Towards a real-time continuous ultrafast ultrasound beamformer with programmable logic

Zhengchang Kou, Qi You, Jihun Kim, Zhijie Dong, Matthew R. Lowerison, Nathiya V. Chandra Sekaran, Daniel A. Llano

Pengfei Song, Senior Member, IEEE, Michael L. Oelze, Senior Member, IEEE

Abstract—Ultrafast ultrasound imaging is essential for advanced ultrasound imaging techniques such as ultrasound localization microscopy (ULM) and functional ultrasound (fUS). Current ultrafast ultrasound imaging is challenged by the ultrahigh data bandwidth associated with the radio frequency (RF) signal, and by the latency of the computationally expensive beamforming process. As such, continuous ultrafast data acquisition and beamforming remain elusive with existing software beamformers based on CPUs or GPUs. To address these challenges, the proposed work introduces a hybrid solution composed of an improved delay and sum (DAS) algorithm with high hardware efficiency and an ultrafast beamformer based on the field programmable gate array (FPGA). Our proposed method presents two unique advantages over conventional FPGA-based beamformers: 1) high scalability that allows fast adaptation to different FPGA platforms; 2) high adaptability to different imaging probes and applications thanks to the absence of hard-coded imaging parameters. With the proposed method, we measured an ultrafast beamforming frame rate of over 3.38 GPixels/second. The performance of the proposed beamformer was compared with the software beamformer on the Verasonics Vantage system for both phantom imaging and in vivo imaging of a mouse brain. Multiple imaging schemes including B-mode, power Doppler and ULM were evaluated with the proposed solution.

Index Terms—Beamforming, FPGA, parallelization, real-time, ultrafast ultrasound, super-resolution imaging.

I. INTRODUCTION

ULTRAFAST ultrasound [1],[2] has been the driving force for many new ultrasound imaging applications such as shear wave elastography[3],[4], super-resolution ultrasound localization microscopy (ULM) [5],[6], and functional ultrasound (fUS) [7]. Traditional line by line or multi-line focused beam scanning cannot provide sufficient frame rates for applications requiring high-speed tracking of tissue motion (e.g., blood flow, shear wave motion, microbubble movement) within a large field-of-view (FOV). Compounding plane wave imaging (CPWI) [8] provides a balance between imaging quality (e.g., spatial resolution, signal to noise ratio (SNR)) and imaging frame rate and is widely used in applications that require ultrafast ultrasound.

Ultrafast ultrasound with CPWI is a demanding technology with high data rate and beamforming computational cost. Ultrafast ultrasound typically requires an ultra-high data rate of over 100 Gb/s for a 128-channel system with a 14-bit, 62.5 MHz analog to digital converter (ADC). State-of-the-art ultrafast imagers such as the Verasonics system use CPU-based software beamformers with PCI-Express interface to handle raw radiofrequency (RF) data transfer [9]. However, the beamforming rate of Verasonics’ CPU beamformer is limited to 133 MPixels/second, according to our measurements. For applications such as fUS and ULM, where long data acquisitions with high frame rate are necessary, CPU-based beamforming does not provide adequate online beamforming speed. As a result, users have to choose between recording unbeamformed RF data to local storage (for offline beamforming) with small time gaps between data blocks (i.e., continuous data acquisition within each block) [10], or recording beamformed data with a relatively large time gap between consecutive time blocks [11]. As compared to a CPU, GPU-based beamformers [12] provide significantly improved beamforming speed[13]. However, due to the data transfer overhead, when using a GPU, more than half of the beamforming time is consumed in data transfer from the Verasonics to the host PC. As such, the bandwidth of the PCI-Express interface remains the key barrier for achieving continuous data acquisition at an ultrafast imaging frame rate[14].

To overcome the existing challenges associated with software beamformers, we propose to develop a novel hardware-based beamformer in a field programmable gate array (FPGA), which provides several unique advantages: 1) the FPGA can directly interface with the analog front end (AFE) chips, which makes the data transfer overhead negligible between the ultrasound data acquisition module and the...
beamformer; 2) the FPGA supports massive parallel beamforming with much higher computational performance and lower power consumption thanks to its fully programmable memory and computational architecture; 3) the beamformed data from the FPGA have much lower data rate than the raw, unbeamformed RF data, which minimizes the data transfer overhead between the beamformer and PC (for downstream post-processing). Therefore, the FPGA provides an enticing solution for continuous recording and beamforming of ultrafast ultrasound data with a speed that is only limited by the speed of sound propagation in soft tissue.

