Sonic Sight: Preliminary Report

1st Justin Okorie
Department of Computer Science
Massachusetts Institute of Technology
Cambridge, MA, USA
jokorie@mit.edu

2nd Dylan Bradford
Department of Electrical Engineering and Computer Science
Massachusetts Institute of Technology
Cambridge, MA, USA
dylanjb@mit.edu

***Abstract*—This project aims to design and implement a real-time system for detecting the movement of persons and objects using an ultrasonic phased array and FPGA-based signal processing. The system emits sweeping ultrasonic waves and processes their echoes to determine the speed, position, and motion patterns of detected entities. We evaluate its performance by simulating transmitted and received waves to determine throughput for processed samples and by measuring latency from signal reception to distance calculation.**

***Index Terms*—Digital systems, Field-programmable gate arrays, Phased arrays**

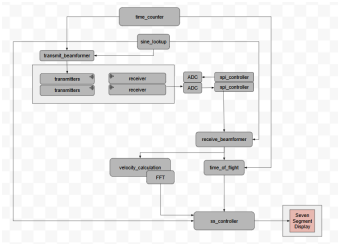

Fig. 1. High Level Block Diagram

I. Introduction

The Sonic Sight system leverages ultrasonic phased array technology combined with FPGA-based signal processing to achieve real-time object detection and motion tracking. By emitting sweeping ultrasonic waves and analyzing the resulting echoes, the system can estimate both the position and velocity of detected entities. Designed to operate efficiently over a range of 1 to 5 meters, the system's applications include motion tracking, object detection, and environmental monitoring. The following sections detail the physical construction, beamforming methodology, and timing mechanisms used to realize this functionality.

II. Physical Construction

The phased array is implemented on a breadboard and consists of the following components:

- **Ultrasonic Transducers:** Separate transmitters and receivers are arranged in a linear configuration. This separation optimizes signal transmission and reception.
- **Driver Electronics:** Each receiver has its own ADC for digitizing incoming signals. Additionally, operational amplifiers are used to enhance signals sent to the transducers and to amplify received echoes before digitization.
- **Processing Unit:** A Xilinx Spartan-7 XC7S50-CSGA324 FPGA is used to perform real-time signal processing and beamforming calculations.

This modular construction enables flexibility in design and straightforward integration of beamforming and signal processing modules.

III. Timing and State Management

The top-level design operates in two distinct states: *active_pulse* and *active_receive*. These states alternate over a fixed global period of 2^{24} clock cycles. The *active_pulse* state lasts 2^{19} clock cycles, during which ultrasonic waves are transmitted. Once this state ends, the system transitions to *active_receive*, where incoming echoes are processed without additional transmissions to avoid cross-talk.

The global PWM module governs state transitions. During the *active_pulse* state, each transmitter generates a square wave with a 40 kHz frequency, using phase delays computed from the beamforming equation. The *active_receive* state enables ADC sampling, where signals are aggregated after compensating for phase delays.

This timing protocol enforces a minimum detectable range of 1 meter while supporting object detection up to 5 meters. The separation between transmission and reception phases ensures reliable echo detection and prevents interference between transmitted and received signals.

IV. Beamforming Design and Signal Processing

A. Dynamic Beamforming

Beamforming allows the system to steer the emitted ultrasonic waves without physically reorienting the array. By introducing precise phase delays between adjacent transmitters, constructive and destructive interference patterns are manipulated to produce wavefronts in a desired direction.

Through experimentation, the transducers were found to perform optimally at a frequency of 40 kHz, corresponding to a wavelength of $\lambda = \frac{343\,\text{m/s}}{40\,\text{kHz}} = 8.575\,\text{mm}$. To reduce grating lobes, the ideal spacing of transducers is $\frac{\lambda}{2}$ but they are spaced 9mm apart which is a limitation of their physical dimensions.

