Tackling the Bottleneck of Delay Tables in 3D Ultrasound Imaging

A. Ibrahim*, P. Hager†, A. Bartolini†, F. Angiolini*, M. Arditi‡, L. Benini† and G. De Micheli*
*LSI, EPFL, Switzerland; {aya.ibrahim, federico.angiolini, giovanni.demicheli}@epfl.ch
†IIS, ETHZ, Switzerland; {pascal.hager, andrea.bartolini, luca.benini}@iis.ee.ethz.ch
‡EPFL, Switzerland

***Abstract*—3D ultrasound imaging is quickly becoming a reference technique for high-quality, accurate, expressive diagnostic medical imaging. Unfortunately, its computation requirements are huge and, today, demand expensive, power-hungry, bulky processing resources. A key bottleneck is the receive beamforming operation, which requires the application of many permutations of fine-grained delays among the digitized received echoes. To apply these delays in the digital domain, in principle large tables (billions of coefficients) are needed, and the access bandwidth to these tables can reach multiple TB/s, meaning that their storage both on-chip and off-chip is impractical. However, smarter implementations of the delay generation function, including forgoing the tables altogether, are possible. In this paper we explore efficient strategies to compute the delay function that controls the reconstruction of the image, and present a feasibility analysis for an FPGA platform.**

I. Introduction

Ultrasound imaging is transitioning towards a new generation of 3D-capable machines, which improve medical diagnostic capabilities [1] by reconstructing a volume per frame - as opposed to the more commonly known 2D images, usually along a sagittal or transverse plane across a patient's body. Notable products include the Voluson E8 Expert system by GE Healthcare [2], for obstetrics, and the iE33 xMATRIX Echocardiography System by Royal Philips N.V. [3] for cardiology. To reconstruct 3D images, these devices must use probes with matrix transducers rather than the usual linear array transducers used for 2D imaging. A matrix transducer comprises a very large number of vibrating elements in a grid, for example 9212 in [4], squaring the typical amount of an array transducer for 2D imaging.

To generate an image, first of all the subject is insonified by transmitting a sound wave from the ultrasound probe into the body. At this point, it is necessary to sample at high frequency (tens of MHz) the amplitude of the acoustic echoes received by each transducer element. These echo signals must then be summed together according to *delay laws* in order to reconstruct the image. This process is called *beamforming*. The overall two-way sound propagation time is sub-millisecond, and is a function of the desired penetration depth. Therefore, in principle, multi-kHz frame rates are possible, and have indeed been demonstrated in 2D imaging [5].

The main challenge of 3D ultrasound imaging is the huge bandwidth needed for data processing, since the number of probe elements is about two orders of magnitude higher. Therefore the frame rates achievable in practice can only reach tens of Hz [2], and commercial products are correspondingly expensive and bulky, as are research prototypes like the SARUS [6] system, running on 320 FPGAs.

In this paper, we focus on one specific part of the beamformer design, i.e. the *delay law computation*, which is one of its key components. As shown in Section II, without special design techniques, delay tables of many billions of elements are necessary, which is obviously challenging both in terms of on-chip embedded memory requirements and off-chip memory bandwidth. Our goal is to investigate circuit architectures that bring this design block into the realm of feasibility within single-chip implementations. The main contribution of this work is the optimization of the delay computation architecture, as well as a quantification of its cost, in view of a single-chip 3D beamformer. The architecture's target is flexible and its realization can be either on FPGA or ASIC, but within this paper we will optimize it for an FPGA computation platform.

We will explore two alternative methods to compute delay values. First, we will revisit an architecture originally presented in [7], where all delays are computed on-the-fly with an optimized circuit. This architecture was originally developed with an ASIC target in mind, and in this paper we will assess its implementation on an FPGA target. Next, we will show an approach which relies on a much smaller precomputed delay table, fit for in-FPGA storage, compounded by a small circuit that completes the task. Both methods will be shown to meet the three key challenges of delay computation: accuracy, compactness, and throughput.

II. Delay Calculation in 3D Ultrasound Systems

Ultrasound imaging systems can apply focusing at transmit time, at receive time, or both. Transmit-time focusing consists of exciting transducer elements with such a timing that the sound field in front of the transducer has specific intensity maxima, corresponding to constructive interference areas, while the rest of the field is insonified less intensely. This approach maximizes the image resolution achievable in the constructive interference regions, while unfortunately negatively impacting the resolution elsewhere. An "unfocused" wave can also be emitted, whereby each element is excited at the same time (generating a plane wave) or the excitation profile is such that the overall acoustic wave seems to have been emitted by a "virtual source" behind the transducer. These approaches insonify the whole volume in front of the probe, allowing for a more sophisticated reconstruction of the whole 2D or 3D frame.

