Passive Sonar

Final Report

Eric Chang
Mike Ilardi
Jess Kaneshiro
Jonathan Steiner

Project Setup

Our basic setup is shown on **Figure 1**. Two microphones are spaced a distance x apart. A sound source is located on a plane a distance l perpendicular to the plane in which the microphones are situated. A projection screen is located equidistant between the microphones. d1 and d2 represent the distance from the sound source to the respective microphones. By means of the trigonometric calculations outlined on the attached diagram, we can triangulate the horizontal position of the sound source.

In order to input sound into the Xilinx board it was necessary to fabricate adapters for the parallel pin adapters on the board. Additionally, we used a stereo mixing board as a preamplifier rather than rely upon the those built into the board so that we could have better control over the gain.

Calculations

Before we began implementation we worked out the basic math for mapping a time delay to a physical distance and a distance in pixels. What follows is an explanation of the diagram and associated calculations in **figure 1.**

t1=5/(cosθ)x343) since **cosθ=adjacent/hypotenuse**, it is equal to **5/d1**, which gives us: **t1=d1/343=((25+x^2)^.5)/343 while t2=((25+((5-x)^2)^.5)/343** since **x2=5-x1**

This makes the delay equal to:

y=(1/343)((25+x1^2)^.5-(50-10x1+x1^2)^.5)

We calculated X in terms of Y using Matlab, yielding the somewhat unwieldy function:

X1=[5/2+343/10/(471000y^2-100)(-161000000y^3+34300y+20(13800000000y^4-17600000y^2+3125)^.5)y+11765y^2]

Using this equation we generate 1000 evenly spaced discrete values indexed by delay in our lookup table. The index numbers mapped in a linear manner from zero to 6.038 milliseconds (the maximum possible delay given our proposed project setup) and contained values of x corresponding to a location on the screen from 0-320, representing half the width of a VGA display. In order to save memory, we did not generate the negative values as they are mirrors of the positive 320 locations and simple logic could be used to decide which side of the screen to draw the location indicator on.

Process Flow

Figure 2 contains a block diagram outlining at an abstract level the logical flow of our Passive Sonar.

Stage 1:

The analog sound signals enter the left and right microphone inputs on the audio codec, labeled A on the block diagram. Serial digital audio samples corresponding to the left and right analog inputs feed out of two digital outputs on the codec into separate 16-bit shift registers, labeled B and C on the block diagram.

Stage 2:

Data from the shift registers are read into the peak detectors, labeled F and G, once an entire 16-bit signal has been loaded into the shift registers. The peak detector is from an example of a limiter. The purpose of a general limiter is to provide control over the high peaks in a signal, so the peak detector implemented in it is ideal using a feedback algorithm simulating a differential. We use this peak detector to find when the signal exceeds a certain differential peak threshold (after the amplitude passes a certain "noise threshold"), thereby eliminating almost all noise that would confuse our Sonar. The following limiter written in Matlab code is taken from DAFX by Udo Zölzer:

```
slope=1;
thresh=.5;
rt=0.01;
at=0.4;

xd(1)=0;
a=abs(x(n))-xd(n-1);
if a<0, a=0; end;
xd(n)=xd(n-1)*(1-rt)+at*a;
if xd(n)>thresh
```

When xd>thresh, we have hit a peak in the sound input.

An implementation of this algorithm was attempted in C. We needed to change the attack and release value times to work with our sound source (configuring the AK4565's gain proved to be too difficult) as thresh values to detect sounds at a distance of 5 meters from the mics while still eliminating as much noise as possible.

AT=1-e^(-2.2T/tat)

RT=1-e^(-2.2T/trt)

The sampling period T = 1/48.828kHz, and t_{at} and t_{rt} were estimated. Later, real-life tests were done to get the values of AT and RT that seemed to create good peak detection for our purposes. We used a .wav file of claps on CD that we generated using Matlab with specifically programmed delays between left and right channels (1-6ms).

Out of considerations for speed, we worked on a magnitude one-thousand times the natural values of the constants and data samples. In this manner we avoided having to work in floating point numbers.

Stage 3:

The subtractor D receives the first sample and holds it until it receives the second sound sample. It then subtracts the time difference in receiving, giving us a delay. We use an interrupt to give us the delay to use in the subtractor. We encountered a huge problem here in that every frame we processed in C held 64 samples. Unfortunately, during the processing, we were losing an average of 26 frames! 6 milliseconds fits easily in only 5 frames, so we were obviously losing the valuable data required to find the time difference between the two successive peaks (left and right).