Some existing ultrasound systems, such as the ULA-OP 256, are FPGA-based[15],[16]. In their design, FPGAs are directly connected to AFEs for beamforming. Groups of 32 elements are connected to one FPGA beamformer, which has four DAS units inside. Each DAS unit individually beamforms one line of an image. A continuous beamformed pulse repetition frequency (PRF) of 3,800 Hz was achieved with this system. The beamformer could produce up to 467 MPixels/s which is much higher than 133 MPixels/s achieved by the Verasonics CPU beamformer. Even though the frame rates achieved by FPGA-based beamformers [16], are much higher than CPU and GPU beamformer, the achievable continuous frame rate is still limited and cannot match with the front-end acquisition frame rate (e.g., 10,000 Hz). The reason for this, is that traditional FPGA-based beamformers do not fully exploit parallelization that can be realized on an FPGA. In the ULA-OP, the delay-and-sum (DAS) profiles are calculated in run-time because they are too large to save in the FPGA’s local buffer. According to this runtime generated profile, raw RF data are read from a local buffer. This non-sequential memory access limits the parallelization of local memory resources and further limits utilization of the digital signal processor (DSP) cores. Only a small portion of the thousands of DSP cores integrated in an FPGA are utilized in the ULA-OP and the large internal memory bandwidth of the FPGA is typically underutilized. These factors limit the output frame rates of current FPGA beamformers. A more recent study utilized 8 Xilinx Kintex 7 FPGAs for parallel beamforming which achieved a beamforming rate up to 917 MPixels/s for a 64-channel system [17]. However, the subaperture size for each scan line was only 8 elements, which greatly limits its lateral resolution.

In this paper, we propose a novel FPGA-based hardware beamformer with an optimized DAS profile calculation algorithm that enables ultrafast beamforming for CPWI. By using delay profile reuse, run-time delay calculation is no longer needed. As the same delay profile is used for all the pixels at the same depth, the proposed beamformer could beamform one row of an image simultaneously. Further, memory access for raw RF data is in a 2,048 bits wide vector scheme to increase memory utilization efficiency. Multiple identical buffers are used to provide over 4 Tbps internal memory bandwidth. Over 900 DSP slices inside the FPGA are utilized to provide 233 GMACs computational capabilities.

This article is organized as follows. In Section II, we describe the revised DAS algorithm for parallelized implementation on the FPGA. The experimental setup is explained in Section III, followed by results in Section IV. We finalize the paper with discussion and conclusions.

II. METHODS

A. Delay and Sum (DAS) beamforming

The underlying principle of DAS is to achieve receive focusing by compensating for both transmit and receive delay between each transducer element and an imaging pixel. The traditional DAS is illustrated in Fig.1.

The transmit delay of a plane wave at steering angle $\theta$ to reach a point at $(x,z)$ is

$$\tau_{tx}(\theta,x,z) = \frac{z \cdot \cos \theta + x \cdot \sin \theta}{c}, \quad (1)$$

where $c$ is the speed of sound. The receive delay, which corresponds to the time required by the signal reflected from $(x,z)$ to reach a receive element at $x_n = (n - 1) \cdot d$, is

$$\tau_{rx}(x_n, x, z) = \frac{\sqrt{z^2 + (x_n - x)^2}}{c}, \quad (2)$$

where $n$ is the index of the element and $d$ is the element spacing. From Eqs. 1 and 2 we can observe that the total delay is dependent on both the lateral and axial positions of the target and the lateral position of the receive element. To beamform one frame, we need to calculate the delay matrix $\tau_{rx}$ of D*W*F, where D is the total number samples along the range direction of the beamformed image, W is the total number of samples along the width of the image, and F is the aperture size (i.e., number of elements) used to beamform each scan line.

The bottleneck of efficient beamforming parallelization arises from unstructured RF data retrieval that is dictated by the varying delay profiles of each imaging pixel. During conventional DAS beamforming, reading of the RF sample is fully random memory access because the delay calculation is non-linear. This randomness reduces the improvement of memory read efficiency because of non-sequential reading.

B. Revised DAS Delay

To improve the parallelization efficiency, we first reduce the dimensions of receive delay profile matrix $\tau_{rx}$ from three to
two by replacing \( x_n - x \) by \( \Delta x \), which represents the lateral distance between the target and the receive element. Then, the revised receive delay is

\[
\tau'_{rx}(\Delta x, z) = \sqrt{\frac{\Delta x^2 + z^2}{c}}.
\]

In this way, we do not calculate the delay profile according to both the lateral and axial position of the target and the lateral position of the receive element. Instead, we calculate the delay profile according to the lateral distance between the target and the receive element and the axial position of target.