Receive-time focusing refers to a computational approach applied to the received echoes, back-scattered within the insonified volume. The amplitude of the echoes received by the probe elements is first sampled. The samples from multiple elements are then summed according to a delay profile that models the propagation time necessary to travel from a scatterer (a reflective point in space) back towards each element of the probe. Contrary to transmit focus, receive focus can be applied dynamically: given enough computing power, it is possible to focus on all points in the volume, using a dedicated delay profile for each point. Modern ultrasound systems all use this technique. The crucial requirement is being able to identify, at runtime, what echo samples should be summed to focus. The problem's core can be expressed as follows:

$$s(S) = \sum_{D=1}^{N} w(S)e(D, t_p(O, S, D)), \forall S \in V \qquad (1)$$

978-3-9815370-4-8/DATE15/©2015 EDAA

S is a point in the volume of interest V; the calculation must be repeated $\forall S \in V$. The outcome $s(S)$ is a signal that follows the reflectivity of scatterers at location S, and will eventually be used to calculate the brightness of the corresponding image pixel. N is the number of receive elements accessible in the probe, w is a weighting coefficient beyond this paper's scope expressing apodization and attenuation compensation [8], and e is the amplitude of the echo received by element $D \in 1, \cdots N$ at the time sample t_p. The value of t_p represents the propagation delay that sound waves incur from a given emission reference O, to the point S, and back to the probe's destination element D. This can be formalized as:

$$t_p(O, S, D) = \frac{|\vec{r_S} - \vec{r_O}| + |\vec{r_S} - \vec{r_D}|}{c} \quad (2)$$

where c is the speed of sound in the medium, and $\vec{r_O}, \vec{r_S}, \vec{r_D}$ express those points' coordinates. This formulation holds for both 2D and 3D ultrasound imaging. We refer the reader to Table I to summarize the specifications of our target system.

In the following, we will discuss three key challenges related to the calculation of propagation delays: accuracy, compactness of storage, and throughput.

A. Impact of Delay Calculation on Beamforming

The accuracy of delay calculation is essential for high-resolution beamforming, because the latter relies on fine-grained time differences to locate the position of body features. Any imprecision may result in poor focus, image artifacts, aliasing, etc.

The traditional beamforming approach reconstructs images by *scanlines*. An alternate approach, that optimizes the consumption of the data coming from the probe elements and minimizes table walking, reconstructs the volume one *nappe* [9], i.e. one surface with constant distance from the origin, at a time. Please refer to Algorithm 1 and Figure 1. The core of both approaches involves the same basic calculation, and requires the same delay coefficients. Image quality will be the same regardless of how delays are obtained at runtime, so long as delays are equally accurate. In practice, however, different delay calculation architectures may be generating values at a faster rate when aimed at a particular order of processing; co-designing the delay computation logic and the rest of the beamformer is beneficial. In this paper, we will focus mostly on the nappe-by-nappe technique, while pointing out where inefficiencies could arise if paired with a scanline-by-scanline beamformer.

B. Size of Required Delay Tables

Since t_p is used as an index into e, and e is usually sampled at a frequency that is a multiple of the probe's center frequency

TABLE I
SYSTEM SPECIFICATIONS

Parameter	Symbol	Value
Physical		
Speed of sound in tissue	c	1540 m/s
Transducer Head		
Transducer center frequency	f_c	4 MHz
Transducer bandwidth	B	4 MHz
Transducer matrix size	$e_x \times e_y$	100×100
Wavelength	λ	c/f_c = 0.385 mm
Transducer pitch		$\lambda/2$
Transducer matrix dimensions	d	50λ = 19.25 mm
Beamformer		
Imaging Volume ($\theta \times \phi \times d_p$)		73°×73°×500λ
Sampling Frequency	f_s	32 MHz
Focal Points		$128 \times 128 \times 1000$

```
// Scanline-by-scanline
for θ: −θ_max to θ_max do
    for φ: −φ_max to φ_max do
        for d: 0 to d_p do
            Beamform(θ, φ, d);
        end
    end
end
// Nappe-by-nappe
for d: 0 to d_p do
    for θ: −θ_max to θ_max do
        for φ: −φ_max to φ_max do
            Beamform(θ, φ, d);
        end
    end
end
```

Algorithm 1: Pseudocode of beamforming algorithm in two equivalent flavours.

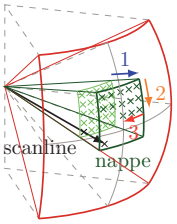


Fig. 1. Focal point calculation order in a nappe-oriented beamformer: first a depth is chosen, then the points at that depth are calculated. A scanline also shown for comparison; a scanline-oriented beamformer will reconstruct points along the whole scanline, then move to the next.