The governing equation for our beamforming is:

$$\Delta t = \frac{d \cdot \sin(\theta)}{c} \tag{1}$$

where Δt is the time delay, d is the element spacing within the phased array, θ (perpendicular to the linear array), and c is the speed of sound in air (343 m/s).

The primary dynamic component of this computation is θ, which allows the system to steer waves in varying directions during operation. During the sweeping motion, the angle θ is incremented by $+10^\circ$ every global period (2^{24} clock cycles), oscillating between $[-30^\circ, +30^\circ]$. This dynamic adjustment ensures that waves are transmitted and received in a continuously updated direction.

To achieve this, a sine lookup table (LUT) is utilized. The LUT takes the signed θ value as input and outputs a scaled sine value alongside a sign bit to represent polarity. The absolute sine value is used to calculate the time delay, while the sign bit determines the directionality (i.e., whether the delay accumulate from the left or right side of the array). Decoupling signedness from magnitude avoids potential casting errors during computations, ensuring reliable beamforming calculations.

The sine LUT values are reused in the receive beamforming module to calculate sample-based delays for each receiver. This consistency ensures accurate time delays across both transmission and reception, enabling precise wave steering and enhanced object detection capabilities.

B. Transmit Signal Generation

The system generates 40 kHz square waves using the FPGA's PMOD pins, which toggle between digital high (3.3 V) and low (0 V) to drive the transducers. Each transmitter generates a square wave with a period equivalent to 2500 cycles of the 100 MHz system clock. A PWM module controls these signals, maintaining a 50% duty cycle (high for 1250 cycles and low for 1250 cycles).

To introduce phase offsets, time delays are calculated for each transmitter. The delay for an array element is given by:

$$\text{delay\_per\_element} = \frac{d \cdot \text{clk\_freq}}{c} \tag{2}$$

This value is multiplied by the element's index and the sine value from the LUT. The resulting offset is stored and applied as a starting point in the square wave cycle, enabling precise phase control.

C. Receive Signal Processing

The ADC101S101 ADCs are used for digitizing the received signals. Operating at a 1 MSPS sampling rate, the ADCs are clocked with a 20 MHz clock. To achieve this, a trigger signal is generated every 100 cycles of the 100 MHz system clock. The ADC receives two inputs: a chip select (CS) bit signaling the start of transmission and an SCLK signal at 20 MHz. The ADC's output, SDATA, provides digitized data from the receiver.

A modified SPI controller interfaces with the ADCs. Each ADC transaction requires 16 SCLK cycles, and the SPI module is configured accordingly, with a data width of 16 bits and a data clock period of 5 cycles to produce the SCLK. A single-cycle trigger initiates transactions when the 1 MHz trigger is high, provided the system is not actively transmitting.

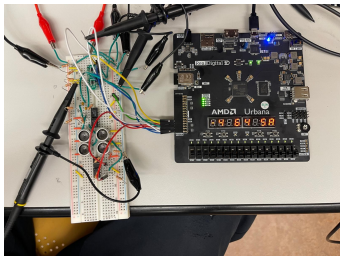

Fig. 2. Transmitter and Receiver 2-element Phased Array

The outputs of the SPI modules are streamed into buffers within the receive beamformer module. To aggregate signals, the module computes which sample in each buffer corresponds to the desired delay for the transmitted angle. The delay is calculated as:

$$\text{sample\_delay} = \frac{d \cdot \text{samp\_rate} \cdot \text{sine\_LUT\_val} \cdot \text{element\_idx}}{c} \tag{3}$$

These delayed samples are summed to create an aggregated waveform, which focuses on signals from the desired direction while attenuating others. This aggregated waveform improves the system's ability to detect and isolate objects based on directional echoes.

V. Signal Processing

A. Time Since Emission and Echo Detection

To determine the time of flight (ToF), the system counts the number of clock cycles from the moment a signal is emitted until its reflected echo is detected. The echo detection process occurs immediately after the receive beamforming module aggregates the receiver inputs, accounting for phase delays.