Stage 4:

Our delay is sent to our lookup table J. The lookup table is filled with an array that is set up to accept an input of the delay and then output the horizontal value from 0 to 320 as a vertical bar. Our lookup table takes 1-1000 as its index, but decimal places are being truncated so there are only 320 different outputs. Since the maximum delay is 6.038 milliseconds, 6.038 milliseconds would give us a horizontal value of 320. The program stores a variable to indicate whether this is to the left or the right, depending upon which microphone is activated first. A delay of zero would output a value of 0 (center).

Stage 5:

The microblaze has final control over the VGA framebuffer. It arbitrates between the output of diagnostic data obtained from the registers and the visualization of the sound source.

Stage 6:

Video output from the framebuffer appears on a display. Our initial project plan called for a projector set up in such a way that a marking of one meter on the display would correspond to one physical meter.

Revised Project Proposal and Project Wrap-up:

Our project aim was to display the horizontal position of a sound source. To accomplish this, we set up two microphones and calculated the delay in the times the two microphones received a peak in their respective sound inputs. This entailed working with an audio codec to accept analog sounds and output digital signals and setting up a peak detector to tell when the sound is first received and to remove ambient noise.

Our plan had always been to write the bulk of the sonar in VHDL and generate an interface in C as it seemed unlikely that the microblaze processor would be able to triangulate sound-sources in real-time. However, this proved difficult to debug so we decided to first try and implement the detector in C with specialized hardware for parallel multiplication. Learning to write a peripheral capable of communicating with the microblaze proved tedious and we ran into addressing problems in which the peripheral was able to recognize that the microblaze was addressing it, but seemed unable to extract the lower half of the address in order to address specific registers we wanted to create within the device.

In the meantime an attempt was made to implement the bulk of the algorithm in C. Given the speed at which samples arrive, 48.828 kHz, and the time necessary to multiply large 16-bit numbers, a great quantity of frames of audio data (64 32-bit stereo samples) were lost (approximately 26 frames were lost for every one successfully received and processed). Unfortunately, just 5 frames of data represents 6 ms of time, and so the odds of capturing corresponding peaks successfully were quite low.

Within the last few days it was decided that a system in which interrupts from the audio peripheral would drive a frame count and 6 ms of data subsequent to the discovery of a peak would be recorded in a buffer. Unfortunately, this method failed to ever function correctly.

Project Breakdown

Audio codec – Provided by Cristian
Manipulating Christian's audio peripheral – Jess and Mike
Interrupts – whole group
C programming – mainly Jess and Mike, Jon and Eric to a lesser extent
Display – Eric
Peak Detector – Jon and Jess
Wires and hookup – whole group
Matlab – Eric, Jess, and Jon
.MHS and .MSS – Jess and Mike
Nonfunctioning Multiplication Peripheral - Mike
Write up – Jon
Look Up Table – Eric
Figures and Trigonometry – Jon