(32 MHz in our example), t_p should be calculated with a very fine grain of about 30 ns. Moreover, the values of t_p must be calculated for each receiving element D and for each point in space S. This results in many delay values having to be calculated. A typical 2D system may require a few million coefficients, which may be stored in a pre-computed table. However, in a 3D system as discussed in this paper, to reconstruct a $128 \times 128 \times 1000$ voxel volume, given a 100×100-element transducer, the theoretical number of delay values to be calculated is about 164×10^9. Even by exploiting symmetry, this is obviously both impractical to pre-compute, due to the storage requirements, and to calculate in realtime.

C. Access Bandwidth to Delay Tables

Another challenge is that delay values need to be available with high throughput, in order to achieve realtime beamforming. The coefficients must be accessed once per frame; a 3D image requires about 2.5×10^{12} delay values/s for reconstruction at 15 frames/s. This is obviously well outside of the capabilities of any realistic off-chip memory interface, and must be tackled in a smart way.

III. PREVIOUS WORK

Today's state-of-the-art 3D ultrasound systems perform analog beamforming in the transducer head to reduce the number of channels that are carried along the cable from a few thousands to a few hundred [10], [11]. This is achieved by applying a fixed analog delay to the signals received by groups of transducer elements, and compounding them in a single analog signal [10]; this is called "pre-beamforming" and the output, which exhibits much lower bandwidths, is used for digital beamforming. The ACUSON SC2000 Volume Imaging Ultrasound System computes up to 64 beams in parallel, i.e., up to 160 MFP/s, using analog beamforming [12].

Analog pre-beamforming, however, limits the image quality, since applying a fixed delay profile to each element group

is equivalent to setting a fixed focus for that group. A fully-digital (or even software [13]) beamformer is desirable because it has the capability to dynamically set the focus position during receive. However, it runs into a major computation and bandwidth challenge due to the large amount of input signals to be processed individually. To enable high-resolution, high-frame-rate 3D imaging, multiple scanlines need to be beamformed from a single insonification using parallel receive beamforming [14], [12] or ultra-fast imaging [15].

The problem of how to compute delay coefficients to feed such beamformers at a very high throughput has been recognized as critical. For example, Sonic Millip3De [16] implements ultra-fast imaging for 100×100 transducer elements (of which only 1000 are considered per shot) with a powerful die-stacked package. Its main bottleneck is that it requires an external DRAM memory to store beamforming delay coefficients, and several Gb/s of memory bandwidth. Other works [17], [6], [9] have shown that a feasible alternative is to try to compute all delay coefficients on-the-fly on-chip. Since this computation involves the evaluation of complex functions like square roots, it is mandatory to identify accurate, fast and low-area approximation circuits.

In this paper we explore two alternative schemes to tackle the delay generation problem. We first revisit our previous works [9], [7] with increased emphasis on the delay approximation logic and some accuracy improvements. We also describe an alternative approach, based on storing a small reference delay table that serves as the basis for runtime delay calculation; this design stands out for its moderate FPGA resource occupation. We analyze and compare the merits of these architectures in terms of accuracy and throughput, further assessing how feasible a single-FPGA, high-frame-rate embodiment is.

IV. Delay Calculation at Runtime

A. Working Principle

To remove the need for massive precomputed tables, solving the compactness challenge, delay values can be computed on-the-fly. Based on Equation 2, this is the problem to be solved:

$$t_p(O,S,D) = \frac{\sqrt{\Delta x_{SO}^2 + \Delta y_{SO}^2 + \Delta z_{SO}^2}}{c} + \\ + \frac{\sqrt{\Delta x_{SD}^2 + \Delta y_{SD}^2 + \Delta z_{SD}^2}}{c} \quad (3)$$

The second square root must be solved $\forall D$ (100×100 elements), $\forall S \in V$ ($128 \times 128 \times 1000$ points), and at the given frame rate. This demands massive parallelism, but the hardware cost of a precise square root computation block is unacceptably high for replicating it on this scale. Much work has been devoted to approximating the square root with simpler arithmetic functions like additions and multiplications [17], [16] in view of managing the cost. Note that the first square root is comparatively much less critical since, unlike D, O is fixed, at least within each frame, making this term independent of the number of transceiver elements.

B. Architecture

In [9], [7], we have presented an architecture to compute the two-way propagation delay efficiently and accurately. We refer the reader to those papers for more details and present just a brief summary here (see Figure 2).

