Ultrasonic Positioning System

EDA385 – Embedded Systems Design - Advanced Course

Joakim Arnsby, et04ja@student.lth.se
Joakim Baltsén, et05jb4@student.lth.se
Simon Nilsson, et05sn9@student.lth.se
Erik Osvaldsson, et05eo1@student.lth.se

2008-10-19

Abstract

In this project a positioning system is built by using ultrasonic sound waves. The system uses a transmitter and two receivers and the position is calculated in 2 dimensions using trilateration. Due to the physical properties of ultrasonic sound waves the system is quite sensitive to obstacles and the maximum distance between transmitter and beacon is about 1.5m. This distance could probably be increased with optimized filters. The precision of the system is good in some areas but for some reason time measurements that weren't proportional to the distance at longer distances were obtained. This made the precision lower in some areas. However, this project shows that the principle of positioning objects with ultrasonic sound waves is achievable.

Table of contents

Introduction

The main idea of this project is to create a positioning system using distance measurements with ultrasonic sound waves. In some sense the transmitter acts as a master and the receivers together with the FPGA acts as slaves since the transmitter triggers the measuring process by telling the FPGA when to start count and also transmits an ultrasonic signal. When the transmitted signal is received the FPGA stops counting and the position can be software calculated. The basic physical setup is shown in Figure 1.

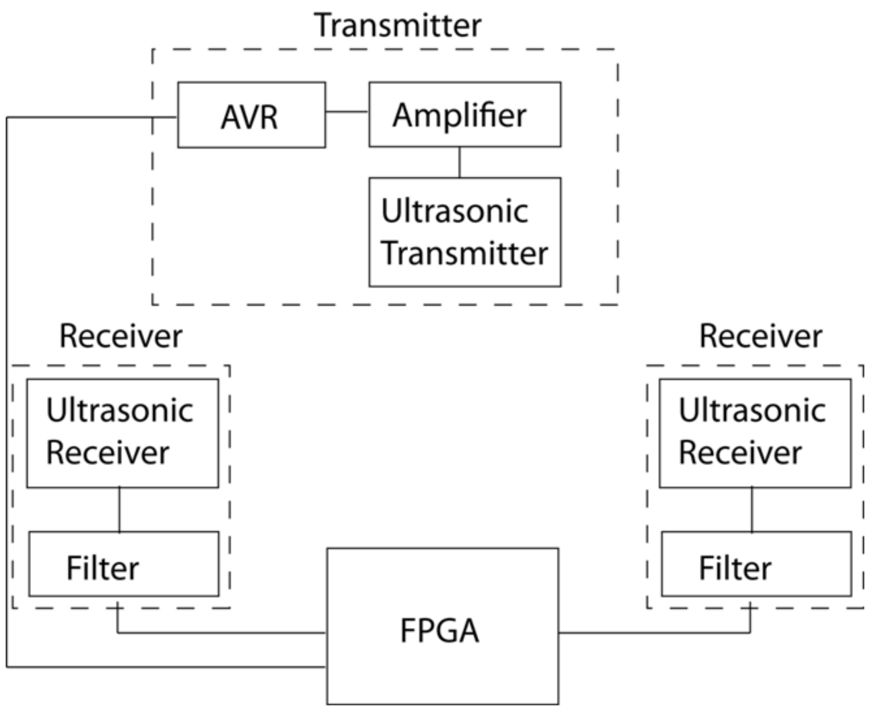

Figure 1: Basic overview showing how the beacons and the transmitter interact via the FPGA.

FPGA hardware

The FPGA design incorporates the following hardware components; 4 GPIOs, 2 UARTs, 1 Interrupt controller, 1 Timer, 1 SPI, 1 Memory controller and 1 MicroBlaze processor. However some of these components are not used in this case, and they will be left out from now on.
How the hardware is connected is shown in Figure 2.

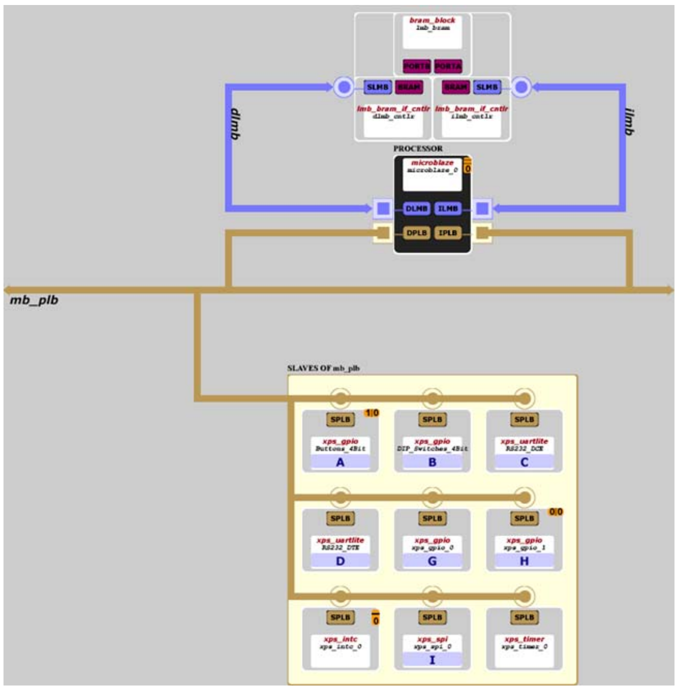

Figure 2: Xilinx building block diagram.