Following the same principle, the transmit delay is revised to:

\[
\tau'_{tx}(\theta, x_n, \Delta x, z) = \frac{z \cdot \cos \theta + (x_n - \Delta x) \cdot \sin \theta}{f_s}.
\]

To remove \( x_n \) from the transmit delay calculation, we can zero-pad at the beginning of each element’s RF data with the number of zeros calculated from Eq. 5

\[
\frac{(N - 1) \cdot d - x_n}{\cos \theta \cdot f_s},
\]

where \( f_s \) is the RF sampling frequency. The extra delay caused by these padded zeros can combined with \( x_n \) \cdot \sin \theta \) in Eq. (4) to form a constant, \( (N - 1) \cdot d \), where \( N \) is the total number of receive elements. We can then rewrite Eq. 4 as

\[
\tau''_{tx}(\theta, \Delta x, z) = \frac{z \cdot \cos \theta + ((N - 1) \cdot d - \Delta x) \cdot \sin \theta}{c}.
\]

Equations 3 and 6 can be combined to write the total delay as

\[
\tau_{total}(\theta, \Delta x, z) = \tau'_{tx}(\theta, \Delta x, z) + \tau'_{rx}(\Delta x, z).
\]

In Eq. 7, we can observe that now the delay is only dependent on the steering angle \( \theta \), relative lateral distance \( \Delta x \), and depth \( z \). By using Eq. 7, we only need to calculate a 2D delay profile \((D^*F)\) matrix for each steering angle instead of a 3D delay profile \((D^*W^*F)\). Moreover, Eq. 7 enables all the targets at the same depth to use the same delay profile because there are no absolute positions in the delay profile calculations. Given a fixed steering angle and depth, the only variable is the lateral distance between the target and the receive elements.

**C. Parallelization**

Parallelization occurs in two parts: memory parallelization and DAS parallelization. To be specific, given fixed \( \theta, \Delta x \) and \( z \), we can read one delay value from the delay profile matrix. This single delay value provides the depth of the raw RF data sample needed for a fixed depth and fixed lateral distance between a target and a receive element. This delay value is the same for all the targets or pixels at the same depth. For example, consider pixel 1 and pixel 2 are two point targets separated by one pitch laterally as shown in Fig. 2. They share the same delay profile, which is determined by their depth. In the subapertures that are used to beamform pixel 1 and pixel 2, the leftmost raw RF samples are located at the same depth which is pointed by the position 1 with dark red background in the delay profile matrix in Fig. 2. Therefore, we could use this delay value as a pointer to a row of raw RF data samples and read one row of raw RF data samples at one time, instead of reading a single raw RF data sample at one time. By doing this, we can beamform a row of scan line samples simultaneously instead of a single scan line sample. To further parallelize, we can fetch all the delay values corresponding to a single depth from the delay profile matrix and read multiple rows of raw RF data samples at the same time according to those delay values. For example, we could read the pointers from position 1 to position F which is the subaperture size from the delay profile matrix in Fig. 2. Then, F rows of RF data can be read according to these F pointers, as the top left of Fig. 2 illustrates.

The second part is DAS parallelization. To be specific, after we read all these rows of raw RF data samples, we can stack them together following the ascending order of \( \Delta x \) to form a stacked raw RF data matrix which is shown on the top right of Fig. 2. From Eq. 7, we can observe that given a fixed steering angle and depth, \( \Delta x \) is the only variable that determines the delay value. Because the matrix is vertically arranged according to \( \Delta x \), if the increment of both \( \Delta x \) and \( x \) are both fixed to \( d \), then the data samples in the first row correspond to location \( n \), and the second row correspond to \( n + 1 \), and so on. From Fig. 2, one can observe that after the stacked RF data matrix is formed, all the RF samples inside the subapertures for pixel 1 and pixel 2 are organized in the diagonal direction. Therefore, we can diagonally sum up the samples in the stacked raw RF data matrix to have all the targets at the same depth beamformed.
images with finer spatial pixel resolution.

By performing these two parallelization operations, we only need to read one row of the delay profile matrix and load rows of raw RF data to the stacked RF data matrix to beamform all targets at one depth, with all of the computation being fully parallel.