The transmit delay can be calculated only once per point S; the implementation cost becomes therefore negligible. For the receive delay, the calculation can also be simplified because Δz_{SD} depends only on S since D has a constant z being on the transducer, while Δx_{SD} and Δy_{SD} depend on both S and D, but with respect to D can be computed only once per row/column of the transducer. Therefore, only two additions and the square root operation have to be evaluated specifically for each D. The architecture relies on a piecewise linear approximation of the square root function; to keep the approximation error below a given δ (chosen to be equal to ± 0.25 delay samples in our case) (Figure 2(a)), we found 70 segments to be needed. The next crucial simplification is to note that the argument of the second square root of Equation 3 only changes very little when the focal points S are computed sequentially, either nappe by nappe or scanline after scanline. The transitions across the approximating segments being gradual, it is not needed to search for the correct piece each time, allowing for large reductions in hardware cost. Figure 2(b) shows that only one multiplier, one adder and a few LUTs are necessary for the task.

To fulfill the throughput demands of 3D ultrasound imaging, it was shown in [7] that this unit must be instantiated once per transducer element (10000 times) with an achievable frame rate of about 1 fps per 20 MHz of operating frequency.

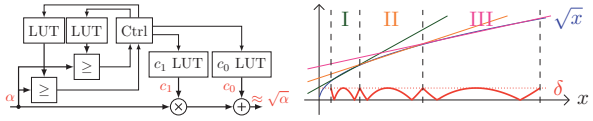


Fig. 2. Calculation of receive delays. Only two additions and one square-root evaluation are needed, (a). The square-root function (blue) shown in (b) is approximated piece-wise linearly such that the error (red, exaggerated) is below a constant δ of choice.

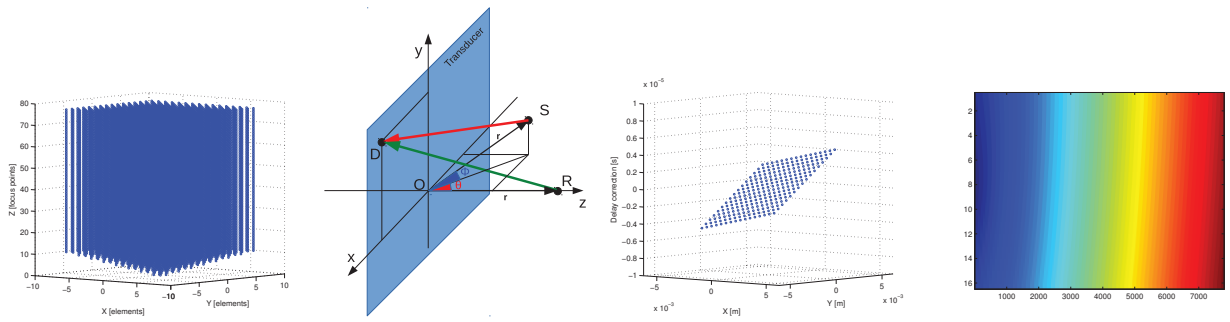
V. Mixed Approach: Delay Tables plus Steering

The technique proposed in the previous Section IV avoids entirely the usage of delay tables, but requires a large number of multipliers instead. An intermediate approach is to keep in working memory a relatively small delay table, and to compute all delay values from this table with very simple mathematical operations. To simplify the computation further, we assume a constant origin O across frames. Techniques like synthetic aperture imaging [15] rely on repositioning O at every insonification; they can be supported by way of multiple precalculated delay tables, at extra hardware cost.

A. Working Principle

The main working principle of this approach can be seen in Figure 3. Let us place the origin O at the center of the transducer, without loss of generality. Considering the set of points along a scanline that coincides with the Z axis, Figure 3(a) represents on XY planes the set of delays that must be stored for a point at that Z depth. The data structure is conceptually a 3D matrix with dimensions $e_x \times e_y \times d_p$, i.e. $100 \times 100 \times 1000 = 10 \times 10^6$ elements. Some table elements are in fact unneeded because probe elements have limited directivity in both emission and reception, and cannot insonify points steeply off-axis. Additionally, the matrix is symmetrical; in the best case, the origin is at the center of the transducer, or anyway vertically aligned with it, and exactly three quarters of the matrix are redundant, thus only $50 \times 50 \times 1000 = 2.5 \times 10^6$ elements need to be stored.

Let us now consider what happens when trying to reconstruct a point S that is off the considered line of sight (Figure 3(b)), i.e. on a steered line of sight. No delay values in the existing table are adequate. However, consider a point



(a) (b) (c) (d)

Fig. 3. (a) Delays must be annotated between each S and each element D of the transducer. The presence of a dot in the figure at (x_D, y_D, z_S) visualizes that a delay value is needed, just for points S on the Z axis. Some elements of the 3D matrix can be pruned because of limited element directivity, as shown. Considering symmetry across the X and Y axes, three quarters of this matrix can be further pruned, not shown here. (For simplicity, the figure shows a $16 \times 16 \times 500$ geometry instead of $100 \times 100 \times 1000$.) (b) For a point on another line of sight, the delay can be computed from the reference delay table plus an angle-dependent offset. (c) When considering both θ and ϕ steering, the required compensation is a plane, whose inclination around the origin is a function of θ, ϕ. (d) A section of the compensated delay table for a given steering angle.