FPGA Software

To handle the different interrupts the interrupt controller identifies the interrupt and activates the corresponding software handler. At first the transmitter activates the timer via an interrupt telling that it has started to transmit ultrasonic pulses. When the ultrasonic pulse is received another interrupt measures the time and the position can then be calculated.

Features not implemented

Initially the intention was to implement the send interrupt via a RF-link but due to time constraints the radio communication part was replaced with a wired connection. Because of this the distance capabilities weren't fully tested since the used wire length was limited.

Theory

When active beacons are used for positioning one can distinguish between two different methods, triangulation and trilateration. With triangulation angles are measured and with trilateration distances are measured. In this project trilateration is used.

To measure the distance between the transmitter and the receiver ultrasonic sound waves are used in this project. The transmitter sends out an ultrasonic sound wave and simultaneously a signal is sent via cable which tells the receiver to start count the time. When the sound wave reaches the receiver it stops counting. By knowing the speed of sound this distance can be calculated with the relation distance=speed*time.

There are different ways to implement a positioning system with ultrasonic sound waves. One way is to have a single transmitter and multiple fixed-location receivers and another way is to have a single receiver and multiple fixed-location transmitters. The first alternative is probably better suited when only one or a few objects are to be located whereas the latter can take care of more objects to be located. In this project the first alternative is chosen mostly to be able to have the computations in the FPGA and since only one object is to be located it works good.

To get the unambiguous position in two dimensions three receivers are needed. However, by limiting the working area with 50% only two receivers are needed. Then the second possible position can be discarded.

Below follows a derivation for the object's coordinates. See Figure 3 for variable definitions.

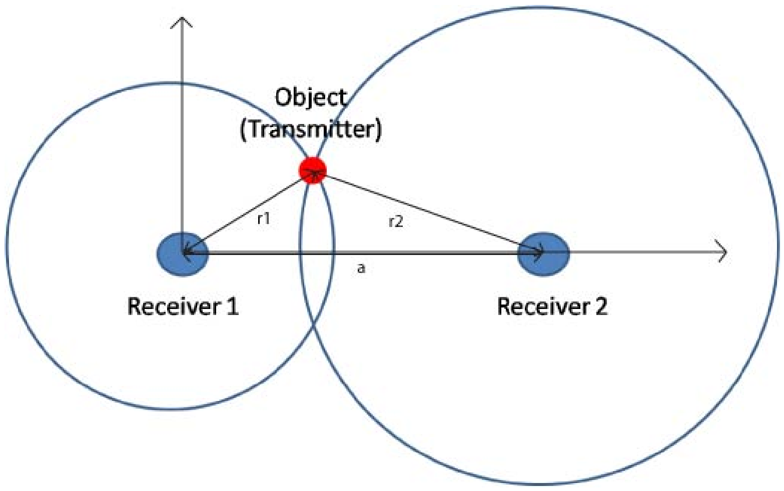

Figure 3: Illustration of trilateration and corresponding equations.

$$\begin{cases} {r_1}^2 = x^2 + y^2 \\ {r_2}^2 = (x-a)^2 + y^2 \end{cases}$$

$$\begin{cases} y = \sqrt{{r_1}^2 - x^2} \\ {r_2}^2 = (x-a)^2 + {r_1}^2 - x^2 \end{cases}$$

$${r_2}^2 = -2ax + a^2 + {r_1}^2$$

$$x = \frac{a}{2} + \frac{{r_1}^2}{2a} - \frac{{r_2}^2}{2a}$$

The different parts

Transmitter

The main components of the transmitter are 1 Atmel AVR Atmega88, 2 ultrasonic transmitters and 2 Maxim max202 chips.

The AVR is configured to send PWM signals with a frequency of 40 kHz. When a button connected to the AVR is pushed it interrupts the processor and it sends out the PWM signal five times, with pauses in between. The output PWM is connected to the max202 chip to make the signal to the transmitter stronger. The final design included two separate drivers with their own transmitter, the reason for this being that the transmitters have a very small angle in which they transmit. To be able to hit both receivers two transmitters are needed.

