure 18. Images per second achieved using GPU for different image sizes for 2R-SAFT and kA-SAFT

pixels, the GPU is able to get 135 images per second, which is in nearly to the acquisition system's rate. The evidence here is that we are using a smaller dataset than that obtained with TFM method but preserving the image quality with all GPU cores completely dedicated to fast computation.

4. Conclusion and future developments

This work has presented how the use of coarray paradigm makes possible the design of ultrasonic imaging systems with reduced hardware requirements. The system is divided into two subsystems, hardware and software respectively. The first one is focused on the development of the data acquisition system, whose design is done analysing the compromise between parallel electronic resources and acquisition time. The second one exploits GPU technology to implement the beamformer via software, compensating the emission and reception distances to each image point, providing the maximum possible quality at each image pixel.

Two solutions, based on the availability of instrumentation in the market, are presented attending to this design following the minimum redundancy coarray model. In one case, it has been emphasized the miniaturization of the hardware (with only two channels in reception), and in the second case the focus has been the reduction of the acquisition time at the expense of increasing and parallelize reception channels (up to a maximum of 32). From the point of view of image quality, both beamforming techniques present similar results. Consequently it is possible to adapt the design of our system to several implementation models depending on the final application requirements.

The problems associated to the low level of the transmitted signals and the signal losses through the material have been analysed. As a solution, we have introduced pulse compression techniques in order to increase the signal to noise ratio. In addition, we have studied the implementation cost of this technique and it has been compared with the TFM technique (based on the FMA capture), verifying that the results are very similar.

Finally, we have made a detailed description of the beamforming process in GPU and it has been quantified the advantage of using the GPU as a processing tool from the image frame rate point of view. So by using a simple graphics card equipped with NVIDIA CUDA technology, rates that go up to 200 images per second were obtained depending on the image size chosen. Therefore, this solution allows the development of high quality imaging systems with low requirements and excellent capabilities in a compact architecture.

Acknowledgments

This work has been supported by the Spanish Ministry of Science and Competitiveness under the project DPI2010-19376.

Author details

D. Romero-Laorden, J. Villazón-Terrazas,
O. Martínez-Graullera and A. Ibáñez

Centro de Acústica Aplicada y Evaluación No Destructiva (CSIC), Madrid, Spain

References

[1] Caroline Holmes, Bruce W. Drinkwater, and Paul D. Wilcox. Post-processing of the full matrix of ultrasonic transmit–receive array data for non-destructive evaluation. *NDT & E International*, 38(8):701–711, December 2005.

[2] Caroline Holmes, Bruce W. Drinkwater, and Paul D. Wilcox. Advanced post-processing for scanned ultrasonic arrays: application to defect detection and classification in non-destructive evaluation. *Ultrasonics*, 48(6-7):636–42, November 2008.

[3] Alan J. Hunter, Bruce W. Drinkwater, and Paul D. Wilcox. The wavenumber algorithm for full-matrix imaging using an ultrasonic array. *IEEE transactions on ultrasonics, ferroelectrics, and frequency control*, 55(11):2450–62, November 2008.

[4] Mustafa Karaman, Pai-Chi Li, and Matthew O'Donnell. Synthetic aperture imaging for small scale systems. *IEEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control*, 42(3):429–442, May 1995.

[5] Jørgen Arendt Jensen, Svetoslav Ivanov Nikolov, Kim Lø kke Gammelmark, and Morten Hø gholm Pedersen. Synthetic aperture ultrasound imaging. *Ultrasonics*, 44:5–15, December 2006.

[6] Carlos J. Martín-Arguedas, O. Martínez-Graullera, G. Godoy, and L. Gómez-Ullate. Coarray synthesis based on polynomial decomposition. *IEEE transactions on image processing*, 19(4):1102–1107, 2010.

[7] Carlos J. Martín-Arguedas, D. Romero-Laorden, O. Martínez-Graullera, M. Pérez-Lopez, and L. Gómez-Ullate. An Ultrasonic Imaging System Based on a New SAFT Approach and a GPU Beamformer. *IEEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control*, 59(7):1402–1412, 2012.

[8] Geoffrey R Lockwood, Pai-chi Li, Matthew O'Donnell, and F. Stuart Foster. Optimizing the Radiation Pattern of Sparse Periodic Linear Arrays. *IEEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control*, 43(1):7–14, 1996.

[9] Svetoslav Ivanov Nikolov. *Synthetic aperture tissue and flow ultrasound imaging*. PhD thesis, Technical University of Denmark, 2001.

[10] Bernard D. Steinberg. *Principles of Aperture and Array System Design*. Wiley, New York, 1976.

[11] John J. Flaherty, Kenneth R. Erikson, and Van Metre Lund. Synthetic aperture ultrasonic imaging systems, Patent number 3548642, 1967.

[12] Christoph B. Burckhardt, Pierre-André Grandchamp, and Heinz Hoffman. An Experimental 2 MHz Synthetic Aperture Sonar System Intended for Medical Use. *IEEE Transactions on Sonics and Ultrasonics*, 21(1):1–6, 1974.

[13] Intel Corporation. Thunderbolt™ technology. http://www.intel.com/thunderbolt, (accessed 6 July 2012).

[14] Carlos J. Martín-Arguedas, O. Martínez-Graullera, and L. G. Reduction of grating lobes in SAFT Images. In *IEEE International Ultrasonics Symposium*, number 1, pages 721–724, Beijing, China, 2008.

[15] Carlos J. Martín-Arguedas. *Técnicas de apertura sintética para la generación de imagen ultrasónica*. PhD thesis, Universidad de Alcalá, 2010.

[16] M. Golay. Golay's complementary series. *IRE Transactions on Information Theory*, pages 273–276, 1961.

[17] Andrzej Nowicki, Igor Trots, Wojciech Secomski, and Jerzy Litniewski. Golay's codes sequences in ultrasonography. *Archives of Acoustics*, 28:313–324, 2003.

[18] National Instruments. 32-channel digitizer module for ultrasound applications. http://sine.ni.com/nips/cds/view/p/lang/en/nid/208657, (accessed 1 October 2012).

[19] Wen-Mei W. Hwu and David B. Kirk. *Programming Massively Parallel Processors : A Hands-on Approach*. Morgan Kaufmann, 2010.

[20] NVIDIA Developer Zone. Software development kit 4.2 version, https://developer.nvidia.com/cuda-education-training, (accessed 1 October 2012).

[21] Jason Sanders and Edward Kandrot. *CUDA by Example*. Addison-Wesley, 2010.

[22] A. V. Oppenheim and W. R. Schafer. *Discrete-Time Signal Processing*. Prentice-Hall, Englewoods Cliffs (NJ), 1989.

[23] S. Lawrence Marple. Computing the Discrete-Time "Analytic" Signal via FFT. *IEEE Transactions on Signal Processing*, 47(9):2600–2603, 1999.