D. Implementation

There are several limitations on the hardware implementation of the beamformer to programmable logic. The first limitation is the total internal memory size that is needed to buffer the RF data. As discussed in previous section we need to simultaneously read multiple rows of RF data from the RF data buffers which are implemented by FPGA’s internal memory (e.g., block ram (BRAM)). If we assume each frame of RF data contains 128 channels (channel number in Fig.3) in width, and each channel has 2,560 samples in depth, then each RF buffer’s data size is 5 Mb if the quantization depth is 16 bits (e.g., 128 x 2,560 x 16 bits = 5 Mb). If the subaperture size is 64 (F in Fig. 3), and we want to finish the beamforming in one clock cycle, then we need 64 RF buffers to read 64 rows of RF data simultaneously if we assume each RF buffer provides a single read port (only one row could be read every clock cycle). The total memory size could reach 320 Mb (e.g., 5 Mb x 64 = 320 Mb), which is much larger than the capacity of most current FPGAs.

The second limitation is the total number of DSP cores that perform the multiply-add operation in the FPGA. Basing on the same assumption from the previous paragraph, we need 8,064 (e.g., 128 x (64-1) = 8,064) DSPs to build the diagonal sum part with subaperture size of 64 that could finish the beamforming of 128 pixels (one row of beamformed image) in one clock cycle. Currently, only several of the largest FPGAs can provide such high numbers of DSPs. These large FPGAs are also more expensive.

To address these two challenges and make the implementation practical, two strategies are utilized. First, instead of buffering the whole frame of RF data to the internal buffer, only a portion of the RF data is buffered, based on the fact that each row of the beamformed sample only needs RF data over a limited range of depth which is defined as dependent range (DR), and the largest DR in all the rows of one frame is defined as maximum dependent range (MDR). Obviously MDR is much smaller than the total depth of one frame.

If the sub-aperture size is fixed, the DR is larger for shallower regions than that for deeper regions where the delay profile is flatter, which is described by the derivative of \( \tau_{\text{total}}(\theta, \Delta x, z) \) with respect to \( \Delta x \).
To further reduce MDR and save internal memory resources, the subaperture size is reduced for the shallower regions (close to the probe surface) in a fixed F-number way to maintain a homogenous lateral resolution.

As only a portion of the RF data are needed for the beamforming of each row of image, the size of the RF buffer can be reduced from the total depth to MDR by using cyclic buffers. To be specific, a modulo operator with divisor equal to MDR is added to the address port of RF data buffers.

As a result, the buffer depth which is the same as MDR can be reduced from 2,560 samples to 150 samples with an FPGA clock frequency of 250 MHz. If the FPGA clock frequency is 250 MHz, the ideal frame rate we can achieve is 12,207 FPS.

The beamforming time of one frame is 2,560 * 8 = 20,480 clock cycles. If the FPGA clock frequency is 250 MHz, the ideal frame rate we can achieve is 12,207 FPS.

The RF data input and beamformed data output occur simultaneously to the beamforming process because we pipeline the whole process. The only overhead is the time used to load the delay profile to internal buffers and the time needed for the internal buffers to load MDR rows RF data to start the beamforming.

III. EXPERIMENT SETUP

A. FPGA deployment

The proposed hardware architecture was written by C++ and synthesized to Verilog by Xilinx Vitis High-Level Synthesis (HLS). Xilinx pragmas were used to instruct HLS to apply the parallelization to the C++ code. This workflow enabled fast implementation and verification of the proposed hardware design. The synthesized results were then exported as IP core to Xilinx Vivado tool in which we added the companion parts to support our beamformer IP.

A Xilinx ZCU106 FPGA development board was used as the verification platform. A Xilinx ZU7EV FPGA, which has quad core ARM processors and programmable logic, is carried on this development board. We used the ARM core as the controller to manage the RF data and delay profile reading from an SD card and saved the beamformed data back to the SD card. The beamformer IP core was connected to the HP ports of the ARM core by the m_axi interface to read directly from PS side DDR memory. It was also directly connected to the MIG IP core by the m_axi interface to directly write to the programmable logic (PL) side DDR memory.

B. Data sets

We used the raw RF channel data recorded from different arrays connected to a Verasonics Vantage system as data input to the beamformer. The beamformed IQ data from the Verasonics and the resulting images were compared to the results obtained using our beamformer.

A tissue-mimicking phantom (CIRS 040GSE) was scanned with a Verasonics L11-5v probe to evaluate lateral resolution and contrast. In this set, the imaging was performed using plane-wave compounding with steering angles from -18° to 18° with a step size of 6°.