When the input to a max202 is low it outputs 10V and when high it outputs -10V. Each max202 has two drivers. By connecting the output from the first driver to the input of the second with help of a resistor and a diode as in Figure 4 a voltage difference of 20V is obtained between the two outputs.

The PWM signal is also connected to one of the interrupt pins on the FPGA. The Zener diode protects the FPGA by making sure that the voltage never reaches higher than 3.6 Volts.

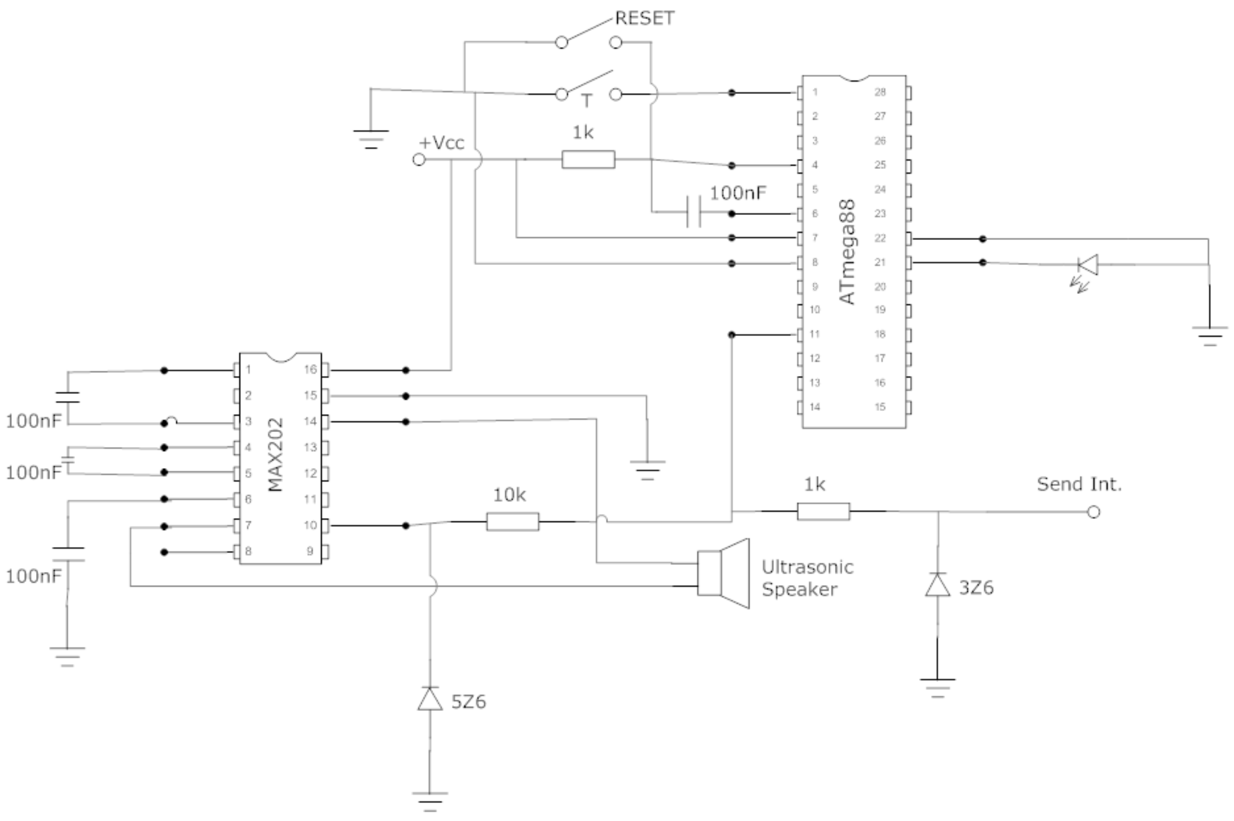

Figure 4: Schematic for the transmitter.

Receivers

The receivers consist of four main components; 1 ultrasonic receiver, 1 max202 chip, 1 active high pass filter and 1 diode bridge. The circuit is shown in Figure 5.

The max202 chip is only used to get a supply voltage for the op-amplifier.

The main task for this circuit is to optimize the receiving part and thereby allow detection of very weak incoming ultrasonic sound waves. The active high pass filter is connected immediately after the ultrasonic microphone and fulfils its purpose by eliminating noisy low frequency signals and amplifying the ones of higher frequency. The filtered and amplified signal is then rectified and fed into the FPGA as an interrupt. As well as in the transmitter a Zener diode protects the FPGA by limiting the input voltage to a maximum of 3.6 Volts.