The detection mechanism uses an empirically determined parameter, *echo_detected*, which serves as the decision point for validating reflections. Signals exceeding this threshold are considered valid echoes. When a valid echo is detected, the *echo_detected* signal is asserted high for the remainder of the global period. This ensures synchronization across downstream processing modules and provides visibility for subsequent calculations.

B. Time of Flight (ToF) Calculation

The *time_of_flight* module operates in conjunction with a global timer that tracks the position within the global period. Upon detecting a valid echo, the module records the current clock counter value. The distance to the detected object is then calculated using the formula:

$$\text{Distance (cm)} = \frac{\text{ToF (clock cycles)} \cdot c}{2 \cdot f_{\text{clk}}} \tag{4}$$

Where:

- $c = 34300\,\text{cm/s}$ is the speed of sound,
- $f_{\text{clk}} = 100\,\text{MHz}$ is the system clock frequency,
- The division by 2 accounts for the round trip of the ultrasonic wave.

This calculation provides precise distance measurements for objects within the detectable range and ensures seamless integration with subsequent processing modules.

VI. Velocity Calculations

Velocity estimation leverages Doppler shift analysis through Fast Fourier Transform (FFT)-based processing. ADC samples are streamed into an FFT module, which outputs the frequency spectrum of the received signal in real time. The FFT module in the system is configured with the following specifications:

- **Transform length:** 2048 samples,
- **Sampling rate:** 1 MSps,
- **Frequency resolution:** 488 Hz

While the FFT processes the samples, the system concurrently tracks the peak frequency in the spectrum. After processing all 2048 samples, the peak frequency is used to compute the Doppler shift. The object's velocity is derived using the formula:

$$v = \frac{f_{\text{doppler}} \cdot c}{2 \cdot f_{\text{carrier}}} \tag{5}$$

Where:

- f_{doppler} is the frequency shift observed in the FFT output,
- $f_{\text{carrier}} = 40\,\text{kHz}$ is the frequency of the transmitted ultrasonic wave,
- $c = 343\,\text{m/s}$ is the speed of sound.

The calculated velocity is sent to the display module for visualization. This two-step process ensures that both distance and velocity of detected objects are computed accurately in real time.

A. Frequency Resolution and Velocity Buckets

Frequency resolution directly affects the smallest velocity increment the system can discern. With the current configuration of a 2048-point FFT and a sampling rate of 1 MSps, the frequency resolution is:

$$\Delta f = \frac{\text{Sampling Rate}}{\text{FFT Size}} = \frac{1\,\text{MSps}}{2048} \approx 488\,\text{Hz} \tag{6}$$

This translates into discernable velocity "buckets" given by:

$$\Delta v = \frac{\Delta f \cdot c}{2 \cdot f_{\text{carrier}}} \tag{7}$$

For the current setup:

$$\Delta v = \frac{488 \cdot 343}{2 \cdot 40000} \approx 2.09\,\text{m/s} \tag{8}$$

To achieve finer velocity increments, we propose reducing the sampling rate to 500 kSps and increasing the FFT size to 4096 samples. This adjustment would yield:

$$\Delta f = \frac{500\,\text{kSps}}{4096} \approx 122\,\text{Hz} \tag{9}$$

$$\Delta v = \frac{122 \cdot 343}{2 \cdot 40000} \approx 0.52\,\text{m/s} \tag{10}$$

Fig. 3. Seven Segment Example Display

By reducing the frequency resolution from 488 Hz to 122 Hz, the system would be able to detect much smaller changes in velocity, significantly improving object motion analysis. This enhancement will be crucial for applications requiring high precision in velocity detection, but its implications are still yet to be considered.

B. Custom FFT Module Integration

The FFT module integrated into our system is a customizable, open-source design optimized for FPGA implementation. It efficiently processes the 2048 input samples to generate a frequency spectrum with minimal latency, meeting the system's performance requirements. This modular design supports high throughput and is adaptable for future enhancements, such as varying sampling rates or increased transform lengths.