As a second test, a mouse brain was scanned with a Verasonics L35-16vX probe. The mouse was injected with microbubbles (DEFINITY, Lantheus Medical Imaging, Inc.) to conduct super-resolution ULM of the mouse brain and to evaluate the speed of our beamformer. Approval of all ethical and experimental procedures was granted by the Institutional Animal Care and Use Committee (IACUC) at the University of Illinois Urbana-Champaign (Protocol No. 19063). The mouse was anesthetized by inducing 4% isoﬂurane mixed with the medical oxygen in a gas induction chamber. The mouse was then transferred to the customized imaging stage and the mouse head was fixed to the stereotaxic frame with ear bars. Furthermore, the anesthesia was maintained by supplying the 2% isoflurane with oxygen through a nose cone. The scalp was removed and both side of the skull between Bregma and Lambda was opened using a rotary Dremel tool to expose the brain. The ultrasound transducer was placed above the cranial window with a coupling gel to image in the coronal plane. A 30-gauge catheter was cannulated through the tail vein. Then, the microbubbles 6 × 10⁶ were continuously infused using the programmable syringe pump (New Era Pump Systems Model 1000) at a flow rate of 10 uL/min. Ultrasound imaging was performed using nine steering angles (-4° to 4° in 1° step size) with a post-compounding frame rate of 1000 Hz. A total of 39 sets of 1,600 frames were acquired. Data in each acquisition had 2,560 samples in the axial dimension.

The RF data and pre-calculated delay profile were then loaded to an SD card. Then, the SD card was mounted to the FPGA development board. After both the RF data and the delay profile were loaded to FPGA’s DDR, the FPGA started to beamform and write the results back to the DDR. After the end of beamforming, the beamformed data on the DDR was written
to the SD card. Matlab R2021b was used to read the beamformed data and display the results.

IV. RESULTS

A. Hardware resource utilization and latency

The latency estimated by HLS is listed in Table I. From Table I, we can observe that the time the beamformer used to beamform one image without loading the delay profile was 20,512 clock cycles, which is close to our estimate from Section II.C. The extra 32 clock cycles were from the pipeline delay. The delay profile only needed to be loaded once before the beamforming process, as we kept the same parameters for the whole process. Therefore, the delay profile loading time would not slow down the beamformer frame rate.

Table I. Latency values.

| Latency (clock cycles) | With delay loading | Without delay loading |
|------------------------|--------------------|-----------------------|
|                        | 36,088             | 20,512                |

The actual hardware resource utilization after Vivado synthesis and implementation is listed in Table II. The resources utilization reported includes the DDR memory interface and interconnect between the beamformer and ARM processor. For an actual ultrasound machine that uses the FPGA as a beamformer, these two parts can be excluded as the RF data can be directly transferred to the beamformer without the need of external DDR memory and ARM processor.

To compare the beamforming speed between the FPGA beamformer and Verasonics beamformer, we measured the time spent by FPGA beamformer on beamforming 3,000 acquisitions out of one set of data’s 14,400 acquisitions (1,600 frames x 9 acquisitions =14,400 acquisitions) due to DDR memory size limitation. The total run time and average frame rate are listed in Table III for the FPGA beamformer. The run time was measured at 0.2911 s for the FPGA beamformer for 3,000 acquisitions while the Verasonics beamformer (Intel Xeon W-2155 10 cores 20 threads 3.3 GHz 64 GB RAM) took 7.35s. The FPGA sped up the beamforming by a factor of 25.

Table II. Hardware resource utilization after implementation.

| Utilization | Available | Utilization% |
|-------------|-----------|--------------|
| LUT         | 70,448    | 30.58        |
| FF          | 85,402    | 18.53        |
| BRAM        | 460,800   | 18.25        |
| DSP         | 1,728     | 54.05        |
| FPGA Power  | 7,428     | 2.98         |

Table III. Measured run time values using the FPGA beamformer.

| Running Time @ 250 MHz | 3,000 Acquisitions | 0.2911 s |
|------------------------|--------------------|----------|
| Average time / Acquisition | 97 µs              |
| Average rate | 10,307 Acqs per second |

B. Beamformed images comparison

The actual beamformed images from the CIRS phantom using our FPGA implementation are shown in Figs. 4 and 5. The Verasonics beamformed image of wire targets is shown in Fig. 4(a) and the FPGA beamformed image is shown on in Fig. 4(b). Visually, there is no lateral resolution degradation of the FPGA beamformed image compared to Verasonics beamformed image. The lateral resolution comparison of Verasonics and FPGA beamformer results is shown in Fig. 4(c).