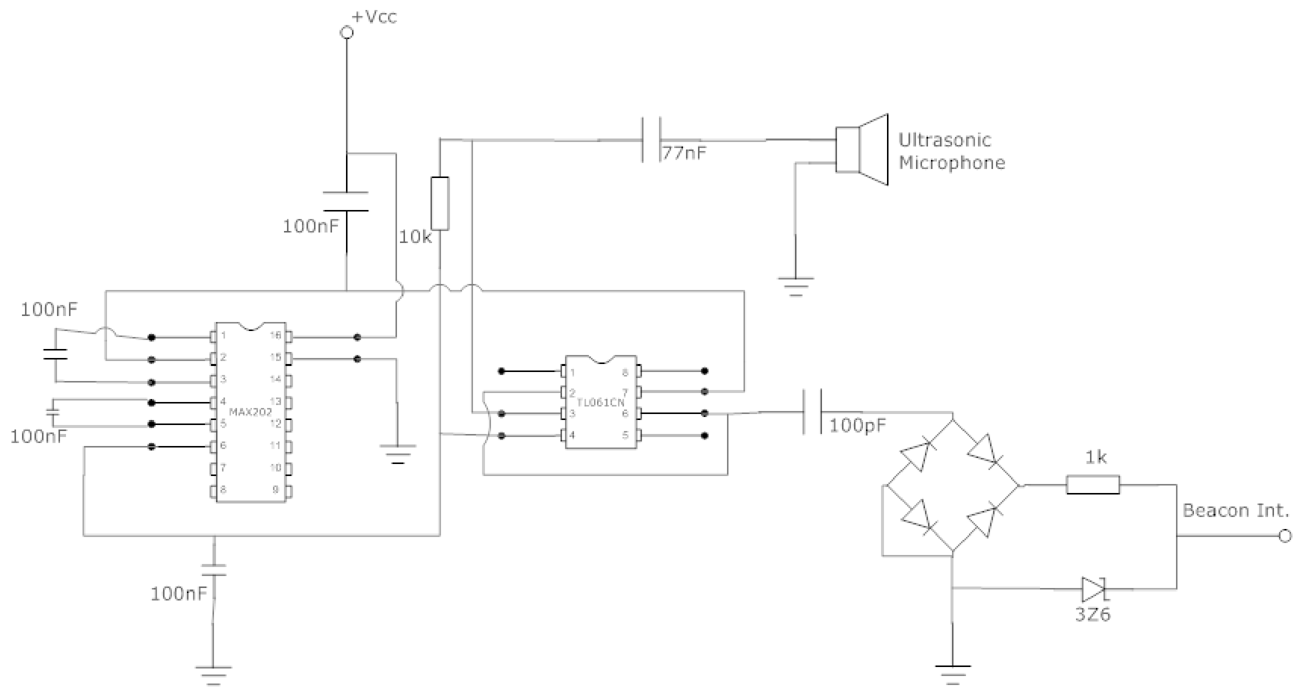

Figure 5: Schematic for the receivers

FPGA layout for positioning

The hardware is built around a 32-bit MicroBlaze processor. To be able to interface with users and external equipment two GPIOs and one UART module are included. Since the display controller wasn´t fully implemented a simple terminal communication is used for system outputs. Furthermore a timer module is added to give the ability to measure the time required for the ultrasound to arrive. All modules are connected to the PLB bus, and the GPIOs also have interrupts connected to the MicroBlaze. Since the MicroBlaze only supports a single hardware interrupt a separate interrupt handling module is added. This module simply "OR"s the different interrupts into a single connection and gives the possibility to check the interrupt origin via the PLB bus. As the original intention was to have the sender totally wireless from the positioning system a SPI module was added to enable communication with a radio module. However this module remained unused because of the lack of time to implement this feature.