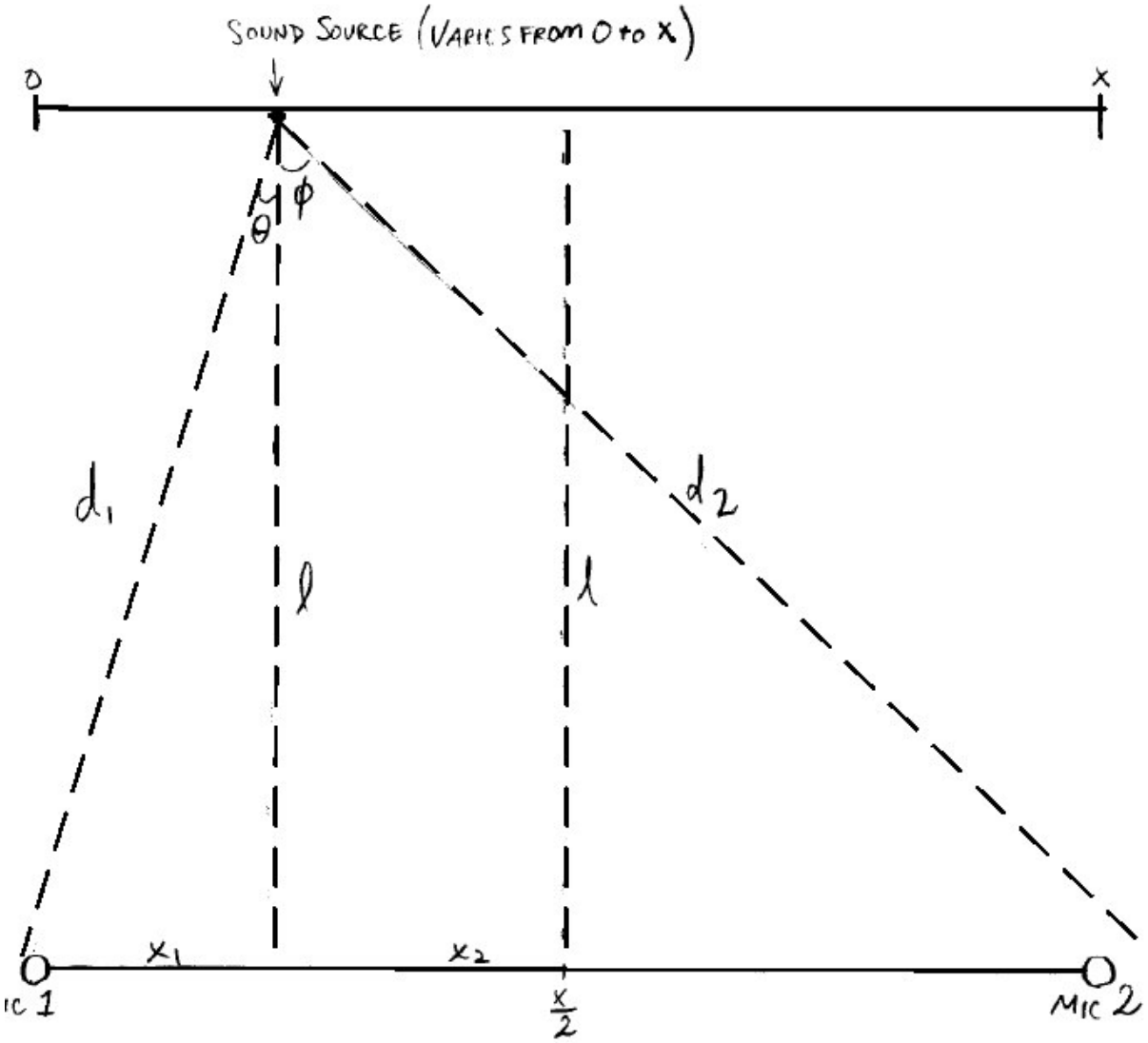
Lessons Learned and Advice for Future Students

1. Flaws in hardware design can result in highly unpredictable behavior. It is very difficult to debug hardware issues. The situation is only made more frustrating by the fact that compilation takes around 5 minutes.
2. Thus, one cannot design hardware the way one programs computer software. This means that trial and error is not a good paradigm for hardware design. Hardware must be carefully planned in excruciating detail.
3. Timing problems abound when working in digital design.
4. Software is slow.
5. VHDL lulls those familiar with computer programming into a false sense of understanding with its imperative language-like constructs and syntax. However, it very rarely deigns to do what the designer had hoped it would. For these reasons it would be advisable for future students to become very familiar with the basics of VHDL prior to attempting to undertake a complex project. Many students are used to learning new computer languages by trying things out and observing similarities to other languages. This will get you nowhere with VHDL.
6. Sometimes things do not work properly and the reasons for this are unclear. Other times things do work and the reasons are equally mystifying.

Included Files

system.mss
system.mhs
main.c
isr.c
lut.h
opb_xsb300_ak4565_v1_00_a.vhd
processor.vhd (didn't work, though we did succeed in sending data over the bus)
processor.pao
processor.mpd

Figure 1



$$d_1 = \frac{l}{\cos\theta} \qquad d_2 = \frac{l}{\cos\phi} \qquad \theta + \phi = \frac{\pi}{4}$$

$$\frac{l}{\cos\theta} = v t_1 = 340 \frac{m}{s} \cdot t_1 \qquad t_1 = \frac{l}{\cos\theta \cdot 340}$$