R along the reference scanline, at the same distance from the sound origin ($r := |\vec{RO}| = |\vec{SO}|$):

$$O = (0,0,0); R = (0,0,r); D = (x_D, y_D, 0) \tag{4}$$

$$S = (rcos\phi sin\theta, rsin\phi, rcos\phi cos\theta) \tag{5}$$

The delay for the point S can thus be expressed as a function of the reference delay table for R:

$$t_p(O,S,D) = t_p(O,R,D) + \frac{|\vec{SD}| - |\vec{RD}|}{c} = \\ = \frac{r}{c}\sqrt{1 + \frac{x_D^2 + y_D^2}{r^2} - \frac{2x_D cos\phi sin\theta + 2y_D sin\phi}{r}} + \\ - \frac{r}{c}\sqrt{1 + \frac{x_D^2 + y_D^2}{r^2}} \tag{6}$$

Using the first-order Taylor expansion of the square root, which is equivalent to the far-field assumption $x_D, y_D << r$, this yields:

$$t_p(O,S,D) \approx t_p(O,R,D) - \frac{x_D cos\phi sin\theta + y_D sin\phi}{c} \tag{7}$$

In concrete terms, this is equivalent to keeping a reference 3D delay table in memory, which applies directly to the orthogonal line of sight, and adjusting it with a tilted plane (Figure 3(c)) every time the considered line of sight is steered. It could be said that the whole delay table is itself "steered". This result is known from literature on 2D ultrasound imaging [18], although we are not aware of any work proposing its application for 3D imaging.

The inaccuracy due to the approximation can be theoretically bounded, e.g. with the Lagrange bound. With calculations that are omitted for space reasons, we derived a bound of about $6.7\mu s$ on the round-trip delay, or 214 signal samples at 32 MHz. However, these extreme occurrences are infrequent and located at the edges of the imaging geometry, where they are eliminated by apodization windows. A practical analysis is given in Section VI.

B. Architecture

Based on the analysis above, it is possible to perform 3D imaging with a 2.5×10^6 element delay table in the best case; the table needs to be proportionally larger as the sound origin is displaced from the vertical of the transducer's center. For each of the 128×128 steered lines of sight, the delay table must be summed with the correction coefficients of Equation 7. The latter can be entirely precomputed, for a total of $100 \times 64 \times 128 + 100 \times 128 = 832 \times 10^3$ values (note that $cos\phi$ is symmetrical around 0).

The delay values are used as an index into an echo buffer containing slightly more than 8000 samples, corresponding to a 32 MHz sampling of the two-way sound propagation time ($2 \times 500\lambda$). This requires 13-bit precision. Since a sum of three values is needed to compute the overall delay, the accuracy can be improved by using a fixed-point representation. Let us assume for the moment, without loss of generality, a 18-bit design, which fits well one of Xilinx's selectable BRAM bank widths. The reference delays are always positive, thus they can be stored in 13.5 unsigned format and they can be sign-extended at the moment of applying the correction. The correction coefficients, which may be negative, must be stored with a signed 13.4 representation. Overall, the total storage is $2.5 \times 10^6 \times 18$ bits = 45 Mb plus $832 \times 10^3 \times 18$ bits = 14.3 Mb. This is demanding, but within of the capabilities of high-end FPGAs; for example, the largest Xilinx Virtex 7 carry up to 68 Mb of Block RAMs [19].

To conserve area - either for other beamforming functions, or to be able to choose a smaller FPGA - it is possible to swap in and out of the FPGA portions of the delay table memory from an external DRAM. In other words, the on-FPGA delay table could be a cache of a complete delay table residing off-chip. Since delay table contents are constant during execution, this cache would be read-only, and the required bandwidth would be unidirectional. A *nappe-by-nappe* beamformer accesses a constant-depth slice of the delay table intensively before moving to the next slice; this suggests to divide the delay matrix in an arbitrary number of chunks, serially loaded into the same memory space as nappes are swept. For example, imagine a design that reconstructs the 3D volume in 64 insonifications per volume (256 scanlines/insonification) at 15 Hz, i.e. 960 insonifications/s; the full delay table would need to be fetched 960 times per second, at a total bandwidth of about 5.3 GB/s. This can be achieved with many modern FPGAs. A suitable design point could for example comprise just 128 18-bit BRAM banks (each having 1k lines, for a total of 2.3

Mb) to store a slice of these delay samples; this BRAM could be managed as a circular buffer, loading new delay samples as the old ones have been used, with an ample margin of 1k cycles of latency to fetch new data. Overall, the required on-chip memory would be reduced from 45 Mb plus 14.3 Mb to 2.3 Mb plus 14.3 Mb. In general, it is possible to see that even though a precomputed delay table is used, the implementation can be sufficiently compact to fit in a single FPGA easily.