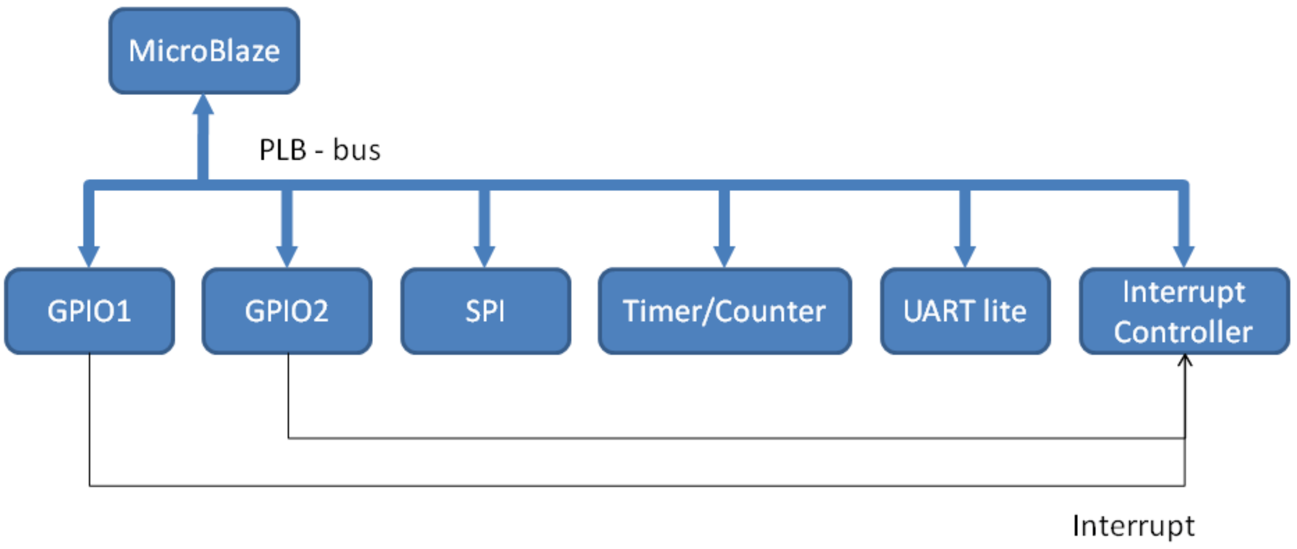

Figure 6: FPGA hardware block scheme.

MicroBlaze software

The software is mainly based around a single thread that simply waits for measurements to arrive, and then calculates and prints the coordinates through the UART interface. The measurements are administrated through a few interrupt driven functions. When the sender sends its ultrasonic pulse it at the same time sends a single pulse through a wire connected to one of the GPIOs. This resets and starts the timer and when the interrupt signal from the receivers arrives the value of the timer is written into a variable monitored by the main thread.

The software also contains a calibration routine that is manoeuvred via two buttons connected to one of the GPIOs. This simply uses measured values to estimate the speed of sound at the current location and circumstances.

Display (and corresponding FPGA layout)

To present the resulting position a graphical display were chosen as the display media. The chosen display has a resolution of 320x240 pixels with 3bits of colour per pixel. To obtain a steady picture, the screen has to be redrawn 70 times every second. To achieve this a VHDL module were developed to read pixel data from a BRAM and generate the drive signal for the display.

To interface the display 12 signals are needed, 8 bits of colour information, one clock input, one load input and one frame input.

Because there is only 3bit of data for each pixel and the data interface is 8bit wide data is sent for 8 pixels for every third byte that is clocked in. This requires 40 bytes of data to be shifted in for every line on the display. And with 240 lines on the display 9600 bytes of data is needed to represent the whole screen.

To draw the screen once 40 bytes are clocked in with the FRM high, this tells the display that it's the first line that should be drawn. Then the Load pin is clocked and 40 bytes for the second line is clocked in. This is repeated until all 240 lines have been filled with data and the procedure is restarted for each frame.

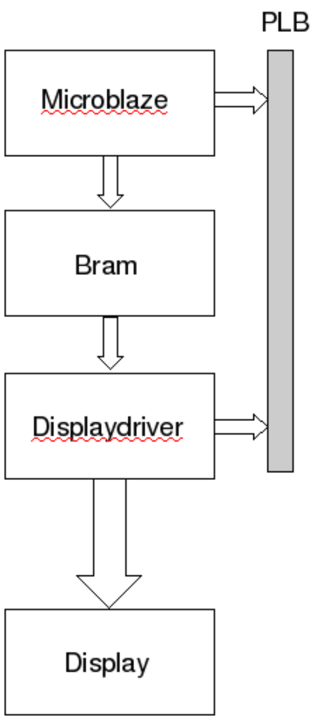

Figure 7: Hardware layout for display driver.