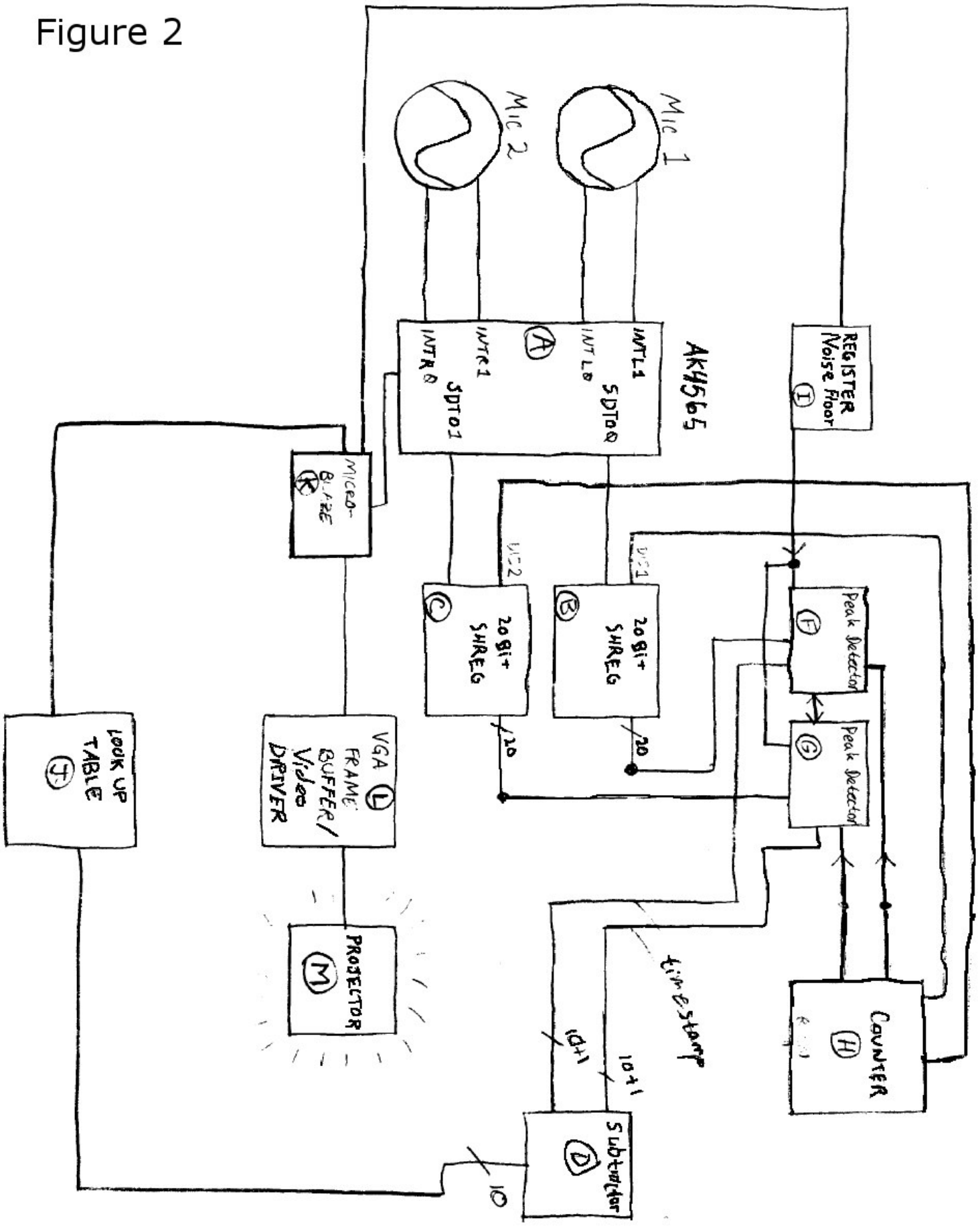
$$\frac{l}{\cos\phi} = v t_2 = 340 \frac{m}{s} \cdot t_2 \qquad t_2 = \frac{l}{\cos\phi \cdot 340}$$

when $t_1 = t_2$, $\phi = \theta$, $d_1 = d_2$

$x_1 = x_2 = \frac{x}{2} =$ Delay (0)

Delay $= t_1 - t_2$ Delay < 0, $x_1 < x_2$; Delay > 0, $x_2 > x_1$

Figure 2


MIC 1
MIC 2
AK4565
INTL1
SDTOQ
INTR1
SDTO1
INTR0
A
REGISTER
Noise Floor
I
MICRO-
K
VC1
VC2
B
20 Bit
SHREG
C
20 Bit
SHREG
20
20
F
Peak Detector
G
Peak Detector
H
COUNTER
time stamp
10+1
10+1
D
Subtractor
10
VGA
FRAME
BUFFER/
Video
DRIVER
L
PROJECTOR
M
LOOK UP
TABLE
J