Images of an anechoic target inside the CIRS phantom were constructed using the Verasonics and the FPGA beamformer. Visually, no differences were observed between the Verasonics beamformed images and FPGA beamformed images. The contrast to noise ratio (CNR) was calculated to compare the performance. CNR is given by

\[
\text{CNR} = \frac{|u_i - u_o|}{\sqrt{\sigma_i^2 + \sigma_o^2}},
\]

where \(u_i\) and \(u_o\) represent the mean pixel intensity inside and outside the anechoic cyst; and \(\sigma_i^2\) and \(\sigma_o^2\) denote the variance of pixel intensity inside and outside the anechoic cyst. The CNR values from the Verasonics beamformer and FPGA beamformer were both 1.0.

Figure 4. B-mode image of CIRS phantom wire target region with Verasonics (a) and FPGA beamformer (b). ROIs (Region of Interests) are marked with red lines. Lateral resolution comparison between Verasonics beamformer and proposed FPGA beamformer with CIRS phantom (c).

Figure 5. B-mode images of an anechoic target in the CIRS phantom with Verasonics (a) and FPGA beamformer (b). ROIs are marked with red rectangle.

Power Doppler images of a mouse brain were created by the accumulation of SVD (Singular Value Decomposition) filtered [18] 1,600 post-compounding frames. The images basing on Verasonics beamformer and proposed method results are shown in Fig.6(a) and Fig.6(b). Visually, they have comparable results in the resolution of vessels. The only noticeable difference is the top part of the image results from FPGA beamformer is slightly darker than that from the Verasonics beamformer. The reason behind this is the subaperture size is smaller for the top part of image results from the FPGA beamformer, which is described in part IID.
ULM images of a mouse brain were generated by 39 sets of data described in IIIB. In the ULM processing, the MB signal with different speed ranges and directions were separated into three groups using 3D Fourier domain filters and processed separately [19]. Velocity maps were generated using a bipartite graph-based MB pairing and tracking algorithm [20],[21],[22],[23]. The final localization and velocity images were the combination of the individual reconstruction images generated from each acquisition. The resulting ULM directional flow images using both the Verasonics and FPGA beamformer are shown in Figs. 6(c) and 6(d), respectively. The zoomed in version images are shown in Figs. 6(f) and 6(g). The cross-section comparison between the directional flow images with Verasonics and FPGA beamformer is shown in Fig. 6(e).

No visually apparent difference can be identified from the ULM images constructed from Verasonics beamformer and FPGA beamformer. The cross-section comparison between two beamformer also shows the similarities between the results of two beamformer.

V. DISCUSSION

A new beamforming algorithm was proposed that integrated with hardware (programmable logic implementation on an FPGA) to achieve ultrafast beamforming for ultrafast ultrasound imaging. The proposed method solves the major obstacle of achieving higher beamforming frame rate by enabling delay profile reuse and parallel beamforming. A beamforming rate of 3.38 GPixels/s was achieved by the proposed ultrasound PWI beamformer.

The image quality of proposed FPGA beamformer is very similar to that of Verasonics beamformer as quantified by the CNR and lateral resolution in the phantom experiments. This indicates that the proposed FPGA beamformer does not sacrifice the quality of the image for speed. The performance of FPGA beamformer with in vivo data was also tested with a microbubble injected mouse brain scan. Power Doppler images and ULM images created with the FPGA beamformer were compared with images created using the Verasonics beamformer side-by-side. Cross-section plots of the ULM images are also provided to directly compare the performance. The correctness and quality of the proposed beamformer can be verified by the visual similarities between the results from the proposed method and the Verasonics beamformer.

The importance of the proposed method consists of potentially enabling continuous unblocked ultrafast ultrasound imaging. This is possible because the proposed beamformer can be easily integrated to FPGAs that can be directly connected to AFEs without the need of PCI-Express interface and host computer in between. The bandwidth of data being transferred to a host PC could be reduced to a much lower value that could match the speed for saving to a hard drive by adding compounding and IQ demodulation to the beamformer. In this
way, long duration, continuous ultrafast ultrasound can be achieved with the proposed beamformer. With continuous unblocked ultrafast ultrasound, ULM could be improved by having a larger number of frames in one data set and much longer tracking duration compared to current ultrasound research platforms. IUS could also be improved by having continuous real-time ultrafast ultrasound imaging. Furthermore, not only beamforming could be done by FPGA, i.e., other time-consuming workload computations on a CPU or GPU, such as compounding, IQ demodulation and high pass filtering, could also be moved to an FPGA to enable faster frame rates and real-time processing.