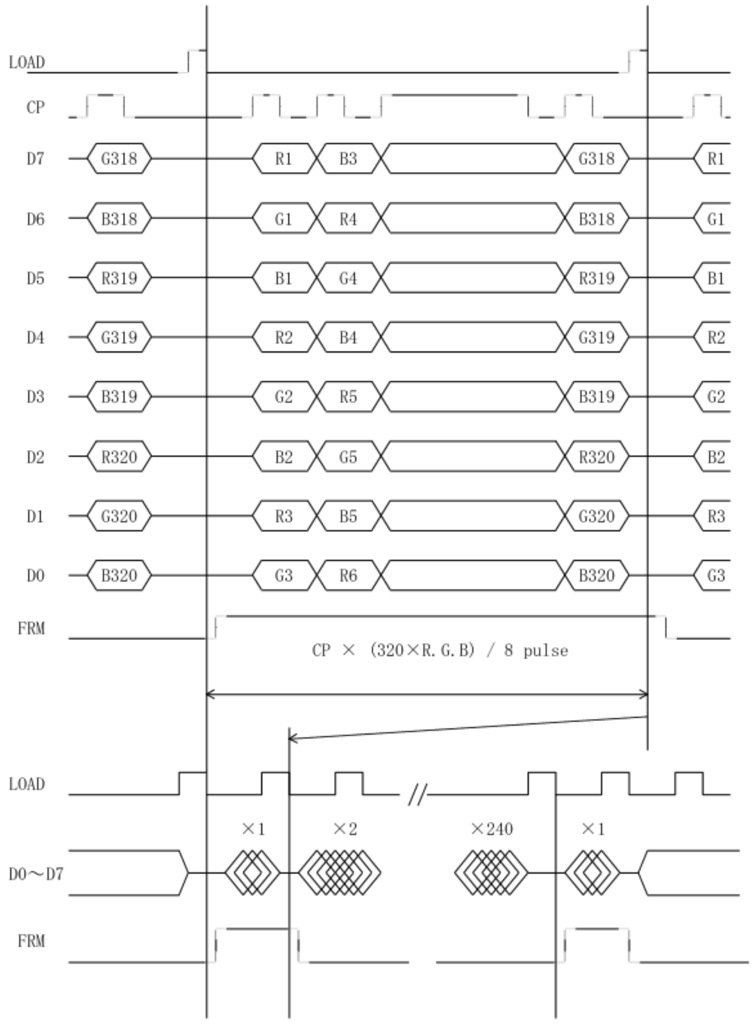

Figure 8: Timing chart for display

Hardware layout

The hardware is divided in four different blocks; Comb_proc, Adr_count, Sync_proc and Sync_out.

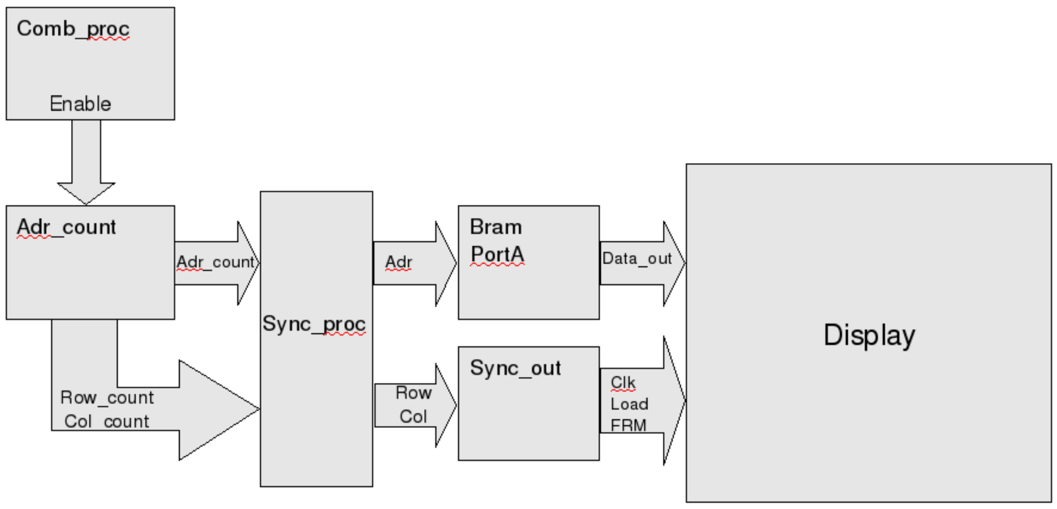

Figure 9: Display driver block layout.

Comb_proc generates a 3 MHz enable signal, this gives the display a refresh rate of about 70 Hz. Adr_counter uses the Enable signal from Comb_proc to generate signals for which row and line that is currently being drawn; it also generates an address signal for the BRAM. The Sync_out process generates the actual signals to the display for clocking in data, changing lines and so on.

The biggest problem in the design is the integration with a MicroBlaze processor, there isn't much documentation available on how to implement the BRAM interface to allow both the MicroBlaze and a custom VHDL module to access it. Once the BRAM interface is included it's very hard to know if the problem is in the EDK connection or in the VHDL code.

Problems encountered and how they were solved

Many of the problems encountered is due to lack of previous experience in designing such designs, a solution would be better documentation or more time for us to familiarize us with the hardware and software.

At first a speed of sound of 340m/s was used in the positioning algorithm as we thought this would give fairly good results even though it varies for different environments. However, the results weren't that good and surprisingly there seemed to be a non-linearity in the measurements. We made tests to determine how the system really behaved and the measurements are plotted in Figure 11. As seen there are non-linearities when the distance gets larger for reasons unknown. To solve this problem we firstly fitted 2nd order polynomials to be used in the calculation. But as can be seen in Figure 12 this would give very strange results for small distances so we decided to stick to the linear calculation. But to get a better value for the speed of sound we implemented a calibration algorithm.

It was quite difficult to get good values from the filters. At first we tried to run the op amplifiers with 0 and +5V. It worked, but it wasn't until the swing was increased to 20V that op-amplifiers performed satisfying.