Assuming a target frame rate of 15 Hz, the required throughput is 2.5×10^{12} delay samples/s calculated in Section II-C. This means that each of the 128 instantiated BRAMs must be able to generate about 100 delay samples per clock to meet specifications at 200 MHz, which is apparently impossible because each BRAM can only provide one delay sample per cycle. However, we propose the architecture shown in Figure 4. It is a *memory-centric* architecture - i.e., the heart of this block is an FPGA BRAM bank, and the block is replicated 128 times. Each such block reads from the BRAM one delay sample per cycle, then applies to it all permutations of 8 x_D and 16 y_D corrections, resulting in delay samples for 128 points of a nappe. This requires $8 + 16 \times 8 = 136$ adders per block, of which 128 must also perform rounding to integer. Collectively, 128 blocks like this, each producing 128 steered delay samples per clock, can reach a peak throughput of 3.3 Tdelays/s at 200 MHz, meeting specifications. For best timing results, the system can be arranged so that each block keeps using the same correction coefficients through each insonification, entirely removing the coefficients from the critical timing path. A control block manages the read and write addresses into the BRAM. To ensure that all BRAMs can operate in parallel, the delay values loaded in each should be staggered rather than consecutive, so that a beamformer trying to fetch delay samples for consecutive nappes can retrieve them from the 128 BRAMs in parallel.

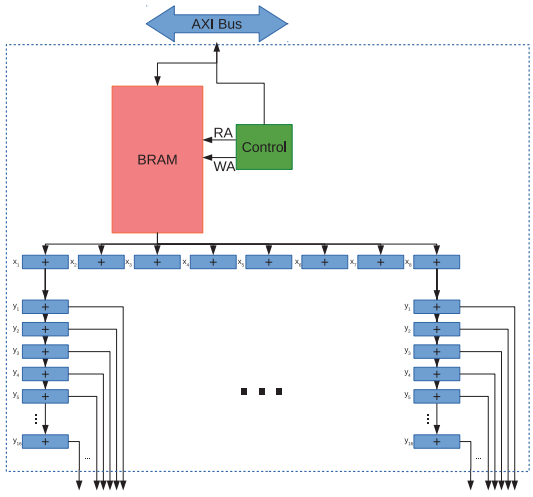


Fig. 4. Proposed architecture of a delay computation block, centered on a BRAM bank.

VI. Experimental Results

In this section, we will present implementation results for the two proposed delay computation architectures. For the sake of brevity, we will refer to the table-free technique as TABLEFREE and to the table-steering technique as TABLESTEER in the subsequent figures and tables. We will first of all comment on how accurate the two methods are, which is the key indicator of the quality of the reconstructed images. We will then present and compare implementation results on a high-end Xilinx Virtex 7 device, XC7VX1140T, speed grade -2, to assess the utilization of resources, and thus ultimately the feasibility of the implementations. We will also evaluate the maximum achievable frequency, and thus the throughput, of the designs, to see how high frame rates can be achieved.

A. Accuracy

The accuracy of the TABLEFREE architecture is entirely controllable by choosing the grain of the piecewise approximation. As a guideline, we chose in [9], [7] to bound the maximum absolute error of the square root approximation such that the error of the delay selection error is ± 1 sample. Setting $\delta = 0.25$, the approximation error can be assumed to be equally distributed in the interval $(-\frac{1}{4}, \frac{1}{4})$. Two square root approximations (Equation 3) are summed up, leading to a mean absolute error of ≈ 0.204 and a maximum absolute error of 0.5. We compared our approximated fixed-point implementation with an exact computation, quantizing both to an integer selection index prior to comparison. We recorded a mean absolute selection error of ≈ 0.2489 and a maximum absolute selection error of 2; the error increase can be explained with fixed-point effects. Note that the average inaccuracy can be arbitrarily reduced with a lower δ and utilizing higher-precision fixed-point computation, at the cost of increasing LUT area and arithmetic circuitry. Conversely, resources can be saved by accepting a less-precise delay calculation.