Using the Xilinx toolchain, the proposed beamformer design can be easily scaled up or scaled down to fit different platforms and applications in a short time period. By solving the common problems related to PWI beamforming, such as limited memory bandwidth and parallelism, the proposed beamformer better utilized memory resources and DSP resources in the FPGA.

A comparison of this work with previous FPGA-based beamformers is shown in Table IV. The resource utilizations of previous works have been converted to the same standard for easy comparison. Our proposed design provides the highest performance with the lowest resource consumption because of delay profile reuse, which eliminates the run-time delay profile calculation and simplifies the memory reading architecture. An important design consideration is the ratio between the number of DSP slices and BRAM blocks available. The number of DSP slices determines how many calculations are performed by the beamformer each clock cycle. The number of BRAM blocks evaluates the total internal memory bandwidth. This DSP to BRAM ratio could provide a simple metric that evaluates the memory utilization efficiency of the architecture. For example, the higher the ratio the more calculations that can be performed with the same amount of memory resources. The number of DSP slices in most Xilinx FPGAs is several times that of the BRAM blocks. Based on Table IV, our proposed solution has this ratio much larger than previous works, which coincides with the FPGA’s resource ratio. In this way, more beamformers were implemented inside the same FPGA than for previous works.

Limited to our current FPGA platform resources, the highest beamforming rate we achieved was 3.38 GPixels/s. Given the size of frames defined in IIIB, the frame rate is 10,307 fps, which is already faster than current ultrafast ultrasound imaging frame rates that have been reported. If a larger FPGA is used, even higher frame rates can be achieved. For example, the beamforming rate can be easily increased to 12 GPixels/s with a larger FPGA such as Xilinx Kintex Ultrascale KU115 by implementing four identical beamformers running in parallel as it has more than four times BRAM and DSP resources compared with the FPGA used in this study.

One of the limitations of the proposed design is the difficulty of using or modifying it by users without FPGA coding experience. Because FPGA coding is totally different from Matlab or GPU programming, it requires a strong hardware background to fully utilize the advantages of an FPGA’s special architecture, which is different from a GPU or CPU. This limitation can be partially solved by providing a toolkit composed of the FPGA bitstream and a user API that allows users to control the FPGA accelerator as a black box instead of diving into FPGA coding. Another limitation is the deployment of the proposed methods to current ultrasound research platform as the FPGA must use PCI-Express to communicate with host PC, which has the same bottleneck associated with using a GPU. While GPU has drivers that are well developed and tested, special drivers need to be written and tested for FPGA accelerator. This can be solved by integrating the proposed methods near the front end of ultrasound machine, which means the data transferred to host PC is already beamformed or using an FPGA accelerator with fully developed drivers such as Xilinx Alveo acceleration card.

### VI. Conclusion

The proposed FPGA implementation of an ultrafast beamformer enabled steered PWI with high versatility and scalability. Due to the versatility of the implementation, there is no need to regenerate the bitstream or reprogram the FPGA to adapt to different probes or steering angles. The HLS allows changing the scale of FPGA beamformer to fit different application scenarios in a very short turnaround. This architecture can be utilized in both high-end ultrasound research platforms that need a frame rate of over 10,000 FPS or for portable pocket ultrasound scanners that need a high efficiency, low power compact FPGA beamformer.

### Acknowledgment

Xilinx University Program (XUP) donated ZCU106 development board.

### References

[1] M. Tanter and M. Fink, "Ultrafast imaging in biomedical ultrasound," in IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control, vol. 61, no. 1, pp. 102-119, January 2014, doi: 10.1109/TUFFC.2014.2882.

[2] J. Bercoff et al., "Ultrafast compound doppler imaging: providing full blood flow characterization," in IEEE Transactions on Ultrasonics,
Ferroelectrics, and Frequency Control, vol. 58, no. 1, pp. 134-147, January 2011, doi: 10.1109/TUFFC.2011.1780.

[3] Bercoff J, Tanter M, Fink M. Supersonic shear imaging: a new technique for soft tissue elasticity mapping. IEEE Trans Ultrason Ferroelectr Freq Control. 2004;51(4):396-409. PubMed PMID: 15139541.

[4] Song P, Urban MW, Manduca A, Zhao H, Greenleaf JF, Chen S. Comb-push ultrasound shear elastography (CUSE): A novel and fast technique for shear elasticity imaging. IEEE International Ultrasonics Symposium; Dresden, Germany 2012. p. 1842-5.