Lessons Learned

It is hard to estimate time when developing with tools that one is not that familiar with and in areas where one hasn't worked before.

The need for concurrent development and that it's not always easy to achieve when there are certain gates in the project plan.

What seems simple in theory might be complex in reality.

Conclusions

The project shows that the principle of positioning with ultrasonic sound waves is possible. Since the ultrasonic transmitters and receivers used have such a narrow angle in which they transmit and receive the system needs quite much help from the user with aiming. Preferably transmitters with wider angle should be used.

Since the system was proved not to be linear at all distances (of reasons unknown) the distance measurements were good in some areas but got an error when the distance got too large. But when several measurements are done on the same place, coordinates calculated are very stable which shows that the interrupt functions works well and that they're not interfered with other functions using the processor.

Contributions

FPGA hardware- Erik, Joakim A
FPGA display software- Joakim A
Software for positioning (FPGA and AVR)-Erik, Joakim B, Simon
Design of transmitters and receivers - Erik, Joakim B, Simon

User Manual

1. Build the transmitter and receivers according to the circuit diagrams in Figure 4 and Figure 5.
2. Connect beacon interrupts to m16 and m14 on the FPGA and send interrupt to k13.
3. Open the xps project file and open coordinateCalculation.c. Change the variable *a* to the actual distance between the receivers.
4. Program the FPGA by using the xps project file.
5. Program the Atmega 88 with code provided.
6. Make a calibration, the output can be observed in a terminal connected to the UART.
    a. Place the the two receivers together and put the transmitter at a distance of 500mm from the receivers.
    b. Make calculation of the coordinates by pushing the transmit button on the AVR till the coordinates calculated are stable.
    c. Push BTN1 on the FPGA board. Then move the transmitter to a distance of 200mm from the receivers.
    d. Make new calculations of the coordinates till values are stable then push BTN2 on the board.
    e. Now the system is calibrated for the speed of sound on the place where the system is actually placed.
7. Set up the receivers and transmitter according to Figure 10.
8. Push the interrupt button on the transmitter to make a position calculation. Since the positioning algorithm calculates an average over the 8 last measurements it takes some measurements till the position is stable.