By combining precise timing, efficient signal processing, and robust beamforming techniques, our system reliably calculates distance and velocity. These capabilities underpin the real-time detection and motion tracking functionality, providing accurate and dynamic performance in various scenarios.

VII. Display Module

The display module plays a crucial role in providing real-time feedback by visually presenting the computed distance and velocity of detected objects. It utilizes an 8-digit Seven-Segment Display, divided into two distinct banks.

A. Seven Segment Display

The first two segments are responsible for conveying the velocity information. Precisely, the first segment displays a "-" sign should we detect the object to be moving away from the device. The second segment represents the radial m/s of the detect object represented in hexadecimal format.

The fourth and fifth segments are reserved for the distance (cm.) output from upstream modules represented in hexadecimal format.

Lastly, the seventh and eighth segments hold the angle of the object in which the wave emission produced the aforementioned distance and velocity results. The first bank of the display module is dedicated to representing the calculated distance from detected objects in centimeters. Key features include:

VIII. Evaluation

Currently, the design is not operating as expected, and we suspect the issue is in passing the values read from the ADC through a threshold check to the downstream modules.

A. Functionality

All of the above mention modules mentioned above have been thoroughly unit tested in simulation, handling edge cases within the expected utility.

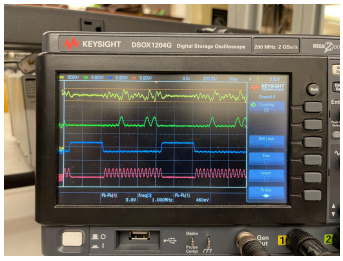

Fig. 4. Receiver Output, ADC Output, CD, ADC Clock

B. Latency and Timing

Our global period before we change the angle and begin emitting our waves is exactly 2^{24} clock cycles which translates into .167 seconds. As such our throughput closely follows. The main bottleneck of this computation was the FFT transform module, however we acknowledge that we could speed up its computation had we utilized additonal DSPs. Our design features 1 DSP block and 0 BRAM in total.

Timing reports indicated that the post-placement Worst Negative Slack was 4.831ns, and the post-route WNS was 4.431ns meaning our longest paths completed with significant margin before the next clock edge.

IX. Implementation Insights

Our modules are implemented such that they are generalizable to a linear array of *n* transmitters. Conveniently we were able to feed square waves into the transmitters, so accounting for time delay in the beamforming stage reduced to just adding a unique reset offset to the pwm driving each of the transmitters.

The use of a sine lookup table for our transmit angle simplified our design as we were able to determine time delays which could be used for both the transmit and receive beamforming modules.

Lastly we used square waves to drive our transducers which allowed us to ignore intermediate values a sine curve would have between its max and min values.

In doing this project again we would create a more involved phased array with at least double the amount of elements for increased signal strength and detection range. We would also put the design on a pcb and use better driver electronics to reduce concerns with wiring.

Together we researched the math and implementation for the design. Justin wrote the overall structure for the modules as well as tested the modules. Dylan set up the analog and digital hardware interfacing.

X. Acknowledgment

We would like to express their gratitude to Dan Gisselquist for the use of their FFT module. We would also like to extend our heartfelt thanks to Professor Joseph Steinmeyer for his invaluable guidance throughout this project and for assisting in acquiring the necessary equipment to bring this work to fruition. Finally, we thank Stephen Kandeh, our teaching assistant, for his support and constructive feedback.

References

[1] Dan Gisselquist, “An Open Source Pipelined FFT Generator” [Online]. Available: https://zipcpu.com/dsp/2018/10/02/fft.html.

[2] P. Delos, B. Broughton, and J. Kraft, “Phased Array Antenna Patterns—Part 1: Linear Array Beam Characteristics and Array Factor,” Analog Devices, [Online]. Available: https://www.analog.com/en/resources/analog-dialogue/articles/phased-array-antenna-patterns-part1.html