For what concerns TABLESTEER, the dominant inaccuracy derives from the algorithmic approximations due to using a first-order Taylor polynomial to "steer" the reference delay table. As seen in Section V-A, the theoretical bound on the inaccuracy of the approximation (214 signal samples) is very loose. However, with an exhaustive exploration in the volume of interest, we discovered that the worst inaccuracies are in practice filtered away by apodization, since they occur at angles beyond the elements' directivity. The maximum absolute error we observed in practice was $3.1\mu s$, i.e. 99 signal samples. Furthermore, fortunately, the far-field approximation's worst errors occur only at extremely short distances from the origin and at the extreme angles of the field of view; both regions are usually the least critical for image quality. The average absolute error over the whole volume due to the algorithm itself was a very moderate $44.641ns$, i.e. ≈ 1.4285 signal samples. A second cause of inaccuracy is the fixed-point sum of the original delay value with the two correction coefficients, and subsequent rounding to an integer index to access the data sample array. It can be derived exactly that in all cases, even when storing delay values as 13-bit integers, the maximum difference between the delay value calculated in hardware vs. a high-precision floating-point computation is of ± 1 sample. Matlab simulation on 10×10^6 random input values shows that 33% of the echo samples experience this additional inaccuracy if using 13 bit integers; this fraction is reduced to less than 2% when using a 18-bit (13.5) fixed point representation.

B. FPGA Implementation Results

We report here post-placement results on the FPGA target, achieved with the Vivado 2014.1 software by Xilinx. For TABLEFREE, since a high-performance design would not fit into a single FPGA, we normalize the results so as to present the resource utilization and performance of the largest design point that can still fit in a chip.

For TABLESTEER, we parameterize the design to use 14-bit or 18-bit delay representations (TABLESTEER-14b/-18b), assessing the accuracy vs. area tradeoff. The memory bandwidth is estimated based just on the volume of data to be fetched, but include no other overheads.

Architecture	LUTs	Registers	BRAM	Clock	Offchip DRAM BW (estimated)	Inaccuracy (|off samples|)	Throughput	Frame Rate	Supported Channels
TABLEFREE	100%	23%	0%	167 MHz	none	avg 0.25, max 2	1.67 Tdelays/s	7.8 fps	42×42
TABLESTEER-14b	91%	25%	25%	200 MHz	4.1 GB/s	avg 1.55, max 100	3.3 Tdelays/s	19.7 fps	100×100
TABLESTEER-18b	100%	30%	25%	200 MHz	5.3 GB/s	avg 1.44, max 100	3.3 Tdelays/s	19.7 fps	100×100

TABLE II
VIRTEX 7 XC7VX1140T-2 SYNTHESIS RESULTS

Several things can be observed in this table. First, TABLEFREE was optimized for an ASIC implementation and leans heavily on the usage of logic rather than memories. Without specific optimizations, it requires too many slices for a single-FPGA realization; an ideal design point filling the whole FPGA with delay generation logic would produce a delay throughput sufficient for a transducer with only 42×42 elements. Additionally TABLEFREE is able to run at only half the frequency of its initial ASIC target, limited by the multiplier in the square root approximation, leading to a frame rate of 7.8 fps instead of 15. However, note that at the already at today's 20nm node, 3D-stacked Virtex UltraScale chips [20] feature twice the LUT count of the Virtex 7 family. Thus, we project that, with additional tuning, in the upcoming 16-nm Virtex family, 10-15 frames per second should be possible in a single FPGA with support for 100×100 elements.

TABLEFREE however has some major advantages. It does not occupy any BRAM space, and it does not require any off-FPGA bandwidth because all necessary coefficients are on-chip. This makes it compatible with integration in the same chip of other portions of the beamformer architecure, or of other post-beamforming functionality.

TABLESTEER was optimized from the start for an FPGA implementation, and therefore makes a more balanced use of the resources of the Virtex chip. As a result, it is possible to fit the delay generation logic necessary to achieve a frame rate of almost 20 fps for 3D ultrasound in a single device. By tuning the precision of the fixed-point representation, it is possible to change slightly the accuracy/resource tradeoff.

Of course, a key price to pay is a loss of accuracy in the beamforming process, but mostly limited to the edges of the imaging volume. Another limitation of TABLESTEER is that it requires a significant amount of off-chip bandwidth to load delay values from an external storage. Although this bandwidth can be managed, it may compete with other image processing functions in the complete system. The off-chip traffic can be eliminated only by storing the whole reference delay table on-chip, at a steep BRAM cost. Finally, TABLESTEER is less accommodating toward non-centered sound origins and non-orthogonal wave emission; these are common in synthetic aperture imaging. To support these modes, an off-chip repository of delay tables may be needed.

VII. CONCLUSIONS

We have shown two radically different approaches to calculating delay values for realtime 3D ultrasound beamforming. Both derive from the need to minimize the storage space required by delay tables in a conventional beamformer; the challenge is solved very differently, by eliminating the tables altogether and computing everything on-the-fly in TABLEFREE, and by using a small reference table in TABLESTEER. The two approaches have different targets, namely an ASIC implementation and an FPGA one, respectively. In this paper, we compared both on an FPGA backend; as expected, the FPGA-targeted TABLESTEER architecture proved a better fit to the available resources, but the table-free architecture is projected to also provide good performance in next year's generation of FPGAs. On the other hand, TABLEFREE guarantees better image quality, leaves off-chip memory bandwidth completely available to other portions of the imaging toolchain, and is more flexible in view of advanced imaging modes.