Appendix

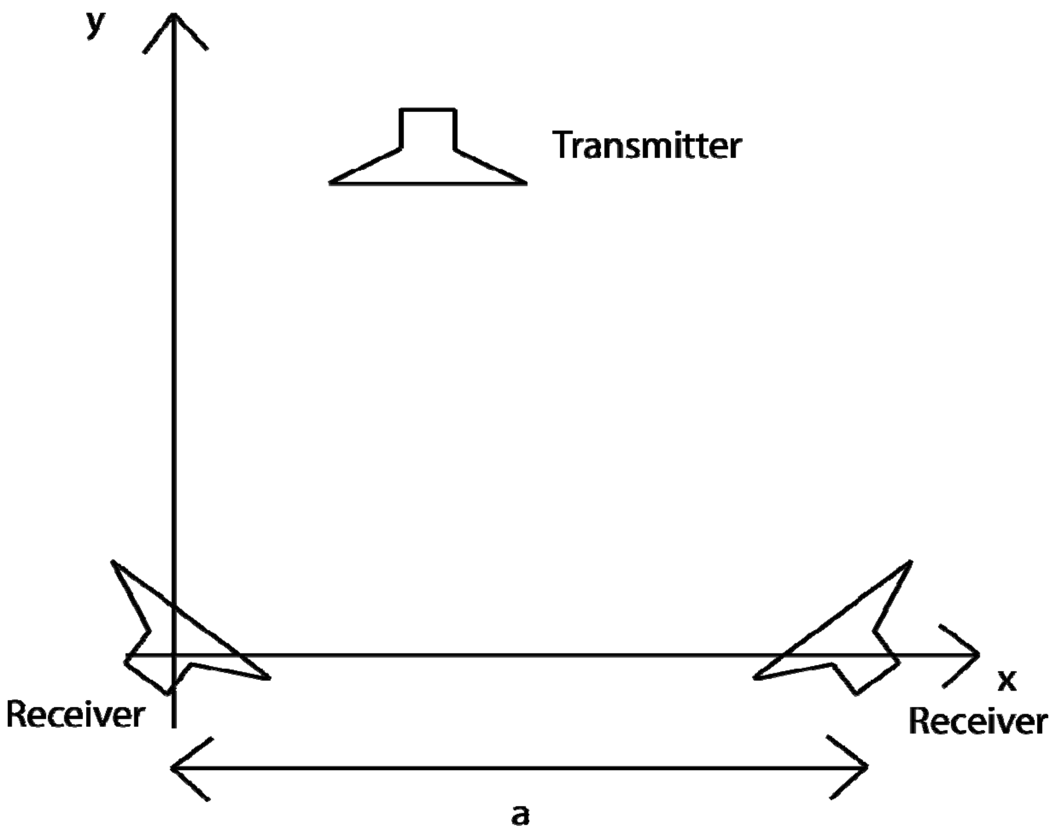

Figure 10: System setup

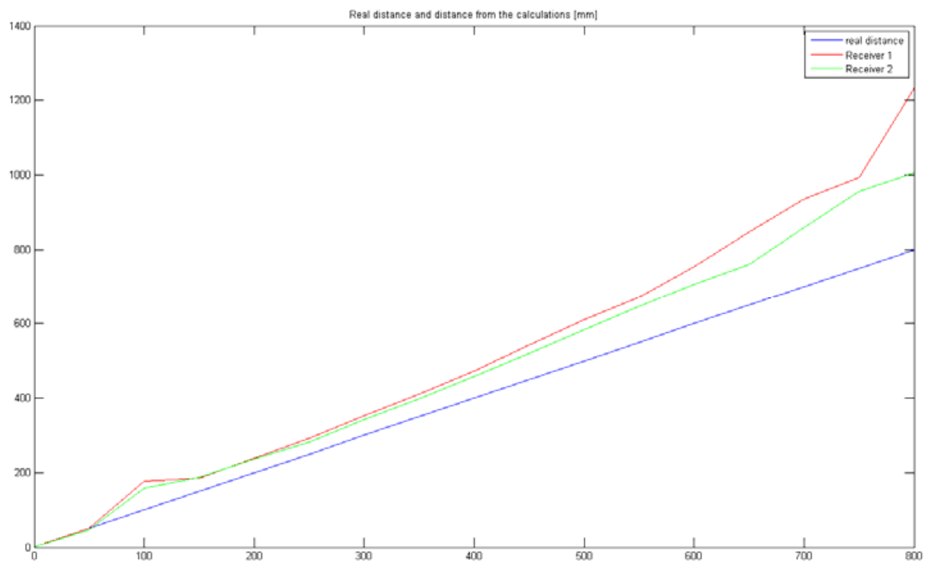

Figure 11

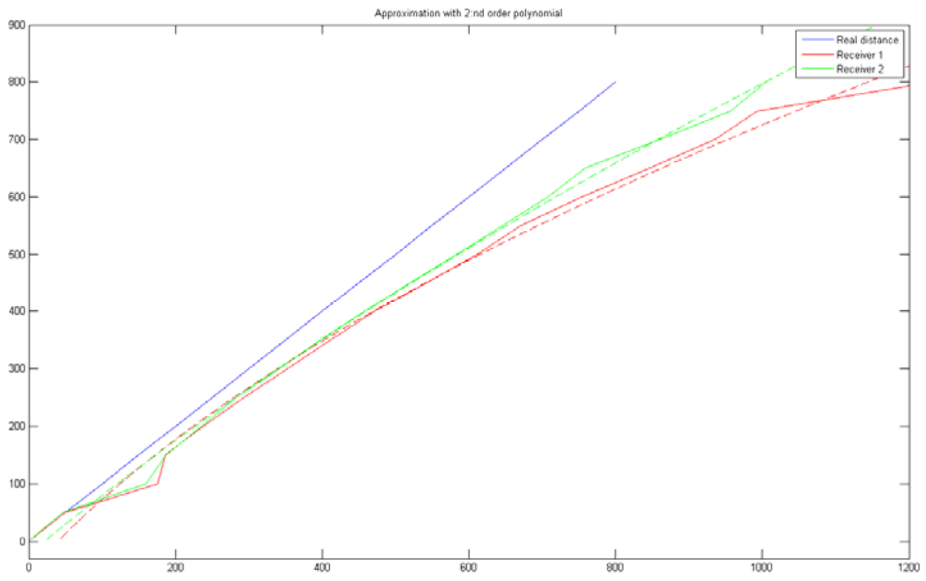

Figure 12

Design Summary Report

Number of External IOBs	25 out of 250	10%
Number of External Input IOBs	13	
Number of External Input IBUFs	13	
Number of LOCed External Input IBUFs	10 out of 13	76%
Number of External Output IOBs	4	
Number of LOCed External Output IOBs	1 out of 4	25%
Number of External Bidir IOBs	8	
Number of LOCed External Bidir IOBs	6 out of 8	75%
Number of BUFGMUXs	1 out of 24	4%
Number of DCMs	1 out of 8	12%
Number of MULT18X18SIOs	3 out of 28	10%
Number of RAMB16s	16 out of 28	57%
Number of Slices	2081 out of 8672	23%
Number of SLICEMs	201 out of 4336	4%

Logic Utilization:

Number of Slice Flip Flops:	1,877 out of 17,344	10%
Number of 4 input LUTs:	2,656 out of 17,344	15%

The compiled c-code is 30 792 B.