[5] Errico C, Pierre J, Pezet S et al. Ultrasound ultrasound localization microscopy for deep super-resolution vascular imaging. Nature 2015, 527, 499-502 (2015). https://doi.org/10.1038/nature16066

[6] Song P, Trzasko JD, Manduca A, Huang R, Kadirvel R, Kallmes DF, Chen S. Improved Super-Resolution Ultrasound Microvessel Imaging With Spatiotemporal Nonlocal Means Filtering and Bipartite Graph-Based Microbubble Tracking. IEEE Transactions on Ultrasound, Ferroelectrics, and Frequency Control. 2018;65(2):149-67. doi: 10.1109/TUFFC.2017.2778941.

[7] Macé E, Montaldo G, Cohen I et al. Functional ultrasound imaging of the brain. Nat Methods 8, 662–664 (2011). https://doi.org/10.1038/nmeth.1641

[8] G. Montaldo, M. Tanter, J. Bercoff, N. Benec and M. Fink, "Coherent plane-wave compounding for very high frame rate ultrasonography and transient elastography," in IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control, vol. 56, no. 3, pp. 489-506, March 2009, doi: 10.1109/TUFFC.2009.1067.

[9] C. Risser, H. J. Welsch, H. Fontara, H. Hewener and S. Tretbar, "High channel count ultrasound beamformer system with external multiplexer support for ultrafast 3D/4D ultrasound." 2016 IEEE International Ultrasonics Symposium (IUS), 2016, pp. 1-4, doi: 10.1109/ULTSYM.2016.7278714.

[10] Demené, C., Robin, J., Dizeux, A. et al. Transcranial ultrafast ultrasound localization microscopy of brain vasculature in patients. Nat Biomed Eng 5, 219–228 (2021). https://doi.org/10.1038/s41551-021-00697-x

[11] Lowerison, M.R., Huang, C., Lucien, F. et al. Ultrasound localization microscopy of renal tumor xenografts in chicken embryo is correlated to hypoxia. Sci Rep 10, 2478 (2020). https://doi.org/10.1038/s41598-020-59338-z

[12] B. Y. S. Yu, I. K. H. Tsang and A. C. H. Yu, "GPU-based beamformer: Fast realization of plane wave compounding and synthetic aperture imaging," in IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control, vol. 58, no. 8, pp. 1698-1705, August 2011, doi: 10.1109/TUFFC.2011.1999.

[13] D. Hyun, Y. L. Li, I. Steinberg, M. Jakovljevic, T. Klap and J. J. Dahl, "An Open Source GPU-Based Beamformer for Real-Time Ultrasound Imaging and Applications," 2019 IEEE International Ultrasonics Symposium (IUS), 2019, pp. 20-23, doi: 10.1109/ULTSYM.2019.8926193.

[14] B. Y. S. Yu, M. Walczak, M. Lewandowski and A. C. H. Yu, "Live Ultrasound Color-Encoded Speckle Imaging Platform for Real-Time Complex Flow Visualization In Vivo," in IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control, vol. 66, no. 4, pp. 656-668, April 2019, doi: 10.1109/TUFFC.2019.2892731.

[15] Boni E, Bassi L, Dalila A, et al. ULA-OP 256: A 256-Channel Open Scanner for Development and Real-Time Implementation of New Ultrasound Methods. IEEE Trans Ultrason Ferroelectr Freq Control. 2016;63(10):1488-1495. doi:10.1109/TUFFC.2016.2566920

[16] E. Boni et al., “Architecture of an Ultrasound System for Continuous Real-Time High Frame Rate Imaging,” in IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control, vol. 64, no. 9, pp. 1276-1284, Sept. 2017, doi: 10.1109/TUFFC.2017.2727980.

[17] N. A. Campbell and J. A. Brown, "A Real-Time Dual-Mode High-Frequency Beamformer for Ultrafast and Focused Imaging," in IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control, vol. 69, no. 4, pp. 1268-1276, April 2022, doi: 10.1109/TUFFC.2022.3151218.

[18] P. Song, A. Manduca, J. D. Trzasko and S. Chen, "Ultrasound Small Vessel Imaging With Block-Wise Adaptive Local Clutter Filtering," in IEEE Transactions on Medical Imaging, vol. 36, no. 1, pp. 251-262, Jan. 2017, doi: 10.1109/TMI.2016.2605819.

[19] C. Huang et al., “Short acquisition time super-resolution ultrasound microvessel imaging via microbubble separation,” Sci. Rep., vol. 10, no. 1, pp. 1–13, Dec. 2020.