As a next step, we plan on studying both architectures within a full on-FPGA beamformer, evaluating the resource tradeoffs with other portions of the beamformer, and investigating further, low-level optimizations to better fit the FPGA fabric.

ACKNOWLEDGMENTS

The authors would like to acknowledge funding from the Swiss Confederation through the UltrasoundToGo project of the Nano-Tera.ch initiative.

REFERENCES

[1] J. Powers and F. Kremkau, "Medical ultrasound systems," *Interface Focus*, vol. 1, no. 4, pp. 477–489, August 2011.
[2] GE Healthcare, "Voluson E8 expert," www.gehealthcare.com.
[3] Philips Electronics N.V., "iE33 xMATRIX echocardiography system," www.healthcare.philips.com.
[4] ——, "Philips iU22 ultrasound with xMATRIX system specifications," 2012, www.healthcare.philips.com.
[5] M. Tanter and M. Fink, "Ultrafast imaging in biomedical ultrasound," *Ultrasonics, Ferroelectrics, and Frequency Control, IEEE Transactions on*, vol. 61, no. 1, pp. 102–119, January 2014.
[6] J. Jensen, H. Holten-Lund, R. Nilsson, M. Hansen, U. Larsen, R. Domsten, B. Tomov, M. Stuart, S. Nikolov, M. Pihl, Y. Du, J. Rasmussen, and M. Rasmussen, "Sarus: A synthetic aperture real-time ultrasound system," *Ultrasonics, Ferroelectrics, and Frequency Control, IEEE Transactions on*, vol. 60, no. 9, pp. 1838–1852, Sep 2013.
[7] P. A. Hager, P. Vogel, A. Bartolini, and L. Benini, "Assessing the area/power/performance tradeoffs for an integrated fully-digital, large-scale 3D-ultrasound beamformer," in *Proceedings of the 2014 Biomedical Circuits And Systems (BioCAS) Conference*, 2014.
[8] K. Thomenius, "Evolution of ultrasound beamformers," in *Ultrasonics Symposium, 1996. Proceedings., 1996 IEEE*, vol. 2, Nov 1996, pp. 1615–1622 vol.2.
[9] P. Vogel, A. Bartolini, and L. Benini, "Efficient parallel beamforming for 3D ultrasound imaging," in *Proceedings of the 24th Edition of the Great Lakes Symposium on VLSI*, ser. GLSVLSI '14. New York, NY, USA: ACM, 2014, pp. 175–180. [Online]. Available: http://doi.acm.org/10.1145/2591513.2591599
[10] B. Savord and R. Solomon, "Fully sampled matrix transducer for real time 3D ultrasonic imaging," in *Ultrasonics, 2003 IEEE Symposium on*, vol. 1. IEEE, 2003, pp. 945–953.
[11] G. Frey and R. Chiao, "4Z1c real-time volume imaging transducer," *Siemens Healthcare Sector, White Paper*, 2008.
[12] K. Üstüner, "High information rate volumetric ultrasound imaging," *Siemens Healthcare sector, White paper*, 2008.
[13] R. E. Daigle, "Ultrasound imaging system with pixel oriented processing," Oct. 16 2012, US Patent 8,287,456.
[14] T. G. Bjåstad, "High frame rate ultrasound imaging using parallel beamforming," *NTNU PhD Dissertation*, 2009.
[15] J. A. Jensen, S. I. Nikolov, K. L. Gammelmark, and M. H. Pedersen, "Synthetic aperture ultrasound imaging," *Ultrasonics*, vol. 44, Supplement, no. 0, pp. e5 – e15, 2006, proceedings of Ultrasonics International (UI05) and World Congress on Ultrasonics (WCU). [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0041624X06003374
[16] R. Sampson, M. Yang, S. Wei, C. Chakrabarti, and T. Wenisch, "Sonic millip3de: A massively parallel 3d-stacked accelerator for 3d ultrasound," in *High Performance Computer Architecture (HPCA2013), 2013 IEEE 19th International Symposium on*, Feb 2013, pp. 318–329.
[17] S. I. Nikolov, J. A. Jensen, and B. Tomov, "Recursive delay calculation unit for parametric beamformer," in *Medical Imaging*. International Society for Optics and Photonics, 2006, pp. 61 470D–61 470D.
[18] S. Holm, "Digital beamforming in ultrasound imaging," 1994.
[19] Xilinx Inc., "Virtex-7 FPGA family," 2014, www.xilinx.com/products/silicon-devices/fpga/virtex-7.html.
[20] ——, "Ultrascale architecture," 2014, www.xilinx.com/products/technology/ultrascale.html.